Design and Implementation of an Efficient Tool to Verify Integrity of Files Uploaded to Cloud Storage

Priyansi Parida
NIT Rourkela,Odisha, India,769008
priyansiparida@gmail.com

Snehalata Konhar
NIT Rourkela,Odisha India,769008
snehalatakonhar@gmail.com

Bharati Mishra
IIIT Bhubaneswar
Odisha, India, 751003
bmbharati@gmail.com

Debsish Jena
IIIT Bhubaneswar
Odisha, India, 751003
dr.djena@gmail.com

Abstract—**Cloud storage system (CSS) provides storage service through which users can upload their files for getting several benefits like ease of access, less cost and robustness. The uploaded files must be encrypted to achieve confidentiality. These files are replicated across multiple nodes distributed over various geographical boundaries by the CSS provider to ensure robust-ness. But these files may get corrupted unintentionally during transmission or intentionally by malicious users. To ensure the integrity of the uploaded file, an efficient integrity verifier tool is essential. In this paper, the design and implementation of an integrity verifier tool for files uploaded to cloud has been presented. Merkle Hash Tree (MHT) and digital signatures have been used to verify the integrity. The tool has been integrated with Dropbox Cloud Storage Provider and its performance results have been presented.**

Keywords—– Cloud Computing; Data Integrity; Advanced Encryption Standard; **Merkle** Hash Tree; Digital Signatures

I. INTRODUCTION

Cloud storage providers like Dropbox [1], Google Drive [2], Amazon S3 [3], [4] are very popular and are used by users and corporate alike to store their personal and business data. They provide ease of access, collaboration, sharing, robustness and cost benefit to users. But users lose control on who can access the files and can be maliciously accessed by unauthorized users. In literature, many researchers [5]–[9] have suggested to encrypt the files before uploading them to cloud. This ensures the confidentiality of the uploaded files. The encrypted files themselves can be corrupted during transmission or intentionally by malicious entities. Moreover, users may want to share their files with other users. The recipient of the files also needs to ensure that the files are not altered. To ensure integrity of files, researchers have proposed to use hashing and digital signature. A Merkle Hash tree(MHT) is a hash based data structure aimed to securely prove that the data files are not modified or damaged [10]. A digital signature is a mathematical scheme aimed to verify the authenticity and integrity of a digital document [11]. In this paper, the design and implementation of a tool that can be used to verify the integrity of the files uploaded to CSS has been presented. The tool has been integrated with the existing Dropbox client and tested for its performance and the results are presented.
In this paper, in section II, existing schemes devised for data storage and security in cloud computing has been discussed. In section III, the theoretical view of the scheme has been presented. In section IV, the system overview has been por-trayed. In section V, the implementation of the scheme, which constructs a Merkle Hash tree and then uses it for verification process, has been discussed. In section VI, the performance of the implemented scheme has been analyzed. In Section VII, the conclusions follows.

II. RELATED WORK

Ateniese et al. [9] first proposed a provable data possession scheme in 2007. In this scheme, homomorphic verifiable tags are used to verify the data integrity. But it involves a heavy workload on the servers and does not facilitate verification of data possession by clients. Shah et al. [12] proposed an auditing scheme to verify data possession by third party auditors(TPA). Eraway et al [13] improved the PDP scheme by Ateniese et al. [9] to include dynamic update of stored data blocks using rank-based authenticated skip lists. Wang et al [6], [14] scheme presented a novel public auditing scheme like Data Integrity Verification and Data Modification and Data Insertion based on MHT using SHA1.

Sood et al. [8] proposed a combined approach to support security of data in cloud by using techniques like SSL (Secure Socket Layer) 128-bit encryption, MAC(Message Authenti-cation Code) which was used for integrity check of data, searchable encryption and division of data into three sections in cloud for storage. Nayana et al [7] scheme achieved data storage security in cloud computing without involving a TPA(third party auditor) but the ELGamal encryption used in the scheme takes more time when compared to RSA leading to delayed pre-processing stage. While Khaba et al [15] devised a flexible Batch Audit scheme with dynamic data support to reduce the performance overheads using a TPA and MHT . This scheme used only symmetric cryptography thus leading to key exchange issues and problems due to use of TPA.

