Exposure Notification Privacy-preserving Analytics (ENPA) White Paper

Apple and Google

April 2021

1 Introduction

The Exposure Notifications System (ENS) [AG20] introduced by Apple and Google is designed to enable automated alerts to users of potential exposure to COVID-19 while providing strong privacy guarantees. In particular, it is designed so that information about a positive diagnosis or potential exposure does not include identifying information. Furthermore, the system was designed so that Google, Apple, and external entities do not learn if exposure notifications were shown on users' devices, nor of other related information such as user activities, contacts, or assessed risk level. Such information is kept locally on users' devices, used locally on those devices for exposure-notification purposes, and never transmitted off the users' devices as part of the ENS protocol.

With the fast changing conditions of the pandemic, it is important for Public Health Authorities (or PHAs) to continuously monitor the evolution of the pandemic and assess the effectiveness of the Exposure Notification System in order to adapt the overall epidemiological responses based on new trends.

Exposure Notification Privacy-preserving Analytics (ENPA) is a privacy-preserving measurement system which enables PHAs to collect metrics in an *aggregated* form only while upholding the strong privacy principles of ENS. Designed alongside ENS, ENPA provides access to a small set of metrics, whose scope is strictly limited to information with epidemiological purposes. ENPA leverages state of the art privacy and cryptographic techniques to enable the computation of these aggregate metrics, which do not reveal any individual data. None of the entities involved in the operation of ENPA, not even Apple or Google, can see the individual contributions of a device. Additionally, the aggregate metrics are only available to the PHA, unless it voluntarily shares them. For example, ENPA can provide information on how many exposure notification alerts are displayed to users, without exposing how many alerts each device has displayed. This is important information that allows PHAs to understand better the effectiveness of the system and adjust the parameters they provide for ENS if deemed necessary.

Additionally, participation in ENPA is under the user's control. Users are asked whether they wish to participate and the privacy-preserving data collection is only allowed with explicit user consent. It can also be disabled or re-enabled by users through settings in their devices at any time.

The following sections explain how ENPA, when enabled, allows PHAs—and only PHAs—to receive informative aggregate metrics that closely approximate the true aggregate statistics about ENS and ENX deployments while preserving the privacy of each user, and follow the strong privacy values of both Apple and Google.

2 ENPA Design Goals

The ENPA design goal is to provide statistical, aggregate, and differentially private metrics to PHAs from the Exposure Notification System while protecting the individual contributions, and therefore the privacy of each

participant. The design aims at achieving this by combining two mechanisms: (1) before transmission to the PHA, each user's client device randomizes the user's metric contribution so as to ensure, after aggregation, differential privacy guarantees, and (2) transmission to the PHA is performed using a cryptographic secure computation protocol that only reveals the user reports to the PHA as anonymous, aggregate sums.

ENPA is also designed to protect the dependability of any collected metrics—i.e., prevent the corruption of the metrics in the face of a variety of errors, faults, or malicious behavior. For this, ENPA relies on cryptographic mechanisms for validating user metric reports, in particular by bounding the number of reports by any one user, and the magnitude of the values in those reports, such that no user's client can skew the aggregate metrics by accident or malice.

Finally, ENPA is designed to provide users and the PHAs with both transparency and control: ENPA provides no metrics for EN deployments unless *both* users and their PHA have opted into ENPA telemetry.

3 Technical Overview

Each user device has values of various metrics related to EN, such as the number of exposure notifications shown to the user, or the corresponding risk scores computed on the device. The functionality of ENPA aggregates the metric values from individual devices and provide these to the PHA.

The privacy preserving design for the above functionality leverages the cryptographic secure aggregation construction introduced by Boneh and Corrigan-Gibbs in the system Prio [CB17]. ENPA further extends this protocol to achieve differential privacy for the output via amplification of small local randomization at the users using the aggregation protocol [BBGN19, CSU+19, UEFM+20, FMT20].

In this section, we provide a high level overview of ENPA with a focus on the devices and servers operated by Apple or Google. A more detailed description of the ENPA protocol deployment architecture is provided in Section 5. The PHA runs one of the Prio servers that performs aggregation, and obtains telemetry statistics. As per the Prio architecture, the ENPA system relies on one additional aggregation server, called an *Helper (Facilitator) server*. This party facilitates the computation in the privacy-preserving protocol but does not obtain the final telemetry statistics. Achieving the privacy properties of our system requires independent parties running the Helper server and the PHA server, and assumes that those parties *will not collude*.

The PHA and the Helper servers do not receive user contributions directly, but instead receive them from an *Ingestion server*; both Apple and Google are running their own ingestion server for their respective user populations. Thus, devices send user encrypted contributions to their respective ingestion server. Each encrypted contribution contains two distinct parts, one is encrypted with the Helper server public key and the other encrypted with the PHA server public key. This encryption protects the confidentiality of the content from both eavesdroppers and from the ingestion servers themselves. Each ingestion server separates the two encrypted parts of contributions and forwards the respective Prio parts to the appropriate Helper and PHA servers—but only after ensuring each contribution is from a legitimate user device, to eliminate illegitimate inputs. Furthermore, the ingestion server ensures that each forwarded contribution is stripped of identifiable information, such as the client IP address or exact upload time. After transmission of data to the PHA and Helper servers, the Ingestion servers do not retain any of the contributions.

Fig. 1 shows an high-level overview of the system architecture and the data flow. Each client adds local noise to its input value, then splits it into two cryptographic shares and computes a distributed zero-knowledge range proof for the value that is shared. The client encrypts one of the shares and the corresponding proof part under the public key of the PHA, and the other share and proof part under the public key of the Helper server. The client sends the resulting ciphertexts to the ingestion server, accompanied with an attestation (i.e., a device-specific cryptographic proof) of being a legitimate device.

