(iii) Client should not tamper with her location data or intermediate results to affect the final result.

The above problem can be formalized as a secure two-party computation problem: Two parties, Client and Server, each with an integer b and an integer interval $[a, e]$ respectively, want to evaluate whether $b \in [a, e]$, such that: (i) Client can not learn the values a, e; (ii) Server learns no information about value b; and (iii) Client should not change b or intermediate results to affect the final result. As formerly stated, we call the problem "private interval check (PIC)".

In the paper, we construct an efficient secure two-party protocol to treat PIC. Obviously, PIC is a special case of Private Set Intersection Cardinality problem (PSI-CA) [6–8], where two parties want to compute the intersection cardinality of their input sets without revealing their input set to the other one. There is then a direct question of why not to use existing PSI-CA protocols to solve PIC problem since, after all, there are many excellent protocols for PSI-CA [6–11]. One major reason is the inefficiency of these protocols to treat PIC, where an integer interval is treated simply as a set of all integers in the integer interval. However, integer interval is a consecutive integer set, which is redundant, and it would be inefficient to simply treat it without further treatment, such as compression. The other reason is that most of these protocols are based on the semi-honest model, where both parties are assumed to act according to their prescribed action in the protocol, which is unrealistic in our application scenario. A protocol is then needed to treat PIC efficiently in the malicious model.

Our major contribution is a private interval check protocol in the malicious model, where interval is not confined to an integer interval but a union of integer intervals. The major challenge is to first "compress" intervals and then prove the knowledge about it. To the best of our knowledge, ours is the first to solve the private interval check problem in the malicious model with constant round and $O(\log^2 n)$ communication and exponentiations for the integer interval length n. The core of our protocol is the using of k-ary tree index and the accompanying zero-knowledge proof techniques. A simulation-based proof is given to the private interval check protocol. We also analyse our protocol's complexity and give it an asymptotic optimization. Furthermore, a special case of our protocol is a secure integer comparison protocol [12–16] in the malicious model.

Besides location-based access control, there are other applications for our protocol, such as electronic election, credential systems, electronic auction, etc.

1.1 Related Works

Our work is related to Private Set Intersection (PSI) [6, 11, 9, 10] and Private Set Intersection Cardinality (PSI-CA) [7, 8, 6]. PSI have been widely looked at by researchers in the last few years due to their potential applications. PSI-CA is a special case of PSI where the output is the cardinality of intersection set but not the intersection. PIC can be seen as a special case of PSI-CA where the input sets are a set containing one integer and a set containing a series of consecutive integers, and the output is either 0 or 1. However, our work is different from the above works. The main difference of our work from PSI and

PSI-CA is that our work focuses on the special case of set, i.e., integer interval or union of integer intervals, which can be compressed to improve the efficiency of protocols. Moreover, the former PSI-CA protocols [7, 8] are not adapted to our application scenarios since their protocols are in the semi-honest model which is unrealistic to our application scenarios.

Our work is also related to the secure integer comparison – the well-known "Millionaires' Problem" [12–16]. Due to its significant applications, such as electronic auction [15], secure interval check [14], location-based access control [17, 18], and proximity test [19–25], it has been studied extensively.

However, our protocol is different from these secure integer comparison protocols; almost all secure integer comparison protocols are based on the semi-honest assumption where each party follows the protocol with the exception that it keeps all its intermediate computations, which is not practical in our application scenarios. Our protocol don't rely on the semi-honest model but assumes the existence of malicious parties which makes our schemes more practical than the formers.

There are other related works in the privacy protection of location-based access control [4, 5, 17, 2] and in location-based social networks [19]. However, these works either assume the honest Server or are based on the semi-honest assumption [4, 5, 17, 19] that are different from ours.

2 Preliminaries

2.1 Basic Notations

For a real number a, $\lfloor a \rfloor$ denotes the largest integer $\leq a$. The notation $m \in_R S$ denotes that m is randomly chosen from a set S. The notation $[x, y]$, for two integers x and y, denotes an interval of integers, i.e., $[x, y] = \{z \in \mathbb{Z} : x \leq z \leq y\}$. We say that m intervals $[a_1, e_1], \ldots, [a_m, e_m]$ are pairwise non-adjacent if $e_i + 1 < a_{i+1}$ for $1 \leq i \leq m - 1$.

2.2 Hardness Assumptions

Our construction relies on the following two hardness assumptions.

Definition 1 (Decisional Deffie-Hellman (DDH) assumption). *Let $\mathbb{G}$ be a group with order q and let $g \in_R \mathbb{G}$, $x, y, z \in_R \mathbb{Z}_q$. The Decisional Deffie-Hellman assumption for $\mathbb{G}$ is that it is infeasible to distinguish (g, g^x, g^y, g^z) from (g, g^x, g^y, g^{xy}).*

Definition 2 (Discrete Logarithm (DL) assumption). *Let $\mathbb{G}$ be a group with order q and let $g, h \in_R \mathbb{G}$. The Discrete Logarithm assumption is that it is infeasible to find $x \in_R \mathbb{Z}_q$ such that $h = g^x$.*

2.3 Commitment Scheme

In this paper, we use Pedersen commitment scheme [26]. Let $q' = 2q + 1$ where q, q' are primes. Set $\mathbb{G}$ to be the unique subgroup of $\mathbb{Z}_{q'}^*$ of order q. Let $\tilde{g}, \tilde{h}$ be two random generators of $\mathbb{G}$; then a commitment to $m \in \mathbb{Z}_q$ is defined as $\mathrm{com}(m, r) = \tilde{g}^m \tilde{h}^r$ where $r \in_R \mathbb{Z}_q$. Commitment to a set is defined as follow.

Definition 3 (Commitment to a set). *Let $B = \{n_0, \ldots, n_t\}$ be a subset of $\mathbb{Z}_q$. Set, for $0 \leq i \leq t$, $c_i = com(n_i, r_{n_i})$ where $r_{n_i} \in \mathbb{Z}_q$. Then the set $C = \{c_0, \ldots, c_t\}$ is said to be* a commitment to *B.*

The commitment scheme is computationally binding and perfectly hiding.

