International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 11 | Nov -2016 www.irjet.net p-ISSN: 2395-0072

Network Coding approach for Secure Cloud Storage

Gauri Swapnil Bobade

Pillai HOC College of Engineering, Rasayani, Dist Raigad, Maharashtra, India

---

Abstract - This paper focuses the ingrained relationship between secure cloud storage and secure network coding. The secure cloud storage protocol is that the user can check the data integrity without possessing the actual data. The secure network coding uses the concept of data fragmentation. Though different and studied independently they can work together to give effective results. It shows systematic construction of secure cloud storage protocol when secure network coding protocol is used with it.

Further two specific secure cloud storage protocols based on two recent secure network coding protocols are proposed. First is security mediated anonymous cloud storage is proposed and second is third party auditable secure cloud storage. It will give us the effective and efficient mechanism for secure cloud storage.

Key Words: **Cloud storage auditing, network coding, security, fragmentation, user anonymity, third-party public auditing**

INTRODUCTION

Cloud storage is a model of data storage in which the digital data is stored in logical pools, the physical storage spans multiple servers and the physical environment is typically owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible, and the physical environment protected and running. 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, web service application programming interface 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.

Network coding is a routing paradigm where a router in the network sends out encoded data packets, which are a function of received data packets, instead of the traditional store-and-forward approach. Encoding can increase the network capacity for multicast tasks. Linear coding, in which a router sends out a linear combination of received data packets, is proved to be sufficient to achieve the increased capacity. This is especially useful in cooperative networks.

SYSTEM ARCHITECTURE

1.1 Secure Cloud Storage

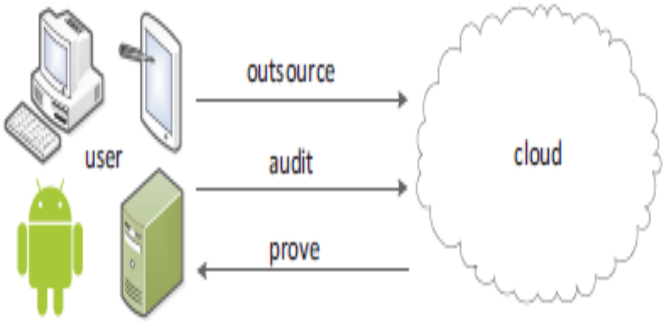

Fig1: Secure Cloud Storage

We model a secure cloud storage system as shown in Fig 1. There are two entities: user and cloud. In practice, user could be an individual, a company, or an organization using a PC or a mobile phone, etc. A cloud could be any CSP, e.g., Amazon S3, Dropbox, Google Drive, etc. The user first outsources its data to the cloud.

Later, the user periodically performs an audit on the integrity of outsourced data. The user can then check whether the proof returned from the cloud is valid or not, meaning that the data remains intact, or obtaining evidence that the data has been tampered which will possibly incur some further action, such as legal action or data recovery. Similar to previous work [3], [4] and as motivated earlier in this paper, we model the cloud as potentially malicious. We assume the communication between the user and the cloud is authenticated, which can be done by standard techniques. Thus, we can focus our attention on the user and the cloud but not communication. A secure cloud storage system that enables a user to check the integrity of the outsourced data is expected to be:

Correct: If the cloud indeed stores the whole outsourced data, the cloud can always prove to the user that the data remains intact.

Secure: If the user's data is damaged, the user can detect with high probability in the audit query, even if the cloud tries to cover the event.

Efficient: The computation, storage, and communication cost of both the user and the cloud should be as small as possible.

Secure cloud storage i.e. SCS protocol contains five efficient algorithms:

SCS = KeyGen, Outsource, Audit, Prove, Verify as follows:

KeyGen (λ) *K*: On input a security parameter λ, the user runs this algorithm to generate a secret key *K* to enable auditing and verification.

Outsource (*F*; *K*) *F*: On input the data *F* to be outsourced, the user runs this algorithm to get the processed data *F* using the secret key *K*. The processed data contains some authentication information of the data *F* and is then sent to the cloud.

Audit (*K*) *q*: The user runs this algorithm to generate an audit query *q* to be sent to the cloud.

Prove (*q; F*) Ґ: On input an audit query *q*, the cloud computes a proof Ґ using the stored data *F′*.

Verify(*q; Ґ;K*) *δ*: On input an audit query *q* and the cloud's proof Ґ, the user checks if the cloud's proof is valid using the secret key *K*. The user outputs $\delta = 1$ if the proof is valid, else outputs $\delta = 0$.