The ingestion server checks the attestation for each client contribution, and for valid contributions forwards the accompanying opaque ciphertexts to the PHA and Helper server, respectively. This forwarding

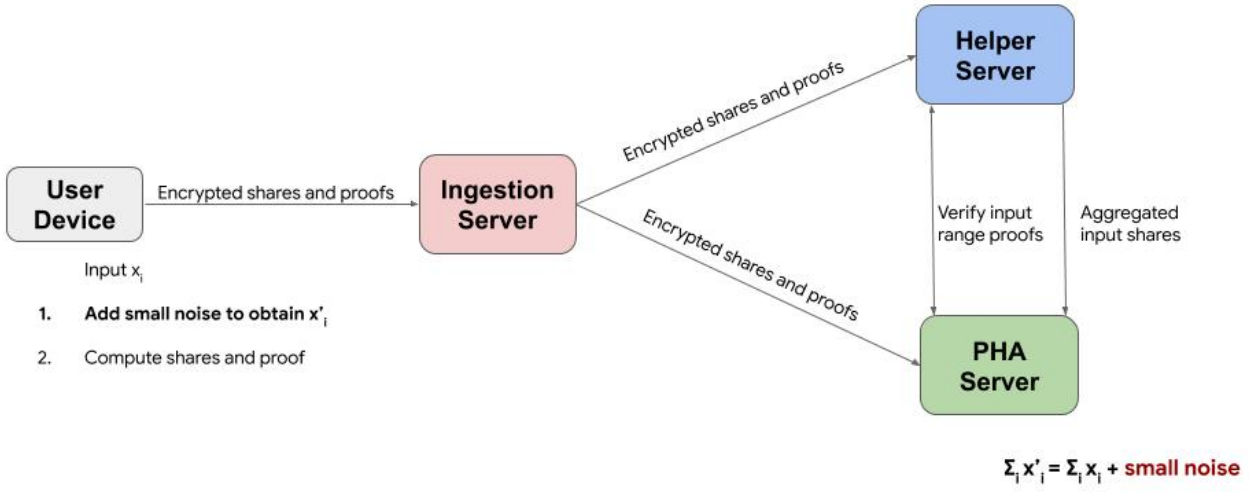

Figure 1: ENPA Architecture

is performed in batches, in a way that limits association between the inputs and outputs of the ingestion server.

The PHA and the Helper servers decrypt the inputs they receive from the ingestion servers and run the distributed verification for the zero-knowledge range proofs for all inputs. The Helper server sums all shares for which the corresponding proofs verified and sends the resulting aggregated result to the PHA. The PHA sums up its shares from inputs that had valid range proofs and adds to that the aggregate received from the Helper server to obtain the final differentially-private aggregate count across all devices.

4 ENPA Components

4.1 Input Format

Each client will report values for multiple metrics. For each metric, its domain is split into a set of intervals (i.e., histogram bins) where the client will be reporting the interval in which its input falls. The reported metrics are combined into a single vector by concatenating the vectors for each metric (i.e., forming a cross product of the histograms, accounting for all possible interval partitions). Each client's input is this full, concatenated binary vector.

4.2 Local Input Randomization for Differential Privacy

The basis of ENPA reporting is the vector $\vec{X} = (x_1, \ldots, x_k)$ constructed as described above. However, this original input vector is never sent directly; rather, it is randomly perturbed and then sent via the encrypted, secret-shared aggregation mechanism described in the rest of this section. In combination, those mechanisms ensure that each client's metric report has local differential privacy, that the report is not linked back to the originating client, and that the final, aggregate metrics received by PHAs are subject to strong differential privacy guarantees.

As a first step, each ENPA client randomizes its original input vector $\vec{X} = (x_1, \ldots, x_k)$ to obtain a report binary vector $\vec{X}' = (x'_1, \ldots, x'_k)$. This randomization is performed in a manner as shown below, such that a local differential privacy guarantee of ε_0 holds for the $\vec{X}'$ report binary vector.

$\vec{X}' := \mathbf{RR}(\vec{X}, \varepsilon_0)$ is computed for each x_j in $\vec{X}$, $1 \leq j \leq k$, using a good source of randomness, as follows

- Randomly sample a Boolean b with probability $(e^{\varepsilon_0} - 1)/(e^{\varepsilon_0} + 1)$ of being true.
- If b is true, then set $x'_j \leftarrow x_j$.
- If b is false, then set $x'_j \leftarrow r$, where r is randomly sampled as 0 or 1 with equal probability.

NB this is equivalent to

- Randomly sample a Boolean b with probability $1/(e^{\varepsilon_0} + 1)$ of being true.
- If b is false, then set $x'_j \leftarrow x_j$.
- If b is true, then set $x'_j \leftarrow \neg x_j$.

This resulting metric report binary vector $\vec{X}'$ is never revealed directly to any party in the ENPA system. Instead, it is reported via the secret-sharing aggregation mechanism described below; the aggregation will greatly amplify the privacy guarantees for each client beyond the original ε_0 local differential privacy guarantee as explained in Section 4.5.

4.3 Input Secret Sharing

ENPA transmits the randomized metric report binary vector $\vec{X}'$ using secret sharing and authenticated encryption to the PHA and Helper aggregation servers, separately. ENPA generates the two secret shares to be transmitted as follows: for each x'_i, the client generates a random r modulo p, and computes shares $[x'_i]_{\mathsf{Helper}} := r \bmod p$ and $[x_i]_{\mathsf{PHA}} := x_i - r \bmod p$. The value of p is such that $p \equiv 1 \bmod 2N$ with $N = 2^{\lceil \log_2(k+1) \rceil}$ in order to facilitate efficient Fast Fourier Transform (FFT) computations.

The information in the shares for the PHA and Helper aggregation servers are

$$[\vec{X}']_{\mathsf{PHA}} = ([x'_1]_{\mathsf{PHA}}, \ldots, [x'_k]_{\mathsf{PHA}}),$$
$$[\vec{X}']_{\mathsf{Helper}} = ([x'_1]_{\mathsf{Helper}}, \ldots, [x'_k]_{\mathsf{Helper}}).$$

In order to optimize the communication of shares, ENPA relies on a pseudorandom generator (PRG) to compress the secret shares for one of the parties. In particular, ENPA generates the shares r for the Helper aggregation server by generating (using good randomness) a 32-byte random $\mathsf{seed}_{\mathsf{Helper}}$, parsed as a 128-bit key and a 128-bit counter, and uses those values to initialize an AES-CTR PRG from which ENPA derives the Helper shares and the corresponding PHA shares. Thus, the shares shares actually sent to the PHA and Helper aggregation servers are

$$[\vec{X}']_{\mathsf{PHA}} = ([x'_1]_{\mathsf{PHA}}, \ldots, [x'_k]_{\mathsf{PHA}}),$$
$$\mathsf{seed}_{\mathsf{Helper}}.$$

Upon receipt and decryption of its share, the Helper aggregation server expands $\mathsf{seed}_{\mathsf{Helper}}$ to obtain $[\vec{X}']_{\mathsf{Helper}} = ([x'_1]_{\mathsf{Helper}}, \ldots, [x'_k]_{\mathsf{Helper}})$. For ease of presentation we omit explicitly describing this step for the rest of the paper and assume that the two servers have the expanded shares.

