Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

PROTECTING CLOUD DATA WITH DYNAMIC SECURITY VIA NETWORK CODING

Dr. Murali Krishna Vanam, *Associate Professor, CSE, Vaagdevi College of Engineering (Autonomous), India*
Gopalapurapu Pravalika, *UG student, CSE, Vaagdevi College of Engineering (Autonomous), India*
Gundaveni Nikhil, *UG student, CSE, Vaagdevi College of Engineering (Autonomous), India*
Aaku Jyothi, *UG student, CSE, Vaagdevi College of Engineering (Autonomous), India*
Banothu Sagar, *UG student, CSE, Vaagdevi College of Engineering (Autonomous), India*

ABSTRACT
In the age of cloud computing, cloud users with limited storage can outsource their data to remote servers. These servers, in lieu of monetary benefits, offer retrievability of their clients' data at any point of time. Secure cloud storage protocols enable a client to check integrity of outsourced data. In this work, we explore the possibility of constructing a secure cloud storage for dynamic data by leveraging the algorithms involved in secure network coding. We show that some of the secure network coding schemes can be used to construct efficient secure cloud storage protocols for dynamic data, and we construct such a protocol (DSCS I) based on a secure network coding protocol. To the best of our knowledge, DSCS I is the first secure cloud storage protocol for dynamic data constructed using secure network coding techniques which is secure in the standard model. Although generic dynamic data support arbitrary insertions, deletions and modifications, append-only data find numerous applications in the real world. We construct another secure cloud storage protocol (DSCS II) specific to append-only data — that overcomes some limitations of DSCS I. Finally, we provide prototype implementations for DSCS I and DSCS II in order to evaluate their performance.

Index Terms—
Secure cloud storage, network coding, dynamic data, append-only data, public verifiability.

1.INTRODUCTION
WITH the advent of cloud computing, cloud servers offer to their clients (cloud users) various services that include delegation of huge amount of computation and outsourcing large amount of data. For example, a client having a smart phone with a low-performance processor or limited storage cannot accomplish heavy computation or store large volume of data. Under such circumstances, she can delegate her computation/storage to the cloud server. In case of storage outsourcing, the cloud server stores massive data on behalf of its clients (data owners). However, a malicious cloud server can delete some of the client's data (that are accessed infrequently) to save some space. Secure cloud storage protocols (two-party protocols between the client and the server) provide a mechanism to detect if the server stores the client's data untampered. Based on the nature of the outsourced data, these protocols are classified as: secure cloud storage protocols for static data (SSCS) [2], [3], [4] and for dynamic data (DSCS) [5], [6], [7], [8]. For static data, the client cannot change her data after the initial outsourcing (e.g., backup/archival data). Dynamic data are more generic in that the client can modify her data as often as needed. In secure cloud storage protocols, the client can audit the outsourced data without accessing the whole data file, and still be able to detect unwanted changes in data done by a malicious server. During an audit, the client sends a random challenge to the server which produces proofs of storage (computed on the stored data) corresponding to that challenge. Secure cloud storage protocols are publicly verifiable if an audit can be performed by any third party auditor (TPA) using public parameters; or privately verifiable if an auditor needs some secret information of the client. The entities involved in a secure cloud storage protocol and the interaction among them are shown in Figure 1. In a network coding protocol [9], [10], each intermediate node (except sender/receiver nodes) on a

