SenseHash: Computing on Sensor Values Mystified at the Origin

Nojan Sheybani, Xinqiao Zhang, Siam Umar Hussain, and Farinaz Koushanfar, *Fellow, IEEE*

Abstract—We propose SenseHash, a novel design for the lightweight in-hardware mystification of the sensed data at the origin. The framework aims to ensure the privacy of sensitive sensor values while preserving their utility. The sensors are assumed to interface to various (potentially malicious) communication and computing components in the Internet-of-things (IoT) and other emerging pervasive computing scenarios. The primary security primitives of our work are Locality Sensitive Hashing (LSH) combined with Differential Privacy (DP) and secure construction of LSH. Our construction allows (i) sub-linear search in sensor readings while ensuring their security against triangulation attack, and (ii) differentially private statistics of the readings. SenseHash includes hardware architecture as well as accompanying protocols to efficiently utilize the secure readings in practical scenarios. Alongside these scenarios, we present an automated workflow to generalize the application of the mystified readings. Proof-of-concept FPGA implementation of the system demonstrates its practicability and low overhead in terms of hardware resources, energy consumption, and protocol execution time.

Index Terms—Security of Sensor Readings, Security at the Source, Secure Locality Sensitive Hashing (SLSH)

1 Introduction

Embedded sensors like fingerprint and iris readers, GPS transceivers, heart rate, and oxygen saturation monitors collect sensitive information about users. Recently, these sensors are being deployed on edge devices like smartwatches for authentication, tracking, and more. Most of these tasks require communication with a web server. However, the majority of the edge devices lack internet connection and usually communicate with the servers through a third entity, henceforth referred to as the *gateway*. For example, the gateway to the web for smartwatches are smartphone apps to which they are connected via Bluetooth. While many of these apps are provided by the vendors of the edge devices, they allow access to the sensor readings to third-party apps as well (e.g., the workout monitoring apps access GPS data/heart rate sensor, password-free login uses fingerprint/iris to authenticate users to different apps). This setup creates new concerns about the privacy of the sensitive sensor readings. On the one hand, the privacy of the sensor readings needs to be protected from both the gateway and the web server. On the other hand, the privacy of the user data stored in the server needs to be protected from the gateways of different sensors. Most importantly, we need to ensure efficient and meaningful computation on the sensor readings without compromising privacy.

Over the past decade, Privacy-Preserving Computation (PPC) has received considerable attention from the research community [1] and, as a result, has seen orders of magnitude improvement in run-time. However, it still entails a large amount of computation and communication that is not suitable for resource-constrained devices. Two of the primary approaches to PPC are Homomorphic Encryption (HE) [2] which allows computation on encrypted data and Multi-Party Computation (MPC) [3] where two or more parties compute on shared data such that the inputs from all the parties are secret from others. HE requires high computational power and large memory, none of which are available in edge devices. MPC protocols require extensive communication, and as a result, cannot be directly applied to the current scenario where the primary source of the data has limited communication capability. Another important consideration in practical systems is scalability since in many cases, the computations on the sensor reading involve querying a large database (e.g., authentication). In both Homomorphic Encryption (HE) and Garbled Circuits (GC), the complexity of such operations is linear in the number of database entries, while a sub-linear complexity is desired.

In this work, we introduce a novel method based on a judicious combination of hardware and algorithmic approaches to protect the privacy of the sensor readings while ensuring their utility. In the proposed method, the sensor readings are secured at the source such that no entity can access them in plain text. We design a lightweight hardware security module colocated on the same chip as the sensor. The sensor readings go through this module before being placed at the I/O port of the chip. Our hardware design is accompanied by corresponding algorithms to extract the intended utility from the secured sensor readings.

The primary security primitives of our work are Locality Sensitive Hashing (LSH) [4] combined with Differential Privacy (DP) and secure construction of LSH. LSH allows performing Near Neighbor Search (NNS) on a database with sub-linear complexity in terms of the database size. A recent work [5] has shown how to preserve the privacy of the database by employing DP. However, this work assumes access to the entire database by a single party in the preprocessing phase. In several practical scenarios, we need to compute aggregate statistics based on private data from multiple sources. We tackle this challenge by adapting the secure aggregation protocol presented in [6] to combine multiple differentially private sketches while maintaining individual privacy. Our customized version of the protocol significantly reduces the communication and computation complexity and makes it independent of the dimension of the sketch.

LSH, in its vanilla version, is susceptible to triangulation attacks where an adversary is able to deduce the input to the hash function (the plain-text sensor readings) by performing multiple queries to the database. The work in [7] introduced Secure Locality Sensitive Hashing (SLSH) that prevents the triangulation attack while still retaining the sub-linear NNS property of LSH. Our hardware module utilizes the differentially private LSH of [5], which we refer to as the RACE LSH (RLSH) scheme. We adopt the basic concept of SLSH presented in [7] and modify (in some cases improve upon) it to fit the different application scenarios.

While the proposed hardware module ensures the security of the sensor readings by preventing direct access to them, it creates a new challenge – performing meaningful computation on the sensor reading. To demonstrate the practicality of our scheme, we develop two illustrative applications – one demonstrating the computation of aggregate statistics and one demonstrating secure NNS. In the first application, we compute statistics such as minimum, maximum, and mean of health vitals (heart rate, blood pressure, oxygen saturation) of the participants of a group exercise. The individual readings of these vitals are protected via DP. This application also benefits from our customized secure aggregation protocol. As the second application, we present efficient indexing of fingerprints to enable authentication in sub-linear (in terms of database size) time. In addition, we develop the corresponding hardware modules where needed to ensure that plain-text sensor reading is never used for any computation outside the sensor chip. As part of the second application, we present hardware implementation of the Minutia Cylinder Coding (MCC), currently the most efficient method to represent fingerprint readings. This is the first hardware implementation of MCC and can be of independent interest to speed up the authentication process.

Note that for more established privacy-preserving primitives like HE or MPC, there are a number of frameworks [8], [9], [10] that allow for the development of any generic functionality. The applications proposed in this work, while being representative of a large class of generic applications, required the development of specific protocols. However, both HE and MPC require high communication and computational power that are not available in our target devices. The proposed scheme provides a lightweight solution that is amenable to the constraints of the available computing platform. Moreover, both HE and MPC were considered theoretical concepts at the beginning and required extensive research efforts over three decades to be considered for practical scenarios. Even though our LSH-based method lacks enough versatility like HE or MPC, we believe that with strong research effort it is possible to generalize it for a subset of useful applications. In Section 5.3, we outline a workflow to develop custom applications around our hardware module.

We present a proof-of-concept of the hardware module implemented on a SPARTAN-7 xc7s25csga225-1IL. Our implementation demonstrates low resource usage, which is a significant requirement of the target scenario. Moreover, we also demonstrate low energy consumption by the module. As a testament to the lightweight nature of SenseHash, it is possible to run the first representative application for an estimated 17 hours on a fully charged Samsung Galaxy Watch.

In brief, our contributions are as follows.

- We present a novel paradigm to secure the sensor reading at the origin such that no entity has access to the plain-text readings.
- We employ a combination of hardware and software security primitives to provide an efficient privacy-preserving scheme for a resource-constrained platform.
- We present two practical applications to demonstrate the effectiveness of the proposed scheme.
- Proof-of-concept implementation of the proposed hardware module on FPGA and the protocol for the applications demonstrate the practicality of our approach.