4.4 Zero-Knowledge Distributed Range Proofs

ENPA uses the distributed zero-knowledge range proofs construction introduced by Prio [CB17], with optimizations for the ENPA specifics. In particular, after input randomization, an ENPA client possesses a vector $\vec{X}' = (x'_1, \ldots, x'_k)$ and will be providing a proof that this is a binary vector, i.e., $x'_i \in \{0, 1\}$ for all $i = 1$ to k. In order to prove this statement, it is sufficient to show that $C(x_i) = 0$ for the function $C(z) = z(z-1)$ over $\mathbb{Z}_p$. The proof of this polynomial identity can be established for all $\{x'_i\}_{i=1}^k$ independently, in parallel using a polynomial of at least degree $k + 1$. Because ENPA uses FFT computation for fast polynomial evaluation

and interpolation, ENPA chooses to uses polynomials of degree the next power of two, $N = 2^{\lceil \log_2(k+1) \rceil}$, where the $2N$-th roots of unity are denoted as $1, \omega, \omega^2, \ldots, \omega^{2N-1}$.

Proof Generation. To create distributed zero-knowledge proofs that establish that the two messages to the PHA server and the Helper server form a binary vector when summed together pointwise in $\mathbb{Z}_p$, the ENPA client generates polynomials $f(x)$ and $g(x)$ as follows:

- $f(1)$ and $g(1)$ are set to be independent random values,
- $f(\omega^{2i}) := x'_i$ for $i = 1$ to k,
- $g(\omega^{2i}) := x'_i - 1$ for $i = 1$ to k,
- $f(\omega^{2i}) := g(\omega^{2i}) := 0$ for all $k < i < N$.

By design, it holds that f and g are polynomials of degree $\leq k$. Next, the ENPA client computes $h(x) = f(x) \cdot g(x)$, which is a polynomial of degree $\leq 2k \leq 2N$. We note that $h(\omega^{2i}) = 0$ for all $i = 1$ to $N - 1$ if-and-only-if the inputs x_i are binary. Thus, knowing the points $h(1)$ and $h(\omega^{2j-1})$ for $j = 1$ to N will be sufficient to recover the target polynomial $h(x)$.

The ENPA client computes the following proof shares for the PHA and Helper aggregation servers:

$$\pi_{\mathsf{PHA}} = \left([f(1)]_{\mathsf{PHA}}, [g(1)]_{\mathsf{PHA}}, [h(1)]_{\mathsf{PHA}}, [h(\omega)]_{\mathsf{PHA}}, [h(\omega^3)]_{\mathsf{PHA}}, \ldots, [h(\omega^{2N-1})]_{\mathsf{PHA}}\right), \text{and}$$
$$\pi_{\mathsf{Helper}} = \left([f(1)]_{\mathsf{Helper}}, [g(1)]_{\mathsf{Helper}}, [h(1)]_{\mathsf{Helper}}, [h(\omega)]_{\mathsf{Helper}}, [h(\omega^3)]_{\mathsf{Helper}}, \ldots, [h(\omega^{2N-1})]_{\mathsf{Helper}}\right),$$

computing the transmitted shares for the polynomial evaluations computed as done for the input in Section 4.3, e.g., also using a PRG to reduce the size of the message to the Helper server.

Proof Verification. For the PHA server and Helper servers to collaboratively verify (without learning anything else) that two messages form a binary vector when summed together pointwise in $\mathbb{Z}_p$, the two servers run the following distributed verification over the two messages $([\vec{X}']_{\mathsf{PHA}}, \pi_{\mathsf{PHA}})$ and $([\vec{X}']_{\mathsf{Helper}}, \pi_{\mathsf{Helper}})$ they have respectively received.

The PHA server interpolates three polynomials $f_{\mathsf{PHA}}(x)$, $g_{\mathsf{PHA}}(x)$ and $h_{\mathsf{PHA}}(x)$ as follows:

- using the points $f_{\mathsf{PHA}}(1) := [f(1)]_{\mathsf{PHA}}$, $f_{\mathsf{PHA}}(\omega^{2i}) := [x'_i]_{\mathsf{PHA}}$ for $i = 1$ to k, and $f_{\mathsf{PHA}}(\omega^{2i}) := 0$ for all $k < i < N$ to interpolate $f_{\mathsf{PHA}}(x)$ of degree $\leq k < N$;
- using the points $g_{\mathsf{PHA}}(1) := [g(1)]_{\mathsf{PHA}}$, $g_{\mathsf{PHA}}(\omega^{2i}) := [x'_i]_{\mathsf{PHA}} - 1$ for $i = 1$ to k, and $g_{\mathsf{PHA}}(\omega^{2i}) := 0$ for all $k < i < N$ to interpolate $g_{\mathsf{PHA}}(x)$ of degree $\leq k < N$;
- using the points $h_{\mathsf{PHA}}(1) := [h(1)]_{\mathsf{PHA}}$, $h_{\mathsf{PHA}}(\omega^{2i}) = 0$ for all $i = 1$ to $N - 1$, and $h_{\mathsf{PHA}}(\omega^{2i-1}) = [h(\omega^{2i-1})]_{\mathsf{PHA}}$ for $1 \leq i \leq N$ to interpolate $h_{\mathsf{PHA}}(x)$ of degree $< 2N$.

