Faster Privacy-Preserving Location Proximity Schemes

(Full Version)*

Kimmo Järvinen[1], Ágnes Kiss[2], Thomas Schneider[2], Oleksandr Tkachenko[2], and Zheng Yang[3]

[1] University of Helsinki, Finland
kimmo.u.jarvinen@helsinki.fi
[2] TU Darmstadt, Germany
{kiss, schneider,tkachenko}@encrypto.cs.tu-darmstadt.de
[3] Singapore University of Technology & Design, Singapore
zheng_yang@sutd.edu.sg

Abstract. In the last decade, location information became easily obtainable using off-the-shelf mobile devices. This gave a momentum to developing Location Based Services (LBSs) such as location proximity detection, which can be used to find friends or taxis nearby. LBSs can, however, be easily misused to track users, which draws attention to the need of protecting privacy of these users.

In this work, we address this issue by designing, implementing, and evaluating multiple algorithms for Privacy-Preserving Location Proximity (PPLP) that are based on different secure computation protocols. Our PPLP protocols are well-suited for different scenarios: for saving bandwidth, energy/computational power, or for faster runtimes. Furthermore, our algorithms have runtimes of a few milliseconds to hundreds of milliseconds and bandwidth of hundreds of bytes to one megabyte. In addition, the computationally most expensive parts of the PPLP computation can be precomputed in our protocols, such that the input-dependent online phase runs in just a few milliseconds.

Keywords: Location Privacy, Proximity, Secure Computation, Homomorphic Encryption

1 Introduction

Nowadays, many mobile devices (e.g., smartphones or tablets) can easily measure and report precise locations in real time, so that several Location-Based Services (LBSs) over mobile networks have emerged in recent years. A basic LBS is location proximity detection that enables a user to test whether or not another user is nearby. This promising function has boosted the development of social applications to help users to find their nearby friends [LRQ+17], Uber cars [HOS17], or

[*] Please cite the conference version of this work published at CANS'18 .

medical personnel in an event of emergency [YLC+17]. Although some users have nothing against sharing their location, many privacy-aware users want to protect it from third parties. The reason for that are the possible privacy threats caused by location proximity detection [STSY10] that may lead to serious consequences, including unintended tracking, stalking, harassment, and even kidnapping. Potential adversaries range from curious social media contacts to abusive family members and even professional criminals (e.g., burglars checking if a victim is at home), and sometimes the level of their technological skills may be high. Hence, it is desirable to provide location proximity detection services which preserve the privacy of the users' exact location. Furthermore, modern law (e.g., the EU General Data Protection Regulation (GDPR)[4]) obligates companies to better protect users' privacy. This affects companies such as smartphone manufacturers that frequently offer built-in LBSs and LBS providers that provide additional privacy-preserving LBSs based on the result of the Privacy-Preserving Location Proximity (PPLP) protocol, e.g., for advertising ongoing movies in nearby cinemas to friends in the vicinity.

1.1 Our Contributions

Our contributions are as follows:

Efficient PPLP Schemes. We design and evaluate practically efficient Euclidean distance-based Privacy-Preserving Location Proximity (PPLP) schemes (i) using a mix of Secure Two-Party Computation (STPC) protocols, (ii) using DGK encryption [DGK09] and Bloom filters [Blo70], and (iii) using exponential ElGamal encryption [Gam85] over elliptic curves (ECs) and Bloom filters. This allows us to provide custom solutions for different PPLP applications with different requirements with respect to communication, computation, and runtime.

Optimizations. We present an optimization of the Boolean circuit for computing Euclidean and Manhattan distance for 32-bit values that reduces the number of AND gates by up to 22 %.

Pre-computation. We consider two scenarios where (i) a precomputation scenario where two parties run a PPLP protocol on an ongoing basis, which allows pre-computations (e.g., overnight while charging) and substantially reduces computation and communication in the online phase, and (ii) an ad-hoc scenario where two strangers run a PPLP protocol only once (e.g., for mobile health care), and pre-computations are not possible.

Extensive Performance Evaluation. We give an extensive communication comparison of our PPLP protocols and the PPLP protocols presented in recent related work. Furthermore, we implement our most efficient protocols (two STPC-based and one EC-ElGamal-based algorithm) and give a runtime comparison of them and the most efficient recently introduced PPLP protocol of Hallgren et al. [HOS15, Hal17]. Additionally, we run our protocols in a real-world mobile Internet setting.

[4] https://www.eugdpr.org/

1.2 Related Work

So far several solutions for privacy-preserving location proximity (PPLP) schemes have been proposed, e.g., [CZBP06, ZGH07, ŠTŠ$^+$09, STSY10, MFB$^+$11, NTL$^+$11, ŠG14, HOS15, ZWL$^+$17, ZLLH17]. In early literature [CZBP06], privacy-preserving location proximity computation is realized by an imprecise location-based range query that allows a user to approximately learn if any of its communication partners is within a fixed distance from her current location. To realize such queries, a user's cloaked location (i.e., the precise location of the user is put into a larger region) is sent to the service provider which handles the service request and sends back a probabilistic result to the user. However, this scheme may leak some location information since the service provider knows each individual is within a particular region.

Since then there is a large number of works (e.g., [STSY10, FVM$^+$10, STSY10, ZLLH17]) on using a (semi-)trusted third party for assisting the clients in location proximity detection. [ŠTŠ$^+$09] introduces a PPLP solution called FriendLocator in the client-server setting. Here, each user first maps her location into a shared grid cell (granule), and the converted location is encrypted and sent to the location server who will blindly compute the proximity results for the user. Similar approaches relying on geographic grid are adopted in [STSY10, FVM$^+$10].

In the recent work [ZLLH17], Zheng et al. proposed a novel scheme which is based on spatial-temporal location tags that are extracted from environmental signals. A user can learn a group of users that are within her vicinity region with the help of a semi-trusted server. However, collaborating with a third party (whose reputation is uncertain) for proximity detection may incur the risk of compromising location privacy or many other security issues. For better privacy protection, it would therefore be of great interest to develop PPLP schemes without requiring the existence of a (semi-)trusted third party.

Zhong et al. [ZGH07] present three PPLP protocols (called Louis, Lester and Pierre) [AD01]. Lester and Pierre do not rely on any third party. The common construction idea behind those protocols is to compute the location distance using additively homomorphic public key encryption (AHPKE) between two principals with a distance obfuscation technique. However, in all their schemes the users learn the mutual distance that might be sensitive information in many situations.

Narayanan et al. [NTL$^+$11] show three PPLP protocols that reduce the proximity detection problem to private equality testing (in the first two protocols) or private set intersection (in the third protocol). Their protocols are run based on the location which is defined as a set of adjacent triangles of a hexagon (that divides a grid). The proximity detection is achieved by testing whether two users share at least one triangle. However, as discussed in [ŠG14], the protocols of [NTL$^+$11] may introduce different errors in practice.

Šeděnka et al. [ŠG14] present three hybrid PPLP protocols that combine AHPKE schemes with secure two-party computation (STPC). Two users would first use the AHPKE scheme (e.g., Paillier [Pai99]) to privately compute the distance of their locations (with different distance equations in each protocol), and then run a STPC protocol (e.g., the private inequality test protocol from [EFG+09], or garbled circuits [KSS09]) to test whether the resulting location distance is within a pre-defined threshold. However, these PPLP protocols incur a high communication and computation overhead. Furthermore their protocols have multiple rounds of communication and using OT-based multiplication, which we use in our PPLP protocol $\mathsf{ABY}_{\mathsf{AY}}$ (cf. §3.2), is substantially more efficient than AHPKE as shown in [DSZ15].

Hallgren et al. [HOS15] develop a PPLP protocol built on only AHPKE to test whether two users' locations are within a given distance threshold (without a trusted third party). Their construction makes use of a similar distance obfuscation method as in [ZGH07]. The main difference is that Hallgren et al.'s scheme hides the exact distance between two users.

In recent work, Zhu et al. [ZWL+17] propose two efficient PPLP schemes for different geometric situations (i.e., polygon or circle). However, their schemes are subject to linear equation solving attacks. Namely, a malicious sender who honestly follows the protocol execution can learn the location coordinates (x, y) of a receiver by solving the relevant linear equation (involving x and y) implied by the proximity answers returned by the receiver (in one query). The major problem of these schemes is that two equations share the same randomness which can be eliminated by a division. We show the attacks against Zhu et al.'s protocols in Appendix A.

2 Preliminaries

General Notations. We let κ be the security parameter and ρ be the statistical security parameter. Let $[n] = \{1, \ldots, n\}$ denote the set of integers between 1 and n. We write $a \overset{\$}{\leftarrow} S$ to denote the operation which samples a uniform random a element from set S. Let $\|$ denote the concatenation operation of two strings, $|a|$ denote the bit-length of a string a, and $\#S$ denotes the number of elements in set S.