The rest of the paper is organized as follows. In the next section, we present a brief overview of the necessary background. Then we review the related work in this field. Next, we present the configuration and implementation of the proposed system. In Section 5, we present two representative practical applications based on our design. In Section 6, we present the performance evaluation results. Finally, Section 8 concludes the paper.

2 PRELIMINARIES

2.1 ϵ-Differential Privacy

Differential Privacy (DP) provides certain privacy by adding a specific kind of noise to the data, which is stored on the server. The noise allows for the computation of statistical properties such as average, variance, etc. of the whole database while ensuring the individual data is kept secure. This kind of privacy approach enhances the privacy in some useful scenarios. For example, if clients are interested in the statistical properties of a whole dataset and they are not interested in individual data, differential privacy can be an effective way to preserve privacy. DP is built on an assumption that the server, which stores all clients' data, is trusted and an attacker will only be able to query the database and then infer as much information as possible based on the results that he receives. An attacker may be able to observe trends within the dataset, but the privacy of the users is guaranteed.

Formally, the degree of privacy is offered with a parameter ϵ.

Definition 2.1. *A randomized function K gives ϵ-Differential Privacy if the Hamming distance for datasets D and D' $d(D, D') \leq 1$ and all $S \subseteq Range(K)$*

$$Pr[K(D) \in S] \leq e^{\epsilon} \times Pr[K(D') \subseteq S] \tag{1}$$

The probability in the equation is taken over the coin tosses of K.

Rather than comparing what an attacker can learn about individual data with and without access to the output of the privacy mechanism, DP focus on limiting the additional risk of every possible situation that may be incurred by an individual. The definition is a property of the mechanism alone and can run in the real world [11]. Some highly accurate differential private solutions are proposed in data mining, statistics, and learning [12], [13], [14], [15].

A main concept of DP is the sensitivity of the real-valued function mapping datasets to real numbers:

Definition 2.2. *Assume all databases have D space. For $f : D \rightarrow R^d$, the sensitivity of f is:*

$$\triangle f = \max_{D,D'} ||f(D) - f(D'))||_1 \tag{2}$$

Generally speaking, low sensitivity allows highly accurate differential private mechanisms [16]. $exponential\ mechanism$ is an insensitive function for evaluating the quality of an output e.g, getting the revenue in an auction. By using this function, high-quality outputs can be obtained in a differentially private fashion [17]. $exponential\ mechanism$ weights each possible data with a density that falls exponentially with its utility based on ϵ, sensitivity, this time, of the utility function. The utility of output is its L_1 distance from the true data.

2.2 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) [18] is an efficient method for approximate Near Neighbor Search (NNS). The notions of both LSH and NNS are associated with a certain measure of similarity. Given a database and a query, NNS returns the entries in the database that have the highest similarity with the query. Standard hash functions aim to avoid collisions and have a plethora of collision-handling techniques. LSH aims to have similar items collide in the hash table. LSH is a family of hash functions with the property that the hash of two inputs with higher similarity has a higher collision probability compared to the hash of two inputs with lower similarity. Formally, for a hash $\mathcal{H}$ to be an LSH

$$Pr_{\mathcal{H}}(h(x) = h(y)) = Similarity(x, y) \tag{3}$$

Different hash functions are available for different measures of similarity. For instance, MinHash is an LSH that preserves Jaccard Similarity and is primarily used for web documents. LSH does suffer from the problem of false positives, in which objects with low similarity are sometimes hashed similarly. This problem can be mitigated by utilizing the compound LSH functions introduced in [19].

2.3 Secure Locality Sensitive Hashing

In LSH, the probability of collision is a monotonically increasing function of the similarity. This makes it vulnerable to the triangulation attack where an adversary can estimate the input to the LSH by performing multiple queries and observing the number of collisions. The work in [7] presents a transformation to convert any generic LSH into its secure version, namely SLSH, which is robust against the triangulation attack. The primary idea of [7] is to consecutively perform several hashes on the output of LSH such that the collision probability is not a monotonically increasing function of the similarity anymore. Instead, the collision probability is high if the two inputs are similar and drops sharply if the similarity decreases. In Section 5, we present our implementation of SLSH to perform fingerprint indexing.

2.4 Private Repeated Array of Count Estimators (RACE) Sketches

[5] introduces a one-pass algorithm to generate a private sketch of a dataset. The result of this algorithm is a Repeated Array of Count Estimators (RACE) sketch $S_D \in \mathbb{R}^{RxW}$, where R is the number of functions from an LSH family $\mathcal{H}$ on a data point x, $\{h_0(x), h_1(x), ..., h_R(x)\}$, and W is the number of columns in the RACE sketch. For every data point, we use the output of the hash function $h_r(x)$ to increment row r and column $h_r(x)$ of the RACE sketch. This RACE sketch can be thought of as a summary of a large dataset. By adding zero-mean Laplacian Noise to this RACE sketch, we obtain an ϵ-differentially private summary of the data, which allows the RACE sketch to release LSH kernel sums without leaking information about the dataset. This work shows the feasibility of performing density estimates on a dataset, but can be simplified to show dataset statistics, such as the mean. Compared to other works, the algorithm presented in [5] achieves a much more efficient generation of private sketches of a dataset while achieving rigorous error bounds.

2.5 Secure Aggregation

Secure aggregation protocols allow N parties to compute the sum of N L-element vectors. In this work, we employ the protocol presented in [6]. In the Honest-but-Curious (HbC) security model, the computation and communication complexities of this protocol are respectively $\mathcal{O}(\log^2 N + L \log N)$ and $\mathcal{O}(\log^2 N + L)$ per client, and $\mathcal{O}(N \log^2 n + NL \log N)$ and $\mathcal{O}(N \log^2 N + NL)$ for the server. It requires three rounds of interactions between the server and clients. It is important to note that the protocol presented in [6] is not a fully MPC protocol. Rather, it builds upon the MPC protocol for secure aggregation presented in [20] by reducing the communication channels, only having communication channels between each client and the server, resulting in a two-party protocol. As mentioned in [6], many MPC protocols require communication between all clients. This often results in problems with scalability with respect to a number of clients. In the semi-malicious setting, the protocol we utilize only requires communication with 3% of clients, which is why the work is able to achieve poly-logarithmic runtime with respect to the number of clients. This method also makes handling dropouts much easier, although we do not discuss it in detail in our work.

2.6 Fingerprint Terminology and Minutia

Fingerprint indexing involves two basic steps. First, the raw sensor data is represented as a Minutia Cylinder-Code (MCC) [21]. Then the LSH of the MCC representation is computed. The LSH is either inserted in the hash table (registration) or used a query (authentication). The raw fingerprint is composed of a set of lines. Most of the lines flow parallel, and some of them make a pattern. There are three fingerprint patterns (1) ridge ending, (2) bifurcation, and (3) short ridge (dot). These patterns represent some specific features of a fingerprint that can be extracted by using a fingerprint scanner. The minutiae, or Galton's characteristics, are determined by the termination or the bifurcation of the ridge lines. The minutiae data we use in this paper are represented in the ISO/IEC 19794-2 [22] format. Each data consists of three values (x, y, θ), which are x location, y location and direction respectively. The minutiae-based representation includes spatial contribution and directional contribution.