The Helper server interpolates three polynomials $f_{\mathsf{Helper}}(x)$, $g_{\mathsf{Helper}}(x)$ and $h_{\mathsf{Helper}}(x)$ as follows:

- use the points $f_{\mathsf{Helper}}(1) := [f(1)]_{\mathsf{Helper}}$, $f_{\mathsf{Helper}}(\omega^{2i}) := [x'_i]_{\mathsf{Helper}}$ for $i = 1$ to k, and $f_{\mathsf{Helper}}(\omega^{2i}) := 0$ for all $k < i < N$ to interpolate $f_{\mathsf{Helper}}(x)$ of degree $\leq k < N$;
- use the points $g_{\mathsf{Helper}}(1) := [g(1)]_{\mathsf{Helper}}$, $g_{\mathsf{Helper}}(\omega^{2i}) := [x'_i]_{\mathsf{Helper}}$ for $i = 1$ to k, and $g_{\mathsf{Helper}}(\omega^{2i}) := 0$ for all $k < i < N$ to interpolate $g_{\mathsf{Helper}}(x)$ of degree $\leq k < N$;
- use the points $h_{\mathsf{Helper}}(1) := [h(1)]_{\mathsf{Helper}}$, $h_{\mathsf{Helper}}(\omega^{2i}) = 0$ for all $i = 1$ to $N - 1$, and $h_{\mathsf{Helper}}(\omega^{2i-1}) = [h(\omega^{2i-1})]_{\mathsf{Helper}}$ for $1 \leq i \leq N$ to interpolate $h_{\mathsf{Helper}}(x)$ of degree $< 2N$.

From the above interpolations, the below functions f', g', and h' can be defined, and these give the necessary verification properties:

- $f'(x) = f_{\mathsf{PHA}}(x) + f_{\mathsf{Helper}}(x)$ since $f'(\omega^{2i}) = f_{\mathsf{PHA}}(\omega^{2i}) + f_{\mathsf{Helper}}(\omega^{2i})$ for $0 \leq i \leq N-1$, and $f'(x) = f(x)$ if the proof is correct;
- $g'(x) = g_{\mathsf{PHA}}(x) + g_{\mathsf{Helper}}(x)$ since $g'(\omega^{2i}) = g_{\mathsf{PHA}}(\omega^{2i}) + g_{\mathsf{Helper}}(\omega^{2i})$ for $0 \leq i \leq N-1$, and $g'(x) = g(x)$ if the proof is correct;
- $h'(x) = h_{\mathsf{PHA}}(x) + h_{\mathsf{Helper}}(x)$ since $h'(\omega^i) = h_{\mathsf{PHA}}(\omega^i) + h_{\mathsf{Helper}}(\omega^i)$ for $0 \leq i \leq 2N-1$ if and only if $x_i \in \{0,1\}$ for $1 \leq i \leq k$, and $h'(x) = h(x)$ if the proof is correct.

The PHA server and the Helper server agree on a randomly-chosen polynomial evaluation point r where $r \leftarrow \mathbb{Z}_p \backslash \{1, \omega, \ldots \omega^{2N-1}\}$. The PHA evaluates $f_{\mathsf{PHA}}(r)$, $g_{\mathsf{PHA}}(r)$ and $h_{\mathsf{PHA}}(r)$ and sends these values to the Helper. The Helper server evaluates $f_{\mathsf{Helper}}(r)$, $g_{\mathsf{Helper}}(r)$ and $h_{\mathsf{Helper}}(r)$ and sends these values to the PHA.

The PHA server and the Helper server accept the proof if they verify the following equation to hold:

$$\left(f_{\mathsf{PHA}}(r) + f_{\mathsf{Helper}}(r)\right)\left(g_{\mathsf{PHA}}(r) + g_{\mathsf{Helper}}(r)\right) = \left(h_{\mathsf{PHA}}(r) + h_{\mathsf{Helper}}(r)\right). \tag{1}$$

Security Properties. The above distributed zero-knowledge (ZK) system is an instantiation of the fully linear PCP construction introduced in the work of Boneh et al. [BBCG+19], which generalizes and optimizes the distributed ZK construction from the Prio protocol [CB17]. As such the construction provides

- *completeness*, which guarantees that an honest client can construct a correct proof;
- *soundness*, which guarantees that client cannot accidentally or maliciously construct a cheating proof;
- *honest-verifier zero knowledge*, which means that the verifiers cannot learn anything more than the validity of the proof statement, as long as one of them is honest.

We refer the reader to the work of Boneh et al. [BBCG+19] for complete proofs. Intuitively soundness follows from the fact that the check of Eq. (1) guarantees with high probability that $f'(x) \cdot g'(x) = h'(x)$, as polynomials. In particular, this implies that $f'(\omega^{2i}) \cdot g'(\omega^{2i}) = h'(\omega^{2i})$ for all $i = 1$ to k, which in turns implies

$$([x'_i]_{\mathsf{PHA}} + [x'_i]_{\mathsf{Helper}}) \cdot ([x'_i]_{\mathsf{PHA}} + [x'_i]_{\mathsf{Helper}} - 1) = 0. \tag{2}$$

Therefore, the input $\vec{X'}$ shared between the PHA server and the Helper server is a binary vector, i.e., $x'_i := [x'_i]_{\mathsf{PHA}} + [x'_i]_{\mathsf{Helper}}$ is 0 or 1 for all $i = 1, \ldots, k$.

The intuition for the zero knowledge property of the protocol is that the two verifiers learn a single evaluation of the polynomials $f(x)$ and $g(x)$ at point r that is chosen randomly and independently of any input value. Since these polynomials have high degree, a single evaluation does not reveal anything about the evaluations of these polynomials $f(\omega^{2i})$ and $g(\omega^{2i})$, which depend on x'_i for $i = 1$ to k.

4.5 Amplification of Differential Privacy via Secret-Shared Aggregation

The randomization performed locally by each client is one of the key foundations of ENPA privacy protection, because it guarantees local differential privacy, and can be performed and verified by clients without further assumptions. However, its benefits go further: it provides an initial level of uncertainty that is strongly amplified by ENPA's cryptographic secret-sharing-based aggregation mechanisms. The two different mechanisms, in combination, ensure that PHAs receive as final output only aggregate sums—the components of which are not linked directly to any contributing client—for which strong differential privacy properties can be guaranteed.

