FAPRIL: Towards Faster Privacy-Preserving Fingerprint-Based Localization (Full Version)*

Christopher van der Beets[1], Raine Nieminen[2(✉)], and Thomas Schneider[2]

[1] Darmstadt University of Applied Sciences, Germany
christopher.vanderbeets@stud.h-da.de
[2] ENCRYPTO, Technical University of Darmstadt, Germany
{nieminen,schneider}@encrypto.cs.tu-darmstadt.de

Abstract. Fingerprinting is a commonly used technique to provide accurate localization for indoor areas, where global navigation satellite systems, such as GPS and Galileo, cannot function or are not precise enough. Although fingerprint-based indoor localization has gained wide popularity, existing solutions that preserve privacy either rely on non-colluding servers or have high communication which hinder deployment.

In this work we present FAPRIL, a privacy-preserving indoor localization scheme, which takes advantage of the latest secure two-party computation protocol improvements. We can split our scheme into two parts: an input independent setup phase and an online phase. We concentrate on optimizing the online phase for mobile clients who run on a mobile data plan and observe that recurring operands allow to optimize the total communication overhead even further. Our observation can be generalized, e.g., to improve multiplication of Arithmetic secret shared matrices. We implement FAPRIL on mobile devices and our benchmarks over a simulated LTE network show that the online phase of a private localization takes under 0.15 seconds with less than 0.20 megabytes of communication even for large buildings. The setup phase, which can be pre-computed, depends heavily on the setting but stays in the range $0.28 - 4.14$ seconds and $0.69 - 16.00$ megabytes per localization query. The round complexity of FAPRIL is constant for both phases.

Keywords: Fingerprint-Based Localization, Indoor Localization, Location Privacy, Data Privacy, Secure Two-Party Computation, Smartphones.

1 INTRODUCTION

Localization is conventionally based on Global Navigation Satellite Systems (GNSSs), such as GPS or Galileo. However, the downside of GNSSs is their decreased accuracy when the satellite signals are blocked by obstacles. This is particularly a problem for indoor areas, such as shopping malls or airports. Hence new techniques have been developed and deployed specifically for Indoor Localization (IL), see [LTP17; Yas+17] for surveys. The IL techniques are commonly based on using Wi-Fi [Lad+05; Tao+03; Hae+04], cellular [TL13], RFID [Cha+13], Bluetooth [Che+11], or Zigbee [NLY08] signals. Localization using Wi-Fi signals is particularly tempting for public buildings having a Wi-Fi Access Point (AP) infrastructure already in place. Besides this, the increase in the number of smartphones and other mobile devices capable of measuring Received Signal Strengths (RSSs) from Wi-Fi APs makes it easy for people to use these techniques without having to buy an extra device. In the future, even cellular signals could become interesting source of RSSs via dense 5G networks [Hak+15], providing accurate localization for both indoors and outdoors.

Relatively cheap and accurate IL has gained interest from different Service Providers (SPs) resulting in an increase in the number of Location-Based Services (LBSs) developed for indoor areas such as museums, shopping malls, airports, exhibition centers, hospitals, and university campuses (see, e.g., [Gua+17; HLC17]). The most interesting services include indoor navigation, routing to a given destination, finding nearby friends, and targeted advertising (see, e.g., [KPV12; All+15; Li+17]). The recent study in [Bar+21] also discusses the

* Please cite the proceedings version of this paper published at SECRYPT'22 [BNS22].

potentials of using different IL technologies for measuring the distance among the users and enhancing social distancing effectively in indoor environments.

However, location information is highly privacy sensitive and could be exploited to predict persons' future movements and even identify them [Bel+13b]. In addition, SPs might have to comply with privacy laws and regulations, such as the EU General Data Protection Regulation (GDPR), forcing them to use privacy-preserving solutions. The privacy aspects of LBSs are well studied (see, e.g., [HOS15; Sti+17; HOS17; Jär+18]). However, the studies mainly focus on how the location information is handled in the service itself after the client has already performed the actual localization. In our paper, the focus is on the privacy concerns of the actual location retrieval, which is a natural prerequisite for privacy-preserving LBSs.

Whereas GNSS based localization is inherently privacy-preserving (see [Che+17; Loh+17a]), for IL techniques privacy is harder to achieve. In our paper, we focus on fingerprint-based localization, which is the most prominent IL technique used with Wi-Fi APs (see, e.g., [Cap+11; Gua+17; Yin+17]). Fingerprint-based localization relies on a pre-constructed database, which contains so called fingerprints. Each fingerprint is basically a vector, where each element is a RSS value from a different source such as a Wi-Fi AP. The fingerprints are pre-measured around the building or area at various known locations called reference points. In the location retrieval step, the user measures its own fingerprint, which is then compared to the fingerprints in the database. In a typical scenario, the Service Provider (SP) holds the database on a server that compares with the user's fingerprint. However, this leaks the user's location to the server, and thus to the SP.

A naïve solution against leaking the user's location would be to send the whole database to the user's device. This maintains privacy for the user, since now the location retrieval includes only local computations. However, it violates the privacy of the server and SP by leaking the database content to the users, and thus releasing the database practically to the public. The reasons why the database should be kept secret from the public are already discussed in several related works (see §1.1). The database has high value, since constructing it is laborious manual work. In practice, SPs still tend to have the power and control over the users on what information is leaked. Since the database is basically the only valuable asset for the SP in fingerprint-based localization and might even leak delicate details of the building construction (e.g., suspiciously thick or thin walls), we expect that the SPs tend to sacrifice users' privacy unless better solutions are available.

Privacy-Preserving Indoor Localization (PPIL) tries to solve the privacy issues regarding both the users' location information and the database. However, the previously proposed PPIL schemes suffer from drawbacks such as low performance and high data transfer rates, which discourage an actual deployment.

1.1 Related Work

One of the earliest fingerprint-based PPIL scheme proposals in the literature for the client-server setting without using trusted third parties was PriWFL [Li+14] by Li et al. presented in INFOCOM'14. The scheme uses Paillier encryption [Pai99] to preserve clients' privacy but tries to hide the database information with an ad hoc masking technique, which was found later insecure by Yang and Järvinen [YJ18].

Shu et al. [Shu+14] proposed a PPIL scheme using Paillier encryption and Oblivious Transfer (OT). The scheme relies on other mobile devices with known location, which is a major drawback in practice. Additionally, computationally heavy operations are performed on the client's mobile device, which leads to performance and battery-power problems.

Ziegeldorf et al. [Zie+14] based their PPIL scheme on the Hidden Markov Model and use Homomorphic Encryption (HE) for the computations. The performance of their solution is poor, since the location retrieval takes around 10 seconds even in a relatively small building when using powerful servers and a fast LAN network.

Konstantidis et al. [Kon+16] proposed a PPIL scheme based on k-anonymity. The downside of their scheme is that the privacy guarantees become weaker if auxiliary information, such as the building map, is provided to the adversary. On the other hand, if the security parameter sizes are increased, the location accuracy decreases and the scheme becomes impractical.

Järvinen et al. proposed PILOT [Jär+19], which was the first efficient solution for PPIL. PILOT uses well known Secure Two-Party Computation (STPC) techniques and outsources the most involving computations to two semi-trusted servers in order to achieve practical performance. Their work includes benchmarks with various combinations of STPC techniques. The outsourcing setting assumes that the two servers are

non-colluding, which are hard to find (e.g., the owner of the building and the mobile network operator). Their underlying protocols are suitable also for the client-server setting, and so we consider [Jär+19] as an excellent basis for developing more efficient PPIL schemes. The main disadvantage of PILOT is the large communication overhead, which is regularly over 100 megabytes per localization query making it unsuitable for setups where parties have limited data plans. In our work, we present techniques, which reduce the total communication by factor 16× with a very efficient online phase.

Nieminen and Järvinen [NJ20] proposed a PPIL scheme in the client-server setting using Paillier encryption and Garbled Circuits (GCs). We call their solution NJ in our paper. NJ does not rely on trusted third parties or outsourcing, but involves relatively expensive operations leading to online times of $1.55 - 8.46$ seconds. On top of this, their scheme requires computationally heavy pre-computations in the setup phase, which take $2.45 - 12.23$ seconds for each localization query. Compared to PILOT [Jär+19], NJ achieves better total communication overheads making it suitable for the mobile client setting. In contrast to NJ [NJ20], we achieve online times of only 0.15 seconds and need only $0.28 - 4.14$ seconds for the setup phase without increasing the communication overhead.

