Provably Secure Contact Tracing with Conditional Private Set Intersection

Jonathan Takeshita, Ryan Karl, Alamin Mohammed, Aaron Striegel, and Taeho Jung(✉)

University of Notre Dame, 46556 Notre Dame IN, USA
{jtakeshi,rkarl,amohamm2,striegel,tjung}@nd.edu

Abstract. The novel coronavirus COVID-19 spreads easily through personal contact, requiring the use of contact tracing to track the spread of the disease. Many existing approaches either trust a public health authority with private data, or publish patients' data, leading to privacy breaches. Private Set Intersection based on Homomorphic Encryption is a promising solution, but it is limited because the management of keys is challenging and further filtering of contacts is not included. We present a protocol for secure and private conditional contact tracing, allowing the tracking of users' contacts subject to extra conditions. We construct and apply our new primitive of Conditional Private Set Intersection and combine it with a Trusted Execution Environment (TEE) to construct a protocol with provable security and a high degree of functionality. Our approach moves the memory- and computation-intensive portions of contact tracing out of the TEE to a cloud server. We also present how multi-hop contact tracing can be done with minimal user communication. Our proof-of-concept implementation with Microsoft SEAL allows users to perform their computation in less than 9 min, and the cloud's per-user computation can be as little as 11 min for a population of 50,000 users with 500 infected (assuming 40 contacts/user) in a day. With other HE libraries/schemes that allows customized parameter sets, our protocol will show much higher scalability.

Keywords: Contact tracing · Lattice-based cryptography · Private set intersection · Trusted execution environment

1 Introduction

The disease COVID-19 is highly contagious and spreads easily through personal contact. Contact tracing is used to determine a person's past personal contacts to aid in tracking the spread of the disease, in particular to find which individuals are sources of or are at risk for infection. Human-based contact tracing by public health authorities is inherently invasive to one's privacy and individuals may be reluctant to participate. Users' smartphones can automatically trace their movements and contacts, which can be useful for accurate contact tracing, but disclosing such information poses a significant threat to individual

J. Garcia-Alfaro et al. (Eds.): SecureComm 2021, LNICST 398, pp. 352–373, 2021.
https://doi.org/10.1007/978-3-030-90019-9_18

privacy. Such concerns can discourage participation in contact tracing, and to incentivize participation in contact tracing, it is desirable to construct a contact tracing protocol that guarantees a high degree of individual privacy.

Various privacy-sensitive smartphone-based contact tracing protocols have weaknesses in privacy and accuracy [3,4,24,26,33,41]. Further, these protocols are generally rigid, and not easily extensible to calculations beyond basic tracing of contacts. There are many different additional conditions that a Public Health Authority (PHA) might wish to apply to the contact tracing (i.e., *conditional contact tracing*). For example, a PHA may only wish to deem users who have had exposure of 10 min as at risk for elderly people while the threshold could be higher for younger people (e.g., 15 min per CDC guidelines [16]). Many other conditions may be needed similarly (e.g., degree of infectiousness, vaccinated or not). Other cryptographic approaches require significant amounts of repeated network communication and/or computation to achieve this [34,37]. Cryptographic protocols such as Private Set Intersection (PSI) can be useful in such problems, but key management makes it challenging to apply it in scenarios with multiple users. Trusted Execution Environments (TEEs) support isolated and secure program execution, but they face difficulties with scalability, parallelism, and latency from memory access at scale [25,38], which leads us to the strategy of combining TEEs and cryptography for scalability.

To address such limitations, we let users provide encrypted lists of contacts and enable more sophisticated contact tracing that meets the needs of privacy and versatility. We combine TEE and homomorphic cryptography to create a protocol that allows a PHA to obliviously perform contact tracing without harming individual privacy. We also extend existing PSI based on HE to devise our novel *Conditional Private Set Intersection (CPSI)* protocol which can enforce additional conditions for PSI , and combine it with TEE to construct protocols for secure contact tracing with minimal privacy loss and conditional contact tracing. We use cryptographic approaches to maintain semantic security of messages to completely protect users' data with provable security, and combine this with the use of the TEE to solve issues such as key management that would otherwise make a straightforward application of HE difficult with multiple users. By performing homomorphic computation outside the TEE, we allow for parallel computation and avoid the overhead incurred by the TEE [38].

Our contact tracing protocol improves upon previous work by guaranteeing provable security and privacy for all users including the infected ones, which is not possible in existing decentralized approaches based on smartphones [2]. The computation and network communication are minimized for all users and parties, and more functionality in filtering out contacts is enabled without violating individual privacy. We selectively use a TEE to avoid excessive overhead caused by large-scale data processing within a capacity-limited TEE. Finally, we present further extensions useful for COVID contact tracing such as multi-hop tracing, while minimizing extra network communication from this. This paper has following contributions:

- A new notion of CPSI is presented, which returns only the intersection of elements that satisfies certain conditions, and it is constructed based on HE. We also present two novel optimization techniques for CPSI to (1) prevent false positive cases and (2) address the trade-off between efficiency of condition calculation and admissible set sizes.
- A protocol combining CPSI and TEE to conditionally trace personal contacts with provable data security is presented, with variants that differ in which party receives the final result. Further, an extension of the contact tracing protocol allowing timestamp-based multi-hop tracing with backward/forward tracking is presented, with no additional communication needed for unexposed users even with multi-hop tracing.
- An open-source proof-of-concept implementation for continued research and reproducibility is provided via anonymized source code repository (URL available in Sect. 6).
- We use real-world bluetooth datasets to perform experiment, and show that our protocol's preprocessing and decryption times are negligible and that both user-side and cloud-side computation are acceptable.

2 Related Work in Contact Tracing

The Google/Apple Exposure Notification (GAEN) from Google and Apple [24] uses daily Temporary Exposure Keys (TEKs) to generate 15-minute rolling Pseudo-IDs. These IDs are broadcast via Bluetooth and are collected by users who come into contact with one another. In the event of an infection, the infected user's TEKs are made public and used to retrieve all past Pseudo-IDs that they have broadcast. This protocol has the large weakness of not lending privacy to infected users: once an infected users' TEKs have been released, their movements can be retroactively tracked through the Pseudo-IDs. Practical attacks on this protocol have been demonstrated in [3] which showed that the weaknesses in GAEN are *protocol-level*, not implementation-level. The weakness in GAEN is that a user ID is publicly broadcast on a list of infected users, which can allow retroactive tracing of an infected user by passive collection of Bluetooth beacons and cross-referencing with the released TEKs. In our work, we repair this vulnerability by not publicly revealing users' ID strings. We also note that GAEN is not capable of easily allowing computation on users' data with private parameters, while our protocol allows such computation.

The BlueTrace protocol (Singapore) [4] is heavily dependent on human intervention; it is not designed to be solely automated. In BlueTrace, Pseudo-IDs and searches on that data are all performed in plaintext, leading to the weakness of a Public Health Authority (PHA) being able to easily track users through their persistent IDs [10,41]. A similar issue is present with the EPIC protocol [1].

Protocols such as Covid Watch [26], Hamagen (Israel) [29], and TrackCOVID [45] have users perform the computation for contact tracing. We aim to not follow this path, but instead aim to remove as much of a computational burden from users as possible, shifting computation to cloud computing and greatly reducing the amount of data that users must download.