network path combines incoming packets to output another packet. These protocols enjoy higher throughput, efficiency and scalability than the store-and-forward routing, but they are prone to pollution attacks by malicious intermediate nodes injecting invalid packets. These packets produce more such packets downstream, and the receiver might not finally decode the file sent by the sender node. Secure network coding (SNC) protocols use cryptographic techniques to prevent these attacks: the sender authenticates each packet by attaching a small tag to it. These authentication tags are generated using homomorphic message authentication codes (MACs) [11] or homomorphic signatures [12], [13], [14], [15]. Due to homomorphic property, an intermediate node can combine incoming packets (and their tags) into a packet and its tag. In this work, we look at the problem of constructing a secure cloud storage protocol for dynamic data (DSCS) from a different perspective. We investigate whether we can construct an efficient DSCS protocol using an SNC protocol. In a previous work, Chen et al. [16] reveal a relationship between secure cloud storage and secure network coding. In particular, they show that one can exploit some of the algorithms involved in an SNC protocol in order to construct a secure cloud storage protocol for static data. However, their construction does not handle dynamic data — that makes it insufficient in many applications where a client needs to update (insert, delete or modify) the remote data efficiently. Further investigations are needed towards an efficient DSCS construction using a secure network coding (SNC) protocol. Network coding techniques have been used to construct distributed storage systems [17], [18] where the client's data are disseminated across multiple servers. However, they primarily aim to reduce the repair bandwidth when some of the servers fail. On the other hand, we explore whether we can exploit the algorithms involved in an SNC protocol to construct an efficient and secure cloud storage protocol for dynamic data (for a single storage server). Although dynamic data are generic in the sense that they support arbitrary update (insertion, deletion and modification) operations, append-only data (where new data corresponding to a data file are inserted only at the end of the file) find numerous applications as well. These applications primarily maintain archival as well as current data by appending the current data to the existing datasets. Examples of append-only data include data obtained from CCTV cameras, ledgers containing monetary transactions, medical history of patients, data stored at append-only databases, and so on. Append-only data are also useful for maintaining other log structures (e.g., certificates are stored using append-only log structures in certificate transparency schemes [39]). In many of such applications, the data owner requires a cloud server to store the bulk data in an untampered and retrievable fashion with append being the only permissible update. Although secure cloud storage schemes for generic dynamic data also work for append-only data, a more efficient solution (specific to append-only data files) would be helpful in this scenario. Our Contribution: Our major contributions in this work are summarized as follows. • We explore the possibility of providing a generic construction of a DSCS protocol from any SNC protocol. We discuss the challenges for a generic construction in details and identify some SNC protocols suitable for constructing efficient DSCS protocols. • We construct a publicly verifiable DSCS protocol (DSCS I) from an SNC protocol [15]. DSCS I handles dynamic data, i.e., a client can efficiently perform updates (insertion, deletion and modification) on the outsourced data. We discuss the (asymptotic) performance and certain limitations of DSCS I. • We provide the formal security definition of a DSCS protocol and prove the security of DSCS I. • As append-only data are a special case of generic dynamic data, we can use DSCS I (which is based on [15]) for append-only data. However, we identify some SNC protocols that are not suitable for building a secure cloud storage for generic dynamic data, but efficient secure cloud storage protocols for appendonly data can be constructed from them. We construct such a publicly verifiable secure cloud storage protocol (DSCS II) for append-only data by using an SNC protocol proposed by Boneh et al. [13]. • We discuss the (asymptotic) performance of DSCS II which overcomes some limitations of DSCS I. • We implement DSCS I and DSCS II and evaluate their performance based on storage overhead, computational cost and communication cost.

2.LITERATURE SURVEY

Cloud service providers offer storage outsourcing facility to their clients. In a secure cloud storage (SCS) protocol, the integrity of the client's data is maintained. In this work, we construct a publicly verifiable secure cloud storage protocol based on a secure network coding (SNC) protocol where the client can update the outsourced data as needed. To the best of our knowledge, our scheme is the first SNC-based SCS protocol for dynamic data that is secure in the standard model and provides privacy-preserving audits in a publicly verifiable setting. Furthermore, we discuss, in details, about the (im)possibility of providing a general construction of an efficient SCS protocol for dynamic data (DSCS protocol) from an arbitrary SNC protocol. In addition, we modify an existing DSCS scheme (DPDP I) in order to support privacy-preserving audits[4]. We also compare our DSCS protocol with other SCS schemes (including the modified DPDP I scheme)[5]. Finally, we figure out some limitations of an SCS scheme constructed using an SNC protocol.

