Manuscript received January 24, 2018; revised March, 2018. DOI: 10.24425/119367

Confidential Greedy Graph Algorithm

Daniel Waszkiewicz, Aleksandra Horubala, Piotr Sapiecha, and Michal Andrzejczak

***Abstract*—Confidential algorithm for the approximate graph vertex covering problem is presented in this article. It can preserve privacy of data at every stage of the computation, which is very important in context of cloud computing. Security of our solution is based on fully homomorphic encryption scheme. The time complexity and the security aspects of considered algorithm are described.**

***Keywords*—cryptography, fully homomorphic encryption, confidential graph algorithm, cloud computing**

I. Introduction

There are some scenarios where outsourcing computation to a cloud service makes sense from a practical and rational economic point of view. Namely, when data is collected or uploaded from many diverse sources or parties, an online service can host the collection, storage, and computation of and on this data without requiring interaction with the data owner. Security is a major barrier to the widespread use of cloud computing architectures. One of solutions is application of property preserving encryption which include deterministic encryption and order preserving encryption to ensure confidentiality of data [1][2]. The problem of modeling leakage of information in those schemes led to some practical attacks on cloud stored data bases [3]. Another popular approach of securing data is usage of multi-party computations protocols, which efficiency increased in lasts years [4]. The difficulty of application of *MPC* is preserving majority of honest or semi-honest parties in highly distributed system.

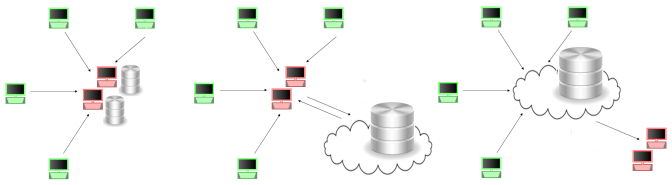

Fig. 1. Paradigms of cloud computations: (left) offline server storage and computation, (middle) cloud servers storage and offline server computation, (right) cloud server storage and computation.

In 2009, Craig Gentry in his doctoral dissertation proposed first fully homomorphic encryption scheme [5]. This system allows to execute, in a confidential manner, addition and multiplication operations on the ciphertexts. After development of lattice-based cryptography and introducing *RLWE* (Ring Learning With Errors) problem a lot of new, more efficient, fully homomorphic encryption schemes have been proposed [6][7][8].

In literature exists application of homomorphic encryption in privacy preserving graph algorithm. The GRECS protocol solves approximate shortest path problem in efficient and secure way [9]. The disadvantage of this solution appears in necessary re-encryption of whole data base in case of any modification of data.

Our representation of graph allows for dynamic modification of data without need of re-encryption and computation of confidential approximate graph vertex cover algorithm.

A. Approximate vertex cover problem

The decision version of vertex cover problem was one of Karp's 21 NP-complete problems and is therefore a classical NP-complete problem in computational complexity theory [10]. The optimization version (NP-hard) can be defined as follows:

Input: Graph G,

Output: $k = min\{|C| \ : \ C-$ vertex cover of a graph $G\}$, where $|C|$ is cardinality of set C.

In 70's F. Gavril and M. Yannakakis independently invented approximate algorithm solving vertex cover problem with constant approximate factor equals to 2. Computational complexity of the Algorithm 1 is $\Theta(n^2)$, where n is number of vertices.

Input: Graph $G = (V, E)$;
Result: Vertex cover C of graph G;
$C \leftarrow \emptyset$;
while $E \neq \emptyset$ **do**
  $(u, v) \leftarrow$ random edge in E;
  $C \leftarrow C \cup \{u, v\}$;
  delete all edges incident with u, v in E;
end

Algorithm 1: Approximate vertex cover algorithm.

B. Fully homomorphic encryption

The security of protocol is mainly based on correctness of fully homomorphic scheme (*FHE*). The biggest advantage of those cryptosystems is ability to evaluate three of four basic arithmetic operations: addition, multiplication and subtraction:

$$Enc_k(m_1) \oplus Enc_k(m_2) \simeq Enc_k(m_1 + m_2),$$

$$Enc_k(m_1) \odot Enc_k(m_2) \simeq Enc_k(m_1 \cdot m_2),$$

$$Enc_k(m_1) \oplus (Enc_k(-1) \odot Enc_k(m_2)) \simeq Enc_k(m_1 - m_2).$$

The very crucial fact, which has to be remembered during using fully homomorphic schemes is lack of operation of

D. Waszkiewicz, A. Horubala and P. Sapiecha are with Warsaw University of Technology, Poland (e-mail: d.waszkiewicz@tele.pw.edu.pl, aleksandra.horubala@gmail.com, p.sapiecha@krypton-polska.com).

