CoTracer and RokWall: Privacy Preserving University Health Apps for COVID-19

Vikram Sharma Mailthody*‡, James Wei*‡, Nicholas Chen*‡, Mohammad Behnia‡‡, Ruihao Yao*‡, Qihao Wang*‡,
Vedant Agarwal*‡, Churan He*‡, Lijian Wang*‡, Leihao Chen*‡, Amit Agarwal*‡, Edward Richter*‡,
Wen-Mei Hwu‡, Christopher W. Fletcher‡, Jinjun Xiong†, Andrew Miller‡, Sanjay Patel‡
† Cognitive Computing Systems Research, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 10598
‡ University of Illinois at Urbana-Champaign, Urbana, IL 61801
* Contributed Equally

***Abstract*—COVID-19 has fundamentally disrupted the way we live. Government bodies, universities, and companies worldwide are rapidly developing technologies to combat the COVID-19 pandemic and eventually enable the reopening of society. Essential analytics tools such as digital contact tracing, super-spreader event detection, and exposure mapping require collecting and analyzing sensitive user information. The demand for such powerful data-driven applications necessitates secure, privacy-preserving infrastructure for computation on personal data.**

In this paper, we detail two privacy-preserving infrastructures being developed at University of Illinois Urbana Champaign to track and mitigate the spread of COVID-19. First, we present CoTracer, a decentralized digital contact tracing system leveraging Google/Apple APIs. Second, we introduce the RokWall architecture for privacy-preserving data analytics on sensitive user data. We discuss the motivation and architecture of these systems, design decisions, various threat models considered, and the challenges we experienced in developing a production-ready system for sensitive data analysis.

I. Introduction

COVID-19 has fundamentally disrupted the way we live. Countless organizations, including government bodies, academic research groups, and companies, are developing and deploying technological solutions to combat the spread of COVID-19 [1], [2], [4], [6], [9], [14], [16]. Technology and data play an important role for safely reopening our communities. Technologies such as such as digital contact tracing, super spreader event detection and tracking, exposure mapping, migration mapping, live queues at testing locations, risk assessment, and effective stress management [8] have been developed to help mitigate the spread of disease. These techniques require the collection of sensitive user information, introducing a delicately balanced trade-off between data driven functionality and personal privacy. As more user information is disclosed, the application can provide more responsive, personalized user experiences; yet the privacy risk increases accordingly [9]. This necessitates trustworthy and secure mechanisms to reduce the risk of sensitive information becoming compromised [5], [7].

We believe a university can play a crucial role in this area as they are viewed as relatively trustworthy entities by the public [15]. University-led apps can create legitimate trust by establishing public auditors and thorough review processes. Furthermore, universities are not reliant on monetizing private data. We expect this credibility to encourage widespread adoption.

In early summer 2020, University of Illinois Urbana Champaign announced plans to resume on-campus instruction for the fall semester. In order to reach this ambitious goal, the university has taken several initiatives, including the development of technologies for managing the spread of COVID-19 using the University of Illinois Urbana Champaign RokWire platform. Started in 2018, RokWire's goal is to serve as an open-source platform for smart communities, such as campuses, cities, and organizations. The prime directive of RokWire is to provide valuable functionality to users while enabling fine-grain control of their data. RokWire does not monetize individual user data and is audited by public authorities. With the emergence of COVID-19, we envisioned that RokWire should become a platform for a scalable, privacy-preserving computing infrastructure.

In this paper, we describe two secure privacy-preserving capabilities developed in the RokWire platform. First, we describe CoTracer, our decentralized system for digital contact tracing based on the recently released Google/Apple protocol. We have overcome significant implementation hurdles to develop a solution that is ready to be deployed at scale, addressing several significant gaps in existing protocols. We provide details on technical challenges, existing shortcomings, and integration into a broader campus workflow.

Data driven insights are critical to forming responsive strategies for pandemic management and public policy. CoTracer provides anonymous, digital contact tracing, but is quite limited from an analytics perspective. Its decentralized architecture makes it difficult as a platform for generating aggregate data. We describe RokWall, a centralized system that addresses this limitation and can perform privacy-preserving data analytics on user sensitive data. RokWall can perform advanced analytics such as super spreader event detection and tracking, exposure mapping, and risk assessment with minimal exposure of a user's private data.