In the matching process, first, the similarity for each minutia called local similarity, is computed. One set of fingerprint data consists of around 60 to 70 minutia data and each data can build a cylinder to calculate the local similarity. The way to calculate it is to compare both cells at the same location of two cylinders. If both of the values are the same, the similarity value of this cell is 1; After going through all the cells in both cylinders, the value of the similarity $\gamma(a, b)$ is computed as follows.

$$\gamma(a,b) = \begin{Bmatrix} 1 - \frac{\|c_{a|b} - c_{b|a}\|}{\|c_{a|b}\| + \|c_{b|a}\|} & if\ C_a\ and\ C_b\ are\ matchable \\ 0 & otherwise \end{Bmatrix} \tag{4}$$

where

$$c_{a|b}[t] = \begin{Bmatrix} c_a[t] & if c_a[t]\ and\ c_b[t]\ are\ matchable \\ 0 & otherwise \end{Bmatrix} \tag{5}$$

$$c_m[lin(i,j,k)] = C_m(i,j,k) \tag{6}$$

$$lin(i,j,k) = (k-1) \cdot (N_s)^2 + (j-1) \cdot N_S + i \tag{7}$$

and i, j, k refers to the location of each cell in a cylinder. The equations above compute the local similarity. The next step is to get the overall similarity. A single value $(globalscore)$ that represents the overall similarity needs to be introduced. Some simple approaches like Local Similarity Sort (LSS), Local Similarity Assignment (LSA), and Local Similarity Sort with Relaxation (LSS-R) are very popular and easy to be implemented on a small design. The basic idea behind LSS is that it sorts all the local similarities and selects the top n_P. In Section 5, we will use minutiae data to do our secure hashing for fingerprint indexing.

3 Prior Work

To the best of our knowledge, SenseHash is the first work that suggests securing the sensor readings at the source and preventing plain-text access to them. In this section, we describe previous works on private RACE sketches, LSH, and fingerprint indexing.

The work [5] presents an algorithm that creates a private summary of a dataset in the form of a matrix $S_D \in \mathbb{R}^{RxW}$, where R is the number of LSH functions performed on each data point and W is the number of columns. For each data point, R independent hash functions are performed, resulting in some number $h_r(x)$, which is used to increment S_D at row r and column $h_r(x)$. Zero mean Laplacian noise is added to S_D to achieve ϵ-differential privacy. Using this private summary of the dataset, S_D, the authors present several machine learning tasks that can be achieved by performing queries. Some of these tasks include linear regression, kernel density estimation, and naive Bayes classification. We simplify this algorithm and allow each user to be the only one with access to the private summary of their sensor readings. The server holds the private summary of all users' data and is updated periodically. We present a hardware module that performs the R LSH functions that are described in [5] and updates the private summary before being shared with the server. Finally, we utilize the secure aggregation technique presented in [6] to ensure that the summary of all users' data on the server is correctly updated with each user's private summaries, without leaking any information.

The work in [23] presents an implementation of a secure approximate search of Content Addressable Memory (CAM) using SLSH. This implementation stores distance-preserving embeddings in the CAM. This allows for faster lookup times, as the previously expensive search for data object similarity has been reduced to a Hamming similarity search. The embeddings in the CAM preserve Jaccard and Cosine distance. We implement the same general SLSH, but our implementation is built on a hardware module and focuses on preserving similarity between fingerprints.

[24] discusses the feasibility and limitations of using HE on commercial off-the-shelf (COTS) Internet of Things (IoT) edge devices to secure sensor data and, on a bigger scale, enable signal processing applications in the encrypted domain. This work highlights the BGV HE scheme, which employs very heavy computation on the server-side. Alongside the heavy server computation, [25] presents a hardware implementation of BGV encryption, in which there is a much higher resource utilization and slower runtime than SenseHash. The combination of high complexity and comparatively slow performance on both the client and server sides shows that HE is not an ideal solution for our proposed applications. Similarly, [26] introduces an FPGA implementation of an MPC encryption scheme. This state-of-the-artwork is only able to achieve runtimes in the ms-range, while requiring much higher resource utilization. HE and MPC are very effective algorithms, but due to their relative novelty, there is still a large research effort required to support lightweight operations in edge devices that can enable applications with runtimes comparable to plaintext operation.

[27] presents an algorithm that preserves location privacy using LSH. This implementation assumes an untrusted server that holds the location data and a trusted server that anonymizes the user's data before sending the data to the untrusted server. This is done by utilizing a K-anonymous approach at the trusted server so that the untrusted server only receives a cloaked version of the user's location. Our implementation does not require the use of any trusted servers, as we are implementing security at the hardware module. This means that data is secure even if the operating system or server is breached. While we do not present an application that preserves location privacy, it is a prevalent problem that can be solved using our SLSH techniques.

For fingerprint indexing, [21] shows the basic idea to represent and match fingerprint minutia data. It is an excellent starting point, and the representation is very computationally efficient. It can convert the value of each cell to a binary representation and use Local Similarity Sort (LSS) or other methods to get a global score of each fingerprint data. [28]

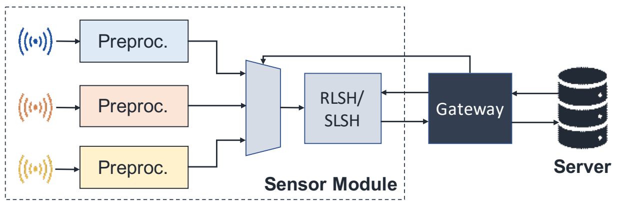

Fig. 1: Overview of the proposed system

proposes a way to improve the regular MCC representation and their experiment result shows the proposed indexing approach dramatically improves the performance of fingerprint indexing. However, they did not consider security, and the data will leak easily at every terminal. [29] uses the Garbled Circuit (GC) protocol to perform fingerprint authentication, but they do not implement them on an FPGA with a sensor. Also, MCC has some advantages over Garbled Circuit. For example, (1) MCC is bit-oriented coding and it is extremely simple and very fast. (2) The border problem can be easily managed without extra burden. Hardware implementation is a little bit tricky because, during MCC, we use the Gaussian function and sigmoid function. [30] introduces a piecewise linear approximation (PLA) method when building a neural network with FPGA and applies it to the $sigmoid$ and $tanh$ function. The result in [30] looks outstanding, and we apply the same method in our paper.

4 System Description

In this section present the hardware module of SenseHash, which is located in the same chip as the sensors. Any sensor reading goes through this module before being placed at the I/O port of the chip for access by external entities.

4.1 Configuration

The configuration of the proposed system is presented in Figure 1. It involves three modules: the *sensor module*, the *gateway*, and the *web server* as described below.

The sensor module may reside either on an edge device or in a dedicated location in an embedded system. It can incorporate any of the sensors in the device (e.g., oxygen saturation, heart rate, blood pressure, fingerprint, etc) and the SLSH/RLSH protocol execution module. Since the sensor readings do not leave the module in plain text, any required preprocessing on the readings before computing the hash is also performed inside the sensor module. We reiterate that the presented method works for sensors that are colocated on a chip, or within the same circuit as the processing unit, which means that we are able to support many sensors in the RLSH application. We discuss this further in section 5.1. The problem SenseHash aims to address is plaintext sensor data reaching a network-connected gateway, so we do not support processing being outsourced to a network-connected gateway.

The gateway may be a smartphone to which the edge device is connected via a local connection (e.g., Bluetooth). Another possible setup is having both the sensor and the gateway on the same platform. In that case, the gateway is the Operating System (OS) of the platform. This ensures that the sensor readings are safe even if the OS is compromised.