The SafePaths (MIT) [32,33] project, along with Hamagen [29], uses GPS-based tracing instead of Bluetooth tracking, which may be less precise at detecting contacts. SafePaths uses the cryptographic protocol of PSI, but has other weaknesses in security and privacy. Namely, infected users lose any privacy under SafePaths because they must release their past location traces to allow contact searching by other users. A contact tracing method based on Secure Multiparty Computation (MPC) [34] that uses imprecise location-based tracking may improve its precision by using bluetooth signals, but such solutions face serious scalability issues [41] and do not provide any privacy to infected users. The PPContactTracing protocol [37] takes a similar approach to SafePaths in combining PSI and locality-sensitive hashing. The Epione [42] system also uses a PSI construction, though their Diffie-Hellman-based construction lacks the quantum security and extensibility to other computations that modern FHE-based computation provides.

The TraceSecure protocol [5] provides a high level of security for users, even preventing any central authority from learning if users are exposed. However, TraceSecure requires frequent communications throughout the day between users and a central server, uses wasteful flooding to mask meaningful messages informing users of exposure, and its design (based on additive HE) is not easily extensible to further functionality.

3 Preliminaries

Mathematical Notation: The set $\mathbb{Z}_t$ is the set of integers modulo t. We use R to denote the quotient ring of $\mathbb{Z}[X]/\Phi(X)$, where $\Phi(X)$ is a properly chosen cyclotomic polynomial of degree N, a power of two. We define $R_t = \mathbb{Z}_t[X]/\Phi(X)$, the subring of R with all coefficients in $\mathbb{Z}_t$. The bitsize of a number t is $bit(t) = \lfloor log_2(t) \rfloor + 1$. A number $a \in \mathbb{Z}_t$ is a quadratic residue (QR) if there is a $b \in \mathbb{Z}_t$ such that $b^2 \equiv a \pmod{t}$ and a non-quadratic residue (non-QR) otherwise.

Trusted Execution Environment: TEEs [12] can provide trusted computing on a platform where other applications or even the host operating systems are untrusted. Secured memory contents are encrypted and not visible to untrusted processes and execution cannot be tampered with externally. TEEs can also perform remote attestation to prove to other parties that they are running a certain program, so that users will know that TEEs are performing the correct computation on users' data. TEEs do have pitfalls including expensive paging, and limitations with memory space and parallel computing [38], making a purely TEE-based approach undesirable.

Fully/Somewhat Homomorphic Encryption: HE schemes allow for computation to be done on encrypted operands. Modern HE schemes derive their post-quantum security from the Ring-Learning With Errors (RLWE) problem, and deal with operands in polynomial rings [6,8,15]. Schemes that allow for arbitrarily-sized arithmetic circuits to be computed are called Fully Homomorphic Encryption (FHE) schemes. In practice, most modern FHE schemes are

implemented as Somewhat Homomorphic Encryption (SHE) schemes to avoid the complex and intensive operation of bootstrapping. Arithmetic circuits in SHE are limited by pre-determined multiplicative depths. This work and related PSI work [8,9] use the B/FV scheme [15] that is parameterized by (N, q, t), where R_t is the plaintext space and R_q is the ciphertext space.

Private Set Intersection (PSI): PSI protocols allow two users to securely compute the intersection of their sets of elements, without revealing any information to eavesdroppers, or any elements not in the intersection to the other party. State-of-the-art PSI protocols have been constructed using HE [8,9]. These constructions are well-suited for a scenario where one user (the *sender*) is much more powerful than the other user (the *receiver*). This makes PSI particularly useful for our use case of contact tracing, where the PHA uses cloud servers and users use COTS phones/computers for contact tracing. Other PSI schemes also exist, but they are not suitable for our scenario. The protocols of De Cristofaro et al. [14] and Ion et al. [20] are insecure against quantum-capable adversaries. The PSI schemes based on MPC [11,30] have poor scalability due to multiple rounds of communication required. Our work is the first to formulate and construct CPSI for the needs of real-world applications.

Bluetooth Message Exchange: Bluetooth Low Energy (BLE) is a technology used for constructing short-range wireless mobile ad-hoc networks. BLE was specifically designed to facilitate low-cost and power-efficient implementations. BLE has been used to detect contact events by most contact-tracing apps [2,28,46] with which smartphones periodically broadcast Bluetooth beacon advertisements. When a smartphone in close proximity detects such advertisements, the underlying operating system notifies the observing application. Furthermore, the app can leverage the Received Signal Strength Indicator (RSSI) to gauge the distance between the two phones [22]. In a typical approach for Bluetooth-based contact tracing, Pseudo-IDs are woven into the Bluetooth advertisement which in turn is leveraged for contact tracing. In this work, we do not consider weaknesses of or attacks against Bluetooth [23,44], as it is not in the scope of the problem we consider.

4 Conditional Private Set Intersection

In this section, we introduce our new cryptographic primitive of CPSI, and show an FHE-based construction of CPSI. In Sect. 5, we deconstruct our CPSI protocol and distribute it among different parties to protect the privacy of both infected and at-risk users during contact tracing.

4.1 Definitions

Ideal Functionality: PSI between parties with sets X and Y computes the intersection $X \cap Y$ [8]. We may wish to subject the result to a further condition, calculating $X \cap_P Y = \{z \in X \cap Y | P\}$ for some predicate P where the parameters

Inputs: The sender and receiver input their secret finite sets X and Y, respectively. The receiver also inputs a secret set Y' of metadata associated with elements of Y. The cardinality of X and Y, as well as the sizes in bits of the elements of X, Y and Y', respectively, are publicly known. A predicate P operating on elements of Y' is computable by the sender, though its parameters may be kept secret from the sender.
Output: The receiver learns $X \cap_P Y = \{z \in X \cap Y | P(z')\}$, where $z' \in Y'$ is the metadata element corresponding to z. The sender does not learn anything. No additional information is learned by either party or any eavesdropper.

Fig. 1. Ideal functionality δ of CPSI.

of P may be kept private from both parties. Formally, we define a CPSI protocol as follows: consider a sender and receiver with respective sets X, Y, where the number of the elements in each set is known, and the domain size (in bits) of the elements is known. The predicate P is known to the sender. A CPSI protocol returns the result $X \cap_P Y$ to the receiver, and nothing to the sender. Neither party learns anything about the other party's held elements not in $X \cap_P Y$. The predicate P may be a predicate on the encrypted elements of X or Y, on attached metadata (encrypted or plain), or something else. We require in the formal definition of CPSI, given in Fig. 1, that P operate on metadata strictly correlated to the elements of the parties and kept private, though these requirements may be relaxed in some applications. It is common in many applications (e.g., machine learning) to wish to allow computation with hidden parameters, which is possible when P is computed homomorphically or otherwise privately. In practice, P might also be a function of a parameter s of the sender, i.e., $P = P_s$. Constructions where P is a function of multiple-valued data from both the sender and receiver are possible, but would require $\mathcal{O}(|X| \cdot |Y|)$ separate calculations, so we do not consider those in this work.

Adversary Model: In this scenario, we consider the parties to be honest-but-curious, i.e., they will follow the protocol accurately. Communication between the parties is through authenticated and non-malleable channels. An adversary may be an eavesdropper, or either party.

Security Model: A CPSI protocol is secure if the execution of the protocol is computationally indistinguishable from the execution of the ideal functionality of CPSI given in Fig. 1. This is formalized in Definition 1.

