International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 5, Issue 2, 2018, PP 7-16
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
DOI: http://dx.doi.org/10.20431/2349-4859.0502002
www.arcjournals.org

Security and Privacy of Identity Information in Cloud Computing

Shikha[1], Dr.Vijay Kumar Tiwari[2]

[1,2]*Department of computer science & Engineering, Centre for Advance Studies, AKTU Luck now, India.*

Corresponding Author:** ***Dr. Vijay Tiwari, *Department of computer science & Engineering, Centre for Advance Studies, AKTU Luck now, India.*

Abstract: *Cloud computing grants to storing and accessing data and program over the internet instead of yours computer's hard drive. Cloud computing allows the user to use services on a utility-like basis and support business processes. It offers a heterogeneity of "users" and resources due to also formation risks for data privacy and danger of multiple, collaborative threats.*

In cloud computing, entities or user's may have multiple accounts associated with a single or multiple service providers (SPs).Sharing sensitive identity information along with associated attributes of the same entity across services can lead to mapping of the identities to the entity, equable to privacy loss.

Identity management (IDM) is one of the core components in cloud privacy and security.

Available solutions are either use trusted third party (TTP) in identifying entities to Service provider's or independent of TTP(untrusted hosts.)

The approach is based on the use of predicates over encrypted data and multi-party computing for consort a use of a cloud service. active bundle scheme—which is a middleware agent that includes PII data, privacy policies, a virtual machine that enforces the policies, and has a set of protection mechanisms to protect itself. An active bundle interacts on behalf of a user to authenticate to cloud services using user's privacy policies.

Keywords: *computing predicates; active bundle; cloud computing; multi-party computing; identity management system; privacy; security .*

1. INTRODUCTION

1.1. Privacy in Cloud Computing

The continuing development and on growing popularity and maturation of cloud computing services is an Information stored locally on a computer can be stored in the cloud, including word processing documents, audio, photos, videos, records, financial information, spread- sheets, appointment calendars, presentations etc .

A cloud *service provider* (*SP*) is a third party that maintains information about, or on behalf of, another entity. A single breach can cause significant loss. Whenever some entity stores or processes information in the cloud, questions may arise for privacy or confidentiality [2].

The nature of cloud computing, there is little or no information available in a cloud to point outwhere data are stored, how secure they are, who has access to them, or if they are transferred to another host this may arises *Privacy concerns* in cloud computing . "The ability to control who can access that information and the ability of an entity to control what information it reveals about itself to the cloud (or to the cloud Service Provider)."[3].

The major problem regarding privacy in cloud is how to secure PII (*personally identifiable information*) from being used by unauthorized users, how to prevent attacks against privacy (such as identity theft) even when a cloud SP cannot be trusted, and how to maintain control over the disclosure of private information. Handing sensitive data to another company is a serious concern. Are data held somewhere in the cloud as secure as data protected in user-controlled computers and networks?

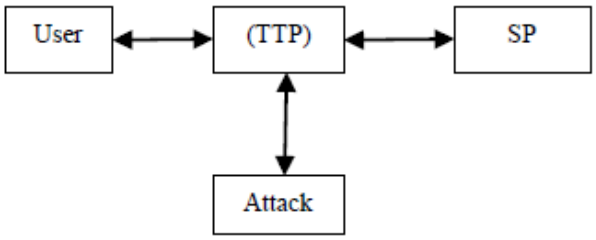
1.2. Identity Management in Cloud Computing

The users hold multiple accounts with different SPs or with a single SP so IDM in cloud is more complex than in traditional web-based systems which is centric access control, where each application keeps track of its collection of users and manages them. while in cloud-based architectures; sharing PIIs of the same entity across services can lead to mapping of PIIs to the entity.[7]

A user needs to authenticate himself or herself to it to use a cloud service. The user has to give away some private information, such as (name, home address, credit card number, phone number, driver's license number, date of birth, etc.) which uniquely identifies the user to SP the other party. the user's PII gives some assurance to SPs about the users. Identity, which helps SP to decide whether to permit access to its service or not. The purpose of an IDM system is to decide upon the disclosure of identity information in a secure manner because this is key for opening access to resources.

