PRIVATE SET INTERSECTION -- STATE OF THE ART

by Andrada Ciulei,
Andrei Miron,
Stefania Andries,
Marian Cretu

BRIEFING:

Private Set Intersection is a cryptographic protocol which allows you to compute the intersection of two datasets without revealing any data, excepting the intersection columns. This has a lot of applications, according to [6], such that: privacy-preserving data mining, human genome research, homeland security, social networks, location sharing, etc. This protocol is crucial, to adhere to international standards of privacy and privacy laws.

MAIN CRYPTOGRAPHIC PROBLEM:

Let's note two sets of elements X and Y, and two communication entities: the sender and the receiver. Each set belongs to one entity.

INPUT: X, Y two datasets, X belongs to sender, Y to receiver
OUTPUT: Receiver: X ⋂Y
Sender: Ø

STATE OF THE ART:

In [6], there is described a PSI protocol which is declared that is very suitable for Big Data applications. This protocol is based on a special base procedure, called **Oblivious Transfer (OT),** which is related by the main authors in the domain as the most efficient protocol algorithm to solve the PSI problem. The inefficiency problem appears in communication, which is critical when we talk about large datasets and large security parameters, on a semi-honest model. Also, the authors are using some additional data structures, such as **Bloom Filters** and its special version **Garbled Bloom Filters**. The results are computed using the following specifications:

The server is a Mac Pro with 2 Intel E5645 6-core 2.4GHz CPUs, 32 GB RAM, and runs Mac OS X 10.8. The client is a Macbook Pro laptop with an Intel 2720QM quad-core 2.2 GHz CPU, 16 GB RAM, and runs Mac OS X 10.7

For the highest security level (256-bit security), the OT procedure, which is most costly, takes almost 1600-1700 s, but for smaller security parameters, the time is lower(100 s) -- in pipelined mode, and parallel mode, for 256-security level, it takes 300 seconds. [to see 6, Figure 4]. Also, some tests compare this algorithm for Java and C, for 80 and 256 security levels, and there are bigger improvements against earlier protocols (*DeCristofaro's*), and C time is half of Java running time. There is a promise for Big Data, and this option should be taken into account.

In [8], there is suggested an alternative of using Oblivious Transfer, by calling the Fully Homomorphic Encryption. To define exactly as in [8], *Fully homomorphic encryption is a powerful cryptographic primitive that allows arithmetic circuits to be evaluated directly on encrypted data, as opposed to having to decrypt the data first.* There are multiple ways of optimizing the brute basic protocol, such that HASHING, BATCHING, CUCKOO HASHING, PERMUTATION-BASED HASHING, etc. There is presented a restriction, which implies that the receiver set must have a negligible size than the sender's one. The results measured are listed below (Table [3] from 8)

Parameters			Optim.		Running time (seconds)									
N_X	N_Y	FHE parameters	α	ℓ	Sender pre-processing				Sender online				Receiver	
					$T=1$	4	16	64	1	4	16	64	Enc.	Dec.
2^{24}	11041	SEAL16384-2	256	1	72.2	18.0	6.2	3.0	42.2	14.4	7.1	5.6	0.3	10.3
			128	2	70.9	19.1	6.3	3.1	38.9	15.6	9.8	9.1	0.5	5.1
		SEAL16384-1	64	3	76.8	20.6	6.7	3.3	41.1	21.6	16.2	16.9	0.9	2.6
	5535	SEAL8192-1	256	1	64.1	17.9	5.5	2.7	36.0	11.8	6.3	5.5	0.2	4.9
			128	2	71.2	18.5	6.3	2.9	36.1	14.2	9.6	9.2	0.3	2.4
			64	3	80.4	21.5	6.7	3.2	41.9	21.5	17.7	17.7	0.5	1.2
2^{20}	11041	SEAL16384-1	128	1	9.1	2.5	1.0	0.5	8.0	2.6	1.2	1.1	0.2	5.1
			64	2	6.9	2.0	0.8	0.4	5.2	1.8	1.1	1.0	0.3	2.7
			32	3	6.4	1.7	0.9	0.6	4.5	2.1	1.3	1.5	0.7	1.3
	5535	SEAL8192-2	128	1	5.1	1.4	0.6	0.4	4.2	1.5	0.8	0.7	0.1	1.9
			64	2	4.4	1.2	0.6	0.3	3.4	1.7	0.7	1.0	0.2	1.0
			32	3	4.3	1.2	0.5	—	3.6	1.4	1.5	—	0.3	0.5
2^{16}	11041	SEAL16384-3	16	1	1.2	0.3	0.2	—	1.3	0.6	0.6	—	0.2	0.5
			8	2	1.0	0.3	0.2	—	1.5	1.2	1.3	—	0.3	0.3
			4	3	0.9	0.3	—	—	1.9	1.7	—	—	0.5	0.1
	5535	SEAL8192-2	32	1	0.9	0.3	0.2	—	0.9	0.4	0.3	—	0.1	0.5
			16	2	0.7	0.2	0.1	—	0.7	0.3	0.3	—	0.1	0.2
			8	3	0.6	0.2	—	—	0.7	0.5	—	—	0.3	0.1