2.4 Homomorphic Encryption Scheme

We use a standard variant of ElGamal encryption scheme [27]. We use the same group $\mathbb{G}$ used for the Pedersen commitment scheme. Let g, h be two random generators in $\mathbb{G}$; then the public key is (g, h) and the secret key is $k = \log_g h$. To encrypt a message $m \in \mathbb{Z}_q$, choose $y \in_R \mathbb{Z}_q$, and compute $E(m) = (g^y, h^y \times g^m)$. To decrypt (α, β), first compute β/α^x, then search a lookup table of (g^m, m) to find m. It is easy to find that the decryption scheme is very inefficient. Fortunately, we don't need any decryption but only need verification of whether $g^m = 1$. Note that $g^m = 1$ if and only if $m = 0$.

It is easy to verify that the modified ElGamal scheme is *additively homomorphic*, i.e., $E(m_1 + m_2) = E(m_1)E(m_2)$. It follows by repeated addition that $E(em) = E(m)^e$ for an integer e.

2.5 Secure Two-Party Computation

A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs. Let f be a two-party functionality, i.e. $f = (f_1, f_2) : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^*$, and let π be a two-party protocol for computing f. The security of π in the malicious model [28, §7.2] is defined as follow.

Definition 4. *Let f and π be as above. Protocol π is said to securely compute f with abort in the presence of malicious adversaries if for every probabilistic polynomial-time adversary $\mathcal{A}$ for the real model, there exists a probabilistic polynomial-time (p.p.t.) adversary $\mathcal{S}$ for the ideal model, such that for every $I \subset \{1, 2\}$,*

$$\{\mathrm{IDEAL}_{f,\mathcal{S}(z),I}(x, y, n)\}_{x,y,z\in\{0,1\}^*,n\in\mathbb{N}} \overset{c}{\equiv} \{\mathrm{REAL}_{\pi,\mathcal{A}(z),I}(x, y, n)\}_{x,y,z\in\{0,1\}^*,n\in\mathbb{N}},$$

where $I \subset \{1, 2\}$ denotes the set of corrupted parties, (x, y) denote the inputs of π, z is the auxiliary input to $\mathcal{A}$, and n is the security parameter.

2.6 Zero-Knowledge Proofs

For a Proof of Knowledge (PK) we use the definition from [29].

Definition 5. *A two-party protocol (P, V) is a proof of knowledge (PK) for a relation $R = \{(x, w)\} \subset \{0,1\}^* \times \{0,1\}^*$ with knowledge error $0 \leq \kappa \leq 1$ if the following properties are satisfied:*

- Completeness: *If P and V follow the protocol on input x and private input w to P where $(x, w) \in R$, then V always accepts.*
- Knowledge soundness (or validity): *If a cheating prover P^* has probability ϵ of convincing V to accept x, there exists an expected polynomial-time algorithm E, called the knowledge extractor, such that when given rewindable black-box access to P^*, E outputs a witness w for x with probability $\epsilon - \kappa$.*

The proof system (P, V) is zero-knowledge if there exists a p.p.t. algorithm $\mathcal{S}$, called the simulator, such that for any $(x, w) \in R$ and any p.p.t. algorithm V^*, the outputs of $V^*(x)$ after interacting with $P(w)$ and that of $\mathcal{S}(w)$ are computationally indistinguishable.

When presenting protocols we express Zero Knowledge (ZK) proofs using the notation introduced by Camenisch and Stadler [30]:

$$PK\{(\alpha, \beta, \ldots) : \text{statements involving } \alpha, \beta, \ldots\}$$

means the prover is proving knowledge of $(\alpha, \beta, \ldots)$ such that these values satisfy statements.

2.7 Perfect Binary Index

The paper [14] presents perfect binary index. We summarize their definitions as follows.

Definition 6 (Tree node). *A* tree node *is a data structure that consists of a unique label (h, o) and possibly two pointers* left *and* right *to other tree nodes. The label has two components: the height h and the order o. A* leaf node *is a tree node with a label $(0, o)$ and no pointers.*

Definition 7 (ℓ Perfect Binary Tree (ℓ PBT)). *An ℓ* perfect binary tree *is a binary tree with a set of leaf nodes $L = \{(0,0), (0,1), \ldots, (0, 2^\ell - 1)\}$ and a set of non-leaf nodes NL such that $(\ell, 0) \in NL$ and for all $(h, o) \in NL$, there exists $(h-1, 2o), (h-1, 2o+1) \in (L \cup NL)$ with $(h, o).left = (h-1, 2o), (h, o).right = (h-1, 2o+1)$. In an ℓ PBT, the root node is the node $(\ell, 0)$.*

Figure 1 shows a 3 PBT.

Definition 8 (Coverage). *Given an ℓ PBT, we say a tree node (h_1, o_1) covers a leaf node $(0, o_2)$ if there exist a path from (h_1, o_1) to $(0, o_2)$ in the tree (e.g., if $o_1 \times 2^{h_1} \leq o_2 < (o_1+1) \times 2^{h_1}$). The* covering set *of a given leaf node v is the set of*

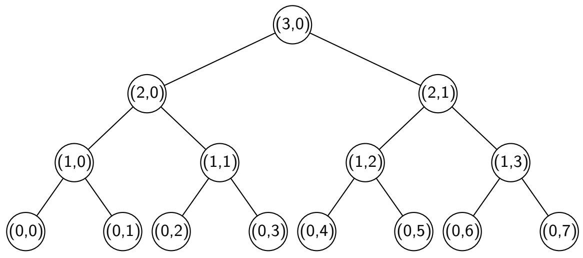

Fig. 1. 3 Perfect Binary Tree

all nodes in the PBT that cover v (e.g., all nodes on the path from v to the root). The coverage *of a tree node $v = (h_1, o_1)$ is the set of all leaf nodes covered by v. The pointer $v.leftLeaf$ [$v.rightLeaf$] returns the left [right] most leaf node in the coverage of v ($v.leftLeaf = (0, o_1 \times 2^{h_1})$, $v.rightLeaf = (0, (o_1 + 1) \times 2^{h_1} - 1)$).*

Definition 9 (Representer set and minimal representer set). *Given an ℓ PBT, a* representer (set) *of a set of leaf nodes $L' \subseteq L$ is a set of nodes $R \subseteq (L \cup NL)$ such that*

- *for all nodes $v \in L'$, there exists a node in R that covers v, and*
- *for all nodes $v \in R$, there is no leaf node $v' \notin L'$ that is covered by v.*

A representer R for the set of leaf nodes L' is minimal, if there is no other representer R' of L' with $|R'| < |R|$.

