Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 October 2022 doi:10.20944/preprints202210.0064.v1

Secure Approximate String Matching using Homomorphic Encryption for Privacy-preserving Record Linkage

Shahidul Islam Khan
Department of Computer Science & Engineering (CSE)
International Islamic University Chittagong
Chittagong, Bangladesh
nayeemkh@gmail.com

Rakib Hosen
Department of Computer Science & Engineering (CSE)
International Islamic University Chittagong
Chittagong, Bangladesh
rkbjr11@gmail.com

Iqbal Hossain
Department of Computer Science & Engineering (CSE)
International Islamic University Chittagong
Chittagong, Bangladesh
iqbalhrasel57@gmail.com

***Abstract*—String matching is an important part in many real world applications. It must robust against variations in string field. In record linkage for two different datasets matching should detect two patients in common in spite of small variations. But it becomes difficult in case of confidential data because sometimes data sharing between organizations become restricted for privacy purposes. Several techniques have been proposed on privacy-preserving approximate string matching such as Secure Hash Encoding etc. Relative to other techniques for approximate string matching Homomorphic encryption is very new.**

In this paper we have proposed a Homomorphic Encryption based approximate string matching technique for matching multiple attributes. There is no solution currently available for multiple attributes matching using Homomorphic encryption. We have proposed two different methods for multiple attributes matching. Compare to other existing approaches our proposed method offers security guarantees and greater matching accuracy.

Keywords—Homomorphic Encryption, Privacy-preserving Record Linkage, approximate string matching

I. Introduction

Record linkage is the task of finding the records in dataset that refers to the same entity across different data sources. Privacy-preserving record linkage is a linkage that reveals the common characteristics of patient present in both datasets. Linking records between patient datasets would help us to better understanding of disease risks and treatment effects. Existing record linkage can be classified in two categories exact matching and approximate matching. In exact matching fixed set of rules is defined to identify records are matching or not. But the real world data is noisy and contain variations of string, so exact string matching is less effective in real world data. Errors can generate from many sources due to typing mistakes and other causes. That's why approximate string matching has become one of the important tasks in privacy-preserving record linkage. Several techniques have proposed on approximate string. Each of them solved different kind of problems. Exact matching between each record is less effective when we consider real world data, so in this thesis we are going to propose Homomorphic encryption technique for approximate string matching.

II. Background Study

A. Approximate Matching

A common and widely used approach for approximate string matching in record linkage where strings are decomposed into sets of n-grams and then a set similarity metric applied between two strings. If n = 2 we call it bigrams. In our proposed method we have used bigram decomposition. We have used _ as a special character to create special bigrams representing the beginning and the end of a string.

The set of bigrams of a x-character long string S is defined as

BIGRAMS(S) = {b(1)b(2),…,b(x)b(x+1)}

For example,

BIGRAMS(ALGORITHM)= {_A, AL, LG, GO, OR, RI, IT, TH, HM, M_}

B. Set Similarity

Given two sets p, q. The similarity of two strings can be defined as

$$Dice(p,q) = \frac{2 \cdot |p \cap q|}{|p| + |q|}$$

For an example let p = BIGRAMS(JAMAL) and q = BIGRAMS(KAMAL)

$$Dice(p,q) = \frac{2 \cdot 4}{6 + 6}$$

So we have to set a threshold t then we are interested in computing the set similarity of two strings and check if approximate matching is possible or not. More specifically we are interested in the given function:

$$ThreshDice(p,q,t) = \begin{cases} 1 & if\ Dice(p,q) \geq t \\ 0 & otherwise \end{cases}$$

C. Secure Approximate Matching

Record linkage is defined between two parties P_A and P_B. Their main goal is to identify all records that referring to the same entity securely. P_A and P_B hold a list of names respectively.

$A = [a_1,\ldots,a_n]$ and $B = [b_1,\ldots,b_n]$

For each $a_i \in A$ and $b_j \in B$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 October 2022 doi:10.20944/preprints202210.0064.v1

Let $a_i \leftarrow$ BIGRAMS(a_i), $b_j \leftarrow$ BIGRAMS(b_j)

Let $d_{ij} \leftarrow$ ThresDice(a_i, b_j, t)

If d_{ij} = 1 then it's means that they are same.

III. Related Work

Secure hash encoding [18] was the first technique for privacy-preserving record linkage. It is one way hash encoding function where a string converted into a hash value. It is converted into a hash value such that it is nearly impossible for current technology to know the original string value. Most popular algorithms for secure hashing are MD5, SHA-1, SHA-2. It can be applied for exact matching.

