PPRKS: A Privacy Preserving Range Keyword Search Scheme

Yu Zhang[1], Zongbin Wang[1], and Tihong Qin[1]

Beijing Infosec Technologies Co., LTD.
zy168612@163.com, pony_wang@hotmail.com,qthxnjd@163.com

Abstract. Privacy preserving keyword search (PPKS) is investigated in this paper, which aims to ensure the privacy of client and server when a database is accessed. Range query has been recognized as a common operation in databases. In this paper, a formal definition of PPKS supporting range query is given, a scheme (PPRKS) is presented in accordance with Paillier's cryptosystem. To the best of our knowledge, PPRKS has been the only existing scheme that can effectively preserve the privacy of range keyword search. Moreover, it is demonstrated that the security of PPRKS is dependent on the semantic security of Paillier's cryptosystem. A detailed performance analysis and simulations are conducted to verify the practicality of PPRKS. As revealed by the theoretical analysis and the experimental results, the proposed scheme is practical.

Keywords: Privacy preserving keyword search · Private information retrieval · Range query · Privacy computing

1 Introduction

Keyword Search (KS) has been found as a fundamental operation of database. Two participants are involved in KS, including a server that holds a database consisting of a set of payloads and relevant keywords, as well as a client, who may send queries composed of keywords and get payloads relating to the above keywords. Privacy preserving keyword search (PPKS) is a vital research field of privacy computing, which focuses on protecting the privacy of client and server: (1) keeping the queries confidential from server (client privacy) and (2) ensuring that client can learn noting but the results of queries (server privacy). PPKS supporting (1) only is expressed as semi-private, while PPKS supporting both (1) and (2) is expressed as symmetric.

To be specific, privacy preserving keyword search problem is defined below. The format of database is $\{(x_i, p_i)\}_{i\in[n_{DB}]}$, where x_i denotes keyword and p_i denotes payload (database record). Here, $[n_{DB}]$ denotes $\{1, 2...n_{DB}\}$. Client sends a query and then obtains the results corresponding to that query. In existing works [1, 2], only accurate keyword matching has been supported, i.e., client can get p_i only if a query consists of searchword ω satisfying $\omega = x_i$ is provided, or get a special symbol $\perp$ otherwise. However, range query is commonly needed. Client should obtain p_i as long as $x_i \in [a, b]$, where $[a, b]$ is the range included in query.

1.1 Related Work

PPKS is developed from private information retrieval(PIR) [12, 13]. In PIR, server selects p_i on input index i from client, which can be considered as a constrained keyword. PIR was introduced by [12], which gave the definition of PIR and propsed two lines of work: information theoretic PIR (IT-PIR) and computational PIR (CPIR). In IT-PIR schemes [14–20], the database is replicated across several servers, which are assumed non-colluding. Client issues a query to the respective server and combines the responses from all servers locally. The benefits of IT-PIR schemes consist of inexpensive computation on servers and information-theoretic privacy guarantee, which means that even adversaries with unlimited computation power cannot crack them. However, since the assumption of non-collusion is difficult, IT-PIR is difficult to deploy.

CPIR schemes [9–11, 21–25] can be used with a single database, under cryptographic hardness assumptions. The disadvantage is expensive computation accompanied. Based on quadratic residuosity assumption, [11] proposed the first single-server PIR protocol. However, the payloads of database are limited to 1 bit, thus making it impractical. In addition, server privacy is not ensured since client can acquire extra information. [9, 10, 21] proposed PIR based on lattice problems, which can resist quantum attack. Impacted by the property of lattice-based croptosystem, the efficiencies of them are low. [22–24] proposed schemes with low communication costs. However, as mentioned above, PIR has special constraints compared with privacy preserving keyword search.

Privacy preserving keyword search was introduced by [2], in which it was termed private information retrieval by keywords. The construction of [2] is based on PIR and any data structure supporting keyword queries, offering server privacy using a trie data structure and a considerable number of rounds. In [1], a symmetric privacy preserving keyword search scheme was developed based on oblivious polynomial evaluation. However, [1, 2] can only support accurate keyword matching, which is not flexible.