2.8 Perfect *k*-ary Index

In this paper, we use a generalized notion of the perfect binary index in Section 2.7, i.e., perfect k-ary index. The generalization is direct. We only give the details of the notion of ℓ perfect k-ary tree (ℓ PKT). Other related notions can be treated similarly.

Definition 10 (ℓ Perfect k-ary tree (ℓ PKT)). *An ℓ* perfect k-ary tree *($k \geq 2$) is a k-ary tree with a set of leaf nodes $L = \{(0,0), (0,1), \ldots, (0, k^{\ell} - 1)\}$ and a set of non-leaf nodes NL such that $(\ell, 0) \in NL$ and for all $(h, o) \in NL$, there exists $(h-1, ko), (h-1, ko+1), \ldots, (h-1, ko+k-1) \in (L \cup NL)$ with $(h, o).c_1 = (h-1, ko)$,..., $(h, o).c_k = (h-1, ko+k-1)$. In an ℓ PKT, the root node is the node $(\ell, 0)$.*

Figure 2 shows a 2 Perfect 3-ary Tree(P3T).

Lemma 1 ([14](Lemma 6)). *Let R be a minimal representer of a set of leaf nodes. For each node $n \in R$, no other node $n' \in R$ is a descendant of n.*

Lemma 2 ([14](Lemma 3)). *Let R be a minimal representer for $\{(0,0), ..., (0,e)\}$ in an ℓ PKT. For each level $0 \leq i \leq \ell$, there can be at most $k-1$ nodes $v \in R$ such that $v.h = i$.*

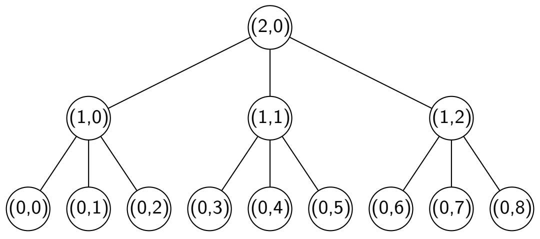

Fig. 2. 2 Perfect 3-ary Tree

Lemma 3 ([14](Lemma 5)). *Given an ℓ PKT, let R be a minimal representer for the set of leaf nodes $\{(0, a), \ldots, (0, e)\}$ and let B be the covering set for the leaf node $(0, b)$. Then $a \leq b \leq e$ if and only if $|R \cap B| = 1$.*

The above three lemmas are the generalizations of the lemmas in [14]. The proof of these three lemmas is similar with those in [14] and we omit them.

Corollary 1. *Let $L_i = [a_i, e_i]$, for $1 \leq i \leq m$. Let R_i, for $1 \leq i \leq m$, be the minimal representer set of L_i and let R be the minimal representer set of $\cup_{i=1}^{m} L_i$ in an ℓ PKT. If L_i are pairwise non-adjacent, i.e., $e_i + 1 < a_{i+1}$ for $1 \leq i \leq m - 1$, then $R = \cup_{i=1}^{m} R_i$.*

Proof. Since L_i, for $1 \leq i \leq m$, are pairwise non-adjacent, we get $R_i \cup R_j = \emptyset$ for different i, j from Definition 9. Then we have $R = \cup_{i=1}^{m} R_i$.

Corollary 2. *Let L_i, R be as in Corollary 1 and B be the covering set of b in the ℓ PKT. Then $b \in \cup_{i=1}^{m} L_i$ if and only if $|R \cap B| = 1$.*

The notion of PKT gives a bijection between a set of integers and a set of leaf nodes in the PBT, such as the integer interval $[a, e]$ is injected to the set $\{(0, a), (0, a + 1) \ldots, (0, e)\}$. In the following sections, we will use the notations $[a, e]$ to denote either the integer interval $[a, e]$ or the set of leaf nodes $\{(0, a), (0, a + 1) \ldots, (0, e)\}$ if without ambiguity.

We define a bijection $g(h, o) = \sum_{j=0}^{h-1} k^{\ell-j} + o$ between the set of all nodes in an ℓ PKT and the integer interval $[0, (k^{\ell+1} - 1)/(k - 1)]$. By using the bijection g each node (h, o) in an ℓ PKT is identical to a unique integer $g(h, o)$. In the following part of the paper, while mentioning a node, it would relate either to (h, o) or to its integral form $g(h, o)$. We would interchangeably use them without ambiguity.

3 Commitment to the Covering Set of an Integer

In this section, we give a protocol to prove the knowledge of commitment to the covering set of an integer. Specifically, setting B be the covering set of an integer $b \in [0, k^{\ell}]$ in an ℓ PKT, Client can convince Server, who only knows a commitment to b, that a set is a commitment to B. We first give a result for B.

Lemma 4. *Let b, B, k, ℓ be as above. Then $B = \{(0, b), (1, \lfloor b/k \rfloor), (2, \lfloor b/k^2 \rfloor), \dots, (\ell, \lfloor b/k^\ell \rfloor)\}$.*

Proof. By Definition 10, the coverage of a node (h, o) is $\{(0, k^h \times o), \dots, (0, k^h \times (o+1) - 1)\}$. Assuming that (i, j) is the node that covers $(0, b)$ in the level i, $0 \le i \le \ell$, then $k^i \times j \le b \le k^i \times (j+1) - 1$. We have $b/k^i - 1 + 1/k \le j \le b/k^i$ and so $j = \lfloor b/k^i \rfloor$. The proof is complete.

Let B, b be as in Lemma 4, and let C, c be commitments to B, b respectively. We now show a two-party protocol by which Client can prove to Server, where Server only knows c, that C is a commitment to B without revealing b and B.