There are two two-way communication channels in the proposed architecture: one between the sensor module and the gateway and the other one between the gateway and the server. Note that any outgoing reading from the sensor module to the gateway goes through SLSH/RLSH. Moreover, there is no direct channel between the sensor module and the server.

We refer to the owner of the sensor module as the *user*. All the sensor readings belong to the user. The user also controls the gateway. However, she wants to ensure the security of the sensor reading in case the gateway is compromised.

4.2 Security Model

We assume an honest-but-curious security model in this work. In this model, all the parties: sensor module, gateway, and web server follow the agreed-upon protocol yet may try to deduce more from the information at hand. Moreover, we assume the absence of collusion between the gateway and the server.

Input Privacy. The goal of the proposed system is to ensure that (1) neither the gateway nor the server has access to the raw sensor reading, (2) none of the parties can deduce the raw sensor reading for the data stored in the server. Note that the gateway device does not hold any sensitive information.

Output Privacy. The output of the computation is revealed to the server. The server should not be able to confidently deduce the user's raw sensor reading.

Limitation. The proposed system ensures that none of the servers and the gateway is able to get access to the plaintext sensor reading. However, one possible attack performed by the gateway is the denial of service (DoS). Since in our setting the sensor modules do not have independent communication capability, prevention of such attack is not possible.

4.3 RLSH Implementation

Algorithm 1 describes the general implementation of the RLSH transformation performed on SenseHash hardware module. This undergoes slight changes based on the application that is being developed, but the changes do not have a significant impact on hardware performance. This concept will be explained further in Section 5. In our applications, we project each sensor reading onto the 2D plane and sample functions from the Euclidean LSH family [4], which preserves similarity in terms of Euclidean distance. As mentioned, Algorithm 1 is implemented entirely on hardware, hashing the sensor data before the data reaches the server.

Algorithm 1 also describes the process of populating the RACE sketch for each data point. Note that adding the Laplacian noise is done when the matrix is created, and not when data is actually added. Adding this Laplacian noise makes the RACE sketch ϵ-differentially private. It is important to note that each user must use the same R independent hash functions from hash family H, to ensure successful computation on the server. The data sent from the sensor module is in the form of a private RACE sketch, as seen in the algorithm.

If data is distributed among N users, each user creates their own private RACE sketch that contains the summary of their own data. When the computation is needed, the servers compute the aggregate of the private RACE sketch of users. To ensure that no data is leaked in the aggregation process, we employ a modified version of the secure data aggregation technique introduced in [6]. This concept is further described in section 4.4.

Algorithm 1: Hashing a data point

```
Input      : Datapoint $X$ containing $T$ sensor readings
Output     : Private RACE Sketch $S_U \in \mathbb{R}^{T\times R\times W}$
Parameters: Number of Users, N
            Number of columns in sketch, W
            Privacy budget $\epsilon$
Initialize : $R$ independent hash functions from LSH family $\mathcal{H}$, $\{h_0, h_1, ..., h_r\}$
             $S_U$ += $\lfloor Z \rfloor$, where $Z$ is $Lap(R(\epsilon N)^{-1})$
1 $i = 0$
2 for $t \in X$ do
3     for $r \in R$ do
4         Increment $S_U[i][r][h_r(t)]$
5     end
6     Increment $i$
7 end
8 return $S_U$
```

[5] presents a general query algorithm that can be used for various tasks, such as calculating kernel density estimates of a dataset with error bounds of 1%. In Algorithm 2, we present a modified version of their query algorithm that simply allows us to compute the average of the collected sensor data, while maintaining ϵ-differential privacy and the 1% error bounds. This is done by declaring minimum and maximum values. The work in [5] provides proof of security highlighting the fact that constructing the RACE sketch as we do allows for unlimited queries without exceeding our privacy budget. The sketch that the server performs computation on is the aggregation of all users' RACE sketches, or, if only one user is present, that user's RACE sketch.

4.4 Collaborative RLSH Implementation

The above construction of RLSH assumes the entire database X to be held by a single entity. However, in certain applications X is distributed among N users such that $X = \bigcup_i X^i$. In this case, each user $i \in [N]$ constructs their individual sketch S_U^i and the final sketch $S_U = \sum_i = S_U^i$. Since, the individual S_U^i may reveal sensitive information about the user, this summation is computed through the secure aggregation protocol of [6]. However, a straight application of this protocol would incur computation and communication complexities of respectively $\mathcal{O}(\log^2 N + L \log N)$ and $\mathcal{O}(\log^2 N + L)$ per client, and $\mathcal{O}(N \log^2 n + NL \log N)$ and $\mathcal{O}(N \log^2 N + NL)$ for the server. In this particular scenario, $L = T \times R \times W$. This would be impractical for the edge devices.

We tailor the protocol for the requirement of our configuration. The secure aggregation scheme is performed on

Algorithm 2: Calculate average of data

```
Input      : RACE Sketch $S \in \mathbb{Z}^{TxRxW}$
Output     : $avg_t$, average of gathered data, per sensor reading type
Parameters: Same $R$ hash functions from Algorithm 1, $\{h_0, h_1, ..., h_r\}$
            For each $t \in T$, data range $[min, max]$
1 for $t \in T$ do
2     $N = R^{-1} * \sum_{i,j} S[t][i][j]$
3     for $q \in range(min, max)$ of $t$ do
4         q_avg = 0
5         for $r \in R$ do
6             q_avg += $R^{-1} * S[t][r][h_r(q)]$
7         end
8         $avg_t += \lfloor q\_avg \rfloor * q * N^{-1}$
9     end
10    return or store $avg_t$
11 end
```

the network-connected gateways. Note that, the ultimate purpose of the sketch S_U is to compute the aggregate statistics on the database X. In our custom protocol, each user i adds a random secret number r_i to each sensor reading $t \in X$ during the computation of Algorithm 1. This ensures that the users can send the sketch S_U^i in plain-text to the server, who simply computes the aggregate sketch S_U. The server then computes the mystified average q' following Algorithm 2, where $q'\_avg = q\_avg + \sum_i r_i$. To recover the actual average q_avg, from q'_avg, the server needs to securely compute $\sum_i r_i$. In this way, we have reduced L to 1 instead of $T \times R \times W$.

Using our optimizations to reduce L, we are able to maintain poly-logarithmic communication and computation complexities overheads on client and server. As we are assuming relatively resource-limited edge devices with a powerful server, we highlight that client computation and communication, respectively, only incur overhead of $\mathcal{O}(\log^2 N + \log N)$ and $\mathcal{O}(\log^2 N)$. The secure aggregation technique that we utilize in the Honest-but-Curious setting can support a billion clients. To measure server runtime for RLSH secure aggregation in a reasonable scenario, we initialize clients with a RACE sketch containing $60k$ cells and vary the number of clients from 30-100. The results can be seen in table 1.

TABLE 1: Server runtime of secure aggregation with varying number of clients

# of Clients	Runtime (ms)
30	131.24
40	184.42
50	214.371
70	321.057
100	427.456

4.5 SLSH Implementation

Privacy parameter k. At this point, we would like to further explain the parameter k, and how it relates to privacy. From [7],

Definition 4.1. ($\epsilon - Secure\ Hash\ at\ Threshold\ s_0$). *For any x and y with Similarity(x,y) $\leq s_0$, we call a 1-bit hashing scheme h_{sec} at threshold s_0 $\epsilon - secure$ if the probability of bit-matches satisfies*

$$\frac{1}{2} \leq Pr(h_{sec}(x) = h_{sec}(y)) \leq \frac{1}{2} + \epsilon$$

k is a privacy knob that can be tuned to generate an $\epsilon - secure$ hash. Higher values of k will result in finer hashes, which means the similarity between objects must be much higher for them to collide.