1.2 Our Contributions

We provide the following contributions in our paper:

- We design and implement FAPRIL, a fingerprint-based client-server Privacy-Preserving Indoor Localization (PPIL) scheme using STPC techniques that enables fast localization for users without leaking the actual location to third parties. Additionally, FAPRIL provides privacy to SPs keeping their database private. We show that FAPRIL achieves practical run-times even in large settings, and when the client is using a regular smartphone. Compared to today's best known client-server PPIL scheme [NJ20], our protocol avoids expensive additively homomorphic encryption and has a 8× faster setup time, 27× faster online time, 6× less online communication, and only 1.11× more setup communication (cf. Tab. 3). Compared to PILOT [Jär+19], we achieve 16× less communication in the setup phase, and 20× less communication in the online phase.
- Our main goal is to optimize the input-dependent online phase which is run over a mobile dataplan in PPIL. For this, we use the state-of-the-art secure two-party computation protocols from ABY2.0 [Pat+21] that allow scalar products with online communication independent of the dimension of the vector. This yields an order of magnitude better online communication than the ABY [DSZ15]-based PPIL protocols of PILOT [Jär+19] (see §5.3). ABY2.0 focused on improving the online phase for different functionalities like scalar product or matrix multiplication, but did not consider optimized methods to compute the corresponding correlated randomness in the setup phase. We observe that in our particular setting of computing squared Euclidean distances we *multiply the same* ABY2.0 shared vector with different vectors and propose a dedicated setup protocol to efficiently compute special multiplication triples for this based on observations from [MRT20]. This in return yields an order of magnitude better setup communication than PILOT, which ultimately makes FAPRIL suitable for the mobile client setting. The protocol uses a technique from [MRT20] which allows to efficiently multiply one plain value with a vector of plain values using Correlated Oblivious Transfer (C-OT) [Gil99; Ash+13; DSZ15]. This can be generalized to improve the setup complexity of general Arithmetic circuits where the same value is used in several multiplications, e.g., as in matrix multiplication which is a common operation in Privacy-Preserving Machine Learning (PPML).
- The design of FAPRIL divides the fingerprint-based localization protocol into two independent operations, namely vector-matrix multiplication with secret shares[3] [Pat+21] and k-Nearest Neighbor Algorithm (kNN)[4] [Son+15; Jär+19], which also have applications in PPML and hence are likely to be optimized further in the future. Therefore, advances in PPML can translate to advances in PPIL.

[3] We use an Arithmetic sharing based approach, since it outperforms solutions relying on public key cryptography such as [WAP17; Mis+21].

[4] We use a Garbled Circuit (GC) based approach to preserve accuracy. Approximate and non-constant round k-Nearest Neighbor Algorithms (kNNs) have been studied, e.g., in [SFR20; Che+20; ZS21; MRT20].

– Our final contribution is an implementation of the most efficient PPIL scheme based fully on Yao's Garbled Circuits (GCs) from [Jär+19] for the client-server setting. We note that the scheme is not directly comparable with FAPRIL, since it cannot achieve the same level of localization accuracy. Nevertheless, we show that the scheme is practical and serves as an alternative solution to FAPRIL for environments, where accuracy requirements can be relaxed.

1.3 Structure

The rest of the paper is structured as follows. §2 covers the relevant background on indoor localization and STPC protocols. We describe FAPRIL in §3, including a theoretical evaluation in §3.2 and a security analysis in §3.3. In §4, we cover the essential optimization technique, which allows FAPRIL to achieve practical communication overheads and outperform previous state-of-the-art schemes. Benchmarks and comparisons to the related works are covered in §5. Finally, we draw conclusions and list future works in §6.

2 PRELIMINARIES

In this section, we describe the general fingerprint-based localization technique commonly used for Indoor Localization (IL) with different distance metrics. Thereafter, we summarize the relevant Secure Two-Party Computation (STPC) techniques used in FAPRIL.

2.1 Fingerprint-Based Localization

Indoor areas, such as airports and shopping malls, need non-Global Navigation Satellite System (GNSS) based techniques for accurate localization, since ceilings or other obstacles often block the satellite signals. A common solution for Indoor Localization (IL) is fingerprint-based localization (see, e.g., [BP00; Liu+07]), which requires the Service Provider (SP) to pre-construct a database holding a set of pre-measured fingerprints. The fingerprints contain Received Signal Strengths (RSSs) measured from different Access Points (APs) (e.g., Wi-Fi). The set of used APs is fixed and public for the localization setup and can be based on, e.g., MAC addresses. We assume to have N **APs** and the pre-measured fingerprints are measured from M **reference points** around the area or building. The database is written as $\mathfrak{D} = \{V_1, \ldots, V_M\}$, where V_i denotes the pre-measured fingerprint from the i-th reference point (denoted as χ_i). A pre-measured fingerprint is written as $V_i = \{v_{i,1}, \ldots, v_{i,N}\}$, where $v_{i,j}$ denotes the RSS at χ_i from the j-th AP. After the construction, $\mathfrak{D}$ is normally placed on a server $\mathcal{S}$ offering the localization service to users, a.k.a. clients.

Location retrieval of a client $\mathcal{C}$ goes as follows: $\mathcal{C}$ measures its own fingerprint $F = \{f_1, \ldots, f_N\}$ and compares F to each V_i, $i \in \{1, \ldots, M\}$ using a specific distance metric (see §2.1). A small distance d_i between F and V_i means that the fingerprints are similar implying that $\mathcal{C}$ is likely to be near χ_i. For many settings, it is beneficial (for better accuracy) to take the average location of multiple reference points χ_i corresponding to V_i for which the distance to F is the smallest. These points can be determined by the k-Nearest Neighbor Algorithm (kNN), which here outputs a set of nearest reference points $\{\chi_{1'}, \ldots, \chi_{k'}\}$. The final location output to $\mathcal{C}$ is normally the centroid of the k reference points $\{\chi_{1'}, \ldots, \chi_{k'}\}$.

Distance Metrics In fingerprint-based localization, the server $\mathcal{S}$ computes distances d_i between F and V_i for $i \in \{1, \ldots, M\}$. Different distance metrics for fingerprint-based IL have been studied before (see, e.g., [Ric+18; Jär+19]). Here we present two of them.

The Manhattan distance is a distance metric, which takes absolute values on the differences between $\mathcal{C}$'s f_j and $\mathcal{S}$'s $v_{i,j}$ for $j \in \{1, \ldots, N\}$ and sums them up as shown in Eq. (1).

$$d_i^M = \sum_{j=1}^{N} |f_j - v_{i,j}| \tag{1}$$

The Euclidean distance is the diagonal distance between f_j and $v_{i,j}$. To optimize performance, we focus on the squared Euclidean distance calculated as shown in Eq. (2). We note that for fingerprint-based localization this is equivalent to the regular Euclidean distance, since squaring is an order preserving operation.

$$d_i^E = \sum_{j=1}^{N} (f_j - v_{i,j})^2 \tag{2}$$

Parameter Size Ranges Fingerprint-based localization includes several parameters, which can impact the localization accuracy. Here we give a brief overview of the most common size ranges for these parameters.

The size of f_j and $v_{i,j}$ follows directly from the RSS, which is normally measured in decibels. Richter et al. [Ric+18] showed that a 4-bit quantization of RSS yields the same positioning accuracy as with unquantized RSS. They also discovered that even 1-bit quantization is feasible for certain applications. Hence, we conclude that f_j and $v_{i,j}$ are typically $1-4$ bits long.

The best choice for k in kNN depends on the building architecture. However, typical values for k are 3 or 4 [BP00; Li+05].

The parameters N and M determine the size of the database $\mathfrak{D}$, which follows from the localization setup and cannot be fixed for general analysis. Several databases for Wi-Fi fingerprint-based localization have been constructed from real buildings (see, e.g., [Men+18; Loh+17b]). In our experiments, we use $N \in [50, 250]$ and $M \in [100, 800]$ in order to cover a large variety of different possible setups. We consider $N = 241$ and $M = 505$ as a special case for comparison reasons with [Jär+19]. The values are based on a real database measured from a four-story building [YJ18]. It might be tempting to conclude that typically $M \sim 2N$, but while this might make sense for many settings, we emphasize that very different setups could occur in the real world. As a final point, we note that a general observation was made in [YJ18] regarding the database: 85.4% of all RSSs in the database are zero, i.e., most of the APs are out of reach from a single reference point. While this might be interesting to optimize certain schemes that rely on public key cryptography such as NJ [NJ20], the run-times of our protocols are independent of the input values.

2.2 Secure Two-Party Computation

A Secure Two-Party Computation (STPC) protocol enables two parties to compute a public function $f(x, y)$ on their respective private inputs x and y without revealing any other information except the output of the function. In this section, we cover two well-known STPC protocols, namely Arithmetic sharing and Yao sharing. Finally, we describe how to securely convert from Arithmetic sharing to Yao sharing.

Arithmetic and Delta Sharing The Arithmetic sharing protocol, due to GMW [GMW87], enables two parties P_0 and P_1 to evaluate a function on secret shared values. The function is expressed as an Arithmetic circuit consisting of addition and multiplication gates and the operations are performed in the ring $\mathbb{Z}_{2^\ell}$. We denote the *Arithmetic sharing* of x with $[x]$ and the random shares with $[x]_0, [x]_1$ respectively for P_0, P_1. In order for P_i, $i \in \{0, 1\}$ to secret share $x \in \mathbb{Z}_{2^\ell}$, P_i chooses a random value $r \in_R \mathbb{Z}_{2^\ell}$ and sets $[x]_i = x - r \pmod{2^\ell}$ and $[x]_{1-i} = r$. Now P_i keeps $[x]_i$ and sends $[x]_{1-i}$ to P_{1-i}. It is easy to see that $[x]_0 + [x]_1 = x \pmod{2^\ell}$.

