ReSet: A Protocol for Private Matching

Mahmood Ahmad
Department of Computer Engineering
Kyung Hee University, South Korea
rayemahmood@oslab.khu.ac.kr

Byeong Ho Kang
University of Tasmania
Australia
Byeong.Kang@utas.edu.au

Sungyoung Lee
Department of Computer Engineering
Kyung Hee University, South Korea
sylee@oslab.khu.ac.kr

ABSTRACT

Private matching (PM) has a vast domain of applications to get its benefit including social media, e-health and commerce. The core concept of PM comprises on revealing only common values between two parties and ensuring the privacy of the remaining ones. In this paper we have proposed a protocol *ReSet*, for PM that works with random values rather than original datasets. Utilizing random values further ensures the privacy by introducing an additional layer in between original values and encryption or hashing technique. *ReSet* takes only two communications between the communicating parties and can work with or without involving services of any third party. The unique feature of randomness minimizes the additional disclosure of information that is exposure of similarity magnitude for values being matched. Our experimental evaluation reveals that if protocol is executed more than once for similar set of values, the output results will resist against pattern identification if intercepted by a malicious user.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Information flow controls

General Terms

Security and Privacy

Keywords

Private Matching; Privacy and Security; Information disclosure

1. INTRODUCTION

The private computation of two datasets intersection is a useful primitive for various applications including social media [15],[20] business community [3] and e-health applications [8]. When this intersection is restricted to reveal only the common values and nothing else, Private Matching (PM) plays its pivotal role. Ensuring privacy aspect of shared information is the core idea for the PM protocol. In general, PM protocol is a set of operations through which two parties learn their common values without disclosing any additional information. The solution for PM has been introduced by Freedman et al [7] and according to its definition, if two parties, Alice and Bob having set of values $A = \{a_1, a_2...a_n\}$ and $B = \{b_1, b_2...b_n\}$ respectively, then at the end of protocol they should only learn which values they have in common and nothing else as given in equation 1.

$$A \cap B =: \{a_u | \exists_v, a_u = b_v\} \leftarrow PM(A, B) \quad (1)$$

Since the inception of PM, it has evolved with few variations. These variations satisfies the constraints as imposed by the communicating parties. Considering Alice and Bob as communicating parties, these constrain may fall in one of the following categories.

- Only Alice is interested to know either she has any value in common with Bob or not. It is called the asymmetric PM
- Both, Alice and Bob are interested to know about their common value(s), it is symmetric PM
- If Alice learns the outcome of PM at first place then she is expected to share it with Bob. To ensure honesty on her share, services of third party may be required.
- If Eve is acting as third entity between Alice and Bob then she can also learn about values being matched. Using services of third party can disclose information on matching values, that is not desired by Alice or Bob.

Other than symmetric and asymmetric classification given in [16], there is yet another scenario where total number of common elements are required to be known rather exact values match. In PM, this is known as magnitude similarity. All these variations are aimed to protect the user confidence that nothing beyond common value(s) will be disclosed to either party but nonetheless the set cardinality. The first two scenarios are susceptible to disclose set cardinality. For example, Alice and Bob are delivering their products 'a' and 'b' in a certain metropolitan area. There are few customers 'c', who are using both products, 'a' and 'b'. Delivery service 'd' of Danial is used for this purpose. To save the delivery cost, Alice and Bob decided to send their products to 'c' customers as one package. For this, Alice and Bob need to

IMCOM '15, January 08 - 10 2015, BALI, Indonesia

http://dx.doi.org/10.1145/2701126.2701190.

use the PM to know the list of their common customers 'c'. After executing the PM protocol, now they know the list of their common customers, however; while intersecting they need to share complete list of their customers which can disclose the total strength of customer dealt by both Alice and Bob.

The privacy preserving protocols are aimed to protect the sensitive data along with sufficient deterrence against the additional information leakage too. Besides protecting the direct visibility of data, additional leakage of information can be minimized with added complexity of encryption or hashing techniques. In this paper we have proposed a protocol for PM between two parities. At the end of protocol execution both parties will end up with a set of random values that will reflect the exact matchings. Due to this reason the protocol is named as *ReSet*.

The idea is based on constructing the finite degree polynomials with random roots on behalf of original values. Individually, these roots are independent from original values to be matched. For multiple iterations of matching process, communicating parties evaluate their values with these polynomials. Effect of random roots makes the *ReSet* outcome dynamic every time it is used. Secondly, selecting random roots are far below in computation than choosing encryption and decryption process for generating different output with similar values.

