Privacy preserving record linkage using homomorphic encryption

Sean M. Randall
Centre for Data Linkage
Curtin University
Perth, Australia
sean.randall@curtin.edu.au

Adrian P. Brown
Centre for Data Linkage
Curtin University
Perth, Australia
adrian.brown@curtin.edu.au

Anna M. Ferrante
Centre for Data Linkage
Curtin University
Perth, Australia
a.ferrante@curtin.edu.au

James H. Boyd
Centre for Data Linkage
Curtin University
Perth, Australia
j.boyd@curtin.edu.au

James B. Semmens
Centre for Population Health Research
Curtin University
Perth, Australia
james.semmens@curtin.edu.au

ABSTRACT

The bloom filter method for privacy preserving record linkage [24] has been shown to be both efficient, and provide equivalent linkage quality to that achievable with unencoded identifiers [23]. However in some situations, the bloom filter method may be vulnerable to frequency attacks, which could potentially leak identifying information [18]. In this paper we extend the bloom filter protocol to include a homomorphic encryption step which removes the vulnerability to frequency attacks. We evaluate our method by conducting a de-duplication of emergency presentation data.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration - Security, integrity, and protection

General Terms

Algorithms, Security

Keywords

Record linkage, privacy preserving record linkage, homomorphic encryption

1. INTRODUCTION

Record linkage is the process of identifying which person-based records from disparate data collections belong to the same individual. Throughout Australia, numerous operational record linkage units carry out this process, providing linked datasets to researchers, administrators and planners. Traditionally, linkage for research purposes has predominantly focused on the health sector, where it has had a significant impact on medical knowledge, and led to changes in health policy [5].

Administrative health data is highly sensitive, containing both medical and personal information collected about an individual during contact with health services and systems. The use of record linkage methods which implement privacy preserving techniques aims to satisfy privacy concerns regarding the release of named information, while allowing record linkage to take place.

Privacy preserving record linkage involves conducting record linkage on 'scrambled data', whereby records are identified as belonging to the same individual without the disclosure of personally identifying information. While these techniques provide safeguards around spontaneous recognition, they do not completely remove the privacy risk associated with large and complex datasets which are still susceptible to disclosure through unique combinations of the 'content' data.

Privacy preserving record linkage has recently become a popular area of research, with an array of protocols emerging. These protocols differ in their methods, maturity, practicality and suitability for large scale linkages. Comprehensive reviews of these methods exist in the literature [29].

1.1 Privacy preserving protocols - differences and requirements

Privacy preserving protocols can be divided into which utilise the data owners only (often known as two-party protocols) and those which include one or more independent third parties, who do not own data (often known as three-party protocols). Under a two-party protocol, only the organisations that hold data are involved in the linkage process. Under a three party model, data custodians provide encoded or encrypted data to an independent third party, which perform a specialised linkage of this data.

In Australia, when linking administrative data, the usefulness of two-party protocols appears limited. Two-party protocols require data custodians to take a substantial and ac-

This paper was presented at the First International Workshop on Population Informatics for Big Data (PopInfo'15), Sydney, 10 August 2015. Copyright of this work is with the authors.

tive part in the linkage process. However, data custodians exist to manage the quality and security of their collections and linking data is not part of their core business. While custodians are often happy for their datasets to be used for linked research, they typically do not have the resources to undertake linkage themselves, and in many cases conducting linkage does not offer them any direct benefit. At the same time, there are already a number of dedicated ‘third party’ linkage centres around Australia with significant expertise, and the resources to undertake record linkage [13, 1, 4].

Privacy preserving protocols also differ in the level of privacy they provide. The lowest level of privacy are provided by techniques such as the statistical linkage key (SLK) [16], which simply amalgamate personally identifying attributes (like name, date of birth and gender) into one variable in clear text. The next level of privacy techniques encodes data using hash functions so that those with access cannot learn any information directly from the encoded values; however these encoded values are vulnerable to frequency attacks, which can leak personally identifying information. A final class of privacy techniques encrypts data in such a way that it is not possible to learn any information about individuals. Such methods utilise cryptographic techniques similar to those used in modern computing. Few methods such as these exist, and those that do typically require data custodians to carry out multiple computations and communication steps [29, 7, 31].

