This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2025.3546850

Privacy-Preserving Epidemiological Modeling on Mobile Graphs

Daniel Günther*, Marco Holz*, Benjamin Judkewitz†, Hellen Möllering*, Benny Pinkas‡, Thomas Schneider*, Ajith Suresh§
*Technical University of Darmstadt, Germany †Charité-Universitätsmedizin, Germany ‡Bar-Ilan University, Israel
§Technology Innovation Institute (TII), Abu Dhabi

***Abstract*—The latest pandemic COVID-19 brought governments worldwide to use various containment measures to control its spread, such as contact tracing, social distance regulations, and curfews. Epidemiological simulations are commonly used to assess the impact of those policies before they are implemented. Unfortunately, the scarcity of relevant empirical data, specifically detailed social contact graphs, hampered their predictive accuracy. As this data is inherently privacy-critical, a method is urgently needed to perform powerful epidemiological simulations on real-world contact graphs without disclosing any sensitive information.**

In this work, we present RIPPLE, a privacy-preserving epidemiological modeling framework enabling standard models for infectious disease on a population's real contact graph while keeping all contact information locally on the participants' devices. As a building block of independent interest, we present PIR-SUM, a novel extension to private information retrieval for secure download of element sums from a database. Our protocols are supported by a proof-of-concept implementation, demonstrating a 2-week simulation over half a million participants completed in 7 minutes, with each participant communicating less than 50 KB.

***Index Terms*—epidemiological modeling, private information retrieval, secure multi-party computation**

I. Introduction

The COVID-19 pandemic has profoundly impacted daily life, leading to heightened mental illness and domestic abuse cases [1]. Governments globally have taken steps, such as lockdowns and institution closures, to curb the virus while supporting the economy. Despite these measures, infections surged, and many lives were lost. Afterwards, new infectious diseases like monkeypox have spread, resulting in quarantines in Europe [2].

In the context of COVID-19, contact tracing apps were used all over the world to notify contacts of potential infections [3]–[10]. Unfortunately, there is a fundamental limitation to contact tracing: It only notifies contacts of an infected person *after* the infection has been detected, i.e., typically after a person develops symptoms, is tested, receives the test result, and can connect with contacts [11], [12]. Tupper et al. [11] report that in British Columbia in April 2021, this process ideally took five days, reducing new cases by only 8% compared to not using contact tracing. They conclude that contact tracing must be supplemented with multiple additional containment measures to control disease spread effectively.

To overcome these inherent limitations of contract tracing, we consider epidemiological modeling, which allows predicting the spread of an infectious disease in the *future*, as done in [13]–[19]. It allows to assess the effectiveness of containment measures by mathematically modeling their impact on the spread. As a result, it can be an extremely valuable tool for governments to select effective containment measures [18]. For example, Davis et al. [20] predicted in early 2020 that COVID-19 would infect 85% of the British population without any containment measures in place, causing a massive overload of the health system (13-80× the capacity of intensive care units). Their forecast also indicated that short-term interventions, such as school closures and social distancing, would not effectively reduce the number of cases. As a result, the British government decided to implement a lockdown in March 2020, effectively reducing transmissions and stabilizing the health system [18].

Access to detailed information about a population's size, density, transportation, and health care system enables accurate epidemiological modeling to forecast disease transmission in various scenarios [21]. Precise, up-to-date data on movements and physical interactions is crucial for forecasting transmission and assessing the impact of control measures before implementation [22]. In practice, these simulations can quickly model the spread of a disease, project the number of infections based on specific actions, and predict how the disease might spread to specific areas.

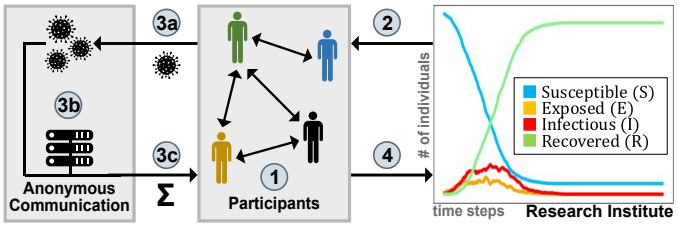
However, data on personal encounters is scarce, limiting accurate assessments of containment measures' impacts [21]. This scarcity arises because so far encounter data is often obtained through surveys, which fail to capture the reality of random encounters in public places [22]. Additionally, social interaction patterns can change rapidly, as seen with social distancing measures, making collected data quickly outdated. Therefore, existing data cannot realistically simulate person-to-person social contact graphs. Ideally, epidemiologists need a complete physical interaction graph of the population, but strict privacy regulations make accurate tracking of interpersonal contacts unacceptable.

To address the issue of preserving privacy while obtaining up-to-date contact data, we present RIPPLE, a practical framework for epidemiological modeling that allows precise disease spread simulations using current contact information, incorporating control measures without leaking information about individuals' contacts. RIPPLE provides a privacy-preserving method for collecting real-time physical encounters and can compute arbitrary compartment-based epidemiological mod-

els[1] on the latest contact graph of the previous days. RIPPLE is not only applicable to COVID-19, but to *any* infectious diseases. We expect that RIPPLE's strong privacy guarantees will encourage wider participation, enabling epidemiologists to conduct more accurate simulations and develop effective containment strategies.

Additionally, given the success and public acceptance of contact-tracing apps during COVID-19, users are familiar with the value of digital tools in reducing disease spread [23], [24]. RIPPLE builds on this trust by offering a privacy-preserving solution that can be seamlessly integrated into existing contact-tracing apps, thus expanding beyond individual contact tracing to support community-wide epidemiological modeling. Similar to traditional contact-tracing apps, RIPPLE involves only a minimal exchange of two tokens when two participants encounter each other. To minimize any impact on device performance, we propose that the final simulation of disease spread is executed at night while most participants sleep and their phones are charging. This timing ensures that RIPPLE runs unobtrusively on participants' mobile devices and preserves a smooth user experience. Our work mainly focuses on providing an initial cryptographic solution for privacy-preserving epidemiological modeling and leaves challenges for practical deployment as future work.

Our Contributions: This paper introduces RIPPLE (cf. Fig. 1), a framework for expanding the scope of privacy research from contact tracing to epidemiological modeling. While the former only warns about potential infections in the past, epidemiological modeling can predict the spread of infectious diseases in the future. Anticipating the effects of various control measures allows for the development of informed epidemic containment strategies and political interventions before their implementation.



① Mobile apps collect anonymous encounter tokens during interactions. ② Research Institute begins the simulation by providing initialization parameters. ③a Participants securely upload infection likelihood to servers. ③b Servers securely compute cumulative infection likelihood per participant. ③c Participants retrieve their cumulative infection likelihood. ④ The aggregate results (#S,#E,#I,#R) are sent to the Research Institute.

Fig. 1: Overview of RIPPLE Framework.

RIPPLE uses a fully decentralised system similar to the federated learning paradigm [25] to achieve high acceptance and trust in the system and to motivate many participants to join the system to generate representative contact information. All participant data, such as encounter location, time, and distance, are kept locally on the participants' devices. Participants in RIPPLE communicate through anonymous communication channels enabled by a group of *semi-honest* central servers.

RIPPLE is instantiated with two methods for achieving privacy-preserving epidemiological modeling, each covering a different use case. The first is RIPPLE$_{\mathsf{TEE}}$, which assumes each participant's mobile device has a Trusted Execution Environment (TEE). The second method is RIPPLE$_{\mathsf{PIR}}$, which eliminates this assumption by utilizing cryptographic primitives such as Private Information Retrieval (PIR). Along the way, we develop a multi-server PIR extension that allows a client to retrieve the sum of a set of elements (in our case, infection likelihoods) from a database without learning individual entries.

We assess the practicality of our methods by benchmarking core building blocks using a proof of concept implementation. Our findings indicate that, with adequate hardware, both protocols can scale up to millions of participants. For instance, a simulation of 14 days with 1 million participants can be completed in less than half an hour.

Our contributions are summarized as follows:

1) We present RIPPLE, the *first* privacy-preserving framework for epidemiological modeling on contact information stored on mobile devices.

2) RIPPLE formalizes the notion of *privacy-preserving* epidemiological modeling and defines privacy requirements in the presence of both semi-honest and malicious participants.

3) We present RIPPLE$_{\mathsf{PIR}}$, an instantiation of RIPPLE that combines anonymous communication techniques with PIR and anonymous credentials.[2]

4) We propose PIR-SUM, an extension to existing PIR schemes, that allows a client to download the sum of τ distinct database entries without learning the values of individual entries or revealing which entries were requested.

5) We demonstrate the practicality of our framework by providing an open source implementation and a detailed performance evaluation of RIPPLE.

II. Related Work

This section discusses related works on disease modeling and privacy-preserving epidemiological modeling.

a) Disease Modeling: There are various ways to model a disease mathematically [27]–[30]. Popular compartment models use a few continuous variables linked by differential equations to capture disease spread. In the SEIR model [28], [31], individuals are assigned to four compartments: susceptible (S), exposed (E), infectious (I), and recovered (R). These models are useful for understanding macroscopic trends and are widely used in epidemiological research [32], [33]. However, they condense complex individual behaviors into a few variables, limiting predictive power [34]. In contrast, agent-based epidemiological models [35] simulate the spread by initializing numerous agents with individual properties (e.g., location, age) and interaction rules. This allows for more realistic disease transmission simulations by modeling

[1] The implementation of concrete simulation functions is outside the scope of this work and referred to medical experts. More details on epidemiological modeling are given in §II.

[2] A TEE-based technique is presented in the full version [26].

individual behaviors. Combining agent-based models with compartment models enhances the realism and accuracy of disease forecasting. Simulations with varying parameters, like interaction reductions or targeted vaccinations, are run to predict the effects of different policy interventions.

A key challenge is modeling agents' contact behaviors. Older models used survey-based contact matrices to estimate average contacts within age ranges [22], which improved over uniform assumptions but still fell short. Aggregated network statistics can't replicate real network dynamics, including super-spreaders with numerous contacts [36]. Ideally, epidemiologists would like to use real-world contact graphs of all individuals, but this is often challenging due to privacy concerns.

b) Contact Tracing for Privacy-Preserving Epidemiological Modeling: If contact information collected through contact tracing apps was centralized, an up-to-date full contact graph could be constructed for epidemiological simulations [29], [30]. However, contact information is highly sensitive and should not be shared. Contact information collected via mobile phones can reveal who, when, and whom people meet, which is sensitive and must be protected. Beyond, such information also enables to derive indications about the financial situation [37] and personality [38]. One can think about many more examples: By knowing which medical experts are visited by a person, information about the health condition can be anticipated; contact with members of a religious minority as well as visits to places related to religion might reveal a religious orientation, etc. Thus, it would be ideal for enabling precise epidemiological simulations without leaking individual contact information.

One way to achieve privacy-preserving epidemiological modeling using contact tracing apps is by allowing each participant's device to share its contact information secretly with a set of non-colluding servers. These servers can then run simulations using secure multi-party computation (MPC). Araki et al. [39] demonstrated efficiently running graph algorithms on secret shared graphs via MPC. However, despite the common non-collusion assumption in the crypto community, public trust issues may arise if all contact information is disclosed once servers collude. To address this, RIPPLE distributes trust by enabling participants to keep their contact information local while anonymously sending messages to each other to simulate the disease spread. Only aggregated simulation results are shared with research institutes, ensuring no direct identity or contact data is disclosed. This method resembles Federated Learning [25] and the contact tracing designs by Apple and Google.[3] This distributed design can increase trust and facilitate broad adoption of the system.

To the best of our knowledge, RIPPLE is the first framework that allows executing any agent-based compartment model on the distributed real contact graph while maintaining privacy.

III. The RIPPLE Framework

RIPPLE's primary goal is to enable the evaluation of the impact of multiple combinations of potential containment measures defined by epidemiologists and the government, and to find a balance between the drawbacks and benefits of those measures, rather than to deploy the measures in "real-life" first and then analyse the impact afterwards. Such measures may include, for example, the requirement to wear face masks in public places, restrictions on the number of people allowed to congregate, the closure of specific institutions and stores, or even complete curfews and lockdowns within specific regions.