While RokWire provides user-controlled privacy settings, as most apps do, it is in general still up to the service provider to ensure this privacy is respected. To address this, we build RokWall using secure enclaves to ensure that the user privacy settings are maintained. We discuss the overall architecture of RokWall using the Intel SGX platform [3] and provide detailed description of several different threat models considered. We present several technical challenges we faced (and continue to face) while building the RokWall architecture.

Although developing infrastructure is challenging, it is

necessary to be transparent to the public with how their data is handled, and take their opinions, feature requests and feedback into account early on. Trust is required for mass adoption and integrating early feedback is helpful to ensure users are comfortable with RokWire's data policies. To better understand the public's perception, we presented our efforts from the university over a series of public seminars. In this paper we discuss the feedback we received from these sessions.

To summarize, we make following main contributions:

1) CoTracer, a digital contact tracing application based on the Google/Apple protocol for privacy-preserving exposure notification.
2) RokWall, a privacy architecture for secure privacy-preserving computing using secure enclaves.
3) Discuss several technical challenges we face in developing secure privacy-preserving computing.

We hope this paper fosters discussion within the research community about developing a privacy-preserving computing infrastructure.

II. CoTracer: A Decentralized App

Exposure notification technologies have become integral components of public health strategies worldwide to curb the spread of COVID-19 infections, often as a digital supplement to manual contract tracing. Early success at staving off the virus in South Korea and Singapore prompted researchers worldwide to develop protocols for effective contact tracing through smartphone devices without significantly compromising individual's privacy. As with other public health strategies to combat the pandemic, such as facemasks and social distancing, exposure notifications rely on high community adoption rates. Simulation-based studies estimate that nearly 60% of individuals within a region need to be actively using digital exposure notification in order to be effective [8]. Our goal with the RokWire project was to develop an exposure notification solution that could be deployed at scale to around 100,000 users within the University mobile app. The University requested a production ready system by August 2020 to inform public health policies throughout the Fall semester.

The app is built around a simple concept: it holds a digital version of your COVID-19 health status. If you are tested on campus, or by a provider in the surrounding community, your COVID-19 test results are, with your consent, pushed onto your device. The app then manages the test results by invalidating them after a certain time period determined by county health officials, say 5 days, prompting the user to get re-tested [13]. The results can also be invalidated by a recent encounter with someone whom is later determined to have been infectious at the time, through digital exposure notification. To enter a University space, for example, you might be asked to present your digital health status to show that you pose minimal infection risk to others. Those who opt-out would be asked to show test results by paper or digital image[1].

As security-conscious consumers ourselves, we adopted a privacy-centric philosophy from the onset. We chose decentralized, privacy-preserving protocols when available. We opted to keep our codebase open-source, for additional transparency. We adopted a minimal data policy, gathering as little data as possible to meet the functionality of the application.

The CoTracer architecture involves four components: (a) exposure notification, (b) integration with testing facilities, (c) administration panel for public health authorities, and (d) an upload server for positive diagnoses. The complexity of our system is primarily in the exposure notification system, so we will focus our discussion in this paper on that component, with briefer discussions on the others.

The design space for exposure notification includes a choice of proximity estimation (i.e, Bluetooth, WiFi, ultrasonic, GPS, etc), centralized vs. decentralized vs. hybrid architecture, cryptographic protocol, etc. Our approach was to leverage the ongoing work by various security experts and communities worldwide, who were creating open-source protocols for digital exposure notification.

We evaluated three protocols in depth, namely the Temporary Contact Number (or `TCN`) protocol [4], Decentralized Privacy-Preserving Proximity Tracing (or DP-3T) [16] and the Google/Apple Exposure Notification (or GAEN) protocol [1], [2], each of which we summarize briefly below.

TCN protocol works on the principle of generating Temporary Contact Numbers (or `TCN`), a psuedo-random identifier derived from a seed every 15 minutes. A unique `TCN` is generated, exchanged via Bluetooth Low Energy (or BLE) and stored when two devices come in close proximity. When a user tests positive, a report is sent to a centralized server with the list of `TCN`s exposed. This report is pulled by each user devices to determine matching `TCN` to see if the user has been exposed.