1.2 Secure Network Coding

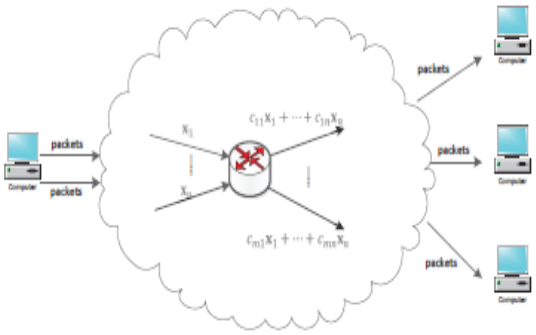

Fig 2: Secure Network Coding

Fig 2 shows a typical system that employs the network coding technique. There are three types of entities: sender, router, and receiver. A sender wants to broadcast some data to a group of receivers. The sender divides the data into packets and sends a linear combination of the packets via the network. A router in the network also sends a linear combination of the received data packets to its next hops. When a receiver obtains sufficient encoded data packets, it can decode them to recover the original data by solving a system of linear equations. To prevent a malicious router from modifying a packet, the sender attaches some authentication information with each data packet. When a router receives a series of packets, the router first checks their correctness, then combines the received correct packets, and finally sends out the combined packet together with the combined authentication information. The combined authentication information is computed according to the details of a specific protocol.

A secure network coding (SNC) protocol contains four efficient algorithms:

SNC = (KeyGen, Auth, Combine, Verify) as follows:

KeyGen (λ) (*SK; PK*): On input a security parameter λ, the sender runs this algorithm to generate a secret key *SK* and a public key *PK* to enable packet authentication.

Auth (x*i*; *SK*) (x*i*; *ti*): On input a packet **x***i* pF_p^{n+m} to be sent out in the network, the sender computes an authentication information *ti* and then sends out (**x***i*; *ti*).

Combine ({u*i*; *ti*}*i*=1;:::;*l*; {*c*1; : : : ; *cl*}) → (w; *t*): On receiving a group of packets **u***i* F_p^{n+m} and their authentication information *ti*'s, a router runs this algorithm to generate a combined packet **w** F_p^{n+m} with coefficients {*c*1; : : : ; *cl*} and the combined authentication information *t*.

Verify (w; *t*) _: On input a packet **w** F_p^{n+m} and its authentication information *t*, a receiver or a router runs this algorithm to check whether a packet is modified maliciously. If the packet is correct, it outputs δ = 1, else outputs δ = 0.

CONSTRUCTION

Construction of a secure cloud storage protocol SCS = (KeyGen, Outsource, Audit, Prove, Verify) given a well-designed secure network coding protocol
SNC = (KeyGen, Auth, Combine, Verify) is done.
Construction follows the following algorithm:

SCS = (KeyGen, Outsource, Audit, Prove, Verify)

KeyGen (λ) *K*: The user determines the finite field F*p* where the network coding works over. The user also determines the packet size *n* and the total number of packets *m*. Then the user runs SNC. **KeyGen (λ) (*SK; PK*).** The key is *K* = (*SK; PK*). Outsource(*F*;*K*) *F*: On input the data *F* to be outsourced, the user takes *F* as a collection of vectors {v*i*}*i*=1;:::;*m* in F_p^n. Take each v*i* as a codeword and then attach it with a coefficient vector e*i* to get x*i* = [v*i* e*i*] F_p^{n+m}. The user runs SNC.

Auth(x*i*; *SK*) → (x*i*; *ti*) to get the authentication information. The user then outsources the processed data *F*′ = {v*i*; *ti*} *i*=1;:::;*m* together with the public key *PK* to the cloud. Only the v*i*'s are outsourced but not the x*i*'s.

Audit(*K*) *q*: The user runs this algorithm to generate a collection of uniformly random numbers {*ij* ; *cj*}*j*=1;:::;*l* where 1 ≤ *ij* ≤ *m* and *cj* F*p*. The user sends the query *q* = {*ij* ; *cj*}*j*=1;:::;*l* to the cloud. To achieve a good security level, the user sends multiple independent audit queries during one audit process.

Prove (*q; F*) Ґ: On receiving an audit query *q* = {*ij* ; *cj*}*j*=1;:::;*l* where *l* is the length of the query, the cloud augments **v***ij* with the unit coefficient vector **e***ij* to get a codeword **x***ij* for all *j*. The cloud runs SNC. Combine({**u***i*; *ti*}*i*=1;:::;*l*; {*c*1; : : : ; *cl*}) (**w**; *t*) where **u***i* = **x***ij* . The cloud extracts the first *n* entries of **w** as a vector **y** F_p^n . The cloud sends back (**y**; *t*) as a proof of the corresponding query. The equation **y** = Σ*l j*=1 *cj* · **v***ij* holds.