Any LSH function drawn from some LSH family $\mathcal{H}$ can be transformed into an SLSH function. The work in [31] presents a novel LSH family that preserves MCC similarity, which we will refer to as $\mathcal{H}_{MCC}$. To perform the SLSH protocol and create a secure L-bit embedding of a single fingerprint, we perform k independent hashes drawn from $\mathcal{H}_{MCC}$, L times. The first step is to hash each data point x_i k times. These hashes are used in a universal hash function, seen in the following equations [7], to generate a 1-bit representation of the data point.

$$h_{univ}(x_1, ..., x_k) = (\sum_{i=1}^{k} a_i x_i) mod p, mod 2$$

$$h_{sec}(x) = h_{univ}(h_1^{mcc}, ..., h_k^{mcc})$$

where a are fixed random integers and p is a large prime number. This will output a 1-bit representation of the data point, but it is not yet secure. $h_{sec}(x)$ should be performed L times to create an L-bit embedding of any data point x_i. If k is equal to 1, the output will be the same as standard LSH.

We assume each value is represented as a 32-bit floating-point value, so each time, the module reads one data point and stores it to an array. We design a finite-state machine (FSM) to execute the SLSH protocol. For the fingerprint indexing application, the SLSH module will only operate when needed and will not be working continuously, so dynamic power consumption is not a pressing issue.

4.6 Security Analysis

Although we modify the private RACE sketch generation to support a distributed setting, the security analysis presented in [5] is still applicable to our work. We modify the scaling of the Laplacian noise generation function to be sensitive to the number of users alongside the number of hash functions. By doing this, we maintain the zero-mean Laplacian noise that is necessary for the ϵ-differential privacy we introduce in section 2.1. In our SLSH implementation, we perform a novel combination of an LSH function for fingerprint embeddings and the SLSH transformation in our work. As the security of the SLSH transformation is not dependent on the LSH function it is applied on, the security of SenseHash's SLSH module holds under the analysis shown in [7].

5 Applications

In the previous section, we describe the hardware module that mystifies the sensor reading at its origin. However, preventing access to the plain-text sensor reading makes their efficient use a challenge. In this section, we present two practical applications that are realizable via SenseHash. The primary goal while developing these applications is to ensure that sensor readings are used only for the intended purpose. We present end-to-end implementations of all the proposed applications and evaluate their performance.

5.1 Monitoring Health Vitals

Wearable devices are becoming more prevalent, which is resulting in a massive uptick of personal data that must be protected. Many current privacy protocols rely on performing operations on plaintext data, but this can result in leakage of information. As edge devices collect more information about the users, such as health vitals, it is important to be able to perform meaningful computation on the collected data, while ensuring that no data is being leaked in the process. Using our RLSH scheme, we present an ϵ-differentially private application that allows a server to monitor the health vitals (e.g. heart rate) of a group, such as a gym class, without any information about a specific user being leaked.

We adopt a simplified technique of the scheme presented in [5]. In this setting, users each have their own private RACE sketch S_U, for each type of sensor reading they are tracking. For instance, if we wanted to monitor heart rate and oxygen saturation, each user would store two sketches, $S_{U-heart}$ and $S_{U-oxygen}$, which can be conceptualized as a $2 \times R \times W$ matrix where $S_U[0]$ is $S_{U-heart}$, and update them accordingly. For the sake of simplicity, we assume that we are only monitoring heart rate, knowing that we can easily scale to monitor more vitals if necessary.

The user's private sketch must be initialized with zero-mean Laplacian noise to uphold the ϵ-differential privacy utility. The noise is scaled by the number of users so that it does not grow too large when all the data is aggregated. Upon each sensor reading, we update the user's private sketch using the algorithms shown in Section 4. The server

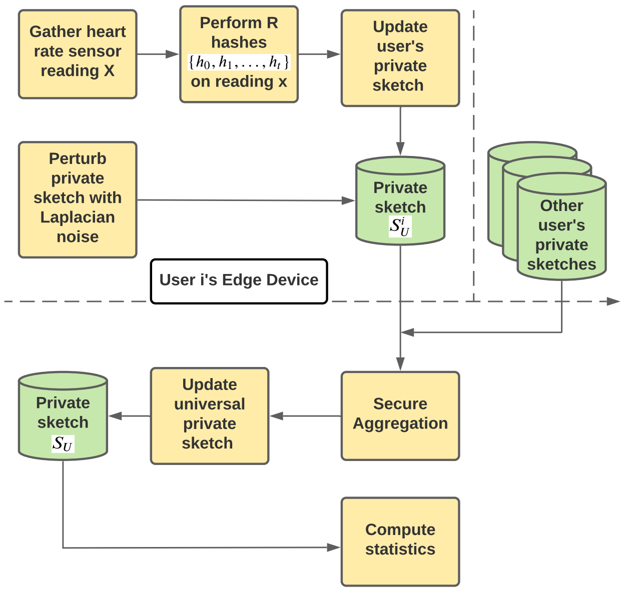
Gather heart rate sensor reading X
Perform R hashes $\{h_0, h_1, \ldots, h_t\}$ on reading x
Update user's private sketch
Perturb private sketch with Laplacian noise
Private sketch S_U^i
Other user's private sketches
User i's Edge Device
Private sketch S_U
Update universal private sketch
Secure Aggregation
Compute statistics

Fig. 2: Heart rate monitoring workflow

has a single or multiple universal private RACE sketches, S_{univ} that contain the aggregation of all users' private sketches. Due to the Laplacian noise, being scaled by the number of users, the aggregation of the sketches is also ϵ-differentially private.

The server uses the secure aggregation technique introduced in [6] to collect and update S_{univ} periodically. The server contains multiple S_{univ} sketches if we are aiming to look at different periods and make comparisons of the data collected within those periods. For instance, if we are looking at the trends of the group's average heart rate at each minute, we would have a private sketch on the server for each minute that we collect data, while the private sketch on the edge device would be the same process, but with each user's S_U reinitialized each minute. This also works if we want to make smaller subgroups, such as calculating statistics by weight.

Once enough data has been processed, the server can initiate the process described in Section 4 to calculate the average heart rate of the group, maintaining ϵ-differential privacy and the 1% error bounds achieved in [5]. The general workflow for this application can be observed in the figure 2. We reiterate that, while in this context we use heart rate as the data, this application would work with any health vitals and multiple health vitals at one time.

This application could also be achieved without users storing their own private sketches by using SLSH hashes instead of LSH hashes in the RLSH process. In this case, the users would share the results of their R SLSH hashes with the server, and the server would update the universal sketch with increments in indices based on those results. In theory, this works and avoids the triangulation problem that faces LSH, but it does not ensure differential privacy, so it is not further explored in this work.

5.2 Fingerprint Indexing

Biometric sensors such as fingerprint/iris readers are one of the most security-critical components of modern devices. The majority of smartphones in recent years include either or both of them. Fingerprint sensors are also on the verge of appearing on smartwatches [32]. One of the most complex tasks involving fingerprints is indexing or continuous classification [33], which provides an efficient method to identify a fingerprint in a large database. We now show that such indexing is compatible with the setup proposed in this work.