Table 3: Running time in seconds for our protocol with $T \in \{1, 4, 16, 64\}$ threads; $\lambda = 40$, $\sigma = 32$, $h = 3$. Since we implemented multi-threading by dividing the α partitions evenly between threads, having $T > \alpha$ offers no performance benefit. These cases are denoted by "—" in the table.

In [1], Emiliano De Cristofaro and Gene Tsudik suggest more variants of Private Set Intersection and Authorize Private Set Intersection (PSI and APSI) protocols based on Privacy-preserving Policy-based Information Transfer (PPIT) with RSA setting.
A comparison of these variants is given in Table 3. Among the authorized protocols, the one which incurs linear computation (presented in Figure 2) performs better than the one based on PPIT-RSA (Figure 1) when both, client and server present sets with bigger lengths. For the PSI protocols, significant results are given by the blind RSA-based protocol (Figure 4) and the one which dismisses the RSA setting (Figure 3); choosing between them depending on the importance of client or server performance. These protocols were implemented in ANSI C, using the OpenSSL library and tested on a Dell Precision PC on a 2.33GHz CPU and 8GB RAM.

In [3], was raised the question if the generic protocols with garbled circuits can be better than the custom protocols and the main comparison was made with the protocols proposed in [1]. In Table 1 are presented the five protocol approaches with their number of non-free gates. The authors mentioned Fairplay and TASTY implementations for garbled circuits. Experiments ["two standard desktop computers (Dell Intel R CoreTM 2 Duo E8400 3GHz) connected through a 100 Mbps LAN"] show that for smaller element space, the BWA

protocol (Bitwise AND) is the best, while for larger sets SCS-WN protocol (Set-Compare-Shuffle with Waksman Network) is a better option. Their results show that protocols based on generic secure computation can offer performance that is competitive with the best-known custom protocols. The authors offer their framework implementation under an open-source license at http://www.MightBeEvil.com.

[4] comes as a new approach for designing PSI protocols, which enables to reduce the length of items mapped to bins while ensuring that no collisions occur. The resulting protocol is up to 5 times faster than the previously best Sort-Compare-Shuffle circuit of Huang et al.

Phasing can be also applied to the OT-based PSI protocol of Pinkas et al. (USENIX Security 2014), which is the fastest PSI protocol to date. Together with additional improvements that reduce the computation complexity by a logarithmic factor, the resulting protocol improves run-time by a factor of up to 20 and can also have similar communication overhead as the previously best PSI protocol in that respect. The new protocol is only moderately less efficient than an insecure PSI protocol that is currently used by real-world applications and is, therefore, the first secure PSI protocol that is scalable to the demands and the constraints of the current real-world setting.

These improvements are obtained by generalizing the hashing approach of [B. Pinkas, T. Schneider, and M. Zohner. Faster private set intersection based on OT extension. In USENIX Security Symposium, pages 797–812. USENIX, 2014] and applying it to generic secure computation-based PSI protocols. The hash function in [B. Pinkas, T. Schneider, and M. Zohner. Faster private set intersection based on OT extension. In USENIX Security Symposium, pages 797–812. USENIX, 2014] is replaced by a permutation which enables to reduce the bit-length of internal representations.

Protocol	LAN				WAN			
	$n=2^8$	$n=2^{12}$	$n=2^{16}$	$n=2^{20}$	$n=2^8$	$n=2^{12}$	$n=2^{16}$	$n=2^{20}$
Yao's garbled circuits [25]								
SCS [12]	309	3,464	63,857	—	2,878	20,184	301,512	—
Circuit-Phasing §5	376	3,154	39,785	—	3,004	17,133	178,865	—
Goldreich-Micali-Wigderson [11]								
SCS [12]	626	2,175	38,727	—	11,870	21,030	218,378	—
Circuit-Phasing §5	280	1,290	14,149	168,397	2,681	8,681	81,534	846,510

Table 5: Run-time in ms for generic secure PSI protocols in the LAN and WAN setting on $\sigma = 32$-bit elements.