Patra et al. [Pat+21] proposed an optimized variant of the general Arithmetic sharing protocol, which we refer to as *Delta sharing* protocol from now on. With their technique, the online communication per multiplication gate is reduced in half. More importantly for us, they present an efficient protocol for scalar product, which achieves an online communication complexity *independent* of the vector dimension. We denote the Delta sharing of x with $\langle x \rangle$ and the random shares with $\langle x \rangle_i$ for P_i, $i \in \{0, 1\}$. Addition gates are local computations. Multiplication requires an interactive protocol and Multiplication Triples (MTs). In our work, we generate the MTs via the Correlated Oblivious Transfer (C-OT) protocol of [Ash+13]. We refer the reader to [Pat+21] for the details on the Delta sharing semantics.

Yao Sharing Yao sharing is based on Yao's Garbled Circuits (GCs) [Yao86] and enables two parties to securely evaluate a function expressed as a Boolean circuit. The basic idea of GCs is the following: one of the parties, called the garbler, assigns two randomly chosen symmetric keys to all wires called garbled values. For all gates, the garbler uses the input keys to encrypt the corresponding output key. Next, the garbler sends the encrypted gates (called garbled circuit) to the other party, typically referred to as the evaluator, along with the symmetric keys corresponding to its input bits. The evaluator obtains the symmetric keys corresponding to its input bits via an Oblivious Transfer (OT) protocol implemented efficiently using OT extension [Ish+03; Ash+13]. Finally, the evaluator evaluates the garbled circuit by decrypting the garbled gates obtaining the garbled output values. In order to reveal the actual output values to the evaluator, the garbler provides the evaluator with information to decode the output wires.

Over the years, several optimization techniques for GCs have been introduced. These include point-and-permute [BMR90], free-XOR [KS08], fixed-key AES garbling [Bel+13a], and half-gates [ZRE15]. Recently, Rosulek and Roy [RR21] introduced a new GC optimization, which defeats half-gates [ZRE15]. However, we exclude this technique from our implementation of FAPRIL, mainly in order to have a fair comparison with the related works, which can also benefit from this optimization.

From Arithmetic/Delta to Yao Sharing The conversion from Arithmetic shares to Yao shares is described in [DSZ15]. The conversion for the Delta sharing protocol is described in [Pat+21], but follows the same idea: the Boolean circuit for Yao's GCs begins by adding the Arithmetic shared values from both parties. With Delta sharing the C-OT step can be performed in the setup phase in contrast to Arithmetic sharing, where this is in the online phase.

3 SYSTEM DETAILS

In this section, we give an overview of FAPRIL, our PPIL scheme in the client-server setting based on mixed-protocol Secure Two-Party Computation (STPC), namely Delta and Yao sharing (see §2.2). A high-level overview is given in §3.1 and a more detailed description follows in §3.2 containing also the complexity analysis. Finally, the security aspects are discussed in §3.3.

3.1 Overview

Here we give a high-level overview of FAPRIL, which uses Delta and Yao sharing to provide privacy-preserving fingerprint-based localization. Our scheme follows the idea from [Jär+19], where the most cost efficient localization was achieved using the combination of Arithmetic sharing, which we improve to Delta sharing, and Yao sharing. Additionally, we select the squared Euclidean distance to be our distance metric (see §2.1), since it gives arguably the best cost-accuracy ratio for Privacy-Preserving Indoor Localization (PPIL), see [Jär+19]. We denote the number of used Access Points (APs) with N and the number of reference points with M (see §2.1).

After the client $\mathcal{C}$ has measured its fingerprint F, it Delta shares F with the server $\mathcal{S}$ such that $\langle F \rangle = \{\langle f_1 \rangle, \ldots, \langle f_N \rangle\}$. Additionally, $\mathcal{C}$ Delta shares $\sum_{j=1}^{N} f_j^2$ with $\mathcal{S}$. Similarly, $\mathcal{S}$ Delta shares each pre-measured fingerprint V_i along with $\sum_{j=1}^{N} v_{i,j}^2$, $i \in \{1, \ldots, M\}$ from the database $\mathfrak{D}$. After the sharing, both parties have obtained their respective shares $\langle F \rangle$, $\left\langle \sum_{j=1}^{N} f_j^2 \right\rangle$ and $\left\{ \langle V_i \rangle, \left\langle \sum_{j=1}^{N} v_{i,j}^2 \right\rangle \right\}_{i=1}^{M}$, where $\langle V_i \rangle = \{\langle v_{i,1} \rangle, \ldots, \langle v_{i,N} \rangle\}$.

Next, both parties compute the squared Euclidean distances using their respective shares using Eq. (3), which is directly derived from Eq. (2).

$$\langle d_i \rangle = \left\langle \sum_{j=1}^{N} f_j^2 \right\rangle + \left\langle \sum_{j=1}^{N} v_{i,j}^2 \right\rangle - 2 \sum_{j=1}^{N} \langle f_j \rangle \cdot \langle v_{i,j} \rangle \tag{3}$$

Eq. (3) requires N multiplications, and thus the total number of multiplications to obtain all $\{\langle d_i \rangle\}_{i=1}^{M}$ is $N \cdot M$.

We note that $\sum_{j=1}^{N} \langle f_j \rangle \cdot \langle v_{i,j} \rangle$ is a scalar product between $\langle F \rangle$ and $\langle V_i \rangle$, and thus we can compute it with the optimized Delta sharing scalar product of [Pat+21]. More generally, we can reduce the problem of computing $\{\langle d_i \rangle\}_{i=1}^{M}$ down to a vector-matrix multiplication (plus cheap addition operations with the shares afterwards). More formally, we transform the fingerprint F to a vector of length N and the database $\mathfrak{D}$ into a $M \times N$ matrix. The Delta shared versions are as follows:

$$\left\langle F^N \right\rangle = \left[\langle f_1 \rangle \ \langle f_2 \rangle \cdots \langle f_N \rangle \right] \quad ,$$

$$\left\langle \mathfrak{D}^{M \times N} \right\rangle = \begin{bmatrix} \langle v_{1,1} \rangle & \langle v_{1,2} \rangle & \cdots & \langle v_{1,N} \rangle \\ \langle v_{2,1} \rangle & \langle v_{2,2} \rangle & \cdots & \langle v_{2,N} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle v_{M,1} \rangle & \langle v_{M,2} \rangle & \cdots & \langle v_{M,N} \rangle \end{bmatrix} \quad .$$

The distances are obtained by first computing the vector-matrix multiplication $\left\langle F^N \right\rangle \cdot \left\langle \mathfrak{D}^{M \times N} \right\rangle^{\mathsf{T}}$, where T is the transpose operator[5]. The result is a vector of length M containing $\left\{ \sum_{j=1}^{N} \langle f_j \rangle \cdot \langle v_{i,j} \rangle \right\}_{i=1}^{M}$. These values are used with Eq. (3) requiring a total of 3 additions. Note that all the operations are in $\mathbb{Z}_{2^\ell}$, where ℓ is the bit-length of d.

The Delta shares $\{\langle d_i \rangle\}_{i=1}^{M}$ are converted to Yao shares as described in §2.2. In FAPRIL, the server $\mathcal{S}$ takes the role of the garbler and constructs a garbled circuit, which first adds the shares $\{\langle d_i \rangle\}_{i=1}^{M}$ together (the conversion) and then runs the k-Nearest Neighbor Algorithm (kNN) on the distances, finally returning the k indices corresponding to the smallest distances. An efficient circuit construction of kNN for Yao sharing was presented in the appendix of [Jär+19] and we omit further details here.

In the last step of FAPRIL, the client $\mathcal{C}$ uses the k indices to obtain the k reference points $\{\chi_{1'}, \ldots, \chi_{k'}\}$. Here we assume, similarly to [NJ20], that the server $\mathcal{S}$ has published the set of reference points $\{\chi_i\}_{i=1}^{M}$ to the public.

3.2 Details and Complexity

We move on to a more precise description of the steps of FAPRIL and the complexity of these steps. Certain parameters, such as the list of Access Points (APs) and the security parameter κ, are negotiated between the client $\mathcal{C}$ and the server $\mathcal{S}$ in a one-time initialization step. The resulting overhead is not very interesting, since it does not accumulate over time. Moreover, the parameters can be included in the client application as hard coded values.

FAPRIL uses the Oblivious Transfer (OT) extension protocol of [Ash+13], which requires so called base OT in the initialization step. In fact, our implementation of FAPRIL (see §5) uses two different OT flavors and we need to run two different base OT steps. However, the steps are independent of M and N, and require under 1 second and 30 kilobytes of communication in total. They can also be reused for multiple protocol runs between $\mathcal{C}$ and $\mathcal{S}$. We conclude that these overheads are negligible.

We separate FAPRIL into two phases, namely setup and online phase. The setup phase is independent of $\mathcal{C}$'s fingerprint F and can be computed in advance, e.g., multiple times overnight using a relatively fast Wi-Fi connection. The online phase depends on F and must be performed only after $\mathcal{C}$ has measured the Received Signal Strengths (RSSs).