1.3. Existing Idm System

In decentralized authentication protocol a user has to remember one username and password.—an Open ID —and can log onto websites with this Open ID. That helps cloud users in managing their multiple digital identities with a greater control over sharing their PIIs.



- Use of a Trusted Third Party (TTP). for verifying or approving PII. The major is dispute are: (1.) TTP could be a cloud service, or SP; So, TTP may not be an independent trusted entity anymore (2) using a single TTP is a centralized approach, cause with its inherent danger that compromising TTP results that compromising all PIIs of its users as well.
- Prohibiting untrusted hosts. A client application holding PII must be executed on a trusted host to prevent malicious hosts from accessing PII .

Problems arise:

1) ***Authenticating without disclosing PII:*** When a user sends PII to authenticate for a service, the user may encrypt it. However, PII is decrypted before an SP uses it. As soon as PII is decrypted, it becomes intent to attacks.

2) ***Using services on untrusted hosts:*** The available IDM solutions require user to execute IDM from a trusted host. They do not recommend using IDM on untrusted hosts, such as public hosts. Since in cloud computing data may reside anywhere in the cloud (on any host),this issue needs to be addressed. E.g. User herself may be on a cloud Virtual Machine.

Note that goal in the paper is to assure that IDM does not use TTP for verifying credentials. This implies that IDM could use TTPs for other purposes, such as the use of a TTP by IDM for management of decryption keys.

2. Proposed Idm Approach for Protecting Pii (Personally Identifiable Information) in Cloud Computing

2.1. Use of Predicates with Encrypted Data and Multiparty Computing

2.1.1. Predicate Encryption

Alice use Gmail service which is provided by Google. However, she is concerned about her privacy, so does not wish Google to read her emails. For privacy concerns, Alice is to use public-key encryption to protect the secrecy of her emails. Alice generates a public key/private-key pair. Alice's public-key PK used by her friends to encrypt the emails before sending them to Alice. Now all emails will be stored in encrypted format at Google, and Alice is being able to protect her privacy.

If Alice wishes to explore for all email whose" (sender = Bob)and(date with in [2016,2017])". Unfortunately, Google can no longer explore her emails, stored in encrypted format, and without the secret key, the emails are indistinguishable from random numbers to Google. Either Alice can download all emails from Google, decrypt them using own private key and search them locally. But what if there are too many emails to download? Alternatively, Alice can give away her private-key to Google, but of course, that beats the purpose of encryption.

This problem can be solved using by a new type of encryption, called *predicate encryption*. Using predicate encryption, Alice can compute a capability (refer as Token) corresponding to her query, e.g."(sender =Bob) and (date within [2016, 2017])". Alice gives this capability to Google, Then Google can test the capability against Alice's encrypted emails. In this way, Google is able to become aware which emails match the query; and beyond this information, Google gain nothing more about the encrypted emails. Traditiona encryption versus Predicate encryption offers release partial information about the encrypted data in a controlled manner.

2.1.2. Public-Key predicate Encryption

Let PII or X = (x1, x2, . . . , xn) ∈ $\{0, 1\}^n$ denote a plaintext. to evaluate from the cipher text Boolean functions (also referred to as *predicates*) on X, that is f : $\{0, 1\}^n \rightarrow \{0, 1\}$. Functions that output multiple bits can be regarded as concatenation of boolean functions. Let F denote a family of boolean functions from $\{0, 1\}^n$ to {0, 1}. For example, F can be the set of all conjunctions on (x1, x2, . . . ,xn) ∈$\{0, 1\}^n$. A token allows one to evaluate from the cipher text a predicate f ∈ F.

A Public-Key Predicate Encryption (PK-PE) scheme is formation of the following (possibly randomized) algorithms.

Setup (1^λ,F)**:** The Setup algorithm takes as input a security parameter 1^λ the predicate family F being considered; and outputs a public key PK and a master secret key MSK.

