2016 IEEE International Conference on Cloud Computing in Emerging Markets

Securing Files in the Cloud

Bharati Mishra[1], Debsish Jena[2]
Department of Computer Science and Engineering
IIIT Bhubaneswar, Bhubaneswar
Odisha, India, 751003
Email: bmbharati@gmail.com[1], dr.djena@gmail.com[2]

***Abstract*—Dropbox, Google Drive, OneDrive, Box etc are popular File Storage providers for the Cloud. Recently reported vulnerability like Heartbleed for servers using OpenSSL has created panic among Cloud users. Hence, it is essential to encrypt the files by the users before uploading them to the cloud. Many File encrypting tools like Veracrypt, AxCrypt, Boxcrypt, are in place to encrypt files. But they have implemented AES and RSA encryption algorithm which are not secure against quantum computing. In this paper, we have implemented a lattice based encryption algorithm based on hardness of Ring LWE problem, which is secure against quantum computing to secure files stored in Cloud Storage.**

Keywords*— *Cloud, Security, Quantum Computing, Ring-LWE,File Encryption

I. Introduction

Cloud storage is a data storage model in which files are stored in logical partitions where as the physical storage spans multiple servers in multiple locations and the physical environment is owned and managed by a hosting company. These cloud storage providers are responsible for providing availability and security of user files. People and organizations buy or lease storage capacity from the providers to store user, organization, or application data.Cloud storage services may be accessed through a co-located cloud computer service, a web service application programming interface (API) or by applications that utilize the API, such as cloud desktop storage, a cloud storage gateway or Web-based content management systems.Files are stored in cloud storage systems in plain text format.Again multiple copies of the files are maintained in multiple locations for faster access and availability.If proper security measures are not taken, malicious users can gain access to the files and misuse it.Moreover, when the files move from user location to cloud storage location, they can be illegally accessed by exploiting various vulnerabilities like "Heartbleed" that exist in open SSL protocols.Therefore, it is essential to encrypt the files at the user location before uploading them to the Cloud.

In this paper, in section II, we discuss about the available tools and existing techniques used for encrypting files for cloud storage.In section III, we describe the theory and framework used in our work.In section IV , we depict our system model and implementation steps used to develop the encryption tool for encrypting files for cloud storage.In section V, we discuss and compare our results and perform security analysis.In section VI, we put forward our conclusion and future work.

II. Related Work

Tools like Boxcryptor, Veracrypt, AxCrypt, CloudFile, SpiderOk available in the market for encrypting files[4].

- **Boxcryptor:** It creates a virtual drive on your computer that allows you to encrypt your files locally before uploading them to your cloud or clouds of choice. Boxcryptor uses the AES-256 and RSA encryption algorithms. [5]
- **Veracrypt:** It is a software for establishing and maintaining an on-the-fly-encrypted volume (data storage device).It uses AES-128 bit, Serpent and twofish encryption algorithm for file encryption[6]
- **AxCrypt:** It is the leading open source file encryption software for Windows. It integrates seamlessly with Windows to compress, encrypt, decrypt, store, send and work with individual files[10]. AxCrypt uses the AES(128-bit keys) in Cipher Block Chaining mode with a 'random' IV for the data encryption. For integrity verification AxCrypt uses Hash Message Authentication Code using SHA-1 with 128-bit output and key.
- **SpiderOk:** It uses a layered approach to encryption, using a combination of 2048 bit RSA and 256 bit AES.[11]

Public-key cryptography schemes use number theoretic problems such as factoring or discrete logarithms for providing digital signatures, key-exchange, and confidentiality. For sufficiently large key sizes, these public-key crypto systems are practically unbreakable with present day computers, super computers, and special hardware clusters. In 1994 Peter Shor proposed a quantum algorithm [9] for integer factorization. A modified version of Shors algorithm can solve the elliptic curve discrete logarithm problem (ECDLP). Shors algorithms cannot be used on classical computers, and can only execute on powerful quantum computers to solve the factoring or the discrete logarithm problems in polynomial time. In the present decade, significant research is being performed to develop powerful quantum computers. Therefore, it is imperative to use efficient public-key algorithms which can be **secure against quantum computing**.