Cloud Computing has been envisioned as the next-generation architecture of IT Enterprise. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, we consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of the client through the auditing of whether his data stored in the cloud are indeed intact, which can be important in achieving economies of scale for Cloud Computing. The support for data dynamics via the most general forms of data operation, such as block modification, insertion, and deletion, is also a significant step toward practicality, since services in Cloud Computing are not limited to archive or backup data only. While prior works on ensuring remote data integrity often lacks the support of either public auditability or dynamic data operations, this paper achieves both. We first identify the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works and then show how to construct an elegant verification scheme for the seamless integration of these two salient features in our protocol design. In particular, to achieve efficient data dynamics, we improve the existing proof of storage models by manipulating the classic Merkle Hash Tree[6] construction for block tag authentication. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multiuser setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed schemes are highly efficient and provably secure.

We introduce a new class of problems called network information flow which is inspired by computer network applications. Consider a point-to-point communication network on which a number of information sources are to be multicast to certain sets of destinations. We assume that the information sources are mutually independent. The problem is to characterize the admissible coding rate region. This model subsumes all previously studied models along the same line. We study the problem with one information source, and we have obtained a simple characterization of the admissible coding rate region. Our result can be regarded as the max-flow min-cut theorem for network information flow. Contrary to one's intuition, our work reveals that it is in general not optimal to regard the information to be multicast as a "fluid" which can simply be routed or replicated. Rather, by employing coding at the nodes, which we refer to as network coding, bandwidth can in general be saved. This finding may have significant impact on future design of switching systems.

Consider a communication network in which certain source nodes multicast information to other nodes on the network in the multihop fashion where every node can pass on any of its received data to others. We are interested in how fast each node can receive the complete information, or

equivalently, what the information rate arriving at each node is. Allowing a node to encode its received data before passing it on, the question involves optimization of the multicast mechanisms at the nodes. Among the simplest coding schemes is linear coding, which regards a block of data as a vector over a certain base field and allows a node to apply a linear transformation to a vector before passing it on. We formulate this multicast problem and prove that linear coding suffices to achieve the optimum, which is the max-flow from the source to each receiving node.

Network coding has been shown to improve the capacity and robustness in networks. However, since intermediate nodes modify packets en-route, integrity of data cannot be checked using traditional MACs and checksums. In addition, network coded systems are vulnerable to pollution attacks where a single malicious node can flood the network with bad packets and prevent the receiver from decoding the packets correctly. Signature schemes have been proposed to thwart such attacks, but they tend to be too slow for online per-packet integrity. Here we propose a homomorphic MAC which allows checking the integrity of network coded data. Our homomorphic MAC is designed as a drop-in replacement for traditional MACs (such as HMAC) in systems using network coding.

3. PROBLEM STATEMENT

we look at the problem of constructing a secure cloud storage protocol for dynamic data (DSCS)from a different perspective. We investigate whether we can construct an efficient DSCS protocol using an SNC protocol. In a previous work, Chen et al. [16] reveal a relationship between secure cloud storage and secure network coding. In particular, they show that one can exploit some of the algorithms involved in an SNC protocol in order to construct a secure cloud storage protocol for static data. However, their construction does not handle dynamic data — that makes it insufficient in many applications where a client needs to update (insert, delete or modify) the remote data efficiently. Further investigations are needed towards an efficient DSCS construction using a secure network coding (SNC) protocol [7]-[13]. Network coding techniques have been used to construct distributed storage systems where the client's data are disseminated across multiple servers. However, they primarily aim to reduce the repair bandwidth when some of the servers fail. On the other hand, we explore whether we can exploit the algorithms involved in an SNC protocol to construct an efficient and secure cloud storage protocol for dynamic data (for a single storage server).

3.1 LIMITATION OF SYSTEMS

A client having a smart phone with a low-performance processor or limited storage cannot accomplish heavy computation or store large volume of data. Under such circumstances, she can delegate her computation/storage to the cloud server. Secure cloud storage protocols enable a client to check integrity of outsourced data [14]. For static data, the client cannot change her data after the initial outsourcing (e.g., backup/archival data)

4. SNC PROTOCOL

Our major contributions in this work are summarized as follows. We explore the possibility of providing a generic construction of a DSCS protocol from any SNC protocol. We discuss the challenges for a generic construction in details and identify some SNC protocols suitable for constructing efficient DSCS protocols.