Recall that the secret shares transmitted by each client i are the encoding of a randomized report binary vector $\vec{X'}_i = (x'_{i,1}, \ldots, x'_{i,k})$. If there are n clients that send reports, after the final aggregation of the secret shares for those metric reports, the PHA receives element-wise sums of the form

$$\left(z'_1 = \sum_{i \in [n]} x'_{i,1}, \;\; z'_2 = \sum_{i \in [n]} x'_{i,2}, \;\; \ldots, \;\; z'_k = \sum_{i \in [n]} x'_{i,k} \right).$$

n	$\delta < 1/10$	$\delta < 1/100$	$\delta < 1/1000$	$\delta < 1/10{,}000$	$\delta < 1/100{,}000$	$\delta < 1/1{,}000{,}000$
10,000	$\varepsilon_c < 0.53$	$\varepsilon_c < 7.67$	$\varepsilon_c < 7.98$	$\varepsilon_c < 8.00$	$\varepsilon_c < 8.00$	$\varepsilon_c < 8.00$
100,000	$\varepsilon_c < 0.01$	$\varepsilon_c < 0.25$	$\varepsilon_c < 0.46$	$\varepsilon_c < 0.66$	$\varepsilon_c < 0.84$	$\varepsilon_c < 1.02$
1,000,000	$\varepsilon_c < 0.01$	$\varepsilon_c < 0.04$	$\varepsilon_c < 0.10$	$\varepsilon_c < 0.15$	$\varepsilon_c < 0.19$	$\varepsilon_c < 0.23$
10,000,000	$\varepsilon_c < 0.01$	$\varepsilon_c < 0.01$	$\varepsilon_c < 0.03$	$\varepsilon_c < 0.04$	$\varepsilon_c < 0.06$	$\varepsilon_c < 0.07$

Figure 2: The central differential privacy guarantees ε_c that result from unlinkable aggregation of $\varepsilon_0 = 8$ metric report binary vectors, as used in ENPA. As shown in the table, the central guarantee can never be weaker than the original, local ε_0 guarantee; however, the central guarantee gets much stronger as the number of participating clients increases. The ε_c central guarantees are upper bounds (i.e., privacy may be better, but not worse), always stated with the caveat that it is not known whether they hold with δ probability. To quantify the potential impact of this caveat, the columns of the table show what bounds are known to hold for different δ probabilities—e.g., may be expected to hold for 90% and 99% of clients (first two columns) through to 99.9999% of clients (last column).

Subsequently, to estimate the count for each bit in the set of binary vectors input by clients, the PHA need only debias these element-wise sums by computing $z_i := \frac{1}{e^{\varepsilon_0}-1}\left((e^{\varepsilon_0}+1)z'_i - n\right)$ for $1 \leq i \leq k$. The vector $(z_1, \ldots, z_k)$ is an unbiased statistical estimate of the true aggregate counts [DR14].

Intuitively, there is an increased amount of uncertainty in the final element-wise aggregate sums of client contributions seen by the PHA (i.e., the $(z'_1, \ldots, z'_k)$ vector and the unbiased vector $(z_1, \ldots, z_k)$ derived from it). Since they are aggregated over multiple contributions, and each contribution has uncertainty, the uncertainty is compounded such that the final sums allow very little to be inferred about any of the original client contributions. In particular, the sums do not provide any means for discovering which values were present in any single client's contribution: neither whether that client had any bit set in its metric report binary vector, nor whether there was some pattern or correlation in that client's report binary vector.

In short, for $n > 1$ there is no means of discovering the bits of any client's metric report binary vector—in general—by direct inspection of the aggregate sums. However, the possibility remains that (high-confidence) inferences might be drawn indirectly, by inspection of the aggregate sums. Fortunately, this risk can be precisely quantified using the techniques of differential privacy [DR14].

Formally, we can mathematically characterize how privacy of client contributions is improved by the uncertainty in the final element-wise aggregate sums using the techniques of differential privacy. A series of recent results, colloquially known as *amplification by shuffling* [BBGN19, CSU+19, FMT20, UEFM+20], have precisely characterized how the privacy of locally-randomized metric reports is amplified by central processing that eliminates knowledge of which client sent what report—such as the secret-shared aggregation in ENPA.

The most recent of these results, by Feldman et al. [FMT20], gives a tight bound of the central (ε_c, δ) differential privacy guarantee that can result from summing together of n unlinkable ε_0 differentially-private report binary vectors, as in ENPA:

$$\varepsilon_c = O\left(\frac{\sqrt{e^{\varepsilon_0}\log(1/\delta)}}{\sqrt{n}}\right).$$

However, this bound is only stated asymptotically (as per the big-O notation), and is not applicable to all n that might be seen in different-sized ENPA deployments. Also, as is often the case with differential privacy guarantees [DR14], this central ε_c guarantee is given with respect to a δ caveat, i.e., with probability $1 - \delta$ the ε_c guarantee is known to hold, but with probability δ it is not known whether the ε_c guarantee holds—although a weaker ε'_c guarantee is known to hold. For every value of δ a corresponding ε_c guarantee is known to hold; however, this guarantee will get weaker as the δ probabilities imply higher confidence (i.e., as the δ caveat gets smaller). This said, whatever the δ, the known ε_c guarantee can never be weaker than the original ε_0 local guarantee.

Given a specific δ caveat probability, the utility of the final aggregation (i.e., its error compared to an ideal aggregation of respondents' true values) can also be stated asymptotically for k-bit metric report binary vectors—given the number of respondents n and the ε_c bound derived from the δ caveat—as the following ℓ_∞-error bound α, which holds with probability $1 - \beta$:

$$\alpha = \Theta\left(\frac{\sqrt{\log(k/\beta)\log(1/\delta)}}{n\varepsilon_c}\right).$$

Asymptotic bounds can be difficult to reason about concretely—especially when the bounds apply to values such as ε_c and δ that are themselves bounds. Fortunately, to understand the empirical guarantees that will apply in practice, the recent work by Feldman et al. [FMT20] also provides numerical means to evaluate what concrete (not asymptotic) bounds are known to hold for different choices of parameters.

