AFRL-RI-RS-TR-2021-038

PRIVACY-ENHANCED ANDROID RESEARCH AND LEGACY SYSTEMS (PEARLS)

RAYTHEON BBN TECHNOLOGY CORPORATION

FEBRUARY 2021

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Defense Advanced Research Projects Agency (DARPA) Public Release Center and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2021-038 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
CARL R. THOMAS
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor,
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
FEBRUARY 2021	FINAL TECHNICAL REPORT	OCT 2015 – JUL 2020

4. TITLE AND SUBTITLE

PRIVACY-ENHANCED ANDROID RESEARCH AND LEGACY SYSTEMS (PEARLS)

5a. CONTRACT NUMBER
FA8750-16-C-0006

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)

David Levin, Jack Dietz, and Zachary Ratliff

5d. PROJECT NUMBER
BRAN

5e. TASK NUMBER
RA

5f. WORK UNIT NUMBER
YT

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Prime
Raytheon BBN Technology Corporation
10 Moulton St.
Cambridge, MA 02138

Subcontractor
Two Six Labs, LLC
901 N Stuart St, Suite 1000
Arlington, VA 22203-4129

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, VA 22203-1714

AFRL/RITA
525 Brooks Road
Rome, NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2021-038

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. DARPA DISTAR CASE # 33846
Date Cleared: 12 JAN 2021

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Raytheon BBN Technologies Corp. (BBN) and Two Six Labs led the Mobile Collaborative Research Team (CRT) and developed Privacy Enhanced PE for Android, as a fork of the Android Open Source Project (AOSP). Technologies developed by the CRT were implemented and tested in the PE Android architecture. CRT technologies included; Privacy- Enhancing Technologies (PET), Human-Data Interaction (HDI), Experimantal Systems, and Metrics and Analysis. The final version of PE Android is available through the Android Open Source project, http://android-privacy.org

15. SUBJECT TERMS
Android, Privacy, Android Open Source Project, AOSP, Privacy-Enhancing Technologies, PET, Human-Data interaction, HDI

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**CARL R. THOMAS**
U	U	U	UU	36	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES ..... **ii**
LIST OF TABLES ..... **ii**
1.0 SUMMARY **1**
2.0 INTRODUCTION **2**
3.0 METHODS, ASSUMPTIONS, AND PROCEDURES **5**
3.1 Brandeis Programmatic Details 5
3.2 Assumptions 5
3.3 PEARLS Development Approach 6
3.4 Experimentation platform and Test Environment 8
4.0 RESULTS AND DISCUSSION **9**
4.1 Scored Facial Recognition using Machine Learning Demonstrations 9
4.1.1 PP Scored Facial Recognition, Results 11
4.2 Heat Map Demonstrations 11
4.2.1 Function Secret Sharing for Location/Time Aggregation Preserving Privacy (Stealth) 12
4.2.2 Software Guard Extensions Enclave and Data Capsules for Privacy Preserving Database Queries (UC Berkeley) 14
4.3 Anonymous Rendezvous Demonstrations 14
4.3.1 Anonymous Rendezvous with Central Server 15
4.3.2 Anonymous Rendezvous on Phone 15
4.3.3 Anonymous Rendezvous, Results 17
4.4 PRESNA Demonstrations 17
4.4.1 Privacy Preserving (PP) Crisis monitoring prototype 18
4.4.2 PP Crisis Detection Spike 18
4.4.3 PP Crisis Detection Double Spike 19
4.4.4 PP Crisis Detection Double Spike, Cooperating PAL Modules 19
4.5 Scheduling Demonstrations 19
4.5.1 Scheduling App 19
4.5.2 Privacy-Preserving Scheduling App 19
4.6 PrivacyStreams 20
4.7 Privacy Policy 20
4.7.1 Initial User Privacy Policy Concept 21
4.7.2 PE for Android Permission Model and Policy Enforcement 22
4.7.3 Off-Device Policy 23
4.8 Privacy-preserving Machine Learning Experiments 23
4.9 Privacy-preserving Assessment of Trust (PPAT) 23
5.0 CONCLUSIONS **26**
6.0 REFERENCES **28**
LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS **30**

LIST OF FIGURES

Figure **Page**

Figure 1. PE for Android Architecture and Data Flow **3**
Figure 2. Initial PE for Android Architecture **6**
Figure 3. Rapid Gather Demonstration Concept **7**
Figure 4. Experimentation Methodology **7**
Figure 5. FRiPAL Framework **10**
Figure 6. Heat Map Demonstration View **12**
Figure 7. Stealth's PULSAR Function Secret Sharing **13**
Figure 8. UC Berkeley's Helio Secure Database System **14**
Figure 9. Coffeebreak Anonymous Rendezvous Architecture **16**
Figure 10. PRESNA Data Flow Concept **17**
Figure 11. Privacy-preserving Crisis Detection Scenario: Data Flow **18**
Figure 12. PPAT framework **24**

LIST OF TABLES

Table **Page**

Table 1. Brandeis Technical Areas **2**
Table 2. Privacy Policy in the App Lifecycle **20**
Table 3. Privacy Palette Example (GPS) **22**
Table 4. PPAT's Key Features are Based on Innovative Technologies and Approaches **24**

1.0 SUMMARY

This is the Final Technical Report for Privacy-Enhanced Android Research and Legacy System (PEARLS), a proposal by Raytheon BBN Technologies Corp. (BBN) and its subcontractor, Two Six Labs [1] to the DARPA Brandeis program. The Brandeis program sought to extend the privacy tools and techniques available to reliably protect a person's private or sensitive information. While Brandeis research encompassed three research areas, Mobile, Enterprise, and IoT, PEARLS was solely focused on mobile privacy, with the twin goals of providing a mobile-based solution and leading a team of Brandeis mobile privacy researchers. At the first Brandeis Principal Investigator (PI) meeting, the Brandeis performers self-selected into collaborative research teams (CRT), which remained intact throughout the program.

The Brandeis program defined various technical areas: privacy-preserving computation; privacy-enhancing technologies (PET), Human-Data Interaction (HDI), experimental systems, metrics and analysis. The Mobile CRT had team members from each of these technical areas. Each performer team on the CRT advanced their own privacy research, supported the CRT's efforts to build blended experiments comprised of interacting privacy technologies, and contributed to the PEARLS prototype.

CRT development started with a single mobile application and expanded to multiple privacy-preserving experiments over the course of Brandeis. The experiments were developed, integrated, and tested in the BBN-provided mobile test environment.

By the end of the program, BBN and Two Six Labs offered an open source release of Privacy Enhancements for Android (PE for Android) [2], which is available at http://android-privacy.org. With the release of PE for Android, we provide a platform that opens privacy-preserving application development to the larger Android application development community by enabling privacy capabilities within the Android application programming interface (API).

This team developed PE for Android as a platform for exploring concepts in regulating access to private information on mobile devices. Our goal was to create an extensible privacy system that abstracts away the details of various privacy-preserving technologies. We strived to allow app developers to safely leverage state-of-the-art privacy techniques without knowledge of esoteric underlying technologies. Additionally, PE for Android helps users to take ownership of their private information by presenting them with more intuitive controls and permission enforcement. We developed PE for Android as a fork of the Android Open Source Project (AOSP) [3] release for Android 9 "Pie." This effort modified 11 of Android 9's Git repositories and added 3 new repositories of supplemental resources to the code base.

BBN and Two Six Labs released PE for Android as a fork of the AOSP in May 2020. This fork of the Android operating system (OS) directly addresses the Brandeis program goal of enabling an information system that allows individuals, enterprises, and U.S. government agencies to keep personal and/or proprietary information private.

2.0 INTRODUCTION

The overall goal of the DARPA Brandeis program was to develop tools and techniques that enable building systems in which private data can only be used for its intended purpose and no other. To achieve this goal, performers in the various technical areas (TA) self-organized into three CRTs, Mobile, Enterprise, and Internet of Things (IoT). As the lead for the Mobile CRT, BBN with its subcontractor, Two Six Labs [1], presents this report of its results.

The Brandeis Mobile CRT focused on revolutionary advances in privacy-enhancing technologies for mobile devices running the open source Android operating system. The CRT consists of Brandeis performers in the following TAs:

Table 1. Brandeis Technical Areas

Technical Area	Description
TA1	Privacy-preserving computation; privacy-enhancing technologies
TA2	Human-Data Interaction
TA3	Experimental Systems
TA4	Metrics and Analysis

The members of the mobile CRT worked together to create the PEARLS prototype.

Initially, the PEARLS focus was on a single application (app) and evolved to focus instead on privacy preserving techniques that spanned a wide variety of apps.

The result is PE for Android [2], the core of the TA3 mobile privacy approach, which provides a platform that opens privacy-preserving application development to the larger Android application development community by enabling privacy capabilities within the Android API.

This team developed PE for Android as a platform for exploring concepts in regulating access to private information on mobile devices. Our goal was to create an extensible privacy system that abstracts away the details of various privacy-preserving technologies. We strived to allow app developers to safely leverage state-of-the-art privacy techniques without knowledge of esoteric underlying technologies. Additionally, PE for Android helps users to take ownership of their private information by presenting them with more intuitive controls and permission enforcement. We developed PE for Android as a fork of the AOSP [3] release for Android 9 “Pie.” This effort modified 11 of Android 9’s Git repositories and added 3 new repositories of supplemental resources to the code base.

BBN and Two Six released PE for Android as a fork of the AOSP in May 2020. This fork of the Android OS directly addresses the Brandeis program goal of enabling an information system that that allow individuals, enterprises, and U.S. government agencies to keep personal and/or proprietary information private.

