What are we PSIgning up for? Analyzing Applications of Two-Party Private Set Intersection

Vivek Arte[1]

07 December 2021

Abstract

A private set intersection (PSI) protocol allows two (or more) parties, each with sets from a common domain, to learn the intersection of their sets, while being "assured" that no information about their sets beyond the intersection has been revealed to the other parties. The level of this "assurance" provided depends on the model that is used while proving security of the protocol. We review some of the common models of security for two-party protocols used in the literature, and relate them to each other via implications and separations.

PSI protocols have found use in a wide variety of applications in different walks of life, from measuring the effectiveness of online advertisements and checking whether passwords have been compromised, to genomics and public health problems such as contact tracing. In this work, we look at a number of applications using two-party PSI protocols or common variants, and study the appropriateness of the applied security model to the problem at hand.

A common trend in existing work is the use of the weaker semi-honest model of security. While this allows for more efficient constructions, we point out, both via an explicit example and via a generic construction, how we can break security via minor deviations from the protocol specification when a protocol is secure only against semi-honest adversaries.

[1] Department of Computer Science & Engineering, University of California San Diego, USA. Email: `varte@eng.ucsd.edu`. URL: `cseweb.ucsd.edu/~varte/`.

Contents

1 Introduction **3**

2 Notions of Security **5**
2.1 The Simulation Paradigm 6
2.2 Semi-honest security 6
2.3 Security against malicious adversaries 7
2.4 Security against covert adversaries 8
2.5 The augmented semi-honest model 10
2.6 Relations between these models of security 10
2.7 Discussion 12

3 Private Set Intersection and Variants **12**

4 Applications **13**
4.1 Ad-conversion systems 14
4.2 Private contact discovery 14
4.3 Checking password compromise, and sharing security incident information 15
4.4 Genomics 15
4.5 Contact Tracing 16
4.6 Proximity Testing and Ride Sharing 17
4.7 Relationship path discovery in social networks 18

5 Security considerations **18**
5.1 People 19
5.2 Power 20
5.3 Provision of incorrect inputs 20

6 Do we need to achieve security in the malicious model? **21**
6.1 The semi-honest protocol from CM 21
6.2 A simple attack on the CM protocol by a malicious receiver 23
6.3 A generic example of an attack 23

7 Conclusions **23**

8 Acknowledgements **24**

References **24**

1 Introduction

The online world is built on the back of interactions between multiple parties. Online interactions usually involve the collection and processing of (possibly sensitive) user data, sometimes to make the user's life easier, or at times for the profit of the service provider. For some of these interactions, the parties might not know or trust who they are interacting with. Even when there is trust between parties, the plethora of data breaches that happen online mean that one can never be sure their data is safe once it is used online. This would make parties wary about sharing their personal or sensitive information in these protocols. A fundamental problem considered in cryptography deals with the feasibility of parties interacting with each other in such a way that the parties are able to compute some joint function of their private inputs, while minimizing the amount of information that is revealed during this computation. This is the Secure Multiparty Computation problem.

Secure Multiparty Computation (MPC). MPC has been an area of theoretical study in cryptography for many years, starting from works by Yao [Yao86] for the two-party setting, and by Goldreich, Micali and Wigderson (GMW) [GMW87] for the multiparty setting. In the MPC setting, there are n parties $P_1, P_2, \dots P_n$. The party P_i has an input $x_i \in \{0,1\}^*$ (for every $i \in [n]$). There are n (not necessarily distinct) functions $f_1, f_2, \dots f_n$, where for $i \in [n]$, the party P_i is interested in learning the output of function $f_i : \{0,1\}^* \times \dots \times \{0,1\}^* \to \{0,1\}^*$ on the inputs of all the parties, $f_i(x_1, x_2, \dots, x_n)$. Every party would like to do this while revealing as little as possible about their input. Protocols for MPC allow multiple parties to compute such a joint function of their inputs without resorting to the presence of a trusted party, and while having confidence that their private information is not being unduly leaked. This is an essential requirement of today's internet, given its inherently decentralized nature. MPC protocols find uses in a wide variety of online protocols, including those for auctions [BCD+08], biomedical computations [BBD+11, CWB18], data analysis [LP08, BKL+14, BKK+16], and in finance [AKL12, BTW12].

Security models. The security-critical nature of these applications calls for clear knowledge of exactly how much "confidence" participants should have in the security of the protocol. This is done via the use of different security models that provide differing levels of guarantees to the parties. These models assume that some fraction of the parties are "corrupted" by an adversary that is trying to subvert the protocol execution. Security is achieved if the protocol is able to withstand an attack by these corrupted parties. The high-level goals aimed for by these security models are as follows [HL10]:

- *Privacy.* No party P_i should learn anything more than the value of the function f_i on all the parties' inputs. In other words, the only information a party P_i should learn about any other party's inputs is information that can be derived from the value of function f_i on the parties' inputs.
- *Correctness.* Each party P_i should be guaranteed to receive the value of the function it is interested in computing on the parties' inputs, $f_i(x_1, \dots x_n)$, from the protocol.
- *Independence of Inputs.* The inputs of the corrupted parties should be picked independently of the inputs of the honest parties.
- *Guaranteed Output Delivery.* The corrupt parties should not be able to prevent the other parties from receiving their outputs.
- *Fairness.* The definition of fairness asks that the corrupted parties receive their outputs iff the honest parties receive their outputs.

Note that in the two-party setting, it is impossible to achieve the goals of guaranteed output delivery and fairness for general computations.

It is non-trivial to formally define all these goals, and it is also not necessary that the above list is an exhaustive one for achieving protocol security. The standard approach to defining security for such protocols is via the *simulation paradigm*. This approach defines an "ideal world" in which there exists an external trusted party that helps the parties perform the computation. In the "real world," the parties run some protocol between themselves (without any trusted party). Security is then formulated by showing that for any adversary that successfully attacks the protocol in the real world, there exists an adversary that can carry out the same attack in the ideal world.

Security models are classified based on the power given to the adversary in the model. The simplest security model is the **semi-honest model** (also called the honest-but-curious model). This model restricts parties corrupted by the adversary to following the specifications of the protocol. However, the adversary can look at the messages sent in the protocol and try to learn information about the other party's inputs. A stronger model (that is, one that gives the adversary more power) is the **malicious model**, where the adversary can instruct corrupted parties to deviate from the protocol. The **covert model** lies in between the above two models in strength – it allows an adversary to cause arbitrary deviations from the protocol (as in the malicious model), with the caveat that if the adversary does so, such a deviation will be detected by the honest parties with some given probability.

EFFICIENCY. There is a trade-off between security and efficiency that must be balanced while building MPC protocols. Achieving security in weaker models of security can generally be done more efficiently, and achieving stronger forms of security requires more computation and communication overhead between the parties.

Multiparty computation protocols usually are constructed for generic computation, meaning they can then be used for a variety of problems simply by compiling the problem into the format required by the protocol. However, achieving this generality also has efficiency drawbacks. There has therefore been a thrust towards developing specialized protocols for specific applications. The increased specificity of these protocols allows for efficiency improvements. One important specialization that has seen significant research interest is that of Private Set Intersection (PSI).

PRIVATE SET INTERSECTION. Consider a set of parties, each of whom have an input set from a common domain. The parties are interested in learning the intersection of all their sets, while each keeping their inputs as private as possible. This problem was first considered by Freedman, Nissim and Pinkas [FNP04] for the two-party setting. Since then, there has been a large amount of work in developing efficient protocols for the PSI problem, especially in the two-party setting [DMRY09, DKT10, KLC12, HEK12, DCW13, PSZ14, PSSZ15, KKRT16, RR17a, RR17b, CLR17, RA18, PRTY19, PRTY20, CM20, KK20, RT21], but also in the multi-party setting [KS05, SS08, NAA+09, CJS12, HV17, KMP+17, IOP18, GN19, BMRR21, NTY21]. The literature has also branched out into a number of related problems that compute different functions of the set intersection [DGT12, IKN+17, IKN+19, MPR+20], or compute the intersection on certified sets [CZ09, DKT10, DT10]. In this work, we restrict our focus to two-party PSI protocols, some common variants in the two-party setting, and their applications.

APPLICATIONS. There are a number of applications that make use of PSI protocols in order to achieve their aims in a privacy preserving manner. The straightforward applications are those in which both the participants have sets of information, and are interested in checking for the overlap of their sets. These include in law enforcement, such as terror watch lists and commercial flight rosters, and for checking whether passwords belong to a list of passwords known to be compromised. We review some of these applications in more detail in this work, and study how they make use of PSI protocols.

APPLICATION SECURITY. The level of security we need to aim for depends heavily on the context of the application being tackled using a PSI protocol. Questions that need to be considered include the sensitivity of the data at risk of being revealed, and the amount of trust that we should have in the participants.

A significant fraction of the PSI protocols constructed in the literature are shown to be secure only against semi-honest adversaries. A common reason for doing so is to maintain a reasonable level of efficiency, without which the time taken might be too long to be practical or marketable. We however discuss how restricting to semi-honest adversaries may not be sufficient for most applications. We look at possible alternatives such as execution in trusted environments like Intel SGX, and code audits, and what assurances they could provide. We also consider the amount of benefit that using the stronger security models would give these applications.

We also note that no matter which model of security we use, a PSI protocol is unable to defend against someone giving different inputs to the protocol. We look into the variants of PSI that have been developed to help with this shortcoming, and analyze their benefits.

ORGANIZATION. The rest of this report is organized as follows. We begin with defining the different notions of security used in secure multi-party computation, and follow it up with a discussion on their positives and negatives in Section 2. We then narrow focus to PSI protocols and some of their common variants in Section 3. In Section 4, we discuss some popular applications that make use of PSI protocols (or PSI variants) as building blocks. Due to the general tendency of protocols in the literature to focus on semi-honest security, we then explicitly show a simple but effective attack on a semi-honest protocol ofChase and Miao [CM20] when a party deviates from the protocol in Section 6. We conclude with some directions for future research in Section 7.

2 Notions of Security

The existing literature on Private Set Intersection protocols uses a variety of different definitions of security. All definitions, however, follow the same general framework known as the simulation (or REAL/IDEAL) paradigm [Can00, Bea91, MR92]. In this section, we first start with some notation and preliminaries, and then describe the simulation paradigm in the two-party setting. Then, we proceed to formally define the different notions of security considered in the literature. The definitions that follow are largely adapted from Hazay and Lindell (HL) [HL10].