Encrypt (PK,X): The Encrypt algorithm takes as input a public key PK, a plaintext X = (x1, x2, . . . , xℓ) ∈ $\{0, 1\}^\ell$ and outputs a cipher text CT.

Gen Token (PK,MSK, f): The Gen Token algorithm takes as input a public key PK, master secret key MSK, and a query predicate f ∈ F. It outputs a token TK_f for evaluating the predicate f from a cipher text.

Query(PK, CT, TK_f): The Query algorithm takes as input a public key PK, a token TK_f for the predicate f, and a cipher text CT. Suppose CT is an encryption of the plaintext X; the algorithm outputs f(X).

Alice uses a *Setup* algorithm to generate a public key *PK* and a secret key *MSK*. then after, Alice uses *PK* to encrypt (with algorithm *Encrypt*) her PII and gets cipher text *CT*. Then, she can store *CT* (the encrypted *PII or X*) on an untrusted host (e.g., in a cloud). She may also publish *PK*, so that it can be used to encrypt data that she can access. Alice has the function *f* representing a predicate that she wishes to evaluate for her encrypted PII. She uses the *Key Gen* algorithm, *PK*, *MSK* and f to output the token TK_f. Then, she gives TK_f to the host that evaluates the token (with *f* included in the token) for *CT* (the encrypted PII), and returns the result f(X) or *f(PII)* to Alice.

Key Gen uses the secret key *MSK* as input. Hence, Alice can use *KeyGen* to generate TK_f for *f*. Alice can give TK_f to an untrusted host while protecting PII. (Observe that if Alice gave *Key Gen* and *MSK* to the host, the scheme would not be secure.—it would not protect PII.

For cloud service use, we combine computing predicate over encrypted data with secure multiparty computing. The secret key *MSK* is split between *n* parties by Shamir's technique. Shamir [4] proposes threshold secret sharing. First, a secret data item D is divided into *n* shares *D1*,..., *Dn*. Then,

a threshold *k* is chosen so that: (a) to recover *D*, *k* or more of arbitrary *Di*'s are required; (b) using any *k-1* or fewer *Di*.'sleaves *D* completely undetermined.

Then, the algorithm *Key Gen* is provided to *n* parties, and computed by them collaboratively using their shares of the secret key, function *p* representing a predicate, *PK*, and *TK*.

This is done by protocol of Ben-Or, Goldwasser and Wigderson [5] for multi-party computing. According to this protocol, for multi-party computing of a function *f* using, secret input from all the

parties. The protocol involves *n* "regular" parties, which calculate only partial function outputs. one of the players is selected as the *dealer* and is provided the partial function outputs to find out the full results of function computation.

Let *f* be a *linear function* of degree *n* known to each of the *n* parties, and *t* be an arbitrary threshold value. Let *Pi* denote Party *i*, and *xi* denote the *secret* input of *Pi* for *f*. Dealer *DLR* will receive from the *n* parties the partial outputs of *f* calculated by the *n* parties using their respective secret inputs *x1* , *x2*, …., *xn*. each *Pi* computes a portion of function *f* using shares input that it has (its own) or received from *n*.–1 other parties.

In our algorithm, an owner *O* encrypts PII or x using algorithm *Encrypt* and *O.*'s public key *PK*. *Encrypt* outputs *CT.*—the encrypted PII. SP transforms his request for PII to a predicate represented by function *f*. Then, *SP* sends shares of to the *n* parties who hold the shares of *MSK*. The *n* parties execute together *KeyGen* using *PK*, *MSK*, and f, and return TK_f to SP. Next, SP calls the algorithm *Query* that takes as input *PK*, *CT*, TK_f and produces *f*(*PII*) which is the evaluation of the predicate. The owner *O* is allowed to use the service only when the predicate evaluates to "true".

2.2. Security

We describe a query security game between a challenger and an adversary. This game tokens reveal no unintended information about the plaintext. In this game, the adversary asks the challenger for a number of tokens. The adversary should not be able to deduce any unintended information from these tokens. The game proceeds as follows:

- Setup: The challenger runs the Setup algorithm, and gives the adversary the public key PK.
- Query 1: The adversary adaptively makes a polynomial number of queries. In each query, the adversary specifies a predicate f ∈ F, and asks the challenger for a token for that Predicate.

