Bastion: A Novel Technique to Deal with Key Leakage in Cloud

Prof. S.G. Pawar, Abhishek Kumar, Ankit Kumar, Swapnil Jain
Department of Computer Engineering, SITS, Narhe, Pune

Abstract:
In cloud computing data is generated day by day so there is a need to save this data on the cloud for data privacy. But there is a problem with data security that attacker can hack the data and get the encryption key. To avoid this issue, we are going to develop a system which is useful when the key of the user is leaked and the privacy of data is in danger. In the recent years, the powerful attacker hacks the data confidentiality by getting cryptographic keys. By means of coercion or backdoor in the cryptographic software once the key is exposed you can only save the data confidentiality by limiting the access to the ciphertext. This can be achieved by sending the cipher text blocks on different servers within different admin domains. The single server decrypts the data blocks and stores into the cloud. Here we are dealing with data confidentiality against the attack. Here we proposed a bastion, a novel scheme that protects the data confidentiality even if the key is leaked. Here we are sending the blocks of a file on different servers can get their key. Bastion techniques is useful in the commercial storage system. It semantically secures the encryption modes.

Keywords: *Key exposure, data confidentiality, dispersed storage.*

Introduction

In the recent days, there is a generation of data in huge amount. And there is a massive surveillance program which breaks user's policy. So there are many security issues to access the data. For instance, though these schemes are depending on data encryption schemes to guarantee the data security the necessary things by means of backdoors, coercion. In cloud computing data is generated day by day so there is a need to save this data on the cloud for data privacy. But there is a problem with data security that attacker can hack the data and get the encryption key. To avoid this issue, we are going to develop a system which is useful when the key of the user is leaked and the privacy of data is in danger. In the recent years, the powerful attacker hacks the data confidentiality by getting cryptographic keys. By means of coercion or backdoor in the cryptographic software once the key is exposed you can only save the data confidentiality by limiting the access to the cipher text. This can be achieved by sending the cipher text blocks on different servers within different admin domains. The single server decrypts the data blocks and store into the cloud. Here we are dealing with data confidentiality against the attack. Here we proposed a bastion, a novel scheme that protects the data confidentiality even if the

key is leaked. Here we are sending the blocks of a file on different servers an get their key. Bastion techniques is useful in the commercial storage system. It semantically secures the encryption modes.

To counter with the problem of adversary attack. We proposed a bastion novel and efficient scheme which gives the authority that plaintext data cannot be recovered till the data blocks are distributed to the servers. When the encryption key is exposed so bastion achieves this by combining the use of encryption functions with the linear transform. After encryption of the data, the data is XORed and then the linear transformation is performed on data.

Problem Definition:

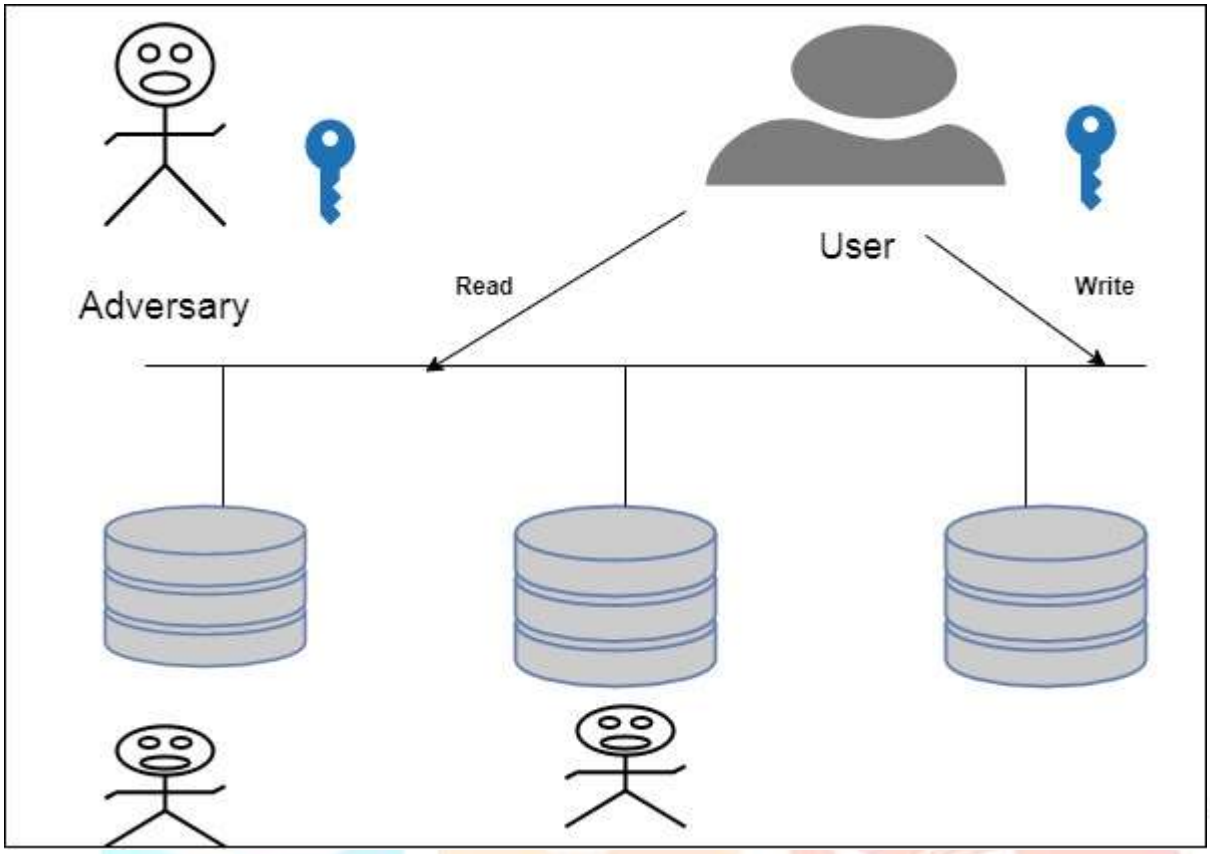
Services relied on encryption mechanisms to guarantee data confidentiality; the necessary keying material was acquired by means of backdoors, bribe, or coercion. If the encryption key is exposed, the only viable means to guarantee confidentiality is to limit the adversary's access to the ciphertext. Adversary invalidates the security of most cryptographic solutions. To counter such an adversary, a novel and efficient scheme is necessary which ensures that plaintext data cannot be recovered as long as the adversary has access to at most all but two ciphertext blocks, even when the encryption key is exposed.

