Distributed Data Security Enhancement in Cloud Computing

Rajiv K M*, Manoj Challa *2

* Pursuing MTech, Dept of Computer Science & Engg
CMRIT, Bangalore, India

* Associate Professor, Dept of Computer Science & Engg
CMRIT, Bangalore, India

Abstract— Rapidly growing information technology and connectivity to the internet in all forms of hand-held devices to mainframe computers of bigger organizations have one important need and that is connectivity to source of data and internet from any location possible in a secure and dependable manner. The cloud computing comes to us as a solution to that craving need which is in vogue in the IT industries today. Its multiple features which addresses secure data storage, provision of infrastructure on demand and scaling as and when required, provision of platform and software as a service to the clients is one of the most striking features.

Our focus regarding cloud computing is strictly concentrated on the dependability, reliability, data integrity and security of the data stored in the cloud. Shamir’s secret sharing algorithm is the main method that we have found very intuitively useful for the purpose. It is again padded with extra security layer by using HMAC signature hashing method. Our concept of securing the data makes such an attractive sandwich of security to data that it will be in a systematic way of positive discussion for sure.

Keywords— Shamir’s secret sharing algorithm, cloud computing, data storage, data integrity, HMAC

I. INTRODUCTION

The cloud computing is a cost effective, service availability, versatile and on-demand service delivery platform for providing business through the internet[2]. Cloud computing guarantees to extend the speed with that applications are deployed, increase innovation, and lower prices all right away while increasing business flexibility and productivity.

The main concept of cloud computing is moving huge amounts of data off the site to a trusted computing environment which will take care of the data with a set of well defined and efficient ways. The cloud service providers make sure of the security, ease of access, scalability of data on demand systematic speedy delivery of services. The major issue that the cloud is facing these days is security.

Aim of this paper is focused on this very need. The data integrity must be preserved irrespective of the system failures, network failures and form any kind of unauthorized access and corruption of data. Our approach makes it sure as long as the cloud infrastructure and environment of it is up and running, it can take on any kind of situations of any kind with great ease.

II. OBJECTIVE

Cloud computing is a concept that is making lot of news due to its features and flexibility at low cost. Well its relatively a new concept but it is based on not so many new technologies. However there is a basic security criteria need to be met. The objectives of our paper being on security is as follows.

- Data integrity
- Providing enhanced security measures for the data to be stored by the users
- Deal with data corruption
- Preserve correctness of data

III. ALGORITHM USED

Shamir’s Secret Sharing [4] is associate algorithmic rule in cryptography. It’s a type of secret sharing, when a secret is split into components, giving every participant its own distinctive part, wherever a number of the components or all of them are required so as to reconstruct the key.

Counting on all participants to mix along the key may well be impractical, and so sometimes the threshold theme is employed wherever any of the components are enough to reconstruct the first secret.

A. Mathematical definition

Formally, our goal is to divide some data \( D \) (e.g., the safe combination) into \( n \) pieces \( D_1, \ldots, D_n \) in such a way that:

1. Knowledge of any \( k \) or more \( D_i \) pieces makes \( D \) easily computable.
2. Knowledge of any \( k - 1 \) or fewer \( D_i \) pieces leaves \( D \) completely undetermined (in the sense that all its possible values are equally likely).

This scheme is called \((k, n)\) threshold scheme. If \( k = n \) then all participants are required to reconstruct the secret.

B. Shamir’s secret-sharing scheme

The essential plan of Adi Shamir’s threshold theme is that a pair of points are adequate to describe a line, three points are adequate to outline a cubic, four points to outline a cubic curve so forth. That is, it takes \( k \) points to describe a polynomial of degree \( k - 1 \).
Suppose we want to use a \((k, n)\) threshold scheme to share our secret \(S\), without loss of generality assumed to be an element in a finite field \(F\) of \(0 < k \leq n < P\) where \(P\) is a prime number.

Choose at random \(k - 1\) coefficients \(a_1, \ldots, a_{k-1}\) in \(F\), and let \(a_0 = \tilde{S}\). Build the polynomial

\[
f(x) = a_0 + a_2 x + a_2 x^2 + a_2 x^3 + \cdots + a_{k-1} x^{k-1}.
\]

Let us construct any \(n\) points out of it, for instance set \(i = 1, \ldots, n\) to retrieve \((i, f(i))\). Every participant is given a point (a pair of input to the polynomial and output). Given any subset of \(k\) of these set of pairs, we will find the coefficients of the polynomial victimization interpolation and the secret is that the constant term \(a_0\).