Protocol 1
- Common input: *A string c and a set $\{c_0, \dots, c_\ell\}$, where $c, c_0, \dots, c_\ell \in \mathbb{G}$.*
- Auxiliary input for Client: *A tuple (b, r) such that $c = com(b, r)$, where $b, r \in \mathbb{Z}_q$.*
- Initial message: *Client computes*
  1. *$c_{b_i} = com(b_i, r_{b_i})$, for $i = 0, \dots, \ell - 1$, where $r_{b_i} \in_R \mathbb{Z}_q$ and b_i is the ith digit (from right to left) in the k-ary notation of b.*
  2. *$e_0 = b$ and $c_{e_0} = c_b$.*
  3. *for $1 \le i \le \ell$, $e_i = (e_{i-1} - b_{i-1})/k$ and $c_{e_i} = com(e_i, r_{e_i})$, where $r_{e_i} \in_R \mathbb{Z}_q$.*
- The protocol:
  1. *Client sends $\{c_{b_0}, \dots, c_{b_{\ell-1}}\}$ and $\{c_{e_0}, \dots, c_{e_\ell}\}$ to Server.*
  2. *For each $0 \le i \le \ell - 1$, Client proves to Server that c_{b_i} commits to the ith digit b_i in the k-ary notation of b, i.e.,*

     $$PK\{(b_i, r_i) : c_{b_i} = g^{b_i} h^{r_i} \wedge b_i \in [0, k-1]\},\ for\ 0 \le i \le \ell - 1$$

     and

     $$PK\{(\theta) : c / \prod_{i=0}^{\ell-1} c_{b_i}^{k^i} = h^\theta\}.$$
  3. *For $1 \le i \le \ell$, Client proves to Server that $e_i = (e_{i-1} - b_{i-1})/k$, i.e.,*

     $$PK\{(\delta_i) : c_{e_{i-1}} / (c_{b_{i-1}} c_{e_i}^k) = h^{\delta_i}\},\ for\ 1 \le i \le \ell.$$
  4. *Client proves to Server that $\{c_0, \dots, c_\ell\}$ is a set of commitments of B, i.e., for $0 \le i \le \ell$,*

     $$PK\{(\tau_0, \tau_1, \dots, \tau_\ell) : c_0/(c_{e_i} g^{f_i}) = h^{\tau_0} \vee c_1/(c_{e_i} g^{f_i}) = h^{\tau_1} \vee \dots \vee c_\ell/(c_{e_i} g^{f_i}) = h^{\tau_\ell}\},$$

     where $f_i = \sum_{j=0}^{i-1} k^{\ell - j}$.

Note that $\{c_0, \dots, c_\ell\}$ is a shuffle of a commitment to B, i.e., each c_i committing to which node in B is unknown to Server.

Remark 1. In Step 2, the first ℓ PKs can be achieved using the range proof techniques in [31] and the last PK is just a proof of knowledge of discrete logarithm. The PK in Step 4 can be evaluated using the batch proof techniques in [32].

Remark 2. The complexity of Protocol 1: The initial message step needs 4ℓ exponentiations in $\mathbb{G}$; In Step 2, the first ℓ PKs needs $O(\ell \log k)$ exponentiations in $\mathbb{G}$ using the range proof protocol in [31] and the last PK needs $\ell+3$ exponentiations in $\mathbb{G}$; Step 3 needs 4ℓ exponentiations in $\mathbb{G}$; Step 4 needs $O(\ell^2)$ exponentiations in $\mathbb{G}$ [33, 32]. The total computation is $O(\ell \times \max(\ell, \log k))$ exponentiations in $\mathbb{G}$. The total communication is $O(\ell \times \max(\ell, \log k)|\mathbb{G}|)$ bits.

Theorem 1. *Protocol 1, when constructed with a secure ZK protocol, is a zero-knowledge proof for the assertion that $\{c_0, \dots, c_\ell\}$ is a shuffle of a commitment to B.*

Proof. Step 2 proves that c_{b_i} commits to the ith digit in the k-ary notation of b. Step 3 proves that c_{e_i} commits to $\lfloor b/k^i \rfloor$. Step 4 illustrates that $\{c_0, \dots, c_\ell\}$ and $\{c_{e_0} g^{f_0}, \dots, c_{e_\ell} g^{f_\ell}\}$ commit to the same set, where $c_{e_i} g^{f_i}$ commits to the node $(i, \lfloor b/k^i \rfloor)$ in B. In all, completeness and soundness are complete.

Zero-knowledge property follows from the fact of perfectly hiding property of the commitment scheme and of zero-knowledge property of each ZK protocol in Protocol 1.

4 Private Interval Check Protocol

We now consider the private interval check problem. We first give the problem a formal definition and then present a high level description of our protocol. Finally, we present the detailed protocol.

4.1 Definitions and Overview

We consider a more general case of the private interval check problem where the interval may not be an interval but a union of intervals. Let $L_i = [a_i, e_i]$, for $1 \le i \le m$, be m pairwise non-adjacent intervals, and let R_i be the minimal representer set of L_i in an ℓ PKT. Let R be the minimal representer set of $L = \cup_{i=1}^{m} L_i$. Let b be a nonnegative integer $\le k^\ell - 1$ and B be its covering set in the ℓ PKT.

Definition 11 (Private Interval Check of Committed Integer). *Let L, b be as above. Set $c = com(b, r)$ be a commitment of b. The function of private interval check of committed integer $\mathcal{F}_{\mathcal{PIC}}$ is defined as*

$$((b, c), (L, c)) \mapsto (\lambda, |\{b\} \cap L|).$$

Definition 12. *Let Π be a protocol to compute $\mathcal{F}_{\mathcal{PIC}}$. Π is said to securely compute $\mathcal{F}_{\mathcal{PIC}}$ with abort in the presence of malicious adversaries if for every* *probabilistic polynomial-time adversary $\mathcal{A}$ for the real model, there exists a probabilistic polynomial-time adversary $\mathcal{S}$ for the ideal model, such that for every $I \subset \{Client, Server\}$,*

$$\{\mathrm{IDEAL}_{\mathcal{F_{PIC}},\mathcal{S}(z),I}((b,c),(L,c))\} \overset{c}{\equiv} \{\mathrm{REAL}_{\Pi,\mathcal{A}(z),I}((b,c),(L,c))\}.$$

Suppose that Client has (b,c) and that Server has $(L_1,\ldots,L_m,c)$. By Corollary 2, $b \in L$ if and only if $|R \cap B| = 1$. Therefore, we only need to check whether $|R \cap B| = 1$ in order to check whether $b \in L$. Our protocol functions as follows.

– Client computes a commitment C to B, and proves the knowledge of C to Server using protocol 1.
– Server evaluates the minimal representer R of $L_1,\ldots,L_m$ in the ℓ PKT and computes $f(x) = \prod_{a_i \in R}(x - a_i) = \sum_{i=0}^{v} \alpha_i x^i$ where $v = |R|$.
– Server encrypts α_i, for $i = 0,\ldots,v$, using its ElGamal encryption function $E(\cdot)$ and prove to Client that $E(\alpha_i)$ is a correct encryption.
– For each $n_i \in B$, Client evaluates $w_i = E(s_i f(n_i))$ where $s_i \in \mathbb{Z}_q/\{0\}$ and proves to Server that it is evaluated correctly. Note that $n_i \in B \cap R$ if and only if $f(n_i) = 0$.
– Server decrypts w_i to get $g^{s_i f(n_i)}$ and counts the number of w_i if the corresponding decryption is $g^{s_i f(n_i)} = 1$ and denotes the count as e. Note that $n_i \in B \cap R$ if and only if $g^{s_i f(n_i)} = 1$ and that $g^{s_i f(n_i)}$ would be a random element to Server if $n_i \notin B \cap R$.
– Server accepts that $b \in L$ if and only if $e = 1$. Note that e is the number of node in $B \cap R$.