  The challenger computes the requested token by calling the Gen Token algorithm, and returns the token to the adversary.
- Challenge**:** The adversary outputs two strings $X^*_0, X^*_1 \in \{0, 1\}^\ell$ subject to the constraint that for any predicate f queried by the adversary in the Query1 stage, the following must be true:$f(X^*_0)=f(X^*_1)$ (1)
- Next, the challenger flips a random coin b, and encrypts X∗b. It returns the cipher text to the adversary.
- Query 2: Repeat the Query 1 stage. All predicates queried in this stage should satisfy the same condition as above.
- Guess: The adversary outputs a guess b′ of b.

2.2.1. Match revealing security

The version relaxed of security called *match revealing* security and the strict version of security called match *concealing* security.

In *match-concealing* security, the adversary does not learn any additional information about the plaintext whether the output of the predicate is true or not. So it is called "two-sided" security.

By contrast, *match-revealing* security can be thought of as "one-sided" security:

- When the predicate evaluates to true, the adversary does not learn any additional information about the plaintext encrypted;
- When the predicate evaluates to false, we no longer care about preserving the secrecy of the plaintext. the query predicate.

The formal definition of match-revealing clearly match-concealing security implies match-revealing security. However, we are also interested in match-revealing security, because in some cases, using the relaxed version of security can lead to more efficient and practical constructions. Meanwhile, in many practical applications, we no longer care about the secrecy of the encrypted entry if it matches security is almost the same as match-concealing security, with the exception that Equation (1) is now the following new equation: $f(X^*_0) = f(X^*_1) = 0$

2.3. Secret-Key Predicate Encryption

In secret-key encryption, encryption and decryption are both performed using the secret-key. While public-key encryption, anyone can encrypt using the public-key. In both schemes, only the secret-key owner can decrypt.

A Secret-Key Predicate Encryption (SKPE) system consists of the following (possibly randomized) algorithms.

Setup (1^{λ}): The Setup algorithm takes as input a security parameter 1^{λ}, and outputs a secret key MSK.

Encrypt (MSK, x): The Encrypt algorithm takes as input a secret key MSK, a plaintext $x \in \{0, 1\}^{\ell}$ and outputs a ciphertext CT.

GenToken (MSK, f): The GenToken algorithm takes as input a secret key MSK, and a query predicate $f: \{0, 1\}^{\ell} \rightarrow \{0, 1\}$. It outputs a token TK_f that allows one to evaluate f(x) over an encryption of x. the query predicate can be encoded with a bit string of length m.

Query (TK_f, CT): The Query algorithm takes as input a token TK_f for the predicate f, and a ciphertext CT which is an encryption of $x \in \{0, 1\}^{\ell}$, the algorithm outputs f(x).

2.3.1. Security for Secret-Key Predicate Encryption: Hiding both the Plaintext and the Query

Public-key predicate encryption schemes guarantee the secrecy of the ciphertext; however, they do not guarantee the secrecy of the tokens. In fact, for public-key predicate encryption, it is inherently impossible to achieve ciphertext secrecy and token secrecy simultaneously. This is due to the fact that anyone is able to encrypt using the public-key.

In the Gmail example, if Google would like to know whether a token corresponds to the query "TITLE =cryptography", Google can simply encrypt an email whose "TITLE = cryptography" using the public-key, and test the token against the resulting ciphertext.

In secret-key predicate encryption, it is possible to guarantee the secrecy of both the plaintext (encoded in a ciphertext) and that of the query (encoded in a token). This provides even stronger privacy guarantees in practice. We now formally define the security for secret-key predicate encryption aims to guarantee the secrecy of the plaintext, as well as the query.