For a privacy preserving record linkage protocol to be practical, it needs to be secure, efficient and provide high linkage quality; ideally both linkage efficiency and quality would be comparable to what can be achieved with un-encoded personal identifiers. Record linkage is computationally expensive, and while tight turnaround times are not always required for record linkage processing, slower algorithms can result in impractical processing times and unworkable solutions [10]. In addition to responsive linkage services, researcher expectations also include high quality matching to ensure they can draw the correct conclusions from their research [12].

1.2 Privacy preserving record linkage using Bloom filters

A protocol for privacy preserving linkage that appears most promising utilises Bloom filters to encode data in a way that is both efficient, and allows string similarity measures (important for ensuring high linkage quality) to be computed. The use of Bloom filters for privacy preserving record linkage was first proposed by Schnell in 2009 [24]. Since then, there have been numerous variants, extensions and evaluations of this protocol [23, 25, 19, 8, 30, 15]. The method has been shown to provide similar linkage quality to that found in probabilistic record linkage with un-encoded identifiers, and to be efficient enough for large scale linkages [23].

However recent evaluations have shown this method may be vulnerable to frequency attacks; first in its original field level form [22, 19], and then later for record level Bloom filters [18]. As such, in situations where very high levels of privacy are required, this method may not be sufficient.

1.3 Objectives of this paper

In this paper we outline an extension to the generic Bloom filter protocol, which utilises a somewhat homomorphic encryption scheme that allows us to calculate a similarity metric on fully encrypted identifiers. We implement and evaluate this method on a sample of real data sourced from hospital emergency departments.

2. PROTOCOL

2.1 Overview

Our proposed protocol is a ’four party’ protocol; it utilises two independent parties to conduct linkage. One has responsibility for conducting the actual linkage (the *linker*), while the second has responsibility for decrypting the similarity score of the resulting record-pairs (the *decrypter*). In our protocol, data is first encoded into Bloom filters using the methods developed by Schnell [24]. We utilise record level Bloom filters [25] (where all fields from a record are placed within a single Bloom filter) although our method would also work with field level Bloom filters. These Bloom filters are then encrypted using the system described below, again at an individual record level. This encryption will use as input a public key supplied by the decrypting third party. This two-stage encryption process (personal identifiers encoded into Bloom filters which are then encrypted) is carried out by the data custodians. It should be noted that our protocol does not limit the number of data custodians to two; any number of data custodians can be involved in the linkage.

The encrypted data is then sent to the *linker*, who conducts the required linkage. The output of this linkage (a list of the record-pairs which have been compared along with their encrypted similarity score) is then sent to the *decrypter*, who, with possession of the private key, can decrypt the similarity score. The role of the decrypter must be separate from the linker, as giving the linker access to the private key to decrypt the encrypted similarity score would also allow them to decrypt the encrypted Bloom filters. An outline of these data movements is shown in Figure 1.

2.2 Bloom filter method

A Bloom filter is a binary vector of a set length with all values initially set to zero. Using the method outlined by Schnell [24], bigrams (overlapping sets of two letters) of personal identifiers are hashed, with their modulus taken with respect to the length of the Bloom filter. The corresponding position in the Bloom filter is then set to 1. There are several variations to this method; in our implementation all personally identifying fields (i.e. first name, surname, date of birth, sex, and address) are placed within a single large Bloom filter.

Bloom filters can be compared using typical set similarity comparisons. In this implementation we focus on the dice coefficient metric, outlined in section 2.6.

2.3 Homomorphic encryption

A homomorphic encryption scheme allows computations to be carried out on encrypted data producing encrypted results; when this encrypted data is finally decrypted, the decrypted results match the results of those same operations performed on an unencrypted version of the data. While

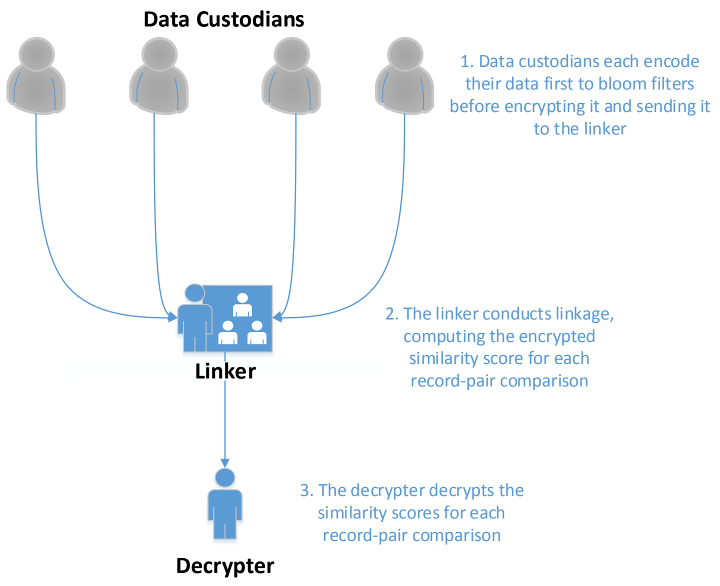
Data Custodians
1. Data custodians each encode their data first to bloom filters before encrypting it and sending it to the linker
Linker
2. The linker conducts linkage, computing the encrypted similarity score for each record-pair comparison
3. The decrypter decrypts the similarity scores for each record-pair comparison
Decrypter