4.2 A Concrete Construction

We now present the detailed protocol.

Protocol 2 (Private Interval Check of Committed Integer)
– Common input: *A string* $c \in \mathbb{G}$.
– Auxiliary input for Client: *A tuple* (b,r) *such that* $c = com(b,r)$ *where* $b, r \in \mathbb{Z}_q$.
– Auxiliary input for Server: *m pairwise non-adjacent intervals $L_1,\ldots,L_m$.*
– Initial message: *For two positive integer k and ℓ,*
  - *Client computes b's covering set B in the ℓ-PKT.*
  - *Server computes L's minimal representer set S in the ℓ-PKT and computes $f(x) = \prod_{a_i \in R}(x - a_i) = \sum_{i=0}^{v} \alpha_i x^i$ where $v = |R|$.*
– The protocol:
  1. *Client proves the knowledge of a commitment to B by invoking Protocol 1 with Client's input (c,b,r), denoting the ZK constructed for Protocol 1 as P_1. Suppose that the output of Protocol 1 is the set $C = \{c_i : c_i = com(n_i,\delta_i), n_i \in B, \delta_i \in_R \mathbb{Z}_q, 0 \le i \le \ell\}$.*
  2. *Client sends C and P_1 to Server.*
  3. *Server verifies P_1, and aborts if it fails.*
  4. *Server evaluates $E_i = E(\alpha_i) = (g^{r_i}, g^{\alpha_i} h^{r_i})$, $r_i \in_R \mathbb{Z}_q$ for $i = 0,\ldots,v$.*

5. *Server creates*

$$P_2 = PK\{(\alpha_0, \ldots, \alpha_k, r_0, \ldots, r_k) : E_{i,1} = g^{r_i} \wedge E_{i,2} = g^{\alpha_i} h^{r_i} \text{ for } i = 0, \ldots, v\},$$

where $E_{i,1}$ and $E_{i,2}$ denote the first and the second item of E_i respectively.

6. *Server sends $(E_0, \ldots, E_v)$ and P_1 to Client.*
7. *Client verifies P_2, and aborts if it fails.*
8. *Client homomorphically evaluates f at elements in B by computing*

$$v_i = (\prod_{j=0}^{v} E_{j,1}^{n_i^j}, \prod_{j=0}^{v} E_{j,2}^{n_i^j})$$

for each $n_i \in B$ in random order, then computes $w_i = (v_{i,1}^{s_i}, v_{i,2}^{s_i})$ for $s_i \in_R \mathbb{Z}_q/\{0\}$. Note that $w_i = E(s_i f(n_i))$.

9. *Client creates the proof*

$$P_3 = PK\{(n_0, \ldots, n_\ell, s_0, \ldots, s_\ell, \delta_0, \ldots, \delta_\ell) : w_i = (\prod_{j=0}^{v} E_{j,1}^{n_i^j s_i}, \prod_{j=0}^{v} E_{j,2}^{n_i^j s_i}) \wedge$$

$$c_i = com(n_i, \delta_i), \text{ for } i = 0, \ldots, \ell\}.$$

10. *Client sends $(w_0, \ldots, w_\ell, P_3)$ to Server.*
11. *Server verifies P_3, and aborts if verification fails. Server also checks that $s_i \neq 0$ by verifying that $w_i \neq (1, 1)$ for $i = 0, \ldots, \ell$.*
12. *Server initializes a counter $e = 0$, evaluates $D(w_i)$ to get $g^{s_i f(n_i)}$ and increments e if $g^{s_i f(n_i)} = 1$.*
13. *Server outputs e.*

Some explanations are needed for P_2 and P_3. The ZK of P_2 needs a proving the knowledge of the discrete logarithm α_i and a proving the equality of the discrete logarithm r_i. The ZK of P_3 is more involved. For notational simplicity, for each $n \in B$, we set $w = (w_1, w_2) = (\prod_{j=0}^{v} E_{j,1}^{n^j s}, \prod_{j=0}^{v} E_{j,2}^{n^j s})$ and set, for $j = 1, \ldots, v$, $d_j = \text{com}(n^j, u_j)$ where $s, u_j \in_R \mathbb{Z}_q$. For each $n \in B$, Client creates

$$PK\{(n, u_1, \delta_2, \ldots, \delta_v) : d_1 = \text{com}(n, u_1) \wedge d_2/d_1^n = \text{com}(0, \delta_2) \wedge \ldots \wedge d_v/d_{v-1}^n = \text{com}(0, \delta_v)\}$$

and

$$PK\{(\beta_1, \ldots, \beta_v, \delta_1, \ldots, \delta_v, s) : w_1/E_{1,1} = \prod_{j=1}^{v} E_{j,1}^{\beta_j s} \wedge w_2/E_{1,2} = \prod_{j=1}^{v} E_{j,2}^{\beta_j s} \wedge$$

$$d_1 = \text{com}(\beta_1, \delta_1) \wedge \ldots \wedge d_v = \text{com}(\beta_v, \delta_v)\},$$

where the first ZK proves that d_j commits to n^j for $j = 1, \ldots, v$ and the second ZK proves that w is computed correctly by the product of E_i's correct exponentiation.

Remark 3. The complexity of Protocol 2: Step 1 needs $O(\ell \times \max(\ell, \log k))$ exponentiations in $\mathbb{G}$; Step 4 needs $3(v+1)$ exponentiations in $\mathbb{G}$; Step 5 needs $8(v+1)$ exponentiations in $\mathbb{G}$; Step 8 needs $2(v+2)\ell$ exponentiations in $\mathbb{G}$; Step 9 needs $O(v\ell)$ exponentiations in $\mathbb{G}$. The total computation is $O(\ell \times \max(\ell, \log k, v))$ exponentiations in $\mathbb{G}$. The total communication is $O(\ell \times \max(\ell, \log k, v)|\mathbb{G}|)$ bits.