Lattice-based cryptography[13] consists of asymmetic cryptographic primitives based on lattices. Cryptographic constructions based lattice hold a great promise for post-quantum cryptography, as they enjoy very strong security proofs based on worst-case hardness, relatively efficient implementations, as well as great simplicity. In addition, lattice-based cryptography is believed to be secure against quantum computers.

The innovative ways in which lattice can be used in cryptography was discovered in a break-through paper by Ajtai [14].More recently, Regev [15] defined the learning

978-1-5090-4573-0/16 $31.00 © 2016 IEEE
DOI 10.1109/CCEM.2016.15

with errors (LWE) problem and proved that it is being as hard as worst-case lattice problems, hence making all cryptographic constructions which are based on it secure under the assumption that worst-case lattice problems are hard.A main drawback of schemes based on the LWE problems, however, is that they are not efficient enough for practical applications. Even the simplest primitives, such as one-way and collision resistant hash functions, have very large key sizes and require computation times that are at least quadratic in the main security parameter[16,17].Lyubashevsky et al[18]defined a ring-based variant of LWE ,known as ring-LWE and proved its hardness under worst-case assumptions on ideal lattices.

An encryption scheme based on the ring-LWE problem was proposed in [18].An efficient version of the encryption scheme which minimizes the number of costly NTT operations is proposed in [19].Ruan de Clercq et al[8] implemented the the encryption scheme proposed in [18] on an 32-bit ARM Cortex-M4F microcontroller as the target platform. Their contribution includes optimization techniques for fast discrete Gaussian sampling and efficient polynomial multiplication.Interested readers are advised to go through the lattice survey [28] to understand how the inefficiencies in LWE overcome through ring-LWE based schemes.

In this paper, we have used the techniques of Ruan de Clercq et al.[8], to encrypt files before uploading them to cloud storage.We have implemented the above encryption scheme for the parameter sets (n, q, σ) from [20], namely $P_1 = (256, 7681, 11.31/\sqrt{2\Pi}\ )$ and $P_2 = (512, 12289, 12.18/\sqrt{2\Pi}\ )$ that have medium-term and long-term security respectively

III. THEORY FRAMEWORK AND MODELING

A **lattice** is a set of points in n-dimensional space with a periodic structure, such as the one illustrated in Figure 1. More formally, given n-linearly independent vectors $\mathbf{b_1} \cdots \mathbf{b_n} \in \mathbb{R}^{\mathbf{n}}$,the lattice generated by them is the set of vectors

$$\mathbb{L}(\mathbf{b_1} \cdots \mathbf{b_n}) = \mathbb{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in (Z) \right\}$$

A lattice basis B is not unique: for any unimodular matrix $U \in Z_n$ (i.e., one having determinant ± 1), $B.U$ is also a basis of L(B), because $U.Z_n = Z_n$. This property of lattices is the core of all security schemes based on lattices.

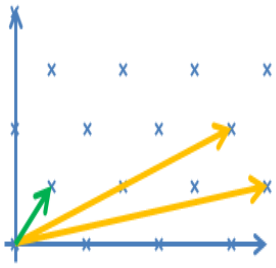

Fig. 1: A two-dimensional lattice and two possible bases

Definition. ***(Ring-LWE Problem)*** *In ring-LWE problem[8] two polynomials a and s are chosen uniformly from a polynomial ring $R_q = Z_q[x]/f$ where f is an irreducible polynomial of degree $n-1$.An error polynomial e of degree n is sampled from an error distribution $\varkappa$.The error distribution is usually a discrete Gaussian distribution $\varkappa_\sigma$ with standard deviation σ.*

Definition. ***(Ring-LWE distribution)****[28]. For an $s \in R_q$ called the secret, The ring-LWE distribution $A_{s,\varkappa}$ over $R_q \times R_q$ is sampled by choosing $a \in R_q$ uniformly at random, choosing $e \leftarrow \varkappa$, and outputting (a, b = s . a + e mod q).*

Definition. ***(Decision-Ring-LWE)****[28]. Given m independent samples $(a_i, b_i) \in Rq \times Rq$ where every sample is distributed according to either:*

1) *$A_{s,\varkappa}$ for a uniformly random $s \in R_q$ (fixed for all samples), or*
2) *the uniform distribution,*

The Decision-Ring-LWE problem is to distinguish which is the case (with non-negligible advantage).