DP-3T protocol differs from the TCN protocol on how anonymous IDs are generated (random seed with deterministic hash + truncation vs asymmetric key-pair with deterministic hash ratchet in TCN), what information is reported (`EphID` and seed of all relevant epochs vs public key with start and end time and `tck` for regenerating `TCN` for timeblock) and what information is stored (`hash(EphIDs)` and epoch `i`, proximity, duration and coarse time indication vs `TCN` value).

GAEN protocol leverages concepts from the DP-3T and TCN protocols, including the use of BLE for proximity detection, with key differences in anonymous ID generation (Rolling Proximity Identifiers, or `RPIs` generated through Temporary Exposure Keys, or `TEKs` every 10 minutes) and reporting of tested positive cases (`TEKs` and a timestamp represented as an epoch interval number). Unlike DP-3T and TCN, the GAEN protocol is publicly described, but is closed-source and access to the implementations are only granted to public health authorities operating at the state or country-level. At the time of this writing, they were not available to our team.

We evaluated these protocols in April and May 2020, at a time when these concepts were still undergoing intense development and any codebases were not yet mature. The open-source code had known shortcomings, such as failing in BLE background mode for iOS devices. We decided to adopt the GAEN approach, and build our own implementation, in case we received API entitlements from Google and Apple due to our affiliation with a large University[2].

[1]None of this should be surmised as official University of Illinois Urbana Champaign Policy. That is still being determined at time of writing.

[2]We have not yet received such entitlements

In the overall user workflow of the app, an individual can get tested on campus using one of several testing sites. The user presents their University ID when a test is administered, thereby linking their results to a University ID number. Since the user must authenticate within the app using their University credentials, their test results can be linked to the user via the app. The user is notified by the app once the test results are available, typically within an hour. Test results can be encrypted using the user's public key and pulled onto the user's device with the user's consent.

If the diagnosis is positive, the user can choose to upload a history of their `TEKs` onto a diagnosis upload server. The published `TEKs` then enable RokWire to notify users who came into contact with the positive individual. Apps of users participating in exposure notification will periodically download published TEKs from the diagnosis upload server, decode the TEKs into rolling proximity identifiers, and check for matches with `RPIs` stored in the local device database. As a further security measure, the upload server will use one-time codes that are electronically shared with the testing sites. A single code is provided alongside each test result to the user device, which is then used to establish that a chain-of-authenticity from the testing cite to the upload server, via the user device.

In the case that a matching `RPI` is found, an exposure score is calculated using a variety of parameters such as duration of exposure, reception and transmission strength of the Bluetooth signal, an estimated onset date of infection, models of the efficacy of testing, etc. How such parameters can be used to estimate the risk of infection and is an on-going area of work both within University of Illinois Urbana Champaign and elsewhere [14]. RokWire contains an admin control panel, provided to public health authorities, containing a limited ability to adjust the parameter weighting system used to score an exposure. If the score is above the threshold, indicating exposure risk, then the user's most recent test result is invalidated, provided the test occurred prior to the exposure, prompting the user to be retested.

Complementing this workflow is the exposure notification functionality, running continuously on each device. CoTracer directly follows the specification defined by the GAEN protocol in generating and exchanging exposure keys. Every day, each user generates a unique Temporary Exposure Key which constructs a user's Rolling Proximity Identifier Key and subsequent `RPIs` to be exchanged with other users. In addition, the TEK generates an Associated Encrypted Metadata (`AEM`) Key which, along with an `RPI`, can be used to encrypt a few bytes worth of optional metadata.

Each user broadcasts their `RPI` and corresponding `AEM` with a rolling period of approximately 10 minutes. Whenever a contact is registered by being within the range of the device's effective Bluetooth range, the device saves the found `RPI` with the duration and Bluetooth received signal strength - known as RSSI - of the contact to local storage. The device also securely saves the user's daily `TEK` and a timestamp to be uploaded to a server in case the user tests positive for COVID-19.