Theorem 2. *Protocol 2, when constructed with a secure ZK protocol, is a secure protocol to compute $\mathcal{F}_{\mathcal{PIC}}$ assuming the DDH and DL assumptions hold.*

Proof. We separately prove security in the case that no parties are corrupted, the case that *Client* is corrupted, and the case that *Server* is corrupted.

No parties are corrupted. Step 1 ensures that C is a commitment to B. Step 4 and Step 5 ensure that the function $f(t)$ is encrypted correctly. Step 9 ensures that, for each $n_i \in B$, the encryption $E(s_i f(n_i))$ is computed correctly. Since $\mathbb{G}$ is prime order and $s_i \neq 0$ it must be that $g^{s_i f(n_i)} = 1$ if and only if $f(n_i) = 0$ in Step 12. Hence, e is the number of elements n_i in B such that $f(n_i) = 0$. On the other hand, $f(t) = 0$ if and only if $t \in R$ by the definition of $f(t)$ which results in that e is the number of elements of $B \cap R$. By Corollary 2, $b \in L$ if and only if $|B \cap R| = 1$. Therefore, $b \in L$ if and only if $e = 1$ and then the protocol is correct.

Server is corrupted. Let $\mathcal{A}$ denote an adversary controlling *Server*. We construct a simulator $\mathcal{S}$ as follows.

1. $\mathcal{S}$ evaluates, for $i = 0, \ldots, \ell$, $c_i = \mathrm{com}(n_i, \delta_i)$ for randomly chosen (n_i, δ_i), and forges the proof P_1.
2. $\mathcal{S}$ sends $c_0, \ldots, c_\ell$ and P_1 to $\mathcal{A}$.
3. $\mathcal{S}$ receives from $\mathcal{A}$ the encrypted polynomial $E_0, \ldots, E_v$ and P_2. If P_2 is invalid or if $\mathcal{A}$ has aborted, $\mathcal{S}$ stops and returns $\perp$ to the trusted party. Otherwise, $\mathcal{S}$ receives $\alpha_0, \ldots, \alpha_v$ from $\mathcal{A}$ and verifies the validity.
4. $\mathcal{S}$ evaluates $w_0, \ldots, w_\ell$ and creates the proof P_3.
5. $\mathcal{S}$ sends $w_0, \ldots, w_\ell$ and P_3 to $\mathcal{A}$.
6. $\mathcal{S}$ outputs whatever $\mathcal{A}$ does.

To prove that $\mathcal{A}$'s views in the real and simulated executions are computationally indistinguishable, we construct a sequence of games and show that the corresponding random variables $G_i^{\mathcal{A}(z)}(c, b, L)$ that consist of the view of $\mathcal{A}$ in game G_i are computationally indistinguishable. Recall that the commitment scheme *com* is perfectly hiding, and that the ElGamal encryption scheme is semantically secure.

Game G_0: The simulated execution.

Game G_1: $\mathcal{A}$ follows Protocol 2 except that $\mathcal{A}$ chooses a set S' to compute $f(t)$ which is the minimal representer set of a set L' where $L \neq L'$. In this case, the output of $\mathcal{A}$ is the same as the output of $\mathcal{S}$ where the input is L'. Therefore, $G_0^{\mathcal{A}(z)}(c, b, L)$ and $G_1^{\mathcal{A}(z)}(c, b, L)$ are computationally indistinguishable.

Client is corrupted. Let $\mathcal{A}$ denote an adversary controlling Client. We construct a simulator $\mathcal{S}$ as follows.

1. $\mathcal{S}$ receives C and P_1 from $\mathcal{A}$, verifies P_1, and extracts $(n_0, \delta_0), \dots, (n_\ell, \delta_\ell)$.
2. $\mathcal{S}$ evaluates, for $i = 0, \dots, v$, $E_i = E(\alpha_i) = (g^{r_i}, g^{\alpha_i} h^{r_i})$ for randomly chosen (r_i, α_i), and forges the proof P_2.
3. $\mathcal{S}$ sends $E_0, \dots, E_v$ and P_2 to $\mathcal{A}$.
4. $\mathcal{S}$ receives $w_0, \dots, w_\ell$ and P_3 from $\mathcal{A}$.

Since the commitment scheme is perfectly hiding and the ElGamal encryption scheme is semantically secure, we can see that the view of $\mathcal{A}$ is computationally indistinguishable from the view of $\mathcal{S}$.

In all, the proof is complete.

Corollary 3. *If $L = [0, b]$, Protocol 2, when constructed with a secure ZK protocol, is a secure integer comparison protocol in the malicious model assuming the DDH and DL assumptions hold.*

4.3 Asymptotic Complexity Analysis

We now give an asymptotic analysis of the computational and communication costs of our protocol.

Let b, k, ℓ, v, L, R, B be as in Section 4.1 and Section 4.2. For simplicity, we assume that $L = [a, e]$. The value v is then the cardinality of the minimal representer set of $[a, e]$. From Remark 3, the total computation is $O(\ell \times \max(\ell, \log k, v))$ exponentiations in $\mathbb{G}$ and the total communication is $O(\ell \times \max(\ell, \log k, v)|\mathbb{G}|)$ bits. Therefore, in order to reduce the computation and communication complexity, we need to reduce ℓ and $\max_k(\ell, \log k, v)$ by allocating k. Note that b, a, e take values from range $[0, k^\ell]$ where k^ℓ (setting $z = k^\ell$) is fixed. Intuitively, we can reduce the value of $\ell = \log_k z$ by raising the value k and therefore reduce the complexity. However, raising k will raise the value $\log k$. Furthermore, it is unclear how does v change when raising k. It is then needed to make sense how does v change.

We give some results for v.

Lemma 5. *Let $R = \{r_1, \dots, r_t\}$ be the minimal representer of a set of leaf nodes in an ℓ PKT. Let T_i be the coverage of node r_i for $1 \le i \le t$. Then $T_i \cap T_j = \emptyset$ for $1 \le i, j \le t$ and $i \ne j$.*

Proof. It is easy to verify that r_i is the minimal representer of T_i. Assuming that there exists a leaf node $e \in T_i \cap T_j$, then r_i, r_j both cover e and so one is an ancestor of the other, which is contrary to Lemma 1.

We first give a result for the case of $a = 0$, i.e., $L = [0, e]$.