Figure 4.5 shows what concrete guarantees are known to hold for different δ caveats for the concrete ENPA parameters at the different deployment sizes (i.e., choices of n) for which ENPA is designed. The bounds in the figure are computed numerically using the iterative techniques of [FMT20] to derive tight concrete ε_c bounds. In particular, as shown in the table, for aggregation over 100 thousand clients a local randomized response with $\varepsilon_0 = 8$ is amplified to obtain $\varepsilon_c \approx 0.83$ at a δ probability of one-in-100,000. (Not shown in the figure is the concrete expected utility, but for this setting of $n = 100{,}000$ and $\delta = 1/100{,}000$ the expected standard deviation of any sum is approximately 6.)

5 ENX ENPA Construction

The previous section overviewed the main techniques and the general cryptographic design for a private aggregate measurement system. In this section, we describe some additional components of the concrete[1] instantiation we use in the private analytics implementation in ENX.

In its deployment in the United States, the ENPA system is a five-party system presented in Figure 3. In this system, Apple and Google run their respective ingestion servers for the contributions coming from iOS and Android phones. The Helper server is run by the Internet Security Research Group (ISRG)[2]. The PHA server is split in two components: an aggregator with similar functionality to the Helper server, run by the National Cancer Institute in the National Institutes of Health (NIH)[3], and a reconstructor server that adds the shares evaluated by the two computation servers and provides the results in a graphic user interface to the respective health authority, run by the MITRE Corporation[4].

5.1 Asynchronous Communication

An important remark is that ENPA system does *not* need to be run synchronously. Indeed the only requirement is that the final reconstructor server sums the aggregated shares corresponding to the same *batches* (and that the number of valid contributions from the batches over the aggregation window is large enough to offer the privacy guarantees aimed for—cf. Section 4.5). Therefore, the servers communicate with each other by writing signed messages in their respective cloud storage buckets, which are treated as mailboxes.

5.2 Bootstrapping Trust

The ENPA system requires the different parties to know one or several public keys from each other (for authenticity), their cloud identity (for ACL), and the location of their respective storage buckets. In particular, for the public keys:

[1] We refer the interested reader to the open source implementation of the aggregation server has been developed by ISRG and available at https://github.com/abetterinternet/prio-server, and to the open source implementation of the Android client available at https://github.com/google/exposure-notifications-android.

[2] www.abetterinternet.org

[3] www.nih.gov

[4] www.mitre.org

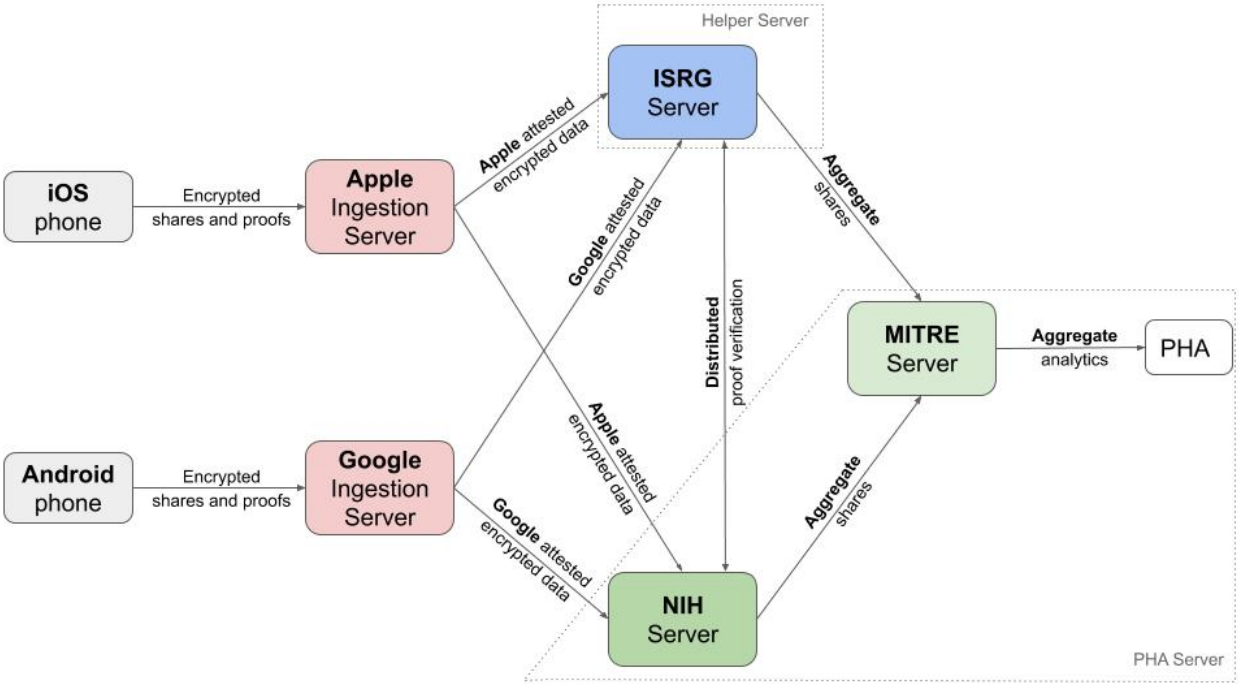

Figure 3: US ENX ENPA Architecture

- The aggregation servers generate encryption keys $\mathsf{PK}^{enc}_{\mathsf{NIH}}$ and $\mathsf{PK}^{enc}_{\mathsf{ISRG}}$ (which needs to be known by the user devices), and generate signature keys $\mathsf{PK}^{sig}_{\mathsf{NIH}}$ and $\mathsf{PK}^{sig}_{\mathsf{ISRG}}$ to attest to the authenticity of the messages they produce (which needs to be known by each other and by the reconstructor server);
- The ingestion servers generate signature keys $\mathsf{PK}^{sig}_{\mathsf{APPLE}}$ and $\mathsf{PK}^{sig}_{\mathsf{GOOGLE}}$ to attest to the authenticity of the messages they produce (which needs to be known by the aggregation servers).