To explain the intuition behind our security definition, consider a privacy-preserving remote storage application, where Alice stores her encrypted documents on a remote server, and later issues tokens to the server to search for matching documents. Our goal is to leak as little information to the storage server as possible. Under our model, Alice makes a query by submitting a token to the server, and the server learns exactly which of her encrypted documents match the query, and returns the matching documents to Alice. Therefore, in this framework, the server inevitably learns Alice's access pattern, which documents Alice retrieves with each query.

We would like to define security in the strongest sense possible: informally, the storage server should learn only Alice's access pattern, and nothing more.This implies that the server learns nothing about Alice's encrypted documents, or what queries she is making.

To capture the notion that the server learns only Alice's access pattern, the access pattern is the outcomes of q predicates on n plaintexts.

X = (x1, x2. . . xn) denote an ordered list of n plain texts, where $xi \in \{0, 1\}^{\ell}$ for $1 \leq i \leq n$. Let F = (f1, f2. . . fq) denote an ordered list of q query predicates, where $fi \in \{0, 1\}$ m for $1 \leq i \leq q$. The access pattern on X and F is an q × n matrix:

$$\text{ACCESS PATTERN}(X, F) = \begin{vmatrix} f1(x1), f1(x2), \ldots, f1(xn) \\ f2(x1), f2(x2), \ldots, f2(xn) \\ fq(x1), fq(x2), \ldots, fq(xn) \end{vmatrix}$$

We now proceed to define the security for SKPE. Let $X = (x_1, x_2. . . x_n)$, $X' = (x'_1, x'_2, \ldots, x'_n)$ denote two ordered lists of plaintexts. Let $F = (f_1, f_2. . . f_q)$, $F' = (f'_1, f'_2, \ldots, f'_q)$ denote two ordered lists of queries predicates Suppose the two worlds have the same access pattern, i.e.,

ACCESSPATTERN(X, F) =ACCESSPATTERN (X′, F′).

Informally, the server should not be able to distinguish between the two worlds. The security definition presented below describes a game between a challenger and an adversary, and is intended to capture this notion of in distinguishability between two these worlds. Moreover, the definition considers an adaptive adversary: an adversary who can choose what ciphertext/token queries to make depending on the previous interactions with the challenger.

(SKPE full security) We say that an SKPE scheme is fully secure, if no polynomial time adversaries has more than negligible advantage in the following game.

Setup: The challenger runs the Setup algorithm, and retains the secret key MSK to itself. In addition, it flips a random coin b, and keeps the bit b to itself as well. Define four ordered lists, X_0, F_0, X_1, F_1, where (X_0, F_0) will record plaintexts and predicates queried by the adversary in World 0, and (X_1, F_1) will record plaintexts and predicates queried by the adversary in World 1. Initially, all four lists are empty.

Query: The adversary adaptively makes the following types of queries. The adversary can make up to a polynomial number of these queries.

Ciphertext query: The adversary specifies two plaintexts x_0, $x_1 \in \{0, 1\}^{\ell}$ to the challenger. The challenger encrypts x_b and returns the ciphertext to the adversary. Append x_0 to the list X_0, and x_1 to the list X_1.

Token query: The adversary specifies two predicates f_0, $f_1 \in \{0, 1\}_m$ to the challenger.

The challenger computes a token for the predicate f_b, and gives the resulting token to the adversary. Append f_0 to the list F_0, and f_1 to the list F_1.

All queries made in this stage should be identical by access pattern.At the end of the game, all queries made should satisfy the following condition: ACCESSPATTERN(X_0, F_0) = ACCESSPATTERN(X_1, F_1).

2.4. Applications of Predicate Encryption

- *Network audit logs***.** Large-scale efforts has made by intrusion detection community in network to collect network audit logs from different sites. An Internet Service Provider (ISP) or network gateway can submit network traces to an audit log repository. The network traces contains the presence of privacy sensitive information, the gateway will allow only authorized parties to search their audit logs. We consider the following four types of entities: a gateway, an authority, an untrusted repository, and an auditor.