[5] Note that the parties do not actually have to perform the transpose operator, because it only gives a different representation of the database $\mathfrak{D}$. The reason for our initial representation is mainly to be consistent with the literature.

Setup Phase In the setup phase, we generate the Multiplication Triples (MTs) for the multiplication operations with the Delta shares as shown in Eq. (3). MTs can be generated in various ways using OT or HE [RSS19]. Due to the necessity to implement on a mobile device, we choose to use the OT-based multiplication protocol by Gilboa [Gil99] with Correlated Oblivious Transfer (C-OT) as in [DSZ15]. With the optimization technique from [DSZ15], the total communication needed to construct an ℓ-bit MT is $2\ell(\kappa + (\ell + 1)/2)$ bits, where κ is the security parameter. We set $\kappa = 128$ for FAPRIL. The multiplications are performed in $\mathbb{Z}_{2^\ell}$, where ℓ is the bit-length of d_i, $i \in \{1, \ldots, M\}$. We can calculate ℓ with Eq. (4), where ℓ_{RSS} is the bit-length of f and v, i.e., ℓ_{RSS} follows from the quantization of RSSs.

$$\ell = \left\lceil \log_2 \left((2^{\ell_{RSS}} - 1)^2 \cdot N \right) \right\rceil \tag{4}$$

For FAPRIL, we fix $\ell_{RSS} = 4$ as it was shown to provide the same level of accuracy as longer bit-lengths [Ric+18] (see also §2.1). We simplify Eq. (4) accordingly to get Eq. (5).

$$\ell = \lceil \log_2 225N \rceil \approx 8 + \lceil \log_2 N \rceil \tag{5}$$

As stated earlier, the total number of multiplications needed is $N \cdot M$. Hence, we need a total of $N \cdot M \cdot 2\ell(\kappa + (\ell+1)/2)$ bits of communication. If $N = 241$ and $M = 505$, we have $\ell = 16$ and the communication needed for the MT generation is 63.4 megabytes. However, later in §4 we propose an optimization technique to reduce this communication down to 4.2 megabytes making FAPRIL more practical for smartphone clients with limited mobile data plans. Each MT generation with C-OTs requires 6ℓ symmetric cryptographic operations, so $N \cdot M \cdot 6\ell$ in total.

The client $\mathcal{C}$ needs to obtain its garbled input values from the server $\mathcal{S}$ for the conversion step from Delta to Yao sharing. This is normally performed with a C-OT protocol in order to comply with the free-XOR GC optimization technique [KS08]. In general, this step should be in the online phase, since the inputs (the Delta shares of distances) depend on $\mathcal{C}$'s input. However, due to the nature of Delta shares we can move this step in the setup phase (as was already pointed out in [Pat+21]). The required communication for this step is $M \cdot 2\ell\kappa$ (252.5 kilobytes for $N = 241, M = 505$) and the total number of symmetric cryptographic operations is $M \cdot 3\ell$.

The final setup phase step is the construction and sending of the garbled circuit, which computes the addition for the sharing conversion and kNN. The communication and computation overheads follow directly from the AND-size of the circuit. The conversion requires $M \cdot \ell$ AND gates and the size optimized kNN based on [Jär+19] requires $M \cdot k(2\ell + \lceil \log_2 M \rceil)$ AND gates. This gives us the total number of AND gates, namely $\#\text{AND} = M \cdot (2k\ell + k\lceil \log_2 M \rceil + \ell)$. The required communication is $2\kappa \cdot \#\text{AND}$ bits and the garbler needs to perform $4 \cdot \#\text{AND}$ fixed-key AES operations. With $N = 241$, $M = 505$, and $k = 3$ we get $\#\text{AND} = 70195$, which translates to 2.1 megabytes of communication. We note that with the recent optimization of [RR21] this communication can be reduced by factor $\approx 2/1.5\times$ down to 1.6 megabytes.

Finally, we note that the MT generation is independent of the other steps and can be performed in parallel in order to reduce the total run-time. The total theoretical setup communication with state-of-the-art optimizations (including [RR21] and §4) is 6.0 megabytes, when $N = 241, M = 505, k = 3$.

Online Phase In the online phase, the client $\mathcal{C}$ measures its fingerprint F and runs a private localization query with the server $\mathcal{S}$. The first step for $\mathcal{C}$ is to Delta share F and $\sum_{j=1}^{N} f_j^2$ with $\mathcal{S}$ as explained in §3.1. This requires communication of $(N + 1) \cdot \ell$ bits from $\mathcal{C}$ to $\mathcal{S}$ and only simple computations. For $N = 241$, this step requires 0.5 kilobytes of communication.

Then both parties compute the distances $\{\langle d_i \rangle\}_{i=1}^{M}$ (in parallel) following Eq. (3) and using the efficient scalar product from [Pat+21]. As stated in §2.2, the communication is independent of the vector length (here N) and the total communication is $M \cdot 2\ell$ requiring only one round of communication. This step also requires only simple computations when MTs are pre-computed in the setup phase. For $N = 241, M = 505$, this step requires 2.0 kilobytes of communication.

Next, $\mathcal{S}$ sends its garbled input values corresponding to its shares $\{\langle d_i \rangle\}_{i=1}^{M}$ to $\mathcal{C}$. This step does not require any computations and the size of the garbled values is $\kappa \cdot M \cdot \ell$. For $N = 241, M = 505$, this step requires 126.3 kilobytes of communication.

In the last step, $\mathcal{C}$ evaluates the garbled circuit and obtains k indices. The evaluation requires $2 \cdot \#\text{AND}$ fixed-key AES operations with [ZRE15]. This step does not require communication. The permutation bits [BMR90] of the output garbled values can be used to reveal the actual output values. Alternatively, $\mathcal{S}$ can send the signal bits for all output wires, requiring $k \lceil \log_2 M \rceil$ bits of communication. For $M = 505, k = 3$, this step requires 27 bits of communication.

Finally, we conclude that the communication complexity of the online phase is orders of magnitude smaller than that of the setup phase. The total theoretical online communication is 128.8 kilobytes, when $N = 241, M = 505, k = 3$. FAPRIL shifts very efficiently most of the computation and communication in the setup phase, making it very attractive for applications which require fast location retrievals but allow pre-computations, e.g., overnight.

3.3 Security and Privacy Discussion

In this section, we give a brief security analysis of FAPRIL in §3.3. Furthermore, we discuss different privacy aspects regarding the reference points in §3.3.

FAPRIL is secure in the semi-honest adversary model, a.k.a. "honest-but-curious" adversary model, which assumes that both parties follow the protocol specification, but try to learn additional information from the transcript, i.e., from the received messages. Most of the related work on Privacy-Preserving Indoor Localization (PPIL) relies on the same semi-honest adversary model and it is widely used in many other applications including Privacy-Preserving Machine Learning (PPML).

Security Analysis FAPRIL uses well-known Secure Two-Party Computation (STPC) techniques in a rigorous way. Technique wise, FAPRIL can be divided into three parts, namely computation of an Arithmetic circuit in Delta sharing, conversion from Delta to Yao sharing, and computation of a Boolean circuit in Yao sharing (see §2.2). The latter two are based on Yao's Garbled Circuit (GC) protocol, which was proven secure for semi-honest adversaries in [LP09]. For the security proof of the other parts, we refer the reader to the full version of [Pat+21]. In summary, the security follows from the security assumptions of the underlying primitives, which for FAPRIL is Oblivious Transfer (OT), which was proven secure in [Ash+13]. The security of FAPRIL persists for multiple sequential queries. However, auxiliary information, such as how frequently the client makes queries is always "leaked" outside the protocol execution to the Service Provider (SP).

It is also important that the client communicates with the servers over the mobile network but not via Wi-Fi APs, which are controlled by the SP. This would leak the client's location, since the SP can see from which AP the client communicates and can use triangulation to determine the location. However, this is not related to the RSS values used in FAPRIL and is not a leakage of our scheme.

In conclusion, FAPRIL utilizes all the primitives as a black box, from which the security guarantees follow automatically. FAPRIL guarantees passive security, i.e., security against semi-honest adversaries. We note that since our scheme is based on STPC techniques which also have variants to withstand stronger security models, the security guarantees of FAPRIL can be enhanced using standard methods.

Privacy of Reference Points As pointed out at the end of §3.1, FAPRIL assumes that the set of reference points $\{\chi_i\}_{i=1}^{M}$ is made public by the server. While this is the case in related works such as [NJ20; Li+14], in [Jär+19] the reference points were considered "private". More precisely, PILOT [Jär+19] runs a similar protocol as FAPRIL, but extends the garbled circuit for kNN with an Oblivious Array Access (OA) circuit, which returns only the k closest coordinates $\{\chi_{1'}, \ldots, \chi_{k'}\}$. While this seemingly keeps the reference points $\{\chi_i\}_{i=1}^{M}$ private, we note that an adversary could simply walk around the building or area and obtain at least most of the coordinates revealing a (almost complete) subset of $\{\chi_i\}_{i=1}^{M}$.