C. Usage

Example

The following instance illustrates the fundamental plan.

Preparation

Suppose that our secret is 129 \((S = 129)\) and our prime number is \((P = 257)\).

We wish to divide the secret into six parts \((n = 6)\), where any subset of 3 parts \((k - 3)\) is sufficient to reconstruct the secret. At random we obtain 2 \((k - 1)\) numbers: 43 and 199.

\((a_1 = 43; a_2 = 199)\)

Our polynomial to produce secret shares (points) is therefore:

\[
f(x) = 129 + 43x + 199x^2
\]

We construct six points from the polynomial:

\((1, 114); (2, 240); (3, 250); (4, 144); (5, 179); (6, 98)\)

We provide every participant a distinct single point (both \(x\) and \(f(x)\)).

Reconstruction

In order to reconstruct the secret any three points are enough. Let us consider

\[(x_0, y_0) = (2, 240); (x_1, y_1) = (4, 144); (x_2, y_2) = (5, 179)\]

We compute Lagrange basis polynomials:

\[
\ell_0 = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{x - 2}{1 - 2} \cdot \frac{x - 3}{1 - 3} = \frac{1}{2} x^2 - \frac{5}{2} x + 6
\]

\[
\ell_1 = \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} = \frac{x - 2}{1 - 2} \cdot \frac{x - 3}{1 - 3} = \frac{1}{2} x^2 - \frac{1}{2} x - 3
\]

\[
\ell_2 = \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} = \frac{x - 2}{3 - 1} \cdot \frac{x - 3}{3 - 1} = \frac{1}{2} x^2 - \frac{3}{2} x + 2
\]

Fitting to the finite field we get:

\[
\ell_0 = \frac{1}{2} x^2 - \frac{252}{2} x + \frac{6}{2} = \frac{1}{2} x^2 - 126 x + 3
\]

Therefore:

\[
f(x) = \sum_{j=0}^{2} y_j \cdot \ell_j(x)
\]

\[
= 240 \cdot \left(\frac{1}{2} x^2 - 126 x + 3\right) + 144 \cdot \left(-x^2 - 253 x - 3\right) + 179 \cdot \left(\frac{1}{2} x^2 + 127 x + 1\right)
\]

\[
= 199 x^2 + 43 x + 128
\]

Recall that the the key is the free coefficient, that \(S = 129\), and its done.

HMAC: Secret Key-Hashing for Message Authentication

HMAC[5], a mechanism for message authentication using cryptographic hash functions. HMAC may be used with any repetitive cryptographical hash function, e.g., MD5, SHA-1 together with a secret shared key. The cryptographical strength of HMAC depends on the properties of the underlying hash function. Providing the way to visualize the integrity of data transmitted over or contained in an unreliable medium is a prime necessity within the world of open computing and communications. Mechanisms that offer such integrity check supported a secret key are colloquially reffered to as "message authentication codes" (MAC). Uniquely, message authentication codes are used between 2 parties that share a secret key so as to validate data transmitted between these parties. During this document we offer such a MAC mechanism supported cryptographical hash functions. This mechanism, referred to as HMAC, relies on work by the authors [5] where the development is conferred and cryptographically analyzed. We check with to that work for the elaboration on the principle and security analysis of HMAC, and its comparison to alternative keyed-hash strategies. HMAC may be employed in combination with any iterated cryptographical hash function. MD5 and SHA-1 are illustrations of such hash functions. HMAC conjointly uses a secret key for calculation and verification of the message authentication values. The important goals behind this construction are

* To use, with no modifications, current hash functions. Particularly, hash functions perform well in software packages, and that code is freely and openly accessible.

* To protect the authentic conduct of the hash function with no incurring major degradation.

* To use and handle keys in very straightforward approach.

* To obtain a well understood cryptographical analysis of the strength of the authentication mechanism supported reasonable assumptions on the underlying hash function.
Specific instantiations of HMAC got to outline a specific hash function. Present candidates for similar hash functions embody SHA-1 [9], MD5 [8], RIPEMD-128/160 [10]. These completely different realizations of HMAC are going to be denoted by HMAC-SHA1, HMAC-RIPEMD, HMAC-MD5, etc.

To the date MD5 and SHA-1 are the foremost popularly used cryptographical hash functions. MD5 has been recently shown to be susceptible to collision search attacks [7]. This attack and other presently better known weaknesses of MD5 don’t compromise the employment of MD5 inside HMAC as specified in this document (see [7]); but, SHA-1 seems to be a cryptographically stronger function.