Definition. ***(Search-Ring-LWE)****[8] Given a polynomial number of sample pairs (a, t) from $A_{s,\varkappa}$, it is very difficult to find s.This problem is known as the search ring-LWE problem.*

A. Discrete Gaussian Sampling

The ring-LWE cryptosystem requires samples from a discrete Gaussian distribution to construct the error polynomials during the key generation and encryption operations. There are various methods for sampling from a discrete Gaussian distribution. The most well known techniques are rejection sampling, inversion sampling, and the random bit model. Efficiency (time and space) of the sampling algorithms depends on the standard deviation σ of the distribution. A detailed comparative analysis of the sampling algorithms can be found in [21]. In [8] the Knuth-Yao algorithm [22]is used to sample from a discrete Gaussian distribution. The Knuth-Yao algorithm is based on the random-bit model and uses, on average, a near-optimal number of random bits. This algorithm requires storage of the probabilities of the sample points. The small standard deviation in the ring-LWE encryption scheme means that the memory requirement is small and can easily be satisfied on memory constrained mobile devices.

B. Polynomial Multiplication

For large polynomial multiplications, the Fast Fourier Transform (FFT) is considered as the fastest algorithm due to its n $(logn)$ complexity. In [8], the Number Theoretic Transform (NTT) is used for polynomial multiplication which corresponds to a FFT where the primitive roots of unity are from a finite ring (thus integers) instead of complex numbers. For efficient implementation, polynomial arithmetic is performed in $R_q = Z_q[x]/f$ where f is $f = x^n + 1$,and $n = 2^k$ and q is a prime such that $q = 1 mod 2n$. Such a choice of the parameters allows us to use an n-point NTT instead of a 2n-point NTT during a polynomial multiplication in R_q. However there is an additional scaling overhead associated with the negative wrapped convolution. This technique is known as the negative wrapped convolution. In [8],optimization from [19] is used during the NTT computation. In this paper we use the steps from [8] to implement a fast encryption scheme to encrypt files for cloud storage. The encryption operation samples three error polynomials, computes three forward NTTs, two

coefficient-wise polynomial multiplications, and three polynomial additions. The decryption computes one coefficient-wise polynomial multiplication, one polynomial addition, and one inverse NTT. For more details the reader is referred to [8].Below we describe the steps from [8] used in our implementation.

1) *KeyGeneration($\tilde{a}$)*:Two polynomials r_1 and r_2 are sampled from $\varkappa_\sigma$ using a discrete Gaussian sampler. The following computations are performed.

$$\tilde{r_1} = NTT(r_1); \tilde{r_2} = NTT(r_2); \tilde{p} = \tilde{r_1} - \tilde{a}\tilde{r_2} \quad (1)$$

$\tilde{r_2}$ is the private key and $(\tilde{a}, \tilde{p})$ is the public key.

2) *Encryption($\tilde{a}, \tilde{p}, m$)*:The input message m is encoded to a polynomial $\bar{m} \in R_q$.Error polynomials $e_1, e_2, e_3 \in R_q$ are generated from $\varkappa_\sigma$ using a discrete Gaussian sampler. The following computations are performed to compute the ciphertext $(\tilde{c_1}, \tilde{c_2})$.

$$\tilde{e_1} = NTT(e1); \tilde{e_2} = NTT(e2) \quad (2)$$

$$\tilde{c_1} = \tilde{a} * \tilde{e_1} + \tilde{e_2}; \quad (3)$$

$$\tilde{c_2} = \tilde{p} * \tilde{e_1} + NTT(e_3 + \bar{m}) \quad (4)$$

3) *Decryption($\tilde{c_1}, \tilde{c_2}, \tilde{r_2}$)*:The inverse NTT is performed to compute $\acute{m}$.

$$\acute{m} = INTT(\tilde{c_1} * \tilde{r_2} + \tilde{c_2}) \quad (5)$$

The original message m is recovered from $\acute{m}$ by using a decoder.