Definition 1. *A CPSI protocol Γ securely computes the ideal CPSI functionality δ if for every probabilistic polynomial-time (PPT) adversary A against Γ, there exists a PPT adversary S (the "simulator") against δ such that for every possible combination of input sets X, Y, Y' (with $|Y| = |Y'|$) with sizes polynomial in the security parameter λ, the views of $S_\delta(\lambda, X, Y, Y')$ and $A_\Gamma(\lambda, X, Y, Y')$ are computationally indistinguishable w.r.t. the security parameter λ. The view generated by S from the execution of δ is $S_\delta(\lambda, X, Y, Y')$, and similarly the view generated by A from the execution of Γ is $A_\Gamma(\lambda, X, Y, Y')$.*

4.2 Novel CPSI Construction Without False Positives

Our novel CPSI is shown in Fig. 2. Let the FHE plaintext modulus t be a prime number, so there are $\frac{t-1}{2}$ elements of $\mathbb{Z}_t$ that are not quadratic residues modulo t. In Chen et al.'s FHE-based PSI protocol [8], a decrypted result of the protocol is zero if and only if the corresponding element is in the set intersection, and is a random nonzero number otherwise. We construct our CPSI protocol to maintain that property by adding a nonzero predicate value to the PSI circuit result to force a result of zero to be nonzero when the condition is not met.

Unfortunately, this naïve approach may result in a false positive when a result is zero by coincidence even though the corresponding element is not in the intersection. We prevent this with a novel technique: we force the sum of the PSI and predicate values to be zero if and only if an element is in the intersection and the predicate is fulfilled. This is achieved by squaring the PSI result to force it to be a QR, and having the nonzero predicate value added be a non-QR, so that the sum of those values can never be zero modulo t unless both values are zero. More specifically, we let $t-k$ with $k \in [1,t)$ be a non-QR modulo t. We then require that the predicate $P(\cdot)$ be zero when the element being tested should be included in the PSI result (so that the computation is unaffected), and that $P(\cdot) = k$ when the element should be excluded from the intersection. Because no element of $\mathbb{Z}_t$ squared results in $t-k \pmod t$, $P(c'_i) + (\prod_{x\in X}(c_i - x))^2$ will only be zero modulo t exactly when both P and $\prod_{x\in X}(c_i - x)$ are zero modulo t. Because $t-k$ is a non-QR, adding k to any squared element of $\mathbb{Z}_t$ will never result in zero modulo t, preventing false positives caused by adding in a predicate value.

Input: The sender's set X, the receiver's set Y and the metadata set Y' associated with X, and the sender's predicate P operating on Y' are private inputs. The cardinalities of X, Y, and $Y'$are public inputs.
Output: The receiver learns $X \cap_P Y = \{z \in X \cap Y | P\}$.

1. **Setup:** The sender and receiver agree on appropriate parameters for a FHE scheme. The receiver generates a public-private key pair (pk, sk), and retains the secret key privately.
2. **Set Encryption:** The receiver encrypts its elements as $c_i = FHE.Enc_{pk}(y_i)$ for $y_i \in Y$, $c'_i = FHE.Enc_{pk}(y'_i)$ for $y'_i \in Y'$, and sends the ciphertexts c_i, c'_i to the sender.
3. **Computing the Intersection:** For each c_i, the sender samples a random nonzero plaintext element r, and homomorphically computes
$$d_i = r \cdot (P(c'_i) + (\prod_{x\in X}(c_i - x))^2) \tag{1}$$
The values d_i are returned to the receiver.
4. **Intersection Decryption:** The receiver outputs the conditioned intersection $X \cap_P Y = \{y_i | FHE.Dec_{sk}(d_i) = 0\}$

Fig. 2. Conditional private set intersection (CPSI) protocol Γ.

When computing the predicate P over metadata that should be kept private (e.g. times or durations of contact), we can homomorphically encrypt the metadata, and use the homomorphic polynomial interpolation function calculation [40] described in Sect. 5.2 to compute the predicate. Correctness is easy to see: if the predicate is zero, then the result is the same as the ordinary PSI circuit. If the predicate is nonzero, then adding it to a zero-valued PSI result results in k, which is nonzero. If the PSI result is nonzero, then adding a nonzero predicate to it will never result in a value that is zero modulo t, as shown above. This CPSI construction allows the filtering of elements based on arbitrary predicates, with no additional work required for decryption. Like related work in PSI, this basic protocol may have a high multiplicative depth (linear in $|X|$), which should be mitigated through the optimizations presented in Sect. 4.4.

4.3 Proof of Security

Theorem 1. *The CPSI protocol Γ ((Fig. 2) is secure under Definition 1.*

Proof. The views of the sender, receiver, and an external eavesdropper in the real and ideal protocols are computationally indistinguishable, so that a simulator operating in the ideal world can generate a view indistinguishable from that of the view a party or eavesdropper would see in the real world [21].

Sender: In both the real and ideal views, the sender sees the sets X and X', and knows $|Y|$ (and $|Y|'$). In the real view, the sender also sees the ciphertexts c_i, c'_i. A simulator $\mathcal{S}$ simulating the execution to an adversary $\mathcal{A}$ acting as a sender in the ideal case can easily provide HE ciphertexts. Due to the semantic security of the underlying HE scheme [15], these views are computationally indistinguishable w.r.t. the security parameter λ.

Receiver: In both the real and ideal views, the receiver sees the sets Y and Y', and learns $X \cap_P Y$. In the real view, the receiver also sees the decrypted results d_i, which are zero if $y_i \in X \cap_P Y$ and a random nonzero element otherwise. A simulator $\mathcal{S}$ in the ideal model can generate an identical view to that of the real model for the adversary $\mathcal{A}$ acting as a receiver by constructing the set $D = \{\tilde{d}_i\}$, where $\tilde{d}_i$ encrypts zero if $y_i \in X \cap_P Y$ and a random nonzero value otherwise. Thus the receiver's view of a real execution is indistinguishable from a view that a simulator can generate in the ideal execution.

Eavesdropper: In the ideal and real views, an eavesdropping adversary $\mathcal{A}$ knows $|X|$, $|Y|$, $|X'|$, $|Y'|$, and the bit-sizes of their elements. In the real view they additionally see the ciphertexts c_i, c'_i, d_i. Similarly to the case of the sender, semantically secure ciphertexts add nothing to the view generated, as the sizes of X, Y, Y' are already known. Indeed, a simulator $\mathcal{S}$ can easily generate ciphertexts c_i, c'_i, d_i, thus simulating the real view to $\mathcal{S}$. □

4.4 Compatibility with Standard Optimizations

Standard optimizations in FHE and FHE-based PSI can be applied to our protocol with little modification needed as we discuss below. Our contact tracing

implementation presented in Sect. 6 uses the most important optimizations of windowing, batching, and partitioning.

Windowing: With windowing, by precomputing powers and polynomial coefficients the original PSI protocol can be reduced to have a depth as low as one through precomputation of some terms and some additional communication [8]. Our CPSI protocol requires only one additional homomorphic multiplication and one homomorphic addition, after the original PSI circuit and predicate have been calculated. The circuit depth incurred is only two more than the original PSI protocol, though calculation of P may incur additional depth and computation.

Batching: A well-known optimization in lattice-based cryptography is batching, which encodes vectors of up to N elements of $\mathbb{Z}_t$ into a single plaintext element of R_t [18]. Homomorphic operations on data encoded in this manner can be carried out with parallelism (i.e., batching). As previously shown in previous PSI work [8], batching can be applied to reduce the network communication and computation by a factor of N. The only modification to the original PSI protocol is to sample random elements to be a batched vector instead of singleton elements. Our CPSI protocol can use the batching in the same way.