Wang et al [16] proposed a scheme for Identity-based remote data possession checking in public clouds. They have proved the security of the scheme based on standard compu-tational Diffie-Hellman problem. Erway et al. [17] proposed a dynamic provable data possession scheme in 2015 which uses a new version of authenticated dictionaries based on

978-1-5386-1860-8/17/$31.00 ©2017 IEEE
DOI 10.1109/CSNT.2017.13

rank information. Saxena et al. [18] proposed an integrity verification tool for files stored on cloud based on a variant of the Paillier homomorphic cryptography system [19] with homomorphic tag and combinatorial batch codes.

Yi et al. [20] proposed an efficient method for integrity verification of replicated data in cloud using Fully Homomorphic Encryption [21] to generate data block replicas and allow third party public verification.

Lin et al. [22] proposed a data integrity verification scheme using a hash tree data structure and a Boneh Lynn Shacham short signature scheme [23].

The proposed scheme enhances the lattice based hybrid encryption scheme for files uploaded to cloud proposed by Mishra et al. [24] which is based on Ring-LWE based public key encryption [25] and Adavanced Encryption Stan-dard(AES). The above scheme has been extended to include integrity verification of the files uploaded to cloud. Lattice based cryptography [26] holds promise for post quantum computing era [27]. Security schemes based on lattice based cryptography cannot be broken by existing computers and not even by quantum algorithms as proposed by Peter Shor
[28]. The proposed scheme uses Merkle Hash Tree and short signatures to add integrity verification capability to the existing scheme.

III. THEORY

A. Merkle Hash Tree

Hash trees are based on the concept of binary trees which helps in secure storage and helps for integrity verification of large data.

In MHT [10], the data file is broken into a number of blocks. Hashing is applied to each original file block. The hash value of each block is stored in a leaf node. Hash values of two leaf nodes are concatenated to obtain the data for the parent node. Then the parent node is generated by rehashing two children hash nodes. This procedure is repeated until a tree with a single root is obtained. The MHT is generated by the owner of the file and is stored at the server side.

A Merkle Hash Tree is illustrated in Figure 1. The tree has 8 leaf nodes for the 8 original blocks.

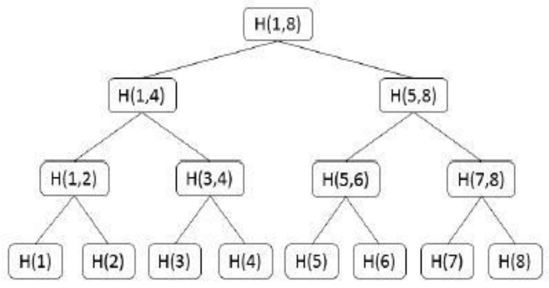


Fig. 1: A Merkle Hash Tree

Here H(1; 2) = H(H(1) k H(2))where k denotes con-catenation. Thus, through iterative combination and rehashing we obtain the single root of the tree i.e. H(1,8) which is used for verification purposes. [29]

B. SHA Functions

The proposed scheme uses SHA-256 for MHT Generation. SHA-256 produces a 256-bit (32-byte) hash value [30]. It is usually represented as a hexadecimal number of 64 digits. An implementation of SHA-256 is given by Appel and Andrew
[31] which can be referred for further understanding of SHA-256.

C. Digital Signature

Digital Signature technique [32] is based on asymmetric cryptography also known as public key cryptography. In digital signature private key is used for signing the data, and the public key is used to verify the data file. The person having the private key can sign the data. Any person can verify the message with the help of corresponding public key. Digital signatures has been implemented in a secure auction service as discussed in paper by Franklin et al. [33], and Even et al. [34].In our proposed scheme digital signature is used to sign the obtained hash data for the file uploaded to cloud, to ensure that the hash value itself is not modified.