Euclidean Distance. For computing the distance in our Privacy-Preserving Location Proximity (PPLP) protocols, we use Euclidean distance, which is computed as follows for two dimensions: $d \leftarrow \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$. However, since the computation of square root is costly in secure computation, we calculate the squared Euclidean distance as $d^2 \leftarrow (x_0 - x_1)^2 + (y_0 - y_1)^2$ and compare it with the squared threshold $d^2 \overset{?}{<} T^2$ to determine if two users of the PPLP protocol are close to each other.

In the following, we review the cryptographic tools used in our paper.

2.1 Secure Two-Party Computation

We implement our PPLP protocols using the ABY framework for mixed-protocol Secure Two-Party Computation (STPC) [DSZ15]. We make use of two sharing types implemented in ABY: Yao and Arithmetic sharing.

Yao Sharing. Yao sharing denotes Yao's Garbled Circuits (GCs) protocol [Yao82]. Using GCs, two mistrusting parties P_0 and P_1 can securely compute a public function f on their respective inputs x_0 and x_1. For this, P_0 garbles the plaintext Boolean circuit C (a Boolean circuit which represents f) into garbled circuit $\widetilde{C}$. P_0 sends $\widetilde{C}$ and its garbled inputs to P_1. P_1 obtains its garbled inputs using Oblivious Transfer (OT) [NP01, ALSZ13], and P_1 evaluates $\widetilde{C}$. Depending on which party gets the output, either P_0 sends the decryption keys for the output to P_1, or P_1 sends the obtained garbled outputs to P_0, or both parties do this if they both get the output. We denote shares of input bit x as $((k^0, k^1), k^x)$, where P_0 holds both keys $\langle x\rangle_0^Y = (k^0, k^1)$ and P_1 holds the key that corresponds to its input bit x, i.e., $\langle x\rangle_1^Y = k^x$. In Yao sharing, evaluation of XOR gates is performed locally without communication [KS08], whereas evaluation of AND gates requires sending $2k$ bits [ZRE15].

Arithmetic Sharing. Arithmetic sharing denotes a generalization of the GMW protocol [GMW87] for unsigned integer numbers in the ring $\mathbb{Z}_{2^\ell}$. In Arithmetic sharing, an integer x is shared between P_0 and P_1 as $x = \langle x\rangle_0^A + \langle x\rangle_1^A \mod 2^\ell$, where P_0 holds $\langle x\rangle_0^A$ and P_1 holds $\langle x\rangle_1^A$. The function to be evaluated is represented as arithmetic circuit, which operates on unsigned integer values and consists of addition, subtraction, and multiplication gates modulo 2^ℓ only. Addition and subtraction gates can be evaluated locally without interaction between the parties, whereas evaluation of multiplication gates requires interaction and OT-based precomputations [DSZ15].

Notation. A share of value x held by Party P_i in sharing $t \in \{A, Y\}$, where A denotes Arithmetic sharing and Y denotes Yao sharing, is written as $\langle x\rangle_i^t$. In protocol descriptions, the party index is omitted because both parties perform the same operations. Operation $\odot$ on shares $\langle x\rangle^t$ and $\langle y\rangle^t$ in sharing t is denoted as $\langle z\rangle^t = \langle x\rangle^t \odot \langle y\rangle^t$. We write a conversion of Yao sharing $\langle x\rangle^Y$ to Arithmetic sharing as $\langle x\rangle^A = Y2A(\langle x\rangle^Y)$ and a conversion of Arithmetic sharing to Yao sharing as $\langle x\rangle^Y = A2Y(\langle x\rangle^A)$.

2.2 Additively Homomorphic Public-Key Encryption Scheme

An additively homomorphic public-key encryption (AHPKE) scheme is a probabilistic encryption scheme which consists of the following three algorithms:

- **Key Generation** (KGen). Given the security parameter κ, the algorithm returns the public and private key pair (pk, sk).
- **Encryption** (Enc). This algorithm takes a message $m \in \mathcal{M}$ from a plaintext space $\mathcal{M}$ and a public key pk as inputs, and outputs a ciphertext $c \in \mathcal{C}$ where $\mathcal{C}$ is the ciphertext space.
- **Decryption** (Dec). This algorithm takes the secret key sk and a ciphertext as inputs, and outputs the plaintext m.

	DGK [DGK09]	Lifted ElGamal [ElG84]
$\mathsf{KGen}(\kappa)$	1. Choose two random large primes p, q s.t. $\|p\| = \|q\| = \kappa/2$; 2. $n := p \cdot q$; 3. Choose ℓ-bits prime u, s.t. $u\|(p-1)$ and $u\|(q-1)$; 4. Choose ϕ-bits primes (v_p, v_q), s.t. $v_p\|(p-1)$ and $v_q\|(q-1)$; 5. Choose (g, h) of orders (uv_pv_q, v_pv_q); 6. $pk = (n, g, h, u)$, $sk = (p, q, v_p, v_q)$.	1. Choose ϕ-bits prime p; 2. Choose points $P, Q \in EC$; 3. $y \xleftarrow{\$} \mathbb{Z}_p^*$, $Y = yP$; 4. $pk = (p, P, Q, Y)$, $sk = y$.
$\mathsf{Enc}(pk, m)$	1. $r \xleftarrow{\$} \mathcal{R}_{\mathsf{D}} = \{0,1\}^{2.5\phi}$; 2. $C = g^m \cdot h^r \bmod n$.	1. For $m \in \mathbb{Z}_p$, $r \xleftarrow{\$} \mathbb{Z}_p^*$; 2. $C = (R, V) = (rP, rY + mQ)$.
$\mathsf{Dec}(sk, C)$	1. $C^{v_p} \bmod p = g^{v_p m} \bmod p$; 2. Calculate m by Pohlig-Hellman Alg. [PH78].	1. $mQ = V - yR$. Full decryption is not required.

Table 1: Additively homomorphic public-key encryption (AHPKE) schemes used in this paper.

For two ciphertext $C_1 = \mathsf{Enc}(pk, m_1)$ and $C_2 = \mathsf{Enc}(pk, m_2)$, we have the following additively homomorphic properties:

$$\mathsf{Dec}(sk, C_1 \cdot C_2) = m_1 + m_2 \text{ and } \mathsf{Dec}(sk, C_1 \cdot C_2^{-1}) = m_1 - m_2.$$

Using the above homomorphic additions, it is also possible to efficiently compute multiplications and divisions by a plaintext value $v \in \mathcal{M}$ using the square-and-multiply algorithm:

$$\mathsf{Dec}(sk, C_1^v) = v \cdot m_1 \text{ and } \mathsf{Dec}(sk, C_1^{v^{-1}}) = m_1/v.$$

We require that the AHPKE schemes used in our implementations satisfy standard semantic security. In Table 1, we briefly review two concrete instantiations of AHPKE, i.e., the construction by Damgård et al. (DGK) [DGK09] and Lifted ElGamal [ElG84] instantiated with Elliptic Curve Cryptography (ECC). Let $EC : y^2 = x^3 + ax + b$ denote an elliptic curve over a prime field $\mathsf{GF}(p)$ with curve parameters $a, b \in \mathsf{GF}(p)$. When the modulus n is clear from the context, then the modular operation mod n may be omitted.

Bloom filter (BF) [Blo70] is a probabilistic data structure that provides space-efficient storage of a set and that can efficiently test whether an element is a member of the set. The probabilistic property of BF may lead to false positive matches, but no false negatives. It is well-known that the more elements are added to the BF, the larger the probability of false positives gets. To reduce the false positive rate, we follow the approach of [PPR05], i.e., a BF with $1.44\epsilon N$ bits for a set with size N has a false positive rate (FPR) of $2^{-\epsilon}$.

We review the algorithms of a Bloom filter as follows:

- **Filter initiation** ($\mathsf{BF.init}$). On input a set size N, this algorithm initiates the Bloom filter of bit length $1.44\epsilon N$.
- **Element insertion** ($\mathsf{BF.insert}$). This algorithm takes an element m as input, and inserts m into BF.

- **Element check** ($\mathsf{BF.check}$). This algorithm returns 1 if an element m is in BF, and 0 otherwise.
- **Element change** ($\mathsf{BF.Pos}$). This algorithm computes positions to be changed for element m in BF.

Random oracles were first introduced by Bellare and Rogaway [BR93] as a tool to prove security of a cryptographic scheme. In this work, we assume that the hash function is modeled as a random oracle. Basically, a random oracle is stateful, i.e., for a random oracle query $H(m)$ for some input $m \in \{0,1\}^*$, it proceeds as follows:

- With respect to the first query on m, the oracle just returns a truly random value r_m from the corresponding domain, and records the tuple (m, r_m) into its query list HList.
- If $m \in \mathsf{HList}$, then the oracle just returns its associated random value r_m recorded in HList.