Participants in RIPPLE collect personal encounter data anonymously and store it locally on their mobile devices, similar to privacy-preserving contact tracing apps. For epidemiological modeling, RIPPLE must derive a contact graph without leaking sensitive personal information to simulate disease spread over a specific period, like two weeks. Most countries have a 6-hour period at night when most people are asleep, and their mobile devices are idle, connected to WiFi, and possibly charging—an ideal time for running RIPPLE simulations. Medical experts can then analyze the results to understand the disease better, and political decision-makers can identify the most effective containment measures to implement.

To acquire representative and up-to-date physical encounter data, widespread public usage of RIPPLE would be ideal. One way to encourage this is to piggyback RIPPLE on most countries' official contact tracing applications. On the other hand, politicians can motivate residents beyond the intrinsic incentive of supporting public health by coupling the use of RIPPLE with additional benefits such as discounted or free travel passes.

A. System and Threat Model

RIPPLE comprises of p participants, denoted collectively by $\mathcal{P}$, a research institute RI who is in charge of the epidemiological simulations, and a set of MPC servers $\mathcal{C}$ responsible for anonymous communication among the participants. Tab. I summarizes the notations used in this work.

	Parameter	Description
Entities	$\mathcal{P}$	Set of all participants; $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_\mathsf{p}\}$
	RI	Research Institute
	$\mathcal{C}$	Communication Servers $\{\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2\}$
Simulations	$\mathsf{param}_\mathsf{sim}$	Simulation parameters defined by RI
	N_sim	# distinct simulations (executed in parallel)
	N_step	# steps per simulation
	$\mathsf{class}_\mathsf{inf}$	Infection classes; $\mathsf{class}_\mathsf{inf} = \{\mathsf{class}_\mathsf{inf}^1, \ldots, \mathsf{class}_\mathsf{inf}^{\mathsf{N}_\mathsf{inf}}\}$
	I_i^s	$\mathcal{P}_i$'s infection class in simulation step $\mathsf{s} \in [0, N_\mathsf{step}]$
	$\mathcal{E}_i$	Encounter tokens of $\mathcal{P}_i$
	E_i^max	# max. encounters by $\mathcal{P}_i$ in pre-defined time interval
	E^avg	Average number of encounters
Protocols	κ	Computational security parameter $\kappa = 128$
	r_e	Unique token for encounter $e \in [0, E^\mathsf{max}]$
	$\delta_i^{r_e}$	$\mathcal{P}_i$'s infection likelihood w.r.t token r_e
	Δ_i	$\mathcal{P}_i$'s cumulative infection likelihood
	m_i^e	Metadata of an encounter e by $\mathcal{P}_i$
	$(\mathrm{pk}_i, \mathrm{sk}_i)$	$\mathcal{P}_i$'s public/private key pair
	σ_i^e.	$\mathcal{P}_i$'s signature on message about encounter e

TABLE I: Notations.

We assume that the research institute and MPC servers are semi-honest [40], meaning they follow protocol specifications correctly while attempting to gather additional information. These semi-honest MPC servers also establish an anonymous communication channel. A protocol is secure if nothing is

[3] https://covid19.apple.com/contacttracing

leaked beyond what can be inferred from the output. While the semi-honest security model is not the strongest, it offers a good trade-off between privacy and efficiency, making it popular in practical privacy-preserving applications such as privacy-preserving machine learning [41]–[43] and genome/medical research [44], [45]. This model protects against passive attacks by curious administrators and accidental data leakage and often serves as a foundation for developing protocols with stronger privacy guarantees [46], [47]. We consider this a reasonable assumption, as the research institute and servers will be controlled by generally trusted entities such as governments or public medical research centers, potentially collaborating with NGOs like the EFF[4] or the CCC[5].

Given the widespread interest in discovering effective containment measures, we expect a high level of intrinsic motivation among participants for successful epidemiological modeling. However, we cannot assume that millions of potential participants are behaving honestly. Therefore, our set of participants $\mathcal{P}$ can include clients who are malicious and may attempt to compromise the privacy of the scheme. This includes scenarios where some clients deviate from the protocol to gain unauthorized information. This stronger security notion, referred to as *malicious-privacy*, has been the focus of several recent works [48], [49] and provides a higher level of privacy than semi-honest corruptions.

Our primary focus is to protect against malicious clients who want to gain unauthorized information. A stronger threat involves malicious clients undermining the correctness of the protocol, thereby disrupting or compromising the accuracy of the simulation. Addressing the issue of malicious clients targeting correctness is left as future work. Such a correct behaviour could for example be enforced using trusted execution environments like ARM Trust Zone available in many smartphones.

B. Phases of RIPPLE

RIPPLE is divided into four phases as shown in Fig. 1: i) Token Generation, ii) Simulation Initialization, iii) Simulation Execution, and iv) Result Aggregation. While our framework can be applied to any compartment-based epidemiological modeling of any infectious disease (cf. §II), we explain RIPPLE using the prevalent Covid-19 virus and the SEIR model [31], [50] as a running example. For simplicity, we assume that an app that emulates RIPPLE is installed on each participant's mobile device and that the participants locally enter attributes such as workplace, school, regular eateries, and cafes in the app after installing the app.

Fig. 2 summarises the phases of the RIPPLE framework in the context of a single simulation setting and we give details below. Multiple simulations can be executed in parallel. The concrete number of simulation runs with the same parameters or different parameters should be determined by epidemiologists. Note that simulations are run on collected data, e.g., from the last days, and not on real-time encounter information. This combines efficiency requirements with maximally up-to-date encounter information.

[4] https://www.eff.org [5] https://www.ccc.de/en/

① - Token Generation: During a physical encounter, participants exchange data via Bluetooth LE to collect anonymous encounter information, similar to contact tracing [8], [51]. These tokens are stored locally on the users' devices and do not reveal any sensitive information (i.e., identifying information) about the individuals involved. In addition to these tokens, the underlying application will collect additional information on the context of the encounter, known as "metadata" for simulation purposes. This varies depending on the underlying instantiation of the protocol and can include details such as duration, proximity, time, and location. The metadata can include or exclude different encounters in the simulation phase, allowing the effect of containment measures to be modeled (e.g., restaurant closings by excluding all encounters that happened in restaurants). The token generation phase is not dependent on the simulation phase, so no simulation-dependent infection data is exchanged. The token generation phase is modelled as an ideal functionality $\mathcal{F}_{\mathsf{gen}}$ that will be instantiated later in §IV.

Protocol RIPPLE

① - Token Generation

- $\mathcal{P}_i \in \mathcal{P}$ executes $\mathcal{F}_{\mathsf{gen}}$ all the time (on its mobile device), collecting encounter data of the form (r_e, m_e) with $e < E_i^{\mathsf{max}}$. For an encounter e, r_e is the token and m_e is the metadata.

② - Simulation Initialization

- $\mathcal{P}_i \in \mathcal{P}$ gets $\mathsf{param}_{\mathsf{sim}}$ from RI and sets $I_i^1 = I_i^{\mathsf{init}}$.

③ - Simulation Execution

For each simulation step $\mathsf{s} \in [N_{\mathsf{step}}]$, $\mathcal{P}_i \in \mathcal{P}$ execute:

- Filter encounters using $\mathsf{param}_{\mathsf{sim}}$ to obtain encounter set $\mathcal{E}_i^{\mathsf{s}}$.
- For each token $r_e \in \mathcal{E}_i^{\mathsf{s}}$, compute the infection likelihood $\delta_i^{r_e}$ locally using the formula from RI.
- Invoke $\mathcal{F}_{\mathsf{esim}}$ with the input $\{\delta_i^{r_e}\}_{r_e \in \mathcal{E}_i^{\mathsf{s}}}$ and obtain $\Delta_i^{\mathsf{s}} = \sum_{r_e \in \mathcal{E}_i^{\mathsf{s}}} \hat{\delta}_i^{r_e}$.
- Update the infection class I_i^{s} using Δ_i^{s} and guidelines from RI.

④ - Result Aggregation

For each simulation step $\mathsf{s} \in [N_{\mathsf{step}}]$, execute:

- $\mathcal{P}_i \in \mathcal{P}$ prepares $\{\mathsf{v}_\mathsf{i}^1, \ldots, \mathsf{v}_\mathsf{i}^{\mathsf{N_{inf}}}\}^{\mathsf{s}}$ with $\mathsf{v}_\mathsf{i}^{\mathsf{k}} = 1$ if $I_i^{\mathsf{s}} = \mathsf{class}_{\mathsf{inf}}^{\mathsf{k}}$ and $\mathsf{v}_\mathsf{i}^{\mathsf{k}} = 0$ otherwise, for $k \in [N_{\mathsf{inf}}]$.
- Invoke $\mathcal{F}_{\mathsf{agg}}$ with inputs $\{\mathsf{v}_\mathsf{i}^1, \ldots, \mathsf{v}_\mathsf{i}^{\mathsf{N_{inf}}}\}^{\mathsf{s}}$ to enable RI obtain the tuple $\{\mathsf{C}_{\mathsf{inf}}^1, \ldots, \mathsf{C}_{\mathsf{inf}}^{\mathsf{N_{inf}}}\}^{\mathsf{s}}$, where $\mathsf{C}_{\mathsf{inf}}^{\mathsf{k}} = \sum_{\mathcal{P}_i \in \mathcal{P}} \mathsf{v}_\mathsf{i}^{\mathsf{k}}$ for $k \in [N_{\mathsf{inf}}]$.

Fig. 2: RIPPLE Framework (for one simulation setting).

② - Simulation Initialization: The research institute RI initiates the simulation phase by sending a set of parameters, denoted by $\mathsf{param}_{\mathsf{sim}}$, to the participants in $\mathcal{P}$. The goal is to "spread" a fictitious infection across N_{sim} different simulation settings. To begin a simulation, each participant $\mathcal{P}_i$ is assigned to an infection class $I_i^{\mathsf{init}} \in \mathsf{class}_{\mathsf{inf}}$ (e.g., {S}usceptible, {E}xposed, {I}nfectious, {R}ecovered for the SEIR model) as specified in $\mathsf{param}_{\mathsf{sim}}$. For each individual simulation, $\mathsf{param}_{\mathsf{sim}}$ defines a set of containment measures, such as school closings and work from home, which the participants will use as filters to carry out the simulation in the

next stage. In addition, RI publishes a formula to calculate the infection likelihood δ. The likelihood is determined by several parameters in the underlying modeling, such as encounter distance and time. For example, this likelihood might range from 0 (no chance of infection) to 100 (certain to get infected).

③ **- Simulation Execution:** Once RI initialises the simulation, N_{step} simulation steps (steps ③a, ③b, ③c in Fig. 1) are performed for each of the N_{sim} simulation settings (e.g., $N_{\mathsf{step}} = 14$ days). Without loss of generality, consider the first simulation step and let $N_{\mathsf{sim}} = 1$. The simulation proceeds as follows:

1) Participant $\mathcal{P}_i \in \mathcal{P}$ filters out the relevant encounters based on the containment measures defined by RI. Let the set $\mathcal{E}_i$ represent the corresponding encounter tokens.

2) For each token $r_e \in \mathcal{E}_i$, $\mathcal{P}_i$ computes the infection likelihood $\delta_i^{r_e}$ using the formula from RI, i.e., the probability that $\mathcal{P}_i$ infects the participant met during the encounter with identifier token r_e.

3) Participants use the likelihood values δ obtained in the previous step to execute an ideal functionality called $\mathcal{F}_{\mathsf{esim}}$, which allows them to communicate the δ values anonymously through a set of MPC servers $\mathcal{C}$. Furthermore, it allows each participant $\mathcal{P}_j$ to receive a cumulative infection likelihood, denoted by Δ_j, based on all of the encounters they had on the day being simulated, i.e., $\Delta_j = \sum_{r_e \in \mathcal{E}_j} \hat{\delta}_j^{r_e}$. In this case, $\hat{\delta}_j^{r_e}$ denotes the infection likelihood computed by participant $\mathcal{P}_f$ and communicated to $\mathcal{P}_j$ for an encounter between $\mathcal{P}_f$ and $\mathcal{P}_j$ with identifier token r_e. As will be discussed later in §III-C, $\mathcal{F}_{\mathsf{esim}}$ must output the cumulative result rather than individual infection likelihoods because the latter can result in a breach of privacy.

4) Following the guidelines set by the RI, $\mathcal{P}_j$ updates its infection class I_j using the cumulative infection likelihood Δ_j acquired in the previous step.