One attractive way to prevent the previous attempt is to compute the centroid among the k coordinates also inside the garbled circuit. However, this also reveals information about $\{\chi_i\}_{i=1}^{M}$ by triangulation. We omit the detailed description of the possible attacks, but note that it is certainly possible to obtain at least a close replica of $\{\chi_i\}_{i=1}^{M}$ in fingerprint-based localization, even if the client only obtains its location coordinate from the server (as it is already the case in the general non-privacy-preserving protocol described in §2.1).

In many real-world scenarios, the reference point pattern could be easily guessed from the building map (e.g., when it is based on individual rooms). The reference point coordinates basically only reveal the strategy for constructing $\mathfrak{D}$, but does not help the adversary to construct a similar database, since the adversary does not have the same equipment for the Received Signal Strength (RSS) measurements. We also note that an attack which tries to build a replica of $\mathfrak{D}$ *laboriously*, is not an attack against FAPRIL.

To fully hide the used distance metric and all the parameters, one can use Private Function Evaluation (PFE) [Alh+20] which incurs substantial overhead compared to a public distance metric.

4 CORRELATED MULTIPLICATION TRIPLE GENERATION

In this section, we show how to interactively generate Multiplication Triples (MTs) for the special case where one secret shared value is multiplied with several values. More specifically, we want to generate multiplications of the form $([a]_{0,i} + [a]_{1,i}) \cdot ([b]_0 + [b]_1) = ([c]_{0,i} + [c]_{1,i}) \pmod{2^\ell}$ for $i \in \{1, \ldots, M\}$. As sub-protocol we use a protocol from [MRT20; DSZ15] to multiply a value known in the clear by one party with a vector of values known in the clear by the other party using 1-out-of-2 Correlated Oblivious Transfers (C-OTs). The authors of [MRT20; DSZ15] also considered computing squared Euclidean distances where the same value is reused, but using the old ABY [DSZ15]-style Arithmetic sharing which results for $N = 241$ APs and $M = 505$ reference points in online communication of 3.7 megabytes whereas we require only 2.5 kilobytes.

The authors of [MRT20; DSZ15] also propose an optimization that uses 1-out-of-n Oblivious Transfers (OTs) from [Des+17], but this requires more online communication than using 1-out-of-n C-OT. They also benefit from the fact that client and server know the values in the clear and we leave the combination of such plaintext inputs with the ABY2.0-style protocols as future work.

4.1 C-OT-based MT generation

We start by giving a short overview of how C-OT [Ash+13] is used to generate a single MT as described in [Gil99; DSZ15]. We want to compute a MT consisting of Arithmetic shares $[a]_0, [a]_1, [b]_0, [b]_1, [c]_0, [c]_1 \in \mathbb{Z}_{2^\ell}, \ell \in \mathbb{Z}$ such that $([a]_0 + [a]_1) \cdot ([b]_0 + [b]_1) = [c]_0 + [c]_1 \pmod{2^\ell}$ where party P_i gets to know only its respective shares $[a]_i, [b]_i, [c]_i$. Multiplying out one sees that the challenge is to securely compute $[[a]_0[b]_1]_i$ where the shares are held by different parties. (By symmetry, the other cross-term $[[a]_1[b]_0]_i$ can be computed by running the same protocol in parallel in the opposite direction.)

The protocol starts by P_0 randomly generating $[a]_0 \in_R \mathbb{Z}_{2^\ell}$ and P_1 randomly generating $[b]_1 \in_R \mathbb{Z}_{2^\ell}$. Now P_0 and P_1 run several C-OT protocols in parallel. Without loss of generality, we assume that P_0 is the sender and P_1 is the receiver. For each j-th bit B_j in $[b]_1$, the parties run a 1-out-of-2 C-OT protocol, where P_0 inputs the correlation function $f_{\Delta_j}(x) = ([a]_0 \cdot 2^j + x) \pmod{2^\ell}$ and obtains $(s_{j,0} = r_j, s_{j,1} = [a]_0 \cdot 2^j + r_j)$, where $r_j \in_R \mathbb{Z}_{2^\ell}$, and P_1 obtains s_{j,B_j}. Now the respective shares are $[[a]_0[b]_1]_0 = \sum_{j=0}^{\ell-1}(-r_j)$ and $[[a]_0[b]_1]_1 = \sum_{j=0}^{\ell-1} s_{j,B_j}$. Correctness and security were shown in [DSZ15].

Each 1-out-of-2 C-OT protocol [Ash+13] requires P_1 to send κ bits to P_0, who replies with ℓ bits. As we run the protocol twice also for the other cross-term, this yields total setup communication per MT of $2\ell(\kappa + \ell)$ bits.

4.2 Correlated MT Generation

Now, the goal is to generate an additional MT using the same share $[b]_i$, i.e., Arithmetic shares $[a']_0, [a']_1, [b]_0, [b]_1, [c']_0, [c']_1$ that satisfy $([a']_0 + [a']_1) \cdot ([b]_0 + [b]_1) = ([c']_0 + [c']_1) \pmod{2^\ell}$. As observed in [MRT20], P_1 inputs the same values corresponding to the $[b]_1$ into the C-OT protocols for the first MT and hence the messages from P_1 to P_0 can be reused. Therefore, only the second message from P_0 to P_1 consisting of ℓ bits has to be sent in each C-OT, i.e., an additional $2\ell^2$ bits for the second MT. Repeating this construction in parallel M times yields a total communication of $2\ell(\kappa + M \cdot \ell)$ bits for M multiplication triples. An additional optimization described in [DSZ15, §III-A5] sends only back those bits of the second C-OT message that are needed for the answer resulting in communication $2\ell(\kappa + M \cdot (\ell + 1)/2)$ bits. The benefit of this optimization technique depends on the parameters.

4.3 Improvement Factors

In this section, we discuss in more details how much this technique is able to decrease the total communication overhead compared to [DSZ15]. Generating M MTs with 1-out-of-2 C-OT requires $M \cdot 2\ell(\kappa + \ell)$ bits of communication, but this reduces down to $\alpha = M \cdot 2\ell(\kappa + (\ell + 1)/2)$ bits with the optimization from [DSZ15, §III-A5]. We note that our optimization can benefit from [DSZ15] as well and results in $\beta = 2\ell(\kappa + M \cdot (\ell + 1)/2)$ bits of communication. The achieved improvement factor is $\mathcal{R} = \alpha/\beta$. It is clear that the improvement depends on the used parameters, but at the same time approaches certain limits.

In FAPRIL, we fix $\kappa = 128$ and ℓ depends on the number of access points N (see Eq. (5) on page 8). If we fix $\ell = 16$, then $N \in [146, 291]$ and the improvement factor is shown in Eq. (6).

$$\mathcal{R}_M = \frac{273M}{17M + 256} \tag{6}$$

This is also shown in Fig. 1, from where we can observe that we are fairly quickly approaching improvement factor 15×. We can also set $M \to \infty$ in Eq. (6) and observe that $\mathcal{R}_M \to 273/17 \approx 16$.

For general cases, it is interesting to also know what is the expected improvement for shares in different rings $\mathbb{Z}_{2^\ell}$. For this, we take $\mathcal{R}$ and fix $\kappa = 128, M \to \infty$. The result is shown in Eq. (7).

$$\mathcal{R}_\ell = \frac{256}{\ell + 1} + 1 \tag{7}$$

We also show this in Fig. 2. We can observe that our optimization is very beneficial for small rings, i.e., when ℓ is small. E.g., we can gain up to 86× improvement in communication, when $\ell = 2$ (factor 80× is already reached when $M = 1100$). While these parameters are not reasonable for FAPRIL, other applications could greatly benefit from our optimization technique.

4.4 Application to Matrix Multiplication

The improved correlated MT generation technique described in §4.2 can also be applied to other applications where values re-occur such as matrix multiplication on Arithmetic/Delta shares, which is a very common operation, e.g., in Privacy-Preserving Machine Learning (PPML) [Ria+18; Pat+21]. For two matrices $\mathbf{A}^{p \times q}$ and $\mathbf{B}^{q \times r}$, we need to generate pqr MTs, which requires $pqr \cdot 2\ell(\kappa + \ell)$ bits of communication in the setup phase using the basic approach of [DSZ15]. This communication cost was also given in [Pat+21].