Hashing and Partitioning: The computation and depth of the protocol depend on the size of the sender's set. Hashing and partitioning are applied in previous PSI work to divide the sender's set into smaller, more manageable subsets [8,31]. With partitioning, the sender's set is simply partitioned into α subsets of approximately equal size, which reduces network communication and computation by a factor of α. Hashing similarly divides elements into hash buckets, and the parties only need to compare elements in the same bucket. Both can be applied to our CPSI protocol.

Modulus Switching: This technique reduces a ciphertext in R_q to one in $R_{q'}$, with $q' < q$, such that the switched-down ciphertext decrypts to the same value [6]. This can be performed in our CPSI protocol after the sender's computation to reduce the size of the response to the receiver by $\frac{log(q)}{log(q')}$, which is observed empirically by [8] to be a reduction of up to 20%.

4.5 Novel Optimization with Dual Plaintext Space

When a predicate P is homomorphically calculated on the receiver's private metadata, the two goals of our protocol come into conflict. To make the homomorphic calculation of P via polynomial interpolation as efficient as possible, the plaintext space of our HE scheme should be held as small as possible [40]. On the other hand, to express many distinct set elements, it is desirable to have larger plaintext spaces.

With batching, both these goals are possible. Let c be an integer factor, and suppose that we have a plaintext space of $\mathbb{Z}_t$. Recall that the batching allows encoding up to N values into a single ciphertext. We can then encode elements from $\mathbb{Z}_{t^c}$ to $\mathbb{Z}_t^c$ by placing each base-t "digit" of the element into a separate

slot. The corresponding metadata value in $\mathbb{Z}_t$ can then be duplicated across all c corresponding slots in the metadata plaintext polynomial. By doing so, we can deal with the randomly generated IDs in the space $\mathbb{Z}_{t^c}$ with the smaller actual message space $\mathbb{Z}_t$. This allows a CPSI protocol to be run with a large set of possible elements and a smaller predicate metadata domain, while maintaining a smaller and more efficient plaintext space for interpolation.

Using this strategy requires the receiver to check that all c slots that a value is decomposed into before concluding that an element is in the intersection, as two values are equal if and only if their t-digit decompositions are equal. This method of representing and packing elements thus does not affect correctness of CPSI. Using this dual plaintext space is most useful when a large plaintext space is desired to handle many distinct set elements (e.g., Pseudo-IDs) but the space of the metadata can be smaller.

5 Protocol for Conditional Contact Tracing

In this section, we present a CPSI-based contact tracing system that can filter contacts to be recorded only when some additional predicate is satisfied. Our system has a high degree of functionality and privacy guarantees, with the tradeoff of more complexity and computation. The core idea behind our protocol is computing a CPSI of users' encountered and broadcast Pseudo-IDs, so that a nonempty intersection indicates exposure. We take the strategy of using the TEE to handle some trusted computation such as key distribution, preprocessing, and result distillation, while still outsourcing the heavy homomorphic computations to a cloud server. By encrypting all data sent to the cloud, we fix the weakness in the GAEN [2] which discloses unencrypted Pseudo-IDs of infected users. While it is possible to use a TEE for all private outsourced computation, doing so may severely limit the ability to scale and compute concurrently over large workloads, due to both a bottleneck at the TEE's memory transfer and vulnerabilities in TEE multithreading [25]. Further, large workloads may incur extremely high overhead due to the TEE's overhead in paging and memory encryption [38]. For these reasons, we aim to use the TEE as little as possible, and shift as much work as possible to scalable and parallel cloud computation. Communications are assumed to be authenticated and nonmalleable, with authorized access controls.

Our protocol has two variations, determining which of the user and public health authority gets the final results. Some countries with more libertine policies may prefer the former approach, while countries with more centralized governmental authority may prefer the latter.

5.1 Scenario of Conditional Contact Tracing

System Model: Our formulation considers three types of parties: (1) *Ordinary users*, who participate through a smartphone application. The users' smartphones can receive and record information from nearby users' smartphones via Bluetooth. During nights, a user's smartphone can sync while charging with

their laptop or desktop computer, which is then able to perform operations such as encryption on the phone's behalf, making the overhead of homomorphic encryption less of an issue for users. We aim to minimize the amount of data that users must download to participate, and only send a constant and small amount of data to users. (2) The *computation server* C (possibly untrusted), which performs the computation of CPSI, acting in the modified role of sender in a CPSI protocol (C does not have access to plaintext data, but carries out the computation of the sender). (3) The PHA's *key management server* (PHA_M), who distributes keys and manages interaction between the parties. This entity has access to a TEE. We assume the PHA knows the IDs of the participating users but nothing else. They receive the final result identifying exposed users, and will choose if and how to inform those users. Any party can perform remote attestation to verify the integrity of the code running inside the TEE.

Condition to be Used: As described in Sect. 1, there are many possible additional conditions that the PHA might wish to apply to contact tracing. In our proof-of-concept system, we use thresholding with a private bound as the condition in the predicate P (e.g., if $x \geq h$, $P = 0$; $P = k$ otherwise), as such conditions are likely useful in real-world contact tracing (e.g., age, degree of infectiousness, number of contacts) and other applications.

Adversary Model: Adversaries in this scenario may be any of the three parties above, an external eavesdropping adversary, or any collusion thereof. We assume communications between users and both the TEE and cloud server are authenticated and nonmalleable. We assume that participants in this protocol are honest-but-curious, meaning that they may carry out arbitrary computation to try to learn other users' private data, but will otherwise participate honestly in the protocol. As noted in other work in secure computation, honest-but-curious parties are a reasonable assumption, as even participation in this protocol requires some level of trust between participants [43].

* Users periodically broadcast randomly-rotating Pseudo-IDs.
Inputs: Infected users input their previously broadcast Pseudo-IDs from the past 14 days (or other interval). Uninfected users input their previously encountered Pseudo-IDs from the past 14 days (or other interval), along with any corresponding metadata used to filter contacts conditionally. A PHA does not give any input.
Output: The PHA learns which users have both been in contact with infected users and satisfy the conditions.

Fig. 3. Ideal functionality of conditional contact tracing.

Ideal Functionality and Informal Security Model: Our ideal functionality, shown in Fig. 3, is for users to input their data, and for each uninfected user to learn only if they are exposed. Informally, security of a protocol is maintained if the view of an adversary or collusion of adversaries is indistinguishable from the view of an execution of the ideal protocol.

5.2 Strategy in Constructing Our Protocol

A user can be said to be at risk if the set of people they have encountered intersects the set of infected persons, and the PHA can set conditions for further filtering. However, naively using the CPSI protocol has security and privacy issues. In particular, the sender would need to know the entire unencrypted set of infected persons, and the receiver could learn when and from whom they were exposed by noting which elements in the intersection are held in common. We thus apply the ability of HE to separate knowledge and computation, and deconstruct the role of sender in a CPSI protocol into that of a sender and computer. In this formulation, the sender is a party much like the receiver, who sends their encrypted data, and the computer is tasked with the actual CPSI computation. Due to the need for preprocessing of the sender's data (for windowing and partitioning), the sender is further split into data holders and a party who collects the aggregate data and performs the preprocessing. The role of the receiver is also split - one party will receive and decrypt the CPSI result from the computing party, and will send a final distilled result indicating exposure or non-exposure to users. CPSI is used to allow for further filtering, though if no additional condition on the intersection is required then regular PSI can be used. Though CPSI requires that the sizes of users' sets be publicly known, for contact tracing we want to conceal the number of contacts a user has had; batching masks this information. The number of batching slots can be increased as needed (at the cost of extra computation), allowing such masking.