We adopt the indexing method presented in [31]. The raw fingerprint reading is represented in the ISO/IEC 19794 [22] minutiae template. A minutia m is a triplet ($m = x_m, y_m, \theta_m$), where x_m and y_m are the minutia location and $\theta_m \in [0, 2\pi]$ is the minutia direction. Each minutia is first encoded into a fixed-length binary vector through Minutia Cylinder Codes (MCC) [21]. MCC is a very effective way to map a minutiae-based representation into a set of binary vectors [28]. The general flow is shown in Figure 3.

The minutiae-based representation includes (1) spatial contribution and (2) directional contribution[21]. $C_S^m(m_t, p_m^{i,j})$ is the spatial contribution that minutia m_t offers to cell (i, j, k). Spatial contribution is defined as a euclidean distance function between m_t and $p_m^{i,j}$:

$$C_S^m(m_t, p_m^{i,j}) = G_S^m d_S(m_t, p_m^{i,j}) \tag{8}$$

where $G_S(t)$ is a Gaussian function with zero mean and σ_s standard deviation.

Directional contribution of m_t is represented as $C_S^D(m_t, d\varphi)$. A directional contribution function consists of (1) $d\varphi_k$ and (2) the directional difference between two initiates. If (1) and (2) are close to each other, a high contribution value will be got. The directional contribution is:

$$C_m^D(m_t, d\varphi_k) = G_D(d(d\varphi_k, d_\theta(m, m_t))), \tag{9}$$

where $d_\theta(\theta_1, \theta_2)$ is the difference between two angles θ_1 and θ_2. This value is in range $[0, \pi]$. $G_D(\alpha)$ is the area under Gaussian distribution.

After calculating all the contributions, we multiply the two contributions and get the overall contribution $C_m(i, j, k)$, where i, j, k is the location of each cell. Then, after an activation faction (sigmoid), a threshold value of μ_Ψ is set to select cells (i, j, k) that have a high contribution value. By applying this, we convert all the contribution results to bit-based results. The bit-based values are inputs to the SLSH module and get the final hash value.

In this project, the minutiae template data is from [21], [31], [34], [35]. Since the format of a random number is a 32-bit floating-point, and for each calculation, the MCC module uses around 500 random values, and they are about $16k$ bits for each calculation. We minimize the usage of input ports by feeding the random number sequentially. Since the fingerprint indexing job does not need to run at all times and it only runs when someone needs indexing, the energy consumption is negligible.

Both spatial contribution and directional contribution use the Gaussian distribution function We build a look-up table inside the FPGA module and use it as the output of the Gaussian function. The size of the look-up table is 256 numbers each so the value fits in an 8-bit binary number. Note that after the multiplication of two contributions, a sigmoid activation function is attached. We also apply the look-up table for this function. Since this non-linear activation function is less complicated than the Gaussian function, we utilize the piecewise linear approximation (PLA) technique [30]. In our design, we segment the sigmoid function uniformly in the range $y = [-5, 5]$, [30] shows that the mean squared error(MSE) of the piecewise linear approximation of the logistic sigmoid function. The error is only 10^{-5} if we use 10 segments for the sigmoid function. Also, since the function is symmetric, we just need to calculate the range $y = [0, 5]$. For an input $y_n < 0$, the output of our sigmoid function is $f_l(y_n) = 1 - f_l(|y_n|)$. Therefore, a look-up table with 10 values is used to represent the sigmoid function in the MCC design. Each MCC output data has 600 features.

The work in [31] presents an LSH construction along with an effective search algorithm to retrieve the fingerprints. We apply the transformation of [7] on the LSH construction to form an SLSH compatible with fingerprint indexing. Moreover, since the MCC encoding is performed before LSH, we design a hardware module for the encoding on the sensor reading. The resulting output of the hardware module is directly compatible with the search algorithm.

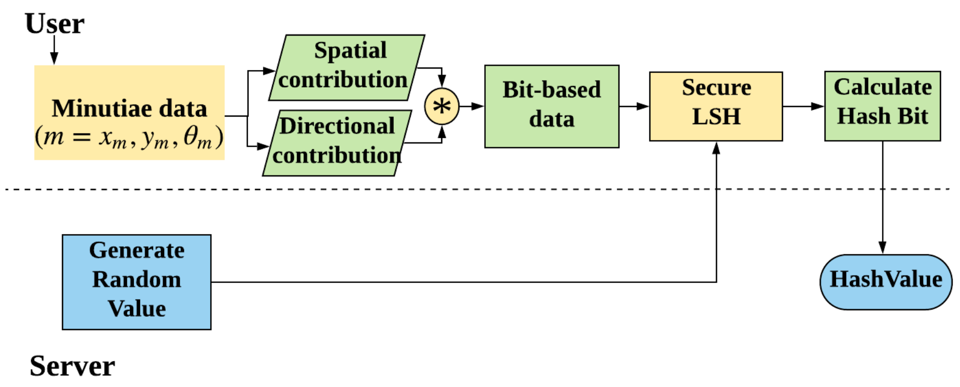

Fig. 3: Global flow of SenseHash Minutia Cylinder Codes application.

Note that the output of the hardware module is still useful for more regular applications, such as authentication. In this case, we use the same setup above only difference being the size of the database, which includes only a limited number of fingerprints all belonging to the owner or the edge device.

5.3 General Purpose Applications

While we present two specific applications in this work, we reiterate that the algorithms used to achieve these applications are general and allow for a multitude of applications without too much change. The RLSH scheme works with any sensor data as long as a minimum and the maximum value is known (e.g. air quality). The only changes that are required with general RLSH applications are changing the periodicity of updates to the universal private sketch on the server, and what kind of statistics you would like to calculate. The general structure of an RLSH application can be seen in figure 4.

The SLSH scheme is a bit more difficult to generalize, but still quite straightforward. The main aspect of the SLSH scheme that must be modified is the LSH functions that are transformed into SLSH functions. Different LSH families preserve different similarities. For instance, the MinHash LSH family preserves Jaccard distance, which works well for web documents but performs poorly on GPS coordinates. The family of LSH functions that we draw from is highly dependent on the input data and application. Hardware modules can be designed that define the LSH functions used to work with the hardware module we have built that performs the SLSH transformation. Server operations can be modified to fit the application, if indexing is not the desired application. The general structure of an SLSH application can be seen in figure 5.

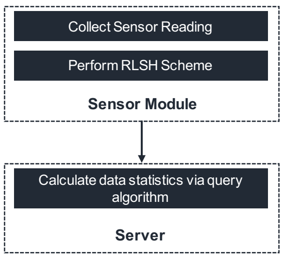

Fig. 4: General RLSH application workflow

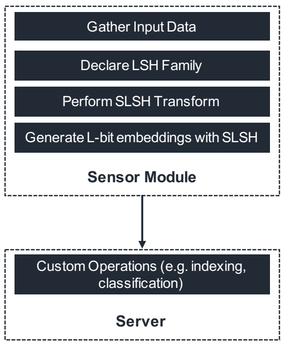
Gather Input Data
Declare LSH Family
Perform SLSH Transform
Generate L-bit embeddings with SLSH
Sensor Module
Custom Operations (e.g. indexing, classification)
Server

Fig. 5: General SLSH application workflow

6 Evaluation

We present a proof-of-concept implementation of the proposed systems as well as the application on the SPARTAN-7 xc7s25csga225-1IL. Our design applications use Xilinx Vitis HLS 2020.2 and Vivado 2020.2 tools. In this section, we first describe the implementation details and then present the evaluation of resource utilization, run-time, and power consumption.