Figure 1: Data movements for the proposed protocol

homomorphic encryption protocols have existed for many years, protocols prior to 2000 only supported simple operations of either addition or multiplication. In 2009, Gentry developed the first fully homomorphic encryption system which allowed arbitrary calculations [11], and since then a large number of advances in this area have been made. However fully homomorphic systems are still too slow to be practical for most purposes [20].

Somewhat homomorphic encryption schemes only support a limited number of operations on encrypted data; however they are much faster and thus far more practical. In this paper we utilise a somewhat homomorphic encryption scheme developed by Lauter, Naehrig and Vaikuntanathan [20], along with a packing method for encrypting data developed by Yasuda [32] which allows us to compute similarity measures.

2.4 Encryption method

This scheme of Lauter, Naehrig and Vaikuntanathan [20] bases its security on the *ring learning with errors* problem. In colloquial terms, this problem is based on the difficulty of distinguishing a true signal (in this case, the secret) from noisy data. The problem, while relatively recent, is believed to be exponentially hard [20], and forms the basis for numerous modern cryptosystems [2, 21].

The scheme used in this paper allows an arbitrary number of additions of encrypted values, along with a set number of multiplications.

The system utilises several parameters. These include;

- The dimension n, which is a multiple of 2, and the corresponding cyclotomic polynomial $f(x) = x^n + 1$.

- The modulus q, a prime. Together, q, n and $f(x)$ define the rings $R := \mathbb{Z}[x]/f(x)$ and $R_q := R/qR = \mathbb{Z}_q[x]/f(x)$.

- The standard deviation σ of a discrete Gaussian error distribution χ.

- An integer $t < q$, which defines the message space.

Description of the algorithms key generation, encryption and decryption are given below. These are taken verbatim from Yasuda et al [32].

Key Generation We choose an element $R \ni s \leftarrow \chi$ and sample a random element $a_1 \in R_q$ along with an error $R \ni e \leftarrow \chi$. We define the public key pk as (a_0, a_1), where $a_0 := -(a_1 \cdot s + t \cdot e)$, and we define the secret key sk as s.

Encryption For a plaintext message $m \in R_t$, with public key (a_0, a_1), the encryption samples $R \ni u, f, g \leftarrow \chi$ and computes $Enc(m, pk) = (c_0, c_1) = (a_0 u + tg + m, a_1 u + tf) \in (R_q)^2$, where $m \in R_t$ is considered an element of R_q.

Decryption For a ciphertext $ct = (c_0, ..., c_\xi) \in (R_q)^{\xi+1}$ (homomorphic multiplication will increase ciphertext size), with private key s, decryption is computed by $Dec(sk, ct) = [\widetilde{m}]_q mod\, t \in R_t$ where $\widetilde{m} = \sum_{i=0}^{\xi} c_i s^i \in R_q$.

2.5 Packing method

The homomorphic encryption scheme described above will allow us to encrypt individual numbers, and perform operations on these encrypted numbers. It is possible then to use the scheme to compute the dice coefficient of two Bloom filters, by first encrypting each element in the two Bloom filters individually, multiplying the elements of each position together, and summing these results. However such a scheme would be extremely slow, requiring a large number of encryptions and computations for every comparison.

Packing methods provide an alternative, allowing a vector of values to be encrypted in a single operation. Operations can then be homomorphically computed on this vector. In this work we utilise a packing method developed by Yasuda [32]. This method allows us to encrypt an entire Bloom filter (essentially a binary vector) at once, and compute its inner product using a single multiplication operation.

For a Bloom filter A of length n with elements $A_0, \ldots, A_{n-1}$ we define two packed ciphertexts.

$$ForwardPack(A) = \sum_{i=0}^{n-1} A_i x^i$$

$$BackwardPack(A) = -\sum_{i=0}^{n-1} A_i x^{n-i}$$

where Σ refers to the regular summation operator. Both of these polynomials are then encrypted as described in 2.4. Each Bloom filter is both forward and backward packed; that is, there are two encrypted values for each Bloom filter.