The definition of HMAC needs a cryptographical hash function, that we denote by H, and a secret key K. We assume H to be a cryptographical hash function where information is hashed by iterating a basic compression function on blocks of information. We denote by B the byte-length of such blocks (B=64 for all the previously mentioned samples of hashed functions), and as of L the byte-length of hash results (L=16 for the MD5, and L=20 for the SHA-1). The authentication key K may be of any length till B, the block length of hashing function. Applications that use keys longer than B bytes can first hash the key utilizing H then use the resultant L byte string because the actual key to HMAC. In any case the lowest suggested length for K is L bytes (as the hash output length).

See section three for lot of data on keys.

We outline 2 fixed and totally different strings ipad and opad as follows

(1) add zeros to the end of K to create a B byte string (e.g., if K is of length twenty bytes and B=64, then K will be attached after with 44 zero bytes 0x00)

(2) XOR (bitwise exclusive-OR) the B byte string computed in step (1) with ipad

(3) append the stream of information 'text' to the B byte string obtained from step (2)

(4) use H to the stream populated in step (3)

(5) XOR (bitwise exclusive-OR) the B byte string computed in step (1) with opad

(6) append the H result from step (4) to the B byte string resulting from step (5)

(7) employ H to the stream generated in step (6) and output the result

The key for HMAC may be of any length (keys longer than B bytes are 1st hashed utilizing H). However, less than L bytes is powerfully discouraged because it would decrease the safety strength of the function. Keys long enough than L bytes are acceptable however the additional length wouldn’t considerably increase the function strength. (A longer key could also be recommended if the randomness of the key is thought of weak.)

Keys got to be chosen haphazardly (or employing a cryptographically robust pseudo-random generator seeded with a random seed), and timely refreshed. (Current attacks don’t indicate a particular suggested frequency for key changes as these attacks are practically unfeasible. However, periodic key refreshment is a basic security regular habit that helps against potential weaknesses of the function and keys, and limits the harm of an exposed key.) HMAC is outlined in such a way that the underlying hash function H may be used with no modification to its code. Specially, it uses the function H with the pre-defined initial value IV (a fixed value such as by every repetitious hash function to initialize its compression function). But, if desired, a performance improvement may be achieved at the price of (possibly) modifying the code of H to support variable IVs.

The concept is that the intermediate results of the compression function on the B-byte blocks (K XOR ipad) and (K XOR opad) may be precomputed just once at the time of generation of the key K, or before its 1st use. These intermediate results are saved and so utilized to initialize the IV of H every time that a message must be authenticated. This technique saves, for every authenticated message, the applying of the compression function of H on 2 B-byte blocks (i.e., on (K XOR ipad) and (K XOR opad)). Such a savings could also be vital during authenticating short streams of data. Should stress that the stored intermediate values got to be treated and guarded the same as secret keys.

Selecting to implement HMAC within the above manner is a resolution of the native implementation and has no result on inter-operability. The safety of the message authentication mechanism conferred here depends on cryptographical properties of the hash function H: the resistance to collision finding (limited to the case where the initial value is secret and random, and where the output of the function isn’t externally accessible to the miscreant), and the message authentication property of the compression function of H at what time applied to single blocks (in HMAC these blocks are partly unknown to the miscreant as they contain the results of the inner H computation and, in specific, can’t be totally chosen by the attacker). These properties, and really stronger ones, are normally assumed for hash functions of the type used with HMAC. In specific, a hash function that the above properties don’t hold would become unsuitable for many (probably, all) cryptographical applications, together with alternative message authentication schemes supported on such functions. Given the restricted confidence gained up to now as for the cryptographical strength of candidate hash functions, it is vital to look at the following 2 properties of the HMAC construction and its secure use for message authentication:

1. The development is independent of the details of the actual hash function H in use so the latter may be replaced by any other secure (iterative) cryptographical hash function.
2. Message authentication, as against to encryption, contains a “transient” result. A published breaking of a message authentication method would lead to the replacement of that scheme; however would don’t have any adversarial result on data authenticated in the past. This is often in sharp distinction with encryption, where data encrypted these days might suffer from exposure in the future if, the encryption algorithm is deciphered. The strongest attack noted against HMAC is relies on the frequency of collisions for the hash function H ("birthday attack") [7],[6], and is completely impractical for least reasonable hash functions.