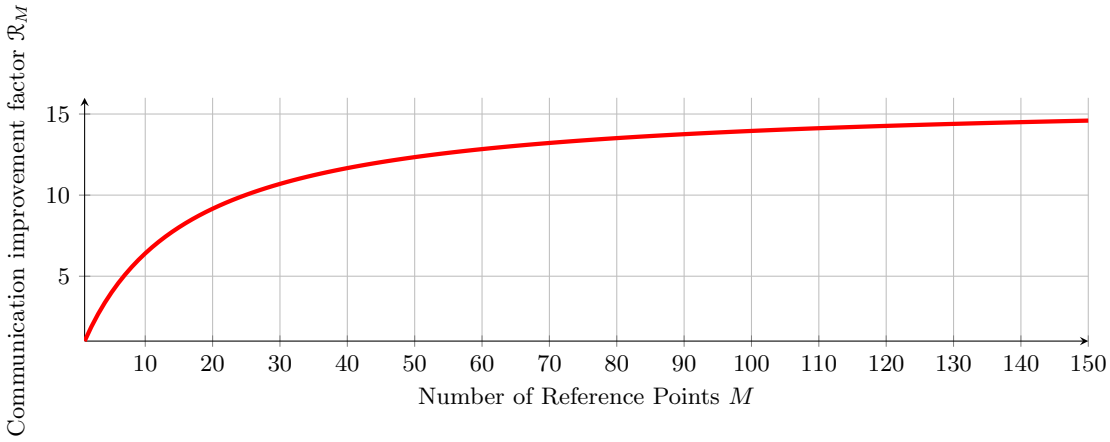

Fig. 1: Communication improvement factor $\mathcal{R}_M$ (see Eq. (6)) of our MT generation protocol over that of [DSZ15] as a function of the number of reference points M with security parameter $\kappa = 128$ and bit-length of the shares $\ell = 16$.

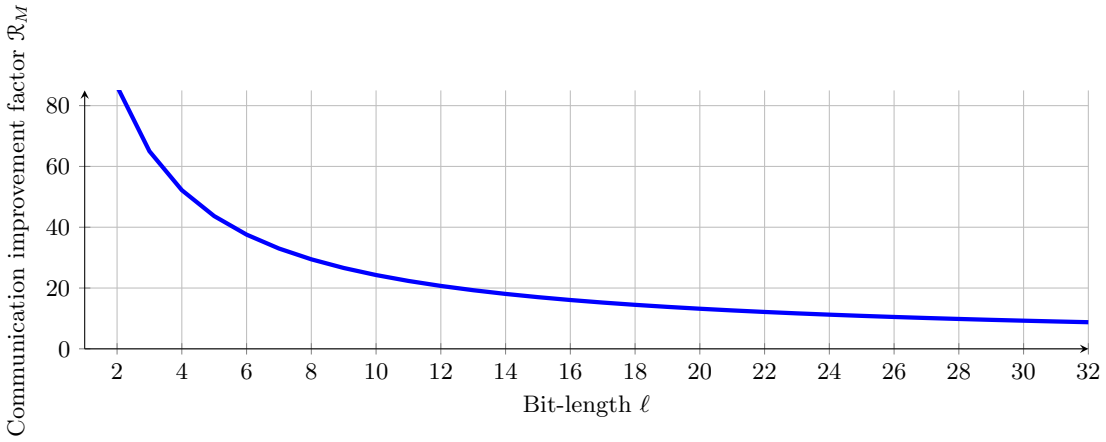

Fig. 2: Communication improvement factor $\mathcal{R}_\ell$ (see Eq. (7)) of our MT generation protocol over that of [DSZ15] as a function of the bit-length of the shares ℓ with security parameter $\kappa = 128$ and number of reference points $M \to \infty$.

With the optimization, the new computation cost is $2pq\ell(\kappa + r\ell)$ bits in the setup phase, which can be reduced even further with the optimization from [DSZ15, §III-A5] to $2pq\ell(\kappa + r(\ell+1)/2)$ bits.

If we fix $\kappa = 128, \ell = 32, p, q, r = 1000$, one matrix multiplication with the basic approach combined with the optimization [DSZ15, §III-A5], i.e., the communication cost is $2pqr\ell(\kappa + (\ell+1)/2)$ bits, requires 1.05 terabytes of communication. This is optimized to only 124 gigabytes resulting in factor 8.6× improvement in communication.

5 PERFORMANCE EVALUATION

In this section, we benchmark FAPRIL with our implementation using artificial databases of different sizes. We start by inspecting the setup and online phase overheads in §5.1. In §5.2, we inspect the overheads separately for each step of FAPRIL (described in §3.2). In §5.3, we compare our results against the currently best known Privacy-Preserving Indoor Localization (PPIL) schemes of [Jär+19; NJ20].

We implemented the client side of FAPRIL on Android smartphones. Our implementation relies on two open source libraries, namely *Mobile Private Contact Discovery*[6], providing the basis for the Garbled Circuit (GC) protocol, and *libOTe*[7], providing the basis for the Correlated Oblivious Transfer (C-OT) protocol. We develop the Delta sharing protocol based on [Pat+21]. Our full implementation is publicly available under the MIT License[8].

We use a Huawei P20 smartphone as our client device and a commodity server as specified in Tab. 1. For our network setup, we simulate a real-world LTE connection as follows: we use simulation scripts on the server side to increase the outgoing latency to 10 milliseconds (ms) and limit the bandwidth to 50 megabits per second (Mbit/s). The smartphone uses a Wi-Fi network to connect to the server, which increases the total Round-Trip Time (RTT) to approximately 16 ms.

For the following experiments, we fix the Received Signal Strength (RSS) quantization to 4 bits, use $k = 3$ for the k-Nearest Neighbor Algorithm (kNN), and fix the security parameter $\kappa = 128$. In order to get precise results, we run our experiments 10 times for each case and use the averages.

[6] https://github.com/contact-discovery/mobile_psi_cpp
[7] https://github.com/osu-crypto/libOTe
[8] https://github.com/encryptogroup/ppIndoorLocalization

Table 1: The hardware details of our devices.

	CPU Type	Clock Rate	Cores	RAM
Client	HiSilic. K970	1.8 GHz	8	4 GB
Server	Intel i9-7960X	2.8 GHz	16	128 GB

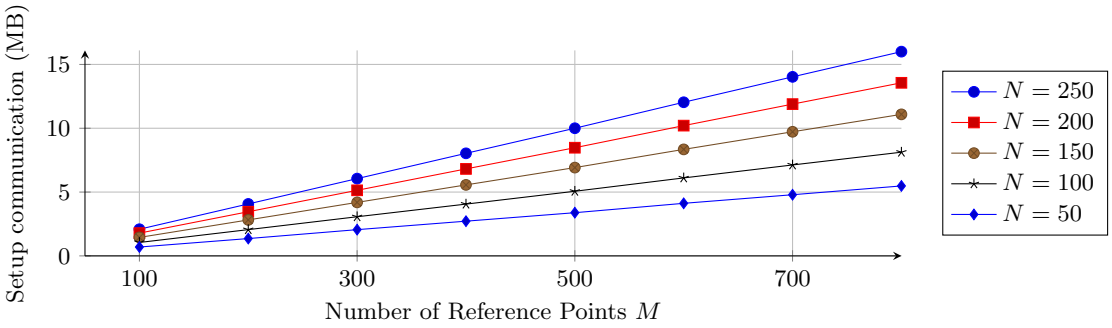

Fig. 3: The setup phase communication in megabytes (MB), when altering the number of Access Points N.

5.1 Benchmarks

The setup phase communication is shown in Fig. 3. As expected, it grows linearly in the number of Access Points (APs) N and the number of reference points M.

The setup run-times grow also linearly. When $N = 50, M = 100$, we only need about 0.28 seconds, but the run-time gets up to 1.48 seconds when we increase $M = 800$. For $N = 250, M = 100$, we have a run-time of 0.59 seconds, which grows to 4.14 seconds for the largest case, namely $N = 250, M = 800$. The online phase is more efficient, and even for our largest setting ($N = 250, M = 800$) we require only 0.20 megabytes of communication and the run-time is 0.15 seconds.

5.2 Detailed Benchmarks

We move to the more detailed view of the overheads. For this, we fix $N = 241, M = 505$ as it represents a real four-story building (see §2.1) and collect the results in Tab. 2. It is clear that the setup phase dominates the total costs for FAPRIL. More precisely, the clear bottleneck is the Multiplication Triple (MT) generation, which requires 7.55 megabytes of communication and a run-time of 2.22 seconds. We note here that our implementation does not take advantage of the optimization of [DSZ15, §III-A5] due to integration problems with the *libOTe* library. Therefore, the communication is almost doubled compared to the theoretical case, which is 4.2 megabytes (see §3.2). Our implementation also does not take full advantage of parallelization, which could possibly decrease the run-time significantly. FAPRIL can benefit directly from further improvements regarding MT generation, e.g., the recently proposed Silent OT technique [Boy+19b; Boy+19a] that improves communication at the expense of more computation or MT generation with Ring Learning With Errors (RLWE)-based additively Homomorphic Encryption (HE) [RSS19]. We leave implementation of these on mobile phones as future work.

Comparing our concrete communication overheads from Tab. 2b with the complexity analysis, we conclude that our benchmarks are aligned with the theoretical numbers (see §3.2), except for the MT generation (as explained previously) and for Client Garbled Inputs. The reason for the latter is that our implementation takes advantage of the more efficient "Delta-OT" [Bur+21] as implemented in *libOTe*.

5.3 Performance Comparison to Related Work

In this section, we compare the performance of FAPRIL to the client-server scheme NJ [NJ20], the Arithmetic and Yao sharing (AY) based scheme using squared Euclidean distance from [Jär+19], and our implementation

Table 2: Detailed performance evaluation of FAPRIL with $N = 241$ Access Points (APs), $M = 505$ reference points, $\kappa = 128$ security parameter, and $k = 3$ for the k-Nearest Neighbor Algorithm (kNN). The steps are explained in detail in §3.2.