PE for Android is a set of extensions and interfaces integrated into the Android OS, similar to Google Play services and SE for Android. Our platform introduces new APIs, services, and a Privacy Abstraction Layer (PAL). The goal of these components is to move sensitive data processing out of the application process space, and into these new services that implement privacy

preserving technologies. Once the sensitive information is transformed, it may then be returned to the application.

PE for Android provides:

- TA1 privacy technologies such as secure multi-party computation
- TA2 technologies such as providing more visibility into what data an app is accessing and more intuitive privacy configuration
- TA4 technologies for quantifying the degree of privacy maintained by the system and the quality of data controls available to the user

As shown in Figure 1, these APIs facilitate the development of swappable components that are invoked when apps request private data. This includes *Policy Managers* that can log requests and help users decide on to access to sensitive information. Further, PE for Android offers the *Private Data Service* and associated modules dubbed micro Privacy Abstraction Layer modules (*µPALs)*, which transform private data into less sensitive forms (e.g., from full resolution global positioning system (GPS) coordinates to just a zip code). This modular architecture allows for the experimentation of various models for how apps consume sensitive data and how users can gain insight into their use.

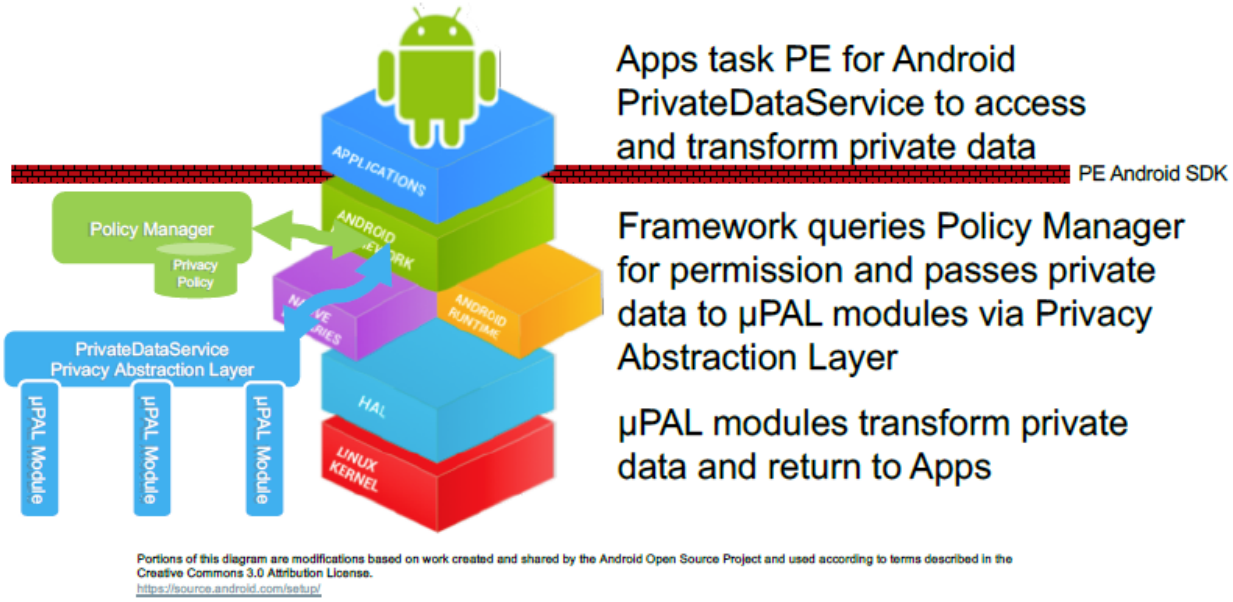

Figure 1. PE for Android Architecture and Data Flow

Policy Managers regulate all access to dangerous permissions [4] on the system, as well as requests issued to the Private Data Service. In making these decisions, Policy Managers operate on run-time contextual factors such as the calling app, its visibility, and (if specified by the app developer) the purpose of the data request. Policy Managers also receive information about app metadata, declared permissions, and developer-set policies at install time. Developers and researchers can rapidly implement various permissions models as Policy Managers. Policy Managers are user-

space Android apps, which eliminates the need to modify permissions logic within the Android platform itself. PE for Android allows users to install multiple Policy Managers and select the active one without needing to re-provision the device.

In lieu of existing permission-controlled APIs like LocationManager, apps targeting PE for Android can opt to send requests for sensitive data to the Private Data Service. Through this, apps obtain the information they need in a "least privileges" [5] manner, and do so without inadvertently extending overly broad permissions to bundled third-party services. The Private Data Service is the entry point for the data transformation and isolation techniques implemented by a μPAL. The Private Data Service and associated μPALs are a collection of trusted processes responsible for the direct access of sensitive resources (e.g., high-accuracy geolocation data) and the subsequent extraction of reduced-resolution information the app needs (e.g., a zip code for a weather app). This model eliminates the need for apps to have direct access to private data otherwise exposed by the stock Android API.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

The PEARLS approach was to build candidate versions of PE for Android and experiment with it by testing applications and use cases that either supported or refuted the underlying hypothesis for the build candidate.

To that end, BBN provided a test environment for privacy experimentation, Two Six provided the PE for Android OS, the TA1 performers provided privacy technologies, and the TA2 performer provided novel approaches for informing the user of an app's privacy posture and the implications to the user.

3.1 Brandeis Programmatic Details

To support this experimentation, BBN and Two Six led a team of other Brandeis performers, which self-selected at the beginning of Brandeis Phase 1. Teams from all the Brandeis TAs comprised the Brandeis Mobile CRT. As the TA3 (integration) performer, BBN led the CRT. Initially, there were three TA1 performers:

- University of California (UC) Berkeley (during the program, the co-PI moved to University of Vermont (UVM)
- Stealth Software Technologies
- Iowa State University (ISU)/Princeton (during the program, the ISU team moved to the University of South Florida (USF)

Carnegie-Mellon University (CMU) provided the privacy policy HDI as TA2 performer.

Cybernetica and Galois rounded out the team as TA4 performers, who provided privacy analysis and metrics.

There were a number of team changes during phase 2 of Brandeis.

- Knexus Research Corporation joined Brandeis in this phase and joined the Mobile CRT
- Dr. Josh Baron cut the ISU/Princeton from Brandeis altogether and directed the Galois team to stop supporting the Mobile CRT
- The UC Berkeley co-PI moved to UVM (in this document, UC Berkeley and UVM refer to the same team)

The Mobile CRT held face-to-face meetings every quarter and conducted status teleconferences every other week.

3.2 Assumptions

In developing our approach, we assumed that the CRT could develop tests and experiments in the test environment that would be directly portable to phones running PE for Android, once that OS was available.

This assumption held true throughout the effort, allowing faster development and experimentation by the CRT.

We also assumed that a single application, RapidGather, would effectively demonstrate all scenarios, use cases, and situations needed by a majority of first responder communities.

Ultimately, this assumption was disproved and the CRT broadened its focus to any application or scenario that showcased the privacy experimentation agenda.

3.3 PEARLS Development Approach

In Brandeis Phase 1, there were two nine-month development cycles. The first cycle focused on building the prototype experimentation system, an initial implementation of PE for Android, and several use cases to test the system and validate an initial set of privacy concepts. The second focused on enhancing the system and PE for Android based on the first set of experiments and added additional experiments.

Figure 2 shows the initial PEARLS architecture, including RapidGather.

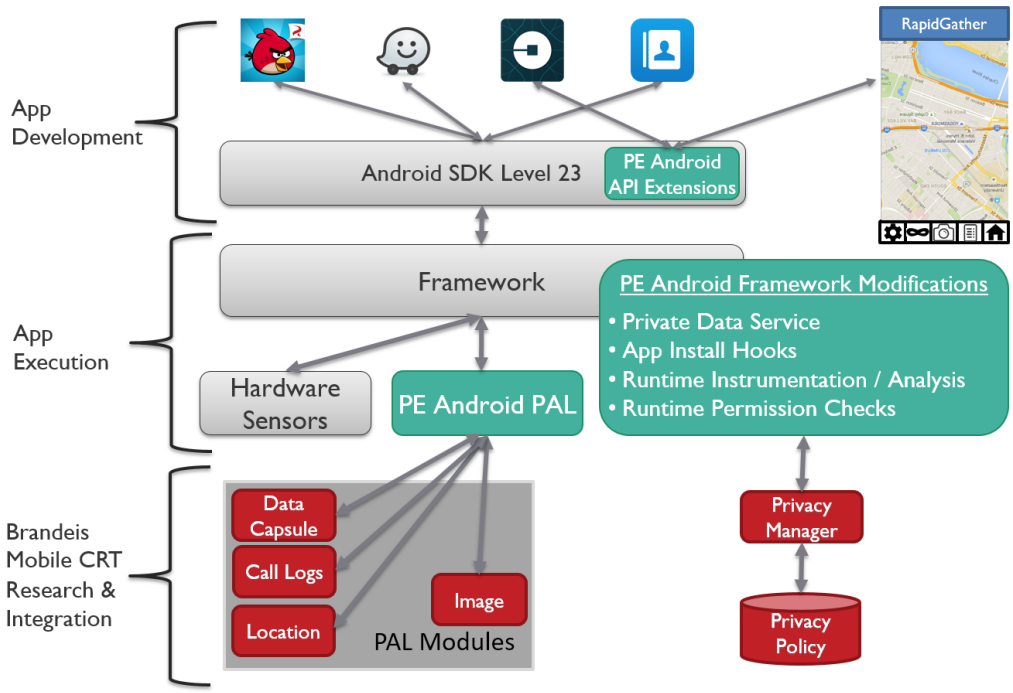

Figure 2. Initial PE for Android Architecture

In Brandeis Phase 1, the focus was on PE for Android and a single application, RapidGather, but this proved both too restrictive and too broad. The RapidGather vision was as a common operations picture (COP) command center (CC) to provide a common situational awareness (SA) [6]. Research by CRA, on the Galois TA4 team, discovered that for any event, each agency involved (e.g., municipal law enforcement (LE), state LE, federal LE, crowd control, health providers) had their own CC to meet their unique SA needs and only used the COP CC for coordination with other agencies.