NOTATION AND PRELIMINARIES. If $\boldsymbol{w}$ is a vector then $|\boldsymbol{w}|$ is its length (the number of its coordinates) and $\boldsymbol{w}[i]$ is its i-th coordinate. Strings are identified with vectors over $\{0,1\}$, so that $|Z|$ denotes the length of a string Z and $Z[i]$ denotes its i-th bit. By ε we denote the empty string or vector. By $x \| y$ we denote the concatenation of strings x, y. If x, y are equal-length strings then $x \oplus y$ denotes their bitwise XOR, and $x \wedge y$ denotes their bitwise AND. If S is a finite set, then $|S|$ denotes it size. If S is a set, we use $P(S)$ to denote the power set of S

If X is a finite set, we let $x \leftarrow\$ X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. If A is an algorithm, we let $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1,\ldots;\omega)$ denote running A on inputs $x_1,\ldots$ and coins ω, with oracle access to $\mathrm{O}_1,\ldots$, and assigning the output to y. By $y \leftarrow\$ A^{\mathrm{O}_1,\cdots}(x_1,\ldots)$ we denote picking ω at random and letting $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1,\ldots;\omega)$. We let $[A^{\mathrm{O}_1,\cdots}(x_1,\ldots)]$ denote the set of all possible outputs of A when run on inputs $x_1,\ldots$ and with oracle access to $\mathrm{O}_1,\ldots$. Running time is worst case, which for an algorithm with access to oracles means across all possible replies from the oracles. "PT" stands for "polynomial time" while "PPT" stands for "probabilistic polynomial time." An adversary is an algorithm. Unless specifically stated otherwise, we assume adversaries to be PPT. We use $\perp$ (bot) as a special symbol to denote rejection, and it is assumed to not be in $\{0,1\}^*$. By 1^λ we denote the unary representation of the integer security parameter $\lambda \in \mathbb{N}$.

A function $\nu\colon \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p\colon \mathbb{N} \to \mathbb{R}$ there is a $\lambda_p \in \mathbb{N}$ such that $\nu(\lambda) \leq 1/p(\lambda)$ for all $\lambda \geq \lambda_p$. A *probability ensemble* $X = \{X(a,\lambda)\}_{a\in\{0,1\}^*;\lambda\in\mathbb{N}}$ is an infinite sequence of random variables sequenced by the parties inputs (denoted by a) and the security parameter λ. Two probability ensembles $X = \{X(a,\lambda)\}_{a\in\{0,1\}^*;\lambda\in\mathbb{N}}$ and $Y = \{Y(a,\lambda)\}_{a\in\{0,1\}^*;\lambda\in\mathbb{N}}$ are said to be *computationally indistinguishable*, denoted by $X \overset{c}{\equiv} Y$, if for every non-uniform PT algorithm D there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0,1\}^*$ and every $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[D(1^\lambda, a, X(a,\lambda)) = 1\right] - \Pr\left[D(1^\lambda, a, Y(a,\lambda)) = 1\right]\right| \leq \mu(\lambda)$$

<u>DIGITAL SIGNATURES.</u> A (digital) signature scheme DS specifies PT algorithms for key-generation, signing and verifying, as follows. Via $(sk, vk) \leftarrow_{\$} \mathsf{DS.K}(1^\lambda)$, the signer generates a secret signing key sk and public verification key vk. Via $\sigma \leftarrow_{\$} \mathsf{DS.S}(1^\lambda, sk, vk, m)$, the signer generates a signature of a message $m \in \{0,1\}^*$. Via $\mathrm{vf} \leftarrow \mathsf{DS.V}(1^\lambda, vk, m, \sigma)$, the verifier deterministically generates a boolean decision as to the validity of σ. Correctness requires that $\mathsf{DS.V}(1^\lambda, vk, m, \sigma) = \mathsf{true}$ for all $\lambda \in \mathbb{N}$, all $\sigma \in [\mathsf{DS.S}(1^\lambda, sk, vk, m)]$, all $(sk, vk) \in [\mathsf{DS.K}(1^\lambda)]$ and all $m \in \{0,1\}^*$. The security metric for digital signature schemes is unforgeability (UF) [GMR88]. We refer the reader to Section 6 of Arte and Bellare [AB20] for more details on the definition of unforgeability with this notation.

2.1 The Simulation Paradigm

The basic idea underlying the simulation paradigm is the comparison of the behavior of a "real world" protocol with an "ideal world" functionality (or *ideal functionality*). The ideal functionality is constructed in such a way as to be secure by definition – usually via the presence of a universally trusted party that performs the computation. We assume adversaries are allowed to "corrupt" one of the two parties participating in the protocol. For the purpose of this work, we assume the corruption is *static* – that is, that the corrupted party is fixed at the start of the protocol and does not change throughout the protocol. The view of an adversary consists of the inputs, the internal randomness of the parties that have been corrupted by the adversary, and the messages the corrupted parties receive during the protocol. Proving security in the simulation paradigm then involves constructing a simulator in the ideal world that can generate a view for the adversary that is indistinguishable from the view of the adversary in the real world.

<u>TWO-PARTY FUNCTIONALITIES.</u> A two-party protocol is constructed to solve a specific problem, which is called a *functionality*. A functionality maps pairs of inputs to pairs of outputs (one for each party), and we denote it by $\mathcal{F} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $\mathcal{F} = (f_1, f_2)$. That is, when the first party has an input x and the second party has an input y, the functionality calculates $f_1(x,y)$ for the first party and $f_2(x,y)$ for the second party. This is also denoted by $(x,y) \mapsto (f_1(x,y), f_2(x,y))$. Functionalities can be probabilistic, but for our purposes – that is, private set intersection – we only need to deal with deterministic functionalities.

It is generally unavoidable to reveal some knowledge of the input lengths involved in the computation. This is accounted for by implicitly assuming that the functionality is such that the parties know the length of all inputs.

2.2 Semi-honest security

In the semi-honest model, it is assumed that all parties (whether corrupted by the adversary or honest) exactly follow the protocol specification, and do not misrepresent their inputs in any way. Security is achieved when the adversary is unable to obtain any additional information from participating the protocol as compared to participating in the ideal world.

Consider a two-party PPT functionality $\mathcal{F} = (f_1, f_2)$, and let Π be a two-party protocol for computing $\mathcal{F}$. We start with the following notation:

Views. The *view* of the i-th party ($i \in \{1, 2\}$) during an execution of the protocol Π on inputs (x, y) with security parameter λ is denoted by $\mathsf{view}_i^{\Pi}(x, y, \lambda)$. A party's view includes the input of the party, the contents of its internal random tape, and the transcript of messages the party received during the protocol.

Outputs. The *output* of the i-th party during the execution of the protocol Π on on inputs (x, y) with security parameter λ is denoted by $\mathsf{output}_i^{\Pi}(x, y, \lambda)$. A party's output can be computed from its view of the execution. The joint output of the execution is denoted by $\mathsf{output}^{\Pi}(x, y, \lambda) = \big(\mathsf{output}_1^{\Pi}(x, y, \lambda), \mathsf{output}_2^{\Pi}(x, y, \lambda)\big)$

Definition 2.1 ([HL10], Definition 2.2.1) *Let $\mathcal{F} = (f_1, f_2)$ be a two-party functionality, and let Π be a two-party protocol. We say that Π securely computes $\mathcal{F}$ in the presence of static semi-honest adversaries if there exist PPT algorithms $\mathcal{S}_1$ and $\mathcal{S}_2$ such that*

$$\{\mathcal{S}_1(1^{\lambda}, x, f_1(x, y)), \mathcal{F}(x, y)\}_{x,y,\lambda} \overset{\mathsf{c}}{\equiv} \{\mathsf{view}_1^{\Pi}(x, y, \lambda), \mathsf{output}^{\Pi}(x, y, \lambda)\}_{x,y,\lambda} \,,$$

$$\{\mathcal{S}_2(1^{\lambda}, y, f_2(x, y)), \mathcal{F}(x, y)\}_{x,y,\lambda} \overset{\mathsf{c}}{\equiv} \{\mathsf{view}_2^{\Pi}(x, y, \lambda), \mathsf{output}^{\Pi}(x, y, \lambda)\}_{x,y,\lambda} \,,$$

where $x, y \in \{0, 1\}^*$ and $\lambda \in \mathbb{N}$.

Note that the simulators for the parties simulate the view of the party after having access solely to the party's input and the output of the functionality on all the parties' inputs. The restriction of adversaries to conform to the protocol specification makes this a rather weak model of security. In general, the semi-honest model is useful for ensuring that there is no inadvertent leakage of information from the messages sent in the protocol, or in cases where there are external reasons to trust that parties will follow the specifications of the protocol perfectly.

2.3 Security against malicious adversaries

The malicious model is a stronger model of security, wherein the adversaries are allowed to arbitrarily deviate from the specification of the protocol. Security is achieved when even this added strength provided to the adversary does not allow the adversary to obtain any additional information from participating in the protocol. Defining security in this scenario is significantly more complicated that in the previous, semi-honest case. For example, it is not possible to prove security just by constructing a simulator that can generate the view of a corrupted party. This is because a malicious party can choose to use a different input than what is locally present.

As we have previously mentioned, achieving guaranteed output delivery and fairness is in general not possible in the two-party setting. This limitation needs to be incorporated into the ideal setting. We do this by allowing the adversary to obtain the corrupted party's output without the honest party necessarily receiving its output. This is sometimes also known as *security with selective abort*. We assume that the adversary corrupts exactly one of the two parties in the protocol.

Ideal model execution. Let the two participating parties be P_1 and P_2. Let the original input of party P_1 be $x \in \{0, 1\}^*$ and that of party P_2 be $y \in \{0, 1\}^*$. We assume an adversary A has an auxiliary input $z \in \{0, 1\}^*$. We further assume the adversary corrupts a single party, and we denote the corrupted party by P_i where $i \in \{1, 2\}$, and the honest party by P_j where $j \in \{1, 2\}, j \neq i$. An ideal execution for a two-party functionality $\mathcal{F} = (f_1, f_2)$ proceeds as follows:

- *Sending inputs to trusted party.* The honest party P_j sends its input to the trusted party. The corrupted party P_i is controlled by the adversary A, and the adversary can decide to either abort, or to send the original input, or to send a different input of the same length. The adversary decides this based on the input of the corrupted party and the auxiliary information. Let (x', y') denote the inputs sent to the trusted party – note that at most one of x' and y' could differ from the original inputs x and y respectively.
- *The early abort option.* If the trusted party receives a command to abort, abort_i, from the corrupted party, then it sends abort_i to the other party as well and the execution terminates.
- *Adversary receives output.* The trusted party uses the inputs (x', y') it receives to compute $f_1(x', y')$ and $f_2(x', y')$. The trusted party then sends the corrupted party its output. That is, it sends $f_i(x', y')$ to P_i.
- *Adversary decides whether or not to halt.* The adversary then instructs the trusted party to continue the execution (via $\mathsf{continue}_i$) or to abort (via abort_i). If the adversary's instruction is to continue, then the trusted party sends the honest party its output – that is, it sends $f_j(x', y')$ to P_j. On the other hand, if the adversary's instruction is to abort, it sends abort_i to the honest party P_j.
- *Outputs.* The honest party always outputs the output value it receives from the trusted party. On the other hand, the adversary computes the output of the corrupted party as some polynomial time computable function of the original input of the corrupted party, the auxiliary input z, and the value received from the trusted party, $f_i(x', y')$.