IV. System Model

In the proposed scheme as illustrated in figure 2, the owner of the file performs encryption of the plaintext file using AES-256 algorithm using a symmetric key K. Then the owner encrypts the file Key K using RLWE encryption and saves it in a file. After that, the MHT for the encrypted file is created. The MHT is written to a file which is signed using digital signature. The owner uploads the encrypted file, encrypted file key K and the signed MHT file. When a user who wants to use the encrypted file, she downloads the encrypted file, encrypted key file and the signed MHT file. She then verifies the integrity of the encrypted file by calculating the new MHT and compare it with the uploaded MHT. If they match, the integrity of the encrypted file is ensured. Then she can go forward to decrypt the encrypted file as discussed in Mishra et al. scheme [24].

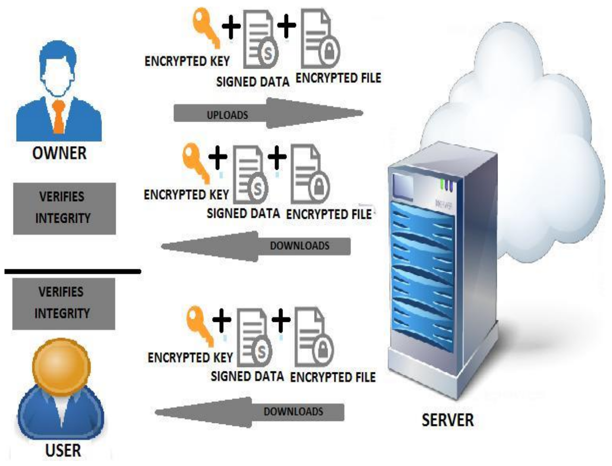


Fig. 2: System Overview

V. Implementation

A. Existing Dropbox client

Existing Dropbox clients upload their data in the form of plain text files to the Dropbox server after authentica-tion.Bharati Mishra and Debasish Jena [24] put forward an encryption scheme for files uploaded to cloud using lattice based cryptography. Authenticated users can download the ci-phertext files and retrieve the original content after decryption.

B. ShortComing in Mishra et al. schemea [24]

Malicious users can acquire a copy of ciphertext file either during upload of the files or from Dropbox server altering the contents of the file leading to loss of data as decryption algorithm fails to retrieve the original contents.

C. Modified Dropbox client

In this paper, Bharati Mishra and Debasish Jena [24] scheme has been enhanced to perform an integrity check of files downloaded from Dropbox to detect data loss or modification.

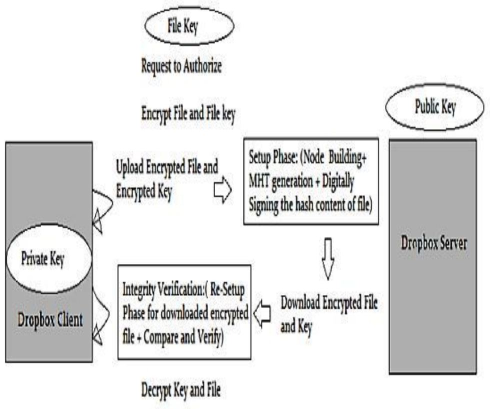

File Key
Request to Authorize
Encrypt File and File key
Public Key
Upload Encrypted File and Encrypted Key
Setup Phase: (Node Building+ MHT generation + Digitally Signing the hash content of file)
Private Key
Dropbox Client
Dropbox Server
Integrity Verification:(Re-Setup Phase for downloaded encrypted file + Compare and Verify)
Download Encrypted File and Key
Decrypt Key and File

Fig. 3: Modified Dropbox client

Initialization

The public key K_{pub} and private key K_{pri} are generated for digitally signing, using RSA based digital signature algo-rithm [35]. The file to be uploaded is en-crypted using AES encryption algorithm.

Setup Phase

(1) Node Generation: Encrypted file is di-vided into a number of blocks, each of which is hashed using SHA-256 algo-rithm and then converted to hex format.