With *ReSet*, following contributions have been made in the area of private matching.

- During PM, involving parties may or may not include services of third party.
- For PM, two parties need to communicate only twice to find out common values.
- If there are three parties P_1,P_2,P_3 and only P_1 is doing PM with P_2 and P_3 such that all three parties have same set of values. Then, the set of values being matched between P_1,P_2 and P_1,P_3 will be altogether different. This difference will resist the additional information leakage that P_2 and P_3 have similar values.

The rest of the paper is organized as follows.
Section 2 is about related work.Section 3 covers the main idea and its methodology. Evaluation and experimental results are given in Section 4. Discussion with respect to possible threat model and limitations are given in Section 5. Section 6 is about future work directions. Paper concludes in Section 7.

2. RELATED WORK

In very initial work, Freedman et al. presented his idea on private matching[7]. With gradual evolution on PM mechanism various algorithms and techniques have been proposed so far. These variations are mainly due to different requirements by parties involving in PM or it is result of fine tuning to achieve least computational cost. Other than values which are found common, the residual data during or after the PM can assist a curious user to find additional information which is not desirable to be known otherwise. Keeping this in view, we will present the related work which is about PM in general and avoiding slipway of additional information in particular. In recent years the protocol of Private Set Intersection (PSI) [7],[14],[13],[10] and Private Set Union (PSU) [8],[11],[12] are considered as main driving factors in the area of PM. In [4], these two approaches have been highlighted for loosing adequate privacy on server end. The Private Set Intersection Cardinality (PSI-CA) or Private Set Union Cardinality (PSU-CA) [4] is for a scenario where clients are only allowed to learn the magnitude of common values rather the exact values. This approach encourages to find out total number of matching values first and that too without knowing them. After executing PSI-CA or PSU-CA the involving parties can further decide either to go for value matching protocol or not.

With PM protocol, values that are found common can lead for record linkage problem. The linkage works with attempts to match records stored at distinct parties (e.g., hospitals), but which represent the same entity (e.g.,patient). Records found at one place can be matched with record on other place. To handel this issue two approaches have been proposed which is the sanitization [16],[18] and secure multiparty communication (SMC)[9]. Total reliance on first approach is not effective in terms of accuracy. Results acquired from santized matching can end up in false positives. On the other hand SMC overcomes this issue but with computational overhead. The computational cost of SMC operations performed in [7],[19],[14] end up in $O(m * n)$ cryptographic operations. If $m = n = 100$ then integration of this task will require 10^8 cryptographic operations. The idea on santized query has been upgraded to differential privacy in [5]. With this technique, instead of sending the perturbed records after sensitization, user query is evaluated statistically. For detailed description on differential privacy reader may refer to [5],[6]

In [1] , Agrawal et al. proposed a model for private matching under the provision of encryption function $\mathcal{E}$ and $\mathcal{E}'$ such that $x \in X, \mathcal{E}(\mathcal{E}'(x)) = \mathcal{E}'(\mathcal{E}(x))$. Matching values using this approach will reveal common elements to only one party at first place. Now it is the discretion of that party who has knowledge of common elements whether to share it with other party or not. Even if sharing is done, there is another question that either all elements are shared honestly or not. Besides, using double encryption Agrawal, et al suggested to use simple hashing mechanism to find out the common elements between two parties. Matching common elements with hashing technique will definitely require a common hash function at first place to be decided between the involving parties. Besides common elements found using hash technique, $H(A) \cap H(B)$, a curious party can employ a brute force attempt using same hash function to find out those elements which are not common over a finite domain of elements. Hashing techniques are not appropriate in PM as they are more towards ensuring integrity of data contents.