The concept of privacy preserving range keyword search was introduced by [26], in which it was termed privacy-preserving range queries from keyword queries. However, during the executions of schemes proposed by [26], the numbers of matched records corresponding to client's queries are leaked to server, which do not meet the definition of indistinguishability [1]. Accordingly, schemes from [26] should not be considered as effective.

1.2 Contributions

The contributions of this paper are listed below.

We primarily give the formal definition of privacy preserving range keyword search scheme, which comprises Correctness, Client's privacy and Server's privacy.

Second, a novel privacy preserving rang keyword search (PPRKS) scheme is designed based on Paillier's cryptosystem and best range-covering (BRC) technique. To our knowledge, this is the first PPKS that supports range query effectively, which is more flexible.

Third, we prove that the security of PPRKS is based on the semantic security of Paillier's cryptosystem [3, 4], which has been proven.

Lastly, an in-depth theoretical performance analysis and simulations are conducted to evaluate the practicality of the proposed scheme. The theoretical analysis and the experimental results confirm that the proposed scheme is practical.

2 Problem Statement and System Model

2.1 Problem Statement

Consider the scenario mentioned in section 1, where a server holds database $\{(x_i, p_i)\}_{i \in [n_{DB}]}$, and a client will make a query and achieve the corresponding results. The problem is how to enable client to finish this query efficiently and securely with the following constraints:

- Both client and server are semi-honst, i.e., they are assumed to act according to their prescribed actions in the protocol, while being curious to acquire extra information by monitoring and inferencing.
- Query from client is hidden from server.
- Client is prevented from learning anything except for the results of the query.
- Range query is supported.

2.2 System Model

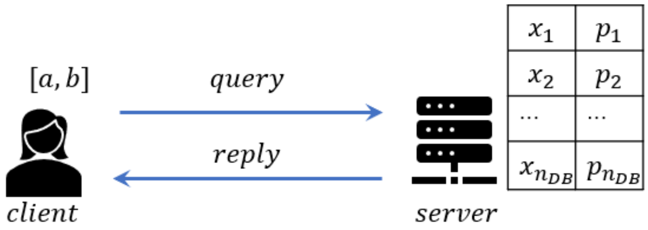

Fig. 1. System architecture

The system architecture is illustrated in Fig. 1. Server holds the database DB with n_{DB} entries (x_i, p_i). Client generates query with range [a, b] and obtains the corresponding payloads. To be specific, the proposed scheme consists of the following algorithms.

- Setup(Λ)$\rightarrow (PK, SK)$. Takes a security parameter Λ, client generates public key PK and private key SK.
- GenQuery$(a, b, PK)\rightarrow$ $Query$. With range $[a, b]$ and PK as the input, client generates $Query$, which implicitly contains $[a, b]$ and will be sent to server.

- GenReply($Query, PK, DB$)$\rightarrow$ *Reply*. With $Query$, PK and DB as the input, server generates *Reply*, which will be sent to client.
- GetResult(*Reply*, SK)$\rightarrow$ *Result*. With *Reply*, SK as the input, client obtains *Result* that corresponding to $[a, b]$.

2.3 Definitions

In the present section, some definitions are given for ease of understanding.

Definition 1. (Correctness.) If both parties are honest, after the scheme is run on inputs (a, b, DB), client outputs p_i so $(x_i, p_i) \in DB \wedge x_i \in [a, b]$, or $\perp$ if no such p_i exists.

Definition 2. (Client's privacy: indistinguishability.) For any probabilistic polynomial-time (PPT) S' executing the server's part and for any inputs $DB, (a_1, b_1), (a_2, b_2)$, the views that S' sees on input DB, when client uses (a_1, b_1) and it uses (a_2, b_2), are computationally indistinguishable.

Definition 3. (Server's privacy.) For any PPT C' taking the place of client, after accomplishing the scheme with Query on (a, b), C' can get nothing besides $\perp$ or p_i so $(x_i, p_i) \in DB \wedge x_i \in [a, b]$.

Definition 4. (Privacy preserving range keyword search scheme.) A two-party scheme meeting Definition 1, 2, and 3.

It is noteworthy that $a = b$ is permitted in the proposed scheme. In that case, it is equivalent to accurate keyword matching.