In order to bootstrap the trust in the system, the public keys and cloud storage buckets related information are stored in manifest files, published at well-specified and unique URLs pre-shared among the parties (served over HTTPS). When setting their system up or verifying signatures, each party will therefore read the information it needs from the other parties' manifest files. An additional detail is that Apple CA receives $\mathsf{PK}^{enc}_{\mathsf{NIH}}$ and $\mathsf{PK}^{enc}_{\mathsf{ISRG}}$ in Certificate Signing Requests (CSRs) and issues certificates for these public keys. Finally, the clients are provided with those certificates (which will be verified on device), the parameter ε_0 used in the local randomizer, and the modulus p used for the cryptographic secret sharing.

5.3 Client Processing

For each collected metric the user phones process their inputs as described in Figure 4 and send the results to their respective (Apple or Google) ingestion server. The specific message formats used depend on the ingestion services and out of scope of this whitepaper. However, note that it is (somewhat obviously) important that the user devices and aggregation servers use matching encryption schemes and PRGs; indeed, recall that the helper server will need to expand a seed to obtain the secret shares in the exact same manner as the devices expanded the seed. In ENPA, the devices use ECIES with SHA256 over P256 and AES-GCM with 128-bit key and 128-bit IV for encryption, and the PRG is AES-CTR with 128-bit key and 128-bit IV where each 4-byte output blocks is accepted if it is smaller than p (i.e., rejection sampling is used to ensure that the output produces uniform integers modulo p).

Client Input Processing

- Each client prepares the vector of its metric contributions $\vec{X}_i = (x_{i,1}, \ldots, x_{i,k})$.
- It applies randomized response with parameter ϵ_0 on each value in the vector $\vec{X}_i$ to obtain

$$\vec{X}'_i = (x'_{i,1}, \ldots, x'_{i,k}).$$

- It samples uniformly at random (using good randomness) a 32-byte seed $\mathsf{seed}^{(i)}_{\mathsf{ISRG}}$, parsed as a 128-bit key concatenated with a 128-bit IV.
- It evaluates AES-CTR with the above key and IV and perform rejection sampling until it obtains $k+3+N$ random elements $\rho_1, \ldots, \rho_{k+3+N}$ modulo p.
- For each $j \in \{1, k\}$, the client computes the NIH random share of $[x'_{i,j}]_{\mathsf{NIH}} = x'_{i,j} - \rho_j \bmod p$.
- For each $j \in \{1, k\}$, the client computes a distributed range proof $\pi^{(i)}_{\mathsf{NIH}}$ for the fact that $x'_{i,j} \in \{0, 1\}$ as in Section 4.4, where

$$[f(1)]_{\mathsf{ISRG}} = \rho_{k+1}, [g(1)]_{\mathsf{ISRG}} = \rho_{k+2}, [h(1)]_{\mathsf{ISRG}} = \rho_{k+3}, [h(\omega^{2m-1})]_{\mathsf{ISRG}} = \rho_{k+3+m} \text{ for } m = 1, \ldots, N$$

- The client uses an authenticated encryption scheme and creates

$$\mathsf{ct}^{(i)}_{\mathsf{NIH}} = \mathsf{Enc}\left(\mathsf{PK}_{\mathsf{NIH}}; [\vec{X'}_i]_{\mathsf{NIH}} || \pi^{(i)}_{\mathsf{NIH}}\right)$$
$$\mathsf{ct}^{(i)}_{\mathsf{ISRG}} = \mathsf{Enc}\left(\mathsf{PK}_{\mathsf{ISRG}}; \mathsf{seed}^{(i)}_{\mathsf{ISRG}}\right)$$

- The client sends $(\mathsf{ct}^{(i)}_{\mathsf{NIH}}, \mathsf{ct}^{(i)}_{\mathsf{ISRG}})$ to the ingestion server together with information necessary for device attestation, specific for each company.

Figure 4: ENPA: Client Processing

5.4 Ingestion Server Processing

The ingestion servers for iOS and Android run their respective device attestation checks, and if those pass they process the data they have received as described in Figure 5. The messages posted on the cloud storage buckets of the aggregation servers are Avro encoded[5], and signed using ECDSA with SHA256 over P256 by the ingestion servers. The filenames are defined relative to a deployment-specific bucket and path prefix, and defined in terms of *aggregation id* (a unique identifier/name for the metric, spanning batches), a *batch id* (a unique identified for a specific batch of data), and timestamps.

5.5 Aggregation Servers Processing

The aggregation servers process all the batches they received from the ingestion servers for each aggregation id, for which the signatures are valid, during an *aggregation window* (e.g., an aggregation window may span the 24 hours at days boundaries for the UTC timezone). The cryptographic steps are described in Figs. 6 and 7. Note that the validity zero-knowledge range proofs are organized in the same batches as the inputs, and posted (and signed) on the other aggregation server cloud storage bucket. Finally, both aggregation

[5]The exact encoding of all messages exchanged by servers is specified at https://github.com/abetterinternet/prio-server/tree/main/avro-schema.

Ingestion Server Processing

- For each input that ingestion server receives, it verifies the device attestation. If the check passes, it extracts the two ciphertexts $(\mathsf{ct}_{\mathsf{NIH}}^{(i)}, \mathsf{ct}_{\mathsf{ISRG}}^{(i)})$. If the device attestation does not verify, then the ingestion server discards the packets.
- The ingestion server generates a random identifier u_i for the ciphertexts $(\mathsf{ct}_{\mathsf{NIH}}^{(i)}, \mathsf{ct}_{\mathsf{ISRG}}^{(i)})$ as well as a random challenges r_i for the verification of the range proof included in these inputs.
- The ingestion server forwards in batches the values $(u_i, r_i, \mathsf{ct}_{\mathsf{NIH}}^{(i)})$ to NIH, and the values $(u_i, r_i, \mathsf{ct}_{\mathsf{ISRG}}^{(i)})$ to ISRG.

Figure 5: ENPA: Ingestion Server Processing

servers aggregate the data for which the zero-knowledge range proofs verify over the aggregation window, and post (and sign) the sum to the cloud storage bucket of the reconstructor server. The latter then provides the aggregated statistics to the health authorities at the granularity of the aggregation window (e.g., daily).