3. MAIN IDEA

PM protocols are not limited with client server communication. Utilizing services of third entity which could be a cloud or semi trusted third party (TTP) is another option to consider. Using services of third entity can overcome the issue where both parties need to know common values at the same time. The technique of Oblivious Polynomial Evaluation (OPE) using the homomorphic encryption [17] can assist to achieve this model as given in [7]. It utilizes the

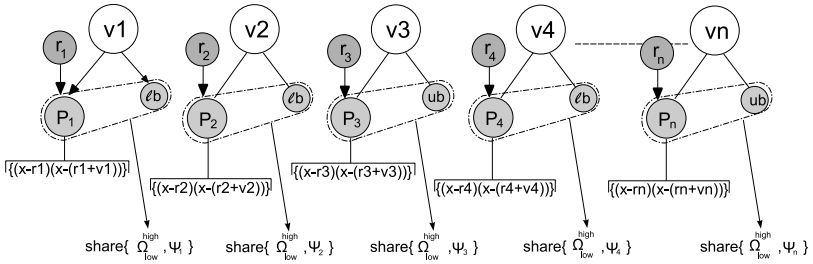

Figure 1: *ReSet* Architecture

roots and coefficients of polynomial. Encrypted coefficients are evaluated by cloud server or any TTP to find if values hold equality or not. The output which is revealed at server or third party gives no clue for actual value being matched, no matter equality holds or not. Using the same principles of polynomial we have used it in such a way that instead of evaluating polynomial $P(x)$ on valid roots, it will be evaluated on random values. The proposed protocol has achieved a notable performance in terms of communication overhead and local execution time. In terms of communication, it takes place only twice, however; computation becomes complex when matching values are very high in numbers.

In existing PM protocols, encryption or hashing techniques are applied on actual values to be matched. In *ReSet*, before applying specific hashing $\mathcal{H}$ or encryption $\mathcal{E}$ techniques on actual values we first transform original value into a polynomial with random roots. Later, one of these root value is encrypted and is shared with the other party where this root value is independent from original value. Utilizing random roots this way makes the protocol with dynamic output every time it is used. During evaluation of system, the same effect of randomization holds for a same value that is matched for more than once. This feature makes the system more resilient for an eavesdropper or offline attack that are explained in the evaluation section.

3.1 Notations and Assumptions

The notations used in proposed protocol are given in Table 1. Values to be matched constitute a finite set V, and a polynomial for value evaluation by each user is represented by $P(.)$. Coefficient identifier is represented by Ω, which is shared by each party during the PM. Final value which will discover either the original values being matched are same or not is represented by Δ. we assume that distribution of asymmetric keys have already been done securely and matching is done on a finite and not very large number of values

3.2 Methodology

Alice and Bob are considered as the communicating parties for the brief explanation of proposed methodology. Alice is running a fertilizer company and Bob is dealing in pesticides. Alice and bob are using carriage services of Eve to deliver their respective products in various markets of the city. For their mutual interest, they decided to share names of their common customers. With this sharing, fertilizers and pesticides will be shipped as a one package reducing the freight charges around 50%. To find out their common customers we assume that address of each customer is unique

Table 1: Notations used in the descriptive detail

Notation	Description
$\mathcal{V} = \{v_1, v_2, .., v_n\}$	Set of values to be matched for some finite domain
v_i	A particular value instance v_i, where $v_i \in \mathcal{V}$
$\mathcal{P}(.)$	Polynomial created by using r and $(r + v_i)$, where r is a random number
$\bigodot_x$	Set of polynomials held with user x
$\sum_{i=1}^{n} \left\{\Omega_{low}^{high}, \Psi_i\right\}$	Set of values exchanged between users during the PM process. First parameter Ω is coefficient identifier which is either *low* or *high*. Value of second parameter Ψ depends upon the value of Ω, such that $\Psi = \begin{cases} (r) & \text{if } \{\Omega_{low}\} \\ (r + v_i) & \text{if } \{\Omega_{high}\} \end{cases}$
$\sum_{i=1}^{n} \{\Delta_i\}$	After receiving $\sum_{i=1}^{n} \left\{\Omega_{low}^{high}, \Psi_i\right\}$, these values are used to evaluate results using all polynomial in $\bigodot$ one by one. The evaluation results are then stored in Δ.
$\mathcal{E}_A$, $\mathcal{D}_A$	Asymmetric encryption and decryption algorithms
k_{pub}, k_{pri}	Public and private key pair for asymmetric encryption algorithm

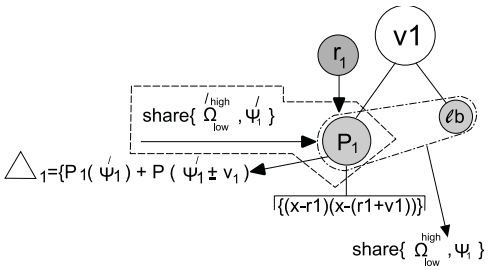

Figure 2: Sharing and Evaluation

and is represented in numeric format. For example a customer address with zip code 500, block 2, street no 8 and building no 66 will appear as 5002866. From here onward,

a particular address will be considered as a unique instance, $v_i \in \mathcal{V}$.