3 Privacy Preserving Range Keyword Search Scheme(PPRKS)

3.1 Mathematical Background

Paillier's Cryptosystem The proposed scheme uses a semantically-secure public-key encryption scheme that preserves the group homomorphism of addition and allows multiplication by a constant, which can be satisfied by Paillier's cryptosystem[3, 4]. In other words, it supports the following operations without private key: (1) Given two ciphertexts *Enc*(PK, m_1) and *Enc*(PK, m_2) encrypted by the same public key PK, *Enc*$(PK, m_1 + m_2)$ can be computed efficiently. (2) Given *Enc*(PK, m) and constant c from the same group with m, *Enc*$(PK, c * m)$ can be computed. The following corollary of the above two properties lays the foundation of the proposed scheme: Given $\{Enc(PK, a_i)|i = 0, 1, ...k\}$, where $\{a_i|i = 0, 1, ...k\}$ are the coefficients of a k degree polynomial P, and a plaintext x, *Enc*$(PK, P(x))$ can be computed.

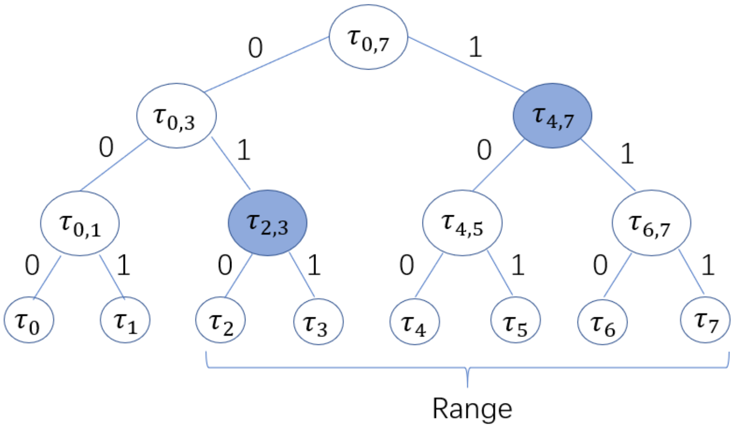

Fig. 2. Example of BRC

Best Range-Covering Technique The proposed scheme is also created by best range-covering (BRC) technique [5, 6]. Assuming a perfect binary tree with 2^m leaves and a numerical domain $D = \{0, 1...2^{m-1}\}$. Beginning from the left, the symbol τ_i represents the i-th leaf node, to which numerical value i is assigned. The respective non-leaf node is denoted by $\tau_{i,j}$, where τ_i and τ_j denote the left-most and right-most descendant of that node, respectively. The range of values $\{i, i+1, ..., j\} \subset D$ are assigned to node $\tau_{i,j}$. Given a range with continuous values over the domain D, BRC denotes the minimum set of tree nodes that covers the range accurately. The example in Fig. 2 suggests that $D = \{0, 1, ...7\}$, BRC of range $[2, 7]$ is $\{\tau_{2,3}, \tau_{4,7}\}$, which is illustrated by patterned nodes. For ease of use, in the proposed scheme, nodes of BRC are denoted by the path from root to themselves. In other words, the above BRC is expressed as $\{'01', '1'\}$, where $'01'$ and $'1'$ are character strings. GetBRC algorithm [7, 8] is defined below.

$GetBRC(a, b, m) \rightarrow BRC$. With the height m of a binary tree and two numerical values a, b as the input, where $[a, b] \subset [0, 2^m - 1]$, it outputs BRC of range $[a, b]$.

3.2 The Proposed Scheme

As described in 2.2, the proposed scheme comprises of four algorithms, which are Setup, GenQuery, GenReply, GetResult. First, client executes Setup to generate public key and private key. Subsequently, by running GenQuery, client obtains Query and sends it to server. On receiving Query, server executes GenReply to get Reply and sends it to client. Next, client executes GetResult on input of Reply and obtains the querying result.

Setup The input of this algorithm is security parameter Λ. Subsequently, Paillier's key generation scheme [3, 4] is executed by client below.