These steps above are repeated for each of the N_{step} simulation steps in order and across all the N_{sim} simulation settings.

④ **- Result Aggregation:** For a given simulation setting, each participant $\mathcal{P}_i \in \mathcal{P}$ will have its infection class I_i^{s} updated at the end of every simulation step $\mathsf{s} \in [N_{\mathsf{step}}]$. This phase allows RI to obtain each simulated time step's aggregated number of participants per class (e.g., #S, #E, #I, #R). For this, we rely on a *Secure Aggregation* functionality, denoted by $\mathcal{F}_{\mathsf{agg}}$, which takes a N_{inf}-tuple of the form $\{\mathsf{v}_\mathsf{i}^1, \ldots, \mathsf{v}_\mathsf{i}^{\mathsf{N_{inf}}}\}^{\mathsf{s}}$ from each participant for every simulation step s and outputs the aggregate of this tuple over all the p participants to RI. In this case, $\mathsf{v}_\mathsf{i}^{\mathsf{k}}$ is an indicator variable for the k-th infection class, which is set to one if $I_i^{\mathsf{s}} = \mathsf{class}_{\mathsf{inf}}^{\mathsf{k}}$ and zero otherwise. Secure aggregation [52]–[54], [54] is a common problem in cryptography these days, particularly in the context of federated learning, and there are numerous solutions proposed for various settings, such as using TEEs, a semi-trusted server aggregating ciphertexts under homomorphic encryption, or multiple non-colluding servers that aggregate secret shares. In this work, we consider $\mathcal{F}_{\mathsf{agg}}$ a black box that can be instantiated using existing solutions compatible with our framework.

C. Privacy Requirements

A private contact graph necessitates that participants remain unaware of any unconscious interactions. This means they cannot determine if they had unconscious contact with the same person more than once or how often they did, which is considered a privacy issue, exemplified in the next paragraph. We remark that an insecure variant of RIPPLE, in which each participant $\mathcal{P}_i$ receives the infection likelihood $\hat{\delta}_i^e$ for all its encounters $e \in E_i$ separately, will not meet this condition.

a) **Linking Identities Attacks:** To demonstrate this, observe that when running multiple simulations (with different simulation parameters $\mathsf{param}_{\mathsf{sim}}$) on the same day, participants will use the same encounter tokens and metadata from the token generation phase in each simulation. If a participant $\mathcal{P}_i$ (Alice) can see the infection likelihood $\hat{\delta}_i$ of each of her interactions, $\mathcal{P}_i$ can look for correlations between those likelihoods to see if another participant $\mathcal{P}_j$ (Bob) was encountered more than once. We call this a *Linking Identities Attack* and depict it in Fig. 3, where, for simplicity, the infection likelihood accepts just two values: 1 for high and 0 for low infection likelihood.

Consider the following scenario to help clarify the issue: Alice and Bob work together in the same office. As a result, they have numerous conscious encounters during working hours. However, in their spare time, they may be unaware that they are in the same location (e.g., a club) and may not want the other to know. Their phones constantly collect encounters even if they do not see each other. Assume the research institute (RI) sent the participants a simple infection likelihood formula that returns 0 (not infected) or 1 (infected). Furthermore, since the data is symmetric, Alice and Bob have the same metadata (duration, distance, etc.) about their conscious and unconscious encounters. Let Bob be modelled as infectious in the first simulation. As a result, he will send a 1 for each (conscious and unconscious) encounter he had (including those with Alice). If multiple simulations are run on the same day (i.e., with the same encounters), Alice will notice that some encounters, specifically all conscious and unconscious encounters with Bob, always have the same infection likelihood: If Bob is not infectious, all will return a 0; if Bob is infectious, all will return a 1. Thus, even if Alice had unconscious encounters with Bob, she can detect the correlations between the encounters and, as a result, determine which unconscious encounters were most likely with Bob.

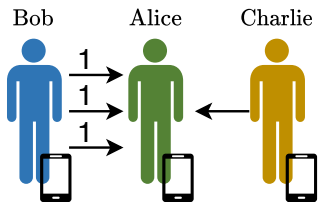


Fig. 3: Linking Identities Attack. Alice and Bob had several encounters, but Alice and Charlie only had one.

The more simulations she runs, the more confident she becomes. Since every participant knows the formula, this attack can also be extended to complex infection likelihood functions. While it may be more computationally expensive than the simple case, Alice can still identify correlations. This attack works even if all of the encounters are unconscious. In such situations, Alice may be unable to trace related encounters to

a single person (Bob), but she can infer that they were all with the same person (which is more than learning nothing).

To avoid a Linking Identities attack, RIPPLE ensures that each participant receives an aggregation of all infection likelihoods of their encounters during the simulation. It cannot be avoided that participants understand that when "getting infected" some of their contacts must have been in contact with a (simulated) infectious participant. As this is only a simulated infection, we consider this leakage acceptable.

In summary, the Linking Identities attack demonstrates that repeated simulations with identical metadata (tokens) enable participants to infer connections between conscious and unconscious encounters. In RIPPLE, participants never learn individual infection likelihoods, but only an aggregated infection likelihood, which highly decreases the practicality of this attack.

b) **Sybil Attack:** While the Linking Identities Attack is already significant in the semi-honest security model, malicious participants may further circumvent aggregation mechanisms that prevent access to individual infection likelihoods. They could, for example, construct many *sybils*, i.e., multiple identities using several mobile devices, to collect each encounter one by one and then conduct a Linking Identities Attack with the information.

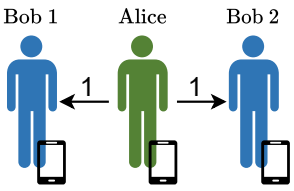


Fig. 4: Sybil Attack.

Apart from the privacy threat, the sybil attack can affect the accuracy of the simulation as potentially more devices than people in one location will participate in the simulation, which does not correspond to reality. If a malicious participant who exploits a sybil attack is simulated as infectious, their visited locations will tend to spread the simulated infection more significantly and thus classify the risk of infection as higher than usual.

A registration system can be used to increase the costs of performing sybil attacks, i.e., to prevent an adversary from creating many identities. This assures that only legitimate users can join and participate in the simulation. In a closed ecosystem, such as a company, this can be achieved by letting each member receive exactly one token to participate in the simulation. On a larger scale at the national level, one can let each citizen receive a token linked to a digital ID card. In such authentication mechanisms, anonymous credentials can be used to ensure anonymity, and we leave the problem for future work, as our initial work on privacy-preserving epidemiological modeling focuses on malicious clients targeting privacy, but not yet correctness (cf. §III-A).

To summarize, from a privacy perspective, Sybil attacks enable a Linking Identities Attack by allowing malicious participants to create multiple identities. Furthermore, they have an impact on the accuracy of the simulation.

c) **Inference Attacks:** Note that although RIPPLE mimics the spirit of Federated Learning (FL) [25], it is not susceptible to so-called inference attacks [55], [56] in the same sense as FL. First, RIPPLE only reveals the final output (to a research institute RI) and no individual updates/results that ease information extraction. We, however, note that the analysis results provided to RI (cf. §III-A) contain information about the spread of the modeled disease in a specific population (otherwise, it would be meaningless to run the simulation). The ideal functionality does not cover leakage from the final output but protects privacy during the computation. Thus, our security model does not consider anything that might be inferred from the output. We also argue that it is in the public interest to provide such aggregated information to the RI for deciding upon effective containment measures against infectious diseases.

In summary, while RIPPLE mitigates inference attacks by restricting individual data leakage, the final aggregated output may reveal trends relevant to public health, balancing privacy with the societal need for actionable insights.

IV. Instantiating $\mathcal{F}_{\mathsf{esim}}$

We propose two instantiations of $\mathcal{F}_{\mathsf{esim}}$ that cover different use cases and offer different trust-efficiency trade-offs. Our first design, RIPPLE$_{\mathsf{TEE}}$, is presented in the full version [26, §4.1] and assumes the presence of trusted execution environments (TEEs) such as ARM TrustZone on the mobile devices of the participants. In our second design, RIPPLE$_{\mathsf{PIR}}$ (§IV-A), we eliminate this assumption and provide privacy guarantees using cryptographic techniques such as PIR and anonymous communications.

A. *RIPPLE*$_{\mathsf{PIR}}$

The main challenge for instantiating $\mathcal{F}_{\mathsf{esim}}$ is to prevent the leakage of individual infection likelihoods of the encounters. To get around this privacy issue, we need to find a way to aggregate the infection likelihoods so that the participants can only derive the sum, not individual values.

Functionality $\mathcal{F}_{\mathsf{pirsum}}$

$\mathcal{F}_{\mathsf{pirsum}}$ interacts with M servers, denoted by $\mathcal{C}$, and participant $\mathcal{P}_i \in \mathcal{P}$.

Input: $\mathcal{F}_{\mathsf{pirsum}}$ receives τ indices denoted by $\mathcal{Q} = \{q_1, \ldots, q_\tau\}$ from $\mathcal{P}_i$ and a database D from $\mathcal{C}$.

Output: $\mathcal{F}_{\mathsf{pirsum}}$ sends $\sum_{j=1}^{\tau} \mathsf{D}[q_j]$ to $\mathcal{P}_i$ as the output.

Fig. 5: Ideal functionality for PIR-SUM (semi-honest).

Private Information Retrieval (PIR) is one promising solution for allowing participants to retrieve infection likelihoods sent to them anonymously. PIR enables the private download of an item from a public database D held by M servers without leaking any information to the servers, such as which item is queried or the content of the queried item. However, classical PIR is unsuitable for our needs because we need to retrieve the sum of τ items from the database rather than the individual ones. As a result, we introduce the ideal functionality $\mathcal{F}_{\mathsf{pirsum}}$ (Fig. 5), which is similar to a conventional PIR functionality but returns the sum of τ queried locations of the database as a result. For the remainder of this section, we consider $\mathcal{F}_{\mathsf{pirsum}}$ to be an ideal black-box and will discuss concrete instantiations in §V.

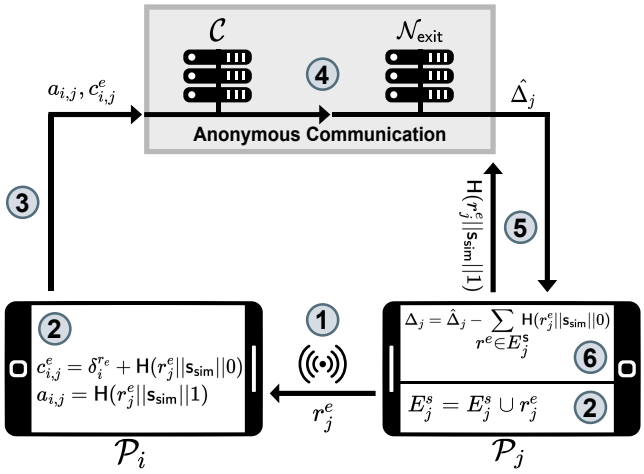


Fig. 6: RIPPLE$_{\mathsf{PIR}}$ Overview.

We now detail the protocol steps of RIPPLE$_{\mathsf{PIR}}$ with respect to the overview provided in Fig. 6.

Token Generation (step ① in Fig. 6): During a physical encounter e among participants $\mathcal{P}_i$ and $\mathcal{P}_j$, they generate and exchange unique random tokens denoted by r^e_i and r^e_j. Simultaneously, both participants compute and record metadata m_e, such as the time, location, and duration of the encounter, and store this information. Thus, at the end of a simulation step $\mathsf{s} \in [N_{\mathsf{step}}]$ (e.g., a day), $\mathcal{P}_i$ holds a list of sent encounter tokens, denoted by $E^{\mathsf{s}}_i = \{r^e_i\}_{e \in \mathcal{E}_i}$, where $\mathcal{E}_i$ is the complete (sent/received) set of encounters of $\mathcal{P}_i$, and a list of received tokens, denoted by $R^{\mathsf{s}}_i = \{r^e_j\}_{e \in \mathcal{E}_i}$. Looking ahead, these random tokens will be used as addresses to communicate the corresponding infection likelihood among the participants.