We can compute the inner product of two Bloom filters by multiplying one Bloom filter's forward packing by the others backward packing, as shown below.

$$ForwardPack(A) \times BackwardPack(B)$$

$$= (\sum_{i=0}^{n-1} A_i x^i) \times (-\sum_{i=0}^{n-1} B_i x^{n-i})$$

$$= \cdots - (\sum_{i=0}^{n-1} A_i B_i x^n) + \ldots$$

$$= \cdots + A \cdot B + \ldots$$

in R_t, since $x^n = -1$ with all other terms non-constant. Thus after a multiplication, upon decryption, the value of the constant term in the resulting polynomial will be our inner product.

2.6 Computing similarity measures

The most common metric used in Bloom filter similarity calculations is the dice coefficient, typically expressed as

$$Dice\ Coefficent_{A,B} = \frac{2h}{a+b}$$

where h refers to the number of positions in both bloom filters set to 1, and a and b refer to the number of positions set to 1 in bloom filters A and B respectively.

This equation can be re-written as

$$Dice\ Coefficent_{A,B} = \frac{2A \cdot B}{A \cdot A + B \cdot B}$$

where $\cdot$ refers to the inner product operation. This allows us to compute the dice coefficient using the packing method described above.

The cryptosystem employed does not allow integer division; instead, we calculate the encrypted values of the numerator and denominator separately. Both of these values are provided (encrypted) to the *decrypter* for each record pair. Once decrypted, the *decrypter* can calculate the dice coefficient from these two provided values.

2.7 Related work

Our protocol aims to allow linkage to be conducted with only the minimum participation of data custodians, and to a level of security where frequency based information is not available to the independent third parties.

There have been a number of related works published in the literature. A range of secure set intersection protocols have been proposed [26, 27, 17], many of which adopt homomorphic encryption methods to ensure security. While these methods have strong security equivalent to our protocol, they operate without the use of an independent third party, and instead require multiple communication steps from data custodians.

The closest protocol to the one described in this paper is by Kantaricioglu et al. [14], who provides a method for privacy-preserving joins utilising homomorphic encryption and two independent third parties. Similar to our work, in this protocol data custodians are only required to encrypt and transfer their data, taking no further part in the protocol. A uniquely identifying key is used to determine whether two records should be joined. A homomorphic subtraction operation is then performed when comparing individual records; where this subtraction (when decrypted) equals to 0, the two records have the same unique identifier, and so are joined.

The main difference between our method and Kantaricioglu's is that ours is aimed at the problem of record linkage, where we do not have keys which uniquely identify individuals across distinct datasets. Our proposed method tolerates the full range of 'noisy' data, utilising approximately matching techniques to handle missing values, misspellings, incorrect values and changing values over time. Previous evaluations of the approximate matching method used in our protocol have shown it to perform as well as probabilistic linkage on un-encoded identifying information [23].

3. EVALUATION

3.1 Evaluation details

We evaluated this system by performing a deduplication of 275,626 event records (one years' worth) from an emergency presentation data collection. First name, surname, date of birth, sex, address and postcode fields were used in linkage. These fields were mapped into a single 512 bit bloom filter, using weighting methods developed by Durham et al [9]. A standard blocking method was used to enable timely linkage; the date of birth field was used as the sole block.

Bloom filters were then encrypted using the encryption scheme described above. Our system utilised the parameters $n = 1024$, $\sigma = 8$, $t = 512$, and q, a 54 bit prime. These parameters were chosen to be the most efficient possible, while both ensuring correctness of results, and a security level equivalent to 128 bits; the detail of determining ac-

Table 1: Results from de-duplication of emergency presentation data

Linkage Type	Precision	Recall	F-Measure
Linkage on un-encoded identifiers	0.985	0.978	0.981
Linkage with unencrypted bloom filters	0.985	0.977	0.981
Linkage with encrypted bloom filters	0.985	0.977	0.981

curate and secure parameters is described in Lauter et al [20].

Our linkage quality results were evaluated using precision and recall measures, as recommended in the record linkage literature [6]. Efficiency and privacy were also evaluated with reference to measures described within the privacy preserving literature [28]. The emergency presentation dataset had been previously independently linked by a data linkage unit with their results made available to us. The results were used as the 'truth set' with which we compared our results.