M. Andrzejczak is with Military University of Technology in Warsaw, Poland (e-mail: michal.andrzejczak@wat.edu.pl).

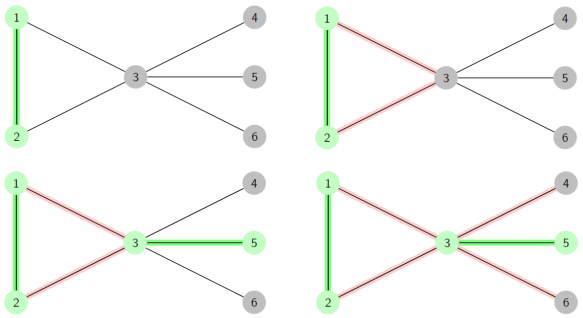

Fig. 2. Execution of Algorithm 1. Green edges are chosen to vertex cover. Red edges are deleted from graph.

division on ciphertexts. There are no known methods that can be used to solve this problem.

II. Confidential graph algorithm

Security definition (at high level):
No efficient adversary can learn any partial information about the graph, beyond what is explicitly allowed by the leakage function (bounds on maximum size of graph). This holds even for allowed modifications that are adversely influenced.

In our setting, we establish the security notion of confidential graph algorithms as follows:

Definition 1: The graph algorithm (L) taking as input encrypted graph $Enc_k(G = (V, E))$ is *(n,m)-Confidential* if any adversary (A), who has access to encrypted graph $Enc_k(G = (V, E))$, encryption oracle Enc_k and evaluations functions $\oplus, \odot$ can not learn any information about graph G other than maximum bounds on number of vertices and edges, which stand for n and m respectively.

$$Pr_{Enc_k(G)}[A(f) = 1] = negl(1^k)$$

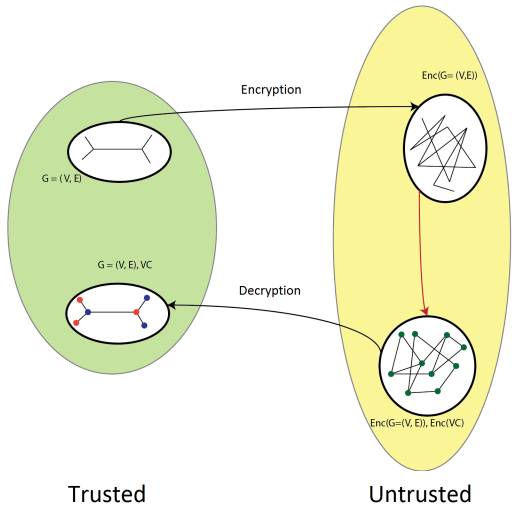

Fig. 3. Confidential graph algorithm.

The definition above captures the fact that, given the encrypted oracle and its view of the query protocol, an adversarial server cannot learn any information about the oracle beyond the leakage. The $\mathcal{L}(\Omega) = (n, m)$ where it is maximum bounds on number of vertices and edges respectively.

III. Data structure - Graph encoding

Our solution follows **Algorithm 1** for approximate vertex cover. The graph $G = (V, E)$ is encoded as matrix M_m^{2n+1}. Values n and m are bounds on the size of sets V and E respectively. Because m is only bound on $|E|$, there is $m-|E|$ rows in matrix M, which do not describe edges in graph G. If first value in row M_i is equal to 1 then this row describes edge. Next $2n$ values are encodings of two vertices.

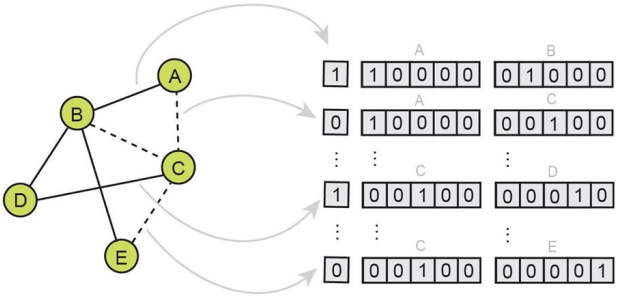

Fig. 4. Data structure - graph encoding.