3 Efficient PPLP Schemes from ABY

We show in this section how the ABY framework for Secure Two-Party Computation (STPC) [DSZ15] can be used for Privacy-Preserving Location Proximity (PPLP). For describing ABY-based protocols, we use the following notation described in Table 2. We design two protocols for PPLP: (i) based on Yao sharing only and (ii) based on a mix of Arithmetic and Yao sharing, which we describe in the following.

Term	Description
P_0, P_1	Parties that perform secure computation
$t \in \{A, Y\}$	Sharing types: Arithmetic or Yao
$\langle x \rangle_i^t$	Share x in sharing t held by party P_i
$\langle z \rangle^t = \langle x \rangle^t \odot \langle y \rangle^t$	Operation $\odot$ on shares $\langle x \rangle^t$ and $\langle y \rangle^t$
$\langle x \rangle^Y = A2Y(\langle x \rangle^A)$	Sharing conversion from Arithmetic to Yao sharing

Table 2: Notation used for describing our ABY-based protocols.

3.1 $\mathsf{ABY_Y}$: A PPLP Scheme from Yao Sharing

The advantage of the Yao-based PPLP protocol is that it has a small and constant number of rounds, which makes it well-suited for high-latency networks. Since we operate on unsigned integers in ABY, we must make sure that no underflows occur for which we see two possible options for computing the Euclidean distance: (i) compute the extended equation of Euclidean distance, i.e., $x_0^2 + x_1^2 - 2x_0x_1 + y_0^2 + y_1^2 - 2y_0y_1$, or subtract the smaller coordinate from the larger coordinate, i.e., $(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2$. PPLP calculation using the extended equation results in 6 multiplications (which are very expensive operations as they require a number of AND gates which is quadratic in the bitlength of the operands), whereas determining maxima requires only a linear overhead in the bitlength and only two expensive multiplications. This is why we choose and further improve approach (ii) as follows. The intuitive approach for the Yao-based Euclidean distance computation requires two MUX gates for each dimension for selecting x_{max} and x_{min} (resp. y_{max} and y_{min}). The functionality of multiplexer $c \leftarrow \text{MUX}(a, b, s)$ on inputs a and b, and selection bit s is defined as $c \leftarrow s == 0\ ?\ a : b$. We slightly improve this circuit by observing that instead of individually selecting the maximum and the minimum, we can also swap the order of the two x values if $x_1 > x_0$ (and the same for the y values). Hence, we replace the two MUX gates by one Conditional Swap gate, which using the construction of [KS08] has the same costs as only *one* MUX gate (ℓ AND gates, where ℓ is the bitlength of the operands). The functionality of Conditional Swap $(a', b') \leftarrow$CondSwap$(a,\ b,\ s)$ on inputs a and b, and selection bit s is defined as $a' \leftarrow a \oplus [(a \oplus b) \wedge s]$, $b' \leftarrow b \oplus [(a \oplus b) \wedge s]$. Although this technique brings only slight performance improvement for Euclidean distance (0.4 % fewer AND gates for 32-bit coordinates), it gains more significance when used in other privacy-preserving distance metrics, e.g., Manhattan distance (22 % fewer AND gates for 32-bit coordinates).

In our Yao-based PPLP scheme (denoted as $\mathsf{ABY_Y}$) given in Figure 1, the following gates are used: $2(\text{GT}(\ell)+\text{CondSwap}(\ell)+\text{SUB}(\ell)+\text{MUL}(\sigma))+ADD(\sigma)$, where $\text{GT}(\ell)$ is an ℓ-bit greater-than circuit (ℓ AND gates [KSS09]), $\text{CondSwap}(\ell)$ is an ℓ-bit Conditional Swap gate (ℓ AND gates [KS08]), $\text{SUB}(\ell)$ is an ℓ-bit subtraction circuit (ℓ AND gates [KSS09]), $\text{MUL}(\sigma)$ is a σ-bit multiplication circuit ($2\sigma^2 - \sigma$ AND gates [KSS09]), and $\text{ADD}(\sigma)$ is a σ-bit addition circuit (σ AND gates [BPP00]). The values ℓ and σ are the bitlengths of the computed values. In our setting, $\ell = 32$ bits (the bitlength of a coordinate) and $\sigma = 64$ bits (the bitlength of the resulting squared value).

The aforementioned gates result in the following communication requirements between parties: $6\ell + 4\sigma^2 - \sigma = 16\,512$ AND gates $= 528\,384$ bytes of communication with 256 bit communication per AND gate using the half-gates technique of [ZRE15]. This scheme requires 2 messages in the online phase.

3.2 $\mathsf{ABY_{AY}}$: A PPLP Scheme from Arithmetic and Yao Sharing

We design a protocol for PPLP using a mix of Arithmetic and Yao sharing, which we denote as $\mathsf{ABY_{AY}}$. The use of Arithmetic sharing is advantageous for this scheme — it (i) decreases the communication and computation overhead for the PPLP, and (ii) can decrease protocol runtimes in low-latency networks. However, $\mathsf{ABY_Y}$ can still be significantly faster in high-latency networks, such as LTE in areas with very poor signal reception, which is, however, uncommon in crowded areas where people usually meet. Our protocol requires the following gates: $6 \cdot \text{MUL}_\mathsf{A}(\sigma) + \text{A2Y}(\sigma) + \text{GT}(\sigma)$, where $\text{MUL}_\mathsf{A}(\sigma)$ is a σ-bit multiplication in Arithmetic sharing, $\text{A2Y}(\sigma)$ is a σ-bit Arithmetic to Yao sharing conversion,

```
$\langle \mathit{isNear} \rangle^Y \leftarrow \mathsf{ABY_Y}(\langle x_0 \rangle^Y, \langle x_1 \rangle^Y, \langle y_0 \rangle^Y, \langle y_1 \rangle^Y, \langle T \rangle^Y)$
$\langle \mathit{gtX} \rangle^Y \leftarrow \mathrm{GT}(\langle x_1 \rangle^Y, \langle x_0 \rangle^Y)$
$\mathit{swappedX} \leftarrow \mathrm{CondSwap}(\langle x_0 \rangle^Y, \langle x_1 \rangle^Y, \langle \mathit{gtX} \rangle^Y)$
$\langle x_{\mathit{max}} \rangle^Y \leftarrow \mathit{swappedX}[0]$
$\langle x_{\mathit{min}} \rangle^Y \leftarrow \mathit{swappedX}[1]$
$\langle \mathit{gtY} \rangle^Y \leftarrow \mathrm{GT}(\langle y_1 \rangle^Y, \langle y_0 \rangle^Y)$
$\mathit{swappedY} \leftarrow \mathrm{CondSwap}(\langle y_0 \rangle^Y, \langle y_1 \rangle^Y, \langle \mathit{gtY} \rangle^Y)$
$\langle y_{\mathit{max}} \rangle^Y \leftarrow \mathit{swappedY}[0]$
$\langle y_{\mathit{min}} \rangle^Y \leftarrow \mathit{swappedY}[1]$
$\langle d \rangle^Y \leftarrow (\langle x_{\mathit{max}} \rangle^Y - \langle x_{\mathit{min}} \rangle^Y)^2 + (\langle y_{\mathit{max}} \rangle^Y - \langle y_{\mathit{min}} \rangle^Y)^2$
return $\langle d \rangle^Y < \langle T \rangle^Y$
```

Fig. 1: Our PPLP protocol $\mathsf{ABY_Y}$ using only Yao sharing in ABY [DSZ15].

GT(σ) is a σ-bit greater-than gate (σ AND gates [KSS09]), and σ is the bitlength of the squared distance. Our protocol for mixed-protocol SMPC-based PPLP is shown in Figure 2.

In total, 6 multiplication gates in Arithmetic sharing, 1 Arithmetic to Yao conversion gate, and σ AND gates in Yao sharing are required in this scheme. This results in $12\sigma^2 + 19\kappa\sigma$ bits of communication. In our setting with the bitlength of the squared value $\sigma = 64$, this yields 45 056 bytes of communication, which is a factor 11× improvement over $\mathsf{ABY_Y}$. However, this scheme requires 4 messages in the online phase (2× more than for $\mathsf{ABY_Y}$).

```
$\langle \mathit{isNear} \rangle^Y \leftarrow \mathsf{ABY_{AY}}(\langle x_0 \rangle^A, \langle x_1 \rangle^A, \langle y_0 \rangle^A, \langle y_1 \rangle^A, \langle T \rangle^Y)$
$\langle x \rangle^A \leftarrow (\langle x_0 \rangle^A)^2 + (\langle x_1 \rangle^A)^2 - \langle x_0 \rangle^A \langle x_1 \rangle^A - \langle x_0 \rangle^A \langle x_1 \rangle^A$
$\langle y \rangle^A \leftarrow (\langle y_0 \rangle^A)^2 + (\langle y_1 \rangle^A)^2 - \langle y_0 \rangle^A \langle y_1 \rangle^A - \langle y_0 \rangle^A \langle y_1 \rangle^A$
$\langle d \rangle^Y \leftarrow \mathrm{A2B}(\langle x \rangle^A + \langle y \rangle^A)$
return $\langle d \rangle^Y < \langle T \rangle^Y$
```

Fig. 2: Our PPLP protocol $\mathsf{ABY_{AY}}$ using Arithmetic and Yao sharing in ABY [DSZ15].