(2) Merkle Hash Tree Generation: The list containing hashed values of file is con-verted into a merkle hash tree of height 2 containing 3 nodes. The root node is formed by the continuous concatenation and rehashing of left and right child nodes. Each of the hashed node formed is converted and stored in hex format in a text file. This file is called the MHT file.

(3) Digital Signature: The MHT file is signed with private key K_{pri} and then stored in a new signed MHT file, a copy of which is uploaded along with encrypted file and file key to Dropbox.

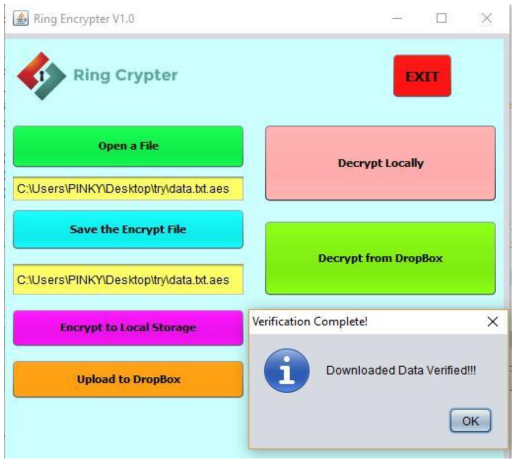
Integrity Verification:

(1) Rerun through Setup Phase: Files to be downloaded from cloud server are se-lected by user. The downloaded AES encrypted file is run through Setup phase again to generate a new hashed text file.

(2) Compare and Verify: The signed MHT file is verified for the signature. If the signature verification succeeds, then the MHT file itself has not been modified. Then user proceeds further to compare the uploaded MHT file with the calcu-lated MHT file. If there is a match, the user ensures that the encrypted file has not been modified and proceeds further for decryption. Otherwise, the user stops and do not proceed further for decryption as file might have been modified mali-ciously or during transmission error has occurred.

VI. Performance Analysis

The tool is implemented using java programming language version 1.8, Dropbox Core SDK version 2.0-beta, Jackson Core version 2.6.1 on a 32-bit Windows OS running Windows 7 Professional having Intel(R) Core TM(i3) CPU @2.40 GHz and 4 GB RAM. The tool can be used to check the data integrity of many types of files like audio, text, image, video, document, pdf, excel and many more. Using varying types and sizes of files, the performance of the tool against time in milliseconds has been studied.



C:\Users\PINKY\Desktop\try\data.txt.aes

C:\Users\PINKY\Desktop\try\data.txt.aes

Verification Complete!

Downloaded Data Verified!!!

Fig. 4: Snapshot of the Data Integrity Verifier tool

For analysis of time taken by Setup Phase and Integrity Verification Phase, a total of 10 files of varying sizes for a particular type i.e. text files, as shown in Table I have been considered. The graphs in figure 5 and figure 6 show the Setup time and Integrity Verification time in milliseconds required for various file sizes.

TABLE I: Table Showing Different File Sizes taken for Time Analysis

Serial No	File Size((KB))
1	10
2	20
3	50
4	100
5	150
6	200
7	250
8	500
9	750
10	1000

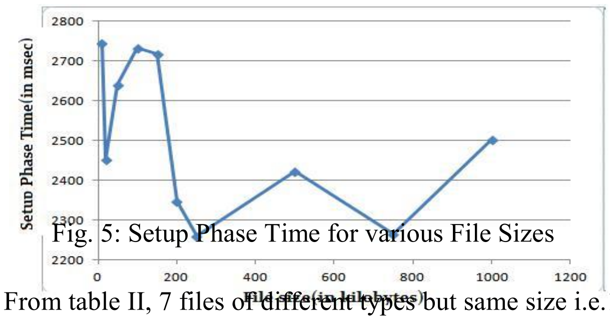


Fig. 5: Setup Phase Time for various File Sizes

From table II, 7 files of different types but same size i.e.100 kilobytes each, have been considered and comparison has been done with their Setup and Verification times in milliseconds.The graphs depicted in figure 7 and figure 8 show the Setup time and Integrity Verification time required for varying file types.