Protocol	$n=2^8$	$n=2^{12}$	$n=2^{16}$	$n=2^{20}$	Asymptotic
Number of AND gates					
SCS [12]	229,120	5,238,784	107,479,009	*2,000,000,000	$\sigma(3n\log_2(n)+4n)$
Circuit-Phasing §5	297,852	3,946,776	49,964,540	600,833,968	$(\sigma-\log_2(n)-2)(6(1+\varepsilon)n\frac{\ln n}{\ln\ln n}+sn)$
Communication in MB for Yao's garbled circuits [25] and GMW [11]					
SCS [12]	7	169	3,485	*64,850	$2\kappa\sigma(3n\log_2(n)+4n)$
Circuit-Phasing §5	9	122	1,550	18,736	$2\kappa(\sigma-\log_2(n)-2)(6(1+\varepsilon)n\frac{\ln n}{\ln\ln n}+sn)$
Number of communication rounds for GMW [11]					
SCS [12]	85	121	157	193	$(\log_2(\sigma)+4)\log_2(2n)+4$
Circuit-Phasing §5	5	5	5	5	$\log_2(\sigma)$

Table 6: Number of AND gates, concrete communication in MB, round complexity, and failure probability for generic secure PSI protocols on $\sigma = 32$-bit elements. Numbers with * are estimated.

Setting	LAN					WAN			
Protocol	$n=2^8$	$n=2^{12}$	$n=2^{16}$	$n=2^{20}$	$n=2^{24}$	$n=2^8$	$n=2^{12}$	$n=2^{16}$	$n=2^{20}$
Naive Hashing$^{(*)}$ §3.1	1	4	48	712	13,665	97	111	558	3,538
Server-Aided$^{(*)}$ [15]	1	5	78	1,250	20,053	198	548	2,024	7,737
DH-based ECC [18]	231	3,238	51,380	818,318	13,065,904	628	10,158	161,850	2,584,212
Bit-length $\sigma = 32$-bit									
OT PSI [22]	184	216	3,681	62,048	929,685	957	1,820	9,556	157,332
OT-Phasing §6	179	202	437	4,260	46,631	912	1,590	3,065	14,567
Bit-length $\sigma = 64$-bit									
OT PSI [22]	201	485	7,302	125,697	—	977	1,873	18,998	315,115
OT-Phasing §6	180	240	865	10,128	137,036	1,010	1,780	5,009	29,387
Bit-length $\sigma = 128$-bit									
OT PSI [22]	201	485	8,478	155,051	—	980	1,879	21,273	392,265
OT-Phasing §6	181	240	915	13,485	204,593	1,010	1,780	5,536	37,422

Table 7: Run-time in ms for protocols with $n = n_1 = n_2$ elements. (Protocols with $^{(*)}$ are in a different security model.)

Protocol	$n=2^8$	$n=2^{12}$	$n=2^{16}$	$n=2^{20}$	$n=2^{24}$	Asymptotic [bit]
Naive Hashing$^{(*)}$ §3.1	0.01	0.03	0.56	10.0	176.0	$n_1\ell$
Server-Aided$^{(*)}$ [15]	0.01	0.16	2.5	40.0	640.0	$(n_1+n_2+\lvert X\cap Y\rvert)\kappa$
DH-based ECC [18]	0.02	0.28	4.56	74.0	1,200.0	$(n_1+n_2)\varphi+n_1\ell$
Bit-length $\sigma = 32$-bit						
OT PSI [22]	0.09	1.39	22.58	367.20	5,971.20	$0.6n_2\sigma\kappa+6n_1\ell$
OT-Phasing §6	0.06	0.73	8.74	136.8	1,494.4	$2.4n_2\kappa(\lceil\frac{\sigma-\lfloor\log_2(1.2n_2)\rfloor}{8}\rceil)+(3+s)n_1\ell$
Bit-length $\sigma = 64$-bit						
OT PSI [22]	0.14	2.59	41.78	674.4	10,886.4	$0.6n_2\kappa*min(\ell,\sigma)+6n_1\ell$
OT-Phasing §6	0.09	1.34	18.34	290.4	3,952.0	$2.4n_2\kappa(\lceil\frac{min(\ell,\sigma)-\log_2(n_2)}{8}\rceil)+(3+s)n_1\ell$
Bit-length $\sigma = 128$-bit						
OT PSI [22]	0.14	2.59	46.58	828.0	14,572.8	$0.6n_2\ell\kappa+6n_1\ell$
OT-Phasing §6	0.09	1.34	20.74	367.2	5,795.2	$2.4n_2\kappa(\lceil\frac{\ell-\log_2(n_2)}{8}\rceil)+(3+s)n_1\ell$