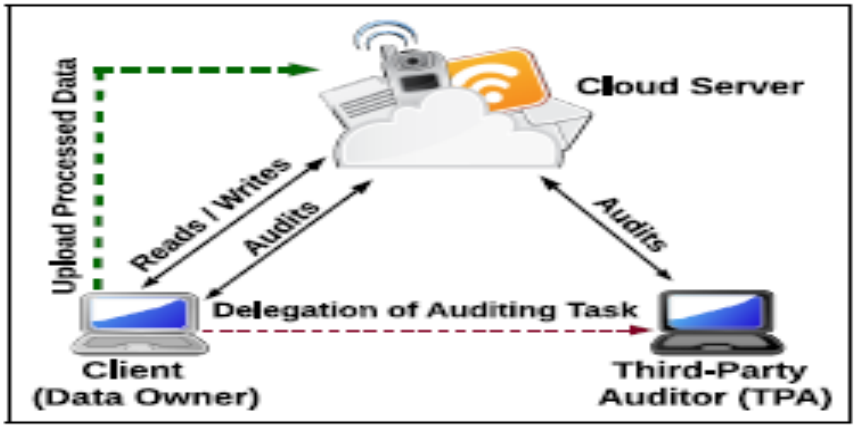
We construct a publicly verifiable DSCS protocol (DSCS I) from an SNC protocol. DSCS I handles dynamic data, i.e., a client can efficiently perform updates (insertion, deletion and modification) on the outsourced data. We discuss the (asymptotic) performance and certain limitations of DSCS I. We provide the formal security definition of a DSCS protocol and prove the security of DSCS I. As append-only data are a special case of generic dynamic data, we can use DSCS I (which is based on) for append-only data. However, we identify some SNC protocols that are not suitable for building a

secure cloud storage for generic dynamic data, butefficient secure cloud storage protocols for appendonly data can be constructed from them. We construct such a publicly verifiable secure cloud storage protocol (DSCS II) for append-only data by using an SNC protocol proposed by Bonehetal [15]-[16]]. We discuss the (asymptotic) performance of DSCS II which overcomes some limitations of DSCS I [17]-[18]. We implement DSCS I and DSCS II and evaluate their performance based on storage overhead, computational cost and communication cost.

4.1 PURPOSE OF SYSTEM:

Secure network coding (SNC) protocols use cryptographic techniques to prevent these attacks: the sender authenticates each packet by attaching a small tag to it. we look at the problem of constructing a secure cloud storage protocol for dynamic data (DSCS) from a different perspective [19]. We investigate whether we can construct an efficient DSCS protocol using an SNC protocol.

5. SYSTEM ARCHITECTURE



6. IMPLEMENTATION

6.1.Data Owner

In this application the owner is one of the main module for uploading the files and view the uploads file which are uploaded by the owner before do all these operations the owner should register with the application and the owner should authorized by the cloud [20].