We use $\textsc{ideal}_{\mathcal{F},A(z),i}(x, y, \lambda)$ to denote the output pair of the honest party and the adversary A in the ideal execution described above.

<u>Real model execution.</u> This is the scenario when there is no trusted party, and the two parties run a two-party protocol Π in order to compute $\mathcal{F}$. The real execution of protocol Π on inputs (x, y), with auxiliary input z to the adversary A (that corrupts a party P_i) and security parameter λ is denoted by $\textsc{real}_{\Pi,A(z),i}(x, y, \lambda)$.

<u>Defining security.</u> The standard definition of security in the malicious setting works by showing that adversaries in the ideal model are able to emulate executions of the protocol in the real model. More formally, we have the following definition.

Definition 2.2 ([HL10], Definition 2.3.1) *Let $\mathcal{F} = (f_1, f_2)$ be a two-party functionality, and let Π be a two-party protocol. We say that Π securely computes $\mathcal{F}$ with abort in the presence of malicious adversaries if for every non-uniform PPT adversary A for the real model, there exists a non-uniform PPT simulator $\mathcal{S}$ for the ideal model such that for every $i \in \{1, 2\}$, we have,*

$$\left\{\textsc{ideal}_{\mathcal{F},\mathcal{S}(z),i}(x, y, \lambda)\right\}_{x,y,z,\lambda} \overset{c}{\equiv} \left\{\textsc{real}_{\Pi,A(z),i}(x, y, \lambda)\right\}_{x,y,z,\lambda}$$

where $x, y, z \in \{0, 1\}^*$ and $\lambda \in \mathbb{N}$.

2.4 Security against covert adversaries

The covert model was first introduced by Aumann and Lindell [AL07]. In this model, the adversaries are allowed to arbitrarily deviate from the specification of the protocol (just as in the malicious model). The main point of difference between this model and the malicious model is that in this model, adversaries are willing to risk being caught with greater than negligible probabilities. The definition of security is again based on the $\textsc{real}/\textsc{ideal}$ simulation paradigm. The general idea is that if the adversary "cheats", then it will be caught by the honest party with some probability. Cheating is informally the ability to do something that is impossible to recreate in the ideal model.

IDEAL MODEL EXECUTION. There are some modifications made to the execution in the ideal model that we saw in the previous subsection. As before, we have two parties, P_1 with original input $x \in \{0,1\}^*$ and P_2 with original input $y \in \{0,1\}^*$. The adversary has an auxiliary input $z \in \{0,1\}^*$, and we assume it corrupts exactly one party P_i as before. As before, the honest party is denoted by P_j. An ideal execution for a two-party functionality $\mathcal{F} = (f_1, f_2)$ with parameter ϵ (where $0 \leq \epsilon \leq 1$) proceeds as follows:

- *Sending inputs to trusted party.* The honest party P_j sends its input to the trusted party. The corrupted party P_i is controlled by the adversary A, and the adversary can decide to either abort, or to send the original input, or to send a different input of the same length. The adversary decides this based on the input of the corrupted party and the auxiliary information. Let (x', y') denote the inputs sent to the trusted party – note that at most one of x' and y' could differ from the original inputs x and y respectively.
- *The early abort option.* If the trusted party receives a command to abort, abort_i from the corrupted party, then it sends abort_i to the other party as well and the execution terminates. The corrupted party can also send a message $\mathsf{corrupted}_i$ saying it is corrupted to the trusted party. In this case, the trusted party forwards $\mathsf{corrupted}_i$ to the honest party, and the execution terminates.
- *The cheat option.* The corrupted party can send a cheat message. cheat_i to the trusted party. If it does, the trusted party, with probability ϵ, sends $\mathsf{corrupted}_i$ to the adversary and the honest party. Otherwise (that is, with probability $1-\epsilon$) the trusted party sends $\mathsf{undetected}$ to the adversary along with the honest party's input. Then the adversary sends the trusted party a value τ of its choice for the honest party. The trusted party sends this value τ to the honest party for its output. The protocol execution ends at this point.
- *Adversary receives output.* The trusted party uses the inputs (x', y') it receives to compute $f_1(x', y')$ and $f_2(x', y')$. The trusted party then sends the corrupted party its output. That is, it sends $f_i(x', y')$ to P_i.
- *Adversary decides whether or not to halt.* The adversary then instructs the trusted party to continue the execution or to abort. If the adversary's instruction is to continue (via $\mathsf{continue}_i$), then the trusted party sends the honest party its output – that is, it sends $f_j(x', y')$ to P_j. On the other hand, if the adversary's instruction is to abort (via abort_i), it sends abort_i to the honest party P_j.
- *Outputs.* The honest party always outputs the output value it receives from the trusted party. On the other hand, the adversary computes the output of the corrupted party as some polynomial time computable function of the original input of the corrupted party, the auxiliary input z, and the value received from the trusted party, $f_i(x', y')$.

We use $\text{IDEALSC}^{\epsilon}_{\mathcal{F},A(z),i}(x, y, \lambda)$ to denote the output pair of the honest party and the adversary A in the ideal execution described above.

REAL MODEL EXECUTION. The execution in the real model and the definition of $\text{REAL}_{\Pi,A(z),i}(x, y, \lambda)$ is the same as in the previous subsection.

SECURITY DEFINITION. The formal definition of security against covert adversaries is as follows:

Definition 2.3 ([HL10], Definition 2.4.1) *Let $\mathcal{F} = (f_1, f_2)$ be a two-party functionality, let Π be a two-party protocol, and let $\epsilon : \mathbb{N} \to [0,1]$ be a function.. We say that Π securely computes $\mathcal{F}$ with abort in the presence of covert adversaries with ϵ-deterrent if for every non-uniform PPT adversary A for the real model, there exists a non-uniform PPT simulator $\mathcal{S}$ for the ideal model such that for every $i \in \{1,2\}$, we have,*

$$\left\{\text{IDEALSC}^{\epsilon}_{\mathcal{F},\mathcal{S}(z),i}(x, y, \lambda)\right\}_{x,y,z,\lambda} \overset{c}{\equiv} \left\{\text{REAL}_{\Pi,A(z),i}(x, y, \lambda)\right\}_{x,y,z,\lambda}$$

$$\begin{tikzcd}
& & \text{semi-honest} \arrow[dll, "\text{Proposition 2.10}"', "/" marking] & \\
\text{malicious} \arrow[urr, bend left, "\text{Proposition 2.7}", "/" marking] \arrow[r, "\text{Proposition 2.4}"'] & \text{covert} \arrow[l, bend left, "\text{Proposition 2.6}"] \arrow[rr, "\text{Proposition 2.5}"'] & & \text{augmented-semi-honest}
\end{tikzcd}$$

Figure 1: Relations among the security models defined in Section 2. A black arrow A $\rightarrow$ B is an implication: every scheme that is A-secure is also B-secure. A blue arrow A $\rightarrow$ B is a "conditional" implication: a scheme that is A-secure is also B-secure, iff a certain condition is satisfied. A red barred arrow A $\not\rightarrow$ B is a separation: there is an A-secure scheme that is not B-secure.

---

where $x, y, z \in \{0,1\}^*$ *and* $\lambda \in \mathbb{N}$.

2.5 The augmented semi-honest model

This is an extension of the semi-honest model introduced by Goldreich [Gol04], which requires adversaries to follow the specifications of the protocol, with the added power to change the input of the corrupted party before execution, and to abort at any point during the protocol. This model has not seen much adoption in practice, due to which we refer the reader to Section 7.4 of Goldreich's textbook [Gol04] for the specifics of the definition.

2.6 Relations between these models of security

At first glance, the expectation is that the semi-honest model is the weakest, with the covert and malicious models being progressively stronger. We highlight here some results that establish the relationship between these models of security, and we summarize these results in Figure 1. We refer the reader to the corresponding works of literature in the cases where we do not sketch the proof out.

Proposition 2.4 ([HL10], Proposition 2.4.3) *Let* Π *be a protocol that securely computes a functionality* $\mathcal{F}$ *with selective abort in the presence of malicious adversaries. Then* Π *securely computes* $\mathcal{F}$ *in the presence of covert adversaries with* ϵ*-deterrent, for every* $0 \leq \epsilon \leq 1$.

Proposition 2.5 ([HL10], Proposition 2.4.3) *Let* Π *be a protocol that securely computes a functionality* $\mathcal{F}$ *in the presence of covert adversaries with* ϵ*-deterrent, for* $\epsilon(\lambda) \geq 1/\operatorname{poly}(\lambda)$. *Then* Π *securely computes* $\mathcal{F}$ *in the presence of augmented semi-honest adversaries.*

Proposition 2.6 ([HL10], Proposition 2.4.5) *Let* μ *be a negligible function. Let* Π *be a protocol that securely computes a functionality* $\mathcal{F}$ *in the presence of covert adversaries with* ϵ*-deterrent, for* $\epsilon(\lambda) = 1 - \mu(\lambda)$. *Then* Π *securely computes* $\mathcal{F}$ *with selective abort in the presence of malicious adversaries.*

Proposition 2.7 *There exists a protocol* Π *such that* Π *securely computes* $\mathcal{F}$ *with selective abort against malicious adversaries, but* Π *does not securely compute* $\mathcal{F}$ *against semi-honest adversaries.*

Protocol $\Pi_\wedge$

The functionality to be computed is $(x, y) \mapsto (\perp, x \wedge y)$.
The inputs $x, y \in \{0, 1\}$.

1. P_1 sends P_2 its input bit x.
2. P_2 outputs the bit $x \wedge y$.

Figure 2: The protocol $\Pi_\wedge$ for computing the binary logical AND function, that is used in Proposition 2.7.

Proof of Proposition 2.7: We make use of Example 1 in Section 2.3.3 of HL. Consider the protocol $\Pi_\wedge$ for computing the binary logical AND function given in Figure 2. We state the following two lemmas, and refer the reader to HL for their proofs.