Bloom Filter [16] is the most popular methods for approximate string matching. In this method party runs strings through a type of locality-preserving hash function where similar strings produce similar hashes. It can be applied for multiple attributes matching. Relative to other methods it is fast but does not provide any formal security guarantees.

Damgard, Geisler, Kroigard proposed a Homomorphic encryption based technique called DGK Cryptosystem for secure integer comparison [6]. It provides formal security guarantees but it is not applicable for string comparison.

Aleksander Essex extended the DGK Cryptosystem for secure string comparison [2]. But it is applicable for only single attribute matching.

IV. Definitions and Notations

Definition 1 (Quadratic Residuosity)**.** An integer x is a quadratic residuo modulo p if there exists some integer y such that $y^2 = x \bmod p$, where x ϵ

It can be defined as follows:

$$QR_p(x) = \begin{cases} 1 & \text{if } x \text{ is a quadratic residue mod } p \\ 0 & \text{otherwise} \end{cases}$$

Definition 2 (Threshold Function)**.** For a threshold t ϵ Z^+ let τ_t be a threshold function which can be defined as follows:

$$\tau_t(x) = \begin{cases} 1, & \text{if } x \geq t \\ 0, & \text{otherwise} \end{cases}$$

Definition 3 (d-approximation of τ_t)**.** Let s be a prime and f be an offset where $0 < f < (s + 1)$ [2] such that

$$QR_p(x+f) = \tau_t(x) \qquad \text{For } 0 \leq x \leq d$$

Then we can say that the quadratic residuosity function QR_s d-approximates threshold function τ_t at offset f.

x	0	1	2	3	4	5	6
$QR_{59}(x + 23)$	0	0	1	1	1	1	1
$\tau_2(x)$	0	0	1	1	1	1	1

TABLE 1

Example: 6-approximation of $\tau_2(x)$ in Z_{59} at 23

Definition 4 (Minimum Set Intersection Cardinality)**.** Minimum set intersection cardinality is the minimum matched required to exceed the given threshold.
For example if two strings are a = "NUMBER" b = "DIGIT"

Len(BIGRAMS(a)) = 7, Len(BIGRAMS(b)) = 6, if the given threshold is 0.9 the MSIC will be = 6, because if the set-intersection of the two inputs less than 6 then their threshold value will be less than 0.9

V. Encryption Scheme

A. Key Generation

For a given security parameter select two x-bit primes p and q and select two y-bit primes u, v. x and y bit are chosen such that prime factorization is impossible. Then compute n = pq such that su | (p-1) and sv | (q-1). Let Z_n^* denote the set of relative primes of n and let G_{sub} denote the unique subgroup of Z_n^* which has order sv in Z_q^* and su in Z_p^*. Let G_{uv} is the unique subgroup which has order uv in Z_n^*. For simplicity let assumes that g is the generator of G_{sub} and h is the generator of G_{uv}. We have to select a prime s and offset f such that $QR_p(x+f) = \tau_t(x)$

So our public key is P-key = (n, g, h, d, y, d, f, s, t), where d is the domain bound

Let $\mu = (uv)^{-1}$ then our private key is S-key = uvμ

B. Encryption

For a message m in the range $0 \leq m \leq d$ select a uniform random element r from $[1,\ldots,2^{y-1}]$ and compute

$$C = g^m h^r$$

So C is the computed ciphertext for a message. The Encryption function will take message m and P-key as a parameter and output ciphertext C.

C. Decryption

For the decryption of the ciphertext C using our private key S-key compute

$$C^{uv\mu} = (g^m)^{uv\mu} (h^r)^{uv\mu}$$
$$= (g^m)^{uv(uv)-1}$$
$$= g^m$$

We have to compute m by applying discrete logarithm $\log_s(g^m)$ then $QR_s(m)$ is our final output

VI. Methodology

A. Multiple Attributes Matching (MAM1)

For single attribute matching proposed solution [2] by Aleksander Essex works fine for smaller string. the group order s increase rapidly if we increase the Maximum set size. So it will take more time and computation will be difficult. For example for string size 20 the group order s is 148139 and the offset is 74050. These values can be computed by brute force method.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 October 2022 doi:10.20944/preprints202210.0064.v1

To match single attributes our job is very simple. But it is challenging when we have to match multiple attributes because string length cannot exceeds the certain limit.