IV. Implementation

In this paper, we have used Dropbox[23] as the cloud service provider. Dropbox provides cloud based file storage service to more than 100 million users.In spite of its widespread popularity, Dropbox as a platform hasnt been analyzed extensively enough from a security standpoint[24]. The Dropbox clients run on around ten platforms and many of these Dropbox clients are written mostly in Python [25].Dropbox being a proprietary software, its client source code is not available. But one can write a Dropbox client in any language and call the Dropbox APIs for authentication,file upload and download, browsing files and delete files.

Accessing Dropboxs website requires one to have the necessary credentials (email address and password). The same credentials are also required in order to link (register) a device with a Dropbox account. In this registration process, the end-user device is associated with a unique host ID which is used for all future authentication operations. In other words, Dropbox client doesnt store or use user credentials once it has been linked to the user account. host ID is not affected by password changes and it is stored locally on the end-user device.

A. Existing Dropbox client

Existing Dropbox clients upload plain text file from user to the Dropbox server after authentication.User or anyone having permission can download the plain text file and use it.
Problem with Existing System: Malicious user can get the copy of plain text file during upload or from Dropbox server and misuse it.

TABLE I: Table showing the software used to Develop the Dropbox Client

Sl. No.	Software	version
1	Java SDK	1.8
2	Dropbox Core SDK	2.0-beta-5[26]
3	Jackson Core	2.6.1[27]

TABLE II: Table showing the System Configuration used to Develop the Dropbox Client

Sl. No.	Hardware	Configuration
1	Processor	Intel(R) Core TM(i3) CPU @2.40 GHz
2	RAM	4GB
3	System Type	32-bit Windows OS running Windows 7 Professional

B. Modified Dropbox client

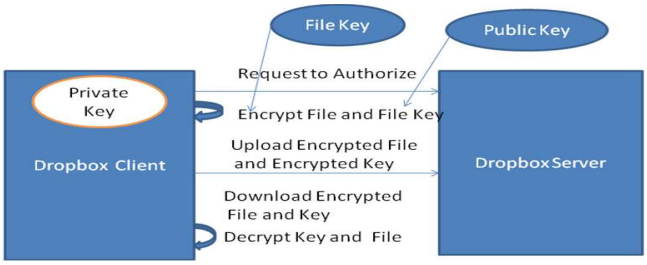

Fig. 2: Modified Dropbox client

Initialization

(1) *System Setup:* User generates the R-LWE public key $\tilde{a}$ and Private Key $\tilde{r_2}$ as per equation(1) section III .
(2) *Authentication:* User provides the app key to connect to the Dropbox and authenticates itself to access its Dropbox account in Dropbox server.

Encryption

(1) *File Key Generation:* User selects the file to be uploaded to Dropbox.A file key is generated using AES-256 algorithm .
(2) *Symmetric Key Encryption:* The file selected in the above step is encrypted with

the generated file key.The encrypted file is stored with extension .aes.

(3) *Public Key Encryption:*The file key is encrypted with R-LWE public key $(\tilde{a}, \tilde{p})$ as per equation (3) and (4) given in section III.

(4) *Upload Encrypted Files:*The encrypted file and encrypted file key is uploaded to Dropbox.

Decryption

(1) *Download Encrypted Files:*The user selects the file to be downloaded.The file along with the encrypted file key is downloaded.

(2) *Decrypt File Key :*The encrypted file key is decrypted using the R-LWE private key $\tilde{r_2}$ as per equation (5) given in section III and then decoded to find the plain text filekey.

(3) *Decrypt File:*The decrypted file key is used to decrypt the file using AES-256 algorithm.

V. Results and Security Analysis

A. Results

Fig. 3: Snapshot of our File Encryption Tool

R-LWE encryption scheme can be used to encrypt any type of file i.e. text,audio,video,image,pdf,excel,binary etc.The following table depicts the the plain text file size verses its encrypted version for various file types and compared it with AES-256 encryption algorithm.

From the table III, we observe that, with Ring-LWE encryption the encrypted file increases by a factor of 32.But, for AES-256 encryption the encrypted file size is same as plain text file.AES-256 secure from attacks by quantum computers and it is faster[2].Hence, we have used AES-256 for encrypting files. But the user cannot keep all the file keys with him .It is also not secure to keep them in plain text format.Hence we have used R-LWE encryption scheme for encrypting file keys.