Conditional statements such as "if $x \geq h$, $P = 0$; $P = k$ otherwise" are not directly supported in FHE schemes. Therefore, we use Lagrange Interpolation to formulate a polynomial function that allows for *homomorphic thresholding*, i.e., comparison and equality over ciphertexts, at the cost of extra multiplicative depth [40]. For any function $f : \mathbb{Z}_t \to \mathbb{Z}_t$ with prime t (or a prime power), there is a polynomial expression of f, defined as $F : \mathbb{Z}_t \to \mathbb{Z}_t$ and constructed as $F(x) = \sum_{x_i \in \mathbb{Z}_t} f(x) \cdot (\prod_{x_a \in \mathbb{Z}_t, x_a \neq x_i} \frac{x - x_a}{x_i - x_a})$, which is well-defined if all of the denominators and t are coprime (i.e., the denominators are multiplicatively invertible). The degree of this polynomial is at most $t - 1$, and its multiplicative depth is logarithmic in t. Precomputing powers of an argument x makes computing this function a simple, single-depth dot product.

Integrating CPSI into our protocol to allow for conditional contact tracing does not require any additional computation on the TEE's part during decryption; nor is there any difference to the TEE's preprocessing computations. Users do have the extra overhead of additional preprocessing (computing and encrypting powers of their metadata), and the cloud server now has to homomorphically compute the predicate P. However, this tradeoff allows the inclusion of private conditional contact tracing by the PHA and its TEE.

Conditional Contact Tracing

Parties: The parties here are users U, the PHA PHA_M with a TEE, and a cloud computation server C.
Setup: PHA_M uses its TEE to generate a public/private key pair for use in FHE. The TEE retains the private (decryption) key. PHA_M distributes the public (encryption) key and any other public parameters. Additionally, the PHA may pass a set of parameters for predicate calculation pr (e.g., a threshold value) to C, either as a cleartext or a homomorphic ciphertext.
ID Collection: Throughout the day (or another specified interval), users collect randomly-chosen Pseudo-IDs broadcast through Bluetooth. These Pseudo-IDs are changed every 15 minutes to protect privacy. (Alternately, they can be changed for every contact, as in [5].)
Encryption and Upload: Each user U formulates a set Y of Pseudo-IDs encountered since its last upload and more recent than 14 days (or another interval), along with any associated metadata (Y'). The Pseudo-IDs and metadata are homomorphically encrypted, and uploaded to C. If a user is confirmed to be infected with COVID-19, then their broadcast Pseudo-IDs are uploaded to the TEE of PHA_M for preprocessing, and thereafter forwarded to C as the homomorphically encrypted set X. The set X can be partitioned by daily or two-day intervals, to reduce the required size of the plaintext space.
Compute Intersection of Contacts:

1. The cloud server C has a set of ciphertexts S of Pseudo-IDs from infected users. The cloud server can also (homomorphically) compute $P_{pr}(\cdot)$.
2. For each user U with a set Y sent to C as ciphertexts c_i and associated metadata Y' sent as ciphertexts c'_i, C will homomorphically compute the PSI circuit of Y and X as $d'_i = \prod_{x \in X}(y_i - x)$ for $y_i \in Y$.
3. The predicate value is calculated as $p_i = P_{pr}(c'_i) \in \{k, 0\}$, where $t - k$ is not a QR mod t. The calculation of p_i can be homomorphically calculated through polynomial interpolation [40].
4. The CPSI result is $d_i = r_i \cdot (p_i + (d'_j)^2)$, for a random nonzero value r_i. (Alternately, if no condition is used, then $d_i = r_i \cdot d'_i$.) This encrypted result is forwarded to the TEE of PHA_M.
5. The TEE will decrypt the result, and determine whether the user is at risk of infection. If the conditioned intersection is nonempty, then there was some contact between a user and an infected individual which puts that user at risk of infection. The TEE can then forward each users' result to the user (first variant) or the PHA (second variant).

Fig. 4. A CPSI-based protocol for conditional contact tracing.

5.3 Protocol Description

Our protocol is detailed in Fig. 4. The key management server PHA_M uses its TEE to derive public-private key pairs for HE, along with any other parameters used in the CPSI protocol. Public (encryption) keys are distributed to users, and the secret (decryption) key is held by the TEE of PHA_M and not distributed.

Table 1. Network communication/computation overhead for TEE

Operation	Plain intersection in TEE	Our protocol (Colocated TEE)	Our protocol (No Colocation)
Input received	$\mathcal{O}(N_x \cdot N_{pid} + N_y \cdot N_c)$	$\mathcal{O}(N_x \cdot N_{pid})$	$\mathcal{O}(N_x \cdot N_{pid})$
Processing	$\mathcal{O}(N_x \cdot N_{pid} \cdot N_y \cdot N_c)$	$\mathcal{O}(N_x \cdot N_{pid} + N_y \cdot N_c)$	$\mathcal{O}(N_x \cdot N_{pid} + N_y \cdot N_c)$
Output sent	$\mathcal{O}(N_y)$	$\mathcal{O}(N_y)$	$\mathcal{O}(N_y + N_x \cdot N_{pid})$

Throughout the day, each users' phone will broadcast randomly changing Pseudo-ID strings to other nearby phones. At the end of each day (or other interval), users will have an list of IDs of contacts. Users who are infected with COVID-19 will (not necessarily homomorphically) encrypt their list, and upload their encrypted Pseudo-IDs to PHA_M, whose TEE will preprocess those values (partition and precompute windowing coefficients) and send the homomorphically encrypted results to C as the set S (with partitioning and other preprocessing performed).

Each uninfected user U will then send their encrypted set Y of encountered Pseudo-IDs with accompanying encrypted metadata Y' to C. Then for each user U, C will run the computation of CPSI on Y, Y', and S, and send the result to PHA_M. The TEE of PHA_M will then be able to decrypt the result, and return only an indication of exposure or nonexposure for each user. The result may be returned to either users (the first variant) or the PHA (the second variant) - this can be easily configured, and users can verify the security of decryption through remote attestation of the TEE. In practice, the precomputation and encryption of uninfected users can be done overnight via a synced laptop or desktop computer, and this computation is "offline", so the time and energy of encryption is not an issue.

5.4 Performance and Optimizations

This scheme aims to shift the greatest burdens of PSI computation away from users or a TEE and onto cloud servers with scalable resources. A system simply using the TEE to compute contacts would be heavily burdened by the amount of communication, computation, and memory used. Let N_x be the number of infected users with N_{pid} IDs each, and N_y be the number of uninfected users with up to N_c collected Pseudo-IDs each. Then the TEE needs to receive and operate upon $N_x \cdot N_{pid} + N_y \cdot N_c$ operands. In our protocol, the TEE only needs to take in $N_x \cdot N_{pid}$ operands and forward $\mathcal{O}(N_x \cdot N_{pid})$ preprocessed ciphertexts, then decrypt and forward N_y results. This represents a significant savings in communication, memory, and computation for the TEE. When the TEE, PHA, and cloud are colocated (in the same machine), the network savings are even greater. This is shown in Table 1. The runtime of comparison in the TEE can be reduced to loglinear with a binary search tree or even linear with a hash table, but this will incur a higher memory overhead. Seeing as the TEE is memory-limited, these strategies may not be advisable.