Verify(*q; Ґ;K*) *δ*: On input an audit query *q* = {*ij* ; *cj*}*j*=1;:::;*l*, the cloud's proof Ґ = (**y**; *t*), the user constructs a vector **w** F_p^{n+m} such that the first *n* entries of **w** are the same as **y**, the (*n* + *ij*)- entry is *cj* , and all other entries are 0. The user runs SNC. Verify (**w**; *t*) *δ* to get an answer δ. If δ = 1, the outsourced data remains intact and output 1, else the outsourced data is damaged and output 0.

Anonymity via Security Mediator

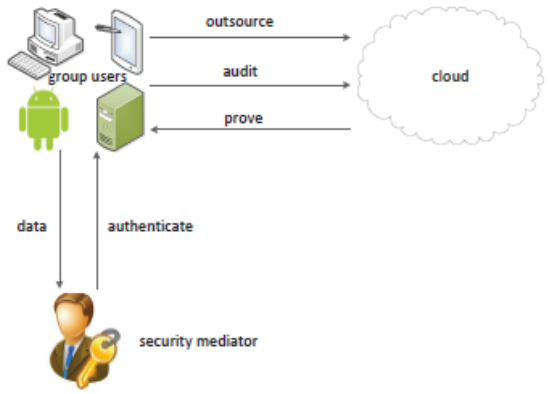

Fig 3: Security mediator

A group of users in an organization outsource their data to the cloud through a gateway called security-mediator. The security-mediator serves as a separate entity that can help generating authentication information of the data for the users. However, the

users desire to keep their data secret from the security-mediator. When one user wants to outsource some data to the cloud, the user first blinds the data in some secret way, and then sends the blinded data to the security-mediator. On receiving the blinded data, the security-mediator authenticates the blinded data and sends the authentication back, from which the data user recovers the real authentication information. Since the authentication information of the outsourced data for every user is generated with the help of the same security-mediator, the cloud cannot differentiate between different users from the outsourced data. In this way, the user anonymity is preserved. Later, the user sends the data and its authentication information to the cloud via an anonymous channel. In case authentication is needed, this can be done by existing mechanism. To ensure the integrity of the outsourced data, one user sends audit queries to the cloud. On receiving an audit query, the cloud answers the query according to the protocol with a proof. Finally, the data user verifies the result returned from the cloud. We can extend our generic construction using the idea of Wang et al into a security-mediated secure cloud storage protocol. Again, we note that the only existing proposal only has a security proof in the random oracle model. We thus obtain an SM-SCS in the standard model if the underlying secure network coding protocol does not need it. It is worth noting that the construction shows a generic way to design anonymous secure cloud storage protocols for the first time.

Security-mediated secure cloud storage protocol (SM-SCS) contains seven efficient algorithms **(KeyGen, Blind, Unblind, Outsource, Audit, Prove, Verify).**

KeyGen(λ) ***K*:** The security-mediator runs this algorithm. SCS. KeyGen employs SNC KeyGen to generate the key. The security-mediator keeps the secret key only known by itself and shares the public key with both the users and cloud.

Blind(w) $\{\mathbf{w}i\}i=1;:::;l$: To get the authentication information for a data block **w** in F_p^{n+m} , the user generates *l* random vectors **w***i* such that **w** =Σ*l* *i*=1 *ci***w***i* for some secret coefficients *ci*'s. The user asks the security-mediator to return the authentication information *ti* for **w***i* using SNC .Auth.

Unblind ({*wi; ti; ci*}) *t*: The authentication information has some linear homomorphism as shown in SCS. Prove which is used by cloud to compute the authentication of some linearly combined data blocks. The user thus can also use this linear homomorphism to compute the authentication information for **w** in the same way as SCS. Prove. Thus, the authentication information for the unmasked data block **w** = Σ*l* *i*=1 *ci***w***i* can be obtained.

Outsource (*F;K*) *F*: The user first blinds the data to be outsourced as in the SM SCS. Blind algorithm and then obtains the real authentication of the data as in SM-SCS. Unblind. Later, the user sends the data and its authentication information to the cloud.

Audit (*K*) *q*: The user runs this algorithm and the detailed process is the same as SCS Audit.

Prove (*q; F*) Ґ: The cloud runs this algorithm and the detailed process is the same as SCS. Prove.

Verify (*q; Ґ; K*) *δ*: The user runs this algorithm as same as SCS. Verify, i.e. the user only needs to check if the authentication of the returned result is correct.