Simulation Execution (steps ② to ⑥ in Fig. 6):

- $\mathcal{P}_i$ blinds each infection likelihood $\delta^{r_e}_i$ computed with the corresponding random token r^e_j received from $\mathcal{P}_j$ and obtains the ciphertext $c^e_{i,j} = \delta^{r_e}_i + \mathsf{H}(r^e_j||\mathsf{s}_{\text{sim}}||0)$. In addition, it computes the destination address for the ciphertext as $a_{i,j} = \mathsf{H}(r^e_j||\mathsf{s}_{\text{sim}}||1)$. Here, $\mathsf{H}()$ is a cryptographic hash function and $\mathsf{s}_{\text{sim}} \in [N_{\mathsf{sim}}]$ denotes the current simulation setting. (step ② in Fig. 6)
  - s_{sim} is used in $\mathsf{H}()$ to ensure that distinct (ciphertext, address) tuples are generated for the same encounters across multiple simulation settings, preventing the exit node $\mathcal{N}_{\texttt{exit}}$ from potentially linking messages from different simulations.
- $\mathcal{P}_i$ sends the tuple $(c^e_{i,j}, a_{i,j})$ anonymously to $\mathcal{N}_{\texttt{exit}}$ with the help of the servers in $\mathcal{C}$. $\mathcal{N}_{\texttt{exit}}$ discards all the tuples with the same address field $(a_{i,j})$ (steps ③ to ④ in Fig. 6). The instantiation details for the anonymous communication channel are given in the full version [26, §B.3]).

As a server in $\mathcal{C}$, $\mathcal{N}_{\texttt{exit}}$ locally creates the database D for the current simulation step using all of the $(a_{i,j}, c^e_{i,j})$ tuples received (part of step ④ in Fig. 6). A naive solution of inserting $c^e_{i,j}$ using a simple hashing of the address $a_{i,j}$ will not provide an efficient solution in our case since we require only one message to be stored in each database entry to have an injective mapping between addresses and messages. This is required for the receiver to download the messages sent to them precisely. Simple hashing would translate to a large database size to ensure a negligible probability of collisions. Instead, in RIPPLE$_{\mathsf{PIR}}$, we use a novel variant of a garbled cuckoo table that we call arithmetic garbled cuckoo table (AGCT, see below), with $a_{i,j}$ as the insertion key for the database.

Once the database D is created, $\mathcal{N}_{\texttt{exit}}$ sends it to the other servers in $\mathcal{C}$ based on the instantiation of $\mathcal{F}_{\mathsf{pirsum}}$ (cf. §V). Each $\mathcal{P}_j \in \mathcal{P}$ will then participate in an instance of $\mathcal{F}_{\mathsf{pirsum}}$ with the servers in $\mathcal{C}$ acting as PIR servers holding the database D. $\mathcal{P}_j$ uses the addresses of all its sent encounters from E^{s}_j, namely $\mathsf{H}(r^e||\mathsf{s}_{\text{sim}}||1)$, as the input to $\mathcal{F}_{\mathsf{pirsum}}$ and obtains a blinded version of the cumulative infection likelihood, denoted by $\hat{\Delta}_j$, as the output (step ⑤ in Fig. 6). The cumulative infection likelihood, Δ_j, is then unblinded as

$$\Delta_j = \hat{\Delta}_j - \sum_{r^e \in\ E^{\mathsf{s}}_j} \mathsf{H}(r^e||\mathsf{s}_{\text{sim}}||0)$$

concluding the current simulation step (step ⑥ in Fig. 6).

1) Security of RIPPLE$_{\mathsf{PIR}}$: For semi-honest participants, the $\mathcal{F}_{\mathsf{anon}}$ functionality, which implements an anonymous communication channel utilising the servers in $\mathcal{C}$, aids in achieving *contact graph privacy* by preventing participants from learning to/from whom they are sending/receiving messages. While the entry node learns who sends a message, it does not learn who receives it. Similarly, the exit node cannot identify the message's destination from the address because it will be known only to the receiving participant. Furthermore, each participant obtains the cumulative infection likelihood directly via the $\mathcal{F}_{\mathsf{pirsum}}$ functionality, ensuring that $\mathcal{N}_{\texttt{exit}}$ cannot infer the participant's encounter details and, thus, *contact graph privacy* is achieved.

Malicious participants can tamper with the protocol's correctness by providing incorrect inputs. However, as stated in the threat model in §III, we assume that malicious participants in our framework will not tamper with the correctness and will only seek additional information. A malicious participant could re-use the same encounter token for multiple encounters during token generation, causing the protocol to generate multiple tuples with the same address. However, as the protocol states, $\mathcal{N}_{\texttt{exit}}$ will discard all such tuples, removing the malicious participant from the system. Another potential information leakage caused by the participant's aforementioned action is that the entry point of the anonymous communication channel can deduce that multiple participants encountered the same participant. This is not an issue in our protocol because we instantiate the $\mathcal{F}_{\mathsf{anon}}$ functionality using a 3-server oblivious shuffling scheme, where all the servers except $\mathcal{N}_{\texttt{exit}}$ will not see any messages in the clear, but only see secret shares.

2) Arithmetic Garbled Cuckoo Table (AGCT): We design a variant of garbled cuckoo tables [57] that we term arithmetic garbled cuckoo table (AGCT) to reduce the size of the PIR database while ensuring a negligible collision likelihood. It uses arithmetic sharing instead of XOR-sharing to share database entries and present the details next.

Assume two key-message pairs $\{k_1, m_1\}$ and $\{k_2, m_2\}$[6]

[6] k corresponds to a key and m to a message in our application.

are to be added to database D with N bins, using two hash function H_1 and H_2 to determine the insertion addresses.

1. Insertion of $\{k_1, m_1\}$:

   a) Compute $a_1 = \mathsf{H}_1(k_1) \mod N$ and $a_2 = \mathsf{H}_2(k_1) \mod N$.

   b) Check if bins a_1 and a_2 are already occupied. Let's assume this is not the case.

   c) Compute the arithmetic sharing of the message m_1: $\langle m_1 \rangle_0 = r_1 \in_R \mathbb{Z}_{2^\ell}$ and $\langle m_1 \rangle_1 = m_1 - \langle m_1 \rangle_0$.

   d) Insert $D[a_1] = \langle m_1 \rangle_0$ and $D[a_2] = \langle m_1 \rangle_1$.

2. Insertion of $\{k_2, m_2\}$:

   a) Compute $b_1 = \mathsf{H}_1(k_2) \mod N$ and $b_2 = \mathsf{H}_2(k_2) \mod N$.

   b) Check if bins b_1 and b_2 are already occupied. Let's assume $b_1 = a_1$, i.e., the first bin is already occupied, but bin b_2 is free.

   c) Compute the arithmetic sharing m_2 with $\langle m_2 \rangle_0 = \langle m_1 \rangle_0$ as $b_1 = a_1$. Then, the other share is $\langle m_2 \rangle_1 = m_2 - \langle m_2 \rangle_0$.

   d) Insert $D[b_1] = \langle m_2 \rangle_0$ and $D[b_2] = \langle m_2 \rangle_1$.

Double Collision: Now the question is how to handle the insertion of a database entry if both addresses determined by the two hash functions are already occupied. An easy solution is to pick different hash functions s.t. no double collision occurs for all n elements that shall be stored in the database. Alternatively, Pinkas et al. [57] demonstrate for a garbled cuckoo table how to extend the database by $d + \lambda$ bins, where d is the upper bound of double collisions and λ is an error parameter, such that double collisions occur with a negligible likelihood. For details, please refer to [57, §5].

V. PIR-SUM: Instantiating $\mathcal{F}_{\mathsf{pirsum}}$

In this section, we will use three semi-honest MPC servers to instantiate our novel $\mathcal{F}_{\mathsf{pirsum}}$ functionality (Fig. 5). In particular, we have three servers $\mathsf{S}_0, \mathsf{S}_1$, and S_2, and we design the $\mathsf{PIR}_{\mathsf{sum}}$ protocol to instantiate the $\mathcal{F}_{\mathsf{pirsum}}$ functionality.

The problem statement in our context is formally defined as follows: Participant $\mathcal{P}_i \in \mathcal{P}$ has a set of τ indices denoted by $\mathcal{Q} = \{q_1, \ldots, q_\tau\}$ and wants to retrieve $\mathsf{res} = \sum_{q \in \mathcal{Q}} \mathsf{D}[q]$. In this case, D is a database with N elements of ℓ-bits each that is held in the clear by both the servers S_1 and S_2. The server S_0 aids in the computation performed by the servers S_1 and S_2. Furthermore, we assume a one-time setup among the servers and $\mathcal{P}_i$ that establishes shared pseudorandom keys among them to facilitate non-interactive generation of random values and, thus, save communication [39], [41], [43].

A. Overview of $\mathsf{PIR}_{\mathsf{sum}}$ protocol

At a high level, the idea is to use multiple instances of a standard 2-server PIR functionality [58], [59], denoted by $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$, and combine the responses to get the sum of the desired blocks as the output. $\mathsf{D}^m = \mathsf{D} + m$ denotes a modified version of the database D in which every block is summed with the same ℓ-bit mask m, i.e., $\mathsf{D}^m[i] = \mathsf{D}[i] + m$ for $i \in [N]$. The protocol proceeds as follows:

- S_1 and S_2 non-interactively sample τ random mask values $\{m_1, \ldots, m_\tau\}$ such that $\sum_{j=1}^{\tau} m_j = 0$.
- $\mathsf{S}_1, \mathsf{S}_2$, and $\mathcal{P}_i$ execute τ instances of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ in parallel, with servers using D^{m_j} as the database and $\mathcal{P}_i$ using q_j as the query for the j-th instance for $j \in [\tau]$.
- Let res_j denote the result obtained by $\mathcal{P}_i$ from the j-th $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ instance. $\mathcal{P}_i$ locally computes $\sum_{j=1}^{\tau} \mathsf{res}_j$ to obtain the desired result.

The details for instantiating $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ using the standard linear summation PIR approach [58] are provided in the full version [26, §C]. The approach requires $\mathcal{P}_i$ to communicate $N \cdot \tau$ bits to the servers, which is further reduced in $\mathsf{RIPPLE}_{\mathsf{PIR}}$ (cf. §V-C).

a) Malicious participants: A malicious participant, for example, could use the same query, say q_j, in all τ instances and retrieve only the block corresponding to q_j by dividing the result by τ. We present a simple verification scheme over the $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ functionality to prevent these manipulations.

For malicious participants, we want to ensure that $\mathcal{P}_i$ used a distinct vector $\vec{\mathsf{b}}$ during the τ parallel instances. One naive approach is to have S_1 and S_2 compute the bitwise-OR of all the τ bit query vectors $\vec{\mathsf{b}}_1, \ldots, \vec{\mathsf{b}}_\tau$, and then run a secure two-party computation protocol to compare the number of ones in the resultant vector to τ. We use the additional server S_0 to optimize this step further. S_1 and S_2 send randomly shuffled versions of their secret shared bit vectors to S_0, who reconstructs the shuffled vectors and performs the verification locally. This approach leaks no information to S_0 because it has no information about the underlying database D. The verification procedure is as follows:

- S_1 and S_2 non-interactively agree on a random permutation, denoted by π.
- S_u sends $\pi([\vec{\mathsf{b}}_j]_u)$ to S_0, $j \in [\tau]$, $u \in \{1, 2\}$.
- S_0 locally reconstructs $\pi(\vec{\mathsf{b}}_j) = \pi([\vec{\mathsf{b}}_j]_1) \oplus \pi([\vec{\mathsf{b}}_j]_2)$, for $j \in [\tau]$. If all the τ bit vectors are correctly formed and distinct, it sends `Accept` to S_1 and S_2. Else, it sends `abort`.

Note that the verification using $\mathcal{P}_0$ will incur a communication of $2\tau N$ bits among the servers. Furthermore, the above verification method can be applied to any instantiation of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ that generates a boolean sharing of the query bit vector among the PIR servers and computes the response as described above, e.g., the PIR schemes of [58]–[60].