TABLE III: Table Comparing Encrypted file sizes for R-LWE vs AES-256

File Type	FileSize (Bytes)	Encrypted FileSize(R-LWE)	Encrypted FileSize (AES-256)
Text	1655	53248	1655
Image	1495557	47895712	1495557
Excel	58028	1857536	58028
PDF	136999	4384768	136999
Video	26246026	839874560	26246026

The graphs in figure 4 and figure 5 show the encryption time and decryption time required for various file types.In Table IV , we have provided the comparison for encrypted file sizes of our encryption tool with that of Boxcrypter. We have taken ten files of different size for a particular type of file and found the aveage size before encryption and after encryption.We find that our solution provides almost comparable results with that of Boxcrypter.

TABLE IV: Comparision Table for Encrypted File Size Ring Cryptor vs Boxcryptor

		Encrypted File Size	
File Type	Original File Size	Ring Encrypter	Boxcryptor
text	1,655	5760	5,760
image	1,495,557	1499664	1,511,952
excel	58,028	62128	62,128
pdf	136,999	141104	141,104
video	26,246,026	26250128	26,291,088

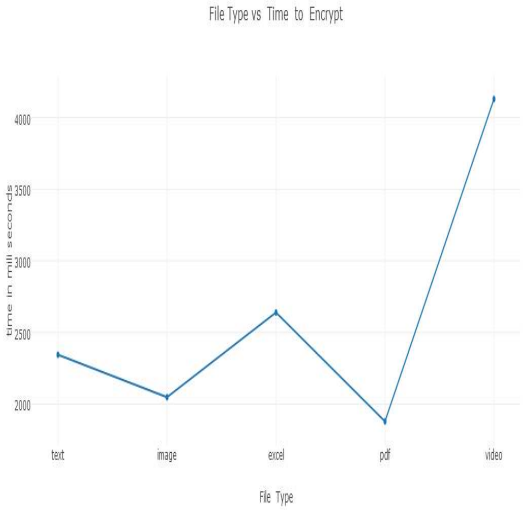

Fig. 4: Encryption Time for various File Types

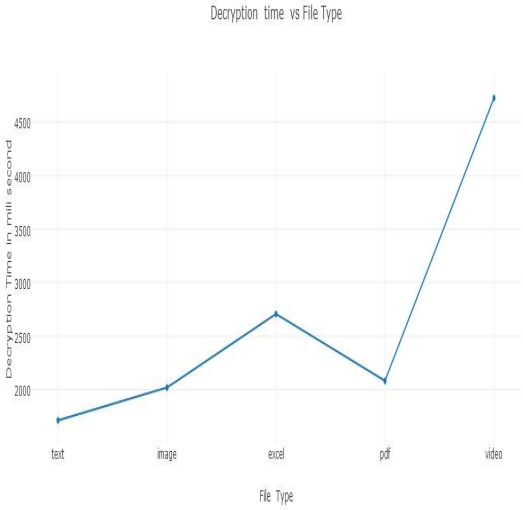

Fig. 5: Decryption Time for various File Types

B. Security Analysis

Suppose an adversary $\mathbb{A}$ gets hold of an encrypted file and encrypted filekey and obtains the public key used to encrypt the filekey, but it won't be able to decrypt the encrypted file key.The reason for this is that, we have used Ring-LWE based public key encryption to encrypt the filekey. Ring-LWE based public key encryption scheme is semantically secure against passive eavesdroppers and its hardness has been proved to be based on worst case lattice problems.Ring-LWE based cryptographic schemes are resistant to quantum computing i.e. no polynomial time algorithm exists till date to break the security of these schemes even when they are executed on super fast quantum computers.Since the filekey cannot not be derived by the adversary, it would not be possible to decrypt the file which has been encrypted with AES-256 bit symmetric key algorithm.Hence files remain confidential even in public clouds.