Lemma 2.8 ([HL10], Claim 2.3.3) *The protocol $\Pi_\wedge$ securely computes the binary logical AND functionality with selective abort in the presence of malicious adversaries.*

Lemma 2.9 ([HL10], Claim 2.3.4) *The protocol $\Pi_\wedge$ does not securely compute the binary logical AND functionality in the presence of semi-honest adversaries.*

These two lemmas complete the separation, as they show the existence of a protocol that securely computes a functionality with selective abort in the presence of malicious adversaries, but does not securely compute the functionality in the presence of semi-honest adversaries. ▌

Proposition 2.10 *Let Π be a protocol that securely computes functionality $\mathcal{F}$ against semi-honest adversaries. There exists a protocol Π' such that Π' securely computes $\mathcal{F}$ against semi-honest adversaries, but Π' does not securely compute $\mathcal{F}$ with selective abort against malicious adversaries.*

Proof of Proposition 2.10: We first provide the construction of the two party protocol Π' using the protocol Π. Let A and B be two parties with private inputs X and Y. Without loss of generality, we assume that party A starts the protocol by sending a message m_1^Π to B, followed by party B sending the message m_2^Π to A, and so on. That is, in the k-th round of the protocol, A sends m_{2k-1}^Π to B, and B sends m_{2k}^Π to A. We use Π to construct a two-party protocol Π' that is also semi-honest secure, but for which there is a simple attack against malicious security. The construction of Π' is as follows:

- Let m_1^Π and m_2^Π be the first messages sent by the parties A and B respectively in the specification of protocol Π. The specification of protocol Π' requires that party A first send $m_1^{\Pi'} = \left(m_1^\Pi, 0\right)$ to party B. That is, the message sent has one additional zero bit appended to the message from the original protocol Π.
- Protocol Π' specifies that party B parse $m_1^{\Pi'} \to (m_1, b)$. If $b = 0$, then party B computes m_2 as per the specification of protocol Π when input from party A is m_1, and then sends $m_2^{\Pi'} = m_2$ to party A. If $b = 1$, then party B sends $m_2^{\Pi'} = Y$ to party A. That is, party B sends its entire private input to party A when $b = 1$.
- The rest of the protocol specifications of Π' and Π are identical.

We now state two lemmas, from which we can complete the proof. The proofs of the lemmas are sketched below this proof.

Lemma 2.11 *The protocol Π' constructed above securely computes the functionality $\mathcal{F}$ in the presence of semi-honest adversaries.*

Lemma 2.12 *The protocol Π' constructed above does not securely compute $\mathcal{F}$ in the presence of a malicious adversary that corrupts party A.*

Note that for the sake of simplicity we are only considering an attack by party A. Allowing for an attack by a malicious party B would require an additional change to the second and third messages m_2 and m_3 sent in the protocol as well. These two lemmas together show that the protocol Π' securely computes $\mathcal{F}$ against semi-honest adversaries, but not against malicious adversaries with selective abort. ▮

Proof of Lemma 2.11: The proof of the proposition is simple, and we sketch it here. The protocol specification requires that the bit b sent at the end of the first message from party A to party B in protocol Π' is 0. As a result, in the presence of semi-honest adversaries, the next message $m_2^{\Pi'}$ sent by party B to party A is identical to the message that would have been sent in the original protocol, m_2^{Π}. The rest of the protocol Π' then proceeds identically to protocol Π. This allows us to reduce the security of protocol Π' against semi-honest adversaries to the security of protocolo Π against semi-honest adversaries, which completes the proof. ▮

Proof of Lemma 2.12: It suffices to describe the attack: Party A sends $m_1^{\Pi'} = \left(m_1^{\Pi}, 1\right)$ to party B. As the party B is honest, it follows the protocol specification, which requires it to send its entire private input to party A in its next message $m_2^{\Pi'}$. This breaks the security of the protocol. ▮

2.7 Discussion

It is important to be mindful of exactly what is guaranteed by security in these models. In the semi-honest model, there are no claims made to security if any of the parties deviate from the protocol. Therefore, the semi-honest model can be thought of as simply ensuring the privacy of the communication in the protocol. The adversary is essentially a passive spectator to the protocol (even though it may have corrupted a party), and all it does is study the messages received from the other parties to try and retrieve information about the other parties private inputs.

Given this, it is important to be confident that **all** parties will not deviate from the protocol before using a semi-honest secure protocol in an application. If such confidence is not achievable, then it is necessary to consider using a stronger notion of security – the covert model is one that provides some confidence by providing some assurance of detection of deviations.

Note however, that even the malicious model does not account for parties that modify their private inputs. That is, we cannot avoid the possibility that a party uses an input different from the one they locally possess. This means that a party could obtain certain information about other parties' private inputs if they maliciously pick their inputs. This is a limitation of the problem we are trying to solve, as such behavior cannot be prevented. We must always be mindful of this while analyzing the usefulness of multi-party computation techniques to different applications. Failing to do so could lead to real-world vulnerabilities in protocols that are "provably-secure" – such as those that can be exploited using enumeration attacks [HWS+21] (which we discuss later).

3 Private Set Intersection and Variants

Private set intersection (PSI) is a special case of the more general multi-party computation problem. One classification of the variants is based on whether the scenario is symmetric or asymmetric. The **asymmetric** scenario is one in which the two parties receive different functions of the inputs, that is, the functionality is of the form $(x, y) \mapsto (f_1(x, y), f_2(x, y))$. In a number of scenarios, a special case is considered such that one party does not learn any information while the other learns some function. That is, the functionality is of the form $(x, y) \mapsto (\perp, f(x, y))$. The **symmetric** scenario is a special case where both the parties receive the same function of the inputs, that is, the functionality is of the form $(x, y) \mapsto (f(x, y), f(x, y))$. In what follows, we assume that X and Y are sets of elements that are subsets of a common domain, which we denote by Dom.

PRIVATE SET INTERSECTION (PSI). Here, both the parties participating in the protocol have sets of elements, and they wish to find the intersections of their sets while revealing no further information about their sets. The symmetric functionality is $(X, Y) \mapsto (X \cap Y, X \cap Y)$, while the asymmetric functionality is $(X, Y) \mapsto (\perp, X \cap Y)$.

FUNCTION-PSI (F-PSI). There are various application scenarios where it is desired not to reveal the entire intersection to a party, but rather only reveal some aggregate computation performed on the intersection. In a function-PSI protocol, the parties have their private input sets, and at least one of the parties receives the result of performing some fixed function on the intersection and no additional information. That is, *neither* party receives the actual intersection. The symmetric functionality is $(X, Y) \mapsto (f(X \cap Y), f(X \cap Y))$, while the asymmetric functionality is $(X, Y) \mapsto (\perp, f(X \cap Y))$.

PSI-SUM. This can be considered to simply be a special case of the function-PSI variant, where the fixed function in question sums up values that correspond to each element in the intersection and returns the result. The usefulness of this variant for applications like ad-conversion has motivated the development of specialized protocols for this variant []. The asymmetric version of this functionality is $(X, Y) \mapsto (\perp, \mathsf{sum}(X \cap Y))$, where we define $\mathsf{sum} : P(\mathsf{Dom}) \to \mathsf{Dom}$ as $\mathsf{sum}(S) = \sum_{s \in S} s$ (Under the assumption that the set Dom is closed under addition, for the appropriate definition of the addition operation).

PSI CARDINALITY (PSI-CA). This is another special case of the function-PSI variant. Here, the fixed function simply returns the cardinality of (i.e. the number of elements in) the intersection. This variant has uses in scenarios where only the size of the overlap between the parties' sets is of interest. The asymmetric version of this functionality is $(X, Y) \mapsto (\perp, |X \cap Y|)$.

AUTHORIZED PSI (APSI). This variant [CZ09, DKT10, DT10] adds in a mechanism to "authorize" client inputs. In addition to the usual two parties, there is additionally assumed to be a trusted certification authority that certifies the input sets. This certification authority has no role to play in the actual intersection protocol, and does not need to be trusted for anything beyond its certificates of the input sets. Let DS be a digital signature scheme, and let the certification authority (CA) have $(vk, sk) \leftarrow\$ \mathsf{DS.K}$ as its verification and signing keys. The asymmetric version of this functionality can be written as $((X, \sigma_X), (Y, \sigma_Y)) \mapsto (\perp, g(X, Y, \sigma_X, \sigma_Y))$, where $\sigma_X \leftarrow\$ \mathsf{DS.S}(1^\lambda, sk, vk, X)$ and $\sigma_Y \leftarrow\$ \mathsf{DS.S}(1^\lambda, sk, vk, Y)$ are the signatures of the CA for the sets of the two parties. The function $g(X, Y, \sigma_X, \sigma_Y)$ returns the set intersection $X \cap Y$ if $\mathsf{DS.V}(1^\lambda, vk, X, \sigma_X) = \mathsf{true}$ and $\mathsf{DS.V}(1^\lambda, vk, Y, \sigma_Y) = \mathsf{true}$, and returns $\perp$ otherwise.

4 Applications

PSI protocols are used in practice in a variety of different applications. We consider a few of these applications in this section. We start by providing some background for each application and the problem it is trying to solve. We then describe the way in which a PSI protocol can be used for that application.

4.1 Ad-conversion systems

Advertising forms the backbone of the business models of most internet services today. Services are often provided for free, in exchange for users being shown advertisements. The income from these advertisements provides, at the very least, the money necessary to keep the service up to date and running. More commonly, advertisements are responsible for the profits made by the companies providing the services. In recent years, services have begun making greater and greater use of the vast amounts of information they possess about their users in order to provide targeted advertising to users. This makes adverts more valuable, since an advertiser can be assured that the advert will be shown to someone who is statistically more likely to be swayed by that specific kind of advert. This new "personalized" business model has been dubbed Surveillance Capitalism [Zub18].

In surveillance capitalism, both the advertiser and the service provider have motivation to learn how effective the ads shown to the users are, a statistic known as the ad conversion rate. The service provider would use this data to improve on its targeting mechanism for ads, i.e. the recommendation system, and to adjust prices for showing adverts (more effective targeting would command higher prices). Similarly, the advertiser would like this data in order to determine how much it is worth spending on advertising with that service provider.

It is thus no surprise that this is the most common motivation for the construction of PSI protocols in recent literature. It is the main application put forth in two works by Google, (GoogleA [IKN+17] and GoogleB [IKN+19]), and is also put forth as a motivating application in other works [PSZ18, PRTY19, PRTY20, CM20].

USE CASE. One party (the service provider) has a set of all users that a particular advert was shown to. The second party (the advertiser) has a set of all users that bought an item displayed in the particular advert, along with the amounts they spent. The user identifiers are assumed to be from some common database (which can be obtained using browser cookies, for example). For competitive reasons, the two parties are not comfortable with sharing their own sets. However, they would like to know (1) the number of users that saw the particular advert and then went on to buy the product from the advertiser, and (2) the amount that they spent on the advertiser's platform. Learning (1) would require a protocol for PSI-CA, while learning (2) would use a protocol for PSI-Sum. GoogleA and GoogleB define a variant called the *Private Set Intersection-Sum with Cardinality*, that combines the variants to solve (1) and (2) simultaneously, and providing the answer to (1) to both parties, but the answer to (2) to only one of the parties.