B. Instantiating $\mathcal{F}_{\mathsf{pirsum}}$

The formal protocol for $\mathsf{PIR}_{\mathsf{sum}}$ in the case of malicious participants is provided in Fig. 7, assuming the presence of an ideal functionality $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ (as will be discussed in $\mathbf{HYB}_2$ below). In $\mathsf{PIR}_{\mathsf{sum}}$, the servers $\mathsf{S}_1, \mathsf{S}_2$ and the participant $\mathcal{P}_i$ run τ instances of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ in parallel, one for each query $q \in \mathcal{Q}$. Following the execution, $\mathcal{P}_i$ receives $\mathsf{D}[q] + \mathsf{r}_q$ whereas S_u receives $\mathsf{r}_q, [q]_u$, for $u \in \{1, 2\}$ and $q \in \mathcal{Q}$. $\mathcal{P}_i$ then adds up the received messages to get a masked version of the desired output, i.e, $\sum_{q \in \mathcal{Q}} \mathsf{D}[q] + \mathsf{mask}_{\mathcal{Q}}$ with $\mathsf{mask}_{\mathcal{Q}} = \sum_{q \in \mathcal{Q}} \mathsf{r}_q$. $\mathsf{S}_1, \mathsf{S}_2$ compute $\mathsf{mask}_{\mathcal{Q}}$ in the same way.

The protocol could be completed by S_1 and S_2 sending $\mathsf{mask}_{\mathcal{Q}}$ to $\mathcal{P}_i$, then $\mathcal{P}_i$ unmasking its value to obtain the desired output. However, before communicating the mask, the servers must ensure that all queries in $\mathcal{Q}$ are distinct, as shown in $\mathcal{F}_{\mathsf{pirsum}}$ (Fig. 8). For this, $\mathsf{S}_1, \mathsf{S}_2$ use their share of the queries $q \in \mathcal{Q}$ and participate in a secure computation protocol with S_0. We capture this with an ideal functionality $\mathcal{F}_{\mathsf{vrfy}}$, which takes the secret shares of τ values from S_1 and S_2 and returns `Accept` to the servers if all of the underlying secrets are distinct. Otherwise, it returns `abort`.

Protocol $\mathsf{PIR}_{\mathsf{sum}}$

Input(s): i) $\mathsf{S}_1, \mathsf{S}_2 : \mathsf{D}; |\mathsf{D}| = N$, ii) $\mathcal{P}_i : \mathcal{Q} = \{q_1, \ldots, q_\tau\}$, and iii) $\mathsf{S}_0 : \perp$.
Output: $\mathcal{P}_i : \mathsf{res} = \sum_{q \in \mathcal{Q}} \mathsf{D}[q]$ for distinct queries, else $\mathsf{res} = \perp$.

Computation

1. For each $q \in \mathcal{Q}$,
   a. $\mathsf{S}_1, \mathsf{S}_2$ and $\mathcal{P}_i$ invoke $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ (cf. $\mathbf{HYB}_2$ in proof of Lemma 1) with the inputs D, q.
   b. Let $\mathsf{r}_q, [q]_u$ denote the output of S_u, for $u \in \{1, 2\}$ and $\mathsf{D}[q] + \mathsf{r}_q$ denote the output of $\mathcal{P}_i$.
2. $\mathcal{P}_i$ computes $\mathsf{res}' = \sum_{q \in \mathcal{Q}} (\mathsf{D}[q] + \mathsf{r}_q)$, while $\mathsf{S}_1, \mathsf{S}_2$ computes $\mathsf{mask}_{\mathcal{Q}} = \sum_{q \in \mathcal{Q}} \mathsf{r}_q$.
3. $\mathsf{S}_1, \mathsf{S}_2$ and S_0 invokes $\mathcal{F}_{\mathsf{vrfy}}$ on the secret shares of queries, denoted by $\{[q]_u\}_{q \in \mathcal{Q}, u \in \{1,2\}}$, to check the distinctness of the queries in $\mathcal{Q}$.
4. If $\mathcal{F}_{\mathsf{vrfy}}$ returns `Accept`, $\mathsf{S}_1, \mathsf{S}_2$ sends $\mathsf{mask}_{\mathcal{Q}}$ to $\mathcal{P}_i$, who computes $\mathsf{res} = \mathsf{res}' - \mathsf{mask}_{\mathcal{Q}}$. Otherwise, `abort`.

Fig. 7: $\mathsf{PIR}_{\mathsf{sum}}$ Protocol.

1) Security of $\mathsf{PIR}_{\mathsf{sum}}$ *Protocol:* Fig. 8 presents the ideal functionality for $\mathsf{PIR}_{\mathsf{sum}}$ in the context of malicious participants. In this case, $\mathcal{F}_{\mathsf{pirsum}}$ first checks whether all the queries made by the participant $\mathcal{P}_i$ are distinct. If yes, the correct result is sent to $\mathcal{P}_i$; otherwise, $\perp$ is sent to $\mathcal{P}_i$.

Functionality $\mathcal{F}_{\mathsf{pirsum}}$

$\mathcal{F}_{\mathsf{pirsum}}$ interacts with servers in $\mathcal{C}$, and participant $\mathcal{P}_i \in \mathcal{P}$.
Input: $\mathcal{F}_{\mathsf{pirsum}}$ receives τ indices denoted by $\mathcal{Q} = \{q_1, \ldots, q_\tau\}$ from $\mathcal{P}_i$ and a database D from $\mathcal{C}$.
Computation: $\mathcal{F}_{\mathsf{pirsum}}$ sets $y = \sum_{j=1}^{\tau} \mathsf{D}[q_j]$ if all the queries in $\mathcal{Q}$ are distinct. Else, it sets $y = \perp$.
Output: $\mathcal{F}_{\mathsf{pirsum}}$ sends y to $\mathcal{P}_i$.

Fig. 8: PIR-SUM functionality (malicious participants).

Lemma 1. *Protocol* $\mathsf{PIR}_{\mathsf{sum}}$ *(Fig. 7) securely realises the* $\mathcal{F}_{\mathsf{pirsum}}$ *ideal functionality (Fig. 8) for the case of malicious participants in the* $\{\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}, \mathcal{F}_{\mathsf{vrfy}}\}$*-hybrid model.*

Proof. The proof follows with a hybrid argument based on the three hybrids $\mathbf{HYB}_0$, $\mathbf{HYB}_1$, and $\mathbf{HYB}_2$ discussed below. Furthermore, any secure three-party protocol can be used to instantiate $\mathcal{F}_{\mathsf{vrfy}}$ in RIPPLE.

We use a standard 2-server PIR functionality, denoted by $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$, to instantiate $\mathcal{F}_{\mathsf{pirsum}}$. The guarantees of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$, however, are insufficient to meet the security requirements of $\mathcal{F}_{\mathsf{pirsum}}$, so we modify $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ as a sequence of hybrids, denoted by $\mathbf{HYB}$. The modification is carried out in such a way that for a malicious participant $\mathcal{P}_i$, each hybrid is computationally indistinguishable from the one before it. As the first hybrid, $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ is denoted by $\mathbf{HYB}_0$.

$\mathbf{HYB}_0$: Let $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ denote a 2-server PIR ideal functionality for our case, with database holders S_1 and S_2, and client $\mathcal{P}_i$. For a database D held by S_1 and S_2 and a query q held by $\mathcal{P}_i$, $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ returns $\mathsf{D}[q]$ to $\mathcal{P}_i$, but S_1 and S_2 receive nothing.

$\mathbf{HYB}_1$: We modify $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ so that it returns $\mathsf{D}[q] + \mathsf{r}$ to $\mathcal{P}_i$, and $\mathsf{S}_1, \mathsf{S}_2$ receive r, where r is a random value from the domain of database block size. In other words, the modification can be thought of as the standard $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ being executed over a database $\mathsf{D}^{\mathsf{r}} = \mathsf{D} + \mathsf{r}$ rather than the actual database D. This modification leaks no additional information regarding the query to the servers because they will receive random masks that are independent of the query q. Furthermore, from the perspective of $\mathcal{P}_i$ with no prior knowledge of the database D, $\mathbf{HYB}_1$ will be indistinguishable from $\mathbf{HYB}_0$ because the values it sees in both cases are from the same distribution. As a result, $\mathbf{HYB}_0 \approx \mathbf{HYB}_1$.

$\mathbf{HYB}_2$: Looking ahead, in $\mathsf{PIR}_{\mathsf{sum}}$, servers $\mathsf{S}_1, \mathsf{S}_2$ and participant $\mathcal{P}_i$ run τ instances of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ in parallel, one for each query $q \in \mathcal{Q}$. As shown in $\mathcal{F}_{\mathsf{pirsum}}$ (Fig. 8), the servers must ensure that all of the queries in $\mathcal{Q}$ are distinct. For this, we modify $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ in $\mathbf{HYB}_1$ to additionally output a secret share of the query q to each of S_1 and S_2. Because the servers S_1 and S_2 are assumed to be non-colluding in our setting, this modification will leak no information about the query q to either server. Since the output to $\mathcal{P}_i$ remains unchanged, $\mathbf{HYB}_1 \approx \mathbf{HYB}_2$ from $\mathcal{P}_i$'s perspective. □

C. Reducing participant's communication

$\mathsf{PIR}_{\mathsf{sum}}$ is realized in $\mathrm{RIPPLE}_{\mathsf{PIR}}$ using two different approaches, each with its own set of trade-offs, with a goal of minimizing the communication at the participant's end. While the first approach, denoted by $\mathsf{PIR}_{\mathsf{sum}}^{\mathsf{I}}$ (Fig. 9), sacrifices computation for better communication, the second approach, denoted by $\mathsf{PIR}_{\mathsf{sum}}^{\mathsf{II}}$ (Fig. 10), reduces both the computational and communication overhead of the participant in $\mathsf{PIR}_{\mathsf{sum}}^{\mathsf{I}}$ with the help of additional server $\mathsf{S}_0 \in \mathcal{C}$.

1) $\mathsf{PIR}_{\mathsf{sum}}^{\mathsf{I}}$ *(Fig. 9):* In this approach, we instantiate $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ using PIR techniques based on Function Secret Sharing (FSS) [59]–[61]. To retrieve the q-th block from the database, $\mathcal{P}_i$ uses FSS on a Distributed Point Function (DPF) [62] that evaluates to a 1 only when the input q is 1 and to 0 otherwise. $\mathcal{P}_i$ generates two DPF keys k_1 and k_2 that satisfy the above constraint and sends one key to each of the servers S_1 and S_2. The servers S_1 and S_2 can then locally expand their key share to obtain their share for the bit vector $\bar{\mathsf{b}}$ and the rest of the procedure proceeds similarly to the naive linear summation method discussed in §V-A. The key size for a database of size N records using the optimised DPF construction in [59] is about $\lambda \log(N/\lambda)$ bits, where $\lambda = 128$ for an AES-based implementation. Fig. 9 provides the formal details of the $\mathsf{PIR}_{\mathsf{sum}}^{\mathsf{I}}$ protocol.

Security: For semi-honest participants, the security of our method directly reduces to that of the 2-server PIR protocol in [59]. However, as mentioned in [59], a malicious participant could generate incorrect DPF keys, risking the scheme's security and correctness. To prevent this type of misbehavior, Boyle et al. [59] present a form of DPF called "verifiable DPF", which can assure the correctness of the DPF keys created by $\mathcal{P}_i$ at the cost of an increased *constant* amount of communication between the servers.

While verifiable DPFs in $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$ ensure the validity of the τ bit vectors generated by $\mathcal{P}_i$, they do not ensure that bit vectors $\vec{\mathsf{b}}_1, \ldots, \vec{\mathsf{b}}_\tau$ correspond to τ distinct locations in the database D. However, we use the correctness guarantee of verifiable DPFs to reduce the communication cost for verification, as discussed in §V-A0a. In detail, all τ bit vectors $\vec{\mathsf{b}}_1, \ldots, \vec{\mathsf{b}}_\tau$ (PIR queries) are secret-shared between S_1 and S_2, each guaranteed to have exactly one 1 and the rest 0. To ensure distinctness, S_1 and S_2 XOR their respective τ shares locally to obtain the secret-share of a single vector $\vec{\mathsf{b}}_c = \oplus_{k=1}^{\tau} \vec{\mathsf{b}}_k$. The challenge is then to check if $\vec{\mathsf{b}}_c$ has exactly $\tau 1$ bits. This can be accomplished by having S_1 and S_2 agree on a random permutation π and reconstructing $\pi(\vec{\mathsf{b}}_c)$ to S_0 and allowing S_0 to perform the check, as in the naive approach (cf. §V-A0a).