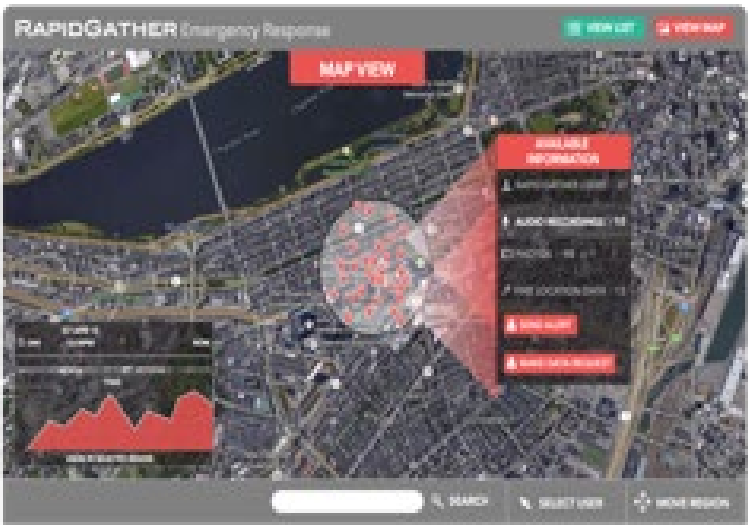

Figure 3. Rapid Gather Demonstration Concept

These efforts made it evident that the scope of building a COP CC exceeded the resources available and that effort would not serve the goals of the Brandeis program, so the DARPA PM approved the PEARLS team's proposal to move to a narrower definition of that application, which allowed a broader range of experiments.

The new approach, starting in Phase 2, was to experiment with privacy enhancing concepts by either proposing PE for Android features and test them with PET from the TA1 performers, or propose privacy use cases and addressing them with TA1 PET or TA2 HDI. The focus was on the privacy aspects of the system and not application development. This resulted in a series of experiments, provided as demonstrations at the Brandeis PI meetings. More important than the demos, the experiments provided real world use cases. The team evaluated each experiment's results and used those findings to define the next set of experiments or new features. This process repeated every six months, allowing three experiment cycles in each phase.

By building realistic scenarios, we explored the utility/privacy tradeoffs to find mechanisms that enable services that respect privacy and learned how to best integrate privacy technologies into the OS. In each cycle, the team flowed the lessons learned into PE for Android and culminated in an AOSP open source release of PE for Android in May 2020.

Figure 4 shows the cyclic nature of the experimentation methodology.

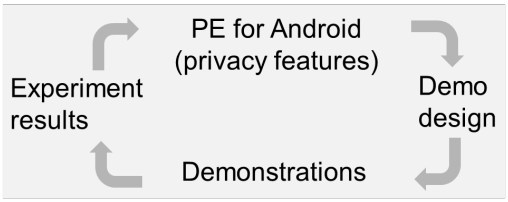

Figure 4. Experimentation Methodology

3.4 Experimentation platform and Test Environment

To achieve the program goals, BBN developed a testing infrastructure to allow the CRT members to collaboratively experiment with their technologies and PE for Android. Initially, BBN focused on building the PEARLS platform and a single app, RapidGather. The key features of the PEARLS platform were its ability to provide a single environment that was seamlessly both a virtualized environment to allow open-ended research, and a physical environment to allow testing on phones. Implemented early in Phase 1, these features persisted and evolved throughout the program's Phase 2 and 3.

The PEARLS platform consisted of physical Android phones running PE for Android, emulators running PE for Android, simulated PE for Android phones and back end services running inside Docker containers, and RabbitMQ as a message queueing communication backplane. Messages used centrally defined schemas in Google Protocol Buffers and representational state transfer APIs. This proved to be robust and scalable and accelerated integration of solutions implemented by multiple teams.

Initially, the PEARLS implementation only existed on the BBN servers, which limited their use by other CRT members. BBN then published the Docker-based implementation to the CRT, which allowed each performer to run PEARLS on their own equipment. Finally, BBN provided PEARLS components on Amazon Web Services (AWS), so that any CRT team could run PEARLS without providing additional hardware. The AWS solution also provided an accessible and simple means to run Mobile CRT experiments anywhere, for example, to provide technology demonstrations at the Brandeis PI meetings.

Initially, the PEARLS focus was on a single application, RapidGather, as shown in Figure 2. It broadened to encompass privacy-preserving techniques that spanned a wide variety of apps.

BBN integrated PEARLS with the CRT code repositories, allowing for automated testing with PEARLS as part of the development cycle.

BBN developed phone simulators to support scaling tests for privacy technologies. These simulated a phone's behavior without requiring a physical phone for each instance. BBN also developed an embedding technique, which allowed testing native Android code in a Java environment.

BBN also developed a Jenkins-based automated test environment and integrated it with the gitlab build environment, to enable continuous integration of PE for Android apps and policy managers to improve our development and testing methodologies.

4.0 RESULTS AND DISCUSSION

This section describes key experiments and demonstrations throughout the three Brandeis phases. All of these contributed to the privacy research and helped shape the ultimate PE for Android architecture. While many are currently abandoned, their contributions to the Mobile CRT privacy research remain strong.

4.1 Scored Facial Recognition using Machine Learning Demonstrations

As the first Mobile CRT experiment and demonstration, *Scored Facial Recognition*, served a pivotal role in providing:

- An initial set of sensitive permissions for PE for Android to protect
- A single experiment shared by the entire CRT
- An initial implementation of the experimentation infrastructure and planned backend services
- A learning platform that allow the CRT members to learn about PE for Android and BBN's experimentation platform

In addition, this experiment allowed the Mobile CRT to expand its understanding of underlying privacy issues, which accelerated and redirected our privacy research.

An initial premise of BBN's effort was that users would only share private information, such as their personal photos, with government as long as the photos were only used for official investigations and were not viewable without explicit permission from the user. Possible solutions were considered that would anonymize both the image and image owner, while allowing investigations, such as searching for a suspect. The use cases for scored facial recognition explored this space and also attempted to address government concerns such as the trustworthiness of anonymous (or otherwise protected) images and how to identify and contact the image owner in order to support a mission, such as a criminal investigation.

ISU provided the technology to privacy-protect the images and UC Berkeley provided the technology to privacy-protect image submission and identification of the image owner.

As part of the RapidGather demonstrations, Facial Recognition in a Privacy Abstraction Layer (FRiPAL) employed conversion of facial features to a set of vectors through Gabor filters and principal component analysis (PCA), and classification through a support vector machine. The intent was to create a database of profiles for persons of interest, which could be matched against newly submitted photographs without being easily reconstructed into the original face image. In doing so, this would protect the privacy of both the persons of interest and of the citizens who submitted new photographs.

The RapidGather demonstration architecture for this demo consisted of a backend server running the FRiPAL software, which stored facial profiles for persons of interest and scored incoming face profiles for matches with those stored profiles. The client software processed photos selected by the user, extracting faces and converting them to profiles to send to the server. The server sent details of matches—including how precise the match was, a reputation score for the submission, and anonymized contact information for follow-up with the submitter—to the RapidGather command center for human follow-up. Berkeley's Helio secure database was used to hold the

associations between submitting users and photos, and neither the submitting user identity nor the facial image was available to the command center operator without user approval.

BBN implemented the Android app and the command center server, while TA1 partners ISU and UC Berkeley implemented the PAL modules and their backend databases. BBN orchestrated the integration for each demonstration.

The order of operations for the FRiPAL demonstrations:

- Protecting image collection and providing image reputation score
  - On startup, the RapidGather app requests PE Android to send a phone call log abstract, including a session universally unique identifier (UUID)
  - Berkeley's Helio calculates a reputation score (based on likelihood that the phone is used by an authentic user) from call log, attached to session UUID; sends to server
  - On photo submission, RapidGather sends a tuple of (photo UUID, session UUID) to Helio server. (Command center cannot query the Helio server for that data directly.)
  - Command Center queries Helio server for reputation of given photo UUID; Helio server returns the reputation score, which is a result of database join between photo UUID and session UUID, without exposing the session UUID
  - Photo matches and associated reputations are shown to command center user
- Command center requests to contact image owner:
  - Command center can send a request for a specific photo UUID to all RapidGather clients
  - RapidGather client checks sent photo UUIDs; if there is a match it presents request to user
  - User can choose to share original photo with command center