A. Security

The security and privacy implications of such exposure notification protocols have been heavily examined by experts, including the DP3T and TCN communities [4], [16]. We briefly summarize the major salient threat models here, for completeness of discussion. We separate these threats into two categories: 1) *inherent attacks* faced by all Bluetooth proximity tracing systems, and 2) *protocol-dependent attacks* which depend on how each protocol generates and exchanges its anonymous identifiers.

Inherent security considerations: Whenever a user gets notified of a possible exposure event, they may be able to identify the infected person by correlating who they interacted with at each time with the time of interaction with an infected person. Even if an application obfuscates the timing, an attacker can create multiple accounts or uses multiple phones at different times, this information can be revealed once the attacker receives an exposure notification on one of their devices. This threat can be further exacerbated as attackers can log other data, such as location, from infected users as well. Moreover, apps that solely rely on Bluetooth to exchange keys can be susceptible to certain broadcasting threats. If an attacker were to set up powerful transmitters to enhance their effective Bluetooth range, false contacts could potentially be logged. Alternatively, an attacker could set up a Bluetooth jammer that could disrupt communication between devices.

Protocol-dependent security considerations: First, anonymous identifiers must not be linkable to one another nor to the transmitting device. The former is achieved in all protocols discussed through cryptographic pseudorandomness while the latter requires at a minimum the synchronization of rotations of Bluetooth MAC address and anonymous identifier. Second, there remains a possibility of replay attacks, where adversaries may record anonymous ids in one area and replay them entirely somewhere else. A proposed solution to this is to not prevent the replay attack from occurring, but actually inhibit notification to a user in the end [10]. All three protocols mitigate this issue to some extent, incorporating timestamps while checking for exposed matches.

B. Implementation Challenges

We selected our approach to exposure notification with an emphasis on wide-scale deployment. Ideally, the protocol could be adopted with minimal impact on the end user. A unique advantage of using the Google/Apple API is its device-level control of Bluetooth and background settings. The TCN and DP3-T approaches are defined at the application level, and thus need to contend with limited access to the Bluetooth stack and application state issues, which became major limitations to the applications developed with their protocols. We do not have entitlements to the GAEN API, so we embarked on a path to address the known issues suffered by DP3-T and others. Below, we describe some of the challenges encountered while trying to implement a production ready system at an application level.

iOS Background Advertising: Background iOS applications restrict Bluetooth advertisement packets coming from the device. Namely, instead of advertising a standard service UUID, all advertisement is moved to an "overflow area" where it is only discoverable by a device explicitly scanning for

TABLE I. AVERAGE BATTERY DRAIN PER HOUR OF COTRACER APP

Device	CoTracer On	CoTracer Off
Google Pixel 3	3.02%	1.96%
iPhone X	4.15%	0.59%

it. Since all iOS background apps share the same overflow area, there is no guarantee that the app is advertising a preset bitmask. Moreover, there is the possibility of collisions if the same bitmask is set, meaning an app may detect a completely different service than what was originally intended. Currently, we don't have a concrete solution to this; however, the likelihood of this occurring is very low, as few apps (if any) would also advertise Bluetooth in the background.

iOS-iOS Background Communication: Callback can be easily set up in Android to detect the overflow bitmask of an iOS background device. On iOS devices, however, this callback would only be triggered if the screen is turned on and beacon ranging is enabled. We found this can be circumvented by sending a local notification, which will illuminate the screen for 10 seconds at the expense of battery life.

Bluetooth Mac Address Changes: It's absolutely essential to have Bluetooth MAC rotations aligned with each RPI change. Otherwise, an attacker can easily link RPIs together coming from a single user. Unfortunately, as of Android 6.0 and iOS 8, an application cannot control the timing of its Bluetooth MAC address changes or even identify when this change occurs. While testing CoTracer on Android, we found the Bluetooth MAC address to change every time our advertising packet changed; however, the same could not be seen on iOS. This still remains an unresolved issue for us.

iOS Background Execution: With iOS devices, we found it difficult to keep an app from being suspended by the OS when in background mode. Suspended apps will not be able to record or transmit `RPIs`. To address this problem, we activate location services to keep the app in background mode indefinitely. We don't use location data within the CoTracer app, but require access to keep the app from being suspended.