Two safe primes of $\Lambda/2$ bits are selected, and then $n = pq$ is computed. g is selected randomly with a non-zero multiple of n order from multiplicative group $Z^*_{n^2}$. Set $\lambda = lcm(p-1, q-1), \mu = (L(g^\lambda modn^2))^{-1} \, mod \, n$, where $L(x) = (x-1)/n, x \in Z^*_{n^2}$. Publish $PK = (n, g)$ as public key, and keep $SK = (\lambda, \mu)$ as private key.

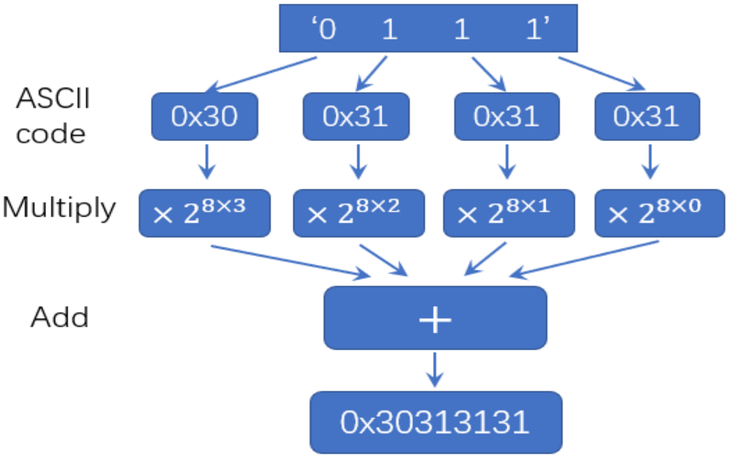

Fig. 3. Example of F_{code}

GenQuery This algorithm contains five steps.

Step 1. With range $[a, b] \subset [0, 2^m-1]$ as the input, client executes $GetBRC(a, b, m)$ and obtains BRC.

Step 2. Set $F_{code} : \{'0', '1'\}^* \rightarrow Z_n$ as a shared algorithm, of which a possible implementation is illustrated in Fig. 3. Takes BRC as input, client obtains the set $BRC_{code} = \{F_{code}(bn) | \forall bn \in BRC\}$ by executing F_{code}.

Step 3. Set $k_p = |BRC_{code}|$. Client uses interpolation to obtain the coefficients of the k_p-degree polynomial $P(y) = \sum_{i=0}^{k_p} a_i y^i$, which satisfies $P(y_i) \equiv 0 \, mod \, n$ for all $y_i \in BRC_{code}$.

Step 4. Client encrypts each of the coefficients with Paillier's encryption scheme and PK, obtains $Query' = \{Enc(PK, a_i) | i = 0, 1, ...k_p\}$

Since the sizes of BRCs corresponding to different ranges may be not consistent(e.g GetBRC(4,7,3) $=\{'1'\}$, GetBRC(3,6,3)$=\{'011', '10', '110'\}$), which can be adopted by server to undermine the indistinguishability defined in 2.3 as queries corresponding to two ranges may be different in scale. This can be mitigated by the following step.

Step 5. Client pads a certain number of encryptions of 0 to $Query'$ as the highest degree coefficients to obtain $Query$.

GenReply In the proposed scheme, x_i, p_i from DB holds $x_i \in Z_{2^m-1}, p_i \in Z_n$, and the low l bits of p_i store checksum of the other bits, where l is a statistical security parameter. In other words, for some shared hash funcion $H : \{0,1\}^* \to \{0,1\}^l$, it holds that $H([p_i]_{HBit\_(-l)}) = [p_i]_{LBit\_l}$, where $[p_i]_{LBit\_l}$ denotes the low l bits of p_i, and $[p_i]_{HBit\_(-l)}$ denotes the other bits.

This algorithm contains two steps.

Step 1. Set $[x_i]_{bin}$ as the binary string of x_i, e.g., $[6]_{bin} =' 110'$. Set $[x]_{HC\_j}$ as the left j-characters of x, e.g., $[[6]_{bin}]_{HC\_2} =' 11'$. For the respective $(x_i, p_i) \in DB$, server calculates the set $S_i = \{[[x_i]_{bin}]_{HC\_j} | j = 1, 2...m\}$. Then, by executing F_{code} on S_i, server obtains $SC_i = \{F_{code}(bn) | \forall bn \in S_i\}$. The output of this step is $DB_C = \{(SC_i, p_i) | i = 1, 2...n_{DB}\}$. Server can perform the above step only once and store the result for efficiency.