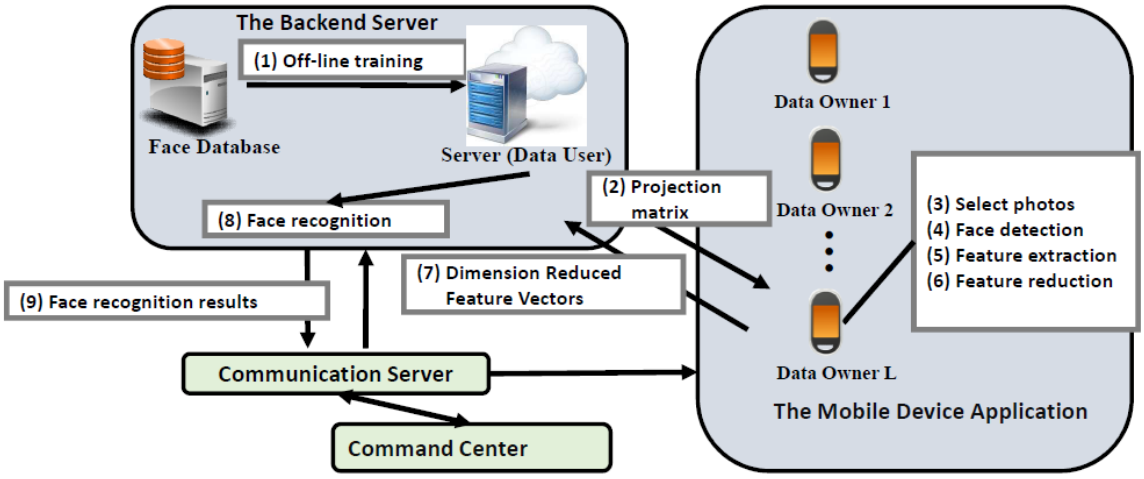

Figure 5. FRiPAL Framework

PP facial recognition, V1

For the initial implementation of the privacy preserving facial recognition demo, ISU and team developed the off-line database creation and the on-line client-side feature reduction and server-side face lookup. [7]

PP scored facial recognition, V2

For the second demo iteration, the team implemented an enhanced algorithm for analysis of facial feature vectors using multi-kernel PCA, with a homomorphically encrypted channel for exchanging kernel PCA parameters between the client and the server. In addition, the scoring for facial feature matches was improved and operationalized. [8]

PP scored facial recognition, V3

The third iteration of the facial recognition demo added the ability for multiple servers to independently process photos of persons of interest into dimensionally reduced profiles, determining a common dimensionality reduction parameter set. This allowed the third demo to show multiple agencies independently processing their own libraries of persons of interest and processing them in the FRiPAL compute server without sharing the unprotected photos with each other or with the RapidGather command center. [9]

4.1.1 PP Scored Facial Recognition, Results

Ultimately, this approach proved unsuccessful as these direct transformations did not provide sufficient mathematical guarantees of privacy protection, and Dr. Josh Baron ended this experimentation.

4.2 Heat Map Demonstrations

As the second Mobile CRT experiment and demonstration, the *Heat Map*, served a pivotal role similar to *Scored Facial Recognition* and introduced new backend requirements that proved out the flexibility of the experimentation infrastructure.

An initial premise of BBN's effort was that users would only share private information, such as their location, with government as long as the data user could not determine the location of any individual. Possible solutions would be anonymized or aggregated use of the data. The heat map use cases explored this space and attempted to answer questions such as "What useful information can we gather to help first responders?" For these use cases, the data is protected, exported from the phone, and used without leaking individual's information.

The RapidGather Heat Map feature, as part of the integrated RapidGather demonstration (Figure 6), was designed to collect and aggregate locations and movement of individuals using the RapidGather app on their PE for Android devices, allowing a command center operator to see the density and motion of crowds without exposing the location of any individual in an identifiable manner. When notified of an emergency in the area, the RapidGather app on user devices would periodically request that PE for Android send a location message to a server using one of the privacy-preserving technologies.

BBN implemented the Android app and the command center server, while TA1 partners Stealth and UC Berkeley implemented the PAL module and their backend databases. BBN orchestrated the integration for each demonstration.

Figure 6. Heat Map Demonstration View

4.2.1 Function Secret Sharing for Location/Time Aggregation Preserving Privacy (Stealth)

Stealth's Private Updateable Lightweight Scalable Active Repository (PULSAR) function secret sharing (FSS) solution uses function secret sharing to split sensitive data into multiple shares, with the shares sent to multiple non-colluding servers. The FSS algorithm ensures that the averaged data for a specific location and time can only be obtained by querying both servers and combining the results. [10]

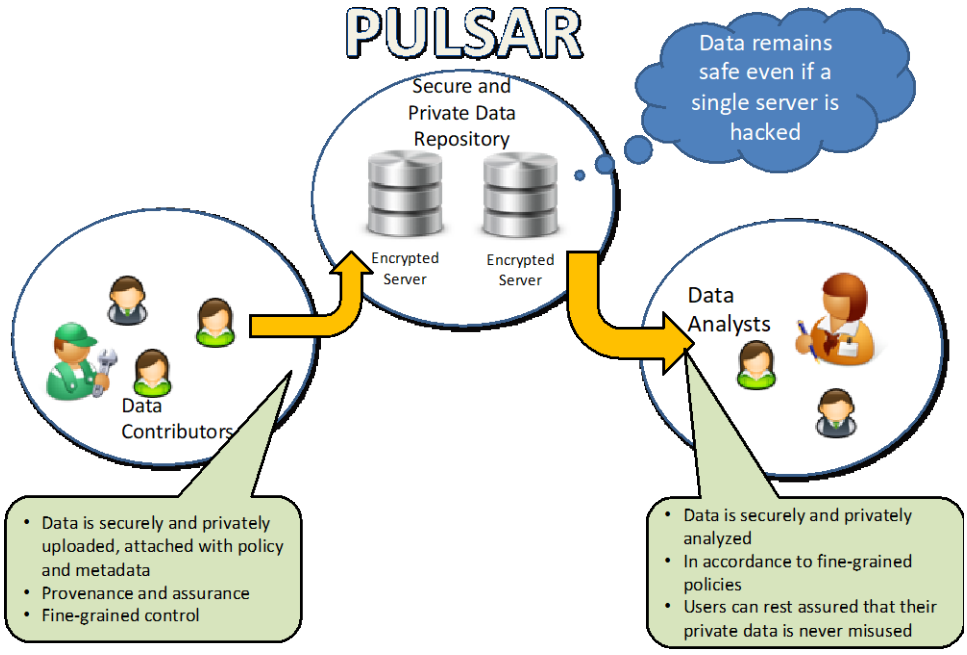

Figure 7. Stealth's PULSAR Function Secret Sharing

PP aggregated location/time heatmap (Stealth), V1

For the initial integration with Stealth's PULSAR FSS technology, Stealth provided an initial FSS server based on distributed point functions, with the ability to capture inputs from user devices within a predefined target area and timespan. Input from multiple emulated Android phones was demonstrated. Integration, testing and demonstration allowed user data to be added on top of preloaded files to demonstrate the effect of large pools of user devices submitting information and test the data analysis processes for the demonstration.

PP aggregated location/time heatmap (Stealth), V2

For the second version of the heatmap demo, Stealth iterated on their server design to improve performance and synchronize updates between the two servers. For the simulated groups of 100 to 1000 phones, this significantly improved data fidelity and update rates for live data ingestion.

PP aggregated location/time heatmap (Stealth), V3

For the final iteration of the heatmap demo, Stealth further improved performance and scalability, and this was demonstrated.

Stealth worked toward implementing a merger between their FSS and secure multi-party computation (MPC) capabilities to allow for policy-based information release as part of their work for other CRTs, but we did not find a good fit for bringing that work into this demo.

PP aggregated location/time heatmap, results

While these experiments demonstrated solutions that don't leak individual locations, the technologies only use PE for Android to protect the data as it leaves the phone and do not rely on PE for Android for ongoing protection.

4.2.2 Software Guard Extensions Enclave and Data Capsules for Privacy Preserving Database Queries (UC Berkeley)

As a second implementation of the RapidGather heat map, UC Berkeley implemented a heatmap using their Helio database technology. Location and motion reports were inserted into data capsules by the Helio PAL and stored in the database; queries from the command center aggregated the results by location and returned them.

UC Berkeley's Helio solution constitutes a secure database, where sensitive data is submitted to the server after it has been encapsulated in an encrypted data capsule [11]. The database can run inside an Intel Software Guard Extensions (SGX) enclave [12], and the queries are written in Helio's Duet language [13] that incorporates differential privacy metrics into all query expressions. The Duet query language interpreter ensured that the results did not identify any individual user.

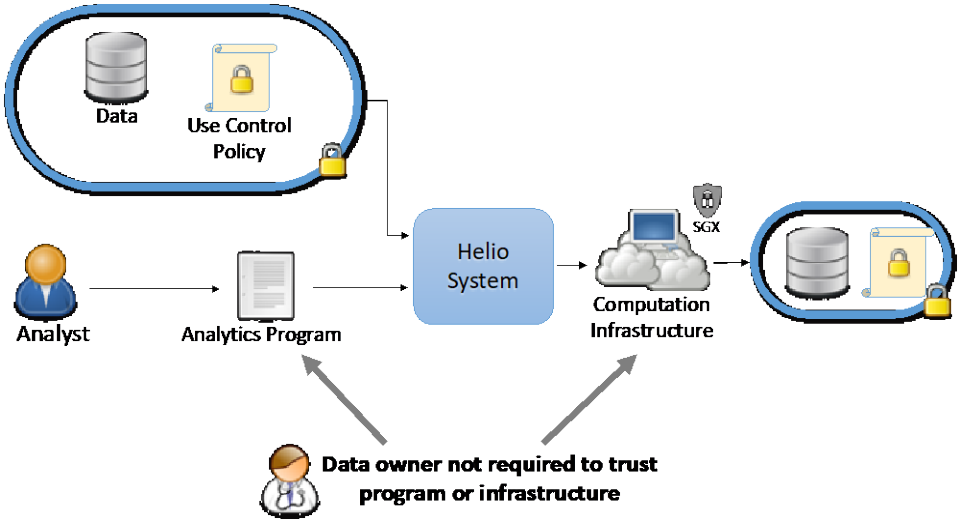

Figure 8. UC Berkeley's Helio Secure Database System

PP aggregated location/time heatmap, results

The Helio system allowed the rapid implementation of the Heat Map demo scenario in database insertions and queries. While fully implemented as a proof-of-concept, this approach was not included in live demos at the PI meetings.