Protocol $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$

Input(s): i) $\mathsf{S}_1, \mathsf{S}_2 : \mathsf{D}; |\mathsf{D}| = N$, ii) $\mathcal{P}_i : \mathcal{Q} = \{q_1, \ldots, q_\tau\}$, and iii) $\mathsf{S}_0 : \perp$.
Output: $\mathcal{P}_i : \mathsf{res} = \sum_{q \in \mathcal{Q}} \mathsf{D}[q]$

Computation S_1 and S_2 sample τ random mask values $\{m_1, \ldots, m_\tau\} \in \mathbb{Z}_{2^\ell}^\tau$ such that $\sum_{j=1}^{\tau} m_j = 0$. For each $q \in \mathcal{Q}$, execute the following:

1. $\mathsf{S}_1, \mathsf{S}_2$ locally compute $\mathsf{D}^{m_q} = \mathsf{D} + m_q$.
2. Execute DPF protocol [59] (verifiable DPF for malicious participants) with $\mathcal{P}_i$ as client with input q. Server S_u obtains $[\vec{\mathsf{b}}_q]_u$ with $\mathsf{b}_q^j = 1$ for $j = q$ and $\mathsf{b}_q^j = 0$ for $j \neq q$, for $u \in \{1, 2\}$.

Verification Let $\{\vec{\mathsf{b}}_{q_1}, \ldots, \vec{\mathsf{b}}_{q_\tau}\}$ denote the bit vectors whose XOR-shares are generated during the preceding steps.

3. Servers verify correctness of q_j, $j \in [\tau]$, by executing the Ver algorithm of the verifiable DPF protocol [59]. It outputs `Accept` to S_1 and S_2 if q_j has exactly 1 one and $(N-1)$ zeroes. Else, it outputs `abort`.
4. S_u computes $[\vec{\mathsf{b}}_c]_u = \oplus_{q \in \mathcal{Q}} [\vec{\mathsf{b}}_q]_u$, for $u \in \{1, 2\}$.
5. S_1 and S_2 non-interactively agree on random permutation π.
6. S_u sends $\pi([\vec{\mathsf{b}}_c]_u)$ to S_0, for $u \in \{1, 2\}$.
7. S_0 locally reconstructs $\pi(\vec{\mathsf{b}}_c) = \pi([\vec{\mathsf{b}}_q]_1) \oplus \pi([\vec{\mathsf{b}}_q]_2)$. It sends `Accept` to S_1 and S_2, if $\pi(\vec{\mathsf{b}}_c)$ has exactly τ ones. Else, it sends `abort`.

Output Transfer Send $\perp$ to $\mathcal{P}_i$ if verifiable DPF or S_0 generated `abort` during verification. Otherwise, proceed as follows:

8. S_u sends $[y_q]_u = \bigoplus_{j=1}^{N} [\mathsf{b}_q^j]_u \mathsf{D}^{m_q}[j]$ to $\mathcal{P}_i$, for $q \in \mathcal{Q}, u \in \{1, 2\}$.
9. $\mathcal{P}_i$ locally computes $\mathsf{res} = \sum_{q \in \mathcal{Q}} ([y_q]_1 \oplus [y_q]_2)$.

Fig. 9: $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$ Protocol.

Computation Complexity (#AES operations): In $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$, the participant $\mathcal{P}_i$ must perform $4 \cdot \log(N/\lambda)$ AES operations as part of the key generation algorithm for each of the τ instances of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ over a database of size N, where $\lambda = 128$ for an AES-based implementation. Similarly, S_1 and S_2 must perform $\log(N/\lambda)$ AES operations for each of the N DPF evaluations. We refer to Table 1 in [59] for more specifics.

2) $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ (Fig. 10): In this approach, we use the server S_0 to reduce the computation and communication of the participant $\mathcal{P}_i$ in $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$. The idea is that S_0 plays the role of $\mathcal{P}_i$ for the PIR protocol in $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$. However, $\mathcal{P}_i$ cannot send its query q to S_0 in clear because it would violate privacy. As a result, $\mathcal{P}_i$ selects random values $q', \theta_q \in [N]$ such that $q = q' + \theta_q$. In this case, q' is a *shifted version* of the index q, and θ is a *shift correction* for q. $\mathcal{P}_i$ sends q' to S_0 and θ_q to both S_1 and S_2. The rest of the computation until output retrieval will now occur solely among the servers.

Protocol $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$

Input(s): i) $\mathsf{S}_1, \mathsf{S}_2 : \mathsf{D}; |\mathsf{D}| = N$, ii) $\mathcal{P}_i : \mathcal{Q} = \{q_1, \ldots, q_\tau\}$, and iii) $\mathsf{S}_0 : \perp$.
Output: $\mathcal{P}_i : \mathsf{res} = \sum_{q \in \mathcal{Q}} \mathsf{D}[q]$

Computation S_1 and S_2 sample τ random mask values $\{m_1, \ldots, m_\tau\} \in \mathbb{Z}_{2^\ell}^\tau$ such that $\sum_{j=1}^{\tau} m_j = 0$. For each $q \in \mathcal{Q}$, execute the following:

1. $\mathsf{S}_1, \mathsf{S}_2$ locally compute $\mathsf{D}^{m_q} = \mathsf{D} + m_q$, i.e., $\mathsf{D}^{m_q}[j] = \mathsf{D}[j] + m_q$, for $j \in [N]$.
2. $\mathcal{P}_i, \mathsf{S}_1, \mathsf{S}_2$ sample random $\theta_q \in [N]$.
3. $\mathcal{P}_i$ computes and sends $q' = q - \theta_q$ to S_0.
4. Servers execute DPF protocol [59] with S_0 as client with input q'. Server S_u obtains $[\vec{\mathsf{b}}_{q'}]_u$ with $\mathsf{b}_{q'}^j = 1$ for $j = q'$ and $\mathsf{b}_{q'}^j = 0$ for $j \neq q'$, for $u \in \{1, 2\}$.
5. S_u locally applies θ_u on $[\vec{\mathsf{b}}_{q'}]_u$ to generate $[\vec{\mathsf{b}}_q]_u$, for $u \in \{1, 2\}$.

Verification Let $\{\vec{\mathsf{b}}_{q_1}, \ldots, \vec{\mathsf{b}}_{q_\tau}\}$ denote the bit vectors whose XOR-shares are generated during the preceding steps:

6. S_k computes $[\vec{\mathsf{b}}_c]_k = \oplus_{q \in \mathcal{Q}} [\vec{\mathsf{b}}_q]_k$, for $u \in \{1, 2\}$.
7. S_1 and S_2 non-interactively agree on random permutation π.
8. S_u sends $\pi([\vec{\mathsf{b}}_c]_u)$ to S_0, for $u \in \{1, 2\}$.
9. S_0 locally reconstructs $\pi(\vec{\mathsf{b}}_c) = \pi([\vec{\mathsf{b}}_q]_1) \oplus \pi([\vec{\mathsf{b}}_q]_2)$. It sends `Accept` to S_1 and S_2, if $\pi(\vec{\mathsf{b}}_c)$ has exactly τ ones. Else, it sends `abort`.

Output Transfer Send $\perp$ to $\mathcal{P}_i$ if S_0 generated `abort` during verification. Otherwise, proceed as follows:

10. S_u sends $[y_q]_u = \bigoplus_{j=1}^{N} [\mathsf{b}_q^j]_u \mathsf{D}^{m_q}[j]$ to $\mathcal{P}_i$, for $q \in \mathcal{Q}, u \in \{1, 2\}$.
11. $\mathcal{P}_i$ locally computes $\mathsf{res} = \sum_{q \in \mathcal{Q}} ([y_q]_1 \oplus [y_q]_2)$.

Fig. 10: $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ Protocol.

The servers run a DPF instance [59] with S_0 acting as the client and input query q'. At the end of the computation, S_1 and S_2 obtain the bit vector $\vec{\mathsf{b}}_{q'}$, which corresponds to q'. However, as discussed in $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$, the servers require an XOR sharing corresponding to the actual query q to continue the computation. S_1 and S_2 do this by using the shift correction value θ_q received from $\mathcal{P}_i$. Both S_1 and S_2 will perform a right cyclic shift of their $\vec{\mathsf{b}}_{q'}$ shares by θ_q units. A negative value for θ_q indicates a cyclic shift to the left.

Stage	$\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$	$\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$
$\mathcal{P}_i$ to servers in $\mathcal{C}$	$2\tau(\lambda+2)\log(N/\lambda)+4\tau\lambda$	$\tau \log N$
Server to server	0	$2\tau(\lambda+2)\log(N/\lambda)+4\tau\lambda$
Servers in $\mathcal{C}$ to $\mathcal{P}_i$	$\tau \cdot 2\ell$	$\tau \cdot 2\ell$
+ Verification (mal.)	$2N+2+\delta$	$2N+2$

TABLE II: Summary of communication costs for $\mathsf{PIR}_{\mathsf{sum}}$ to retrieve sum of τ indices from a database with N elements. λ denotes the AES key size ($\lambda = 128$ in [60]), ℓ denotes the block size in bits ($\ell = 128$ in this work), and δ denotes the constant involved in the verifiable DPF approach [59].

It is easy to see that the XOR shares obtained after the cyclic shift corresponds to the bit vector $\vec{\mathsf{b}}_q$. To further optimise $\mathcal{P}_i$'s communication, $\mathcal{P}_i$ and servers $\mathsf{S}_1, \mathsf{S}_2$ non-interactively generate random shift correction values θ_q first using the shared-key setup, and only the corresponding q' values are communicated to S_0. The rest of the protocol is similar to $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$, and the formal protocol is shown in Fig. 10. In terms of malicious participants, $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ has an advantage over $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$ as there is no need to use a verifiable DPF to protect against malicious $\mathcal{P}_i$, because the semi-honest server S_0 generates the DPF key instead of $\mathcal{P}_i$.

Improving Verification Costs in $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$*:* A large amount of communication is used in the $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ protocol to verify malicious participants. More specifically, in Step 8 of Fig. 10, $2N$ bits are communicated towards S_0 to ensure the distinctness of the queries made by the participant $\mathcal{P}_i$. We note that allowing a small amount of leakage to S_0 could improve this communication and is discussed next.

Consider the following modification to the $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ protocol. Instead of sampling θ_q for each query $q \in \mathcal{Q}$ (cf. Step 2 in Fig. 10), $\mathcal{P}_i, \mathsf{S}_1$, and S_2 sample only one random shift value θ and uses it for all τ instances. Since the queries must be distinct, $\mathcal{P}_i$ is forced to send distinct q' values to S_0 in Step 3 of Fig. 10. If not, S_0 can send `abort` to S_1 and S_2 at this step, eliminating the need for communication-intensive verification. The relative distance between the queried indices would be leaked to S_0 as a result of this optimization. In concrete terms, if we use the same θ value for any two queries $q_m, q_j \in \mathcal{Q}$, then $q_m - q_n = q'_m - q'_n$. Because S_0 sees all q' values in the clear, it can deduce the relative positioning of $\mathcal{P}_i$'s actual queries. However, since S_0 has no information about the underlying database D, this leakage may be acceptable for some applications.

3) Summary of communication costs: Tab. II summarises the communication cost for our two $\mathsf{PIR}_{\mathsf{sum}}$ approaches for instantiating $\mathcal{F}_{\mathsf{pirsum}}$ over a database of size N with τ PIR queries per client. Fig. 11a and Tab. IV show the concrete communication costs of $\mathsf{RIPPLE}_{\mathsf{PIR}}$ for varying average numbers of encounters E^{avg}, which corresponds to the threshold τ in $\mathsf{PIR}_{\mathsf{sum}}$. Our largest benchmarks with a population size of 10M (corresponding to 10M database blocks, leading to a database size of around 1 GBytes for block size $\ell = 128$ bits) with 500 average encounters (corresponding to threshold $\tau = 500$) only require around 500 KBytes communication per participant for $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$ and less than 20 KBytes for $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$.