6.2. TPA(Trusted Third Party)

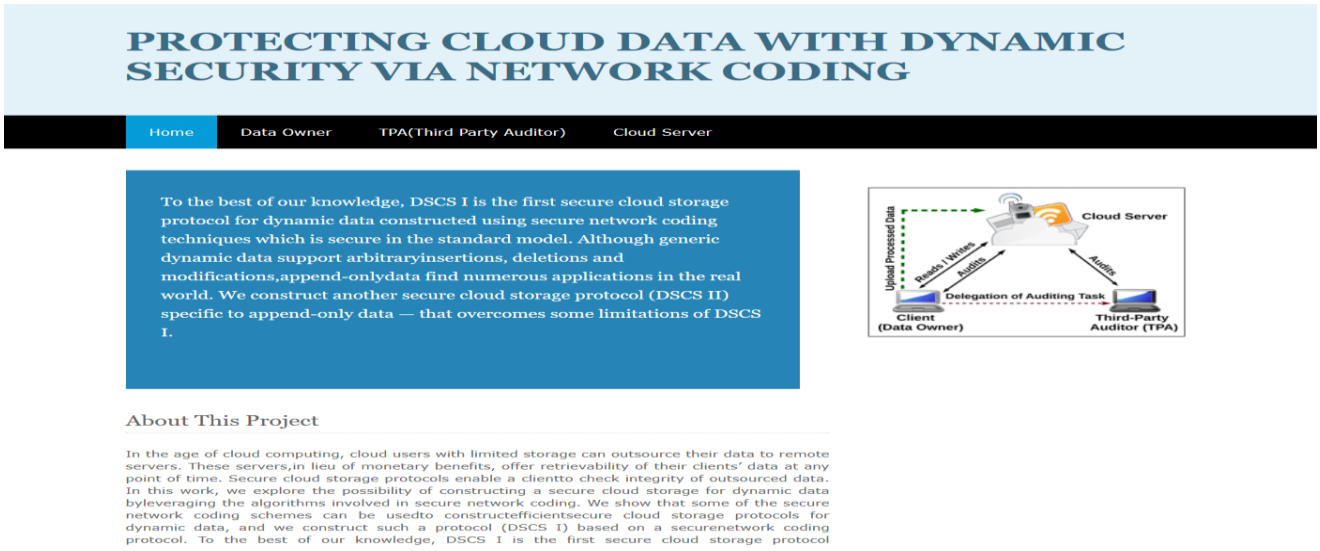
The TPA is used to generate the Auditing Task for the requested users.Here the trapdoor should login directly with the application

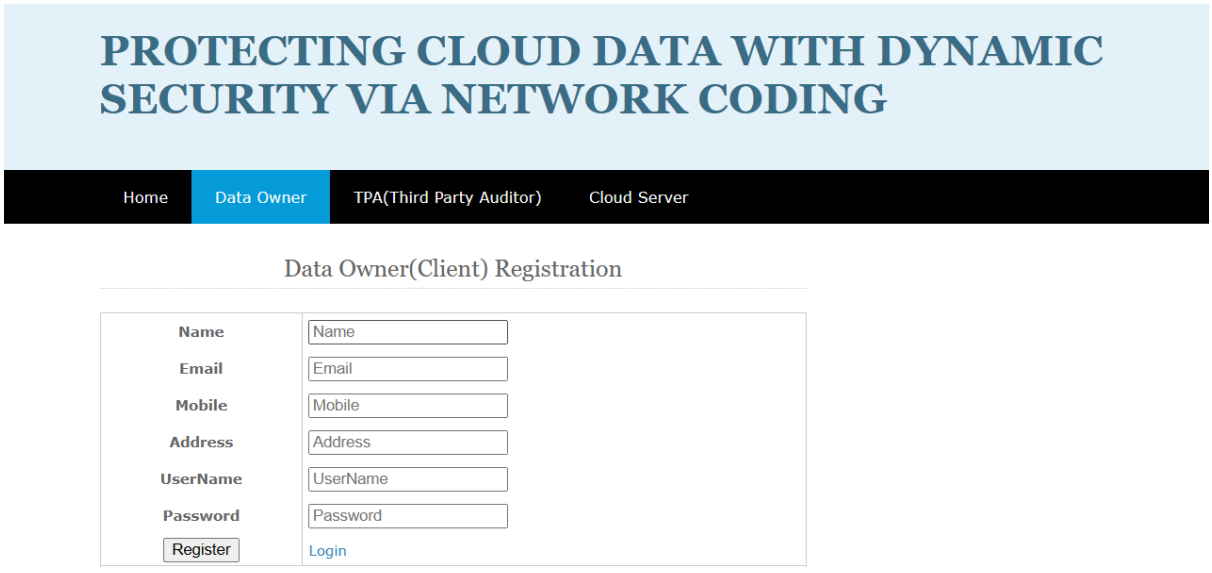
6.3.Cloud Server

The cloud is the main module to operate this project in the users activations, owner activation and also the cloud can check the following operations like search permission provides to the users, can check the top-k searched keyword, top-k similarity in chart, top-k searched keyword in chart. Primarily the cloud should login. Then only the cloud can perform the abovementioned actions.

