International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 4 | ISSN : 2456-3307

Mutual Key administration Protocol via Deniable Attribute-Based Encryption for Cloud Data Sharing

T. Ushapriyanka, Prof. R. Suresh

[1]M. Tech , Department of Computer Science And Engineering, Chadalawada Ramanamma Engineering College ,Chadalawada Nagar , Tirupati, India
[2]Professor, Department of Computer Science And Engineering, Chadalawada Ramanamma Engineering College , Chadalawada Nagar, Tirupati , India

ABSTRACT

Cloud storage services became increasingly in style. As a result of the importance of privacy, several cloud storage encryption schemes are proposed to protect data from those that don't have access. All such schemes assumed that cloud storage providers are safe and can't be hacked; but, in apply, some authorities might force cloud storage providers to reveal user secrets or confidential data on the cloud, so altogether circumventing storage encryption schemes. In this paper, we have a tendency to present our design for a new cloud storage encryption theme that permits cloud storage providers to create convincing fake user secrets to protect user privacy. Since coercers cannot tell if obtained secrets are true or not, the cloud storage providers ensure that user privacy continues to be securely protected.

Keywords : - Cloud Storage, encryption schemes, user privacy.

I. INTRODUCTION

Cloud storage is a type of information stockpiling where the computerized information is put away in consistent pools, the physical stockpiling traverse different servers (and frequently areas), and the physical environment is regularly claimed and taken care of by a facilitating association. These distributed storage suppliers are responsible for keeping the information accessible and available, and the physical environment ensured and running. Considering the shared property of the cloud information, attribute-based encryption (ABE) is viewed as a standout amongst the most reasonable encryption plans for distributed storage. There are various ABE plans that have been proposed. This idea originates from a unique sort of encryption plan called deniable encryption, initially proposed in. Deniable encryption includes senders and recipients making persuading fake proof of fashioned information in cipher texts with the end goal that outside coercers are satisfied.

In this paper, we introduce our plan for another distributed storage encryption conspire that empowers distributed storage suppliers to make persuading fake client privileged insights to secure client protection. Since coercers can't confess if acquired mysteries are valid or not, the distributed storage suppliers guarantee that client security is still safely ensured. In this work, we portray a deniable ABE plot for distributed storage administrations. We make utilization of ABE attributes for securing put away information with a fine-grained get to control component and deniable encryption to avert outside inspecting. Our plan is based on Waters cipher text

policy-attribute based encryption (CPABE) plot. We upgrade the Waters plot from prime request bilinear gatherings to Composite request bilinear gatherings. By the subgroup choice issue suspicion, our plan empowers clients to have the capacity to give fake insider facts that appear to be genuine to outside coercers. A focal security highlight of Attribute-Based Encryption is arrangement resistance: A challenger that grips numerous keys should just be able to get to information if no less than one individual key award get to.

The point picking this attribute-based encryption is that as more responsive, information is shared and put away by outsider locales on the Internet, there will be a need to scramble information put away at these destinations. One disservice of scrambling information is that it can be specifically shared just at a coarse-grained level (i.e., giving another gathering your private key). To beat this impediment we utilized another cryptosystem for fine-grained sharing of encoded information that we call Key Policy Attribute-Based Encryption (KP-ABE). In this cryptosystem, cipher text are marked with sets of attributes and private keys are connected with get to structures that control which cipher text by this the client can without much of a stretch ready to decode the information which was scrambled. The materialness of this development is to share the review log data and communicate encryption furthermore underpins designation of private keys which incorporates the Hierarchical Identity-Based Encryption. These Encryption plans guaranteeing that distributed storage specialist co-ops or trusted outsiders taking care of key administration are trusted and can't be hacked.

II. Proposed System

Deniable CP-ABE Our plan-ahead, bideniable, and multi-distributional CP-ABE scheme is composed of the following algorithms:

• **Setup(1^{λ})** → (P P, MSK): This algorithm takes security parameter λ as input and returns public parameter P P and system master key MSK.

• **KeyGen (MSK, S)** → SK: Given set of attributes S and MSK, this algorithm outputs private key SK.

• **Enc (P P, M, A)** → C: This encryption algorithm takes as input public parameter P P, message M, and LSSS access structure A = (M, ρ) over the universe of attributes. This algorithm encrypts M and outputs a ciphertext C, which can be decrypted by those who possess an attribute set that satisfies access structure A. Note that A is contained in C.