Step 2. Set $[x]_i$ as the i-th element of x. On input of Query and $[DB_C]_i$, using the property mentioned in 3.1, server can calculate $\{Enc(PK, P(x_{i,j})) | \forall x_{i,j} \in SC_i\}$. Then, $Reply_i = \{Enc(PK, r_{i,j} * P(x_{i,j}) + p_i) | \forall x_{i,j} \in SC_i\}$ can be calculated, where $r_{i,j}$ is choosen randomly from Z_n^*. Set $Reply = \{Reply_i | i = 1, 2...n_{DB}\}$

GetResult For each $Reply_i$ from $Reply$, client executes Paillier's decryption scheme with private key SK to get $p_{i,j} = Dec(SK, [Reply_i]_j)$, and the verification is conducted for $p_{i,j}$. Client will accept $p_{i,j}$ only if it satisfies the regulation mentioned in **GenReply**. Once $p_{i,j}$ is accepted, client will handle $Reply_{i+1}$. After all $Reply_i$ are processed, the set of accepted $p_{i,j}$ constitutes the result. If there is no value accepted, the result is $\perp$.

4 Security Analysis

The security of the proposed scheme is considered in the semi-honst model, which can be guaranteed by the following theorems.

Theorem 1. *(Correctness) The scheme PPRKS evaluates the range key word search function with high probability.*

Proof. The property of BRC is considered. If keyword $x_i \in [a, b]$, it is definite that $BRC_{code}(a, b, m) \cap SC_i \neq \emptyset$, which ensures that there exists $x_{i,j} \in SC_i$ satisfying $P(x_{i,j}) = 0 \, mod \, n$. Accordingly, $p_{i,j}$ obtained by client from GetResult satisfies $p_{i,j} = p_i$. Otherwise, client will obtain a random value Res_r from Z_n. The probability that Res_r is accepted by mistake is $1/2^l$, which can be ignored by setting l propertly.

Theorem 2. *(Client's privacy) In PPRKS, the views of server for any two inputs of client are indistinguishable.*

Proof. Assuming $[a_1, b_1]$ and $[a_2, b_2]$ are two rational random inputs of client. Next, $Query_1 = GenQuery(a_1, b_1, m)$ and $Query_2 = GenQuery(a_2, b_2, m)$ are the corresponding queries. Since the risk of difference in scale mentioned in

3.2 is eliminated by Step 5 of *GenQuery*, server can only try to distinguish $Query_1$ from $Query_2$ by ciphertexts in them. Since Paillier's[3, 4] cryptosystem is semantically secure, it is impractical to distinguish the above ciphertexts. Thus, the views of server for any two inputs of client are indistinguishable.

Theorem 3. *(Server's privacy) In PPRKS, it can be ensured that client can only obtain the payloads corresponding to its query.*

Proof. Assuming the input of client is range $[a, b]$. Consider the entry (x_i, p_i) where $x_i \notin [a, b]$. It is definite that $BRC_{code}(a, b, m) \cap SC_i = \emptyset$. Therefore, $\forall x_{i,j} \in SC_i$, it holds that $P(x_{i,j}) \neq 0 \, mod \, n$, $p_{i,j} = r_{i,j} * P(x_{i,j}) + p_i \, mod \, n$ is a random value from Z_n, which keeps p_i confidential from client. Accordingly, client can only obtain the payloads corresponding to range $[a, b]$

5 Performance Analysis

5.1 Theoretical Analysis

Supported features are compared between PPRKS and other existing schemes. Since the index of PIR can be considered as limited keyword, our comparison also includes several representative PIR schemes. In Table 1, server privacy ensures that a client only learns the desired elements, which is also a vital feature. Besides, arbitrary keyword is in contrast with index only. From the comparison, we can find that only PPRKS can comply with all listed requirements. The computation and communication cost of PPRKS are also compared with those of other existing shcemes. The results are listed in Table 3,4. PPRKS does not show any obvious advantage in efficiency, which can be acceptable because of the essential features it provides.