4.3 Anonymous Rendezvous Demonstrations

A broad set of experiments that started with the RapidGather app, but evolved to a series of standalone apps. The set includes *Aid Tent*, *Help Me*, *Coffeebreak*, and multiple *Anonymous Rendezvous* use cases.

The use cases include TA1 approaches provided by Stealth, UC Berkeley, and BBN.

The core idea behind the anonymous rendezvous application is guiding individual users to various locations of interest without revealing their private location.

4.3.1 Anonymous Rendezvous with Central Server

The earliest version of the anonymous rendezvous applications considered a centralized server paradigm. Users send encrypted location data to centralized server(s), which perform a secure computation and return the result. In each case, we use the honest-but-curious model where some subset of the centralized servers act honestly, but may passively attempt to learn extra information from the execution of the protocol itself.

4.3.1.1 Aid Tent, Stealth

BBN extended the RapidGather application to support anonymous rendezvous between a user and a set of aid tents. Each aid tent location and its available resources remain private from one another as well as the user.

Stealth developed the secure MPC protocol used by the aid tent servers to compute functions over private data. The aid tent servers do not learn the location of the RapidGather user, but rather only the result of the computation - the location of the nearest aid tent to the user. This functionality enables users to privately route themselves to a nearby aid tent for medical attention.

4.3.1.2 Help Me, UC Berkeley

The second iteration of the anonymous rendezvous demonstrations utilized UC Berkeley's Helio database technology. BBN integrated the "Help Me" functionality into RapidGather as a feature for assisting an injured individual in finding the closest nearby medical facility.

UC Berkeley developed a Helio PAL module that obtains the fine-grained location information from PE for Android and packages it into an encrypted data capsule. The Helio SGX enclave backend computes known functions on the private location data and returns an encrypted location result to the user. The SGX enclave provides strong security and privacy guarantees by cryptographically attesting to the correct execution of these functions.

4.3.2 Anonymous Rendezvous on Phone

A drawback of the previous versions of the anonymous rendezvous demonstrations is the central server(s) that perform computation on behalf of the users. If an adversary compromises a sufficient subset of the central servers, then the privacy guarantees of the anonymous rendezvous applications are moot. By decentralizing the functionality and placing the privatized computations on the users' phones, we arrive at a more compelling demonstration of privacy-preserving anonymous rendezvous.

4.3.2.1 Starbucks Anonymous Rendezvous Based on Privacy Streams Fuzzed Location

BBN developed an initial version of Coffeebreak, a privacy-preserving meetup application, on the PE for Android architecture. Coffeebreak allows users to invite a group of contacts to compute a mutually agreeable Starbucks location to meet at, without revealing the parties' locations to one another.

BBN developed the phone-to-phone messaging interface using RabbitMQ − an Advanced Message Queuing Protocol (AMQP) middleware library. RabbitMQ functions in the publish-subscribe model, however, direct phone-to-phone communications are achieved using direct exchanges defined over the phone numbers of participating users. Although Coffeebreak uses the publish-subscribe model, any phone-to-phone communication works as a substitute, e.g., short message service (SMS), Extensible Messaging and Presence Protocol (XMPP), Twitter posts, etc.

The initial version of Coffeebreak included a PrivacyStreams µPAL for obtaining a fuzzed location object, i.e., a location with uniform noise added to the latitude and longitude values. The fuzzed location object consists of a 64-bit binary string of fixed decimal latitude and longitude values. The fixed decimal location causes negligible loss in accuracy and complies with the bit-string notation expected by the Stealth MPC library.

Stealth ported their MPC library to Android by compiling an executable callable by the Android Runtime. Coffeebreak passes the fuzzed location object as input to the Stealth executable, which internally performs the MPC protocol over AMQP with the participating parties. The MPC protocol returns a centroid of the participants' locations and Coffeebreak uses this centroid to query the closest Starbucks location.

The final result is a privacy-preserving Coffeebreak application where the location of individual users not only remained private from the respective participating parties, but also from the Coffeebreak application itself.

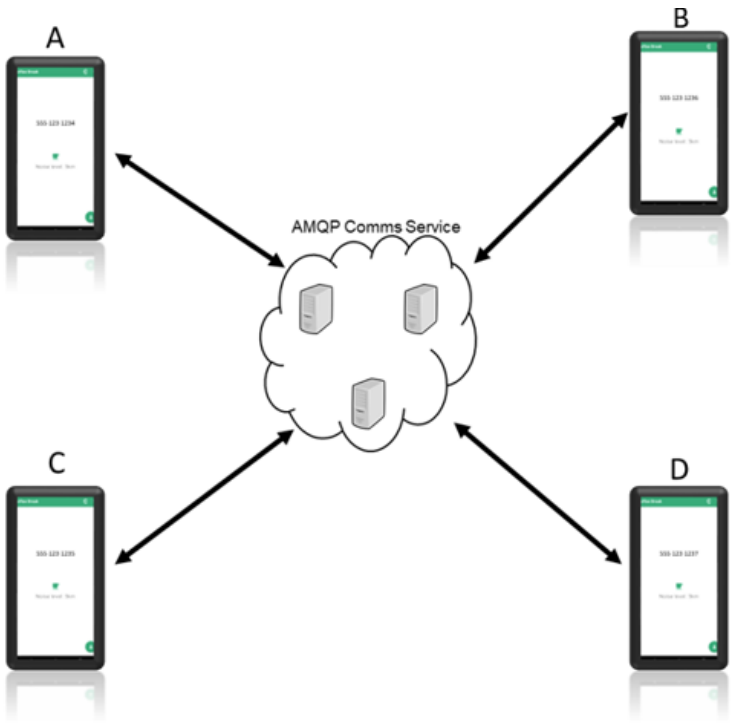

Figure 9. Coffeebreak Anonymous Rendezvous Architecture

4.3.2.2 Coffeebreak – Starbucks rendezvous on stock Android

For the final iteration of the anonymous rendezvous functionality, BBN developed the Coffeebreak Android application on stock Android. The stock Android version of Coffeebreak did not protect sensitive location information from the application, but did protect the individual locations from participating parties.

Stealth implemented a Java Native Interface interface into their MPC library.

4.3.3 Anonymous Rendezvous, Results

The results of each use case inspired one or more follow on use case experiments, culminating in Coffeebreak on stock Android. While not directly supportive of the PE for Android focus, it provided an advanced technical solution that provided a strong privacy guarantee, which assured that each party's location remained hidden from the others while computing a centroid.

4.4 PRESNA Demonstrations

Privacy Enhanced Social Network Analysis (PRESNA) demonstrated an approach for tracking the state and evolution of public safety crises through analysis of collective usage of smartphones in a crisis-affected area. The concept was provided by Knexus, while additional technical area 1 assistance was provided by Stealth and the University of Vermont (UVM).

PRESNA's concept is that when a localized crisis occurs, individuals who are in the immediate vicinity of the crisis will use their mobile devices to contact people who they are socially close to. In turn, people outside the crisis area will contact people they know who are in the crisis area. By gathering privacy-protected data about calls and text messages between participants in the crisis-affected area and their most frequent contacts, a measure can be made of how important and unexpected the event is. [14]

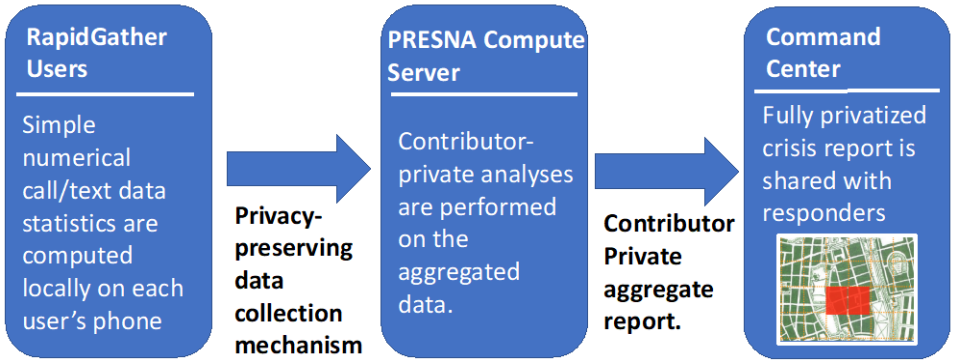

Figure 10. PRESNA Data Flow Concept

PRESNA was implemented as a PE for Android-based client that would send data about pairs of communication endpoints and geographic locations to a privacy-aware database, and a server that would periodically poll the database and search for spikes in activity. Large increases in activity would result in a location-based alert being sent to a crisis command center. Both a PE for Android

implementation for physical and virtual devices and a simulation that scaled to thousands of devices were implemented.

Knexus implemented the server and generated simulation data of user movement within a crisis area using agent-based modeling. UVM wrote data ingestion and Duet query terms for its Helio system. Two Six provided support for the PAL and µPAL modules, and BBN provided guidance and integrated the demonstration system as well as the initial PE for Android app.