• **Dec(P P, SK, C)** → {M, ⊥}: This decryption algorithm takes as input public parameter P P, private key SK with its attribute set S, and ciphertext C with its access structure A. If S satisfies a, then this algorithm returns M; otherwise, this algorithm returns ⊥.

• **OpenEnc (P P, C,M)** → PE: This algorithm is for the sender to release encryption proof PE for (M, C).

• **OpenDec (P P, SK, C,M)** → PD: This algorithm is for the receiver to release decryption proof PD for (M, C).

• **Verify (P P, C, M, PE, PD)** → {T, F}: This algorithm is used to verify the correctness of PE and PD.

• **DenSetup(1^{λ})** → (P P, MSK, PK): This algorithm takes security parameter λ as input and returns public parameters P P, system master key MSK, and system public key PK. PK is known by all system users and is kept secret to outsiders.

• **DenKeyGen(MSK, S)** → (SK, FK): Given set of attributes S and MSK, this algorithm outputs private key SK as well as FK for the user, where FK will be used for generating fake proof later.

• **DenEnc (P P, PK,M,M′ , A)** → C ′ : Aside from the inputs of the normal encryption algorithm, this deniable encryption algorithm needs public key PK and fake message M′ . The output ciphertext must be indistinguishable from the output of Enc.

• **DenOpenEnc (P P, C′ ,M′)** → P ′ E: This algorithm is for the sender to release encryption proof P ′ E for fake message M′ . The output must be

indistinguishable from the result of OpenEnc and must pass the Verify algorithm.

• **DenOpenDec (P P, SK, FK, C′ ,M′)** → P ′ D: This algorithm is for the receiver to release decryption proof P ′ D for fake message M′ . The output must be indistinguishable from the result of OpenDec and must pass the Verify algorithm.

We require the following properties:

1) Security: The tuple {Setup, KeyGen, Enc, Dec} must form a secure CP-ABE scheme in a security model. In this work, we propose a CPA secure scheme and a CCA secure scheme.

2) Bi-deniability: The CP-ABE is bi-deniable if, given public parameter P P, the two distribution tuples (M, C, PE , PD) and (M′ , C′ , P′ E, P′ D) are computational indistinguishable, where M, M′ are claimed messages, C, C′ are normally and deniably encrypted ciphertexts, respectively, and PE, PD, P′ E, P′ D are proofs generated from the normal and deniable open algorithms, respectively. That is, there is no PPT algorithm A for which

$$Adv_A := \left| \begin{array}{cc} & P[A(PP,(M,C,P_E,P_D))=1] \\ - & P[A(PP,(M',C',P'_E,P'_D))=1] \end{array} \right|$$

Is non-negligible.

3) Deniable Receiver Proof Consistency: The deniable CP-ABE is deniable receiver proof consistent if a deniable receiver proof is convincing even when considering all cipher texts in the system. That is, given set of cipher texts C, including normally encrypted cipher texts and deniably encrypted cipher texts, normal proof PD and deniable proof P ′ D, there is no PPT algorithm A for which

$$Adv_A := |P[A(\mathbb{C},P_D)=1] - P[A(\mathbb{C},P'_D)=1]|$$

Is non-negligible.

We note that the last requirement is unusual for deniable encryption schemes. We build our scheme with this requirement for practicality. In a cloud storage service, it is impractical to frequently update security parameters. Therefore, coercers are able to check proofs with all stored encrypted files. For normal provided proofs, there will be no problems. So, our scheme must ensure deniable proofs to pass coercer checks, or coercers will know cheating has occurred. We also note that not all stored files are deniably encrypted. Some files are normally encrypted. A proposed receiver proof, regardless of whether it is normal or deniable, should be convincing for both normally and deniably encrypted files. We focus on receiver proofs instead of sender proofs because in most cases, senders add randomness during encryption. Therefore, any two sender proofs are usually independent, and sender proof consistency is unnecessary. For the above reasons, we build our scheme such that it adheres to the Deniable Receiver Proof Consistency requirement.