4 Efficient PPLP Schemes from Homomorphic Encryption

In this section, we show how to build efficient privacy preserving location proximity (PPLP) schemes which are suitable for mobile devices. In our construction, we will extensively use a one-way hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ which will be modeled as a random oracle in the security analysis, where p is a large prime chosen in each scheme.

Overview. We first give an overview of our constructions. Consider the general scenario that a party A at location with coordinates (x_A, y_A) wants to know whether the other party B at location (x_B, y_B) is close to her without learning any information about B's location. Intuitively, if the distance d between their locations are smaller than a threshold τ, then we can say that they are near. If set $\mathsf{T} = \{d_1, \ldots, d_m\}$ denotes all possible Euclidean distances between two adjacent parties, then the location proximity problem is to determine whether d is in this public set or not. Since Euclidean distances are calculated as the sum of two squares $m = |\mathsf{T}| \approx \lambda \cdot \frac{\tau^2}{\sqrt{2 \ln \tau}}$, where $\lambda = 0.7642$ is the Landau-Ramanujan constant, since we insert only unique elements in T that are smaller than or equal to τ^2 [Fin03, §2.3].

However, the distance d should be also hidden from both parties to preserve privacy. Hence, we cannot let party A directly input the distance $x = d$ to test the location proximity. To protect the distance from A, we make use of additively homomorphic PKE scheme (either DGK or ElGamal shown in Table 1) to enable both parties to jointly compute a distance d based on party A's public key but B blinds d with two fresh random values (i.e., (r, s)). Namely, A will decrypt the ciphertext computed by B to get the blinded distance $\widetilde{d} = s \cdot (r + d) \bmod p$ where p is a prime. Our distance obfuscation method is inspired by the Lester protocol [ZGH07], but is tailored to the additively homomorphic PKE schemes we use. To allow A to get the location proximity result, we further randomize the set T to another set $\mathcal{X} = \{x_1, \ldots, x_m\}$, s.t. $x_i = H(s \cdot (r + d_i) \bmod p)$. It is not hard to see that if $H(\widetilde{d}) \in \mathcal{X}$, then $d \in \mathsf{T}$. We use a Bloom filter to store the set $\mathcal{X}$ to reduce the storage and communication costs.

Security Model. We consider the honest-but-curious (semi-honest) setting where both parties honestly follow the protocol specification without deviating from it in any way, e.g., providing malicious inputs. However, any party might passively try to infer information about the other party's input from the protocol messages. This model is formulated by ideally implementing the protocol with a Trusted Third Party (TTP) $\mathcal{T}$ which receives the inputs of both parties and outputs the result of the defined function. Security requires that, in the real implementation of the protocol (without a TTP), none of the parties learns more information than what is returned by $\mathcal{T}$ in the ideal implementation. Namely, for any semi-honest adversary that successfully attacks a real protocol, there must exist a simulator $\mathcal{S}$ that successfully attacks the same protocol in the ideal world. Let Dist be a function which takes as input the coordinates (x_A, y_A, x_B, y_B) of the two parties and outputs the distance d between them. In the following, we define an ideal functionality of PPLP.

An ideal functionality $\mathcal{F}_{\mathsf{PPLP}}$ of our upcoming PPLP protocol with private inputs x_A, y_A and x_B, y_B and a public distance set T with threshold $\tau \in \mathbb{N}$, is defined as:

$$\mathcal{F}_{\mathsf{PPLP}} : (x_A, y_A, T_i, x_B, y_B, \mathsf{T}) \rightarrow (\perp, (\mathsf{Dist}(x_A, y_A, x_B, y_B) \in \mathsf{T}?1 : 0)).$$

We say that a PPLP protocol Π securely realizes functionality $\mathcal{F}_{\mathsf{PPLP}}$ if: for all Probabilistic Polynomial Time (PPT) adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$, such that

$$\mathsf{REAL}(\Pi, \mathcal{T}, \mathcal{A}) \approx \mathsf{IDEAL}(\mathcal{F}_{\mathsf{PPLP}}, \mathcal{T}, \mathcal{S}),$$

where $\approx$ denotes computational indistinguishability.

4.1 Σ_{DGK}: A PPLP Scheme from DGK

We fist introduce our PPLP protocol Σ_{DGK} from DGK ($\mathsf{KGen}^{\mathsf{D}}$, $\mathsf{Enc}^{\mathsf{D}}$, $\mathsf{Dec}^{\mathsf{D}}$ as shown in Table 1), which provides a fast pre-computation phase. This PPLP scheme running between two parties A and B is shown in Figure 3. A learns the location proximity result, i.e., whether or not the distance between A and B is smaller than a pre-defined threshold T.

Remark 1. In our PPLP scheme, we consider some possible optimizations on generating the blinded distance. We separate the ciphertexts C_1, C_2, and C_3 into two steps. We observe that the exponentiations (e.g., $R_1 = h^{\widetilde{r_1}}$) related to the random values $(\widetilde{r_1}, \widetilde{r_2}, \widetilde{r_3})$ of these ciphertexts can be precomputed. Note that each computation of $R_i = h^{\widetilde{r_i}}$ needs a full exponentiation with 2.5ϕ bits exponent (e.g., $\phi = 256$). In contrast, the size of the location coordinate and the blinded distance, i.e., the encrypted message, is much smaller, e.g., $\rho = 16$ bits. Hence, an exponentiation related to a message (e.g., g^{-2x_A}) can be done more efficiently online. For the online phase, we only need to compute the exponentiations related to the messages, so that only three exponentiations with 'small' exponents (depending on the message space) are required at party A. We can do similar pre-computations at party B. Furthermore, in order to compute the ciphertext C_d at party B, we can use simultaneous multi-exponentiation (with variable bases) [MVOV96, Algorithm 14.88] to speed up the computation. Then, the computation of C_d roughly needs 1.3 times that of a regular modular exponentiation.

Theorem 1. *If DGK is semantically secure and the hash function H is modeled as random oracle, then the proposed PPLP scheme Σ_{DGK} in Fig. 3 is a secure computation of $\mathcal{F}_{\mathsf{PPLP}}$ in the honest-but-curious model.*

Proof. We present the security analysis with respect to two aspects: (i) no corrupted party B can learn the input set of an honest party A; (ii) no corrupted

Initialization

Party A		Party B
$((p,q,v_p,v_q),(n,g,h,u)) =$ $(sk_A^{\mathsf{D}}, pk_A^{\mathsf{D}}) \xleftarrow{\$} \mathsf{KGen}^{\mathsf{D}}(\kappa)$		Insert the sum of all suares up to τ^2 in $\mathsf{T} = \{d_1, \ldots, d_m\}$
	$\xrightarrow{pk_A^P}$	

Setup Phase

$(\widetilde{r_1}, \widetilde{r_2}, \widetilde{r_3}) \xleftarrow{\$} \mathcal{R}_{\mathsf{D}}$ For $i \in [3]$: $\quad R_i := h^{\widetilde{r_i}}$		$(r,s,w) \xleftarrow{\$} \mathbb{Z}_u^*$, $\widetilde{r} \xleftarrow{\$} \mathcal{R}_{\mathsf{D}}$ Define a space $\mathcal{X} = [x_1, \ldots, x_m]$ $\mathsf{BF.init}(m)$ For $i \in [m]$: $\quad \widetilde{d_i} = s \cdot (r + d_i) \bmod u$ $\quad x_i := H(\widetilde{d_i} \vert\vert w)$ $\quad \mathsf{BF.insert}(x_i)$ $C_r = \mathsf{Enc}^{\mathsf{D}}(pk_A^{\mathsf{D}}, r \cdot s)$ $R := h^{\widetilde{r}}$
	$\xleftarrow{\mathsf{BF}}$	
Store $(\{R_i\}_{i \in [3]}, \mathsf{BF})$		Store (C_r, s, R)

Online Phase

Input: Location (x_A, y_A)		Input: Location (x_B, y_B)
$U := x_A^2 + y_A^2$ $C_1 := g^U \cdot R_1$ $C_2 := g^{-2x_A} \cdot R_2$ $C_3 := g^{-2y_A} \cdot R_3$		$C_z = \mathsf{Enc}^{\mathsf{D}}(pk_A^{\mathsf{D}}, s \cdot (x_B^2 + y_B^2))$
	$\xrightarrow{\{C_1, C_2, C_3\}}$	
		$C_d = (C_1 \cdot C_2^{x_B} \cdot C_3^{y_B})^s \cdot C_z$ $\quad := (C_1 \cdot C_2^{x_B} \cdot C_3^{y_B})^s \cdot g^Z R$ $\widetilde{C_d} := C_d \cdot C_r$
	$\xleftarrow{\widetilde{C_d}, w}$	
$\widetilde{d} := \mathsf{Dec}^{\mathsf{D}}(sk_A, \widetilde{C_d})$ $\overline{d} := H(\widetilde{d} \vert\vert w)$ Output $\mathsf{BF.check}(\overline{d})$		

Fig. 3: Our PPLP protocol Σ_{DGK} using DGK encryption.

party A can learn the resulting distance. The security against corrupted party B is guaranteed by the security of DGK, since all inputs of A are encrypted. Hence, the simulator $\mathcal{S}$ can just replace the real ciphertexts with random ones. Any adversary distinguishing this change can be used to break DGK.