Encryption, linkage and decryption were performed on a 64-bit Windows Server virtual machine with an Intel Xeon E5-2609 CPU at 2.4GHz, with 32GB of memory. Our implementation utilised a single core.

3.2 Results

The results for the linkage of emergency presentation data using encrypted Bloom filters, unencrypted Bloom filters, and un-encoded personal identifiers are shown in Table 1. As expected, there was no difference in quality between encrypted Bloom filters and unencrypted Bloom filters. The Bloom filter methods result in linkage quality equal to that achieved by linkage with un-encoded identifiers.

The encrypted Bloom filter linkage took slightly over 12 hours to complete, while the encryption step took 4 hours and 20 minutes, and the decryption of the answer file took almost 17 hours. A total of 1,164,305 record comparisons were performed.

In terms of individual operations, a single inner product calculation took, on average, 31 milliseconds, while encryption of a single record took 58 milliseconds, and decryption of a single record-pair took 52 milliseconds.

Our implementation was significantly slower than the more optimised implementation reported on by Yasuda et al [32]. Using equivalent parameters, our inner product calculation (i.e. our linkage) was 27 times slower, while our encryption and decryption of data was 23 and 14 times slower, respectively. While their CPU was slightly faster (Intel Xeon X3480 at 3.07GHz), the majority of this difference appears to be due to code optimisations.

In terms of privacy, using the privacy metrics of Vatsalan [28], our protocol on its own has a degree of privacy of 0.0 (absolute privacy), as all records have completely different ciphertext values. However our protocol is not complete; for efficiency, it requires a blocking component to be used in conjunction which itself may decrease privacy.

4. DISCUSSION

As expected, the linkage quality achieved through our protocol was the same as that achieved using the regular Bloom filter method, and the same as that achieved through probabilistic linkage. The advantage of the presented methodology is a far higher level of security over the Bloom filter method. This method provides a level of security equivalent to that provided by regular encryption algorithms, and removes the possibility of frequency attacks; the same plaintext value can encrypt to a very large number of ciphertext values.

By building upon the Bloom filter methods previously published, our methodology can be expected to achieve the same level of linkage quality as other Bloom filter methods. It can also leverage off the significant work already conducted to improve and refine the Bloom filter methodology, such as Durham's weighting method (used in this paper) [9].

A key limitation to our proposed method is speed. As currently implemented, our method is only suitable for small linkages. However, our naive implementation is approximately 14 to 27 times slower than the more optimised version developed by Yasuda [32]. By optimising the code used in our implementation, our method would be suitable for larger dataset sizes. Additional performance improvements could be made by using distributed computing techniques. Given the high security level of our encryption method, it may also be feasible to utilise public cloud computing resources to perform our inner product calculations, which would provide substantial potential for scalability. The blocking method used (comparing only records with the same date of birth) is relatively strict, and similarly strict blocks may be a requirement to ensure the efficiency of this method.

5. CONCLUSIONS

As far as we are aware, this is the first record linkage protocol which provides a demonstrably high level of security, without requiring numerous communication steps by data custodians. Future developments will focus on improving performance to a comparable level with that achieved by Yasuda et al [32].

This paper presents a protocol for record comparison, and does not provide any recommendations for private blocking systems. However, a private blocking scheme is necessary for a complete private linkage system. Future work will explore the use of more secure blocking methods.

Our protocol provides protection against attacks by the third or fourth party; however it does not protect against collusion by these two parties. Should these parties collude, the security of our system reduces to that of the regular privacy preserving linkage using Bloom filters (which has been evaluated previously [18]).

6. ACKNOWLEDGMENTS

This project is supported by the Australian Government National Collaborative Research Infrastructure Strategy's Population Health Research Network.

The authors would also like to thank Wenjie Lu, whose publically available code formed an initial reference point for our implementation of this cryptosystem.

7. REFERENCES

[1] J. H. Boyd, A. M. Ferrante, C. M. O'Keefe, A. J. Bass, S. M. Randall, and J. B. Semmens. Data linkage infrastructure for cross-jurisdictional health-related research in australia. *BMC health services research*, 12(1):480, 2012.

[2] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[3] Z. Brakerski and V. Vaikuntanathan. *Fully homomorphic encryption from ring-LWE and security for key dependent messages*, pages 505–524. Springer, 2011.

[4] E. Brook, D. Rosman, C. Holman, and B. Trutwein. Summary report: research outputs project, WA data linkage unit (1995–2003). Perth: WA data linkage unit, 2005.