After encoding graph G, all elements of matrix M are encrypted by fully homomorphic scheme $(Enc(M))$.

IV. An Algorithm

All the time during computation of the confidential algorithm matrix M has m rows and $2n + 1$ columns and only operations on the matrix are additions and multiplications. In description of algorithm we will use symbols of $\oplus, \odot$ as addition and multiplication of ciphertext in fully homomorphic schemes respectively.

Input: $M = Enc(M)$, n, m;
Result: $C = Enc(C)$, where C is vertex cover;
$C \leftarrow \emptyset$;
for $i = 1, \ldots, \lfloor \frac{n}{2} \rfloor$ **do**
  $M \leftarrow SamplingD_1(M, n, m, 0, m)$;
  $e \leftarrow M_1$;
  $C \leftarrow C \cup e$;
  $M \leftarrow RemoveIncidentEdges(M, n, m)$;
end
return C

Algorithm 2: Confidential approximate vertex cover algorithm.

In Algorithm 2 functions $SamplingD_1$ and $RemoveIncidentEdges$ are modifying representation of graph (matrix M) using fully homomorphic operations: addition and multiplication. First row (M_1) is added to set of encrypted edges C.

A. Emptiness of encrypted set

We can not check if all the values of first column are encryption of 1 without decryption. The solution is changing line 2 of Algorithm 1 to loop FOR from 1 to $\lfloor \frac{n}{2} \rfloor$, which is the number of iterations in the worst case scenario.

B. Blind sampling

To ensure correctness of algorithm in case of introducing new loop FOR, we have to provide sampling procedure $\mathcal{D}$ over encrypted matrix $M = Enc_k(\delta(G = (V, E)))$, which outputs rows describing edge G if graph is not empty.

$$Pr[\mathcal{D}(M) \in E | E \neq \emptyset] = 1$$

Problem can be reformulated as sampling over encrypted vector $v \in \{0,1\}^k$, which can be represented as:

$$Pr[\mathcal{D}(Enc(v_1, \ldots, v_k)) = i \wedge v_i = Enc(1) \mid v \neq 0^k] = 1$$

If Enc scheme is fully homomorphic encryption scheme, then we can compute tournament method for finding maximum in array in logarithmic time. Finally rows in matrix M gets new value $M_{i,1} = 0, 1$ which describe if i row is defining the edge in graph G.

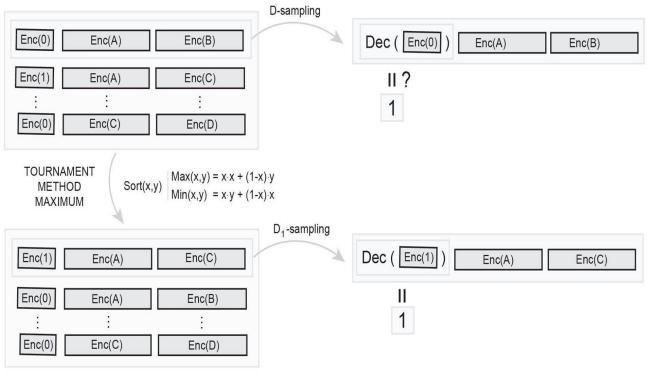

Fig. 5. $SamplingD_1$ method.

Procedure $SamplingD_1$ works by inserting row which describes edge (if graph is not empty) in first row of matrix M.

Input: $M = Enc(M)$, n, m, i, j;
Result: M;
for $k = 1, \ldots, (2n + 1)$ **do**
  $x \leftarrow M_{i,k}$;
  $y \leftarrow M_{j,k}$;
  $yBx \leftarrow M_{i,1} \odot y$;
  $xBx \leftarrow M_{i,1} \odot x$;
  $M_{i,k} = xBx \oplus y \oplus (yBx \odot Enc(-1))$;
  $M_{j,k} = yBx \oplus x \oplus (xBx \odot Enc(-1))$;
end
return M

Algorithm 3: $Swap$.

In Algorithm 3 operations $\oplus, \odot$ are fully homomorphic addition and multiplication of ciphertext. Complexity of $Swap$ is $O(n)^*$.