Both Alice and Bob will perform similar operations to share and find out their common customers, therefore, we will explain the set of operations performed by Alice alone. After preparing $\mathcal{V}$, Alice generates random numbers such that for each $v_i \in \mathcal{V}$ there exist one random number r. Alice then construct the polynomials $\mathcal{P}$ using r and $(r + v_i)$ and save them as $\bigodot_{Alice}$. Against each polynomial $\mathcal{P}(.)$, Alice creates the set of values as given in equation 2

$$\sum_{i=1}^{n} \left\{ \Omega_{low}^{high}, \Psi_i \right\} \tag{2}$$

Although these value does not provide any clue for original values, however; in reply to these values from Bob to Alice, the eavesdropper can utilize this plain information for some statistical attack. With this consideration, these values are encrypted with the public key of Bob and then sent as shown in equation 3.

$$Alice \xrightarrow{\sum_{i=1}^{n} \left\{ \mathcal{E}_A \left(k_{pub}, \left(\Omega_{low}^{high}, \Psi_i \right) \right) \right\}} Bob \tag{3}$$

Similarly Bob shares his value with Alice. The block diagram for polynomial construction is shown in Figure 1 whereas the values that are received by each party are shown in Figure 2, enclosed in dotted lines whereas the complete information flow is shown in Figure 3. After receiving the values they are decrypted with respective secret key as shown in equation 4

$$\sum_{i=1}^{n} \left\{ \mathcal{D}_A \left(k_{pri}, \left(\Omega_{low}^{high}, \Psi_i \right) \right) \right\} \tag{4}$$

After receiving these values from Bob, Alice then uses these values one by one for evaluating the previously saved polynomials and prepare the Δ_i. Each element in Δ_i is calculated using the algorithm given in 1[1]. Alice keep a copy of this set, Δ_i, with her and send the encrypted copy to Bob as shown in equation 5

$$Alice \xrightarrow{\sum_{i=1}^{n} \left\{ \mathcal{E}_A \left(k_{pub}, \Delta_i \right) \right\}} Bob \tag{5}$$

Bob will also send his set Δ_i to Alice using the same way given in equation 5.

After receiving values through equation 5 , both Alice and Bob will decrypt it and match it with their values. The number of matching entries will reveal the exact address and number of common customers to both parties. Complete flow for information sharing that take place between Alice and Bob is shown in figure 3

4. EVALUATION RESULTS

To evaluate the system output with values of Δ, we made comparison with data sets having ten (10) values each for Alice and Bob. There would be 100 values in each Δ_{Alice} and Δ_{Bob}. We put only 50% matching values that means, while matching 100 values, only 5 will be same. The evaluation has been performed without using any encryption to find prior encryption values distribution and randomness especially for multiple iterations for similar data sets. Figure 4 shows the similar values at 5 places and shown with a small rectangle enclosed in a square. Same set of values is used again and outputs the results as shown in Figure 5. Although the original value set is same for both but the evaluated results holds different each time. This phenomenon avoids inference of any pattern recognition for similar values over and over. The encryption process on these values $\mathcal{E}_A(\Delta_i, k_{pub})$, further strengthens the overall architecture.

The is a multiplicative growth trend in elements of Δ. Figure-6 shows the growth of time in milliseconds along y-axis. These values are generated without involving encryption process. Along x-axis we have considered same as well as different elements count for two parties. The time to generate $100,000$ values in Δ took around 6 seconds on a machine with Core i3 processor running Windows-7 Enterprise (64bit) Operating System. With $100,000$ values we mean that, for any two datasets $\mathcal{V}_1$ and $\mathcal{V}_2$ having m and n number of elements such that $m * n = 100,000$.