(a) Run-times in milliseconds (ms).

Step	**Run-time (ms)**
Setup Phase	
MT Generation	2,218.8
Client Garbled Inputs	74.7
Garbled Circuit	256.4
Total	2,549.9
Online Phase	
Fingerprint Sharing	0.0
Distance Computation	16.0
Server Garbled Inputs	42.5
Garbled Circuit Eval.	37.5
Total	96.0

(b) Communication in kilobytes (kB) from client's perspective.

Step	**Uplink (kB)**	**Downlink (kB)**
Setup Phase		
MT Generation	3,867.3	3,867.3
Client Garbled Inputs	144.2	32 B
Garbled Circuit	–	2,193.6
Total	4,011.5	6,060.9
Online Phase		
Fingerprint Sharing	0.5	–
Distance Computation	1.0	1.0
Server Garbled Inputs	–	126.3
Garbled Circuit Eval.	–	27 B
Total	1.5	127.3

of the fully Yao sharing (Y) based scheme using 1-bit RSS quantization with Manhattan distance from [Jär+19]. The results are collected in Tab. 3.

We can see that FAPRIL clearly outperforms NJ [NJ20] except for the setup phase communication, which is expected to be in the favor of NJ even more when the number of APs N increases. On the other hand, we argue that the run-time advantages of FAPRIL makes it more attractive for practical usage. The run-time improvements of 8× for the setup phase and 27× for the online phase is significant. We also note that the improvements are expected to be considerably greater for larger settings.

Our comparison to AY [Jär+19] is not completely fair, since it was run in an outsourced setting, where all the computations occur on commodity servers. On the other hand, AY [Jär+19] includes an extra step at the end to obtain the location coordinates using Oblivious Array Access (OA). However, this step could be directly included in FAPRIL (see §3.3). Based on their evaluations, the step requires 1.3 megabytes of communication and takes 46 milliseconds. Even with this, FAPRIL achieves similar online run-times and almost 15× less communication in total compared to AY [Jär+19].

Finally, we implement the "Manhattan (1-bit, Y)" scheme from [Jär+19] and include the performance evaluations in Tab. 3. We emphasize that Y [Jär+19] cannot be compared with FAPRIL directly, since it is using only 1-bit RSS quantization (compared to 4 bits in FAPRIL). However, the authors of [Jär+19] noted that this scheme has potential also for efficient client-server localization with Yao sharing mainly

Table 3: Performance of FAPRIL compared to the related work with $N \in \{50, 241\}$ Access Points (APs), $M \in \{150, 505\}$ reference points, $\kappa = 128$ security parameter, and $k = 3$ for the k-Nearest Neighbor Algorithm (kNN). The most efficient values are marked in bold (excluding Y [Jär+19]).

	Run-time (s)			**Comm. (MB)**		
	Setup	Online	Total	Setup	Online	Total
$N = 50, M = 150$						
FAPRIL	**0.3**	**54 ms**	**0.4**	1.0	**34 kB**	**1.0**
NJ	2.5	1.5	4.0	**0.9**	0.2	1.1
$N = 241, M = 505$						
FAPRIL	2.5	**96 ms**	2.6	**9.8**	**129 kB**	**9.9**
AY[a]	**0.8**	0.15	**1.0**	164.0	2.7	167.0
Y [Jär+19][b]	0.7	123 ms	0.8	4.8	16 kB	4.8

[a]Includes Oblivious Array Access step (not in FAPRIL).
[b]Our implementation of "Manhattan (1-bit, Y)" scheme from [Jär+19].

due to the fact that the distance calculation is actually equivalent to Hamming weight circuit, which can be AND-optimized with [BP08]. We conclude that even though Y [Jär+19] seems to outperform FAPRIL, it is not very attractive due to its weaker accuracy guarantee (see [Jär+19; Ric+18]). We observe that (surprisingly) FAPRIL provides faster run-time in the online phase and the fully optimized MT generation for FAPRIL would result in 6.5 megabytes of communication in the setup phase, which is only 1.4× more compared to Y [Jär+19].

6 CONCLUSION AND FUTURE WORK

We presented FAPRIL, a Privacy-Preserving Indoor Localization (PPIL) scheme in the client-server setting based on state-of-the-art Secure Two-Party Computation (STPC). We implemented FAPRIL for Android smartphones and evaluated the performance with various benchmarks representing real world settings. With our optimization technique for generating Multiplication Triples (MTs), we showed that FAPRIL achieves practical run-times and communication. More precisely, for a real world university building setting with 241 Access Points (APs) and 505 reference points, we achieve a online run-time of 96 milliseconds and 129 kilobytes online communication. The setup phase takes 2.5 seconds and requires 9.8 megabytes of communication. With these results, we substantially outperform previously proposed solutions for PPIL and are even comparable to a less accurate lightweight PPIL scheme.

Lastly, we give a list of interesting topics for future research:

- Analyzing FAPRIL's power consumption on smartphones.
- Improving the efficiency of the vector-matrix multiplication with secret shares using different MT generation techniques, e.g., Ring Learning With Errors (RLWE)-based additively Homomorphic Encryption (HE) [RSS19].
- Developing protocols that hide the size of the database, the used distance metric, and/or the reference point coordinates from the clients.

Acknowledgments

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, and by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE.

References

[Alh+20] M. Y. Alhassan, D. Günther, Á. Kiss, and T. Schneider. "Efficient and scalable universal circuits". In: *J. Cryptology* (2020).

[All+15] S. Alletto, R. Cucchiara, G. Del Fiore, L. Mainetti, V. Mighali, L. Patrono, and G. Serra. "An indoor location-aware system for an IoT-based smart museum". In: *IEEE Internet of Things Journal* (2015).

[Ash+13] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. "More Efficient Oblivious Transfer and Extensions for Faster Secure Computation". In: *CCS*. 2013.

[Bar+21] P. Barsocchi, A. Calabrò, A. Crivello, S. Daoudagh, F. Furfari, M. Girolami, and E. Marchetti. "COVID-19 & privacy: Enhancing of indoor localization architectures towards effective social distancing". In: *Array* (2021).

[Bel+13a] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. "Efficient Garbling from a Fixed-Key Blockcipher". In: *S&P*. 2013.

[Bel+13b] S. M. Bellovin, R. M. Hutchins, T. Jebara, and S. Zimmeck. "When enough is enough: Location tracking, mosaic theory, and machine learning". In: *NYU Journal of Law & Liberty* (2013).

[BMR90] D. Beaver, S. Micali, and P. Rogaway. "The Round Complexity of Secure Protocols". In: *STOC*. 1990.

[BNS22] C. van der Beets, R. Nieminen, and T. Schneider. "FAPRIL: Towards Faster Privacy-Preserving Fingerprint-Based Localization". In: *SECRYPT*. 2022.

[Boy+19a] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. "Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation". In: *CCS*. 2019.

[Boy+19b] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. "Efficient Pseudorandom Correlation Generators: Silent OT Extension and More". In: *Advances in Cryptology – CRYPTO*. 2019.

[BP00] P. Bahl and V. N. Padmanabhan. "RADAR: An in-building RF-based user location and tracking system". In: *INFOCOM*. 2000.

[BP08] J. Boyar and R. Peralta. "Tight bounds for the multiplicative complexity of symmetric functions". In: *Theoretical Computer Science* (2008).

[Bur+21] S. S. Burra, E. Larraia, J. B. Nielsen, P. S. Nordholt, C. Orlandi, E. Orsini, P. Scholl, and N. P. Smart. "High-Performance Multi-party Computation for Binary Circuits Based on Oblivious Transfer". In: *J. Cryptology* (2021).

[Cap+11] S. Capkun, S. Ganeriwal, F. Anjum, and M. Srivastava. "Secure RSS-based localization in sensor networks". In: *Technical Report/ETH Zurich, Department of Computer Science* (2011).

[Cha+13] K. Chawla, C. McFarland, G. Robins, and C. Shope. "Real-time RFID localization using RSS". In: *ICL-GNSS*. 2013.

[Che+11] L. Chen, H. Kuusniemi, Y. Chen, L. Pei, T. Kröger, and R. Chen. "Information filter with speed detection for indoor Bluetooth positioning". In: *ICL-GNSS*. 2011.

[Che+17] L. Chen, S. Thombre, K. Järvinen, E. S. Lohan, A. Alén-Savikko, H. Leppäkoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. N. Ferrara, S. Honkala, J. Lindqvist, L. Ruotsalainen, P. Korpisaari, and H. Kuusniemi. "Robustness, Security and Privacy in Location-Based Services for Future IoT: A Survey". In: *IEEE Access* (2017).

[Che+20] H. Chen, I. Chillotti, Y. Dong, O. Poburinnaya, I. P. Razenshteyn, and M. S. Riazi. "SANNS: Scaling Up Secure Approximate k-Nearest Neighbors Search". In: *USENIX Security*. 2020.

[Des+17] G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, S. Zeitouni, and M. Zohner. "Pushing the Communication Barrier in Secure Computation using Lookup Tables". In: *NDSS*. 2017.