Battery Efficiency: In addition to keeping a service alive in the background, constant Bluetooth scanning and advertising takes a substantial toll on battery life. While the GAEN protocol sets scanning intervals to be up to 5 minutes apart, Android and iOS SDKs provide little control over these intervals. Android provides 3 distinct scan settings although the actual times may differ by manufacturer, while no parameterizations are documented for iOS. Even in the most power-saving state, scans occur every couple seconds. Table I shows the average battery drain in % per hour across iOS and Android platforms. We set CoTracer to be the only application running with it constantly scanning another device. However, these numbers may vary by other factors including, device usage, how many other devices are scanned, OS level, and device model.

III. ROKWALL: A SECURE ENCLAVE

The CoTracer application demonstrates decentralized computation in RokWire platform by avoiding usage of sensitive data. However, desirable queries such as exposure mapping, super-spreader event detection, and density heatmaps require centralized analysis. A centralized infrastructure requires users to place greater trust in service providers' benevolence and honesty. While reasonable for a highly transparent university

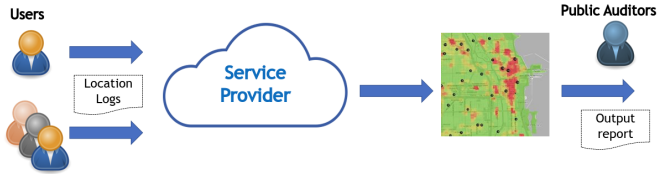

Fig. 1. Exposure mapping application on sensitive location data. User uploads sensitive location logs to the service provider. The service provider generates an exposure map along with a output report for public auditors to audit.

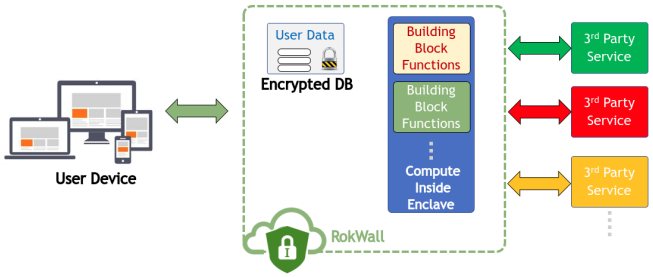

Fig. 2. Overall strawman architecture of RokWall is shown.

organization, users may justifiably remain skeptical of private businesses or other third parties accessing their data within the RokWire system. Thus, we need to develop secure privacy-preserving computing infrastructure inside RokWire for centralized analytics.

RokWall is guided by the following fundamental principles: *(a)* Sensitive user data is only used by services authorized by the user. Users have assurance that a third-party service provider cannot exploit beyond the declared capabilities. *(b)* No party, including service providers and manufacturers, can access data beyond the computation's output, *(c)* Users or public auditors can review the code bases, verify program binaries and ensure it meets all security and privacy guidelines.

Exposure Mapping Application: We present a COVID-19 exposure mapping application in Figure 1 as an example of computing on sensitive data. Exposure mapping aggregates user location logs to calculate a heat map for a region, visualizing the risk of infection exposure. This application helps health authorities to assess the likelihood of super-spreader events and warn the general public of high risk areas. GPS location data is highly sensitive, so the service provider should follow previously discussed fundamental principles: *(a)* Only perform a limited set of queries on the user's location data to preserve privacy, *(b)* Ensure the data is secure and is used only for exposure mapping application purposes, and *(c)* Enable auditors to verify these guidelines with public information such as output report to hold the service provider accountable.

To this end, we present RokWall, a secure architecture (see § III-A) for sensitive data computation. We apply RokWall to COVID-19 exposure mapping while preserving the desired security and privacy guarantees for user location information. We analyze various threat models (see § III-B) considered for the exposure mapping application and RokWall's protection against various attack vectors. Finally, we present various technological challenges (see § III-C) faced during deployment and provide potential solutions.