As for a corrupted party A, we claim that A cannot obtain any useful information from a blinded distance $\widetilde{d_i} = s \cdot (r + d_i)$ and the Bloom filter BF. We first show that the inputs (i.e., blinded distances) of the random oracles are unique in a query, so that each $\overline{d_i}$ is unique as well. Consider two possible blinded distances $\widetilde{d_1} = s \cdot (r + d_1)$ and $\widetilde{d_2} = s \cdot (r + d_2)$ in a location proximity query. Since each d_i is unique, so is $\widetilde{d_i}$. Hence, each x_i is generated by the random oracle with unique input in a query, so that it is independent of all others. In particular, there is no false negative. With respect to different queries, although A may obtain two distances $\widetilde{d_1} = s_1 \cdot (r_1 + d_1)$ and $\widetilde{d_2} = s_2 \cdot (r_2 + d_2)$ such that $\widetilde{d_1} = \widetilde{d_2}$, these two distances are associated with different random numbers $w_1 \neq w_2$. Hence, the blinded distance $\widetilde{d}$ and the random value w together would ensure the input of the random oracle to be unique through all queries with overwhelming probability. As a result, in the ideal world $\mathcal{S}$ could use randomly chosen strings to build a set $\mathcal{X}$ in a location proximity query instead of the results from the random oracle. Due to the properties of the Bloom filter, A cannot infer the position of a $\overline{d_i}$ (after decryption) in $\mathcal{X}$ from BF, where $\overline{d_i} = H(\widetilde{d_i}||t)$ is inserted in BF.

Furthermore, since a distance d is blinded by freshly chosen random values r and s, party A can neither infer r nor s from $\widetilde{d}$ with an overwhelming probability. Thus, A cannot decrypt the distance nor test 'candidate' $\widetilde{d'}$ (of her own choice) based on BF without knowing either r or s.

To summarize, the PPLP scheme is secure under the given assumptions.

4.2 Σ_{ElG}: A PPLP Protocol from ElGamal

In this section, we propose a PPLP protocol Σ_{ElG} from ElGamal ($\mathsf{KGen}^{\mathsf{E}}$, $\mathsf{Enc}^{\mathsf{E}}$, $\mathsf{Dec}^{\mathsf{E}}$ as shown in Table 1). The construction of Σ_{ElG} is similar to Σ_{DGK}. However, we observe that the full decryption in the online phase is not necessary for party A who only needs to know the location proximity result. Thus, we replace DGK with the ECC-based lifted ElGamal scheme which results in better online communication complexity. Moreover, when increasing the security parameter, the performance of ECC operations is better than that of arithmetic modulo an RSA modulus in DGK, so Σ_{ElG} is better suited for long-term security. The Σ_{ElG} PPLP protocol is shown in Figure 4.

Theorem 2. *If ElGamal is semantically secure and the hash function H is modeled as random oracle, then the proposed PPLP scheme Σ_{ElG} in Fig. 4 is a secure computation of $\mathcal{F}_{\mathsf{PPLP}}$ in the honest-but-curious model.*

The proof of this theorem is analogous to that of Theorem 1 and thus omitted.

5 Comparison and Experimental Results

In this section, we compare our proposed protocols with the state-of-the-art PPLP protocol of Hallgren et al. [HOS15, Hal17]. We instantiate all primitives in our PPLP protocols to achieve a security level of $\kappa = 128$ bits. The secret-shared coordinates in our benchmarks are of bitlength $\ell = 32$ bit and the secret-shared squared results are of bitlength $\sigma = 64$ bit. However, we restrict the cleartext domain of the coordinates to $\{0, \ldots, 2^{31.5} - 1\}$ s.t. the squared Euclidean distance fits into a 64-bit unsigned integer. This is sufficient for any coordinates on earth with sub-meter accuracy.

We benchmark our prototype C++ implementations of our PPLP protocols on two commodity servers equipped with Intel Core i7 3.5 GHz CPUs and 32 GB RAM. During our benchmarks, however, the maximum RAM requirements

	Initialization	
Party A		Party B
$(y, (p, P, Q, Y)) = (sk_A^E, pk_A^E) \xleftarrow{\$}$ $\mathsf{KGen}^{\mathsf{E}}(\kappa)$		Insert the sum of all suares up to τ^2 in $\mathsf{T} = \{d_1, \ldots, d_m\}$
	$\xrightarrow{pk_A^P}$	
	Setup Phase	
$(\widetilde{r_1}, \widetilde{r_2}, \widetilde{r_3}) \xleftarrow{\$} \mathbb{Z}_p^*$ $R_i := \widetilde{r_i}P$ $F_i := \widetilde{r_i}Y$		$(r, s, \widetilde{r}, w) \xleftarrow{\$} \mathbb{Z}_p^*$ Define a space $\mathcal{X} = [x_1, \ldots, x_m]$ $\mathsf{BF.init}(m)$ For $i \in [m]$: $\widetilde{d_i} = s \cdot (r + d_i)Q$ $x_i := H(\widetilde{d_i} \Vert w)$ $\mathsf{BF.insert}(x_i)$ $C_r = (R_r, V_r) = \mathsf{Enc}^{\mathsf{E}}(pk_A^E, r \cdot s)$ $R_z := \widetilde{r}P$ $F_z := \widetilde{r}Y$
	$\xleftarrow{\mathsf{BF}}$	
Store $(\{R_i, F_i\}_{i \in [3]}, \mathsf{BF})$		Store (C_r, s, R_z, F_z, w)
	Online Phase	
Input: Location (x_A, y_A)		Input: Location (x_B, y_B)
$u := x_A^2 + y_A^2$ $C_1 := (R_1, V_1 = F_1 + uQ)$ $C_2 := (R_2, V_2 = F_2 - 2x_AQ)$ $C_3 := (R_3, V_3 = F_3 - 2y_AQ)$		$z = x_B^2 + y_B^2$ $C_z := (R_z, F_z + zQ)$
	$\xrightarrow{\{C_1, C_2, C_3\}}$	
		$C_d := C_1 \cdot C_2^{x_B} \cdot C_3^{y_B} \cdot C_z$ $\widetilde{C_d} := C_d^s \cdot C_r$
	$\xleftarrow{\widetilde{C_d}, w}$	
$\widetilde{d} := \mathsf{Dec}^{\mathsf{E}}(sk_A, \widetilde{C_d})$ $\overline{d} := H(\widetilde{d} \Vert w)$ Output $\mathsf{BF.check}(\overline{d})$		

Fig. 4: Our PPLP protocol Σ_{ElG} using ElGamal encryption.

were in the order of a few dozen megabytes. The two machines are connected via Gigabit Ethernet. Each benchmarking result is averaged over 100 executions.

As shown in Tab. 3, our protocol Σ_{ElG} has the lowest online communication and also more efficient arithmetic than Σ_{DGK} due to the usage of ECC instead of modular arithmetic over a 3072-bit RSA modulus. Therefore, we implemented Σ_{ElG}, but not Σ_{DGK} because we expect its runtimes to be worse. When utilizing Bloom filters, we use a false positive rate of $2^{-\rho}$, where ρ is the statistical security parameter ($\rho = 40$) as before.

The underlying framework for our Σ_{ElG} implementation is the mcl library[5] that includes an optimized lifted ElGamal implementation. We use lifted ElGa-

[5] https://github.com/herumi/mcl

mal encryption over the elliptic curve `secp256k1` with key size of 256 bits and 128-bit security. The mcl library supports point compression, and therefore each elliptic curve point can be represented by $256 + 1$ bits. An ElGamal ciphertext consists of two elliptic curve points, i.e., 514 bits in total.