7. EXPECTED OUTPUT RESULTS

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024





Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

8.CONCLUSION

In this work, we have proposed a secure cloud storage protocol for dynamic data (DSCS I) based on a secure network coding (SNC) protocol. To the best of our knowledge, this is the first SNC-based DSCS protocol that is secure in the standard model and enjoys public verifiability. We have discussed some challenges while constructing an efficient DSCS protocol from an SNC protocol. We have also identified some limitations of an SNC-based secure cloud storage protocol for dynamic data. However, some of these limitations follow from the underlying SNC protocol used. A more efficient SNC protocol can give us a DSCS protocol with better efficiency. We have also identified certain SNC protocols suitable for append-only data and constructed an efficient DSCS protocol (DSCS II) for appendonly data. We have shown that DSCS II overcomes some limitations of DSCS I. Finally, we have provided prototype implementations of DSCS I and DSCS II in order to show their practicality and compared the performance of DSCS I with that of an SNC-based

9. REFERENCES

[1] B. Sengupta and S. Ruj, "Publicly verifiable secure cloud storage for dynamic data using secure network coding," in ACM Asia Conference on Computer and Communications Security, 2016, pp. 107– 118.

[2] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in ACM Conference on Computer and Communications Security, 2007, pp. 598–609.

[3] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files," in ACM Conference on Computer and Communications Security, 2007, pp. 584–597.
[4] H. Shacham and B. Waters, "Compact proofs of retrievability," Journal of Cryptology, vol. 26, no. 3, pp. 442–483, 2013.
[5] C. C. Erway, A. Kupc¸ ¨ u, C. Papamanthou, and R. Tamassia, ¨ "Dynamic provable data possession," ACM Transactions on Information and System Security, vol. 17, no. 4, pp. 15:1–15:29, 2015.
[6] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 5, pp. 847–859, 2011.
[7] D. Cash, A. Kupc¸ ¨ u, and D. Wichs, "Dynamic proofs of ¨ retrievability via oblivious RAM," in EUROCRYPT, 2013, pp. 279–295.
[8] E. Shi, E. Stefanov, and C. Papamanthou, "Practical dynamic proofs of retrievability," in ACM Conference on Computer and Communications Security, 2013, pp. 325–336.
[9] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," IEEE Transactions on Information Theory, vol. 46, no. 4, pp. 1204–1216, 2000.
[10] S. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," IEEE Transactions on Information Theory, vol. 49, no. 2, pp. 371–381, 2003.
[11] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in International Conference on Applied Cryptography and Network Security, 2009, pp. 292–305.
[12] D. X. Charles, K. Jain, and K. E. Lauter, "Signatures for network coding," International Journal of Information and Coding Theory, vol. 1, no. 1, pp. 3–14, 2009.
[13] D. Boneh, D. M. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in International Conference on Practice and Theory in Public Key Cryptography, 2009, pp. 68–87.
[14] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure network coding over the integers," in International Conference on Practice and Theory in Public Key Cryptography, 2010, pp. 142–160.
[15] D. Catalano, D. Fiore, and B. Warinschi, "Efficient network coding signatures in the standard model," in International Conference on Practice and Theory in Public Key Cryptography, 2012, pp. 680–696.
[16] F. Chen, T. Xiang, Y. Yang, and S. S. M. Chow, "Secure cloud storage meets with secure network coding," in IEEE International Conference on Computer Communications, 2014, pp. 673–681.
[17] A. G. Dimakis, B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," IEEE Transactions on Information Theory, vol. 56, no. 9, pp. 4539–4551, 2010.
[18] K. Omote and T. T. Phuong, "DD-POR: Dynamic operations and direct repair in network coding-based proof of retrievability," in International Computing and Combinatorics Conference, 2015, pp. 713–730.
[19] "Secure cloud storage," 2018. https://github.com/akankshadixit/ Secure Cloud Storage
[20] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," Discrete Applied Mathematics, vol. 156, no. 16, pp. 3113–3121, Sep. 2008.