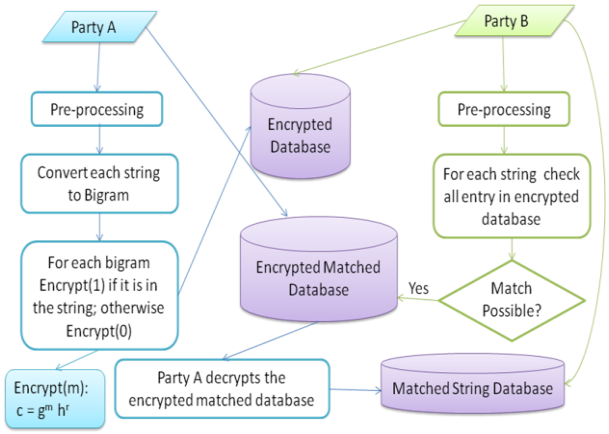

Fig 1. Basic work flow

We have proposed two different methods for multiple attributes matching. The first solution is simple, we have concatenated all attributes together. The functionality is defined as follows:

Multiple Attributes Matching I:

- **Public Parameters**: Public Key
- **Private Inputs**: Party A holds a list of strings A = $[a_1, a_2, \ldots, a_n]$ and Party B holds a list of strings B = $[b_1, b_2, \ldots, b_n]$
- **Functionality**:
    1) Party A and Party B pre-process their own datasets independently and each party concatenates their attributes independently.
    2) For each $a_i \in A$, Party A computes Ciphertext $C_i \leftarrow \text{EncryptedBigrams}(a_i)$ and send each Ciphertext C_i to Party B
    3) For each Ciphertext C_i and for each $b_j \in B$ Party B computes Set Intersection Cardinality. Let's denotes the function as SIC(plaintext, ciphertext). So Party B computes set intersection cardinality for C_i and b_j.

       $r_{ij} \leftarrow \text{SIC}(b_j, C_i)$
    4) For each r_{ij} Party A and Party B privately computes the threshold dice coefficient d_{ij}
    5) For each d_{ij} Party B send d_{ij} and i to party A.
- **Output:** If $\text{Decrypt}(d_{ij}) = 1$, then Party A outputs i'th entry

So this is the basic functionality of Multiple Attributes Matching I (MAM1). The main limitation of this approach is that the concatenated string length should be smaller otherwise it will be more complex. So this is one of the solution for Multiple Attributes Matching.

B. Multiple Attributes Matching II

As we stated earlier previous method (MAM1) have some limitations. In case of two same name, same age but different gender it can show matched in case of approximate matching. For example, if a patient is shahin, age 27 and gender Male in dataset A and other patient name shahin, age 27 and gender Female exists in dataset B. After concatenation it will be SHAHINBAM in dataset A and SHAHINBAF in dataset B. Surely it will match. But in real life these two patients are not same so to resolve this problem we have proposed another method for Multiple Attributes Matching. The main procedure of this method is given below:

1) Sort attributes according to their priority i.e. more important attribute comes earlier than less important attributes.
2) Set threshold for each attributes. As priority of each attributes are not same so the threshold for each attributes may not be the same. So we have to set threshold for each attributes.
3) If match possible for each attributes for a patient then record it as match.

The functionality of this method is defined as follows:

Multiple Attributes Matching II:

- **Public Parameters**: Public Key
- **Private Inputs**: Party A holds a list of strings A = $[a_1, a_2, \ldots, a_n]$ and Party B holds a list of strings B = $[b_1, b_2, \ldots, b_n]$
- **Functionality**:
    1) Party A and Party B pre-process their own datasets independently.
    2) Both Party sort their attributes according to the same rules.
    3) For each attributes and for each $a_i \in A$, Party A computes Ciphertext $C_i \leftarrow \text{EncryptedBigrams}(a_i)$ and send each Ciphertext C_i to Party B.
    4) For each attributes and for each Ciphertext C_i and for each $b_j \in B$ of same attribute Party B computes Set Intersection Cardinality. Let's denotes the function as SIC(plaintext, ciphertext). So Party B computes set intersection cardinality for C_i and b_j.