VI. Evaluation

In this section, we evaluate and compare the computation and communication efficiency of our $\mathsf{RIPPLE}_{\mathsf{PIR}}$ protocol presented in §IV. A fully-fledged implementation, similar to existing contact tracing apps, would necessitate collaboration with industry partners to develop a real-world scalable system for national deployment. Instead, we carry out a proof-of-concept implementation and provide micro benchmark results for all major building blocks.[7] We focus on the simulation phase for benchmarking, which is separate from the token generation phase. The simulations can ideally be done overnight while mobile phones are charging and have access to a high-bandwidth WiFi connection. According to studies [63], [64], sleeping habits in various countries provide a time window of several hours each night that can be used for this purpose.

a) Setup and Parameters: We run the benchmarks on the server-side with three servers (two for FSS-PIR and one as a helper server as discussed in §V-C) with Intel Core i9-7960X CPUs@2.8 GHz and 128 GB RAM connected with 10 Gbit/s LAN and 0.1 s RTT. The client is a Samsung Galaxy S10+ with an Exynos 9820@2.73 GHz and 8GB RAM. We use the code of [65] to instantiate FSS-PIR. We implement the AGCT in C++ and follow previous work on cuckoo hashing [66] by using tabulation hashing for the hash functions.

We instantiate our protocols in RIPPLE with $\kappa = 128$ bit security. For $\mathsf{RIPPLE}_{\mathsf{PIR}}$, we use the FSS-PIR scheme of [59], [65] as the baseline. The addresses are hashed with SHA-256 and trimmed to $40 - 1 + \log_2(\mathsf{p} \cdot E^{\mathsf{avg}})$ bits, where p is the number of participants and E^{avg} represents the average number of encounters per participant per simulation step. We set $E^{\mathsf{avg}} = 100$ while benchmarking based on numbers provided by research on epidemiological modeling [67], [68]. To avoid cycles when inserting n messages into the AGCT (cf. §IV-A2), we set its size to $10n$. This can be further improved as discussed in §IV-A2 [57]. A typical simulation step corresponds to one day, such that 14 simulation steps can simulate two weeks.

A. Communication Complexity

Here, we look at the communication costs that our protocols incur. To analyse the scalability of our protocols, we consider p participants ranging from thousand (1K) to twenty million (20M). Tab. III summarises the communication costs of each

[7] Note that we are not attempting to create the most efficient instantiation. More optimizations will undoubtedly improve efficiency, and our protocols can be heavily parallelized with many servers. Instead, our goal here is to demonstrate the viability of RIPPLE protocols for large-scale deployment.

Entities	Protocol	Population (p)									
		1K	10K	50K	100K	500K	1M	2M	5M	10M	20M
Participants in $\mathcal{P}$ (in KB)	RIPPLE$_{\mathsf{PIR}}$: PIR$^{\mathsf{I}}_{\mathsf{sum}}$ (§V-C1)	51.63	62.42	69.97	73.22	80.77	84.02	87.27	91.56	94.81	98.06
	RIPPLE$_{\mathsf{PIR}}$: PIR$^{\mathsf{II}}_{\mathsf{sum}}$ (§V-C2)	3.45	3.49	3.52	3.53	3.56	3.57	3.59	3.60	3.62	3.63
Servers in $\mathcal{C}$ (in GB)	RIPPLE$_{\mathsf{PIR}}$ (§V)	0.01	0.10	0.48	0.96	4.80	9.60	19.20	48.00	96.00	192.00

TABLE III: Communication costs per simulation step in our RIPPLE instantiations.

participant as well as the communication servers ($\mathcal{C}$) for one simulation step in a specific simulation. One simulation step includes all protocol steps, beginning with participants locally computing their infection likelihood δ and ending with them obtaining their cumulative infection likelihood Δ for that step.

1) Participant Communication: We depict the participants' communication in Fig. 11a with varied average number of encounters E^{avg} ranging from 10 to 500 for a population of 10M. The participant communication increases sub-linearly with the database size. In particular, the participant's communication in PIR$^{\mathsf{I}}_{\mathsf{sum}}$ ranges from 51.63KB to 98.06KB. The communication in PIR$^{\mathsf{II}}_{\mathsf{sum}}$, on the other hand, is about 3.5KB for all participant sizes we consider. This reduced communication is due to the optimization in PIR$^{\mathsf{II}}_{\mathsf{sum}}$, which offloads the DPF key generation task to the helper server S_0 (cf. §V-C2). A participant in PIR$^{\mathsf{I}}_{\mathsf{sum}}$ must communicate approximately 7MB of data for a 2-week simulation for a 10M population with $E^{\mathsf{avg}} = 500$, whereas it is only 0.25MB in the case of PIR$^{\mathsf{II}}_{\mathsf{sum}}$.

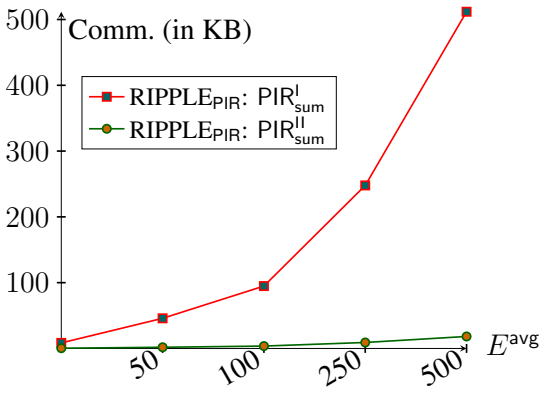
Tab. IV provides the communication per participant for multiple population sizes in PIR$^{\mathsf{I}}_{\mathsf{sum}}$ and PIR$^{\mathsf{II}}_{\mathsf{sum}}$, while varying the average number of encounters E^{avg} per simulation step from 10 to 500. The communication cost in RIPPLE$_{\mathsf{PIR}}$ increases sub-linearly with the population size and linearly in E^{avg}.

2) Server Communication: The servers' communication is primarily attributed to the anonymous communication channel they have established, which provides unlinkability and, thus, privacy to the participants' messages. Communicating M messages through the channel requires the servers to communicate $3M$ messages in RIPPLE$_{\mathsf{PIR}}$ (cf. [26, §B.3]).

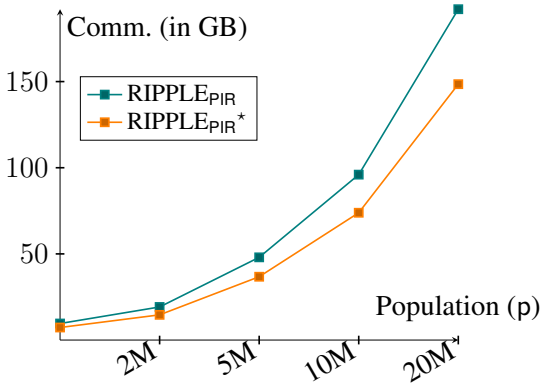
For a population of 10M, the servers must communicate 96GB of data among themselves.Setting the proper bit length for the address field in the messages can further reduce communication. For example, a population of 20M with $E^{\mathsf{avg}} = 100$ can be accommodated in a 70-bit address field. Using this optimization will result in an additional 23 % reduction in communication at the servers, as shown in Fig. 11b and Tab. IV.

B. Computation Complexity

This section focuses on the runtime, including computation time and communication between entities. Tab. V summarizes the computation time with respect to a participant $\mathcal{P}_i$ for a two-week simulation over a half-million population. $\mathcal{P}_i$'s computation time in RIPPLE$_{\mathsf{PIR}}$ is at most 5 milliseconds for PIR$^{\mathsf{II}}_{\mathsf{sum}}$, while it increases to around 165 milliseconds for PIR$^{\mathsf{I}}_{\mathsf{sum}}$. The increased computation time in PIR$^{\mathsf{I}}_{\mathsf{sum}}$ is due to DPF key generation, which scales sub-linearly with population size.



(a) Participant's communication with varying E^{avg} for a population of 10M.



(b) Servers' communication per simulation step for varying population. * denotes the results for optimized bit addresses in RIPPLE$_{\mathsf{PIR}}$ (cf. full version [26]).

Fig. 11: Communication Costs of RIPPLE.

Population p	Protocol	E^{avg}				
		10	50	100	250	500
100K	RIPPLE$_{\mathsf{PIR}}$: PIR$^{\mathsf{I}}_{\mathsf{sum}}$	6.24	34.99	73.22	193.79	403.83
	RIPPLE$_{\mathsf{PIR}}$: PIR$^{\mathsf{II}}_{\mathsf{sum}}$	0.35	1.76	3.53	8.87	17.81
1M	RIPPLE$_{\mathsf{PIR}}$: PIR$^{\mathsf{I}}_{\mathsf{sum}}$	7.32	40.38	84.02	220.78	457.81
	RIPPLE$_{\mathsf{PIR}}$: PIR$^{\mathsf{II}}_{\mathsf{sum}}$	0.35	1.78	3.57	8.98	18.01
10M	RIPPLE$_{\mathsf{PIR}}$: PIR$^{\mathsf{I}}_{\mathsf{sum}}$	8.40	45.78	94.81	247.77	511.79
	RIPPLE$_{\mathsf{PIR}}$: PIR$^{\mathsf{II}}_{\mathsf{sum}}$	0.36	1.80	3.62	9.08	18.22

TABLE IV: Communication (in KB) per participant in a simulation step for varying average numbers of encounters E^{avg} and population sizes p.

We measured the overall runtime of RIPPLE$_{\mathsf{PIR}}$ for a simulation of 2 weeks (including secure shuffling and anonymous

	Per Simulation Step / Simulation ($N_{\text{step}} = 14$)		
	Message Generation (in ms)	PIR Queries (in ms)	Output Computation (in ms)
$\text{PIR}^{\text{I}}_{\text{sum}}$	0.30 / 4.26	11.73 / 160	4.8e-2 / 6.72e-1
$\text{PIR}^{\text{II}}_{\text{sum}}$	0.30 / 4.26	3.0e-3 / 4.2e-2	4.8e-2 / 6.72e-1

TABLE V: Average participant computation times per simulation step distributed across various tasks. Values are obtained using a mobile for a population of 500K with $E^{\text{avg}} = 100$.

communication). The simulation over a population of p=10K takes 74s, and 453s for a population of p=500K.

1) Battery Usage: The token generation phase in RIPPLE consumes the most amount of mobile battery as this phase is active throughout the day. However, the main interaction during this phase is exchanging two random tokens, which is similar to existing contact tracing apps. This usage could be optimized by mobile OS providers like Apple and Google, as discussed by Vaudenay et al. [69] and Avitabile et al. [70] in the context of contact tracing apps. Their technology enables an app to run in the background, thus, significantly improving battery life, which is otherwise impossible for a standard third-party mobile application. Additionally, RIPPLE could offer users the choice to participate only in simulations while charging so as not to cause any unwanted battery drain.

2) Comparison to Related Work: Note that no experimental comparison to related work is (and can be) done, as RIPPLE is the first distributed privacy-preserving epidemiological modeling system. Established contact tracing apps, such as the SwissCovid[8], the German Corona-Warn-App[9], or the Australian COVIDSafe[10] only record contacts for notifying contacts of infected people. Concretely, contact tracing basically relates to RIPPLE's token generation phase, while the other three phases (simulation initialization, simulation execution, and result aggregation, cf. §III-B) are not covered by any contact tracing system. Crucially, the main contribution of our work is how to realize the simulation execution, which has never been done before. Hence, no meaningful comparison between the systems is possible due to differences in the fundamental functionalities.

3) Code availability: Available at DOI: 10.5281/zenodo.6595448.

Acknowledgements

This project received funding from the ERC under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the DFG within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230.

References

[1] D. Maison, D. Jaworska, D. Adamczyk, and D. Affeltowicz, "The challenges arising from the COVID-19 pandemic and the way people deal with them. a qualitative longitudinal study," *PloS One*, 2021.

[2] A. D. Christy Cooney, "High-risk monkeypox contacts advised to isolate," *BBC*, 2022, https://www.bbc.com/news/uk-61546480.

[3] L. Reichert, S. Brack, and B. Scheuermann, "Poster: Privacy-preserving contact tracing of covid-19 patients," *IEEE S&P*, 2021.

[4] N. Ahmed, R. A. Michelin, W. Xue, S. Ruj, R. Malaney, S. S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, and S. K. Jha, "A survey of COVID-19 contact tracing apps," *IEEE Access*, 2020.