Table 1. Feature Comparsion

	Server Privacy	Arbitrary Keyword	Range Keyword
[11]	No	No	No
[9]	Yes	No	No
[10]	Yes	No	No
[1]	Yes	Yes	No
PPRKS	Yes	Yes	Yes

Table 2. Notations for Performance Comparisons

Notation	Meaning
d	the number of dimension that database is structured
n_{DB}	the number of entries in database
F	the ratio between the size of ciphertexts and plaintexts in lattice-based cryptosystem
N	degree of polynomial ring in lattice-based cryptosystem
q	ciphertext modulus in lattice-based cryptosystem
p	plaintext modulus in lattice-based cryptosystem
m	bit length of keyword

Table 3. Performance Comparison: Computation Cost

	Query	Reply	GetResult
[11]	$O(\sqrt{n_{DB}})$	$O(n_{DB})$	$O(1)$
[9]	$O(log(n_{DB}))$	$O(n_{DB})$	$O(1)$
[10]	$O(d\lceil \sqrt[d]{n_{DB}}/N\rceil)$	$O(n_{DB} + F\sqrt{n_D B})$	$O(1)$
[1]	$O(\sqrt{n_{DB}})$	$O(n_{DB})$	$O(\sqrt{n_{DB}})$
PPRKS	$O(m)$	$O(n_{DB} * m)$	$O(n_{DB} * m))$

Table 4. Performance Comparison: Communication Cost

	Query	Reply
[11]	$O(\sqrt{n_{DB}})$	$O((\sqrt{n_{DB}})$
[9]	$O(log(n_{DB}))$	$O(1)$
[10]	$O(d\sqrt{n_{DB}})$	$O(\lceil \frac{2log\, q}{log\, p}\rceil)$
[1]	$O(\sqrt{n_{DB}})$	$O(\sqrt{n_{DB}})$
PPRKS	$O(m)$	$O(n_{DB} * m)$

5.2 Experimental Results

In the present section, PPRKS is encoded in Python 3.8.10 and the simulations are performed on a laptop with Intel Core i5-10210U processor, 16 GB RAM. Nevertheless, the simulations are performed in a virtual machine that adopts 4 core and 8 GB RAM, running Ubuntu 20.04.1 LTS. The security parameter Λ is set to 256. The amount of entries in the database ranges from 100 to 1000. In the simulation of GenQuery, m ranges from 10 to 100, while in the simulations of GenReply and GetResult, m is set to 10,20 and 30. Impacted by the property of BRC(3.2), even though two ranges cover the same amount of elements, the corresponding BRCs may have different sizes. In our simulations, only the worst case is considered, i.e., ranges are selected, so the corresponding $|BRC|$s are always equal to m. In other words, the practical performance of PPRKS can be significantly better than the illustrated experimental results. The experimental results are well consistent with those of our theoretical analysis.

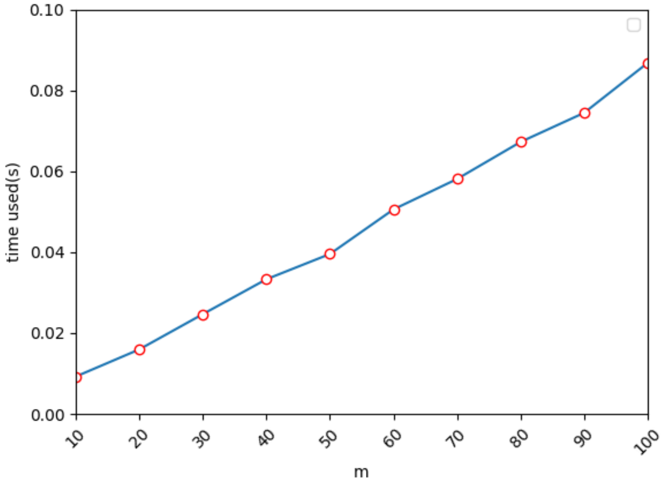

Fig. 4. Computation Cost of GenQuery

As depicted in Fig. 4, the computation cost of GenQuery increases almost linearly with m. When m is set to 100, GenQuery only requires 0.086798 s. Thus, a conclusion is drawn that GenQuery is highly efficient.