Third-party Public Auditing

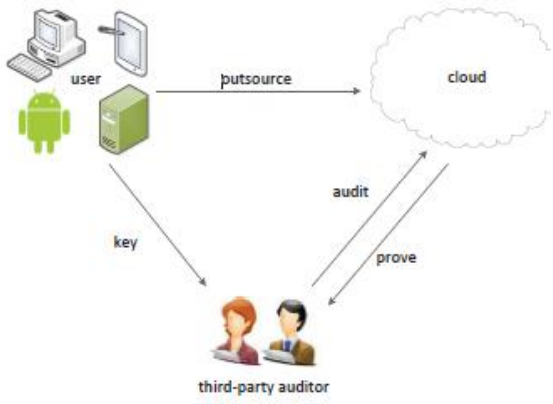

Fig 4: Third party public auditing

For a complete outsourcing solution, it is desirable to have the auditing also outsourced to a third-party. This enhancement can reduce the burden of the user by shifting the auditing responsibility to the third-party public auditor. This paradigm is reasonable since the third-party auditor

could have more experience and knowledge than the user, and thus the auditing is more convincing and neutral to both parties. However, the fact that the auditing does not require the original data does not mean that the auditing does not leak the original data. This problem is identified by Wang et al. [5]. Their solution is a zero-knowledge secure cloud storage protocol. The setup is shown in Fig. 5. The data user generates the key and then outsources the data to the cloud. Any third-party auditor can audit the data by only using the public key. To check if the outsourced data remains intact in the cloud, the auditor sends a series of audit queries to the cloud, and the cloud responds to the queries by returning the proofs. The auditor then can check the proof to see if the data is still intact.

A third-party publicly auditable secure cloud storage protocol **TP-SCS = (KeyGen, Outsource, Audit, Prove, Verify)** as follows:

KeyGen(λ) *K*: Everything is the same as SCS. Besides, the user shares the public key with both the third-party auditor and the cloud.

Outsource (*F*; *K*) *F*: The user first runs SCS. Later, the user also sends some random data blocks **w***i* in F_p^{n+m} with coefficients set to 0 together with their authentications using SNC.Auth.

Audit(*K*) → *q*: The third-party auditor runs this algorithm to generate a collection of uniformly random numbers *{ij ; cj}j*=1;:::;*l* where 1 ≤ *ij* ≤ *m* and *cj* ∈ F*p*. The third-party auditor sends the query *q* = *{ij ; cj}j*=1;:::;*l* to the cloud. To achieve a good security level, the third-party auditor sends multiple independent audit queries during one auditing process. This process is the same as SCS Audit.

Prove (*q; F*) Ґ: When receiving a challenge query from the third-party auditor, the cloud computes the result and proof Ґ in the same way as in SCS. Prove to obtain (**y***; t*). Then the cloud uses the **w***i*'s to randomize Ґ by adding **y** with some random linear combination of **w***i*'s. Since the authentication of **y** and **w***i*'s are known to the cloud, this randomization step is quite easy using the linear homomorphism property of the authentication algorithm as in the original SCS. Prove. The randomization can mask the data and thus the third-party auditor cannot get non-trivial knowledge about the user's data from cloud's response. The returned proof still holds in this case and thus the third-party auditor can indeed check the integrity of the user's data.

Verify (*q; Ґ; K*) → *δ*: The third-party auditor runs this algorithm as same as SCS. Verify, i.e. the third-party auditor only needs to check whether the authentication of the returned result is correct.

Advantages

The cloud storage system is more secure than a traditional system.

Reliability of the data is more.

User identity is not disclosed to the outside world.

Disadvantages

Storage cost is more as compared to the traditional system.

Communication cost has increased.

Computation cost is enhanced.

CONCLUSION

Based on the relationship of secure cloud storage and secure network coding, a systematic way to construct a generic secure cloud storage protocol based on any secure network coding protocol is proposed. We enhance our generic construction to support user anonymity and third-party public auditing. It is also interesting to study the reverse direction, i.e. under what conditions a secure network coding protocol can be constructed from a secure cloud storage protocol.

REFERENCES

[1] Secure Cloud Storage Meets with Secure Network Codin g Chen, F.; Xiang, T.; Yang, Y.; Chow, S. Computers, IEEE Transactions on Year: 2015, Volume: PP, Issue: 99 Pages: 1 - 1, DOI: 10.1109/TC.2015.2456027

[2] International Journal of Scientific & Engineering Research, Volume 4, Issue 8, August-2013 128 ISSN 2229-5518 IJSER © 2013 http://www.ijser.org "The impact of different MAC protocols for Network Coding in Adhoc Network".

[3] A. Juels and B. Kaliski Jr, "PORs: Proofs of retrievability for large files", in ACM Conference on Computer and Communications Security (SP), 2007, pp. 584–597.

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

[5] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy preserving public auditing for secure cloud storage", IEEE Transactions on Computers, vol. 62, no. 2, pp. 362–375, 2013.

[6] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing", IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 9, pp. 1717–1726, 2013.

[7] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow", IEEE Transactions on Information Theory, vol. 46, no. 4, pp. 1204–1216, 2000.