[1]With modification in usage, we have utilized this algorithm from our previous work [2]

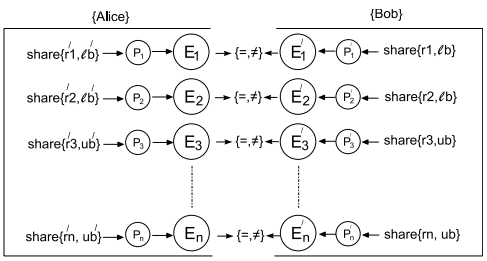

Figure 3: Information sharing and flow

Algorithm 1:

Input: Random number $\mathcal{R}$,Value to be Matched $\mathcal{V}$, shared information Ψ, Higher and Lower bits for Coefficient $\Omega_{high}, \Omega_{low}$ respectively

Output: Final evaluated Result Δ

$\mathcal{R} \leftarrow GenerateRandomNumber()$
$\mathcal{V} \leftarrow ValuetobeMatched$
//Create polynomial $P(x)$
$P(x) = (X - R)(X - (R + \mathcal{V}))$
//Select coefficient to share
if $(lowercofficient)$ **then**
  $\Psi = r$
  $send(\Omega_{Low}, \Psi)$
else
  $\Psi = r + \mathcal{V}$
  $send(\Omega_{high}, \Psi)$
//Receive value and coefficient marker from other party
receive(Ψ,Ω)
//evaluate $P(x)$ on Ψ
eval$_1$=$P(x) \leftarrow \Psi$
if (Ω_{low}) **then**
  eval$_2$=$P(x) \leftarrow (\Psi + \mathcal{V})$
else
  eval$_2$=$P(x) \leftarrow (\Psi - \mathcal{V})$
Δ=eval$_1$+eval$_2$
return Δ

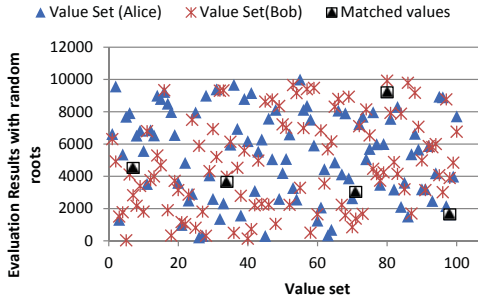

Figure 4: Access structure on encrypted data

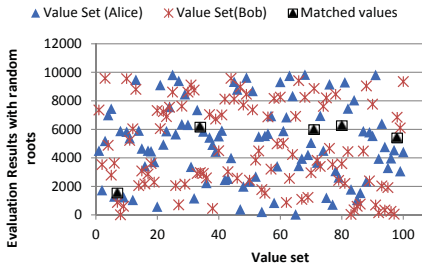

Figure 5: Polynomial evaluation results

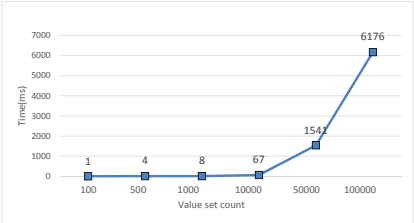

Figure 6: System Workflow

5. DISCUSSION

In this section we will talk about two threat models followed by system limitations.

5.1 Threat Model: eavesdropping

The model has been evaluated for end to end communication between two parties. Considering Danial as an eavesdropper capable of intercepting the network traffic between Alice and Bob, following information will be revealed to him.

Sno	$x \xrightarrow{communicates} y$
1	Alice $\xrightarrow{\sum_{i=1}^{n}\{\mathcal{E}_A(k_{pub},(\Omega_{low}^{high},\Psi_i))\}}$ Bob
2	Bob $\xrightarrow{\sum_{i=1}^{n}\{\mathcal{E}_A(k_{pub},(\Omega_{low}^{high},\Psi_i))\}}$ Alice
3	Alice $\xrightarrow{\sum_{i=1}^{n}\{\mathcal{E}_A(k_{pub},\Delta_i)\}}$ Bob
4	Bob $\xrightarrow{\sum_{i=1}^{n}\{\mathcal{E}_A(k_{pub},\Delta_i)\}}$ Alice

In order to find out common values, the total communication that takes place between Alice and Bob is from serial 1 to 4. In worst case scenario we assume that Danial intercept all of it and tries to learn anything useful from it.