6.1 Monitoring Vitals

We build a low-power heart-rate monitoring application using our RLSH scheme. First, we assume the hardware processes one data point at a time with 1-minute intervals. Then, the sensor module will update its stored private RACE sketch, add Laplacian noise, and send it to the server. We set R to 40 for this application.

TABLE 2: Resource Utilization of Monitoring Vitals

Resource	Utilization
LUT	17
FF	20
IO	46
BUFG	1

Also, we need to note that to generate random numbers for the LSH and Laplacian noise, we use a Pseudo Random

Number Generation called the Linear congruential generator (LCG). After getting the random numbers, both random projections and random Laplacian noise can be generated by doing the necessary calculations. The LCG hardware module produces a random number in $225ns$, and the random numbers can be generated in parallel. The resource utilization for of the LCG hardware module to produce a random number can be seen in figure 3. We highlight that random numbers for differential privacy, as described in section 4.3, and sketch masking, as described in section 4.4, are only generated once during initialization.

TABLE 3: Resource Utilization of LCG

Resource	Utilization
LUT	414
FF	245
DSP	9

TABLE 4: Power Utilization of Monitoring Vitals

Resource	Power Consumption
Total Power	0.068W
Dynamic Power	0.009W
Static Power	0.059W
Clock Power	0.002W

Our design aims to generate random numbers and then run all the hash functions in parallel. We also make the design flexible to adapt different secure level of hash functions. NUM_HASH is a parameter in this application and it can be changed to fit any circumstance.

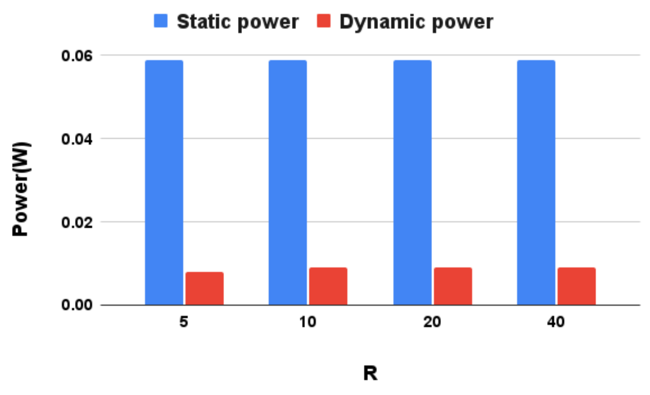

Fig. 6: Power consumption with different values of R.

For the power consumption, Table 4 and Figure 6 show that the total power is $0.068W$ and static power takes most of it. Table 4 also represents that our design is very lightweight with 17 Look Up Tables (LUT), 20 Flip Flops (FF), and 46 Input/Outputs (IO). Moreover, we can see from Figure 6 that both static power and dynamic power remain almost the same as R increases, which indicates that our design can be extended to a bigger or smaller R, resulting in different utility guarantees, without much of an increase in power.

Table 5 shows the parameters we use for our design. The clock period is 3ns.

TABLE 5: Parameter Values for Monitoring Vitals

Parameter	Value
R	40
clock period	3.000ns

The plaintext version of this application does not require any processing before reaching the gateway, as the secure aggregation is done on data that is sent to the gateway. The overhead of the secure aggregation we utilize is discussed in section 4.4. As SenseHash presents a method for mystifying sensor data before reaching a network-connected gateway, the measurements we present can all be considered overhead to make the plaintext version of this application privacy-preserving. The average runtime to hash a data point in the privacy-preserving setting we present is $222.4\mu s$.

6.2 Fingerprint Indexing

Fingerprint indexing is implemented using the proposed SLSH scheme. The raw fingerprint reading is represented in the ISO/IEC 19794 [22] minutiae template. The MCC design before indexing is implemented separately from the SLSH model. Table 7 shows the value of the parameters we use.

TABLE 6: Resource Utilization of Fingerprint Indexing

Resource	Utilization
LUT	1842
FF	2545
IO	193
BRAM	20
DSP	22
LUTRAM	41

TABLE 7: Parameter values of MCC

Parameter	Value	Parameter	Value
R	140	μ_Ψ	0.002
N_S	10	clock period	3.814ns
N_D	6	output feature	600
Max_N_S	32		

From Table 6, we see that our design uses less than $2k$ LUTs and around $2.5k$ FFs, which is very lightweight. The IO could be minimized even smaller to 6, which is left for our future work. This design could be better optimized based on the specific fingerprint data we use. The minimum clock period we use is $cp = 3.814ns$.

TABLE 8: Power Utilization of Fingerprint Indexing

Resource	Power Consumption
Total Power	0.139W
Dynamic Power	0.082W
Static Power	0.057W
Clock Power	0.019W

The average runtime of operation is $tM = 1.799ms$ for each fingerprint, which is acceptable because this application is only run intermittently. Once the MCC module gets all the data, the SLSH module is activated to process the data. The average time to hash the data is $tH = 0.00036ms$. Therefore the total time to get the SLSH hash value from the fingerprint data is $tM + tH = 1.79936ms$. SLSH operation only adds 0.02% of the MCC processing time and provides secure data. Table 8 shows that the total power for consumption the MCC module is still very low at $0.139W$ when assuming the worst case in which the MCC module is active at all times.

The plain-text version of this application achieves approximate fingerprint indexing using LSH, instead of the privacy-preserving SLSH scheme that we utilize in this application. As we mention before, SLSH is simply a transformation that can be done on LSH functions. We perform our application with and without SLSH to compute the overhead of SenseHash. Due to our optimizations, the SLSH module only requires 18 more LUTs, 7 more FFs and 32 more IOs. This, paired with a negligible difference in average runtime, means that SenseHash is a low-overhead and efficient solution for this application.

7 Limitations and Future Work

Our work is focused on protecting sensor data from data breaches on network-connected gateways and honest-but-curious servers, that will follow the protocol but try to deduce information about the data. For this reason, we assume that the clients utilize trusted hardware during operation. While secure in this setting, if the clients do not utilize trusted hardware, the data is vulnerable to side-channel attacks. As mentioned before, LSH is susceptible to the triangulation problem, in which an adversary can estimate the input to the LSH by performing multiple queries and observing the number of collisions. In our RLSH scheme, we increment indices in our RACE sketch corresponding to LSH function outputs. If the attacker is aware of the memory layout of the processing unit, they can see what indices are being accessed and perform a triangulation attack to get an approximation of the user's original value [36], [37], [38]. Our work also does not support the use of passive sensors, due to the nature of our data collection and aggregation [39]. However, we fully support active sensors that provide reading within a known range, which applies to a majority of sensors.

In our future works, we hope to present an expansion of applications for SenseHash, such as facial recognition via SLSH. Many of the newer applications we aim to implement will require the development of new encoding functions and LSH families, similar to how we utilize MCC encodings and an LSH family to preserve MCC similarity in our fingerprint indexing application. Developing these encodings and families is an exciting path that will enable many more applications that would greatly benefit from the secure operation.

8 Conclusion

This paper proposes SenseHash, the realization of security at the sensor for various applications. SenseHash utilizes state-of-the-art Secure Locality Sensitive Hashing and presents the RLSH scheme, which utilizes lightweight hashing mechanisms to achieve differential privacy, ensuring the safety of user data. Our applications ensure data privacy with a compromised server and gateway by providing security on an FPGA before the data is transferred over a network. All following operations are done with the encrypted data, which provides an impenetrable form of privacy. SenseHash addresses the privacy concerns surrounding monitoring health vitals, alongside privacy concerns with fingerprint indexing. Alongside this, SenseHash is a generalizable solution to many problems that require secure computation and handling of data. Our analysis shows that SenseHash is a fully realizable solution to handling sensitive data and is not limited by utilization of resources, computation time, or computation energy.