[5] C. Troncoso, M. Payer, J. Hubaux, M. Salathé, J. R. Larus, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, L. Barman, S. Chatel, K. G. Paterson, S. Capkun, D. A. Basin, J. Beutel, D. Jackson, M. Roeschlin, P. Leu, B. Preneel, N. P. Smart, A. Abidin, S. Gurses, M. Veale, C. Cremers, M. Backes, N. O. Tippenhauer, R. Binns, C. Cattuto, A. Barrat, D. Fiore, M. Barbosa, R. Oliveira, and J. Pereira, "Decentralized privacy-preserving proximity tracing," *IEEE Data Eng. Bull.*, 2020.

[6] S. Vaudenay, "Centralized or decentralized? the contact tracing dilemma," Cryptology ePrint Archive, Report 2020/531, 2020, https://ia.cr/2020/531.

[7] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song, "Epione: Lightweight contact tracing with strong privacy," arXiv preprint arXiv:2004.13293, 2020, https://arxiv.org/abs/2004.13293.

[8] B. Pinkas and E. Ronen, "Hashomer–privacy-preserving bluetooth based contact tracing scheme for hamagen," *Real World Crypto and NDSS Corona-Def Workshop*, 2021.

[9] W. Lueks, S. F. Gürses, M. Veale, E. Bugnion, M. Salathé, K. G. Paterson, and C. Troncoso, "CrowdNotifier: Decentralized privacy-preserving presence tracing," *PoPETs*, 2021.

[10] K. Hogan, B. Macedo, V. Macha, A. Barman, X. Jiang *et al.*, "Contact tracing apps: Lessons learned on privacy, autonomy, and the need for detailed and thoughtful implementation," *JMIR Medical Informatics*, 2021.

[11] P. Tupper, S. P. Otto, and C. Colijn, "Fundamental limitations of contact tracing for covid-19," *FACETS*, 2021.

[12] D. Lewis, "Where covid contact-tracing went wrong," *Nature*, 2020.

[13] G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. Di Filippo, A. Di Matteo, and M. Colaneri, "Modelling the covid-19 epidemic and implementation of population-wide interventions in Italy," *Nature Medicine*, 2020.

[14] A. J. Kucharski, P. Klepac, A. J. Conlan, S. M. Kissler, M. L. Tang, H. Fry, J. R. Gog, W. J. Edmunds, J. C. Emery, G. Medley *et al.*, "Effectiveness of isolation, testing, contact tracing, and physical distancing on reducing transmission of SARS-CoV-2 in different settings: a mathematical modelling study," *The Lancet Infectious Diseases*, 2020.

[15] I. Cooper, A. Mondal, and C. G. Antonopoulos, "A SIR model assumption for the spread of COVID-19 in different communities," *Chaos, Solitons & Fractals*, 2020.

[16] P. C. Silva, P. V. Batista, H. S. Lima, M. A. Alves, F. G. Guimarães, and R. C. Silva, "COVID-ABS: an agent-based model of COVID-19 epidemic to simulate health and economic effects of social distancing interventions," *Chaos, Solitons & Fractals*, 2020.

[17] G. R. Shinde, A. B. Kalamkar, P. N. Mahalle, N. Dey, J. Chaki, and A. E. Hassanien, "Forecasting models for coronavirus disease (covid-19): A survey of the state-of-the-art," *SN Computer Science*, 2020.

[18] R. N. Thompson, "Epidemiological models are important tools for guiding covid-19 interventions," *BMC Medicine*, vol. 18, no. 1, p. 152, 2020.

[19] T. Šušteršič, A. Blagojević, D. Cvetković, A. Cvetković, I. Lorencin, S. B. Šegota, D. Milovanović, D. Baskić, Z. Car, and N. Filipović, "Epidemiological predictive modeling of covid-19 infection: Development, testing, and implementation on the population of the benelux union," *Frontiers in Public Health*, vol. 9, 2021.

[20] N. G. Davies, A. J. Kucharski, R. M. Eggo, A. Gimma, W. J. Edmunds, T. Jombart, K. O'Reilly, A. Endo, J. Hellewell, E. S. Nightingale *et al.*, "Effects of non-pharmaceutical interventions on covid-19 cases, deaths, and demand for hospital services in the UK: a modelling study," *The Lancet Public Health*, 2020.

[21] D. Adam, "Special report: The simulations driving the world's response to COVID-19." *Nature*, 2020.

[22] P. Klepac, A. J. Kucharski, A. J. Conlan, S. Kissler, M. L. Tang, H. Fry, and J. R. Gog, "Contacts in context: large-scale setting-specific social mixing matrices from the BBC pandemic project," *MedRxiv*, 2020.

[23] R. Baron, N. Hamdiui, Y. B. Helms, R. Crutzen, H. M. Götz, and M. L. Stein, "Evaluating the Added Value of Digital Contact Tracing Support Tools for Citizens: Framework Development," *JMIR Res Protoc*, 2023. [Online]. Available: https://doi.org/10.2196/44728

[24] B. Unim, I. Zile-Velika, Z. Pavlovska, L. Lapao, M. Peyroteo, J. Misins, M. J. Forjaz, P. Nogueira, T. Grisetti, and L. Palmieri, "The role of digital tools and emerging devices in COVID-19 contact

[8] https://github.com/SwissCovid [9] https://www.coronawarn.app/en/
[10] https://www.health.gov.au/resources/apps-and-tools/covidsafe-app

tracing during the first 18 months of the pandemic: a systematic review," *European Journal of Public Health*, 2024. [Online]. Available: https://doi.org/10.1093/eurpub/ckae039

[25] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*, 2017.

[26] D. Günther, M. Holz, B. Judkewitz, H. Möllering, B. Pinkas, T. Schneider, and A. Suresh, "Privacy-Preserving Epidemiological Modeling on Mobile Graphs," arXiv preprint, 2022, https://arxiv.org/abs/2206.00539.

[27] F. Brauer, "Compartmental models in epidemiology," in *Mathematical Epidemiology*, 2008.

[28] S. He, Y. Peng, and K. Sun, "Seir modeling of the covid-19 and its dynamics," *Nonlinear Dynamics*, 2020.

[29] V. L. Gatta, V. Moscato, M. Postiglione, and G. Sperlì, "An Epidemiological Neural Network Exploiting Dynamic Graph Structured Data Applied to the COVID-19 Outbreak," *IEEE Trans. Big Data*, 2021.

[30] R. Alguliyev, R. Aliguliyev, and F. Yusifov, "Graph modelling for tracking the COVID-19 pandemic spread," *Infectious Disease Modelling*, 2021.

[31] W. O. Kermack and A. G. McKendrick, "Contributions to the mathematical theory of epidemics—i," in *Bulletin of Mathematical Biology*, 1991.

[32] M. Small and C. K. Tse, "Small world and scale free model of transmission of SARS," in *International Journal of Bifurcation and Chaos*, 2005.

[33] Y.-C. Chen, P.-E. Lu, C.-S. Chang, and T.-H. Liu, "A time-dependent SIR model for covid-19 with undetectable infected persons," *Transactions on Network Science and Engineering*, 2020.

[34] R. M. May and A. L. Lloyd, "Infection dynamics on scale-free networks," *Phys. Rev. E*, 2001.

[35] N. M. Ferguson, D. A. Cummings, C. Fraser, J. C. Cajka, P. C. Cooley, and D. S. Burke, "Strategies for mitigating an influenza pandemic," *Nature*, 2006.

[36] K. Kupferschmidt, "Case clustering emerges as key pandemic puzzle," 2020, https://www.science.org/doi/full/10.1126/science.368.6493.808.

[37] S. Luo, F. Morone, C. Sarraute, M. Travizano, and H. A. Makse, "Inferring personal economic status from social network location," *Nature Communications*, 2017.

[38] Y.-A. d. Montjoye, J. Quoidbach, F. Robic, and A. S. Pentland, "Predicting personality using novel mobile phone-based metrics," in *International conference on social computing, behavioral-cultural modeling, and prediction*, 2013.

[39] T. Araki, J. Furukawa, K. Ohara, B. Pinkas, H. Rosemarin, and H. Tsuchida, "Secure graph analysis at scale," in *ACM CCS*, 2021.

[40] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2009.

[41] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh, "ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction," in *ACM CCSW@CCS*, 2019.

[42] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *USENIX Security*, 2020.

[43] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation," in *USENIX Security*, 2021.

[44] O. Tkachenko, C. Weinert, T. Schneider, and K. Hamacher, "Large-scale privacy-preserving statistical computations for distributed genome-wide association studies," in *ASIACCS*, 2018.

[45] T. Schneider and O. Tkachenko, "EPISODE: efficient privacy-preserving similar sequence queries on outsourced genomic databases," in *ASIACCS*, 2019.

[46] Y. Lindell and B. Pinkas, "An efficient protocol for secure two-party computation in the presence of malicious adversaries," in *EUROCRYPT*, 2007.

[47] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," *Journal of Cryptology*, 2010.

[48] R. Lehmkuhl, P. Mishra, A. Srinivasan, and R. A. Popa, "MUSE: Secure inference resilient to malicious clients," in *USENIX Security*, 2021.

[49] N. Chandran, D. Gupta, S. L. B. Obbattu, and A. Shah, "SIMC: ML inference secure against malicious clients at semi-honest cost," in *USENIX Security*, 2022.

[50] O. Diekmann, H. Heesterbeek, and T. Britton, *Mathematical Tools for Understanding Infectious Disease Dynamics*. Princeton University Press, 2012.

[51] G. F. Hatke, M. Montanari, S. Appadwedula, M. Wentz, J. Meklenburg, L. Ivers, J. Watson, and P. Fiore, "Using bluetooth low energy (BLE) signal strength estimation to facilitate contact tracing for covid-19," 2020, https://arxiv.org/ftp/arxiv/papers/2006/2006.15711.pdf.

[52] Z. Erkin, J. R. Troncoso-pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez, "Privacy-preserving data aggregation in smart metering systems: An overview," in *Signal Processing Magazine*, 2013.

[53] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly aggregation for the smart-grid," in *PETS*, 2011.

[54] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *International Conference on Smart Grid Communications*, 2010.

[55] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *SP*, 2019.

[56] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame *et al.*, "SAFELearn: secure aggregation for private federated learning," in *IEEE Security and Privacy Workshops (SPW)*, 2021.

[57] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "PSI from PaXoS Fast, Malicious Private Set Intersection," in *EUROCRYPT*, 2020.

[58] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *FOCS*, 1995.

[59] E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing: Improvements and extensions," in *ACM CCS*, 2016.

[60] D. Boneh, E. Boyle, H. Corrigan-Gibbs, N. Gilboa, and Y. Ishai, "Lightweight techniques for private heavy hitters," in *IEEE S&P*, 2021.

[61] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, "Riposte: An anonymous messaging system handling millions of users," in *IEEE S&P*, 2015.

[62] N. Gilboa and Y. Ishai, "Distributed point functions and their applications," in *EUROCRYPT*, 2014.

[63] O. J. Walch, A. Cochran, and D. B. Forger, "A global quantification of "normal" sleep schedules using smartphone data," *Science advances*, 2016.

[64] V. Woollaston, *Sleeping habits of the world revealed: The US wakes up grumpy, China has the best quality shut-eye and South Africa gets up the earliest*, 2015, https://www.dailymail.co.uk/sciencetech/article-3042230/Sleeping-habits-world-revealed-wakes-grumpy-China-best-quality-shut-eye-South-Africa-wakes-earliest.html.

[65] D. Kales, O. Omolola, and S. Ramacher, "Revisiting user privacy for certificate transparency," in *EuroS&P*, 2019.

[66] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on OT extension," *TOPS*, 2018.

[67] J. Mossong, N. Hens, M. Jit, P. Beutels, K. Auranen, R. Mikolajczyk, M. Massari, S. Salmaso, G. S. Tomba, J. Wallinga *et al.*, "Social contacts and mixing patterns relevant to the spread of infectious diseases," *PLoS Medicine*, 2008.

[68] S. Y. Del Valle, J. M. Hyman, H. W. Hethcote, and S. G. Eubank, "Mixing patterns between age groups in social networks," *Social Networks*, 2007.

[69] S. Vaudenay and M. Vuagnoux, "Analysis of swisscovid," Tech. Rep., 2020.

[70] G. Avitabile, V. Botta, V. Iovino, and I. Visconti, "Towards defeating mass surveillance and SARS-CoV-2: The Pronto-C2 fully decentralized automatic contact tracing system," 2020. [Online]. Available: https://eprint.iacr.org/2020/493