A. RokWall Architecture

Figure 2 provides a high level illustration of the RokWall architecture. The RokWall architecture supports sensitive data computation by leveraging secure enclaves implemented by

Intel's SGX. Data analysis, such as exposure mapping, people density mapping or super-spreader detection, occurs inside these secure enclaves. Each individual data analytics function is referred to as a "building block function" and are statically linked with enclaves. Each building block function publicly declares a hash of the program binary and each secure enclave generates an output public key. Source code of all the building block functions and APIs are planned to be open sourced and thoroughly audited.

RokWall allows third-party services to upload information, such as a health authority's API updating test results of a specific user using a secure channel. Users upload sensitive data to the RokWall server using a secure encrypted channel such as Transport Layer Security (TLS) along with the hash of the program binary for which the user provides the capability to work on the data and the enclave public key for which the user permits the execution. Inside RokWall, user data is stored in an encrypted database. During the query execution, only the building block function or third-party services whose hash of the program binary matches with the user-approved application can temporarily decrypt and access data within the secure enclave. Critically, this guarantees that unencrypted data never leaves an enclave. An unauthorized building block execution will result in the generation of a useless result.

Remote attestation in RokWall: Remote Attestation allows cryptographic verification of the code allegedly executed inside an SGX enclave. RokWall uses a 3 party EPID based remote attestation mechanism. We reduce the verification effort at the end-user devices by publicly providing verified attestation report generated by RokWall enclave. This report would contain information about the enclave code (given by MRENCLAVE) as well as the public-private key pair generated during enclave initialization. Auditors (or even users) can verify that the MRENCLAVE information in the report matches the publicly available MRENCLAVE generated by building/compiling the enclave code, vetted by interested parties.

Exposure Mapping Function in RokWall: Implementing the exposure mapping function inside RokWall is straightforward. Users upload the sensitive location logs using TLS to the RokWall encrypted database along with hash of exposure mapping function binary and enclave public key. On a regular interval (in this example, once per day), exposure mapping function generates and publishes a heat map as output on a public server along with signature and report for remote attestation. The user app can pull the output from the public server, then verify the signature and report if needed.

B. End-to-end Chain Of Trust In RokWall

Security and privacy guarantees are primary principles of the RokWall design. We consider a three-tiered threat model: (1) *network attackers*, (2) *client attackers* and (3) *service provider attackers*. To safeguard against network attackers, clients communicate with the RokWall server via TLS channel. Unfortunately, we cannot currently prevent client attackers from running malicious code or flooding the system with spoofed data. This is a known problem on systems that do not require user verification. One possible solution, employed by electronic voting systems [11], allows an authority to register public keys of users. University officials could distribute public keys to community members interested in using the service.

Service provider attackers can be classified into three sub-categories: *(1) server-software*, where a service provider runs malicious user-level software, *(2) server-kernel*, where a service provider runs malicious kernel-level software, and *(3) server-hardware*, where a service provider has physical access to the server hardware.

Server-software attacks: Server-software level attacks assume that the service provider is limited to user-level privileges. This includes writing and running malicious code, but excludes kernel privileges or hardware attacks. Server-software attacks can generally be prevented by using SGX enclaves. Remote attestation enforces transparency and enables public auditors to review code, while data sealing ensures that the service provider cannot export raw, decrypted user data. One remaining attack vector is an isolation attack, where a service provider runs the exposure map query with only a single victim user's location logs. This query yields a heatmap exposing the victim's location history, even though the code would correctly pass an audit.

RokWall addresses this with two step solution: *(1)* employ non-volatile counters such that a location log can only be used for a heatmap one time, and *(2)* output a hash of location logs included so a user can verify that their data was used for generating a given heatmap. Then, if a service provider commits an isolation attack, the victim's data will necessarily not be present in the official heatmap. If a user finds that their data is not present in an officially published heatmap, they can then report the service provider to RokWall administrators.

Server-kernel attacks: Server-kernel attacks expand upon user-level code execution and permit the attacker to inspect memory management within SGX. This level of attack can theoretically allow privileged side channel attacks, exposing memory access patterns even in sealed data [17]. We avoid leaking information to these attackers by ensuring data oblivious execution and guaranteeing a constant runtime regardless of input size. In the case of exposure mapping, this entails unsealing and resealing the entire heatmap every time data is uploaded. RokWall currently does not defend against microarchitectural attacks (like cache-timing attacks) as they pose far more sophisticated adversaries.