Objectives

- To ensure confidentiality of data.
- To provide security to data stored on the cloud.
- To protect data in case of key leakage.
- To get the data divided into blocks i.e., chunks by getting their key.

System design:

SYSTEM ARCHITECTURE



Above fig shows system flow. In this system, user enter the personal information and enter the Username and Password. First, the user logins to the system then upload the data on the server. The file is encrypted first then exported and then divided into chunks. And the data blocks are uploaded to cloud. And at the time of file download, the reverse process is done decryption, de'xoring, and then combining the file. And at the time of file download, the key is to be sent to the users and after that user will get the file.

Related Works

SPANStore, a key-value store that exports a unified view of storage services in geographically distributed data centers. To minimize an application provider's cost, we combine three key principles. First, SPANStore spans multiple cloud providers to increase the geographical density of data centers and to minimize cost by exploiting pricing discrepancies across providers. Second, by estimating application workload at the right granularity, SPANStore judiciously trades off greater geo-distributed replication necessary to satisfy latency goals with the higher storage and data propagation costs this entails in order to satisfy fault tolerance and consistency requirements.[32] This paper present DEPSKY, a system that improves the availability, integrity and confidentiality of information stored in the cloud through the encryption, encoding and replication of the data on diverse clouds that form a cloud-of-clouds.[6] This paper describe a new dispersal scheme, called AONT-RS, which blends an All-Or-Nothing Transform with Reed-Solomon coding to achieve high security with low computational and storage costs.[25] This paper gives the first public key encryption scheme that

satisfies the definition of sender-deniability with a single encryption algorithm and the negligible probability of detection[9]. A key-value store (KVS) offers functions for storing and retrieving values associated with unique keys. KVSs have become the most popular way to access Internet-scale "cloud" storage systems. This paper presents an efficient wait-free algorithm that emulates multi-reader multi-writer storage from a set of potentially faulty KVS replicas in an asynchronous environment. [4].

HYPOTHESES

1- We are developing a system that helps a user to get the data if the key is leaked.
2- The system performs the encryption then exoring.
3- This system provides the security to the data and at the time of file download, there are keys which will be sent to the cloud.
4- Here the bastion encryption technique is used to protect data privacy.

DELIMITATION OF THE STUDY

1- To overcome the problem of security and key hacked etc. , we have developed this paper in which
2- In this paper, we used the correlation between data and user.
3- Data is encrypted and saved into chunks in the cloud.

DESIGN OF THE STUDY

Proposed Algorithm:

Input :- i(file) - file with data.

Output :- sp - file

Procedure : -

```
Split file into chunks
        If(1024* size of file)
            If(chunks=encrypt_mode)
                        Encryption
            Then
                        XOR
            Else
                    Decryption
        De'xor
        End if
        Endif
                If(chunks=decrypt_mode)
                        File download
```

End

Advanced Encryption Standard:

```
Cipher(byte in[16], byte out[16], key_array round_key[Nr+1])
begin
byte state[16];
state = in;
AddRoundKey(state, round_key[0]);
for i = 1 to Nr-1 stepsize 1 do
SubBytes(state);
ShiftRows(state);
MixColumns(state);
AddRoundKey(state, round_key[i]);
end for
SubBytes(state);
ShiftRows(state);
AddRoundKey(state, round_key[Nr]);
end
```

Encryption in Bastion:

Procedure Enc(K, x = x[1] . . . x[m])

n = m + 1

y′[n] {0, 1}l ◅ y’[n] is the IV for CTR

for i = 1 . . . n − 1 **do**

y′[i] = x[i] _ FK(y′[n] + i)

end for

t = 0l

for i = 1 . . . n **do**

t = t _ y′[i]

end for

for i = 1 . . . n **do**

y[i] = y′[i] _ t

end for

return y ◅ y = y[1] . . . y[n]

end procedure

Decryption in Bastion:

procedure Dec(K, y = y[1] . . . y[n])

t = 0l

for i = 1 . . . n **do**

t = t _ y[i]

end for

for i = 1 . . . n **do**

y′[i] = y[i] _ t

end for

for i = 1 . . . n − 1 **do**

x[i] = y′[i] _ F−1

K (y′[n] + i)

end for

return x ◅ x = x[1] . . . x[n − 1]

end procedure

SAMPLE OF THE STUDY

This paper is based on security, as we know how security is important; many users' data is hacked when the data is stored on cloud. And if the key is leaked it hard to recover data. So here we are storing our data in blocks i.e., chunks. And the key is access is given to user so that the user can recover the data.

TOOLS USED

Software Requirement: The software design of the system includes the software which requires the project.

- Operating System : Windows 8 and above.
- Application Server : Tomcat5.0/6.X
- Language : Java
- Front End : HTML, JSP
- Database : MySQL

Hardware Requirement: The hardware design of the system includes designing the hardware units and the interface between those units.

- Processor : Intel i3/i5/i7

- RAM : 4 GB (min)
- Hard Disk : 20 GB

STATISTICAL TECHNIQUE USED

We have developed the login registration which manages the user's profile. Here the user's function is to upload a file and there is encryption, decryption, Ex'oring, De'xoring of the system with a linear transformation.

Our Approach:

The System proposed in the project takes a file as an input and uses AWS cloud as a storage platform. Here there are three steps first encryption using the bastion encryption and then there is exoring of the file. And then it stores files into chunks on the cloud. And at the time of file download, it sends the keys for different chunks and by adding those keys a user can download a file. Thus the input of the system is a text file which stores data. And for encryption we are using bastion encryption and decryption we are using bastion decryption and there are exoring and dexoring.

Experiment Result:

Results demonstrate that if the key send to the user is not matched with the key entered by the user then the user will not get the file. And the file is chunked, encrypted, decrypted, before downloading.

Future scope:

The experiment results show that this system can be used in future for the purpose of security where there is a chance of hacking the large amount of data with the key. So here we can save our data if our key gets hacked by the hacker. And in the future, we are going to deal with the pdf files and mp3 or mp4 dataset.

Conclusion:

In our system we addressed the problem of securing data outsourced to the cloud against an attack which has access to the encryption key. For that purpose, we developed a system that a novel security definition that captures data confidentiality against the new adversary. We then developed Bastion technique, a scheme which gives a guarantee the confidentiality of encrypted data even when the adversary has the encryption key, and all but *two* ciphertext blocks. Bastion is most suitable for settings where the ciphertext blocks are stored in multi-cloud storage systems. In these, the attackers would need to get the encryption key and to compromise with *all* servers, in order to recover any single block of plaintext. We analyzed the security of Bastion and evaluated its performance in realistic settings. Bastion considerably improves (by more than 50%) the performance of existing primitives which offer comparable security under key exposure, and only incurs a negligible overhead when compared to existing semantically secure encryption modes (e.g., the CTR encryption mode). Finally, we showed how Bastion can be practically integrated within existing dispersed storage systems.

Reference:

[1] D. Balzarotti, M. Cova, and G. Vigna. Clearshot: Eavesdropping on keyboard input from the video. In IEEE Security & Privacy, 2008.

[2] J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In IEEE Security & Privacy, 2012.

[3] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentications schemes. In IEEE Security & Privacy, 2012.

[4] J. Bonneau, S. Preibusch, and R. Anderson. A birthday present every eleven wallets? The security of customer-chosen banking pins. In Financial Cryptography and Data Security. Springer, 2012.

[5] S. Boztas. Entropies, guessing, and cryptography. Department of Mathematics, Royal Melbourne Institute of Technology, Tech. Rep, 1999.

[6] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung. Fourth factor authentication: somebody you know. In ACM CCS, 2006.

[7] C. Cachin. Entropy measures and unconditional security in cryptography. Ph.D. thesis, Swiss Federal Institute of Technology Zurich, 1997.

[8] P. Cao, H. Li, K. Nahrstedt, Z. Kalbarczyk, R. Iyer, and A. J. Slagell Personalized password guessing: a new security threat. In ACM Proceedings of the 2014 Symposium and Bootcamp on the Science of Security, 2014.

[9] C. Castelluccia, A. Chaabane, M. D¨urmuth, and D. Perito. When privacy meets security: Leveraging personal information for password cracking. arXiv preprint arXiv:1304.6584, 2013.

[10] C. Castelluccia, M. D¨urmuth, and D. Perito. Adaptive password-strength meters from Markov models. In NDSS, 2012.