       $r_{ij} \leftarrow \text{SIC}(b_j, C_i)$
    5) For each r_{ij} Party A and Party B privately computes the threshold dice coefficient d_{ij}
    6) For each d_{ij} Party B send d_{ij} and i to party A.
    7) If $\text{Decrypt}(d_{ij}) \neq 1$ then stop the process

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 October 2022 doi:10.20944/preprints202210.0064.v1

- **Output:** If Decrypt(d_{ij}) = 1, for all attribute then Party A outputs i'th entry

So this is the basic functionality of Mulitple Attributes Matching II (MAM2) compare to the first proposed method it is more complex and it will take more time but it is more efficient than first proposed method.

VII. Implementation and Result

A. Implementation

We have implemented our proposed algorithm in Python 3 and we have used the python gmpy2 library for prime generation, find generator and modular exponentiation. We have pre-computed some values for faster encryption and decryption. We implemented our algorithm in an HP brand computer, which has Intel Core i3 2.60 GHz processor and 4GM RAM runs on Windows 10. The program processed 1000 patient names in an average of 11 minutes to generated 262MB ciphertext data and homomorphic matching generated 202MB ciphertext data.

B. Result

We have implemented single attribute matching to compare our proposed method.

	Unique Match for 100 Records		
	0.8	0.9	1.0
Single Attribute Matching	38	26	20
Multiple Attributes Matching I	32	11	10
Multiple Attributes Matching II	19	10	10

TABLE 2: Number Unique Match for three different thresholds

TABLE 2 shows the number of unique match of three proposed method for three different thresholds. From the Table 4.1 we can see that number match increases if we decreases threshold value. For the same threshold the number of mach for MAM2 is less than MAM1.

C. Matching Accuracy

Unlike bloom filter encoding there is no chance of collision. As we implemented our algorithm using threshold based comparison there are no matching errors. Because its implements its own functionality.

Therefore we are concerned only how often distinct records in dataset A trigger a match. That means we are interested only on the rate of false positives.

For each $a_i \in$ PartyA compared each $b_j \in$ PartyB, if a_i was matched with b_j then a FP recorded.

Method	Accuracy	Precision	Recall	F1 Score
SAM	0.98	0.92	1.0	0.95
MAM1	0.97	0.78	1.0	0.87
MAM2	0.99	0.91	1.0	0.95

TABLE 3: Matching accuracy of each method

False Positive Rate:

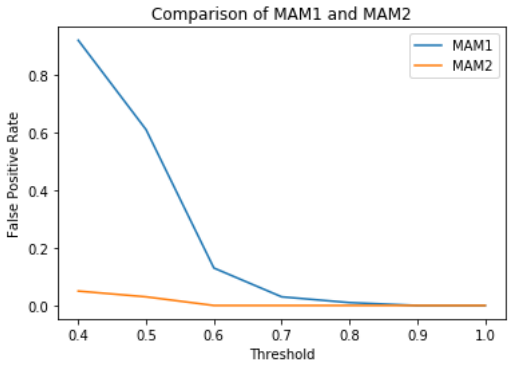

Fig 2. Comparison between MAM1 and MAM2

Fig 2. describes the comparison of False Positive Rate of our two proposed method (MAM1 and MAM2) we can see that MAM2 shows better performance compare to MAM1

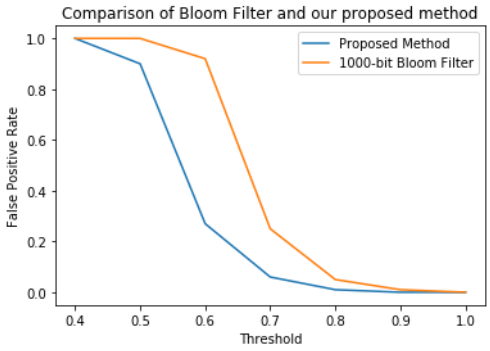

Fig 3. Comparison with 1000-bit bloom filter

Fig 3 shows the comparison of our proposed method with 1000-bit Bloom Filter. So we can see that the False Positive Rate is almost same when threshold is greater than or equal 0.9 but if we decrease the threshold value then the False Positive Rate of Bloom Filter increases compare to our proposed method. So our proposed method shows better performance compare to Bloom Filter.

D. Running Time Analysis

Method	For 100 data		Accuracy For Threshold 0.9
	0.8	0.9	
MAM1	86.5s	75.7s	0.97
MAM2	163.42s	153.47s	0.99

TABLE 4: Running time of each proposed method

From Table 4 We can see that MAM2 takes more time than MAM1 but the accuracy for threshold 0.9 for MAM2 is better than MAM1. MAM2 will show better performance for threshold less than 0.9. Times are given in second.