Server-hardware attacks: Server-hardware attacks involve physically probing or tampering with the enclave's system hardware. We generally expect the cloud service provider to ensure physical security of their servers. We are still investigating additional counter measures to address these attacks and will address them in the future.

C. Technical challenges

Computation on sensitive data has numerous constraints that are only observed when implementing a production ready system. We describe some of the challenges we encountered while developing RokWall system and propose a few solutions.

Monotonic counter on Intel Servers: Rollback attacks present a security problem for enclave solutions. An adversary OS can restart the service with an outdated version of sealed data and leverage it to leak user information. Intel provides a native SGX monotonic counter service to tackle this problem. However, SGX cloud services such as IBM Cloud and Microsoft Azure are currently built on Intel Xeon E3 server-grade

processors, which do not support the Intel Management Engine required for this service. Alternatives to SGX's monotonic counter have been proposed in the form of distributed rollback protection systems such as ROTE [12]. Other solutions include the migration of the counter service to a third-party trusted source or a BFT distributed network such as CCF.

SGX Memory management: Intel SGX provides data sealing for encrypting and saving confidential enclave information to persistent storage. Sealing comes in two forms, using Enclave Identity and Signing Identity. Data sealed with Enclave Identity (MRENCLAVE) will only allow other instances of the same enclave to unseal, whereas Signer Identity allow other versions and builds of the enclave to unseal. However, Intel SGX sealing is not intended for large data objects. Aside from performance degradation, crossing EPC memory bounds require memory management from the enclave itself.

Challenges with Remote Attestation: A major challenge in implementing remote attestation is ensuring reproducible builds between auditors, clients and RokWall server. Inconsistent builds can result in false MRENCLAVE mismatches. Furthermore, the auditors (or users) are required to make sure to use identical backend libraries/packages (as described in the attestation report) in their build process. However, this might be a quite cumbersome task and not auditor-friendly. We are still investigating a reliable, efficient and user friendly scheme to create reproducible builds.

Testing Dataset: When developing the exposure mapping building block, we struggled to find an appropriate, publicly available GPS dataset for simulating infection dynamics. We ultimately decided to test RokWall's location related queries on the T-Drive GPS trajectories data [18]. T-Drive records coordinates for 10000 taxi cabs in Beijing over the course of a week. Some comparative advantages of the T-Drive dataset are its high number of entities, dense population concentration, and high frequency of reporting.

While the T-Drive dataset is sufficient for initial testing, it has several key limitations. Critically, the data isn't perfectly representative of our eventual use cases since the entities are vehicles, rather than people. Taxis are confined to roads and don't enter buildings so we cannot run indoor, intra-building analysis. Moreover, this data can't facilitate algorithmic parameter tuning, such as heatmap granularity or super-spreader event thresholds, because of differences in population density and entity size. Thus, we will likely need to collect organic human data for fine tuning.

IV. Discussion

1) Why design both decentralized and centralized systems?: Contact tracing can be implemented in a decentralized or centralized fashion, which forces developers into making a trade-off between user privacy and analytic capability. While previous works often avoid sensitive user data in favor of decentralized implementations, RokWall enables centralized contact tracing while upholding the sanctity of user privacy.

In order to develop a functional, reliable contact tracing system by start of the Fall 2020 semester, CoTracer leveraged preexisting GAEN APIs in a decentralized system. However, bluetooth-reliant contact tracing carries inherent limitations that can be solved by centralized analysis on user data. GPS data, for instance, can help identify infection hotspots, remedy bluetooth connectivity issues, and enable cross-time location analysis. We envision eventually migrating CoTracer to RokWall and enabling richer analysis via centralized contact tracing.

2) Accuracy of Capturing Exposure Events: Bluetooth can detect an exposure with high accuracy within 2 meters, while the GPS has a tolerance of 10 meters. However, both of these technologies fail to capture with high accuracy if there is a physical obstacle blocking the communication. Additional ultrasound or miniature radar sensors can help if added to user devices. Alternatively, we believe applied machine learning on sensory inputs may be able to detect the unseen.