DENIABLE CP-ABE CONSTRUCTION:-

To build an audit-free secure cloud storage service, we use a deniable CP-ABE scheme as our core technology. We construct our basic deniable CP-ABE scheme, which is based on [4], as follows:

• **Setup (1^λ)** → (P P, MSK): This algorithm generates bilinear group G of order N = p1p2p3, where p1, p2, p3 are distinct primes with bilinear map function e: G × G → GT. GT is also order N. We let Gp1, Gp2, Gp3 denote three orthogonal subgroups in G of order p1, p2, p3, respectively. This algorithm then picks generators g1 ∈ Gp1, g3 ∈ Gp3, and randomly picks α, a ∈ ZN. This algorithm also chooses hash function H1: {0, 1} ∗ → Gp3. Public parameter P P is {G, e, H1, g1g3,(g1g3) a , e(g1g3, g1g3) α} and system secret key MSK is (g1g3) α.

• **KeyGen (MSK, S)** → SK: Given set S of attributes, this algorithm chooses t ∈ ZN randomly and outputs private key SK as:

$$\begin{aligned} SK &= \{(g_1g_3)^{\alpha+at}, (g_1g_3)^t, \{H_1(x)^t\}_{\forall x\in S}\} \\ &= \{K, L, \{K_x\}_{\forall x\in S}\}. \end{aligned}$$

• **Enc (P P,M, A = (M, ρ))** → C: Given message M and LSSS access structure (M, ρ). Let M be a l × n matrix and Mi denote the ith row of M. This algorithm first chooses two random vectors −→v = (s, y2. . . yn) ∈ Z n N and −→r = (r1, . . . , rl) ∈ Z l N . This algorithm then calculates λi = −→v Mi , ∀i ∈ {1, . . . , l}. In addition, this algorithm sets up one-way hash

function H(·, ·) 4 with two inputs. Note that hash function H can be any kind of one-way function and is determined during encryption. Each transaction may have different H. This algorithm flips two coins b0, b1 and picks two random string t0, t1. The output ciphertext C will be:

$$C = \{A_0, A_1, B, (C_1, D_1), \ldots, (C_l, D_l), H, t_0, t_1, V\},$$

where,

$$A_{b_0} = \mathcal{M} \cdot e(g_1g_3, g_1g_3)^{\alpha s}, A_{1-b_0} \xleftarrow{R} \mathbb{G}_T,$$
$$B = (g_1g_3)^s,$$
$$C_i = (g_1g_3)^{a\lambda_i} H_1(\rho(i))^{-r_i}, D_i = (g_1g_3)^{r_i}, i = 1 \ldots l,$$
$$V = H(\mathcal{M}, t_{b_1}) \neq H(A_{1-b_0} \cdot e(g_1g_3, g_1g_3)^{-\alpha s}, t_{1-b_1}).$$

Access structure A is also attached to C.

• **Dec(P P, SK, C) → {M, ⊥}:** To decrypt ciphertext C for access structure A = (M, ρ), this algorithm first checks if attribute set S of SK satisfies A. Suppose S satisfies A and let I ⊂ {1, 2, . . . , l} be defined as I = {i : ρ(i) ∈ S}. Then this algorithm finds a set of constants {w ∈ Zp} such that P i∈I wiλi = s. This algorithm computes M0,M1 as follows:

$$M_{\{0,1\}} = A_{\{0,1\}} \cdot \frac{\prod_{i \in I}(e(C_i, L)e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K)}$$

This algorithm then calculates

$$v_{i,j} = H(\mathcal{M}_i, t_j), \forall i, j \in \{0, 1\}.$$

If vi,j is equal to V , then Mi is the true message and is returned. Otherwise, this algorithm returns ⊥.

• **OpenEnc (P P, C, M)** → PE: This algorithm returns two coins b0, b1 as proof PE.

• **OpenDec (P P, SK, C, M)** → PD: This algorithm directly returns SK as proof PD since this is the most persuasive proof.

• **Verify(P P, C,M, PE, PD)** → {T, F}: To verify PE and PD, this algorithm first runs Dec(P P, PD, C) and checks if the output is equal to declared input M. Then, this algorithm checks PE with correct coins b0, b1 derived in the decryption process. If both requirements are satisfied, this algorithm returns T; otherwise, it returns F.

• **DenSetup (1^λ)** → (P P, MSK, PK): This algorithm runs Setup (1^λ) and obtains P P. System public key PK is {g2g3,(g2g3) a , e(g3, g3) α, e(g2g3, g2g3) α} and system secret key MSK is {(g1g3) α, g1g2g3,(g1g2g3) α}.