Table 8: Communication in MB for PSI protocols with $n = n_1 = n_2$ elements. $\ell = \lambda + \log_2(n_1) + \log_2(n_2)$. Assuming intersection of size $1/2 \cdot n$ for TTP-based protocol. (Protocols with $^{(*)}$ are in a different security model.)

Although generic robust methods are based on Yao's general two-party computations [Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)] are sufficient for computing any two-party functionality, in [2] is presented a fully black-box construction assuming the existence of a homomorphic encryption scheme that is more efficient: it reduces the communication complexity from Ω(m·n) (where n is the input size of the party that receives output and m is the input size of the other party) to O(mk^2 log^2 n + kn) (where k is the security parameter). Also, the number of exponentiations needed by the protocol presented in the article increases only by a poly-logarithmic (i.e. a k^2

log^2 n) factor in comparison to the number of exponentiations required by the semi-honest protocol of [Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)] For all of these, the encryption is only required to possess a few natural properties (and satisfied by known homomorphic encryption schemes, e.g., based on DDH).

In [5] E DeCristofaro, J Kim, and G Tsudik propose a modified version for the Authorized Private Set Intersection in the malicious model, under RSA and DDH assumptions. For the authorization step, they use a CA, a trusted third party that authorizes client input and doesn't have any involvement in the computation of the intersection. Besides the modified APSI, they also presented a modified version of the PSI protocol, secure in the semi-honest model under the One-More-Gap-DH assumption. Proposed protocols offer better efficiency than prior work. In particular, the APSI protocol is the first technique to achieve linear computational complexity.

In [7] Benny Pinkas, Thomas Schneider, and Michael Zohner implemented the previously known protocols for PSI and optimized them, and created a new protocol based on OT extension and hashing. In regards to optimization, they improved the circuit-based protocols, such as GMW by ~40%, and bloom-filter protocol, the run-time being reduced by ~55-60%, both optimizations being done with the help of the OT extension.

The new PSI protocol is based on the most efficient OT extension techniques, in particular, the random OT functionality and the efficient 1-out-of-N OT. This protocol scales very efficiently with an increasing set size. The protocol can be simply extended to perform PSI between sets X and Y by applying the parallel comparison protocol for each element y from Y. The overhead of the protocol can be greatly improved using hashing. The results show that the OT-based protocols have the lowest runtime on a fast network, while the proposed set inclusion protocol achieves both the most efficient runtime and very low communication overhead.

The choice of the preferable PSI protocol, even with the great results obtained by the new protocol in mind, depends on the application scenario. If communication is the bottleneck and computation is vast, then the DH-based PSI using ECC is the most favorable and simple to implement. The circuit-based protocols are based on generic secure computation techniques and can be easily modified to compute more complex variants of PSI. The random garbled Bloom filter protocol doesn't get a big increase in runtime for bigger inputs.

RELEVANT ARTICLES AND BOOKS:

1. Practical Private Set Intersection Protocols with Linear Computational and Bandwidth Complexity, Emiliano De Cristofaro and Gene Tsudik
   https://eprint.iacr.org/2009/491.pdf

2. Efficient Robust Private Set Intersection; Dana Dachman-Soled1, Tal Malkin1, Mariana Raykova1, and Moti Yung
   https://user.eng.umd.edu/~danadach/MyPapers/set-int.pdf
3. Private set intersection: Are garbled circuits better than custom protocols?; Yan Huang, David Evans, Jonathan Katz
   https://www.cs.umd.edu/~jkatz/papers/psi.pdf
4. Phasing: Private set intersection using permutation-based hashing; B Pinkas, T Schneider, G Segev, M Zohner
   https://eprint.iacr.org/2015/634.pdf
5. Linear-complexity private set intersection protocols secure in malicious model; E De Cristofaro, J Kim, G Tsudik
   https://eprint.iacr.org/2010/469.pdf
6. When private set intersection meets big data: an efficient and scalable protocol; C Dong, L Chen, Z Wen
   https://eprint.iacr.org/2013/515.pdf
7. Faster private set intersection based on {OT} extension; B Pinkas, T Schneider, M Zohner
   https://eprint.iacr.org/2014/447.pdf
8. Fast private set intersection from homomorphic encryption; H Chen, K Laine, P Rindal
   https://eprint.iacr.org/2017/299.pdf
9. PSI from PaXoS: Fast, Malicious Private Set Intersection; Benny Pinkas, Mike Rosulek, Ni Trieu, Avishay Yanai
   https://eprint.iacr.org/2020/193.pdf