VI. Conclusion

A lot of effort is being utilized towards developing quantum computers. Once, quantum computers becomes real, then all security schemes based on IFP , DLP or ECDLP will be insecure because they can be broken by quantum computers in matter of seconds.Hence new security schemes are developed based on lattice cryptography which have been proved to be computationally hard and cannot be broken even by quantum computers. Hence, we have implemented Ring-LWE based encryption scheme for file encryption for cloud storage.We have combined AES-256 with R-LWE encryption for developing a dropbox client which encrypts the files and the file key before uploading to dropbox.This scheme can be integrated with any cloud storage systems.In future we shall be designing schemes to share encrypted files with authenticated parties through cloud storage.Scalability of our implemented tool can be enhanced by doing parallel computation for performing NTT operation for polynomials.Compression and decompression techniques can be added to files along with encryption and decryption, to save space in cloud storage.

References

[1] $https://en.wikipedia.org/wiki/Shor's_algorithm$

[2] Daniel J. Bernstein . "Introduction to post-quantum cryptography" (PDF). (Introductory chapter to book "Post-quantum cryptography"),Springer,2009.

[3] Daniel J. Bernstein ,"Grover vs. McEliece" (PDF).

[4] http://lifehacker.com/five-best-file-encryption-tools-5677725/1685273934

[5] https://www.boxcryptor.com/en/boxcryptor

[6] https://veracrypt.codeplex.com/wikipage?title=Introduction

[7] O. Regev, "On Lattices, Learning with Errors, Random Linear Codes,and Cryptography", in 37th Annual ACM Symposium on Theory of Computing, 2005, pp. 8493.

[8] Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede.Efficient software implementation of ring-LWE encryption. In Proceedings of the 2015 Design, Automation and Test in Europe Conference and Exhibition (DATE '15). EDA Consortium, San Jose, CA, USA, 339-344.

[9] P. Shor, Algorithms for Quantum Computation: Discrete Logarithms and Factoring, in Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on, Nov 1994, pp. 124134.

[10] http://www.axantum.com/axcrypt/

[11] https://spideroak.com/

[12] D. E. Knuth and A. C. Yao, "The Complexity of Nonuniform Random Number Generation," Algorithms and complexity: new directions and recent results, pp. 357428, 1976.

[13] Micciancio, Daniele, and Oded Regev. "Lattice-based cryptography." Post-quantum cryptography. Springer Berlin Heidelberg, 2009. 147-191.M.

[14] Ajtai. Generating hard instances of lattice problems. In Complexity of computations and proofs,volume 13 of Quad. Mat., pages 132. Dept. Math., Seconda Univ. Napoli, Caserta, 2004. Preliminaryversion in STOC 1996

[15] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. J. ACM,56(6):140, 2009. Preliminary version in STOC 2005

[16] D. Micciancio and O. Regev. Lattice-based cryptography. In Post Quantum Cryptography, pages 147191. Springer, February 2009.

[17] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In CT-RSA,pages 319339. 2011.

[18] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors Over Rings," Cryptology ePrint Archive, Report 2012/230, 2012.

[19] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, Compact Ring-LWE Cryptoprocessor, in Cryptographic Hardware and Embedded Systems CHES 2014, 2014, vol. 8731, pp. 371391.

[20] Gttert, Norman, et al. "On the design of hardware building blocks for modern lattice-based encryption schemes." Cryptographic Hardware and Embedded SystemsCHES 2012. Springer Berlin Heidelberg, 2012. 512-529.

[21] N. C. Dwarakanath and S. D. Galbraith, Sampling from Discrete Gaussians for Latticebased Cryptography on a Constrained Device, Applicable Algebra in Engineering, Comm.and Computing, pp. 159180, 2014.

[22] D. E. Knuth and A. C. Yao, The Complexity of Nonuniform Random Number Generation,Algorithms and complexity: new directions and recent results, pp. 357428, 1976.

[23] https://www.dropbox.com/home

[24] Kholia, Dhiru, and Przemyslaw Wegrzyn. "Looking Inside the (Drop) Box." WOOT. 2013.

[25] HUNTER, R., "How dropbox did it and how python helped". PyCon 2011 (2011).

[26] https://oss.sonatype.org/content/repositories/releases/com/dropbox/core/dropbox-core-sdk/2.0-beta-5/dropbox-core-sdk-2.0-beta-5.jar

[27] https://oss.sonatype.org/content/repositories/releases/com/fasterxml/jackson/core/jackson-core/2.6.1/jackson-core-2.6.1.jar

[28] Chris. Peikert Decade of Lattice Cryptography. World Scientific, 2016.