4.4.1 Privacy Preserving (PP) Crisis monitoring prototype

The initial iteration of the PRESNA demonstration was used to validate the hypothesis that significant spikes in activity could be detected with the client/server/database architecture. The simulation was implemented and tested with 50 simulated user devices at a time, and the crisis detection server was able to make queries to the Helio database and read the data submitted by the user devices.

4.4.2 PP Crisis Detection Spike

For this iteration, the crisis detection server searched for sudden increases in call activity in cells within the potential crisis area, and compared the rate of calls to close contacts with the rate of calls in general.

This was demonstrated with a mix of up to 500 simulated phones and several emulated PE for Android devices.

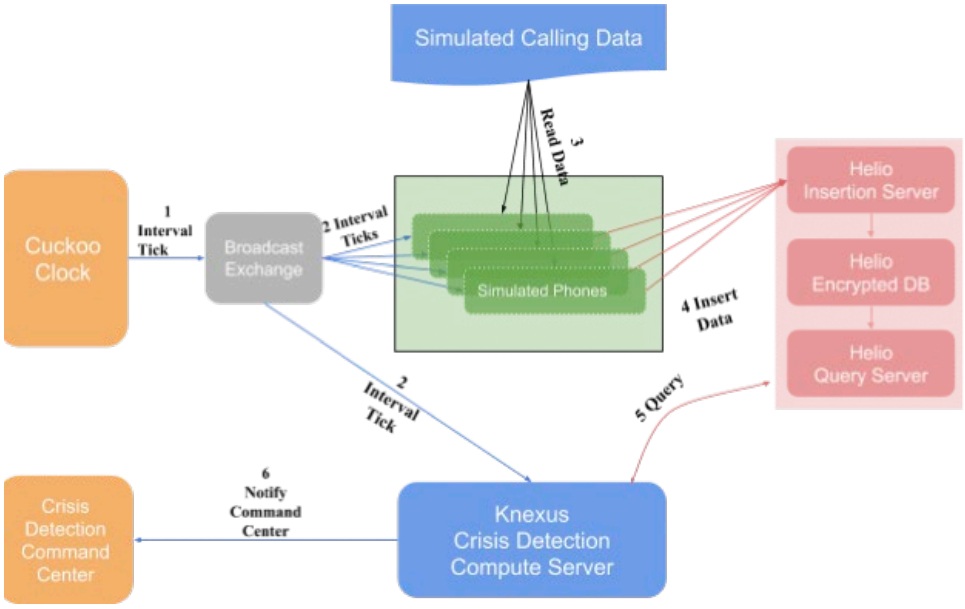

Figure 11. Privacy-preserving Crisis Detection Scenario: Data Flow

4.4.3 PP Crisis Detection Double Spike

This demo iteration changed algorithms to report an alert when a double spike of communications activity is detected in an area; the hypothesized behavior is that there will be a delay between one spike of people inside the area calling out to their closest communication partners, and a second spike of people outside the area learning of the crisis and calling inward to people who are in the immediate vicinity of the crisis. The hypothesized behavior was demonstrated given the agent-based simulation data.

Additionally, this iteration added support for SMS messages in addition to cellular telephone calls. Scalability on the server side was improved, and simulations of 2000 user phones were demonstrated.

As at this point PE for Android was available for off-the-shelf mobile phones, the scenario was demonstrated with a pair of physical devices as well.

4.4.4 PP Crisis Detection Double Spike, Cooperating PAL Modules

This iteration built on the previous double spike algorithm, restructuring the PE for Android application to make use of μPAL technology to better protect the associations between phone and SMS records, contact database entries, and location information within the Helio capsule.

Knexus also improved the agent-based simulation data with a more complex model of human behavior in a crisis event, with goal-oriented responses to perceived threats and crowding.

4.5 Scheduling Demonstrations

In this thread of development, the Mobile CRT demonstrated selecting meeting times from user calendars while keeping the calendar details private. TA1 techniques were provided by Stealth, UC Berkeley, CMU, and BBN.

4.5.1 Scheduling App

BBN developed TARDIS, a baseline scheduling application on PE for Android. The baseline version of TARDIS allows users to select a set of contacts and set-up a meeting. This is similar to existing scheduling applications such as Doodle [15].

BBN developed the PE for Android application, the AMQP communication mechanisms, backend schedule de-confliction service, and the necessary scripts for running the demo. Additionally, BBN provided a baseline PAL module for converting a user's calendar data into a bit string representation of *busy* or *available*.

4.5.2 Privacy-Preserving Scheduling App

The second iteration of TARDIS development involved incorporating the privacy-preserving technologies from the TA1 teams. Each TA1 team extended the baseline PAL module provided by BBN.

UC Berkeley provided a Helio PAL module for sending an encrypted calendar bit-string to a backend Helio SGX enclave. The SGX enclave performs a secure computation over the encrypted data and returns the chosen time slot to each user.

Stealth developed a PAL module for secretly sharing the calendar bit-string data and sending it to *n* back-end MPC servers. The MPC servers jointly compute the schedule de-confliction and return the chosen time slot to the users.

CMU incorporated their PrivacyStreams API into the baseline PAL module.

4.6 PrivacyStreams

The PrivacyStreams development effort consisted of initial experiments with incorporating the paper concept initially developed by the CMU team [16] into PE for Android. A core idea behind the Privacy Streams API is to give application developers fine-grained control over the sensitive data their app receives from the Android OS. For example, rather than collecting raw microphone data, the PrivacyStreams library can return a simple Boolean value to the query: "Is it loud in the room?" The purpose of the functionality is two-fold—give application developers only the data they need and reduce the burden of developing these data transformations themselves.

After developing the core API for the PrivacyStreams library, CMU transitioned to creating PrivacyStreams PAL modules. BBN integrated PrivacyStreams PAL modules into a private scheduling application (TARDIS) and an anonymous rendezvous application (Coffeebreak). Experiments with the PrivacyStreams PAL modules led Two Six Labs to adopt a new paradigm in the PE for Android architecture. PE for Android inherited the core idea of providing many sensitive data transformations to application developers by way of μPALs (micro-PALs), available in the public release.

4.7 Privacy Policy

The policy development effort consisted of a series of concept and mechanism explorations alternately led by BBN and CMU. The overarching goal was to identify the best approaches for eliciting the user's privacy preferences, presenting the approaches to the user, and enforcing decisions based on the chosen preferences. Table 2 show how these mechanisms came into play throughout the application lifecycle.

Table 2. Privacy Policy in the App Lifecycle

Lifecycle	Goal	Notes
Application Development	Developer declare privacy goals within the app source code	Provide the developers that help inform privacy decisions and are easy to use
Application Install	Inform the user of the application's impact on the user's privacy prior to installation and provide opt-out	Help user make informed decisions; avoid intrusive user interface that leads to complacency
Application Runtime	Inform the user of the application's impact on the user's privacy during runtime and provide options to allow or deny	Help user make informed decisions; avoid intrusive user interface that leads to complacency
Off-device representation	Provide a secure means to bind policy to data and ensure no privacy leakage in transit to the secure backend; provide a sufficiently rich policy representation	While PE for Android can provides these guarantees on the device, it is up to the secure backend actually enforce those guarantees

4.7.1 Initial User Privacy Policy Concept

The initial concept, described below, evolved over the course of the program. The main difference between the initial and final concepts is the granularity of each policy entry. It evolved from a *per-activity* to a *per-call* directive and added the ability to verify the purpose assumption by observing the app's current call stack (stack trace).

Each user will define a user privacy policy that mediates what data is available, what form the data takes (e.g., full or reduced precision, blurred), and how the data can be used in the RapidGather (RG) App and command center. A palette s represents the policy that allows the user to have control over the use of their data with their desired degree of granularity. The privacy palette is an array of n-tuples. Each tuple consists of the application, the requested permission, the purpose, a context, and an action.

Application – The application component of the privacy palette is the name of the application accessing the privacy services. For the mobile CRT application, RapidGather, it is *RG App*.

Permission – Permissions within the privacy palette map roughly to Android *sensitive permissions.* For example, when concerned about images, instead of having the *camera* as the permission, the tuple would contain *images* and PE Android would map that to either the mobile camera or phone's file system. Other sensitive permissions would be *location*, either through GPS or network provider-based, and *audio*, with properties similar to images.

Permissions also include hardware sensors such as the accelerometer and magnetometer; however, they are not in scope for Phase 1.

Purpose – The purpose of why the application needs the permission. It is an explanation of how the data can be used. Within the context of RapidGather, it includes the DataUse and AlertLevel properties. When setting their privacy policies, users can provide different data for a DataUse based on AlertLevel, with the opportunity to opt out of data requests that seem intrusive for the level, or opt in for precise data collection at higher AlertLevels.

Context – A context is a user-defined constraint on data requests. It is a user-defined level of privacy for the data request, which allows the user to constrain how or when an application is allowed to collect data. For example, the following would be considered allowable contexts:

- Maximum precision of sensitive data ever allowed
- Data collection restricted while in particular areas (e.g., near user's home)
- Data collection restricted during particular times (e.g., no collection at night)

Action – An action for the device that details what to do when the application needs particular data. The RapidGather platform enforces the user privacy policy for data requests from RG App to the device's data sources and only returns values allowed by the user privacy policy.

An action is what the platform will do when it receives a request from the application for sensor data based on the other conditions within the privacy palette. An action will be either automatic or deferred.

Automatic actions result in the platform taking action without presenting a request to the user. The platform will either allow and grant access to the sensor/source data, or disallow and deny access to the sensor/source data.

Deferred actions result in a query presented to the user for each data request, leaving the final call to the user on a per-case basis.