[DSZ15] D. Demmler, T. Schneider, and M. Zohner. "ABY – A Framework for Efficient Mixed-Protocol Secure Two-Party Computation". In: *NDSS*. 2015.

[Gil99] N. Gilboa. "Two party RSA key generation". In: *Advances in Cryptology – CRYPTO*. 1999.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. "How to Play ANY Mental Game". In: *STOC*. 1987.

[Gua+17] T. Guan, L. Fang, W. Dong, Y. Hou, and C. Qiao. "Indoor localization with asymmetric grid-based filters in large areas utilizing smartphones". In: *IEEE ICC*. 2017.

[Hae+04] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. "Practical robust localization over large-scale 802.11 wireless networks". In: *MobiCom*. 2004.

[Hak+15] A. Hakkarainen, J. Werner, M. Costa, K. Leppänen, and M. Valkama. "High-efficiency device localization in 5G ultra-dense networks: Prospects and enabling technologies". In: *IEEE Vehicular Technology Conference*. 2015.

[HLC17] S. He, W. Lin, and S.-H. G. Chan. "Indoor Localization and Automatic Fingerprint Update with Altered AP Signals". In: *IEEE Transactions on Mobile Computing* (2017).

[HOS15] P. Hallgren, M. Ochoa, and A. Sabelfeld. "Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol". In: *IEEE Privacy, Security and Trust*. 2015.

[HOS17] P. Hallgren, C. Orlandi, and A. Sabelfeld. "PrivatePool: Privacy-preserving ridesharing". In: *IEEE CSF*. 2017.

[Ish+03] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. "Extending Oblivious Transfers Efficiently". In: *Advances in Cryptology – CRYPTO*. 2003.

[Jär+18] K. Järvinen, Á. Kiss, T. Schneider, O. Tkachenko, and Z. Yang. "Faster privacy-preserving location proximity schemes". In: *CANS*. 2018.

[Jär+19] K. Järvinen, H. Leppäkoski, E. S. Lohan, P. Richter, T. Schneider, O. Tkachenko, and Z. Yang. "PILOT: Practical Privacy-Preserving Indoor Localization using OuTsourcing". In: *EuroS&P*. 2019.

[Kon+16] A. Konstantinidis, G. Chatzimilioudis, D. Zeinalipour-Yazti, P. Mpeis, N. Pelekis, and Y. Theodoridis. "Privacy-preserving indoor localization on smartphones". In: *IEEE International Conference on Data Engineering*. 2016.

[KPV12] A. Kushki, K. N. Plataniotis, and A. N. Venetsanopoulos. *WLAN positioning systems: Principles and applications in location-based services*. Cambridge University Press, 2012.

[KS08] V. Kolesnikov and T. Schneider. "Improved Garbled Circuit: Free XOR Gates and Applications". In: *ICALP*. 2008.

[Lad+05] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach. "Robotics-based location sensing using wireless ethernet". In: *Wireless Networks* (2005).

[Li+05] B. Li, Y. Wang, H. K. Lee, A. Dempster, and C. Rizos. "Method for yielding a database of location fingerprints in WLAN". In: *IEEE Proceedings – Communications* (2005).

[Li+14] H. Li, L. Sun, H. Zhu, X. Lu, and X. Cheng. "Achieving privacy preservation in WiFi fingerprint-based localization". In: *INFOCOM*. 2014.

[Li+17] R. Li, T. Song, N. Capurso, J. Yu, J. Couture, and X. Cheng. "IoT applications on secure smart shopping system". In: *IEEE Internet of Things Journal* (2017).

[Liu+07] H. Liu, H. Darabi, P. Banerjee, and J. Liu. "Survey of Wireless Indoor Positioning Techniques and Systems". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* (2007).

[Loh+17a] E. S. Lohan, P. Richter, V. L. Sabola, J. A. Lopez-Salcedo, G. Seco-Granados, H. Leppäkoski, and E. S. Santiago. "Location privacy challenges and solutions – parts I and II". In: *Inside GNSS* (2017).

[Loh+17b] E. S. Lohan, J. Torres-Sospedra, H. Leppäkoski, P. Richter, Z. Peng, and J. Huerta. "Wi-Fi Crowdsourced Fingerprinting Dataset for Indoor Positioning". In: *Data* (2017).

[LP09] Y. Lindell and B. Pinkas. "A Proof of Security of Yao's Protocol for Two-Party Computation". In: *J. Cryptology* (2009).

[LTP17] C. Langlois, S. Tiku, and S. Pasricha. "Indoor localization with smartphones: Harnessing the sensor suite in your pocket". In: *IEEE Consumer Electronics Magazine* (2017).

[Men+18] G. M. Mendoza-Silva, P. Richter, J. Torres-Sospedra, E. S. Lohan, and J. Huerta. "Long-Term WiFi Fingerprinting Dataset for Research on Robust Indoor Positioning". In: *Data* (2018).

[Mis+21] P. K. Mishra, D. Rathee, D. H. Duong, and M. Yasuda. "Fast secure matrix multiplications over ring-based homomorphic encryption". In: *Inf. Secur. J. A Glob. Perspect.* (2021).

[MRT20] P. Mohassel, M. Rosulek, and N. Trieu. "Practical Privacy-Preserving K-means Clustering". In: *Proc. Priv. Enhancing Technol.* (2020).

[NJ20] R. Nieminen and K. Järvinen. "Practical privacy-preserving indoor localization based on secure two-party computation". In: *IEEE Transactions on Mobile Computing* (2020).

[NLY08] A. S. I. Noh, W. J. Lee, and J. Y. Ye. "Comparison of the mechanisms of the Zigbee's indoor localization algorithm". In: *IEEE/ACIS Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*. 2008.

[Pai99] P. Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *Advances in Cryptology – EUROCRYPT*. 1999.

[Pat+21] A. Patra, T. Schneider, A. Suresh, and H. Yalame. "ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation". In: *USENIX Security*. 2021.

[Ria+18] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar. "Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications". In: *ASIACCS*. 2018.

[Ric+18] P. Richter, Z. Yang, O. Tkachenko, H. Leppäkoski, K. Järvinen, T. Schneider, and E. S. Lohan. "Received Signal Strength Quantization for Secure Indoor Positioning via Fingerprinting". In: *ICL-GNSS*. 2018.

[RR21] M. Rosulek and L. Roy. "Three Halves Make a Whole? Beating the Half-Gates Lower Bound for Garbled Circuits". In: *CRYPTO*. 2021.

[RSS19] D. Rathee, T. Schneider, and K. Shukla. "Improved multiplication triple generation over rings via RLWE-based AHE". In: *CANS*. 2019.

[SFR20] H. Shaul, D. Feldman, and D. Rus. "Secure k-ish Nearest Neighbors Classifier". In: *Proc. Priv. Enhancing Technol.* (2020).

[Shu+14] T. Shu, Y. Chen, J. Yang, and A. Williams. "Multi-lateral privacy-preserving localization in pervasive environments". In: *INFOCOM*. 2014.

[Son+15] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, and F. Koushanfar. "Compacting privacy-preserving k-nearest neighbor search using logic synthesis". In: *ACM DAC*. 2015.

[Sti+17] S. Stirbys, O. A. Nabah, P. Hallgren, and A. Sabelfeld. "Privacy-preserving location-proximity for mobile apps". In: *IEEE Parallel, Distributed and Network-based Processing*. 2017.

[Tao+03] P. Tao, A. Rudys, A. M. Ladd, and D. S. Wallach. "Wireless LAN location-sensing for security applications". In: *ACM Workshop on Wireless Security*. 2003.

[TL13] J. Talvitie and E. S. Lohan. "Modeling received signal strength measurements for cellular network based positioning". In: *ICL-GNSS*. 2013.

[WAP17] L. Wang, Y. Aono, and L. T. Phong. "A New Secure Matrix Multiplication from Ring-LWE". In: *CANS*. 2017.

[Yao86] A. C. Yao. "How to Generate and Exchange Secrets". In: *FOCS*. 1986.

[Yas+17] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios. "Recent advances in indoor localization: A survey on theoretical approaches and applications". In: *IEEE Communications Surveys & Tutorials* (2017).

[Yin+17] Z. Yin, C. Wu, Z. Yang, and Y. Liu. "Peer-to-peer indoor navigation using smartphones". In: *IEEE Journal on Selected Areas in Communications* (2017).

[YJ18] Z. Yang and K. Järvinen. "The Death and Rebirth of Privacy-Preserving WiFi Fingerprint Localization with Paillier Encryption". In: *INFOCOM*. 2018.

[Zie+14] J. H. Ziegeldorf, N. Viol, M. Henze, and K. Wehrle. "Poster: Privacy-preserving indoor localization". In: *ACM WiSec* (2014).

[ZRE15] S. Zahur, M. Rosulek, and D. Evans. "Two Halves Make a Whole - Reducing Data Transfer in Garbled Circuits Using Half Gates". In: *Advances in Cryptology – EUROCRYPT*. 2015.

[ZS21] M. Zuber and R. Sirdey. "Efficient homomorphic evaluation of k-NN classifiers". In: *Proc. Priv. Enhancing Technol.* (2021).