• **DenKeyGen (MSK, S)** → (SK, FK): This algorithm runs KeyGen and obtains SK for S. Next, this algorithm picks t ′ ∈ ZN and generates FK as follows:

$$\begin{aligned} FK &= \{(g_1g_2g_3)^{\alpha + at'}, (g_1g_2g_3)^{t'}, \{H_1(x)^{t'}\}_{\forall x \in S}\} \\ &= \{K', L', \{K'_x\}_{\forall x \in S}\}. \end{aligned}$$

• **DenEnc (P P, PK, M, M′, A = (M, ρ))** → C ′: This algorithm prepares λi , ∀i ∈ {1, . . . , l} just as the Enc algorithm does. This algorithm sets up chameleon hash function CH (·, ·). The chameleon hash function is determined during encryption. Note that without the trapdoor, a chameleon hash is just a one-way hash function. That is, a sender can claim this is just a normal hash function without any trapdoor. Output deniable ciphertext C ′ will be:

$$C' = \{A'_0, A'_1, B', (C'_1, D'_1), \ldots, (C'_l, D'_l), CH, t_0, t_1, V\},$$

where,

$$A'_{b_0} = \mathcal{M} \cdot e(g_3, g_3)^{\alpha s}, A'_{1-b_0} = \mathcal{M}' \cdot e(g_2g_3, g_2g_3)^{\alpha s},$$
$$B' = (g_2g_3)^s,$$
$$C'_i = (g_2g_3)^{a\lambda_i} H_1(\rho(i))^{-r_i}, D'_i = (g_1g_3)^{r_i}, i = 1, \ldots, l,$$
$$V = CH(\mathcal{M}, t_{b_1}) = CH(\mathcal{M}', t_{1-b_1}).$$

Based on the property of the chameleon hash, the sender can easily find t_{b_1} and t_{1-b_1} satisfying the above requirements.

• **DenOpenEnc (P P, C′, M′)** → P ′ E: When the sender tries to fool the coercer with the pre-determined fake message, this algorithm returns two coins 1–b1, 1– b2 as its proof P ′ E.

• **DenOpenDec (P P, SK, FK, C′, M′)** → P ′ D: This algorithm directly returns FK as proof P ′ D.

III. Conclusion

In this work, we tend to propose a deniable CP-ABE theme to build an audit-free cloud storage service. The deniability feature makes coercion invalid, and the ABE property ensures secure cloud knowledge sharing with a fine-grained access control mechanism. Our proposed theme provides a possible way to fight against immoral interference with the right of privacy. we hope a lot of schemes may be created to shield cloud user privacy.

IV. REFERENCES

[1]. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H) IBE in the standard model. In Advances in Cryptology—EUROCRYPT 2010, volume 6110 of LNCS, pages 553- 572. Springer, 2010.

[2]. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Advances in Cryptology—CRYPTO 2010, volume 6223 of LNCS, pages 98-115. Springer, 2010.

[3]. Shweta Agrawal and Xavier Boyen. Identity-based encryption from lattices in the standard model. Manuscript, July 2009. http://www.cs.stanford.edu/~xb/ab09/.

[4]. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 99-108, New York, NY, USA, 1996. ACM.

[5]. Miklos Ajtai. Generating hard instances of the short basis problem. In Jir Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, ICALP, volume 1644 of Lecture Notes in Computer Science, pages 1-9. Springer, 1999.

[6]. Miklos Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In STOC, pages 284-293, 1997.

[7]. Joel Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In STACS, pages 75-86, 2009.

[8]. Rikke Bendlin and Ivan Damg˚ard. Threshold decryption and zero-knowledge proofs for latticebased cryptosystems. In TCC, pages 201-218, 2010.

[9]. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, pages 321-334, Washington, DC, USA, 2007. IEEE Computer Society. 15

[10]. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Advances in Cryptology (EUROCRYPT 2004), volume 3027 of LNCS, pages 223-238. Springer, 2004.

[11]. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. Siam Journal on Computing, 36:1301-1328, 2007.

[12]. Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In Proceedings of FOCS 2007, pages 647-657, 2007.

[13]. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Theory of Cryptography, pages 325-341, 2005.

[14]. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In TCC, 2011.

[15]. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphism encryption from ring-LWE and security for key dependent messages. To Appear in CRYPTO 2011, 2011.

[16]. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees or, how to delegate a lattice basis. In Proceedings of Eurocrypt, 2010, 2010. http://eprint.iacr.org/.

[17]. Ling Cheung and Calvin C. Newport. Provably secure ciphertext policy ABE. In ACM Conference on Computer and Communications Security, pages 456-465, 2007.

[18]. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Proceedings of the 8th IMA International Conference on Cryptography and Coding, pages 26-8, 2001.