As an example, one way to think of the palette is that the user can specify a different action for a DataUse, based on AlertLevel or context. For example, using two palette entries, the user can specify that the DataUse "what is your precise location?" is *allow* at AlertLevel high and *disallow* at AlertLevel low. Two more entries would allow DataUse "are you in the RG area of interest?" for both AlertLevels.

In Table 3 we have an example of how permission, purpose, and context dictate the action of *GPS access*.

Table 3. Privacy Palette Example (GPS)

Permission	DataUse (Purpose)	AlertLevel (Purpose)	Context	Action
GPS	"What is your precise location?"	Low	9am-5pm	Disallow
GPS (high precision)	"What is your precise location?"	Medium	9am-5pm	Disallow
GPS (low precision)	"What is your precise location?"	Medium	9am-5pm	Ask
GPS	"What is your precise location?"	High	No constraint	Allow

4.7.2 PE for Android Permission Model and Policy Enforcement

The following summarizes the privacy policy model currently implemented in PE for Android.

Policy Managers regulate all access to dangerous permissions on the system. In making these decisions, Policy Managers operate on run-time contextual factors such as the calling app, its visibility, and (if specified by the app developer) the purpose for the data request. Policy Managers also receive information about app metadata, declared permissions, and developer-set policies at install time. Developers and researchers can rapidly implement various permissions models as Policy Managers. Policy Managers are user-space Android apps, which is an approach that eliminates the need to modify permissions logic within the Android platform itself. PE for Android allows users to install multiple Policy Managers and select the active one without needing to re-provision the device.

PE for Android invokes the current active Policy Manager upon app installation and at runtime as apps request sensitive data. The Policy Manager then returns a decision to allow or deny these requests. To aid in the Policy Manager's decision, permission checks in PE for Android may be tagged with a purpose [16] that describes how the data will be used. Existing research shows that purposes can be inferred in certain cases for apps targeting the stock Android API. [17]

Under Android's default permissions model, developers declare an app's required permissions in the App Manifest [18] file. PE for Android expands on this notion by allowing developers to specify the purpose of those permissions in an additional App Policy file. During app installation, the Policy Manager can use the app policy to decide whether to allow or block the installation. For example, the Policy Manager could show a dialog to the user displaying the app's requested permissions and their corresponding purposes. This gives an overview of how the app would interact with sensitive data, and let the user decide whether to continue with the installation.

Similar to Android's Runtime Permission feature [19], PE for Android forwards all runtime permissions checks to the Policy Manager, which implements the logic to grant or deny these

requests. However, rather than relying on the requesting app to provide context to the user via a dialog, PE for Android provides the Policy Manager with contextual information about the permission check. This information includes: a declared purpose (if supplied); the app component that issued the request; the top Activity visible when the request occurred; and a stack trace of all threads running in the requesting app. App developers can provide additional context via the declaration of purposes using PE for Android's APIs. For example, a maps app can set the purpose to "Navigation" before making a call to "requestLocationUpdates()". When a declared purpose is not available, a Policy Manager might attempt to infer it from the remaining contextual information. For example, if the stack trace shows that the request originated from a known third party advertising library, the Policy Manager could inform the user that this request is likely meant for advertising.

Likewise, the Policy Manager controls access to sensitive data accessed via the Private Data Service. To do this, the Policy Manager receives the type of data being transformed (e.g., Location), and the purpose of the request (e.g., Weather). It also receives a human-readable description of the transformation that will occur. μPAL modules must provide this description for the user to understand what transformed data the app will receive. In the example of the zip code μPAL (described in a further section below), the description string reads "Returns the US zip code of the current location." This text can be displayed within a Policy Manager's user interface to inform the user about the μPAL when configuring the policy.

4.7.3 Off-Device Policy

Off-device policy enforcement requires a secure means to bind a privacy policy to the data that it protects and a trusted platform to enforce the policy. PEARLS supported off device policy using a Helio *data capsule* PAL module communicating with a DuetSGX [12]. Otherwise, the research was limited, since enforcement would occur off the PE for Android device, which was out of scope for PEARLS.

4.8 Privacy-preserving Machine Learning Experiments

In Phase 3 of the program, the UVM Mobile CRT team member considered a series of experiments with mobile data using machine learning on private data to generate actionable results that do not violate user privacy. Two reached the planning stage, a flood detection scenario, prompted by a DHS request, and a privacy-protected general machine-learning pipeline. See the DuetSGX site [12].

4.9 Privacy-preserving Assessment of Trust (PPAT)

Privacy-preserving Assessment of Trust (PPAT) was a BBN proposal to use novel cryptography to provide a reliable trustworthiness score for anonymous submissions.

We predicated PPAT on the hypothesis that the success of mobile sensing frameworks would depend on large volumes of legitimate participation. In order to achieve this, mobile sensing frameworks generally need to guarantee the privacy of individuals who participate, accurately assess the trustworthiness of participant data, and efficiently integrate data validation mechanisms to improve the reliability of trust assessments for recurring participants. BBN proposed to develop PPAT, a framework for mobile participatory sensing that addresses all these requirements. Four

innovations serve as the basis for PPAT, as illustrated in Figure 12: a privacy reputation ledger, privacy-preserving evidence disclosure, trust bootstrapping, and distributed anonymous validation.

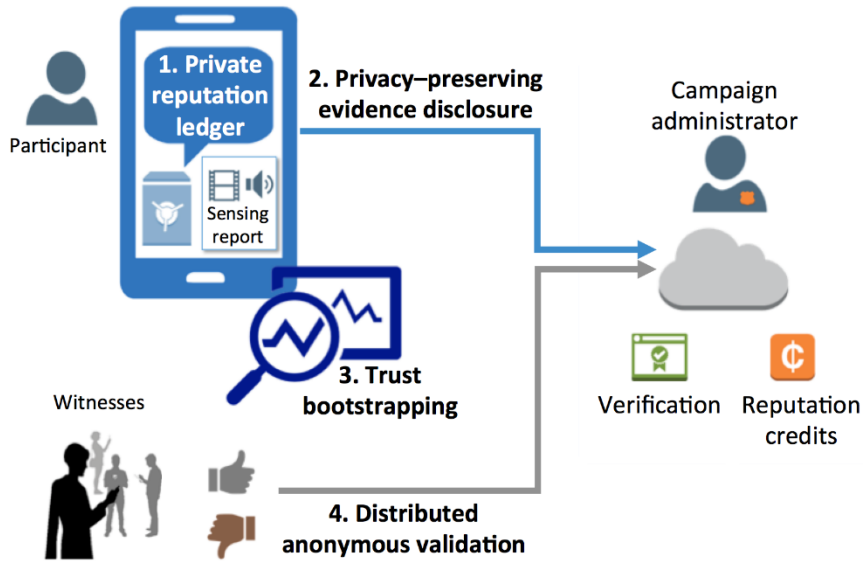

Figure 12. PPAT framework

PPAT would allow campaign administrators to obtain a *reputation score* associated with each *sensing report* while permitting neither association of the report with a pseudo-identity nor the linkage of multiple reports. The key features of PPAT and the accompanying innovations are shown in Table 4.

Table 4. PPAT's Key Features are Based on Innovative Technologies and Approaches

Key feature	Innovation
Anonymous tracking of reputation	**Reputation ledgers**: a reputation self-tracking mechanism that will use reputation ledgers to cryptographically bind participants–through non-interactive zero-knowledge proofs–to their reports without disclosing their identity;
Privacy-preserving reports of evidence	**Evidence disclosures:** a privacy-preserving mechanism for disclosing evidence with a sensing report—including a reputation score. This mechanism will allow participants to selectively attach information that can help during a validation step to assess the trustworthiness of a sensing report
Reputation assessment for new participants	**Bootstrapping trust**: a mechanism for bootstrapping trustworthiness using features of a device, participant behavior, and global auxiliary information when there is not prior evidence that can be used to judge the reputation of a participant
Anonymous evidence report validation	**Distributed anonymous validation:** a mechanism for allowing the collaborative anonymous validation of incoming reports based on the reputation of the submitter and the cross-validation of evidence statements in the reports.

PPAT's framework would allow a participant to submit a sensing report (i.e., a picture, a video, or a tip), automatically accounting for reputation credits associated with previous reports from that participant. In addition, participants will retain control over when and what to share. The mobile client will communicate to the user the risks associated with attaching specific pieces of evidence (e.g., precise location or weather conditions) to a report. The client will also communicate the potential boost to the credibility of a report from attaching such evidence. PPAT will allow for the anonymous cross-validation of some pieces of evidence (e.g., weather or noise conditions) before submitting a report to the campaign administrator – the entity who collects and processes sensing reports. The processing of the report (by a campaign administrator) will involve the verification of a cryptographic proof that guarantees the correct accounting of reputation credits, and the validation of a report by additional trusted parties (i.e., 911 dispatchers).

Dr. Andres Molina-Markham proposed PPAT and submitted it as an engineering change proposal to BBN's PEARLS effort on Brandeis and listed himself as PI. In October 2016, Jeremy Epstein, the then current DARPA Brandeis PM, approved the funding of the PPAT proposal. In November 2016, Dr. Andres Molina-Markham resigned from BBN and DARPA rescinded the award until we could find a PI replacement. We chose Dr. Alina Oprea, a professor at Northeastern University as the PPAT PI. In April 2017, Mr. Dave Gunning, the then current DARPA Brandeis PM, approved the funding for PPAT with the new PI, but Northeastern did not receive authorization to proceed until July 2017. In November 2017, Dr. Josh Baron, the then current DARPA Brandeis PM, cancelled PPAT.