Input: $M = Enc(M)$, n, m, k, l;
Result: M;
if $k + 1 = l$ **then**
  $M \leftarrow Swap(M, n, m, k, l)$;
  return M
end
$M \leftarrow SamplingD_1(M, n, m, k, \lfloor \frac{k+l}{2} \rfloor)$;
$M \leftarrow SamplingD_1(M, n, m, \lfloor \frac{k+l}{2} \rfloor + 1, l)$;
$M \leftarrow Swap(M, n, m, k, \lfloor \frac{k+l}{2} \rfloor + 1)$;
return M

Algorithm 4: $SamplingD_1$.

Complexity of Algorithm 4 is $O(nm)$.

C. Neighbourhood of encrypted vertex

After sampling random edge e in graph G we have to remove all edges incident with e. Procedure is computed by checking if any row in M represents edge with the same vertex as edge e. Due to probabilistic nature of fully homomorphic encryption scheme there is no answer if two ciphertexts c_1 and c_2 are encryptions of the same plaintext m without decryption. We update all rows of matrix M, but only for rows with incident edges values plaintext will change.

Solution is to encode vertices in orthonormal way with function $f : Enc(\delta(V)) \times Enc(\delta(V)) \rightarrow \{Enc(0), Enc(1)\}$. Encoding δ works by mapping vertices to the vector of length n, which is 0 on all values except $i - th$ with value 1, where i is predetermined number of vertex. The scalar function on two encrypted and encoded vertices is defined as:

$$f(Enc(\delta(u)),\ Enc(\delta(v))) \ = \ \bigoplus_{i=1}^{n} (Enc(\delta(u))_i \odot Enc(\delta(v))_i).$$

Now we can define algorithm, which will remove all incident edges in encrypted graph.

Input: $M = Enc(M)$, n, m;
Result: M;
$M_{1,1} \leftarrow Enc(0)$;
for $i = 2, \ldots, m$ **do**
  $x_1 \leftarrow f(M_{1,2:n+1},\ M_{j,2:n+1})$;
  $x_2 \leftarrow f(M_{1,n+2:2n+1},\ M_{j,2:n+1})$;
  $x_3 \leftarrow f(M_{1,2:n+1},\ M_{j,n+2:2n+1})$;
  $x_4 \leftarrow f(M_{1,n+2:2n+1},\ M_{j,n+2:2n+1})$;
  $M_{j,1} \leftarrow$
   $M_{j,1} \odot (Enc(1) \oplus (Enc(-1) \odot (x_1 \oplus x_2 \oplus x_3 \oplus x_4)))$;
end
return M

Algorithm 5: $RemoveIncidentEdges$.

Complexity of Algorithm 5 is $O(nm)$.

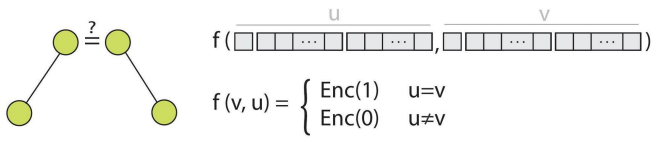

Fig. 6. Orthonormal vertex encoding.

V. Updates

Due to our data structure and fully homomorhic encryption scheme we can do updates of encrypted graph G without need of decryption.

A. Removing edges

We can remove edge $e = (u, v)$ from encrypted graph G given encryption of $e = (Enc(\delta(u)), Enc(\delta(v)))$.

Input: $M = Enc(M)$, n, m, $Enc(e = (u, v))$;
Result: M;
for $i = 1, \ldots, m$ **do**
  $x_1 \leftarrow f(Enc(\delta(u)),\ M_{i,2:n+1})$;
  $x_2 \leftarrow f(Enc(\delta(v)),\ M_{i,n+2:2n+1})$;
  for $i = 1, \ldots, 2n + 1$ **do**
    $M_{i,j} \leftarrow M_{i,j} \odot (Enc(1) \oplus (Enc(-1) \odot x_1 \odot x_2))$
  end
end
return *M*

Algorithm 6: $Remove$.

Complexity of Algorithm 6 is $O(nm)$.

B. Inserting edges

If $|E| < m$ then in matrix M exists row with value 0 in first column. We can put this row in the beginning of matrix M by modifying $SamplingD_1$ method (changing calling of function *Swap*).