Without trying to break into the asymmetric keys, Danial first tries to learn the magnitude of similarities between Alice and Bob's customers by simply observing traffic pattern. Due to Asymmetric encryption, the encrypted signature from serial 1 and 4 will be different within the intercepted traffic.If Danial manages to modify the original messages and replay them in such a way that both parties cannot decrypt them and have to execute the PM process again, then the repeated intercepted traffic will again end up in a unique pattern giving no useful clue to adversary. This effect is due to the random roots and it holds even in presence of symmetric encryption too. In both cases, if the keys are compromised and Danial discovers the underlying scheme that how address translation is made, still he will end up with random values. In worst case scenario he will only learn the magnitude of similarities and not the exact identities

5.2 Threat Model: in presence of semi-trusted third party

In PM, at least one of the communicating party will learn the matching output at first. In order to reveal results to both parties Alice and Bob have decided to use the services of semi-trusted third party (TTP) owned by Eve. Here we are not considering that computation for matching will also take place by the TTP. Eve as TTP, is supposed to execute the protocol in an honest way but for its curiosity tries to learn additional information other than set cardinality. In presence of TTP, resepective output of equation 3 and equation 5 by Alice and Bob will pass through and delivered by Eve. After acquiring these values, the same limitation of learning exact values holds for TTP as for an eavesdropper.

5.3 Limitations

The proposed model creates a layer of randomization between the original value and any encryption methodology. The flexibility of replacing original values with random roots makes it possible to generate different encrypted signature for same input again and again. The cost for comparison grows in $O(n * m)$ where n and m are numbers of values to be matched. In *ReSet*, we accommodate this complexity because values in set Δ are generated without encryption.

6. FUTURE WORK

In this paper we have proposed a methodology to find out commonalities between two sets with finite values. For efficient computing the idea is based on sharing of polynomial random roots on behalf of original values. To avoid the computational complexity with very large number of values to be matched, we will optimize the proposed methodology. Further, we will also provide a solution to perform computational task by TTP rather doing it on commodity hardware.

7. CONCLUSION

Private matching has attracted lot more attention in recent years due to large number of applications and their privacy concern while exchanging information with each other. Encryption and hashing are widely adopted standards for PM and usually applied directly on the original values. Similar values when underwent the hashing or encryption with same key outputs a value that never changes for repetitive matchings. To add randomness in these outputs, the proposed methodology works efficiently yet revealing only the

common values to the communicating parties. This additional layer is least complex on computation as it works prior to encryption or hashing.

Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency)" (NIPA-2014-(H0301-14-1003) and by the Industrial Core Technology Development Program (10049079), Develop of mining core technology exploiting personal big data) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea)

8. REFERENCES

[1] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 86–97. ACM, 2003.

[2] M. Ahmad, Z. Pervez, Y. Yoon, B. H. Kang, and S. Lee. Reflection: A lightweight protocol for private matching. In *Signal-Image Technology & Internet-Based Systems (SITIS), 2013 International Conference on*, pages 673–678. IEEE, 2013.

[3] Q. Dai and R. J. Kauffman. Business models for internet-based e-procurement systems and b2b electronic markets: an exploratory assessment. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2001.

[4] E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and private computation of cardinality of set intersection and union. In *Cryptology and Network Security*, pages 218–231. Springer, 2012.

[5] C. Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284. Springer, 2006.

[7] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology-EUROCRYPT 2004*, pages 1–19. Springer, 2004.

[8] K. Frikken. Privacy-preserving set union. In *Applied Cryptography and Network Security*, pages 237–252. Springer, 2007.

[9] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge university press, 2009.

[10] C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *Theory of Cryptography*, pages 155–175. Springer, 2008.

[11] C. Hazay and K. Nissim. Efficient set operations in the presence of malicious adversaries. *Journal of cryptology*, 25(3):383–433, 2012.

[12] J. Hong, J. W. Kim, J. Kim, K. Park, and J. H. Cheon. Constant-round privacy preserving multiset union. *IACR Cryptology ePrint Archive*, 2011:138, 2011.

[13] S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *Theory of Cryptography*, pages 577–594. Springer, 2009.

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

[15] M. Li, N. Cao, S. Yu, and W. Lou. Findu: Privacy-preserving personal profile matching in mobile social networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 2435–2443. IEEE, 2011.

[16] Y. Li, J. Tygar, and J. M. Hellerstein. Private matching. In *Computer Security in the 21st Century*, pages 25–50. Springer, 2005.

[17] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptologyâĂŤEUROCRYPTâĂŹ99*, pages 223–238. Springer, 1999.

[18] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

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

[20] R. Zhang, Y. Zhang, J. Sun, and G. Yan. Fine-grained private matching for proximity-based mobile social networking. In *INFOCOM, 2012 Proceedings IEEE*, pages 1969–1977. IEEE, 2012.