RELEVANT LINKS:

Excluding the articles listed above, some useful links are these:

1. https://cyber.biu.ac.il/wp-content/uploads/2017/01/15.pdf?fbclid=IwAR3N6iqnJDXjlWXtXPBRplzH41czY50pWiLfvlQfZLM3f8fFo0nUlvbd1U4
2. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pinkas
3. https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-pinkas.pdf

IMPORTANT NAMES IN THE FIELD, RESEARCH TEAMS:

1. For the *Phasing: Private Set Intersection using Permutation-based Hashing* and *Faster Private Set Intersection based on OT Extension* studies some important names are:
   - Benny Pinkas (Bar Ilan University, Israel)
   - Thomas Schneider (TU Darmstadt, Germany)
   - Michael Zohner (TU Darmstadt, Germany)

2. For the *Phasing: Private Set Intersection using Permutation-based Hashing* study together with those mentioned above was:
   - Gil Segev (Hebrew University, Israel)

3. For the *Efficient Robust Private Set Intersection* study:
    - Dana Dachman-Soled (currently a postdoc at Microsoft Research New England. She received her Ph.D. in Computer Science from Columbia University in 2011 under the supervision of Professor Tal Malkin)
    - Tal Malkin (Associate Professor of Computer Science at Columbia University, where she directs the Cryptography Lab)
    - Mariana Raykova (Ph.D. student at Columbia University who is jointly advised by Tal Malkin and Steven Bellovin)
    - Moti Yung (Research Scientist with Google Inc. and an Adjunct Senior Research Faculty at the Computer Science Department, Columbia University)

4. For studying the *Practical Private Set Intersection Protocols with Linear Computational and Bandwidth Complexity* and *Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model:*
    - Emiliano De Cristofaro (University of California, Irvine)
    - Gene Tsudik (University of California, Irvine)

5. In the *Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model* study together with the above two mentioned was involved:
    - Jihye Kim (Department of Mathematical Sciences, Seoul National University)

6. For the *Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?* study:
    - Yan Huang (University of Virginia)
    - David Evans (University of Virginia)
    - Jonathan Katz (University of Maryland)

7. For *When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol* study:
    - Changyu Dong (Dept. of Computer and Information Sciences. University of Strathclyde)
    - Liqun Chen (Cloud & Security Lab, Hewlett Packard Labs)
    - Zikai Wen (Dept. of Computer and Information Sciences, University of Strathclyde)

8. For studying *Fast Private Set Intersection from Homomorphic Encryption* we mention:
    - Hao Chen (Microsoft Research, Redmond, WA, USA)
    - Kim Laine (Microsoft Research, Redmond, WA, USA)
    - Peter Rindal (Oregon State University, Corvallis, OR, USA)

9. For the *PSI from PaXoS: Fast, Malicious Private Set Intersection* study:
    - Benny Pinkas (also mentioned above)
    - Mike Rosulek (Associate Professor at Oregon State University)

- Ni Trieu (Assistant Professor in the Ira A. Fulton Schools of Engineering at Arizona State University)
- Avishay Yanai (Ph.D. student at the Cyber Center in Bar Ilan University, Israel)

RESOURCES AND TOOLS AVAILABLE:

- An OT-Phasing implementation is available online at: https://github.com/encryptogroup/PSI

- Circuit-Phasing implementation is available as part of the ABY framework(A framework for efficient mixed-protocol secure two-party computation.) of D. Demmler, T. Schneider, and M. Zohner at: https://github.com/encryptogroup/ABY

- Fast Secure Computation Using Garbled Circuits framework: http://www.mightbeevil.com/framework/

Python: NumPy, SciPy, PyCryptodome, Cryptography, PyNaCl, SymPy,
C/C++: OpenSSL, CryptoPP
Java: JCA, Bouncy Castle, Web3j

IDE: PyCharm, Intellij, Eclipse, VS