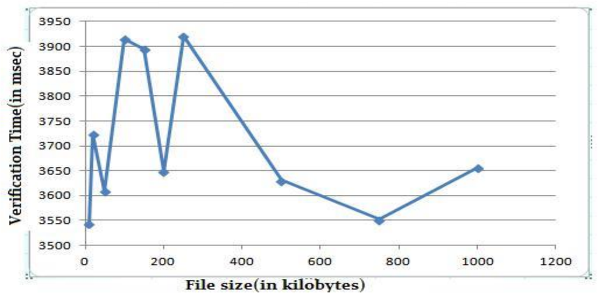


Fig. 6: Integrity Verification Time for various File Sizes

TABLE II: Table Showing Different File Types taken for Time Analysis

Serial No	File Type	File Size((KB))
1	Text	100
2	Audio	100
3	Video	100
4	Image	100
5	Document	100
6	PDF	100
7	Excel	100

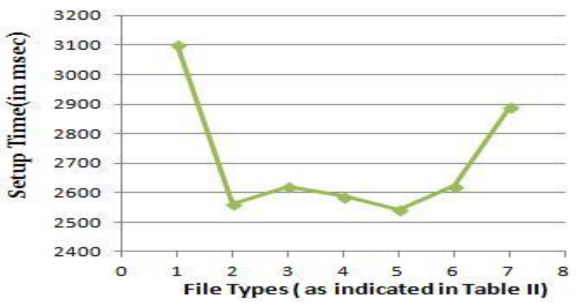


Fig. 7: Setup Phase Time for distinct File Types

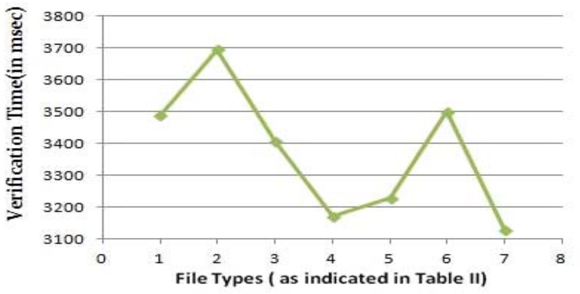


Fig. 8: Integrity Verification Time for distinct File Types

VII. Conclusion

In this paper, the design as well as implementation of a tool to verify the integrity of files uploaded to cloud has been discussed. MHT using SHA-256 has been implemented which is one of the recent unbroken hash function in the hash family that might take several decades to break it. Digital signatures has been implemented to verify the integrity of data. The proposed scheme depicts a data integrity verification protocol for files uploaded to cloud storage by providing data integrity and security to customer's crucial data. Authenticated users and the owner of the data are the only ones allowed to check the integrity of data. In future, our implemented tool can be enhanced by checking for data loss/modification at the block level so that the user can know the exact location of data loss/modification. Also dynamic updating of files with integrity check can be added.

References

[1] I. Drago, M. Mellia, M. M Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: understanding personal cloud storage services," in Proceedings of the 2012 ACM conference on Internet measurement conference. ACM, 2012, pp. 481–494.

[2] I. Ltkebohle, "All About Google Drive," https://www.gcflearnfree.org/googledriveanddocs/all-about-google-drive/1/, [Online; accessed 19-July-2017].

[3] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon s3 for science grids: a viable solution?" in Proceedings of the 2008 international workshop on Data-aware distributed computing. ACM, 2008, pp. 55–64.

[4] S. Garfinkel, "An evaluation of amazon's grid computing services: Ec2, s3, and sqs," 2007.

[5] P. M. Pardeshi and D. R. Borade, "Improving data integrity for data storage security in cloud computing," International Journal of Computer Science and Network Security (IJCSNS), vol. 15, no. 6, p. 75, 2015.

[6] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," Computer Security–ESORICS 2009, pp. 355–370, 2009.

[7] S. Nayana, G. VenifaMini, and J. J. A. Celin, "An effectual integrity verification with reliable mass data storage scheme in cloud computing," Networking and Communication Engineering, vol. 5, no. 4, pp. 147–152, 2013.