4.2 Private contact discovery

This application is of great interest to various secure messaging and social media platforms, especially those with a focus on privacy, such as Signal [Mar14]. The goal of private contact discovery is to allow the platform to let a new user connect to those people on the platform that the user is already connected to via having their contact details stored on their device. Network effects in the information economy [SV00] make it essential for new platforms to make it simple for new users to find people they know on the platform, because new users will only adopt a new platform if enough of their network has already adopted the platform.

USE CASE. This is a client-server setting. The client is the new user, who has a set of contacts stored locally on their device. The server is that of the platform, which has a set of the users already registered on the platform. The goal is to provide the client with the intersection of the two sets, which is those contacts the client has that correspond to people already using the platform. The client would not like their contacts to be stored on the server, or for the platform to obtain the contacts the client has [Mar14]. Correspondingly, the server too would not want to reveal its set of registered users to the client. This is the motivation for performing the operation in a privacy-preserving manner, and it can be achieved using a protocol computing the asymmetric version of the PSI functionality (where only the client is given the intersection).

4.3 Checking password compromise, and sharing security incident information

A **security incident** is the umbrella term used to describe the range of events wherein the data or systems of an organization or individual are compromised. Security incidents regularly make headlines, and result in loss of reputation for the affected organization, depending on the effectiveness of their response in mitigating the impact of the breach. The sharing of information between different incident handlers can go a long way towards quick corrective action. However, incident data usually contains sensitive information, which makes businesses reluctant to share it with outsiders. Even if the data is shared, it is done so with the use of legal agreements to safeguard it. Security incident management globally would be greatly improved through the open sharing of incident data. However, note that an organization only needs to be aware of similar incidents in order to learn possible mitigations. Therefore, it would be possible to use secure computation techniques including PSI protocols in order to reveal only the pertinent information to an organization requesting it, while masking the sensitive parts of the data [PSZ18].

A special scenario of a security incident is the issue of **password compromise**. Here an individual is checking to see if their password exists in a list of passwords that are known to have been hacked into. It is obvious that the individual does not want to reveal his password publicly. Similarly, making the entire list of compromised passwords public gives potential attackers a starting point for trying out possible passwords on a platter. This is the perfect set-up for a PSI protocol (or a private membership testing protocol), where the user wants to learn if there is an overlap between the singleton set of his password and the set of passwords that have been compromised.

Use Case. The IETF has a Managed Incident Lightweight Exchange (MILE) working group [Mor14], that has developed both data formats and transport protocols for the exchange of security incident information. The data format ensures that all incidents are labeled using a common syntax. A company that has had a security incident can enter the protocol for asymmetric PSI with a set with the symptoms of its incident. The protocol is run with the central database, and the company learns which of the symptoms have already been logged into the database (ideally along with successful mitigations).

The password compromise use case is very straightforward. A PSI-CA protocol is run between the user and the service holding the list of compromised passwords. We work in the asymmetric setting which gives only the user the answer. This is because there is no need for the service to learn whether the user has a hacked password. If the PSI-CA protocol outputs a non-zero answer, that means the users password has a high chance of being compromised and needs to be changed.

4.4 Genomics

The sequencing of the human genome has opened a number of possibilities with respect to more personalized forms of healthcare and testing. An individual's genome uniquely identifies them, and also provides information about their heritage, possible disease predispositions, and other traits [BBD$^+$11]. Therefore, the more widespread use of genomic data automatically raises concerns about the privacy of individual person's genomes. Troncoso-Pastoriza et al [TKC07] study the addition of privacy preservation in DNA searching techniques. However, they do not use PSI based solutions, so we consider this out of scope of this paper. Baldi et al (BBDGT) [BBD$^+$11] have considered applications of genomic computation such as Paternity Testing, Personalized Medicine, and Genetic Compatibility Tests, with a view to using different forms of PSI to provide privacy guarantees for individuals genomes.

The human genome is about 99.5% identical for every human being. However, when two individuals are closely related, such as via a parent-child relationship, there is an even greater similarity between their genomes. As a result, a simple computational technique to perform a paternity test is to simply compare the genomes of the two individuals. If the similarity between the genomes is more than a certain threshold above 99.5%, then a positive result is obtained. Similar thresholds are considered for testing genomic compatibility.

Use Case. The two clients interested in the paternity test run a PSI-CA protocol using their respective genomes as input. This protocol will give the size of the overlap between the two clients genomes – which can then be used to confirm a parent-child relation as above, that is, if the overlap is greater than a certain threshold.

However, this solution is very inefficient, and takes about 4.5 days of communication between the two parties [BBD+11]. BBDGT suggests two improvements to improve the efficiency, as follows. The first is to compare a properly chosen fraction of the genome instead of the entire genome (as suggested in domain specific papers [GS06]). The second is to simulate Restriction Fragment Length Polymorphism (RFLP) techniques to use specific enzymes to split the genome into fragments, and then compare the sizes of these fragments between the two parties. If the size of the intersection is greater than some threshold a positive result is obtained. BBDGT then uses a PSI-CA protocol to perform this in a privacy-preserving manner.

4.5 Contact Tracing

The sustained and quick spread of infections caused by the COVID-19 pandemic since early 2020 has led to the development of many techniques to manage the spread of the disease. One of the most effective ways of slowing the spread of an infectious disease is through contact tracing [FG21]. The bare bones structure of contact tracing is as follows. Whenever an individual tests positive for the disease, a list of persons (s)he has been in contact with in the recent past is requested. The duration of this recent past window is based on the specifics of the disease being managed, and is generally accepted to be 14 days for the COVID-19 virus. Once the list of contacts is obtained, those contacts are informed that they have been in contact with someone who has tested positive for the disease, and to quarantine and get tested. This has the benefit of reducing the essential testing requirement to only those who have been in contact with an infected individual, which can be very important when the spread of the disease is rampant and the public health system is under a lot of stress. Quickly obtaining the contact list and proactive testing can help slow the spread of the disease to a level that does not overburden the healthcare system.

However, there are significant privacy concerns with such a procedure, as it requires individuals to reveal who they have been meeting, along with the revelation of sensitive medical information such as the positive diagnosis of an individual to other persons. The severity of the pandemic means that the benefits of contact tracing outweigh privacy concerns to some extent, though to ameliorate these privacy concerns, there have been concerted efforts across the world to build privacy-preserving protocols to perform contact tracing [TPH+20, CFG+20, VABMB+20, RPB20, TSS+20, DIL+20, DPT20].

Use Case. For contact tracing, the players involved are the citizens and the government health authority. The citizens have a set of people that they were in contact with during a certain time window (let's call this the *citizen set*), and the health authority has a set of people who have tested positive for the disease in a window that overlaps with the above-mentioned time window (let's call this the *authority set*). The details of the window sizes will depend on the specifics of the disease, such as the period the virus can transfer to others and so on. A PSI-CA protocol can then be run using these two sets in order to learn if there is any overlap between the citizen set and authority set – the larger the overlap the more the risk of infection. The symmetric version of the functionality might be required in order to allow the health authority to follow up on whether the citizen has gotten tested after the protocol run has shown a non-zero overlap.

Certain health authorities, most notably Singapore's TraceTogether but also the EU's ROBERT and NTK, use centralized techniques for contact tracing. The downside of such an approach is that it requires unconditional trust in the central authority. The central authority will receive, and be able to track, information about the citizens such as their daily routine, and who they interact with. This has the potential for great misuse at the hands of the state. These reasons are why a number of academics have pushed for the use of decentralized applications, which do not require trusting a central authority. Vaudenay [Vau20] provides a more nuanced look at the centralized versus decentralized debate.

4.6 Proximity Testing and Ride Sharing

The goal of **proximity testing** is to compute whether two parties are close to each other, without revealing their distances and positions to each other, or to any other external party. There has been plenty of work towards achieving this in a privacy-preserving manner [NTL+11, ZGH07, STS+09, STSY10, FVM+10, MFB+11, SG14, PAP+15, HOS15]. The usual technique is via a threshold, whereby the parties learn if they are within a certain distance from each other or not, with no further information. The challenge with this technique is that if implemented naively, it can enable a mobile adversary to move around while checking for proximity to the other party, and then triangulate the other party's exact position.

The technology models for **ride-sharing** applications typically share the location of the user with the service. This leaves open the possibility for misuse by employees of the service, something that has happened in the case of Uber [Bes]. In response, Hallgren, Orlandi and Sabelfeld (HOS) [HOS17] proposed a prototype of a ride-sharing app that is decentralized and requires no trust in a third party (i.e. the service application). Optimizing the route when there are multiple parties with different start and end points is a challenging problem even without the privacy aspect. Therefore, HOS focus on two parameters – (1) the *proximity* of the journeys' start and end points, and (2) the *intersection* of the route between the start and end points. The first of these parameters requires a variation on the proximity testing protocols, specifically that the match should only occur when *both* the start and end points are within some distance of each other. This is slightly more involved than running two proximity-testing protocols, since running two independent protocols would reveal information when only one of the pairs of start or end points are close to each other. The second of these parameters can be computed securely through the use of PSI techniques.

USE CASE. One technique to perform private proximity testing is by using of location tags [NTL+11]. A location tag is a secret associated with a point in space and time that is generated by making use of the electromagnetic signals present in the surroundings. If two tags are generated at the same place and time, they match with a high probability (where matching is done using the Hamming distance or set distance) [QBLE09]. At the same time, an adversary that is not at that specific place or time should be unable to produce a tag that matches a tag for that place and time. Using such location tags, private proximity testing can be performed by using a PSI-CA protocol to measure the size of the overlap between the users tags. An intersection size that crosses a certain threshold signifies that the two participants are in close proximity to each other.

4.7 Relationship path discovery in social networks

The relationship graph is a core feature of social networks and online life today. The nodes of the graph are the users, and users that are directly connected to each other (for example, as friends on Facebook) have an edge between their nodes The relationship graph can thus provide the relationship paths that are present between any two users.

The main benefit of the relationship graph to a user is for establishing trust [MPGP09]. When it comes to reviews and recommendations, for example, a user would put more stock in the review of a person they know, as their knowledge of that person would add more information about the accuracy of the review. A relationship path between an employer and an applicant could provide the employer with a means of getting trusted opinions on the applicant.

As things stand currently, most of this data is in the hands of social network companies. These companies provide some of this information to their users, for example via mutual friends on Facebook, or common connections on LinkedIn. This graph represents a lot of sensitive information, and there is a potential for misuse via tracking and monitoring of multiple individuals in the real world by bad actors who gain access to the graph. In order to prevent this, Mezzour et al (MPGP) [MPGP09] describe techniques to discover relationship paths between individuals in a privacy preserving manner.