5.1 Communication

We compare the communication of our protocols in Tab. 3. As can be seen from the table, the online communication and the setup communication of the ABY-based protocols is constant, whereas for the public-key based protocols the setup communication grows superlinearly with $\frac{\tau^2}{\sqrt{2 \ln \tau}}$. The online round complexity of $\mathsf{ABY}_{\mathsf{Y}}$, Σ_{DGK}, and Σ_{ElG} is minimal, but larger for $\mathsf{ABY}_{\mathsf{AY}}$ due to the multiplication in Additive sharing and the conversion from Additive sharing to Yao sharing, which need additional rounds of interaction.

Protocol	$\mathsf{ABY}_{\mathsf{Y}}$ (§3.1)	$\mathsf{ABY}_{\mathsf{AY}}$ (§3.2)	Σ_{DGK} (§4.1)	Σ_{ElG} (§4.2)
Setup Communication [Bytes]	209 555	117 155	$\approx 5.5 \frac{\tau^2}{\sqrt{2 \ln \tau}}$	$\approx 5.5 \frac{\tau^2}{\sqrt{2 \ln \tau}}$
Online Communication [Bytes]	3 656	3 001	1 056	288
# Sequential Messages Online	2	4	2	2

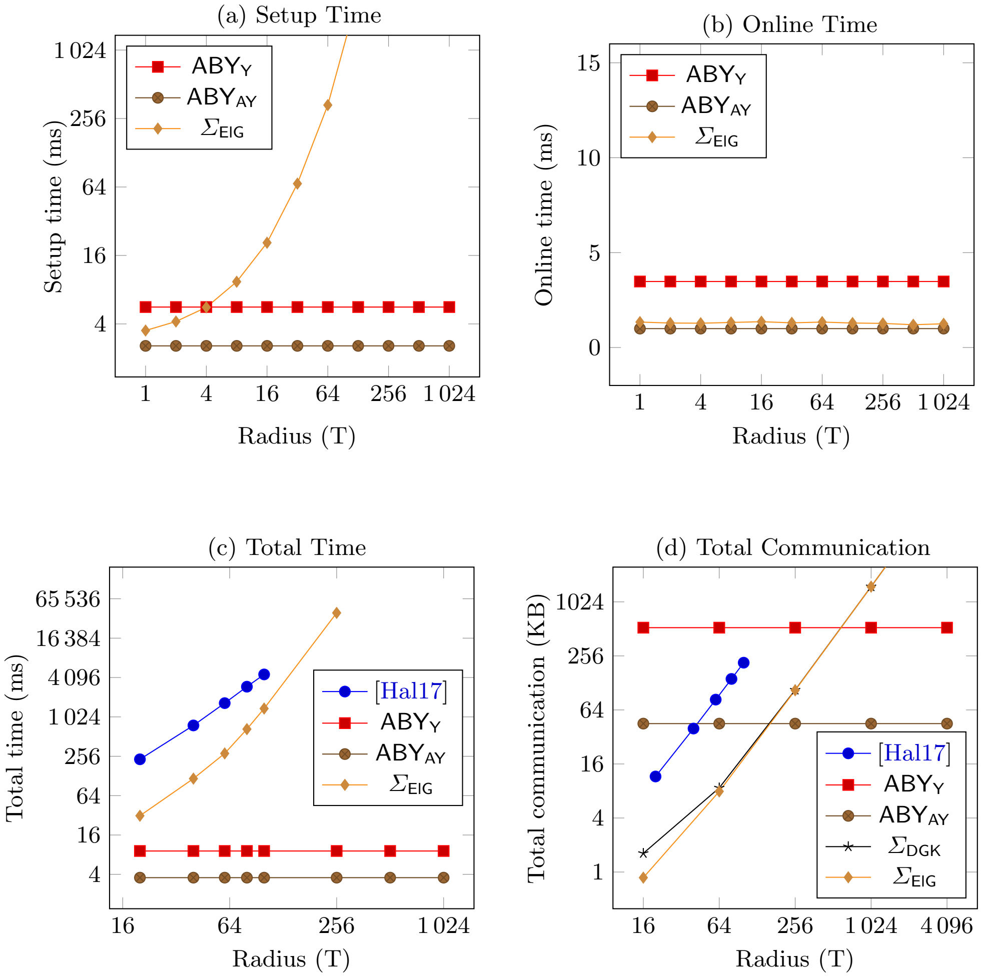
Table 3: Communication in Bytes and round complexities of our PPLP protocols for security level $\kappa = 128$ bit.

5.2 Benchmarks in a Local Network

In the following, we benchmark our protocols in a local Gigabit network with an average latency of 0.2 ms. We depict the runtimes and total communication of our PPLP protocols in Figure 5. We exclude the runtimes for the base-OTs (0.48 s in the LAN setting) for $\mathsf{ABY}_{\mathsf{Y}}$ and $\mathsf{ABY}_{\mathsf{AY}}$, because they need to be run only for the first execution of the protocol and hence are a one-time expense. In the same manner, we exclude the one-time cost of generating the key pair and sending the public key in Σ_{DGK} and Σ_{ElG} (6 milliseconds in the LAN setting).

Figure 5 confirms that the complexity of $\mathsf{ABY}_{\mathsf{Y}}$ and $\mathsf{ABY}_{\mathsf{AY}}$ is independent of τ, whereas the complexity of Σ_{ElG} grows superlinearly in τ. The online runtime also grows due to the growing size of the Bloom filter $\left(\approx 5.5 \frac{\tau^2}{\sqrt{2 \ln \tau}}\right)$ and therefore the number of (non-cryptographic) hash functions that need to be computed. $\mathsf{ABY}_{\mathsf{AY}}$ has the fastest online and setup runtime, and therefore, in total performance, it is substantially better than all other protocols.

As for the communication, Σ_{ElG} is more efficient than all other protocols for $\tau < 256$ (the communication of Σ_{DGK} is similar) and afterwards $\mathsf{ABY}_{\mathsf{AY}}$ is again the most efficient. Thus, Σ_{ElG} and $\mathsf{ABY}_{\mathsf{AY}}$ are beneficial for saving communication fees in mobile data networks which charge per KB. However, $\mathsf{ABY}_{\mathsf{AY}}$ has more


(a) Setup Time

(b) Online Time

(c) Total Time

(d) Total Communication

Fig. 5: Setup (a), online (b), and total (c) runtimes in milliseconds in a local Gigabit network with 0.2 ms average latency, and total communication (d) in Kilobytes of our PPLP schemes with security level $\kappa = 128$ bit in comparison with the ElGamal-based PPLP scheme of Hallgren et al. [HOS15, Hal17] with security level $\kappa = 112$ bit.

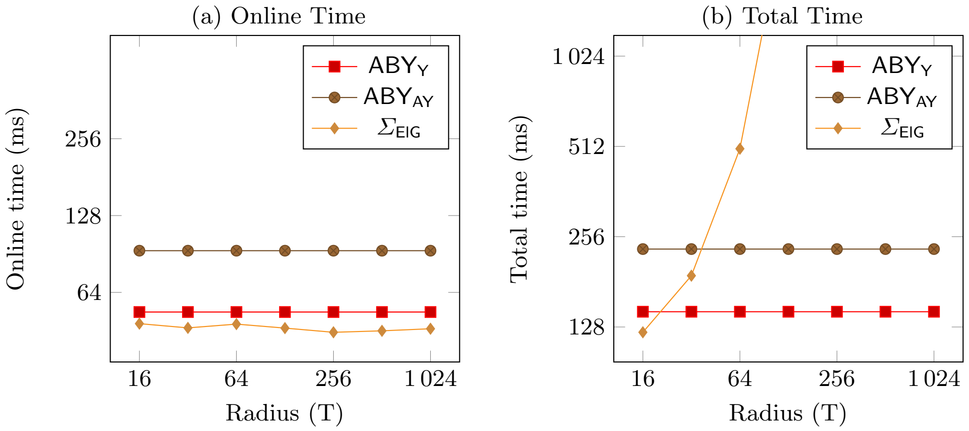


Fig. 6: Online (a) and total (b) runtimes in milliseconds of our PPLP protocols in the mobile Internet setting with 45 ms average network latency and 16 Mbps bandwidth.

communication rounds (cf. Table 3), so it is unclear if it is also more efficient in high-latency networks which we will investigate next.

5.3 Benchmarks in a Simulated Mobile Network

To show the practicality of our PPLP protocols, we simulate a mobile Internet connection, where we restrict the network bandwidth to 16 Mbps and the network latency to 45 ms, which are typical average parameters for mobile Internet nowadays[6]. Although the mobile Internet is still much slower than the cable Internet, most of the developed countries already support LTE[7] that provides transfer channels with bandwidth of dozens of Mbps and a typical transfer latency of just a few dozen milliseconds. Moreover, free Wi-Fi is becoming ubiquitous especially in big cities[8], which provides almost unlimited, fast, and low-latency access to PPLP. Thus, the prerequisites for using our algorithms greatly differ depending on the location of the deployment. Again, in the mobile Internet setting, we exclude the time needed for the base-OTs (0.75 s) and for generating the public key pair and sending the public key (0.05 s) as these are one-time expenses.