Acknowledgments

This work was supported in part by National Science Foundation (NSF) Trust-Hub under award number CNS-2016737, and NSF TILOS under award number CCF-2112665. This work was also supported by the Intel PrivateAI Collaborative Research Institute.

References

[1] M. Samragh, S. Hussain, X. Zhang, K. Huang, and F. Koushanfar, "On the application of binary neural networks in oblivious inference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4630–4639.

[2] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–35, 2018.

[3] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends® in Privacy and Security*, vol. 2, no. 2-3, 2017.

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, ser. SCG '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 253–262. [Online]. Available: https://doi.org/10.1145/997817.997857

[5] B. Coleman and A. Shrivastava, "A one-pass private sketch for most machine learning tasks," 2020.

[6] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1253–1269.

[7] M. S. Riazi, B. Chen, A. Shrivastava, D. Wallach, and F. Koushanfar, "Sub-linear privacy-preserving search with untrusted server and semi-honest parties," 2016.

[8] S. Zahur and D. Evans, "Obliv-C: A language for extensible data-oblivious computation." *IACR Cryptology ePrint Archive*, vol. 2015, p. 1153, 2015.

[9] D. Demmler, T. Schneider, and M. Zohner, "ABY-a framework for efficient mixed-protocol secure two-party computation." in *NDSS*. The Internet Society, 2015.

[10] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library-seal v2. 1," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 3–18.

[11] C. Dwork, "The differential privacy frontier," in *Theory of Cryptography Conference*. Springer, 2009, pp. 496–502.

[12] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar, "Privacy, accuracy, and consistency too: a holistic solution to contingency table release," in *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2007, pp. 273–282.

[13] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: the sulq framework," in *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2005, pp. 128–138.

[14] A. Blum, K. Ligett, and A. Roth, "A learning theory approach to noninteractive database privacy," *Journal of the ACM (JACM)*, vol. 60, no. 2, pp. 1–25, 2013.

[15] S. Raskhodnikova, A. Smith, H. K. Lee, K. Nissim, and S. P. Kasiviswanathan, "What can we learn privately," in *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*, 2008, pp. 531–540.
[16] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.
[17] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 2007, pp. 94–103.
[18] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604–613.
[19] Y. Liu, J. Cui, Z. Huang, H. Li, and H. T. Shen, "Sk-lsh: an efficient index structure for approximate nearest neighbor search," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 745–756, 2014.
[20] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.
[21] R. Cappelli, M. Ferrara, and D. Maltoni, "Minutia cylinder-code: A new representation and matching technique for fingerprint recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 12, pp. 2128–2141, 2010.
[22] I. Formats, "Information technology—biometric data interchange formats—part 2: Finger minutiae data," *Norma ISO/IEC 19794-2: 2011*, vol. 60, 2011.
[23] M. S. Riazi, M. Samragh, and F. Koushanfar, "Camsure: Secure content-addressable memory for approximate search," *ACM Transactions on Embedded Computing Systems*, 2017.
[24] V. Gadepally, M. Isakov, R. Agrawal, J. Kepner, K. Gettings, and M. A. Kinsy, "Homomorphic encryption based secure sensor data processing," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 1–7.
[25] Y. Su, B. Yang, C. Yang, and L. Tian, "Fpga-based hardware accelerator for leveled ring-lwe fully homomorphic encryption," *IEEE Access*, vol. 8, pp. 168 008–168 025, 2020.
[26] D. Kales, S. Ramacher, C. Rechberger, R. Walch, and M. Werner, "Efficient fpga implementations of lowmc and picnic," Cryptology ePrint Archive, Paper 2019/1368, 2019, https://eprint.iacr.org/2019/1368. [Online]. Available: https://eprint.iacr.org/2019/1368
[27] K. Vu, R. Zheng, and J. Gao, "Efficient algorithms for k-anonymous location privacy in participatory sensing," in *INFOCOM*. IEEE, 2012.
[28] W. Zhou, J. Hu, and S. Wang, "Enhanced locality-sensitive hashing for fingerprint forensics over large multi-sensor databases," *IEEE Transactions on Big Data*, 2017.
[29] Y. Zhang and F. Koushanfar, "Robust privacy-preserving fingerprint authentication," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2016, pp. 1–6.
[30] D. Valencia, S. F. Fard, and A. Alimohammad, "An artificial neural network processor with a custom instruction set architecture for embedded applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020.
[31] R. Cappelli, M. Ferrara, and D. Maltoni, "Fingerprint indexing based on minutia cylinder-code," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 1051–1057, 2010.
[32] S. H. Lee, M. K. Jeon, and M. H. Jung, "Fingerprint sensor, display device including the same, and method of operating fingerprint sensor," Jan. 31 2019, uS Patent App. 16/000,259.
[33] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.
[34] M. Ferrara, D. Maltoni, and R. Cappelli, "Noninvertible minutia cylinder-code representation," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1727–1737, 2012.
[35] ——, "A two-factor protection scheme for mcc fingerprint templates," in *2014 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE, 2014, pp. 1–8.
[36] Y. Yarom and K. Falkner, "FLUSH+RELOAD: A high resolution, low noise, l3 cache Side-Channel attack," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 719–732. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom
[37] D. Page, "Theoretical use of cache memory as a cryptanalytic side-channel," Cryptology ePrint Archive, Paper 2002/169, 2002, https://eprint.iacr.org/2002/169. [Online]. Available: https://eprint.iacr.org/2002/169
[38] H. Chabanne, J.-L. Danger, L. Guiga, and U. Kühne, "Side channel attacks for architecture extraction of neural networks," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 3–16, 2021.
[39] L. M. Reindl, "Wireless passive sensors: Basic principles and performances," in *SENSORS, 2008 IEEE*, 2008, pp. 1607–1610.

Nojan Sheybani received the B.S. degree in computer engineering from the University of Virginia, Charlottesville, VA, USA. He is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of California at San Diego, La Jolla, CA, USA. His current research interests include privacy preserving computation, hardware/software codesign, zero knowledge proofs, and quantized neural networks.

Xinqiao Zhang (S'20) received the B.S. degree in automation from Northeastern University, Shenyang, China. He received the M.S. degree in electrical engineering from San Diego State University, San Diego, CA, USA. He is currently pursuing the Ph.D. degree jointly with the University of California, San Diego, La Jolla, CA, USA, and San Diego State University, San Diego, CA, USA. His current research interests include fault-/trojan detection and privacy-preserving computation.

Siam Umar Hussain is a PhD candidate at University of California at San Diego, La Jolla, CA, USA. His research focus is enabling data-intensive systems with provable privacy guarantee in practical settings.

Farinaz Koushanfar is currently a Professor and a Henry Booker Faculty Scholar of Electrical and Computer Engineering with the University of California at San Diego, La Jolla, CA, USA. Her research interests include embedded and cyber–physical systems design, embedded systems security, and design automation of domain-specific/mobile computing. Koushanfar has a PhD in electrical engineering and computer science from the University of California Berkeley, Berkeley, CA, USA. She is a Fellow of IEEE.