As an example, if considering a hash function like MD5 where in the output length equals L=16 bytes (128 bits) the miscreant must acquire the proper message authentication tags computed (with the same secret key K) on about 2**64 noted plaintexts. This may need the processing of a minimum of 2**64 blocks beneath H, an impossible task in any realistic situation (for a block length of 64 bytes this may take 250,000 years during a continuous 1Gbps link, and while not changing the key K throughout all this time). This attack may become realistic only if serious flaws in the collision behavior of the function H are discovered (e.g. collisions found after 2**30 messages). Such a discovery would verify the immediate replacement of the function H (the effects of such failure would be way more severe for the standard uses of H in the context of digital signatures, public key certificates, etc.) An accurate implementation of the above construction, the selection of random (or cryptographically pseudorandom) keys, a safe key interchange mechanism, frequent key refreshments, and sensible secrecy protection of keys are all essential ingredients for the safety of the integrity verification mechanism provided by HMAC.

The algorithms and the concepts have been developed into a program coded in Java and the resultant snapshots of the execution successfully is put up as follows

Fig 1: Registration of clients along with their email address

Fig 2: In due course of uploading, the data is split into different shares and encrypted and hashed shares are shown.

Fig 3: The data corruption is conducted by deleting some shares from the cloud.

Fig 4: The secret key is received immediately after client sign-in, the shares that are corrupted are reported to client.
IV. IMPLEMENTATION

4.1 Data Integrity

It is not an easy task to maintain securely the data that is stored in the cloud as there will be number of users who might be accessing and modifying to their needs. To manage data in cloud computing, it is always risky to the client as they would not be having any copy of the data that they have uploaded to the cloud. The algorithm that we make use of assures us the presence of the data stored by clients in every possible situations.

4.2 Handling data corruption

There is always chance of data corruptions, malicious attacks, unauthorized modifications, etc. To tackle this very important concern about what can be done to control and curb this issue if it arises. This can be handled by using the HMAC, which generates authentic key to access data. The access key is used to check for credibility and correctness of all the blocks of the data that is stored in the different servers of the cloud environment. If in any case the data corruption occurs then immediately data corruption alert is sent to the registered owner of the data. The data then can be accessed by the owner and delete all the corrupt blocks at once. And by saving it again in the cloud implies that the data again goes through the very reliable two layers of algorithm methodologies to keep the data secure. So we can always make sure that the data is always safe and secure.

4.3 Providing enhanced security

Shamir’s secret sharing algorithm is the first important security measure. It does split the data provided by the clients and encrypts the data systematically and the algorithm provides a feature that any of the three of the split blocks of the original data provided by the client is sufficient to regenerate authentic original data. Thus the split blocks or shares of the data can be stored distributively in different servers. But while retrieving it is fine if all shares or blocks are available or not; as it can successfully regenerate data by just three shares itself.

To make sure that data is secure when it is frequently not in use also is very important. This task of periodically checking for correctness of the stored data can be performed by the client himself or he can assign this to a third party auditor(TPA). So now there is a unknown risk added that may eventually become a nightmare. TPA might mischievously try to peak into the data or corrupt it or worse delete it. Thus we deploy HMAC signature method of hashing. The output of the Shamir’s secret sharing algorithm is given as input to the HMAC and generate an unique access key. Thus we have added two more security levels to existing security model. The encrypted data is now converted to hexadecimal format or hashed. And then the access key is provided to the clients via email. The clients can entrust access key in a big organization only to few and make the data accessible only to those needed and give TPA to periodically check for correctness.
V. RESULT AND DISCUSSIONS

Security in cloud computing has become an issue of paramount importance. Thus our approach for securing the data in cloud will be one of very acceptable alternative in times to come. Our approach benefits the private clouds extensively where they seek that the data be intact inspite of any kind of adversaries.

VI. CONCLUSION

The purpose of this paper work is to enlighten on the most effective ways of securing the data in the cloud computing environment using Shamir’s secret sharing algorithm and HMAC method negating the major concerns and risks of security. The Shamir’s secret sharing scheme has a good abstract foundation which provides an excellent framework for proof and applications[11].

Our proposal has been a complete focus on one most important issue that is security of the data to be stored in the cloud. It can further be thought of improvements from this point of view exclusively and timely can be updated too. The aim should always be the same that is securing data and making cloud computing very reliable even in adverse conditions.

ACKNOWLEDGEMENT

I would like to thank Mr. Manoj Challa and Prof. Dr. Jitendranath Mungara for their help and undying faith in me throughout the work and each one very close to me who have encouraged me for this research.

REFERENCES