[5] E. L. Brook, D. L. Rosman, and C. D. J. Holman. Public good through data linkage: measuring research outputs from the western australian data linkage system. *Australian and New Zealand Journal of Public Health*, 32(1):19–23, 2008.

[6] P. Christen and K. Goiser. Quality and complexity measures for data linkage and deduplication. In *Quality Measures in Data Mining*, pages 127–151. Springer, 2007.

[7] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms*, pages 13–22. ACM, 2001.

[8] E. Durham, Y. Xue, M. Kantarcioglu, and B. Malin. Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage. *Information Fusion*, 13(4):245–259, 2012.

[9] E. A. Durham. *A framework for accurate, efficient private record linkage.* Thesis, 2012.

[10] A. Ferrante and J. Boyd. A transparent and transportable methodology for evaluating data linkage software. *Journal of Biomedical Informatics*, 45(1):165–172, 2012.

[11] C. Gentry. *A fully homomorphic encryption scheme.* Thesis, 2009.

[12] K. Harron, A. Wade, R. Gilbert, B. Muller-Pebody, and H. Goldstein. Evaluating bias due to data linkage error in electronic healthcare records. *BMC medical research methodology*, 14(1):36, 2014.

[13] K. A. Irvine and L. K. Taylor. The centre for health record linkage: fostering population health research in NSW. *New South Wales public health bulletin*, 22(2):17–18, 2011.

[14] M. Kantarcioglu, A. Inan, W. Jiang, and B. Malin. Formal anonymity models for efficient privacy-preserving joins. *Data & Knowledge Engineering*, 68(11):1206–1223, 2009.

[15] A. Karakasidis and V. S. Verykios. Secure blocking+ secure matching= secure record linkage. *JCSE*, 5(3):223–235, 2011.

[16] R. Karmel. *Data linkage protocols using a statistical linkage key.* Australian Institute of Health and Welfare, 2005.

[17] L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology–CRYPTO 2005*, pages 241–257. Springer, 2005.

[18] M. Kroll and S. Steinmetzer. Automated cryptanalysis of bloom filter encryptions of health records. *arXiv preprint arXiv:1410.6739*, 2014.

[19] M. Kuzu, M. Kantarcioglu, E. Durham, and B. Malin. A constraint satisfaction cryptanalysis of bloom filters in private record linkage. In *Privacy Enhancing Technologies*, pages 226–245. Springer, 2011.

[20] K. Lauter, M. Naehrig, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.

[21] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):43, 2013.

[22] F. Niedermeyer, S. Steinmetzer, M. Kroll, and R. Schnell. Cryptanalysis of basic bloom filters used for privacy preserving record linkage. *Journal of Privacy and Confidentiality*, 6(2):3, 2014.

[23] S. M. Randall, A. M. Ferrante, J. H. Boyd, J. K. Bauer, and J. B. Semmens. Privacy-preserving record linkage on large real world datasets. *Journal of biomedical informatics*, 50:205–212, 2014.

[24] R. Schnell, T. Bachteler, and J. Reiher. Privacy-preserving record linkage using bloom filters. *BMC Medical Informatics and Decision Making*, 9(41), 2009.

[25] R. Schnell, T. Bachteler, and J. Reiher. A novel error-tolerant anonymous linking code. Report, Working Paper Series No. WP-GRLC-2011-02. Nürnberg, Germany: German Record Linkage Center, 2011.

[26] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM, 2002.

[27] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4):593–622, 2005.

[28] D. Vatsalan, P. Christen, C. M. O'Keefe, and V. S. Verykios. An evaluation framework for privacy-preserving record linkage. *Journal of Privacy and Confidentiality*, 6(1):3, 2014.

[29] D. Vatsalan, P. Christen, and V. S. Verykios. A taxonomy of privacy-preserving record linkage techniques. *Information Systems*, 38(6):946–969, 2013.

[30] D. Vatsalan, P. Christen, and V. S. Verykios. An

efficient two-party protocol for approximate matching in private record linkage. In *Proceedings of the Ninth Australasian Data Mining Conference-Volume 121*, pages 125–136. Australian Computer Society, Inc., 2014.

[31] M. Yakout, M. J. Atallah, and A. Elmagarmid. Efficient private record linkage. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1283–1286. IEEE, 2009.

[32] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. *Practical packing method in somewhat homomorphic encryption*, pages 34–50. Springer, 2014.