VIII. Limitations

In this paper we have extended the DGK Cryptosystem [6] for secure approximate string matching. Relative to other

methods the False Positive Rate of our method is very low, but we have some limitations in this method

Larger String Length: In extended version of DGK Cryptosystem[2] the group order s increases rapidly if we increase the string length. We have allowed maximum 20 characters in string field. But in real world data string can be very large. So it is one of the challenging tasks in this method.

Reducing Encryption Time: One of the most popular method for approximate string matching is Bloom Filter, which is very fast. Though our proposed method offers more formal security guarantees and greater matching accuracy compare to Bloom Filter Encoding our proposed method is not fast enough. So it is another challenging task.

IX. CONCLUSION

In the result section we have seen that there is great difference between our method and Bloom Filter in case of False Positive Rate, which means that our proposed method has greater matching accuracy. We just applied multiple attributes matching on three attributes for simplicity, but we can apply our proposed method Multiple Attributes Matching II (MAM2) for greater than three attributes and we can set different threshold value for each attribute.

So we think our proposed method can be applied on those areas where data security is very important and where two organizations do not want to share their data directly. It can be applied to build national data warehouse linking different datasets securely. We hope that it will contribute in the field secure approximate string matching.

REFERENCES

[1] Shahidul Islam Khan; Abu Sayed Md. Latiful Hoque: "Health Data Integration with secured record linkage", *3rd International Conference on Networking, Systems and Security (NSysS 2017);*

[2] Aleksander Essex : "Secure Approximate String Matching for Privacy-preserving Record Linkage" *IEEE transactions on information forensics and security.*

[3] Boris P. Hejblum, Griffin M. Weber: "Probabilistic record linkage of de-identified research datasets with discrepancies using diagnosis codes."

[4] Emily C. O'Brien, Ana Maria Rodriguez, Hye-Chung Kum, Laura E. Schanberg, Marcy Fitz-Randoph, Sean M. O'Brien, Soko Setoguchi: "Patient perspectives on the linkage of health data for research: Insights from an online community questionnaire" *International Journal of Medical Informatics.*

[5] D. Vatsalan P, Christen, and V.S. Verykios, "A taxonomy of privacy-preserving record linkage techniques." *Information Systems, vol. 38, no. 6, pp. 946-969, 2013*

[6] I.Damgard, M. Geisler, and M. Kroigaard, "Efficient and secure comparison for online auctions," *in Australasian Conference on Information Security and Privacy. Springer, 2007, pp. 416-430*

[7] https://www.ncsbe.gov/data-stats/other-election-related-data

[8] P. Hall, G. Dowling, "Approximate string matching ACM computing surveys" *12 – (4) (1980) 381-402*

[9] P.Jokinen, J. Tarhio, E. Ukkonent, "A coparison of approximate string matching algorithms", *Software practice and experience 26 (12) 1439 – 1458*

[10] D.Vatsalan and P. Christen, "Scalable privacy-preserving record linkage for multiple databases" *in Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, ACM, 2014, pp. 1795-1798*

[11] J.H. Cheon, M. Kim, and M. Lauter, "Homomorphic computation of edit distance," in international conference on financial cryptography and Data Security. Springer, 2015, pp. 194-212.

[12] https://crypto.stanford.edu/pbc/notes/numbertheory/

[13] Handbook of Applied Cryptography: http://cacr.uwaterloo.ca/hac/

[14] Danusha Vatsalan, Peter Christen, "An Iterative Two-Party protocol for scalable privacy-preserving record linkage,"

[15] Fully Homomorphic Encryption: Cryptography's Holy Grail

[16] B.H. Bloom. "Space/time trade-offs in hashing encoding with allowable errors", *Communication of the ACM, vol. 13. No. 7.pp. 422-426*

[17] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms.

[18] L. Dusserre, C. Quantin, H. Bouzelat, "A one way public key cryptosystem for the linkage of nominal files" *in epidemiological studies, Medinfo 8 (1995) 644–647.*

[19] A. Yao, "How to generate and exchange secrets", *27th Annual Symposium on Foundations of Computer Science, IEEE, 1986, pp. 162–167.*

[20] A. Karakasidis, V.S. Verykios, P. Christen, "Fake injection strategies for private phonetic matching," *International Workshop on Data Privacy Management, Leuven, Belgium, 2011*