Input: $M = Enc(M)$, n, m, k, l;
Result: M;
if $k + 1 = l$ **then**
  $M \leftarrow Swap(M, n, m, l, k)$;
  return *M*
end
$M \leftarrow SamplingD_0(M, n, m, k, \lfloor \frac{k+l}{2} \rfloor)$;
$M \leftarrow SamplingD_0(M, n, m, \lfloor \frac{k+l}{2} \rfloor + 1, l)$;
$M \leftarrow Swap(M, n, m, \lfloor \frac{k+l}{2} \rfloor + 1, k)$;
return *M*

Algorithm 7: $SamplingD_0$.

Input: $M = Enc(M)$, n, m, $Enc(e = (u, v))$;
Result: M;
$M \leftarrow SamplingD_0(M, n, m, 0, m)$;
$M_1 \leftarrow (Enc(1), Enc(\delta(u)), Enc(\delta(v)))$;
return *M*

Algorithm 8: $Insert$.

Complexity of Algoirhtm 8 is $O(nm)$.

VI. Complexity

While approximate vertex cover Algorithm 1 has complexity $O(n^2)$, we achieved good complexity for confidential Algorithm 2 equals to $O(n^2m)^*$. The star in our result means that butterfly operation is multiplication of two ciphertexts. This leads to question how complex is multiplications of two ciphertexts? It depends on used fully homomorphic scheme and its parameters. We consider Fan and Vercauteren's fully homomorphic scheme from 2012 [6], which is implemented in libraries NFLlib and SEAL [11][12]. Authors of *FV* showed how to adjusted parameters of their cryptosytem to correctly evaluate circuits with arbitrary multiplicity depth. For confidential algorithm depth is equal to:

$$L(n, m) = \lfloor \frac{n}{2} \rfloor \cdot (\lceil log_2(m) \rceil + 2).$$

In the *FV* cryptosytem single ciphertext is a pair of two polynomials and the computation depends on 4 parameters d, q, t, σ. To assure correctness of computation of arbitrary circuits of depth L following condition has to be met:

$$4 \cdot 2^{(d-1)\cdot L} \cdot (2^{d-1} + 1.25)^{L+1} \cdot t^{L-1} < \lfloor q/\sigma \rfloor,$$

where 2^d is a degree of polynomials of ciphertext, q is the size of ciphertext polynomials' coefficients, t is the size of plaintext coefficients polynomials and σ is the variance of distribution. Value λ in table I is the security parameter of *FV* cryptosystem.

TABLE I
Logarithms of parameters of FV cryptosystem for (n, m)-confidential approximate vertex cover algorithm.

(n, m)	L	$\lambda = 8$	$\lambda = 32$	$\lambda = 64$
$(6, 9)$	18	$(5, 4371, 3, 3)$	$(5, 4456, 3, 3)$	$(5, 4484, 3, 3)$
$(10, 15)$	30	$(6, 8064, 3, 3)$	$(6, 8219, 3, 3)$	$(6, 8272, 3, 3)$
$(20, 50)$	80	$(6, 36633, 3, 3)$	$(6, 37337, 3, 3)$	$(6, 37578, 3, 3)$
$(50, 100)$	250	$(7, 225978, 3, 3)$	$(7, 230324, 3, 3)$	$(7, 231809, 3, 3)$

A. Experimental results

We have run series of experiments of confidential approximate vertex cover algorithm. Algorithm 2 and *FV* cryptosystem were implemented in *Sage* and tested on PC with Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz and 32 GiB 2133 MHz RAM [13]. Results are shown in table II.

TABLE II
Run times of confidential approximate vertex cover algorithm.

(n, m)	L	$\lambda = 8$	$\lambda = 32$	$\lambda = 64$
$(6, 9)$	18	22.6079 s.	24.4759 s.	23.7941 s.
$(10, 15)$	30	490.6876 s.	529.9789 s.	565.2177 s.
$(20, 50)$	80	$\approx$ 12.34 h.	$\approx$ 12.36 h.	$\approx$ 12.37 h.
$(50, 100)$	250	$>$ 36 h.	$>$ 36 h.	$>$ 36 h.