Most of the optimizations from Sect. 4.4 can be applied directly to our CPSI-based protocol. *Batching, hashing,* and *modulus switching* can be applied "out-of-the-box" to reduce computation and communication. *Windowing* and *partitioning* can also be applied to decrease the depth of the homomorphic computations. However, the preprocessing for these optimizations requires knowledge of the entire set of the infected users' broadcast Pseudo-IDs, which is why they

are sent to the TEE for preprocessing. The data of infected and uninfected users can be partitioned by sorting IDs based on the 1-day or 2-day interval in which they are broadcast, making it easier to update and deprecate datasets. The dual plaintext space of Sect. 4.5 can also be applied to allow for a larger pool of random Pseudo-IDs to reduce the probability of collisions, while keeping predicate calculation runtimes tolerable. In general, a high degree of parallelism is applicable to our scheme. The cloud server can calculate each users' PSI in parallel, and that computation can be further parallelized across the partitions of the set of infected IDs.

5.5 Security

We informally discuss the security of our protocol, as the security comes from the security of CPSI proved in Theorem 1. This protocol is essentially deconstructing the CPSI protocol of Sect. 4.2 to aggregate all infected users' data in a secure environment for preprocessing as a single set, and to allow the decrypted result to be privately distilled to a binary result. There may be some privacy leakage in practice: for example, a person only in contact with one other person who receives an indication of exposure can conclude that their contact poses a health risk to them. However, such leakage is unavoidable, as it also occurs in an execution of the ideal protocol of Fig. 3.

The amount of ciphertexts sent may give upper and lower bounds to how many personal contacts a user has had, which may breach privacy. However, it is a generally reasonable assumption that a user will have much fewer than N contacts in a day, so that they only need to send one ciphertext. With this assumption, the exact number of a user's contacts is not leaked in this protocol, and privacy is maintained. For high-density populations, N can be increased as needed. If batching is not used, then padding sets of Pseudo-IDs with encryptions of a dummy element guaranteed to never be chosen as a Pseudo-ID can protect the number of a user's observed contacts.

Uninfected users only see Pseudo-IDs of other users, which rotate every 15 min or upon a contact, so that a user's movement cannot be easily tracked for longer intervals. They do not see anything else throughout the protocol. The cloud server never sees any unencrypted user data. The cloud server sees encrypted Pseudo-IDs from uninfected users, but due to batching, only a single ciphertext is sent to the cloud. Uninfected users learn that they came into contact with an infected user, but are not provided any other information. In extreme cases (e.g. a user only seeing one other person), the user may be able to learn if other people are infected. However, this is an unavoidable leakage as aforementioned. The PHA (sans TEE) learns which users have come into contact with infected people only, as the TEE only returns the final result for each user to the PHA. It is assumed that the PHA knows which users are infected.

Table 2. Performance on Real-world dataset

Operations by different entities	Time (seconds)
User Pseudo-ID Preprocessing & Encryption (for each user on average)	0.03
User Metadata Preprocessing & Encryption (for each user on average)	519.57
User Time for each contact tracing (for each user on average)	**519.60**
One-time Infected Pseudo-ID Preprocessing & Encryption for 27 Pseudo-IDs	0.19
Cloud Predicate (per user on average)	352.01
Cloud CPSI Circuit (per user on average)	0.45
Cloud Time for each contact tracing (per user on average)	**352.46**
Decryption total (for 20 users)	0.78
Total TEE time for each contact tracing (for 20 users)	**0.97**

6 Evaluation with Real-World and Synthetic Datasets

Our proof-of-concept implementation consists of four core programs performing the functionalities described in Sect. 5. These programs are written in standard C++ (no actual TEE was used for the proof-of-concept implementation), with the sole external dependency of Microsoft SEAL [36]. Our code is available at https://gitlab.com/jtakeshi/contact-tracing. We chose to use B/FV [15] and SEAL [36] due to their successful use in previous related work [8,9]. We included the optimizations of batching, windowing, and partitioning.

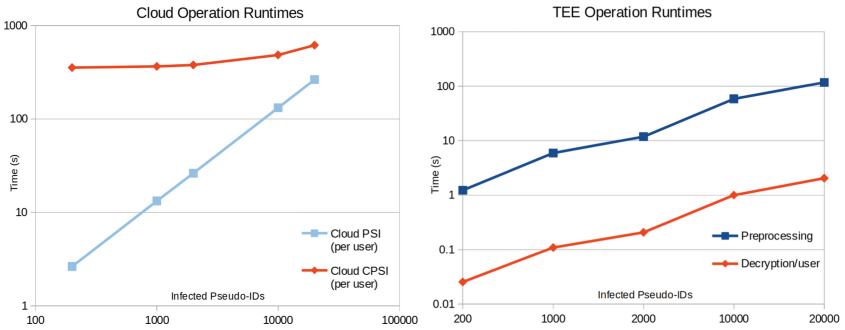
Our tests were run on a computer with an Intel Xeon 20-core CPU operating at 3.7 GHz with Intel SGX support, 128 GB of RAM, and Ubuntu 18.04. We used parameters of $t = 114,689$, $bit(t) = 17$, $N = 8,192$. We chose a partition size of 5 for the infected users' Pseudo-IDs. These parameters are the smallest that can be chosen in Microsoft SEAL to allow all the computations needed for CPSI.

The real-world dataset we use is an anonymized set of collected users' beacon contacts from real-world users. It is obtained via request from Project Tesserae [17], a multi-university research involving the collection, instrumentation and analysis of data from hundreds of smartphones and wearables over several years of continuous data streaming. The dataset includes one-month interactions of 20 users in an office building in 2018. The 20 users averaged about 228 contacts, allowing the use of only a single batched plaintext for all of a user's inputs. There were 27 Bluetooth beacons interpreted as infected Pseudo-IDs broadcast, so with a partition size of 5, 6 partitions were used for infected users' data. The CPSI metadata used was signal strength.

We first tested the performance of our implementation on the real-world dataset (Table 2). Reported figures are an average across all 20 users' computations for the user and cloud computations. Our results show that our protocol is effective and reasonably efficient for a small pool of users. The most intensive steps were the users' and cloud's work in predicate preprocessing and calculation, which is expected as the runtime of interpolation is linear in $t = 114,689$. When not using a condition, both cloud and user computations can be performed in

less than a second; health authorities with less powerful servers and more constituents may thus wish to consider this alternative. These results show a benchmark for execution time on a (small) real-world dataset, and give an estimate of calculation of predicate runtimes.

To test our scheme's scalability, we generated synthetic data with increasing amounts of infected users' Pseudo-IDs broadcast, using 200, 1000, 2000, and 20,000 Pseudo-IDs. Each user in this dataset observes $N = 8192$ Pseudo-IDs including the non-infected users', and 10 users' data was generated, so numbers reported are the average across 10 trials. The real-world dataset already shows the per-user predicate calculation time and user preprocessing time (which are independent of the number of users, infected or unexposed), so our main goal in this experiment is to see how the number of infected users affects the per-user performance of the cloud server. TEE preprocessing and decryption scaled well (as expected); Fig. 5(b) shows these operations taking less than 118 s and 2.1 s/user, respectively, with 20,000 infected users' Pseudo-IDs used. As shown in Fig. 5(a), the cloud's computation also increases linearly (as expected) with the number of infected users' Pseudo-IDs used. The cloud CPSI computations take 616.24 s/user, with 20,000 infected users' Pseudo-IDs used.