3) Public Feedback: When working with sensitive data, it is necessary to maintain public transparency and actively seek feedback throughout the development process. To gauge community reception of the Rokwall architecture, we presented our efforts over a series of public university seminars. We found that a majority of the audience was willing to share sensitive information for public health purposes, pursuant to certain privacy conditions. Users particularly expressed interest towards exposure and density mapping functions during our interactive sessions. Regarding privacy issues, participants were primarily concerned with data storage, data usage policy, and the ability to enable or disable specific third-party applications.

V. Conclusion

We presented CoTracer and RokWall architecture being developed at the University of Illinois Urbana Champaign RokWire. We detailed our design methodology and various threat models considered while implementing a production-ready system. We also presented several technological challenges and lessons learned while implementing these systems. We hope this work fosters discussion in developing a privacy-preserving computing infrastructure.

References

[1] "Apple: Privacy-preserving contact tracing."

[2] "Google exposure notifications: Using technology to help public health authorities fight covid-19."

[3] "Intel® Software Guard Extensions. ," 2020.

[4] "TCN Protocol. https://github.com/TCNCoalition/TCN," March 2020.

[5] S. Altmann, L. Milsom, H. Zillessen, R. Blasone, FredericGerdon, R. Bach, F. Kreuter, D. Nosenzo, S. Toussaert, and J. Abelery, "Acceptability of app-based contact tracing for covid-19:cross-country survey evidence," May 2020.

[6] J. Bay, J. Kek, A. Tan, C. S. Hau, L. Yongquan, J. Tan, and T. A. Quy, "Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders," April 2020.

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

[8] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser, "Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing."

[9] T. Li, Jackie, Yang, C. Faklaris, J. King, Y. Agarwal, L. Dabbish, and J. I. Hong, "Decentralized is not risk-free: Understanding public perceptions of privacy-utility trade-offs in covid-19 contact-tracing apps," 2020.

[10] P. Madhusudan, P. Miao, L. Ren, , and V. Venkatakrishnan, "ConTrail: Privacy-preserving secure contact tracing," June 2020.

[11] Madise, Ülle AND Martens, Tarvi, "E-voting in estonia 2005. the first practice of country-wide binding internet voting in the world," in Electronic Voting 2006 – 2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting.CC. Gesellschaft für Informatik e.V., 2006.

[12] S. Matetic, M. Ahmed, K. Kostiainen, A. Dhar, D. Sommer, A. Gervais, A. Juels, and S. Capkun, "ROTE: Rollback protection for trusted execution," in 26th USENIX Security Symposium (USENIX Security 17). USENIX Association, Aug. 2017.

[13] D. Ranoa, R. Holland, F. Alnaji, K. Green, L. Wang, C. Brooke, M. Burke, T. Fan, and P. Hergenrother, "Saliva-based molecular testing for sars-cov-2 that bypasses rna extraction," Cold Spring Harbor Laboratory Press, United States, WorkingPaper, Jun. 2020.

[14] R. Raskar, G. Nadeau, J. Werner, R. Barbar, A. Mehra, G. Harp, M. Leopoldseder, B. Wilson, D. Flakoll, P. Vepakomma, D. Pahwa, R. Beaudry, E. Flores, M. Popielarz, A. Bhatia, A. Nuzzo, M. Gee, J. Summet, R. Surati, B. Khastgir, F. M. Benedetti, K. Vilcans, S. Leis, and K. Louisy, "Covid-19 contact-tracing mobile apps: Evaluation and assessment for decision makers," 2020.

[15] B. Roberts and et al., "Covid-19 and technology perceptions by key demographics," June 2020.

[16] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salath, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, L. Barman, S. Chatel, K. Paterson, S. Capkun, D. Basin, D. Jackson, B. Preneel, N. Smart, D. Singelee, A. Abidin, S. Guerses, M. Veale, C. Cremers, R. Binns, and T. Wiegand, "Decentralized privacy-preserving proximity tracing. https://github.com/DP-3T/documents," April 2020.

[17] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in 2015 IEEE Symposium on Security and Privacy, 2015.

[18] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," 2011.