We depict the online and total runtimes in the mobile Internet setting in Figure 6. The online time for Σ_EIG is lowest due to the smallest communication and the minimal round complexity, followed by ABY_Y which also has minimal

[6] https://opensignal.com/
[7] https://gsacom.com/
[8] https://wifispc.com/

Protocol	$\mathsf{ABY}_{\mathsf{Y}}$ (§3.1)	$\mathsf{ABY}_{\mathsf{AY}}$ (§3.2)	Σ_{EIG} (§4.2)
Minimal online rounds	✓	✗	✓
Low communication	✗	✓	✓
Mostly symmetric crypto	✓	✓	✗
Performance independent of τ	✓	✓	✗
Resulting use cases	High latency, high bandwidth network; weak device; arbitrary τ	Low latency, medium bandwidth network; weak device; arbitrary τ	High latency, low bandwidth network; powerful device; small τ

Table 4: Summary and use-cases of our most efficient PPLP protocols.

round complexity, but up to factor 12× more communication (cf. Tab. 3). The online time for $\mathsf{ABY}_{\mathsf{AY}}$ is by factor 2× larger due to the larger round complexity. For the total runtimes, we see that Σ_{EIG} is the most efficient protocol for small thresholds of $\tau \leq 25$ from when on the constant runtime of $\mathsf{ABY}_{\mathsf{Y}}$ with 143 ms is most efficient. The total runtime of $\mathsf{ABY}_{\mathsf{AY}}$ is not competitive and almost twice as high as that of $\mathsf{ABY}_{\mathsf{Y}}$ due to the higher round complexity[9].

5.4 Summary

We briefly summarize the properties of and use-cases for our PPLP protocols in Table 4. Since all our PPLP protocols have different strengths, we give possible use-cases in the following: $\mathsf{ABY}_{\mathsf{Y}}$ is advantageous in high-latency networks with high bandwidth; $\mathsf{ABY}_{\mathsf{AY}}$ is better-suited for low-latency networks with medium bandwidth and it is especially beneficial for computationally weak devices; Σ_{EIG} runs fast in any network types for small values of τ.

6 Conclusion

In this work, we designed, implemented, and evaluated multiple practically efficient protocols for PPLP using STPC and AHPKE. Moreover, we introduced optimizations for our protocols: using Bloom filter [Blo70] for our AHPKE-based protocols and using Conditional Swap [KS08] for our Boolean circuit-based protocols. We made extensive use of the pre-computation for computationally heavy parts of our protocols in the cases where the same parties perform PPLP several times, which substantially improves performance. Finally, we evaluated our most efficient protocols in a real-world mobile Internet setting and showed practical total runtimes of below 200 ms and online runtimes of below 50 ms. We leave implementation of our protocols on mobile devices as future work.

[9] In the near future, today's 4G mobile networks will be replaced by 5G, which will significantly reduce the average network latency (average expected latency in 5G networks is around 1 ms [JWEH15]). Therefore, in low-latency 5G networks $\mathsf{ABY}_{\mathsf{AY}}$ will potentially be most efficient (see §5.2).

Acknowledgements. We thank Per Hallgren for providing the raw data of his benchmarks for comparison. This work has been co-funded by the DFG as part of project E4 within the CRC 1119 CROSSING, and by the German Federal Ministry of Education and Research (BMBF) and the Hessen State Ministry for Higher Education, Research and the Arts (HMWK) within CRISP. This work has been also co-funded by the INSURE project (303578) of Academy of Finland.

References

AD01. Mikhail J. Atallah and Wenliang Du. Secure multi-party computational geometry. In *WADS*, 2001.

ALSZ13. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *CCS*, pages 535–548. ACM, 2013.

Blo70. Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 1970.

BPP00. Joan Boyar, René Peralta, and Denis Pochuev. On the multiplicative complexity of boolean functions over the basis $(\wedge, \oplus, 1)$. *TCS*, 2000.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, 1993.

CZBP06. Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. Preserving user location privacy in mobile data management infrastructures. In *PETS*, 2006.

DGK09. Ivan Damgard, Martin Geisler, and Mikkel Kroigard. A correction to 'efficient and secure comparison for on-line auctions'. *IJACT*, 2009.

DSZ15. Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

EFG$^+$09. Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *PETS*, 2009.

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, 1984.

Fin03. Steven R Finch. *Mathematical constants*, volume 93. Cambridge University Press, 2003.

FVM$^+$10. Dario Freni, Carmen Ruiz Vicente, Sergio Mascetti, Claudio Bettini, and Christian S. Jensen. Preserving location and absence privacy in geo-social networks. In *CIKM*, 2010.

Gam85. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 1985.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC*, 1987.

Hal17. Per A Hallgren. *Robust location privacy*. PhD thesis, Chalmers University of Technology, 2017. `http://www.cse.chalmers.se/research/group/security/pages/publications/perh-phd/phd-thesis.pdf`.

HOS15. Per A. Hallgren, Martín Ochoa, and Andrei Sabelfeld. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *PST*, 2015.

HOS17. Per A. Hallgren, Claudio Orlandi, and Andrei Sabelfeld. Privatepool: Privacy-preserving ridesharing. In *CSF*, 2017.

JWEH15. N. A. Johansson, Y.-P. E. Wang, E. Eriksson, and M. Hessler. Radio access for ultra-reliable and low-latency 5G communications. In *ICC Workshop*, 2015.

KS08. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP*, 2008.

KSS09. Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS*, 2009.

LRQ$^+$17. Mengyuan Li, Na Ruan, Qiyang Qian, Haojin Zhu, Xiaohui Liang, and Le Yu. SPFM: scalable and privacy-preserving friend matching in mobile cloud. *IEEE IoT Journal*, 2017.

MFB$^+$11. Sergio Mascetti, Dario Freni, Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *VLDB Journal*, 2011.

MVOV96. Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography.* CRC press, 1996.

NP01. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457. ACM/SIAM, 2001.

NTL$^+$11. Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

PH78. Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (corresp.). *IEEE Trans. Information Theory*, 1978.

PPR05. Anna Pagh, Rasmus Pagh, and S. Srinivasa Rao. An optimal bloom filter replacement. In *SODA*, 2005.

ŠG14. Jaroslav Šeděnka and Paolo Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In *ASIACCS*, 2014.

ŠTŠ$^+$09. Laurynas Šikšnys, Jeppe R. Thomsen, Simonas Šaltenis, Man Lung Yiu, and Ove Andersen. A location privacy aware friend locator. In *SSTD*, 2009.

STSY10. Laurynas Siksnys, Jeppe Rishede Thomsen, Simonas Saltenis, and Man Lung Yiu. Private and flexible proximity detection in mobile social networks. In *MDM*, 2010.

Yao82. A. C. Yao. Protocols for secure computations. In *SFCS*, 1982.

YLC$^+$17. Wenbin Yu, Zhe Liu, Cailian Chen, Bo Yang, and Xinping Guan. Privacy-preserving design for emergency response scheduling system in medical social networks. *Peer-to-Peer Networking and Applications*, 2017.

ZGH07. Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In *PETS*, 2007.

ZLLH17. Yao Zheng, Ming Li, Wenjing Lou, and Y. Thomas Hou. Location based handshake and private proximity test with location tags. *IEEE Dependable and Secure Computing*, 2017.

ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In *EUROCRYPT*, 2015.

ZWL$^+$17. Hui Zhu, Fengwei Wang, Rongxing Lu, Fen Liu, Gang Fu, and Hui Li. Efficient and privacy-preserving proximity detection schemes for social applications. *IEEE IoT Journal*, 2017.

A Our Attack on Zhu et al.'s Protocols

A.1 Protocol Review

We briefly review AGRQ-P and AGRQ-C [ZWL$^+$17] which support both polygon and circle range queries, respectively, for PPLP. In the following protocol description, we ignore the social application server (SS) and the message authentication code (and related protocol steps) that are irrelevant for our attacks. For simplicity, we here mainly focus on the messages which are generated based on the location coordinates of the sender QU and the responder UF. Meanwhile, QU chooses two large primes p and α with bit sizes κ_1 and κ_2 respectively, and a random secret $s \in \mathbb{Z}_p$. We respectively describe the primary protocol steps of each protocol as follows.

AGRQ-P. The sender QU first chooses a set of vertexes forming a convex polygon that comprises of m edges. The vertexes are denoted by a vector of m coordinates, i.e. $\langle(x_{q_1}, y_{q_2}), (x_{q_2}, y_{q_2}), \ldots, (x_{q_m}, y_{q_m})\rangle$.