Lemma 6. *In an ℓ PKT, the cardinality of the minimal representer set of $L = [0, e]$ is $\sum_{i=0}^{n} e_i$, where e_i is the ith (from right to left) digit in the k-ary notation of $e + 1$ and $e_n \ne 0$.*

Proof. Let $R = \{r_1, \dots, r_t\}$ be the minimal representer of L and T_i be the coverage of node r_i for $1 \leq i \leq t$. Then, for $1 \leq i \leq t$, r_i is the minimal representer of T_i. By Lemma 5, $T_i \cap T_j = \emptyset$ for $1 \leq i, j \leq t$ and $i \neq j$ and then $e + 1 = |L| = \sum_{i=1}^{t} |T_i|$. Since the number of nodes covered by $r_i \in R$ is of the form k^{τ_i} for a definite nonnegative integer τ_i by Definition 10, then $e+1 = \sum_{i=1}^{t} |T_i| = \sum_{i=1}^{t} k^{\tau_i} = \sum_{\tau_i, 1 \leq i \leq t} n_i k^{\tau_i}$, where n_i is the times of k^{τ_i} appeared in $\sum_{i=1}^{t} k^{\tau_i}$. By Lemma 2 there are at most $k-1$ nodes $r \in R$ such that $r.h = i$ for $0 \leq i \leq \ell$. Since each node $r \in R$ covers $k^{r.h}$ nodes in L, which says that each node at the same level in R covers the same numbers of leaf node, then $0 \leq n_i \leq k-1$. Hence, $\sum_{\tau_i, 1 \leq i \leq t} n_i k^{\tau_i}$ is the k-ary notation of $e+1$. Therefore, $|R| = \sum_{i=0}^{n} e_i = \sum_{\tau_i, 1 \leq i \leq t} n_i$. The proof is complete.

We now give a result for $L = [a, e]$.

Lemma 7. *Let L, a, e be as above. Let $a_t a_{t-1} \cdots a_0$ and $e_t e_{t-1} \cdots e_0$ be the k-ary notations of a and $e+1$ respectively and let $e_t \neq 0$. Then*

1. *if $a_t < e_t$, $|R_{[a,e]}| = (e+1)_k + ((a_t+1)k^t - a)_k - a_t - 1$,*
2. *otherwise (i.e., $a_t = e_t$), $|R_{[a,e]}| = |R_{[a - a_t \times k^t, e - e_t \times k^t]}|$,*

where $R_{[x,y]}$ denotes the minimal representer of $[x, y]$ in the ℓ PKT and $(x)_k$ denotes the sum of every digits in the k-ary notation of nonnegative integer x.

Proof. We separately prove the lemma in the case that $a_t < e_t$ and the case that $a_t = e_t$.

If $a_t < e_t$, there exist two cases: the case that $e_t = a_t + 1$ and the case that $e_t > a_t + 1$.

(a) If $e_t = a_t + 1$, then $L = [a, (a_t+1)k^t - 1] \cup [e_t k^t, e]$ while $e > e_t k^t - 1$, or else $L = [a, e_t k^t - 1]$ while $e = e_t k^t - 1$. While $e > e_t k^t - 1$, $R_{[a,e]} = R_{[a, e_t k^t - 1]} \cup R_{[e_t k^t, e]}$. By the symmetric property of PKT, $|R_{[a, e_t k^t - 1]}| = |R_{[0, e_t k^t - a - 1]}|$ and $|R_{[e_t k^t, e]}| = |R_{[0, e - e_t k^t]}|$. Then $|R_{[a,e]}| = |R_{[a, e_t k^t - 1]}| + |R_{[e_t k^t, e]}| = ((a_t+1)k^t - a)_k + (e - e_t k^t + 1)_k = \sum_{i=0}^{t-1} e_i + ((a_t+1)k^t - a)_k + (e_t - a_t - 1) = (b+1)_k + ((a_t+1)k^t - a)_k - a_t - 1$. While $e = e_t k^t - 1$, $|R_{[a,e]}| = 1 = \sum_{i=0}^{t-1} e_i + ((a_t + 1)k^t - a)_k + (e_t - a_t - 1)$.

(b) If $e_t > a_t + 1$, $L = \cup_{i=1}^{e_t - a_t - 1} [(a_t + i)k^t, (a_t + i + 1)k^t - 1] \cup [a, (a_t + 1)k^t - 1] \cup [e_t k^t, e]$ while $e > e_t k^t - 1$, or else $L = \cup_{i=1}^{e_t - a_t - 1} [(a_t + i)k^t, (a_t + i + 1)k^t - 1] \cup [a, (a_t + 1)k^t - 1]$ while $e = e_t k^t - 1$. While $e > e_t k^t - 1$, $R_{[a,e]} = \cup_{i=1}^{e_t - a_t - 1} R_{[(a_t+i)k^t, (a_t+i+1)k^t - 1]} \cup R_{[a, (a_t+1)k^t - 1]} \cup R_{[e_t k^t, e]}$. Then $|R_{[a,e]}| = \sum_{i=0}^{e_t - a_t - 1} |R_{[(a_t+i)k^t, (a_t+i+1)k^t - 1]}| + |R_{[a, (a_t+1)k^t - 1]}| + |R_{[e_t k^t, e]}| = (e_t - a_t - 1) + ((a_t+1)k^t - a)_k + (e - e_t k^t + 1)_k = \sum_{i=0}^{t-1} e_i + ((a_t+1)k^t - a)_k + (e_t - a_t - 1) = (e+1)_k + ((a_t+1)k^t - a)_k - a_t - 1$. While $e = e_t k^t - 1$, the conclusion can be proved similarly as the case (i).

If $a_t = e_t$, which says that $(0, a)$ and $(0, e)$ are both covered by the node $(t, a_t - 1)$ in level t, then $(t, a_t - 1)$ is an ancestor of all nodes in $R_{[a,e]}$ and $(t, a_t - 1) \notin R_{[a,e]}$. (Otherwise, if $(t, a_t - 1) \in R_{[a,e]}$, then $(t, a_t - 1)$ is the minimal representer of $[a, e]$. We have $a = a_t k^t$ and $e = a_t k^t + k^t - 1$ and so

$e+1 = a_t k^t + k^t$, which says that $e_t = a_t + 1$, a contradiction to $a_t = e_t$) Therefore, $R_{[a,e]} = R_{[a-a_t\times k^t, e-e_t\times k^t]}$ and so $|R_{[a,e]}| = |R_{[a-a_t\times k^t, e-e_t\times k^t]}|$.