(a) Runtime of Cloud Operations (b) Runtime of TEE Operations

Fig. 5. Experimental results

We judged our efficiency in three areas. First, it is desirable that the computational load on the TEE-based computations be small; this goal is easily fulfilled, as shown with both real-world and synthetic datasets. Second, we want users' computation to be doable during overnight charging of devices; our results show that a user's computation can be performed in approximately 10 min or less (considerably less without metadata processing), easily achieving this goal. Third, we want the cloud to be able to perform its computation efficiently. Suppose that the Pseudo-IDs are rotated upon a new contact, as in [5]. Empirical observation at our university shows that an average of 20 contacts are revealed in contact tracing, so it is reasonable to assume that 20,000 infected users' Pseudo-IDs being broadcast represents 500 infected users (40 contacts per user). Assuming

that 1% of the population are infected users, this gives us a total population of about 50,000. With these conditions, a user's CPSI calculation can be performed by the cloud in about 10 min. Then the entire population's contact tracing can be computed in 12 h with 24 24-core machines (or 12 48-core machines), though this is a slightly optimistic estimate.

We conclude that our CPSI-based protocol can compute overnight contact tracing for a population of thousands of people (e.g., a neighborhood, college, or small city). This scale is useful for contact tracing on small populations, e.g., college communities [7,27], small cities. Our implementation has such a limited scalability due to the rigidity of the B/FV scheme [15] and Microsoft SEAL [36] that we used for ease of prototype system development. In SEAL's B/FV parameter selection, users are forced to use a large plaintext space, which made predicate precomputation/calculation slow. This does not indicate the design of our protocol has low scalability. Other schemes and libraries (e.g. BGV [6] and PALISADE [13]) allow the use of smaller plaintext spaces. If such schemes/libraries are combined with our novel dual plaintext space technique (see Sect. 4.5), our CPSI can be much more efficient and thus scalable. Also note that our implementation is only a proof-of-concept; a fully optimized implementation will be much more efficient. Real-life use of this protocol can achieve much greater scalability by utilizing strategies such as parallelization, cloud computing, and hardware acceleration to improve homomorphic computation [35,39]. Due to the lack of other works' experimental runtime data and the relatively high degree of security and functionality of our scheme, a direct experimental comparison to other work cannot be made.

7 Extensions for More Sophisticated Applications

7.1 Duration-Based Conditioning

Recent CDC guidelines state that people are at risk if they are in contact with infected people for over 15 min in any 24-h windows [16]. Conditions like this can be accounted for by using CPSI.

The homomorphically encrypted intermediate values d'_i can be saved and reduced to a Boolean value as $b_{d'_i} = 0$ when $d_i = 0$ and $b_{d'_i} = 1$ otherwise, using the interpolation discussed in Sect. 5.2 [40]. The duration of each contact can be recorded by a user's phone as dur_i. We can then homomorphically compute $S = \sum_i b_{d'_i} \cdot dur_i$, an encryption of the total duration of a person's exposure. With batching, computing this may require the use of ciphertext rotation [19]. The cloud server can then use the threshold predicate $p = p_{pr}(S)$ to test if $S \geq 15$ in CPSI. The calculation of S requires a multiplicative depth of 2 with windowing, and the calculation of p may require a depth of up to $log_2(t)$. The other parts of CPSI can be calculated with a depth as small as 3, with the final multiplication taking place after the predicate's inclusion, so the first two levels of depth are not dominating terms in the total depth for reasonable t larger than 4. Thus the total required depth is $log_2(t) + 1$.

7.2 Multi-hop Tracing

Consider the scenario of multi-hop contact tracing: if an infected person Charlie was found to have a contact with a person Bob, then it is desirable to detect contacts of Bob before and after that contact to find multi-hop contacts. In existing approaches, entirely rerunning new iterations of the protocol is required for multi-hop tracing. We aim to support the multi-hop tracing, in not forcing uninfected users to participate in multiple rounds of the protocol until and unless they are identified as a contact, while identifying the contacts from backward and forward tracing separately. We can accomplish this by modifying the second variant of our protocol, so that additional rounds of contact tracing can be performed without needing additional participation from unexposed users.

Conditional Contact Tracing with Multi-Hops

Let $N_h \geq 2$ be the number of hops from confirmed infected users that we wish to trace. Then the number of additional rounds N_r is $N_h - 1$.
Parties, Setup, ID Collection, Initial Encryption and Upload: These phases are as originally formulated in Figure 4, with the modification of timestamps being encrypted (not necessarily homomorphically) and attached to all Pseudo-IDs.
Initial Intersection: C performs the CPSI computation as in Figure 2, and returns the result with corresponding encrypted timestamp to the TEE of PHA_M. The result and timestamp are decrypted internally, and infected users as well as the time of contacts are revealed to the TEE.
Multi-Hop Tracing: For $i = 1$ to N_r:

1. The TEE collects broadcast Pseudo-IDs from each previously identified user for preprocessing. The IDs are grouped into sets PRE_i and $POST_i$, based on whether the ID was broadcast before or after the contact of the identified user. (In case of multiple contacts, Pseudo-IDs can be included in both sets.) The TEE then homomorphically encrypts these sets, and sends them to C.
2. For each user U, C will homomorphically compute the CPSI of U's contacts with both PRE_i and $POST_i$, and forward the results to the TEE of PHA_M.
3. The TEE will decrypt the results. From this, the TEE learns who has been in contact with an indicrect contact of a COVID-positive individual, and whether that contact occurred before or after the contact with an infected person. The contacts from PRE_i and $POST_i$ will be used as the individuals whose broadcast Pseudo-IDs comprise PRE_{i+1} and $POST_{i+1}$, and those individuals can be notified to upload their broadcast Pseudo-IDs.

Final Results: After completing all N_r rounds, the TEE of PHA_M returns the result of exposure for all participants not notified in a previous round.

Fig. 6. A CPSI-based protocol for multi-hop contact tracing.

The TEE is already utilized for preprocessing. With minimal extra overhead, it can be used to selectively choose which data is used for different CPSI computations. Consider the earlier example, where Bob would be identified as a close contact of the infected Charlie. When Bob sends his data to the TEE for preprocessing, he can also upload timestamps as metadata. Suppose Bob and Charlie met at $ts_{b,c}$. Then the TEE can sort Bob's broadcast Pseudo-IDs into a set PRE_2 or $POST_2$, depending on if they happened before or after $ts_{b,c}$. (PRE_i is the set of contacts i hops away from the initial set.) After the sets PRE_2 and $POST_2$ have been aggregated from all users who were in contact with those individuals confirmed to be infected, they are preprocessed, homomorphically encrypted, and sent (separately) to C. Then C can rerun the CPSI computation

against these two sets, and return the results to the TEE. This shows the TEE who was in contact with Bob (and others like him who contacted an infected individual) before and after they contacted an infected individual. By having uninfected users also attach timestamps, the TEE will additionally learn when that secondary contact happened. The above process can then be repeated to form sets PRE_3, $POST_3$, PRE_4, $POST_4$, etc. In practice, only a few rounds would be necessary. The above process is detailed in Fig. 6. Unlike some existing approaches [24], our multi-hop tracing does not require uninfected and uncontacted users to perform any additional computation or communication.