ENPA Aggregation Servers Processing

Proof Verification Computation

- The NIH server and the ISRG server decrypt the ciphertexts received from the ingestion server.
- For each input that the NIH server decrypts as

$$[\vec{X}_i']_{\mathsf{NIH}}, \pi_{\mathsf{NIH}}^{(i)} = ([f(1)]_{\mathsf{NIH}}, [g(1)]_{\mathsf{NIH}}, [h(1)]_{\mathsf{NIH}}, [h(\omega)]_{\mathsf{NIH}}, [h(\omega^3)]_{\mathsf{NIH}}, \ldots, [h(\omega^{2N-1})]_{\mathsf{NIH}},$$

  the NIH server evaluates

$$f_{\mathsf{NIH}}(r_i), g_{\mathsf{NIH}}(r_i), h_{\mathsf{NIH}}(r_i)$$

  and sends those values together with u_i to the Helper.
- For each input that the ISRG server decrypts as $\mathsf{seed}_{\mathsf{ISRG}}^{(i)}$, it parses the seed as a 128-bit key concatenated with a 128-bit IV, and evaluates AES-CTR with the above key and IV and perform rejection sampling until it obtains $k+3+N$ random elements $\rho_{i,1}, \ldots, \rho_{i,k+3+N}$ modulo p.
- For each input the ISRG server defines

$$\begin{aligned}[\vec{X}_i']_{\mathsf{ISRG}} &= (\rho_{i,1}, \ldots, \rho_{i,k}),\\ \pi_{\mathsf{ISRG}}^{(j)} &= (\rho_{i,k+1}, \ldots, \rho_{i,k+3+N})\\ &= ([f(1)]_{\mathsf{ISRG}}, [g(1)]_{\mathsf{ISRG}}, [h(1)]_{\mathsf{ISRG}}, [h(\omega)]_{\mathsf{ISRG}}, [h(\omega^3)]_{\mathsf{ISRG}}, \ldots, [h(\omega^{2N-1})]_{\mathsf{ISRG}}\,.\end{aligned}$$

  It then evaluates

$$f_{\mathsf{ISRG}}(r), g_{\mathsf{ISRG}}(r), h_{\mathsf{ISRG}}(r),$$

  and sends those values to the NIH server together with u_i.

Figure 6: ENPA: Aggregation Processing

ENPA Aggregation Servers Processing

Aggregation

- The ISRG server verifies the ranges proofs for each input it has received and let $\mathcal{U}_{\mathsf{ISRG}}^{(B)}$ be the set of identifiers u_i in a batch B of inputs such that

$$\left(f_{\mathsf{NIH}}(r_i) + f_{\mathsf{ISRG}}(r_i)\right)\left(g_{\mathsf{NIH}}(r_i) + g_{\mathsf{ISRG}}(r_i)\right) = \left(h_{\mathsf{NIH}}(r_i) + h_{\mathsf{ISRG}}(r_i)\right).$$

- The NIH server verifies the ranges proofs for each input it has received and let $\mathcal{U}_{\mathsf{PHA}}^{(B)}$ be the set of indices i in a batch B of inputs such that

$$\left(f_{\mathsf{NIH}}(r_i) + f_{\mathsf{Helper}}(r_i)\right)\left(g_{\mathsf{NIH}}(r_i) + g_{\mathsf{Helper}}(r_i)\right) = \left(h_{\mathsf{NIH}}(r_i) + h_{\mathsf{Helper}}(r_i)\right).$$

- For each batch B, the ISRG server computes

$$\vec{S}_{\mathsf{ISRG}}^{(B)} = \sum_{i \in \mathcal{U}_{\mathsf{ISRG}}^{(B)}} [\vec{X}_i']_{\mathsf{ISRG}},$$

  and sends $(\mathcal{U}_{\mathsf{ISRG}}^{(B)}, \vec{S}_{\mathsf{ISRG}}^{(B)})$ to MITRE.

- For each batch B, the NIH server computes

$$\vec{S}_{\mathsf{NIH}}^{(B)} = \sum_{i \in \mathcal{U}_{\mathsf{NIH}}^{(B)}} [\vec{X}_i']_{\mathsf{NIH}},$$

  and sends $(\mathcal{U}_{\mathsf{NIH}}^{(B)}, \vec{S}_{\mathsf{NIH}}^{(B)})$ to MITRE.

- For each batch B, MITRE verifies that $\mathcal{U}_{\mathsf{Helper}}^{(B)} = \mathcal{U}_{\mathsf{NIH}}^{(B)}$. If this does not hold, it discards the batch. Otherwise, it computes

$$\vec{S}^{(B)} = \vec{S}_{\mathsf{ISRG}}^{(B)} + \vec{S}_{\mathsf{NIH}}^{(B)}).$$

Figure 7: ENPA: Aggregation Processing

6 Conclusion

The Exposure Notification System designed by Apple and Google provides a digital solution for detection of possible exposures to COVID-19 that puts user privacy first. This white paper presents Exposure Notifications Privacy-preserving Analytics, which extends the capability of ENS by providing PHAs visibility into the functioning and the effectiveness of ENS while meeting the strong privacy principles of Exposure Notification. Cryptographic secure computation is used to protect the individual contribution of each user, and is augmented with the use of local differential privacy. Range proofs and device attestations are employed for resilience against malicious participants. The combination of these methods allows to provide accurate and significant aggregate metrics to PHAs for epidemiological purposes in a privacy preserving fashion.

References

[AG20] Apple and Google. Exposure notifications: Using technology to help public health authorities fight COVID-19. https://www.google.com/covid19/exposurenotifications/ (accessed 2020-12-08), 2020.

[BBCG+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. Cryptology ePrint Archive, Report 2019/188, 2019. https://eprint.iacr.org/2019/188.

[BBGN19] Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. The privacy blanket of the shuffle model. *arXiv e-prints*, arXiv:1903.02837, 2019. https://arxiv.org/abs/1903.02837.

[CB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX Symposium on Networked Systems Design and Implementation, (NSDI)*, 2017. https://crypto.stanford.edu/prio/ (accessed 2020-12-09).

[CSU+19] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. *arXiv e-prints*, arXiv:1808.01394, 2019. https://arxiv.org/abs/1808.01394.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[FMT20] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. *arXiv e-prints*, arXiv:2012.12803, 2020. https://arxiv.org/abs/2012.12803.

[UEFM+20] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. *arXiv e-prints*, arXiv:1811.12469, 2020. https://arxiv.org/abs/1811.12469.