In all, the proof is complete.

We then have Theorem 3.

Theorem 3. *Let a, e be as above. Let $a_t a_{t-1}\cdots a_0$ and $e_t e_{t-1}\cdots e_0$ be the k-ary notations of a and $e+1$ respectively and let $e_t \neq 0$. Let s be the first integer i such that $a_i < e_i$ for i from t to 0. Then, $|R_{[a,e]}| = \sum_{i=0}^{s} e_i + ((a_s+1)\times k^s - \sum_{i=0}^{s} a_i \times k^i)_k - a_s - 1$.*

From Theorem 3 we have Corollary 4.

Corollary 4. *Let a, e be as above. Then $|R_{[a,e]}| \leq 2(k-1)\log_k e$.*

By Theorem 3, $|R_{[a,e]}| = \sum_{i=0}^{s} e_i + ((a_s+1)\times k^s - \sum_{i=0}^{s} a_i \times k^i)_k - a_s - 1$ which displays that the minimal representer of $[a,e]$ in an ℓ PKT is determined entirely by the digits of the k-ary notations of $e+1$ and of "the complement of a". Hence, different k will result in different $R_{[a,e]}$ and so different $v = |R_{[a,e]}|$. It is interesting that a larger interval may have a very small sizable minimal representer if we choose a suitable k (needs not to be large), e.g., when the weights of $e+1$ and of "the complement of a", i.e. $(a_s+1)\times k^s - \sum_{i=0}^{s} a_i \times k^i$, are both small for k. Therefore, we should choose suitable k to reduce v and then to reduce the complexity of Protocol 2.

Recall that the total computation is $O(\ell \times \max(\ell, \log k, v))$ exponentiations in $\mathbb{G}$ and the total communication is $O(\ell \times \max(\ell, \log k, v)|\mathbb{G}|)$ bits by Remark 3. We should choose suitable k to leverage the values of $\ell, \log k, v$ to reduce the complexity of Protocol 2.

5 Conclusion

Protocol 2 gives an efficient and secure solution to the PIC problem. By choosing suitable k, we can further reduce the complexity of the protocol. This property is especially important in the location-based access control applications where resource is limited. Applying our protocol to the location-based access control applications would be one of our future works.

As stated in the paper, besides the PIC protocol, our work has a byproduct, i.e., a secure integer comparison protocol in the malicious model. It seems that the later protocol would have more future applications than the former, such as in electronic election, credential systems, electronic auction, etc.

Furthermore, we only commit to the input of Client, the integer, in Protocol 2. By considering fairness, it would be needed to extend the protocol to the case that both parties's inputs, i.e., the integer and the interval, are committed which would be our another future work.

6 Acknowledgment

The research is supported by following funds: National Science and Technology Major Project under Grant No.2010ZX01036-001-002 & 2010ZX01037-001-002, and the Knowledge Innovation Key Directional Program of Chinese Academy of Sciences under Grant No.KGCX2-YW-125 & KGCX2-YW-174.

References

1. Manachai Toahchoodee and Indrakshi Ray. On the formalization and analysis of a spatio-temporal role-based access control model. *Journal of Computer Security*, 19(3):399–452, 2011.
2. Claudio Agostino Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Privacy-enhanced location-based access control. In *Handbook of Database Security*, pages 531–552. 2008.
3. Claudio Agostino Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Access control in location-based services. In *Privacy in Location-Based Applications*, pages 106–126, 2009.
4. Michael S. Kirkpatrick, Gabriel Ghinita, and Elisa Bertino. Privacy-preserving enforcement of spatially aware rbac. *IEEE Trans. Dependable Sec. Comput.*, 9(5):627–640, 2012.
5. Elisa Bertino and Michael S. Kirkpatrick. Location-based access control systems for mobile users: concepts and research directions. In *SPRINGL*, pages 49–52, 2011.
6. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, pages 1–19, 2004.
7. Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4):593–622, 2005.
8. Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In *CANS*, pages 218–231, 2012.
9. Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. *J. Cryptology*, 25(3):383–433, 2012.
10. Changyu Dong, Liqun Chen, Jan Camenisch, and Giovanni Russello. Fair private set intersection with a semi-trusted arbiter. In *DBSec*, pages 128–144, 2013.
11. Jan Camenisch and Gregory M. Zaverucha. Private intersection of certified sets. In *Financial Cryptography*, pages 108–127, 2009.
12. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.
13. Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *ACNS*, pages 456–466, 2005.
14. Ahmet Erhan Nergiz, Mehmet Ercan Nergiz, Thomas Pedersen, and Chris Clifton. Practical and secure integer comparison and interval check. In *SocialCom/PASSAT*, pages 791–799, 2010.
15. Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In *ACISP*, pages 416–430, 2007.
16. Juan A. Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography*, pages 330–342, 2007.

17. Geir M. Køien and Vladimir A. Oleshchuk. Location privacy for cellular systems; analysis and solution. In *Privacy Enhancing Technologies*, pages 40–58, 2005.
18. Feng Zhang, Aron Kondoro, and Sead Muftic. Location-based authentication and authorization using smart phones. In *TrustCom*, pages 1285–1292, 2012.
19. Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In *Privacy Enhancing Technologies*, pages 62–76, 2007.
20. Sanjit Chatterjee, Koray Karabina, and Alfred Menezes. A new protocol for the nearby friend problem. In *IMA Int. Conf.*, pages 236–251, 2009.
21. Bin Zan, Tingting Sun, Marco Gruteser, Fei Hu, and Yanyong Zhang. A privacy preserving system for friend locator applications. In *MOBIWAC*, pages 93–100, 2011.
22. 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 J.*, 20(4):541–566, 2011.
23. Dario Freni, Sergio Mascetti, Claudio Bettini, and Marco Cozzi. Pcube: A system to evaluate and test privacy-preserving proximity services. In *Mobile Data Management*, pages 273–275, 2010.
24. Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.
25. Zi Lin, Denis Foo Kune, and Nicholas Hopper. Efficient private proximity testing with gsm location sketches. In *Financial Cryptography*, pages 73–88, 2012.
26. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
27. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
28. Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications.* Cambridge University Press, 2004.
29. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, pages 390–420, 1992.
30. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO*, pages 410–424, 1997.
31. Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *ASIACRYPT*, pages 234–252, 2008.
32. Kun Peng and Feng Bao. Batch range proof for practical small ranges. In *AFRICACRYPT*, pages 114–130, 2010.
33. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.