Use Case. Consider a relationship graph. There are two parties who want to learn whether they have a relationship path between them that is smaller than a certain length. The simple solution is that one party first sends tokens to its neighbors in the graph. Each party that receives tokens then forwards tokens as per the protocol to their neighbors, until all nodes are covered. This is called the *token-flooding phase*. After the tokens have been propagated through the graph, the two parties run a PSI protocol. The input set of the first party is the set of possible tokens that could have been generated during the token flooding (this can be simulated by the first party). The input set of the second party is the set of tokens it received during the token flooding phase. We omit details about randomization and optimization steps during the token generation and propagation from this overview and refer the reader to the protocol specification in MPGP.

5 Security considerations

In this section, we analyze which notions of security would accurately model the real world trust between protocol participants. In the majority of applications that make use of PSI protocols in the literature, security is considered only in the semi-honest model. Only a few recent works consider malicious security for their protocols [RR17a, RR17b, CHLR18, PRTY20].

The main motivation for restriction to the semi-honest model is efficiency – protocols achieving security in the malicious model tend to be much less efficient in terms of communication and computational complexity [PSWW18]. PSI protocols generally do not scale well enough [Mar14] on these complexity metrics to work in real time on large data sets, which is a requirement for most applications today. However, the semi-honest model implicitly requires trusting that the adversary will conform to the protocol specifications. We now discuss whether this is a reasonable assumption to make.

To do so, we categorize the parties involved in these protocols into two classes, along the lines of Kamara's keynote at CRYPTO 2020 [Kam20] – (1) parties can be"**people**", that is, regular individuals, or (2) parties can be organizations and governments, which we call the class of "**power**". Based on this categorization, two-party protocols can be split into three types – (1) those where both parties are "people", (2) those where one party is in the class of "people" and one party

People - People	People - Power	Power - Power
Genomics	Private Contact Discovery	Ad-conversion systems
Proximity Testing	Password Compromise Checking	Security Incident Sharing
Ride Sharing	Contact Tracing	
Relationship Path Discovery		

Figure 3: Table classifying various applications into whether their underlying PSI protocols are people-people, people-power, or power-power protocols.

belongs to "power", and (3) those where both parties belong to the class of "power". A table of the applications using PSI categorized by the type of protocol as above is in Figure 3. We will now look at the security considerations for the two classes of parties.

5.1 People

These are individuals that run protocols on day-to-day devices like mobile phones and laptops. A good majority of these individual users would likely just download the necessary application and let the application run normally. If every user acted as above, using semi-honest protocols would be sufficient. However, we cannot discount the presence of malicious attackers who tweak the executable on their device so as to cause deviations from the specification. It could also be that the users download the application from somewhere other than the official website, in which case they might be tricked into downloading such a tweaked executable. In this case, even though the user is not malicious, the tweaked app could be transmitting any gleaned information to the hacker who tweaked the application.

TRUSTED EXECUTION ENVIRONMENTS. One suggested solution that allows for the use of semi-honest security [TLP+17] is to run the individual's side of the protocol in a secure and tamper-proof environment. The most popular example of such a trusted execution environment is Intel's Software Guard eXtensions (SGX). SGX uses trusted hardware that is built into Intel chips to create a secure container which contains the protocol specification [CD16]. The individual then feeds its data to the protocol, which runs in an isolated environment that the individual is unable to tamper with. A *remote attestation* can be provided to confirm the code has not been tampered with.

This comes with its own set of issues [KRS+19]. The SGX mode of operation has not been shown to be provably secure. More concerningly, there have also been some attacks like Foreshadow [VMW+18] that have subverted this mode of operation. The Foreshadow attack is able not just to extract secrets stored within the SGX enclaves, but also to forge remote attestations. Since SGX is based on trusted hardware, fixing such issues will often require hardware upgrades, which will usually only happen when people buy new devices, which they may not do often. This could leave large numbers of users vulnerable to attack. Furthermore, the code that is loaded into the SGX enclave will be added on installation of the app. Therefore, if the app is downloaded from an untrusted source, the user could still end up with a tweaked executable, which can then enter malicious code into the SGX enclave and learn sensitive information.

STRONGER SECURITY MODELS. Malicious security would clearly prevent a user from learning sensitive private information from the protocol. However, the increase in time such a protocol would take to run would greatly reduce its marketability, since it affects all users, even those who might be benign. For example, a person looking for a rideshare might not want to wait a significantly longer amount of time to get a ride scheduled just because there is a chance that he might be acting maliciously. On the other hand, security against covert adversaries might be an acceptable choice for these protocols. An adversary who deviates from the protocol would be caught with some probability that can be decided based on the use case. The penalties for being caught can be set to be sufficiently harsh so as to be deterrence enough not to attempt. However, it is not clear what these penalties would be. It would be hard to take punitive action beyond banning the user from the service if they are caught. There are however certain scenarios in which there could be legal consequences to being caught, such as in applications dealing with healthcare or genomic data due to HIPAA.

5.2 Power

In this category we consider organizations such as big companies, or even sovereign states. A common feature of parties in this category is that they have access to a large amount of resources, both in terms of the computational power at their disposal, but also in terms of the power they can often wield, especially when interacting with individuals. These organizations are often able to make decisions in their terms of service, and the only option for individuals is to agree to those decisions or stop using the service. Stopping using the service is usually not a feasible option, since there are often no viable alternatives that provide the same service (and reach, in the case of social media). When it comes to governments, there usually is no option but to comply with the legislation (unless there is strong separation of powers and a basis for the legislation being challenged in court), both for individuals and for organizations.

In general, organizations and governments have relatively strong incentives to gather data, be it to improve services, increase profits, or surveil the population and quash dissent. A major reason for them to adopt privacy-preserving techniques is public relations and optics. Being able to claim privacy for users can give companies a marketing boost, as in the case of Signal and Apple. Another related reason for companies to adopt privacy preserving technologies is in order to prevent data from being subpoenaed or obtained by the state through coercion – if the company does not look at or store the data it will be unable to provide such data.

The loss of reputation on being caught would be a major factor in any organization being wary of acting maliciously while running a semi-honest protocol. However, as we see in the next section, any deviation from the specification can be very effective in revealing sensitive information, and is extremely unlikely to be spotted in a semi-honest protocol. There would be greater confidence in an organizations claims of respecting privacy if it used stronger models such as the covert or malicious models.

CODE AUDITS. An alternative that could allow for the use of semi-honest secure protocols is the use of regular and surprise code audits. This option works mostly when the protocol is being run between two organizations (i.e. power-power protocols). The chance of an audit would deter a company from tweaking the protocol to deviate from the specification.

Code audits are of course not a foolproof method for catching malicious behavior. There is always a chance that a subtle change is missed. More insidiously, it is also possible to use a different compiler that introduces malicious behavior, while the behavior is as specified when compilation is done using a standard compiler [Tho84].

5.3 Provision of incorrect inputs

There is also the issue that parties cannot be stopped from picking their own inputs, and therefore can subvert the protocol by simply choosing a different input. When it comes to applications like ride sharing and proximity testing, both users are taking part in the protocol with a view to receiving some benefit from the information gained (such as a common route for a rideshare, or whether they are in the vicinity of the other party). Picking a different input would likely lead to a suboptimal output for the user from the protocol, which would not be in the their best interests. On the other hand, in genomics applications like paternity tests, there could be an incentive for someone to provide an incorrect input (such as someone else's genome) in order to avoid issues that arise from the result not going in their favor. Such situations could be prevented via the use of authorized PSI protocols, which test for the data being signed by a trusted authority (in this case, possibly the lab providing the genome data) and only return the desired output if the signature is valid. However, this has the downside of requiring trust in the certification authority. One could claim that if a third party is being trusted, then the entire computation could be outsourced to the third party without the complication of an MPC protocol. However, authorized PSI protocols trust the CA only with certification of the sets, which means there is no single point of failure if the CA does not store the sets it authorizes.

Enumeration attacks. Certain applications, most notably private contact discovery, have domains that are relatively small in size (the set of valid phone numbers). Some works [SFK+12, HWS+21] have studied the use of simple enumeration attacks to obtain information about the messaging server data set. Enumeration attacks simply use many different accounts with different contact lists in order to learn which numbers are present on the messaging service. This attack does not violate malicious security, since the adversary does not learn any information beyond the set intersection. Protecting against enumeration attacks is usually done via some form of rate-limiting, which prevents a single user from asking more than a certain number of contact discovery queries in a given time period, or bounding the number of contacts a person can check for.

6 Do we need to achieve security in the malicious model?

We have spoken earlier about how even the malicious model of security does not prevent a malicious party from using a different input in order to obtain information about the other party's input. This leads to the question of whether the loss of efficiency that happens when protocols are designed to be secure in the malicious model is worth it. After all, can an adversary not learn significant information about the other party's input via a clever choice of its input?

We stress that the malicious model does still offer significantly stronger guarantees of privacy than the semi-honest model. To highlight this, we pick a recent protocol by Chase and Miao (CM) [CM20] that is provably secure in the semi-honest model, and show how a simple deviation by one party can reveal the entire input set of the other party. Such an attack is clearly much preferred to the "clever choice of input" approach, since it is guaranteed to reveal the entire input set, not just those that match with the chosen adversarial input.

6.1 The semi-honest protocol from CM

The protocol proceeds in three phases – the first stage is a non-interactive **setup** phase, where both the sender and receiver generate some preliminary data. The second stage is the **oblivious transfer** phase, where the sender and receiver run some number of oblivious transfers in order to exchange some data. The third stage is the **evaluation** phase, in which the sender sends the receiver a message, after which the receiver computes the output of the protocol. Before we describe the protocol, we clarify some notation and definitions.

Oblivious Transfer (OT). OT protocols are one of the fundamental building blocks of many MPC protocols. There is a sender that has two input strings, s_0 and s_1, and a receiver that has an input bit b. The OT protocol securely provides the receiver with the string s_b, while revealing

```
$\Pi_{cm}.S(1^\lambda, X)$:
    $s \leftarrow_\$ \{0,1\}^w$

$\Pi_{cm}.R(1^\lambda, Y)$:
    for $i \in [w]$ do: $D_i \leftarrow 1^m$
    $D \leftarrow [D_1\ D_2\ \dots D_w]$ ; $k \leftarrow \{0,1\}^\lambda$
    for $y \in Y$ do:
        $v \leftarrow F_k(H_1(y))$
        for $i \in [w]$ do: $D_i[v[i]] \leftarrow 0$
    $A \leftarrow_\$ \{0,1\}^{m \times w}$ ; $B \leftarrow A \oplus D$

for $i \in [w]$ do:
    $C_i \leftarrow \mathbf{OT}_{\Pi_{cm}.R, \Pi_{cm}.S}((A_i, B_i), s[i])$

$\xleftarrow{k}$

S:
    $C \leftarrow [C_1\ C_2\ \dots C_w]$
    for $x \in X$ do:
        $v \leftarrow F_k(H_1(x))$
        $\psi \leftarrow C_1[v[1]] \| \dots \| C_w[v[w]]$
        $Z \leftarrow Z \cup \{H_2(\psi)\}$

$\xrightarrow{Z}$

R:
    for $y \in Y$ do:
        $v \leftarrow F_k(H_1(y))$
        $\psi \leftarrow A_1[v[1]] \| \dots \| A_w[v[w]]$
        if $(H_2(\psi) \in Z)$ then
            $I \leftarrow I \cup \{y\}$
    return $I$
```

Figure 4: The CM private set intersection protocol, Π_{cm}.

no information about s_{1-b} to the receiver, and no information about b to the sender. The OT functionality can thus be written as $((s_0, s_1), b) \mapsto (\bot, s_b)$. In the description of the CM protocol, we use $s \leftarrow \mathbf{OT}_{S,R}((s_0, s_1), b)$ to denote that the parties S and R run the OT protocol as the sender and receiver respectively, with the sender's inputs being s_0 and s_1, and the receivers inputs being b. The protocol outputs s to only the receiver R.