We can define an edge of the polygon with two adjacent vertexes. For $1 \leq i \leq m$ and $i' := i+1 \mod m$, QU computes $C_{i_1} = s \cdot (x_{q_i} \cdot \alpha + c_{i_j}) \bmod p$, $C_{i_2} = s \cdot (y_{q_i} \cdot \alpha + c_{i_2}) \bmod p$, $C_{i_3} = s \cdot (x_{q_{i'}} \cdot \alpha + c_{i_3}) \bmod p$, $C_{i_4} = s \cdot (y_{q_{i'}} \cdot \alpha + c_{i_4}) \bmod p$, $C_{i_5} = s \cdot (x_{q_i} \cdot y_{q_{i'}} \cdot \alpha + c_{i_5}) \bmod p$, and $C_{i_6} = s \cdot (x_{q_{i'}} \cdot y_{q_i} \cdot \alpha + c_{i_6}) \bmod p$, where c_{i_n} are random numbers with bit size κ_3 and $1 \leq n \leq 6$. Next, QU sets $C_i = C_{i_1}||C_{i_2}||C_{i_3}||C_{i_4}||C_{i_5}||C_{i_6}$, and $C = C_1||C_2||\ldots||C_m$. Then, QU sends the message $\alpha||p||C||QU$ to the receiver UF_j via SS. The secret value $s^{-1} \pmod p$ is kept privately.

Upon receiving $\alpha||p||C||QU$, the receiver UF_j who holds a location (x_j, y_j) computes $D_{i_1} = r_i \cdot \alpha(x_j \cdot C_{i_4} + y_j \cdot C_{i_1} + C_{i_6}) \bmod p$ and $D_{i_2} = r_i \cdot \alpha(x_j \cdot C_{i_2} + y_j \cdot C_{i_3} + C_{i_5}) \bmod p$, where $1 \leq i \leq m$ and each $r_i \in \{0,1\}^{\kappa_4}$ is chosen randomly by UF_j with a bit size κ_4. Next, UF_j sets $D_i = D_{i_1}||D_{i_2}$, and $D = D_1||D_2||\ldots||D_m$. Then, UF_j sends the message $D||UF_j$ to the sender QU via SS.

Upon receiving $D||UF_j$, QU computes the following:

$$\begin{aligned}
E_{i_1} &= s^{-1} \cdot D_{i_1} \bmod p \\
E'_{i_1} &= \frac{E_{i_1} - (E_{i_1} \bmod \alpha^2)}{\alpha^2} \\
&= r_i \cdot (x_j \cdot y_{q_{i'}} + y_j \cdot x_{q_i} + x_{q_{i'}} \cdot y_{q_i})
\end{aligned} \tag{1}$$

$$\begin{aligned}
E_{i_2} &= s^{-1} \cdot D_{i_2} \bmod p \\
E'_{i_2} &= \frac{E_{i_1} - (E_{i_1} \bmod \alpha^2)}{\alpha^2} \\
&= r_i \cdot (x_j \cdot y_{q_i} + y_j \cdot x_{q_{i'}} + x_{q_i} \cdot y_{q_{i'}})
\end{aligned} \tag{2}$$

$$\begin{aligned}
E_i &= E'_{i_2} - E'_{i_1} \\
&= r_i[(x_j \cdot y_{q_i}) + y_j \cdot x_{q_{i'}} + x_{q_i} \cdot y_{q_{i'}}) - \\
&\quad (x_j \cdot y_{q_{i'}} + y_j \cdot x_{q_i} + x_{q_{i'}} \cdot y_{q_i})],
\end{aligned}$$

where $1 \leq i \leq m$.

If there is one E_i such that $E_i < 0$ (for $1 \leq i \leq m$), then QU knows that UF_j is outside the polygon. Otherwise, UF_j is within the polygon. As E_i is masked by a random value r_i which is chosen by UF_j, E_i does not reveal the location of UF_j.

AGRQ-C. QU chooses a location (x_q, y_q) as the center of a circle which has radius r. QU computes ciphtertexts $C_1 = s \cdot (x_q \cdot \alpha + c_1) \bmod p$, $C_2 = s \cdot (y_q \cdot \alpha + c_2) \bmod p$, $C_3 = s \cdot c_3 \bmod p$, and $C_4 = s \cdot c_4 \bmod p$, and $A = x_q^2 + y_q^2 - r^2$. Then, QU sets $C = C_1||C_2||C_3||C_4$, and sends $\alpha||p||A||C||QU$ to UF_j via SS.

Upon receiving $\alpha||p||A||C||QU$, the receiver UF_j having location (x_j, y_j) would compute $D_1 = x_j \cdot \alpha C_1 \bmod p$, $D_2 = y_j \cdot \alpha \cdot C_2$, $D_3 = r_3 \cdot C_3 \bmod p$, $D_4 = r_4 \cdot C_4 \bmod p$, $D = r_5 \cdot \sum_{i=1}^{i=4} D_i$, and $B = r_5 \cdot (x_j^2 + y_j^2 + A)$, where r_3, r_4 and r_5 are chosen randomly by UF_j. Next, UF_j sends the message $D||UF_j$ to the sender QU via SS.

Upon receiving $D||UF_j$, QU computes the following:

$$\begin{aligned}
E &= s^{-1} \cdot D \bmod p \\
&= s^{-1} \cdot s \cdot r_5 \cdot [\alpha^2 \cdot (x_q \cdot x_j + y_q \cdot y_j) + \\
&\quad \alpha \cdot (x_j \cdot c_1 + y_j \cdot c_2) r_3 \cdot c_3 + r_4 \cdot c_4] \bmod p \\
E' &= \frac{E - (E \bmod \alpha^2)}{\alpha^2} = r_5(x_q \cdot x_j + y_q \cdot y_j) \\
R &= B - 2E' = r_5[(x_j - x_q)^2 + (y_j - y_q)^2 - r^2].
\end{aligned}$$

If $R \leq 0$, then QU knows that UF_j is within the circle. Otherwise, UF_j is outside the circle. The privacy of UF_j's location cannot be compromised from R because of the random value r_5 that is freshly chosen by UF_j.

A.2 On the insecurity of AGRQ-P and AGRQ-C

In our attacks, we assume that the sender QU honestly follows the protocol execution but he is curious about the receiver UF_j's location (x_j, y_j), i.e., we study the protocols under the ‘honest-but-curious’ attack setting.

An equation solving attack against AGRQ-P. This attack is based on the observation that the randomness r_i used in Equation 1 and Equation 2 can be deleted by dividing them. Namely, we can have a new equation:

$$T_i = \frac{E_{i_{1'}}}{E_{i_{2'}}} = \frac{x_j \cdot y_{q_{i'}} + y_j \cdot x_{q_i} + x_{q_{i'}} \cdot y_{q_i}}{x_j \cdot y_{q_i} + y_j \cdot x_{q_{i'}} + x_{q_i} \cdot y_{q_{i'}}}. \tag{3}$$

From Equation 3, we can have a closed formulas for x_j and y_j as:

$$x_j = \frac{y_j \cdot (x_{q_i} - T_i \cdot x_{q_{i'}}) + x_{q_{i'}} \cdot y_{q_i} - T_i \cdot x_{q_i} \cdot y_{q_{i'}}}{T_i \cdot y_{q_i} - y_{q_{i'}}}. \tag{4}$$

Recall that x_{q_i}, y_{q_i}, $x_{q_{i'}}$ and $y_{q_{i'}}$ are all chosen by QU. Then T_i is just an equation with two target unknown variables x_j and y_j. Thus, two independent instances of Equation 3 (e.g. T_1 and T_2) are both necessary and sufficient to recover x_j and y_j based on Equation 4. Because there are m edges in a polygon (s.t. $m \geq 3$), the required two equations can be easily obtained from different $E_{i_{1'}}$ and $E_{i_{2'}}$ within just one query.

An equation solving attack against AGRQ-C. We can design a similar attack against AGRQ-C. However, now there are two equations (for generating B and E) involving the target location (x_j, y_j) of UF_j. The first attack step is to extract a new equation from E. This can be done by computing $E'' = \frac{[E-(E \bmod \alpha)] \bmod \alpha^2}{\alpha} = r_5 \cdot (c_1 \cdot x_j + c_2 \cdot y_j)$. Since x_q and y_q are chosen independently from c_1 and c_2, E' and E'' are also independent. We can compute $T_1 = \frac{E'}{E''} = \frac{x_q \cdot x_j + y_q \cdot y_j}{c_1 \cdot x_j + c_2 \cdot y_j}$ and obtain a linear relation between x_j and y_j, i.e.,

$$x_j = \frac{y_j \cdot (c_2 \cdot E'' - E' \cdot y_q)}{E' \cdot x_q - c_1 \cdot E'}. \tag{5}$$

Next, we can obtain another equation:

$$F_2 = \frac{E''}{B} = \frac{c_1 \cdot x_j + c_2 \cdot y_j}{x_j^2 + y_j^2 + A}. \tag{6}$$

Eventually, we take Equation 5 into Equation 6 and solve the resulting quadratic equation to obtain y_j. Finally, we calculate x_j based on Equation 5.