8 Conclusion

We present a secure protocol for COVID contact tracing based on the use of our novel CPSI protocol and TEE. Our contact tracing protocol is extended beyond basic conditioned contact tracing to filtering by duration with CPSI, and is also expanded to forward/backward multi-hop tracing. With our implementation, a users' required computations can be completed in as little as 9 min, and the cloud's per-user computation can be as little as 11 min for 20,000 total Pseudo-IDs broadcast by all infected users, with negligible overhead for TEE preprocessing and decryption.

Acknowledgement. This work was supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) via contract #2020–20082700002. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor. The authors also thank Dr. Alex Perkins (Department of Biological Sciences, University of Notre Dame) for his helpful comments.

References

1. Altuwaiyan, T., et al.: Epic: efficient privacy-preserving contact tracing for infection detection. In: IEEE ICC, pp. 1–6 (2018)
2. Apple and Google. Privacy-Preserving Contact Tracing (2020). apple.co/3bFFWzp
3. Baumgärtner, L., et al.: Mind the GAP: security and privacy risks of contact tracing apps. arXiv preprint (2020). arXiv:2006.05914
4. Bay, J., et al.: BlueTrace: a privacy-preserving protocol for community-driven contact tracing across borders. Tech. Rep. GovTech-Singapore (2020)
5. Bell, J., et al.: Tracesecure: towards privacy preserving contact tracing. arXiv preprint arXiv:2004.04059 (2020)
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM TOCT **6**(3), 1–36 (2014)
7. Burke, L.:. New variant meets its first university (2021)
8. Chen, H., et al.: Fast private set intersection from homomorphic encryption. In: ACM CCS, pp. 1243–1255 (2017)
9. Chen, H., et al.: Labeled PSI from fully homomorphic encryption with malicious security. In: ACM CCS, pp. 1223–1237 (2018)

10. Cho, H., Ippolito, D., Yu, Y.W.:. Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs. arXiv preprint arXiv:2003.11511 (2020)
11. Ciampi, M., Orlandi, C.: Combining private set-intersection with secure two-party computation. In: Catalano, D., De Prisco, R. (eds) Security and Cryptography for Networks. SCN 2018. Lecture Notes in Computer Science, vol. 11035. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_25
12. Costan, V., Devadas, S.: Intel SGX explained. IACR Cryptol. ePrint Arch. **86**, 1–118 (2016)
13. Dave C., Kurt R., Yuriy P., Ryan, G.:. The PALISADE lattice cryptography library (2020). bit.ly/35Bthtz
14. De Cristofaro, E., Gasti, P., Tsudik, G.: Fast and private computation of cardinality of set Intersection and Union. In: Pieprzyk, J., Sadeghi, A., Manulis, M. (eds.) CANS 2012. LNCS, vol. 7712, pp. 218–231. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35404-5_17
15. Junfeng Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch., 144 (2012)
16. Centers for Disease Control and Prevention. Appendix A - Glossary of Key Terms (2020). bit.ly/2LljkK0
17. Garmin. Project Tesserae powered by Garmin (2018). bit.ly/3nI2yBC
18. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES Circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_49
19. Halevi, S., Shoup, V.: Design and implementation of a homomorphic-encryption library. IBM Research (Manuscript) **6**, 12–15 (2013)
20. Ion, M., et al.: Private intersection-sum protocol with applications to attributing aggregate ad conversions. IACR Cryptol. ePrint Arch. 738 (2017)
21. Lindell, Y.: How to simulate it-a tutorial on the simulation proof technique. Tutorials on the Foundations of Cryptography, pp. 277–346 (2017)
22. Liu, S., Jiang, Y., Striegel, A.: Face-to-face proximity estimationusing bluetooth on smartphones. IEEE Trans. Mobile Comput. **13**(4), 811–823 (2014)
23. Lounis, K., Zulkernine, M.: Attacks and defenses in short-range wireless technologies for iot. IEEE Access **8**, 88892–88932 (2020)
24. Michael, K., Abbas, R.: Behind covid-19 contact trace apps: the Google-Apple partnership. IEEE Consumer Electronics Magazine **9**(5), 71–76 (2020)
25. Mofrad, S., Zhang, F., Lu, S., Shi, W.: A comparison study of intel sgx and amd memory encryption technology. In: HASP, pp. 1–8 (2018)
26. Morgan, A.U., et al.: Remote monitoring of patients with covid-19: design, implementation, and outcomes of the first 3,000 patients in COVID Watch. NEJM Catalyst Innovations in Care Delivery, **1**(4) (2020)
27. Nietzel, M.: Duke University suddenly imposes week-long stay-at-home order on all undergraduates (2021)
28. Government of Singapore. TraceTogether (2020). www.tracetogether.gov.sg
29. Benny, P., Eyal, R.: Hashomer-a proposal for a privacy-preserving bluetooth based contact tracing scheme for Hamagen (2020)
30. Benny, P., Thomas, S., Christian, W., Udi, W.:. Efficient circuit-based PSI via cuckoo hashing. In: EUROCRYPT, pp. 125–157 (2018)
31. Benny, P., Thomas, S., Michael, Z.: Faster private set intersection based on {OT} extension. In: Usenix Security, pp. 797–812 (2014)
32. Ramesh, R., et al.: Apps gone rogue: maintaining personal privacy in an epidemic. arXiv preprint arXiv:2003.08567 (2020)

33. Raskar, R., Pahwa, D., Beaudry, R.: Contact tracing: holistic solution beyond bluetooth. IEEE Data Eng. Bull **43**(2), 67–70 (2020)
34. Reichert, L., Brack, S., Scheuermann, B.:. Privacy-preserving contact tracing of covid-19 patients. IACR Cryptol. ePrint Arch. 375 (2020)
35. Riazi, M.S., et al.: HEAX: an architecture for computing on encrypted data. In: ACM ASPLOS, pp. 1295–1309 (2020)
36. Microsoft SEAL (release 3.6) (2020). bit.ly/3qgKCjd
37. Singh, P., et al.: Ppcontacttracing: a privacy-preserving contact tracing protocol for covid-19 pandemic. arXiv preprint arXiv:2008.06648 (2020)
38. Taassori, M., et al.: Vault: reducing paging overheads in SGX with efficient integrity verification structures. In: ASPLOS, pp. 665–678 (2018)
39. Takeshita, J., et al.: Algorithmic acceleration of B/FV-Like somewhat homomorphic encryption for compute-enabled RAM. In: Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 66–89. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81652-0_3
40. Tan, B.H.M., et al.: Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields. IEEE TDSC (2020)
41. Tang, Q.: Privacy-preserving contact tracing: current solutions and open questions. arXiv preprint arXiv:2004.06818 (2020)
42. Trieu, N., et al.: Epione: lightweight contact tracing with strong privacy. arXiv preprint arXiv:2004.13293 (2020)
43. Wang, X.S., et al.: Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In: ACM CCS, pp. 492–503 (2015)
44. Wu, J., et al.: {BLESA}: spoofing attacks against reconnections in Bluetooth low energy. In: 14th {USENIX} Workshop on Offensive Technologies ({WOOT} 20) (2020)
45. Yasaka, T.M., Lehrich, B.M., Sahyouni, R.:. Peer-to-peer contact tracing: development of a privacy-preserving smartphone app. JMIR Mhealth Uhealth, **8**(4), e18936 (2020)
46. Yoneki, E.: Fluphone study: virtual disease spread using haggle. In: CHANTS, pp. 65–66 (2011)