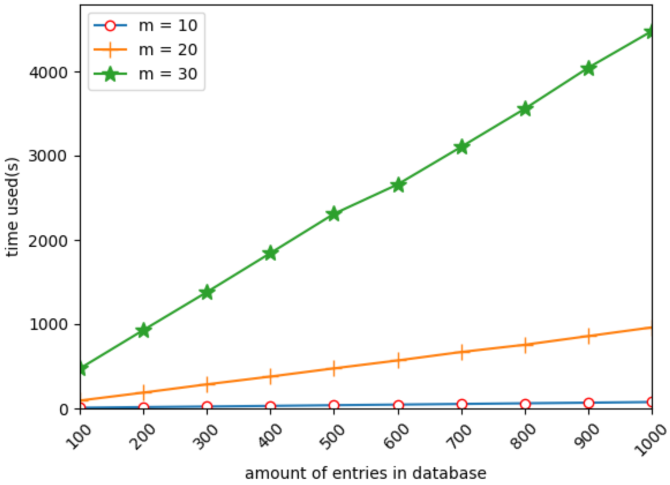

Fig. 5. Computation Cost of GenReply

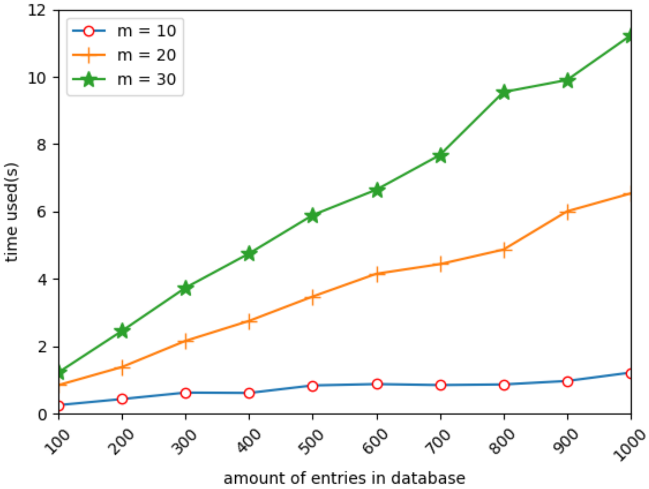

Fig. 6. Computation Cost of GetResult

As depicted in Fig. 5, the cost of GenReply grows approximately linearly with m and amount of entries. However, the affection from m is more significant. When the number of entries is fixed to 1000, GenReply needs 75.582571 s if $m = 10$, while 4477.399097 s if $m = 30$. The above result is achieved probably because m determines the degree of polynomial and the amount of polynomial evaluation operations.

Fig. 6 illustrates the computation cost of GetResult. The curves rise linearly with m and the amount of entries. When there are 1000 entries in the database, it takes client 1.214615 s to execute GetResult if $m = 10$, while 11.229619 s if $m = 30$.

The above analysis reveals that most of the computation overhead in PPRKS occurs on the server side. Naturally, server is configured with powerful computation resources, thus making it easy to afford that.

6 Conclusion

In this paper, a privacy preserving range keyword search scheme (PPRKS) is proposed in accordance with Paillier's cryptosystem. To our knowledge, PPRKS has been the first privacy preserving keyword search scheme supporting range query. PPRKS takes on a critical significance in numerous scenarios, thus becoming more flexible for applications. We have formally confirmed the security of the proposed scheme and analyzed the performance at theoretical and experimental levels.

Future research will focus on the practical application of PPRKS. The proposed prototype system has provided only the most basic functions, and it can be further improved in many aspects.