[8] S. K. Sood, "A combined approach to ensure data security in cloud computing," Journal of Network and Computer Applications, vol. 35, no. 6, pp. 1831–1838, 2012.

[9] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Pro-ceedings of the 14th ACM conference on Computer and communications security. Acm, 2007, pp. 598–609.

[10] G. Becker, "Merkle signature schemes, merkle trees and their cryptanal-ysis," Ruhr-University Bochum, Tech. Rep., 2008.

[11] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 26, no. 1, pp. 96–99, 1983.

[12] M. A. Shah, M. Baker, J. C. Mogul, R. Swaminathan et al., "Auditing to keep online storage services honest." in HotOS, 2007.

[13] C. Erway, A. Kupc¨,u,¨ C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009, pp. 213–222.

[14] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," IEEE Network, vol. 24, no. 4, pp. 19–24, July 2010.

[15] M. Khaba and M. Santhanalakshmi, "Remote data integrity checking in cloud computing," International Journal on Recent and Innovation Trends in Computing and Communication ISSN, pp. 2321–8169, 2013.

[16] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," IET Information Security, vol. 8, no. 2, pp. 114–121, March 2014.

[17] C. C. Erway, A. Kupc¨,u,¨ C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," ACM Transactions on Information and System Security (TISSEC), vol. 17, no. 4, p. 15, 2015.

[18] R. Saxena and S. Dey, "Cloud audit: A data integrity verification approach for cloud computing," Procedia Computer Science, vol. 89, pp. 142–151, 2016.

[19] P. Paillier et al., "Public-key cryptosystems based on composite degree residuosity classes," in Eurocrypt, vol. 99. Springer, 1999, pp. 223–238.

[20] M. Yi, J. Wei, and L. Song, "Efficient integrity verification of replicated data in cloud computing system," Computers & Security, vol. 65, pp. 202–212, 2017.

[21] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, "Single-database private information retrieval from fully homomorphic encryption," IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 5, pp. 1125–1134, May 2013.

[22] C. Lin, Z. Shen, Q. Chen, and F. T. Sheldon, "A data integrity verification scheme in mobile cloud computing," Journal of Network and Computer Applications, vol. 77, pp. 146–151, 2017.

[23] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," Advances in CryptologyASIACRYPT 2001, pp. 514–532, 2001.

[24] B. Mishra and D. Jena, "Securing files in the cloud," in Cloud Computing in Emerging Markets (CCEM), 2016 IEEE International Conference on. IEEE, 2016, pp. 40–45.

[25] V. Lyubashevsky, C. Peikert, and O. Regev, "A toolkit for ring-lwe cryptography," in Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2013, pp. 35–54.

[26] D. Micciancio and O. Regev, "Lattice-based cryptography," in Post-quantum cryptography. Springer, 2009, pp. 147–191.

[27] D. J. Bernstein, J. Buchmann, and E. Dahmen, Post-quantum cryptog-raphy. Springer Science & Business Media, 2009.

[28] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM review, vol. 41, no. 2, pp. 303–332, 1999.

[29] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proceedings of the 4th international conference on Security and privacy in communication netowrks. ACM, 2008, p. 9.

[30] I. Ltkebohle, "SHA256," http://http://www.tips2secure.com/2016/02/sha-256-detailed-study.html, [Online; accessed 19-July-2017].

[31] A. W. Appel, "Verification of a cryptographic primitive: Sha-256," ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 37, no. 2, p. 7, 2015.

[32] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal on Computing, vol. 17, no. 2, pp. 281–308, 1988.

[33] M. K. Franklin and M. K. Reiter, "The design and implementation of a secure auction service," IEEE Transactions on Software Engineering, vol. 22, no. 5, pp. 302–312, 1996.

[34] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital signa-tures," in Conference on the Theory and Application of Cryptology. Springer, 1989, pp. 263–275.

[35] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE transactions on information theory, vol. 31, no. 4, pp. 469–472, 1985.