HAMMING CORRELATION ROBUSTNESS. This is a property of hash functions that was first introduced in Ishai et al in their work on OT extension [IKNP03]. The definition we use is from CM, which is a generalization of the original definition.

Definition 6.1 ([CM20], Definition 2.1) *Let H be a hash function with input length n. Then H is said to be d-Hamming correlation robust if, for any $a_1, a_2, \dots a_m, b_1, b_2, \dots b_m \in \{0,1\}^n$, with*

$\|b_i\|_{\mathsf{H}} \geq d$ *for each* $i \in [m]$*, and* $s \leftarrow_{\$} \{0,1\}^n$*, we have that*

$$(H(a_1 \oplus b_1 \cdot s), \ldots H(a_m \oplus b_m \cdot s)) \overset{\mathsf{c}}{\equiv} (F(a_1 \oplus b_1 \cdot s), \ldots F(a_m \oplus b_m \cdot s))$$

where $\oplus$ *denotes the bitwise XOR,* $\cdot$ *denotes the bitwise AND, and* F *is a random function.*

A detailed description of the protocol (adapted from Figure 3 in [CM20]) is provided in Figure 4. Chase and Miao show this protocol to be secure against semi-honest adversaries in the standard model for appropriate values of the parameters m, w when the underlying OT protocol is also secure against semi-honest adversaries, F is a PRF, H_1 is a collision resistant hash function, and H_2 is a d-Hamming correlation robust hash function ([CM20], Theorem 3.1).

6.2 A simple attack on the CM protocol by a malicious receiver

We now describe a simple deviation from the receiver specification that allows the receiver to learn the sender's input set. The malicious receiver simply sets the matrix D to be the null matrix instead of performing steps 2 through 5 in the specification given in Figure 4. It then follows the specification until it receives Z from the sender. Once it has Z, it can locally check if any element $x \in X$ by checking whether $H_2(A_1[v[1]]\| \ldots \|A_w[v[w]]) \in Z$, where $v \leftarrow \mathsf{F}_k(H_1(x))$. That is, it can use a dictionary attack to learn the senders entire input set.

6.3 A generic example of an attack

For this, we turn back to Proposition 2.10. Recall that there we construct a protocol that securely computes a functionality $\mathcal{F}$ against semi-honest adversaries, but that does not securely compute $\mathcal{F}$ against malicious adversaries. More importantly, we point out that the attack we provide to break malicious security (1) requires the adversary to **deviate from the protocol specification by only a single bit**, and (2) allows the adversary to **learn the entire private input set of the honest party** in the protocol. Thus, while the constructed protocol is contrived, it shows the existence of protocols that are semi-honest secure, yet are completely insecure (in that they completely reveal the honest party's input) with the minimal amount of deviation from the specification (just one bit). It is important to note that the modified construction was just to ensure that an explicit attack could be described. It may be possible to attack the original protocol itself via a malicious adversary, like we do for the CM protocol in the previous subsection.

7 Conclusions

Guarantees of privacy and security are a fundamental part of any decentralized interaction between parties, without which participants would be wary of sharing data freely. The field of multiparty computation works towards achieving privacy and security in the absence of a trusted party helping out with the computation. However, restricting to the semi-honest model of security implicitly requires trusting that the two parties (who may not know each other) will behave exactly as asked for by the protocol. We discuss how such trust might not be reasonable in most settings, and also consider using trusted execution environments and code audits as non-cryptographic ways to support the use of semi-honest protocols. At the same time, these methods also come with their downsides and vulnerabilities. The most important takeaway for application developers is to be clear of the security guaranteed while making use of a PSI protocol in an application.

As a result, it seems a worthwhile goal to aim to develop more efficient protocols for PSI (or variants) that achieve the stronger notions of security such as covert and malicious. It would also be helpful to develop a framework for obtaining provable security guarantees for execution in trusted environments like SGX.

The increased interest in the use of PSI protocols in practical applications makes efficiency an important goal, as we have stated above. However, most works decouple the questions of efficiency and security, in that they prove security asymptotically, and then provide concrete efficiency results (often with an evaluation in practice). We feel that an increase in usage in practice should be accompanied by the development of concrete security analyses for the protocols – that is, we would like to bound the probability with which an adversary could attack the protocol (called the adversary's advantage) in terms of the advantages of adversaries against the building blocks of the protocol. Furthermore, such an analysis would measure the adversary's runtime in terms of that of the adversaries against the building blocks, and the simulator. This would be very useful for picking the parameters appropriately while building the protocol, so that the advantages are small enough for the adversarial power that is relevant to the application.

8 Acknowledgements

Thanks to Mihir Bellare for the helpful comments and suggestions during the writing of this work.

References

[AB20] Vivek Arte and Mihir Bellare. Dual-mode NIZKs: Possibility and impossibility results for property transfer. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 859–881. Springer, Heidelberg, December 2020. (Cited on page 6.)

[AKL12] Emmanuel A. Abbe, Amir E. Khandani, and Andrew W. Lo. Privacy-preserving methods for sharing financial risk exposures. *American Economic Review*, 102(3):65–70, May 2012. (Cited on page 3.)

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 137–156. Springer, Heidelberg, February 2007. (Cited on page 8.)

[BBD$^+$11] Pierre Baldi, Roberta Baronio, Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Countering GATTACA: efficient and secure testing of fully-sequenced human genomes. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011*, pages 691–702. ACM Press, October 2011. (Cited on page 3, 15, 16.)

[BCD$^+$08] Peter Bogetoft, Dan Lund Christensen, Ivan Damgard, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Multiparty computation goes live. Cryptology ePrint Archive, Report 2008/068, 2008. https://eprint.iacr.org/2008/068. (Cited on page 3.)

[Bea91] Donald Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, January 1991. (Cited on page 5.)

[Bes] Chanelle Bessette. Does uber even deserve our trust? *Forbes*. (Cited on page 17.)

[BKK$^+$16] Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. Students and taxes: a privacy-preserving study using secure computation. *Proc. Priv. Enhancing Technol.*, 2016(3):117–135, 2016. (Cited on page 3.)

[BKL$^+$14] Dan Bogdanov, Liina Kamm, Sven Laur, Pille Pruulmann-Vengerfeldt, Riivo Talviste, and Jan Willemson. Privacy-preserving statistical data analysis on federated databases. In *Annual Privacy Forum*, pages 30–55. Springer, 2014. (Cited on page 3.)

[BMRR21] Saikrishna Badrinarayanan, Peihan Miao, Srinivasan Raghuraman, and Peter Rindal. Multi-party threshold private set intersection with sublinear communication. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 349–379. Springer, Heidelberg, May 2021. (Cited on page 4.)

[BTW12] Dan Bogdanov, Riivo Talviste, and Jan Willemson. Deploying secure multi-party computation for financial data analysis - (short paper). In Angelos D. Keromytis, editor, *FC 2012*, volume 7397 of *LNCS*, pages 57–64. Springer, Heidelberg, February / March 2012. (Cited on page 3.)

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. (Cited on page 5.)

[CD16] Victor Costan and Srinivas Devadas. Intel SGX explained. Cryptology ePrint Archive, Report 2016/086, 2016. https://eprint.iacr.org/2016/086. (Cited on page 19.)

[CFG$^+$20] Justin Chan, Dean Foster, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Puneet Sharma, et al. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing. *arXiv preprint arXiv:2004.03544*, 2020. (Cited on page 16.)

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1223–1237. ACM Press, October 2018. (Cited on page 18.)

[CJS12] Jung Hee Cheon, Stanislaw Jarecki, and Jae Hong Seo. Multi-party privacy-preserving set intersection with quasi-linear complexity. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 95-A(8):1366–1378, 2012. Also available at https://eprint.iacr.org/2010/512. (Cited on page 4.)

[CLR17] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017. (Cited on page 4.)

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63. Springer, Heidelberg, August 2020. (Cited on page 4, 5, 14, 21, 22, 23.)

[CWB18] Hyunghoon Cho, David J Wu, and Bonnie Berger. Secure genome-wide association analysis using multiparty computation. *Nature biotechnology*, 36(6):547–551, 2018. (Cited on page 3.)

[CZ09] Jan Camenisch and Gregory M. Zaverucha. Private intersection of certified sets. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 108–127. Springer, Heidelberg, February 2009. (Cited on page 4, 13.)

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 789–800. ACM Press, November 2013. (Cited on page 4.)

[DGT12] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, *CANS 12*, volume 7712 of *LNCS*, pages 218–231. Springer, Heidelberg, December 2012. (Cited on page 4.)

[DIL$^+$20] Samuel Dittmer, Yuval Ishai, Steve Lu, Rafail Ostrovsky, Mohamed Elsabagh, Nikolaos Kiourtis, Brian Schulte, and Angelos Stavrou. Function secret sharing for PSI-CA: with applications to private contact tracing. *CoRR*, abs/2012.13053, 2020. (Cited on page 16.)

[DKT10] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 213–231. Springer, Heidelberg, December 2010. (Cited on page 4, 13.)

[DMRY09] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Efficient robust private set intersection. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 125–142. Springer, Heidelberg, June 2009. (Cited on page 4.)

[DPT20] Thai Duong, Duong Hieu Phan, and Ni Trieu. Catalic: Delegated PSI cardinality with applications to contact tracing. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 870–899. Springer, Heidelberg, December 2020. (Cited on page 16.)

[DT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, Heidelberg, January 2010. (Cited on page 4, 13.)

[FG21] Thiemo Fetzer and Thomas Graeber. Measuring the scientific effectiveness of contact tracing: Evidence from a natural experiment. *Proceedings of the National Academy of Sciences*, 118(33), 2021. (Cited on page 16.)

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004. (Cited on page 4.)

[FVM$^+$10] Dario Freni, Carmen Ruiz Vicente, Sergio Mascetti, Claudio Bettini, and Christian S. Jensen. Preserving location and absence privacy in geo-social networks. In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 309–318. ACM, 2010. (Cited on page 17.)