References

1. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O. (2005). Keyword Search and Oblivious Pseudorandom Functions.In: Kilian, J. (eds) Theory of Cryptography. TCC 2005, pp. 303–324.
2. Chor B , Gilboa N , Naor M . Private Information Retrieval by Keywords[J]. Ndss Symposium, 2000.
3. Paillier, P. (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes.In: Stern, J. (eds) Advances in Cryptology — EUROCRYPT '99.
4. Damgård, I., Jurik, M. (2001). A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System.In: Kim, K. (eds) Public Key Cryptography. PKC 2001.
5. Demertzis, S.Papaopoulos, O.Papapetrou, A.Deligian nakis, and M.N. Garofalakis. Practical private range search revisited. In SIGMOD, 2016.
6. Zuo, C. , et al. Dynamic Searchable Symmetric Encryption Schemes Supporting Range Queries with Forward/Backward Privacy. In ESORICS 2018.
7. WangJiafan, ChowSherman. Forward and Backward-Secure Range-Searchable Symmetric Encryption. Proceedings on Privacy Enhancing Technologies, 2022(1):28-48.

8. Kiayias A , Papadopoulos S , Triandopoulos N , et al. Delegatable pseudorandom functions and applications. ACM CCS2013, pp.669–684
9. Park, Jeongeun and Tibouchi, Mehdi. "SHECS-PIR: Somewhat Homomorphic Encryption-Based Compact and Scalable Private Information Retrieval. ESORICS 2020, pp.86-106
10. S. Angel, H. Chen, K. Laine and S. Setty, "PIR with Compressed Queries and Amortized Query Processing," 2018 IEEE Symposium on Security and Privacy (S&P), 2018, pp. 962-979
11. E. Kushilevitz and R. Ostrovsky, "Replication is not needed: single database, computationally-private information retrieval", Proceedings 38th Annual Symposium on Foundations of Computer Science, 1997, pp. 364-373
12. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS), Oct. 1995.
13. Gertner Y , Ishai Y , Kushilevitz E , et al. Protecting Data Privacy in Private Information Retrieval Schemes[J]. Journal of Computer and System Sciences, 2002, 60(3):592-629.
14. D. Demmler, A. Herzberg, and T. Schneider. RAID-PIR: Practical multi-server PIR. In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), Nov. 2014.
15. C. Devet, I. Goldberg, and N. Heninger. Optimally robust private information retrieval. In Proceedings of the USENIX Security Symposium, Aug. 2012.
16. I. Goldberg. Improving the robustness of private information retrieval. In Proceedings of the IEEE Symposium on Security and Privacy (S&P), May 2007.
17. Beimel, A., Ishai, Y. Information-Theoretic Private Information Retrieval: A Unified Construction. In Proceedings of International Colloquium on Automata, Languages, and Programming(ICALP), 2001, pp. 912–926
18. Banawan K , Ulukus S . The Capacity of Private Information Retrieval from Coded Databases[J]. IEEE Transactions on Information Theory, 2018, 64(3):1945-1956.
19. H. Sun and S. A. Jafar. The Capacity of Robust Private Information Retrieval With Colluding Databases, in IEEE Transactions on Information Theory, vol. 64, no. 4, pp. 2361-2370, April 2018
20. Freij-Hollanti R , Gnilke O , Hollanti C , et al. Private Information Retrieval from Coded Databases with Colluding Servers[J]. SIAM Journal on Applied Algebra and Geometry, 2016, 1(1).
21. C. Aguilar-Melchor, J. Barrier, L. Fousse, and M.-O. Killijian. XPIR: Private information retrieval for everyone. In Proceedings of the Privacy Enhancing Technologies Symposium (PETS), July 2016.
22. C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), May 1999.
23. Y.-C. Chang. Single database private information retrieval with logarithmic communication. In Proceedings of the Australasian Conference on Information Security and Privacy, July 2004.
24. C. Dong and L. Chen. A fast single server private information retrieval protocol with low communication cost. In Proceedings of the European Symposium on Research in Computer Security(ESORICS), Sept. 2014
25. H. Lipmaa and K. Pavlyk. A simpler rate-optimal CPIR protocol. In Proceedings of the International Financial Cryptography Conference, Apr. 2017.

26. Di Crescenzo, G., Ghosh, A. Privacy-Preserving Range Queries from Keyword Queries.In: IFIP Annual Conference on Data and Applications Security and Privacy,2015, pp.35-50