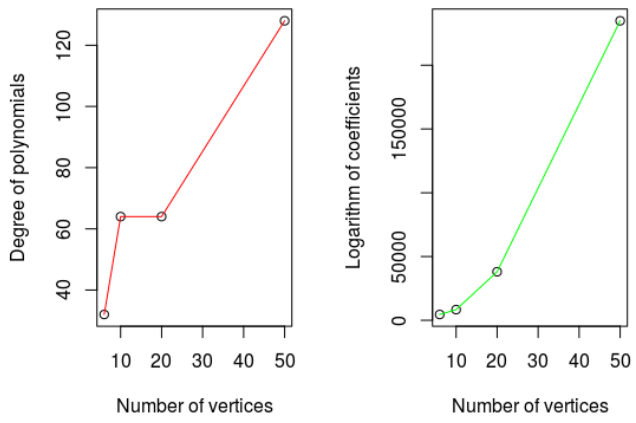

Fig. 7. Degree of polynomials and logarithm of coefficients to number of vertices.

VII. Conclusions

We proposed new confidential algorithm for evaluating vertex cover without leaking information about graph. Main advantage of this algorithm and data structure is possibility of updating encrypted graph without decryption. The area of further optimizations could be encoding of vertices to decrease size of data. Latest implementations of fully homomorphic schemes allow SIMD operations on ciphertexts [14]. While we do not see adjustment of SIMD operations to our solution, this modification could accelerate computation. Another field of optimizations could be FPGA implementetions of fully homomorphic schemes [15].

References

[1] X. Liao, S. Uluagac, and R. A. Beyah, "S-match: Verifiable privacy-preserving profile matching for mobile social services," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2014, pp. 287–298.

[2] S. Chatterjee and M. P. L. Das, "Property preserving symmetric encryption revisited," in *Advances in Cryptology – ASIACRYPT 2015*, T. Iwata and J. H. Cheon, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 658–682.

[3] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 644–655. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813651

[4] A. C. Yao, "Protocols for secure computations." Washington, DC, USA: IEEE Computer Society, 1982, pp. 160–164.

[5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[6] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptology ePrint Archive, Report 2012/144, 2012, http://eprint.iacr.org/2012/144.

[7] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: ACM, 2012, pp. 309–325. [Online]. Available: http://doi.acm.org/10.1145/2090236.2090262

[8] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '05. New York, NY, USA: ACM, 2005, pp. 84–93. [Online]. Available: http://doi.acm.org/10.1145/1060590.1060603

[9] X. Meng, S. Kamara, K. Nissim, and G. Kollios, "Grecs: Graph encryption for approximate shortest distance queries," Cryptology ePrint Archive, Report 2015/266, 2015, http://eprint.iacr.org/2015/266.

[10] R. M. Karp, *Reducibility among Combinatorial Problems*. Boston, MA: Springer US, 1972, pp. 85–103. [Online]. Available: http://dx.doi.org/10.1007/978-1-4684-2001-2_9

[11] C. Aguilar-Melchor, J. Barrier, S. Guelton, A. Guinet, M.-O. Killijian, and T. Lepoint, "NFLlib: NTT-based Fast Lattice Library," in *RSA Conference Cryptographers' Track* , San Francisco, United States, Feb. 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01242273

[12] K. Laine and R. Player, "Simple encrypted arithmetic library - seal (v2.0)," Tech. Rep., September 2016. [Online]. Available: https://www.microsoft.com/en-us/research/publication/simple-encrypted-arithmetic-library-seal-v2-0/

[13] W. Stein and D. Joyner, "Sage: system for algebra and geometry experimentation," *ACM SIGSAM Bulletin*, vol. 39, no. 2, pp. 61–64, 2005. [Online]. Available: http://modular.math.washington.edu/sage/misc/sage_sigsam_updated.pdf

[14] S. Halevi and V. Shoup, *Algorithms in HElib*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 554–571. [Online]. Available: https://doi.org/10.1007/978-3-662-44371-2_31

[15] C. Jayet-Griffon, M.-A. Cornelie, P. Maistri, P. Elbaz-Vincent, and R. Leveugle, "Polynomial multipliers for Fully Homomorphic Encryption on FPGA," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig'15)*, ser. Proceedings IEEE Catalog Number CFP15389-CDR. Mayan Riviera, Mexico: IEEE, Dec. 2015, collection Persyval Lab. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01240658