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 6.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. (Cited on page 3.)

[GN19] Satrajit Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 154–185. Springer, Heidelberg, May 2019. (Cited on page 4.)

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, USA, 2004. (Cited on page 10.)

[GS06] J. Raphael Gibbs and Andrew Singleton. Application of genome-wide single nucleotide polymorphism typing: Simple association and beyond. *PLOS Genetics*, 2(10):1–7, 10 2006. (Cited on page 16.)

[HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012. (Cited on page 4.)

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010. (Cited on page 3, 5, 7, 8, 9, 10, 11.)

[HOS15] P. Hallgren, M. Ochoa, and A. Sabelfeld. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6, 2015. (Cited on page 17.)

[HOS17] Per A. Hallgren, Claudio Orlandi, and Andrei Sabelfeld. PrivatePool: Privacy-preserving ridesharing. In Boris Kpf and Steve Chong, editors, *CSF 2017 Computer Security Foundations Symposium*, pages 276–291. IEEE Computer Society Press, 2017. (Cited on page 17.)

[HV17] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 175–203. Springer, Heidelberg, March 2017. (Cited on page 4.)

[HWS$^+$21] Christoph Hagen, Christian Weinert, Christoph Sendner, Alexandra Dmitrienko, and Thomas Schneider. All the numbers are US: large-scale abuse of contact discovery in mobile messengers. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtual, February 21-25, 2021*. The Internet Society, 2021. (Cited on page 12, 21.)

[IKN$^+$17] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. Cryptology ePrint Archive, Report 2017/738, 2017. https://eprint.iacr.org/2017/738. (Cited on page 4, 14.)

[IKN$^+$19] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Mariana Raykova, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. On deploying secure computing commercially: Private intersection-sum protocols and their business applications. Cryptology ePrint Archive, Report 2019/723, 2019. https://eprint.iacr.org/2019/723. (Cited on page 4, 14.)

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003. (Cited on page 22.)

[IOP18] Roi Inbar, Eran Omri, and Benny Pinkas. Efficient scalable multiparty private set-intersection via garbled bloom filters. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 235–252. Springer, Heidelberg, September 2018. (Cited on page 4.)

[Kam20] Seny Kamara. Crypto for the People, 2020. Invited talk. (Cited on page 18.)

[KK20] Ferhat Karakoç and Alptekin Küpçü. Linear complexity private set intersection for secure two-party protocols. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 409–429. Springer, Heidelberg, December 2020. (Cited on page 4.)

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016. (Cited on page 4.)

[KLC12] Myungsun Kim, Hyung Tae Lee, and Jung Hee Cheon. Mutual private set intersection with linear complexity. In Souhwan Jung and Moti Yung, editors, *WISA 11*, volume 7115 of *LNCS*, pages 219–231. Springer, Heidelberg, August 2012. (Cited on page 4.)

[KMP$^+$17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1257–1272. ACM Press, October / November 2017. (Cited on page 4.)

[KRS$^+$19] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1447–1464. USENIX Association, August 2019. (Cited on page 19.)

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Heidelberg, August 2005. (Cited on page 4.)

[LP08] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. Cryptology ePrint Archive, Report 2008/197, 2008. https://eprint.iacr.org/2008/197. (Cited on page 3.)

[Mar14] Moxie Marlinspike. The difficulty of private contact discovery, January 03, 2014. https://signal.org/blog/contact-discovery/. (Cited on page 14, 18.)

[MFB+11] Sergio Mascetti, Dario Freni, Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *VLDB J.*, 20(4):541–566, 2011. (Cited on page 17.)

[Mor14] Kathleen Moriarty. Internet Engineering Task Force (IETF) Managed Incident Lightweight Exchange (MILE) Working Group, January 13, 2014. https://https://trac.ietf.org/trac/mile/wiki. (Cited on page 15.)

[MPGP09] Ghita Mezzour, Adrian Perrig, Virgil D. Gligor, and Panos Papadimitratos. Privacy-preserving relationship path discovery in social networks. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 189–208. Springer, Heidelberg, December 2009. (Cited on page 18.)

[MPR+20] Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2020. (Cited on page 4.)

[MR92] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 392–404. Springer, Heidelberg, August 1992. (Cited on page 5.)

[NAA+09] G. Sathya Narayanan, T. Aishwarya, Anugrah Agrawal, Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Multi party distributed private matching, set disjointness and cardinality of set intersection with information theoretic security. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2009. (Cited on page 4.)

[NTL+11] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS 2011*. The Internet Society, February 2011. (Cited on page 17.)

[NTY21] Ofri Nevo, Ni Trieu, and Avishay Yanai. Simple, fast malicious multiparty private set intersection. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1151–1165. ACM, 2021. (Cited on page 4.)

[PAP+15] Iasonas Polakis, George Argyros, Theofilos Petsios, Suphannee Sivakorn, and Angelos D. Keromytis. Where's wally?: Precise user discovery attacks in location proximity services. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 817–828. ACM Press, October 2015. (Cited on page 17.)

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 401–431. Springer, Heidelberg, August 2019. (Cited on page 4, 14.)

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Heidelberg, May 2020. (Cited on page 4, 14, 18.)

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015*, pages 515–530. USENIX Association, August 2015. (Cited on page 4.)

[PSWW18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 125–157. Springer, Heidelberg, April / May 2018. (Cited on page 18.)

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 797–812. USENIX Association, August 2014. (Cited on page 4.)

[PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Transactions on Privacy and Security*, 21(2):7:1–7:35, 2018. (Cited on page 14, 15.)

[QBLE09] Di Qiu, Dan Boneh, Sherman Lo, and Per Enge. Robust location tag generation from noisy location data for security applications. In *Proceedings of the 2009 International Technical Meeting of The Institute of Navigation*, pages 586–597, 2009. (Cited on page 17.)

[RA18] Amanda C. Davi Resende and Diego F. Aranha. Faster unbalanced private set intersection. In Sarah Meiklejohn and Kazue Sako, editors, *FC 2018*, volume 10957 of *LNCS*, pages 203–221. Springer, Heidelberg, February / March 2018. (Cited on page 4.)

[RPB20] Ramesh Raskar, Deepti Pahwa, and Robson Beaudry. Contact tracing: Holistic solution beyond bluetooth. *IEEE Data Eng. Bull.*, 43(2):67–70, 2020. (Cited on page 16.)

[RR17a] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 235–259. Springer, Heidelberg, April / May 2017. (Cited on page 4, 18.)

[RR17b] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1229–1242. ACM Press, October / November 2017. (Cited on page 4, 18.)

[RT21] Mike Rosulek and Ni Trieu. Compact and malicious private set intersection for small sets. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1166–1181. ACM, 2021. (Cited on page 4.)

[SFK$^+$12] Sebastian Schrittwieser, Peter Frühwirt, Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Markus Huber, and Edgar R. Weippl. Guess who's texting you? Evaluating the security of smartphone messaging applications. In *NDSS 2012*. The Internet Society, February 2012. (Cited on page 21.)

[SG14] Jaroslav Sedenka and Paolo Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In Shiho Moriai, Trent Jaeger, and Kouichi Sakurai, editors, *ASIACCS 14*, pages 99–110. ACM Press, June 2014. (Cited on page 17.)

[SS08] Yingpeng Sang and Hong Shen. Privacy preserving set intersection based on bilinear groups. In Gillian Dobbie and Bernard Mans, editors, *Computer Science 2008, Thirty-First Australasian Computer Science Conference (ACSC2008), Wollongong, NSW, Australia, January 22-25, 2008*, volume 74 of *CRPIT*, pages 47–54. Australian Computer Society, 2008. (Cited on page 4.)

[STS$^+$09] Laurynas Siksnys, Jeppe Rishede Thomsen, Simonas Saltenis, Man Lung Yiu, and Ove Andersen. A location privacy aware friend locator. In Nikos Mamoulis, Thomas Seidl, Torben Bach Pedersen, Kristian Torp, and Ira Assent, editors, *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009, Aalborg, Denmark, July 8-10, 2009, Proceedings*, volume 5644 of *Lecture Notes in Computer Science*, pages 405–410. Springer, 2009. (Cited on page 17.)

[STSY10] Laurynas Siksnys, Jeppe Rishede Thomsen, Simonas Saltenis, and Man Lung Yiu. Private and flexible proximity detection in mobile social networks. In Takahiro Hara, Christian S. Jensen, Vijay Kumar, Sanjay Madria, and Demetrios Zeinalipour-Yazti, editors, *Eleventh International Conference on Mobile Data Management, MDM 2010, Kanas City, Missouri, USA, 23-26 May 2010*, pages 75–84. IEEE Computer Society, 2010. (Cited on page 17.)

[SV00] Carl Shapiro and Hal R. Varian. *Information Rules: A Strategic Guide to the Network Economy.* Harvard Business School Press, USA, 2000. (Cited on page 14.)

[Tho84] Ken Thompson. Reflections on trusting trust. *Commun. ACM*, 27(8):761763, August 1984. (Cited on page 20.)

[TKC07] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Celik. Privacy preserving error resilient dna searching through oblivious automata. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 519–528. ACM Press, October 2007. (Cited on page 15.)

[TLP$^+$17] Sandeep Tamrakar, Jian Liu, Andrew Paverd, Jan-Erik Ekberg, Benny Pinkas, and N. Asokan. The circle game: Scalable private membership test using trusted hardware. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *ASIACCS 17*, pages 31–44. ACM Press, April 2017. (Cited on page 19.)

[TPH$^+$20] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, et al. Decentralized privacy-preserving proximity tracing. *arXiv preprint arXiv:2005.12273*, 2020. (Cited on page 16.)

[TSS$^+$20] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *CoRR*, abs/2004.13293, 2020. (Cited on page 16.)

[VABMB$^+$20] Sydney Von Arx, Isaiah Becker-Mayer, Daniel Blank, Jesse Colligan, Rhys Fenwick, Mike Hittle, Mark Ingle, Oliver Nash, Victoria Nguyen, James Petrie, et al. Slowing the spread of infectious diseases using crowdsourced data. *IEEE Data Eng. Bull.*, 43(2):71–82, 2020. (Cited on page 16.)

[Vau20] Serge Vaudenay. Centralized or decentralized? The contact tracing dilemma. Cryptology ePrint Archive, Report 2020/531, 2020. `https://eprint.iacr.org/2020/531`. (Cited on page 17.)

[VMW$^+$18] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 991–1008. USENIX Association, August 2018. (Cited on page 19.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Cited on page 3.)

[ZGH07] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In Nikita Borisov and Philippe Golle, editors, *PET 2007*, volume 4776 of *LNCS*, pages 62–76. Springer, Heidelberg, June 2007. (Cited on page 17.)

[Zub18] Shoshana Zuboff. *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power.* 1st edition, 2018. (Cited on page 14.)