There are no PPAT artifacts.

5.0 CONCLUSIONS

Our research successfully demonstrated that privacy protecting features could migrate to the OS, thereby protecting user information without affecting the effectiveness of the apps that process that information.

At the final Brandeis PI meeting on March 30-31, 2020 Dr. Josh Baron, the DARPA program manager leading Brandeis, stated that PE for Android is one of the major achievements for the program.

We released PE for Android on May 6, 2020, accompanied by press releases, including one from DARPA, excerpted here:

> Under DARPA's Brandeis program, a team of researchers led by Two Six Labs and Raytheon BBN Technologies have developed a platform called Privacy Enhancements for Android (PE for Android) to explore more expressive concepts in regulating access to private information on mobile devices. PE for Android seeks to create an extensible privacy system that abstracts away the details of various privacy-preserving technologies, allowing application developers to utilize state-of-the-art privacy techniques, such as secure multi-party computation and differential privacy, without knowledge of their underlying esoteric technologies. Importantly, PE for Android allows mobile device users to take ownership of their private information by presenting them with more intuitive controls and permission enforcement options.
>
> The researchers behind PE for Android today released a white paper detailing the platform's capabilities and functionality, and published an open source release of its code to GitHub. In open sourcing PE for Android, the researchers aim to make it easier for the open-source Android community and researchers to employ enhanced privacy-preserving technologies within Android apps while also encouraging them to help address the platform's current limitations and build upon its initial efforts.
>
> "User privacy should be a first-rate concern for mobile app development, and we are hoping that open-sourcing PE for Android will galvanize the Android developer community," said Dr. Josh Baron, the DARPA program manager leading Brandeis. "While the benefits of this to personal and commercial users may be apparent, military personnel are also heavy users of mobile devices and often bring personal devices to or near work. Changes made to the Android ecosystem will therefore have important implications for privacy and security across the Department of Defense. I encourage the community to take a look at the code, improve it if they find gaps, and figure out which parts are deserving of adoption into the broader Android ecosystem." [20]

PE for Android is available here: https://android-privacy.org. PE for Android is now a working operating system that follows standard Android development guidelines and programming style, yet provides a programming environment that allows developers to easily write applications with strong privacy protection guarantees. In addition, its architecture provides clear and well-defined layer boundaries that provide a straightforward basis to evaluate and prove those guarantees.

The press release also provides a summary of the open source release. “The PE for Android source code release includes several use cases and applications for these key components, many of which were developed by other research teams working under the Brandeis program. This includes a Privacy Checkup tool; the Purposes Policy Manager developed by Carnegie Mellon University, which lets people view and set policies for individual apps as well as all apps on a smartphone; and various μPAL modules capable of performing privacy transformations on different types of sensitive data. The University of Vermont and the Brandeis Helio team are among those responsible for developing the μPAL modules discussed in the white paper.” [20]

In addition, the Mobile CRT test environment remains a useful tool for developing, experimenting with, and testing mobile apps with any version of the Android OS. It is readily reproducible by anyone by following the instructions we provide and using the tools we provide.

The PEARLS project succeeded both in proving the value of embedding privacy controls into an OS as well as in producing an open source version of that OS to allow others to continue the development of systems that provide utility while protecting user’s sensitive data.

6.0 REFERENCES

[1] Due to corporate changes in 2017, Invincea Labs became Two Six Labs. Available: https://www.twosixlabs.com/about/

[2] "PE for Android," [Online]. Available: https://github.com/orgs/twosixlabs/teams/peforandroid/repositories . [Accessed 17 April 2020].

[3] "Android Open Source Project," [Online]. Available: https://source.android.com/ . [Accessed 17 April 2020].

[4] "Permissions overview," [Online]. Available: https://developer.android.com/guide/topics/permissions/overview . [Accessed 17 April 2020].

[5] "Least Privilege | CISA - US-CERT," [Online]. Available: https://www.uscert.gov/bsi/articles/knowledge/principles/least-privilege . [Accessed 17 April 2020].

[6] M. Lepinski *et al.*, "Privacy-Enhanced Android for Smart Cities Applications," in Smart City 360°, Oct. 2015, pp 66-77, doi: 10.1007/978-3-319-33681-7_6

[7] D. Zhuang, S. Wang, and J. M. Chang, "FRiPAL: Face recognition in privacy abstraction layer," in *2017 IEEE Conference on Dependable and Secure Computing*, Taipei, Taiwan, Aug. 2017, pp. 441–448, doi: 10.1109/DESEC.2017.8073826.

[8] T. Chanyaswad, J. M. Chang, and S. Y. Kung, "A compressive multi-kernel method for privacy-preserving machine learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 4079–4086, doi: 10.1109/IJCNN.2017.7966371.

[9] M. Al-Rubaie, P. Wu, J. M. Chang, and S.-Y. Kung, "Privacy-preserving PCA on horizontally-partitioned data," in *2017 IEEE Conference on Dependable and Secure Computing*, Taipei, Taiwan, Aug. 2017, pp. 280–287, doi: 10.1109/DESEC.2017.8073817.

[10] E. Boyle, N. Gilboa, and Y. Ishai, "Function Secret Sharing," in *Advances in Cryptology - EUROCRYPT 2015*, Berlin, Heidelberg, 2015, pp. 337–367, doi: 10.1007/978-3-662-46803-6_12.

[11] L. Wang *et al.*, "Data Capsule: A New Paradigm for Automatic Compliance with Data Privacy Regulations," in *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*, Cham, 2019, vol. 11721, pp. 3–23, doi: 10.1007/978-3-030-33752-0_1.

[12] "DuetSGX GitHub," [Online]. Available: https://plaid.w3.uvm.edu/duet-sgx . [Accessed 17 April 2020].

[13] J. P. Near *et al.*, "Duet: an expressive higher-order language and linear type system for statically enforcing differential privacy," *Proc. ACM Program. Lang.*, vol. 3, no. OOPSLA, pp. 172:1–172:30, Oct. 2019, doi: 10.1145/3360598.

[14] C. Task, "Privacy-Preserving Social Network Analysis," PhD Thesis, Purdue University, 2015.

[15] Doodle (website). https://en.wikipedia.org/wiki/Doodle_(website)

[16] Y. Li, F. Chen, T. J.-J. Li, Y. Guo, G. Huang, M. Fredrikson, Y. Agarwal and J. I. Hong, "PrivacyStreams: Enabling Transparency in Personal Data Processing for Mobile Apps," [Online]. Available: https://dl.acm.org/doi/10.1145/3130941 . [Accessed 17 April 2020].

[17] H. Jin, M. Liu, K. Dodhia, Y. Li, G. Srivastava, M. Fredrikson, Y. Agarwal and J. I. Hong, "Why Are They Collecting My Data?: Inferring the Purposes of Network Traffic in Mobile Apps," [Online]. Available: https://dl.acm.org/doi/10.1145/3287051 . [Accessed 17 April 2020].
[18] "App Manifest Overview," [Online]. Available: https://developer.android.com/guide/topics/manifest/manifest-intro . [Accessed 17 April 2020].
[19] "Request App Permissions," [Online]. Available: https://developer.android.com/training/permissions/requesting. [Accessed 17 April 2020].
[20] "Researchers on DARPA's Brandeis Program Enhance Privacy Protections for Android Applications," DARPA press release. Available: https://www.darpa.mil/news-events/2020-05-06 [Accessed 1 June 2020]

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AFRL	Air Force Research Laboratory
AMQP	Advanced Message Queuing Protocol
AOSP	Android Open Source Project
API	application programming interface
app	application
AWS	Amazon Web Services -One of Amazon's cloud computing service offerings
BBN	Raytheon BBN Technologies Corp.
Brandeis	The DARPA Brandeis program, named for former Supreme Court Justice Louis Brandeis
CC	command center
CMU	Carnegie-Mellon University
COP	common operations (or operational) picture
CRT	collaborative research team
DARPA	Defense Advanced Research Projects Agency
DHS	Department of Homeland Security
FRiPAL	Facial Recognition in a Privacy Abstraction Layer
FSS	function secret sharing
GPS	global positioning system
HAL	Hardware Abstraction Layer
HDI	Human-Data Interaction
Helio	UC Berkeley effort to provide end-to-end security & privacy, using data capsule and secure platform, such as SGX. See for example, https://plaid.w3.uvm.edu/duet-sgx .
IoT	Internet of Things
ISU	Iowa State University
LE	law enforcement
MPC	Multi-party computation
OS	operating system
PAL	Privacy Abstraction Layer
PCA	principal component analysis
PE for Android	Privacy Enhancements for Android
PEARLS	Privacy-Enhanced Android Research and Legacy Systems
PET	privacy-enhancing technology
PI	Principal Investigator
PM	program manager
PPAT	Privacy-preserving Assessment of Trust
PRESNA	Privacy Enhanced Social Network Analysis
PULSAR	Private Updateable Lightweight Scalable Active Repository
RabbitMQ	open-source message-broker software
SA	situational awareness

SDK	Software Development Kit
SGX	Intel Software Guard Extensions; a set of security-related instruction codes that are built into some modern Intel central processing units
SMS	Short message service – a text messaging service component of most telephone, Internet, and mobile device systems
TA	Technical Area
TARDIS	A fictional time machine and spacecraft that appears in the British science fiction television series Doctor Who. https://en.wikipedia.org/wiki/TARDIS
UC	University of California
U.S./US	United States
USF	University of South Florida
UUID	universally unique identifier
UVM	University of Vermont
XMPP	Extensible Messaging and Presence Protocol
μPAL	Micro (μ) PAL