Predicate encryption allows the gateway to submit encrypted audit logs to the untrusted repository. Normally, no one is able to decrypt these audit logs. Whenever malicious behavior is suspected, an auditor may ask the authority for a search capability and the auditor can decrypt entries satisfying certain attack characteristics, e.g., network flows whose destination address and port number fall within a certain range. However, the privacy of all other flows should still be preserved. we can have multiple parties to jointly act as the authority instead of central point of trust. Only when a sufficient number of the parities collaborate, can they generate a valid search capability. Securely splitting the authority into multiple parties can be achieved through secure multi-party computation techniques.

- *Financial audit logs* Sensitive information about financial transactions contained by the Financial audit logs .Predicate encryption allows financial institutions to release audit logs in encrypted format. When ever needed, an authorized auditor, With the decryption key, the auditor can decrypt certain transactions that may be suspected of fraudulent activities. However, the privacy of all other transactions is preserved.

- *Public health monitoring.* Consider a health monitoring program. When Alice moves about in her daily life, a PDA or smart-phone she carries automatically deposits her encrypted location at a storage server. Assume that each pulverize is of the form ((x, y, t), ct), where t represents time, (x, y) represents the location, and ct is Alice's contact information. During an outbreak of an epidemic, Alice wishes to be alerted if she was present at a site borne with the disease during an incubation period, i.e., if (x, y, t) falls within a certain range. However, she is also concerned with privacy, and she does not wish to leak her trajectory if she has not been to a site borne with the disease.

- *Sharing of medical records.* Medical research institutes would like to obtain patients' medical records for their research. However, these medical records are usually privacy sensitive, and it is necessary to enforce access control, such that a cardiologist is allowed to access medical records related to heart diseases, but not records on brain diseases. Using predicate encryption, we can easily enforce such fine-grained access control policies by granting the cardiologist a capability that allows her to decrypt precisely the medical records she needs. Furthermore, if the neurologist would like her assistant to check all cases that happen within the year 2017, she can perform a delegation operation, and generate a sub-capability that allows the decryption of all records on heart diseases and within the year 2017.
- *Stock trading through an untrusted broker.* An investor uses a broker to trade stocks. The investor does not fully trust the broker, and wishes to reveal as little information to the broker as possible. For example, the investor can place an order that says, "buy x amount of stock y if the price falls below p today". The broker should not be able to decrypt this order until the current price satisfies the conditions specified by the order. This problem can be addressed through predicate encryption. A party trusted by the investor (e.g., the stock exchange) issues a new capability to the broker as the stock price changes. The broker can now try to use the capability to decrypt the investor's order. If the current price meets the conditions specified by the order, the decryption is successful, and the order gets executed. If the order is never executed, the broker learns nothing about the contents of the order, except the fact that the conditions specified by the order were never met.
- *Untrusted remote storage***.** Individual users may wish to store emails and files on a remote server, but because the storage server is untrusted, the content must be encrypted before it is stored at the remote server. Emails and files can be classified with multiple attributes. Users may wish to perform certain types of queries and retrieve only data that satisfy the queries.
- *Using biometrics in anonymous IBE.* Biometric-based anonymous Identity-Based Encryption (AIBE) also used Predicate encryption .In this application, a person's biometric features such as finger-prints, blood-type, year of birth, eye color, etc., are encoded as a point X in a multidimensional lattice. Personal data is encrypted using the owner's biometric features as the identity, and the encryption protects both the secrecy of the personal data and the owner's biometric identity. For multi-dimensional queries allows a user with the private key for identity X, and to decrypt an entry encrypted under X.

2.5. Protecting Sensitive Data Using Active Bundles

Protecting privacy of sensitive data throughout their lifecycle using the active bundles scheme. an active bundle includes sensitive data, metadata (including privacy policies) and a virtual machine(VM). Privacy policies include rules for engine and four protection mechanisms: integrity self-check, evaporation (to self-destroy endangered portions of data selectively), apoptosis (to self-destroy all data and metadata), and decoy (to mislead suspected attackers with false but harmless information such as "I don't know" or "No information available").

The main reason for including VMs in active bundles is to prevent hosts that receive these bundles from accessing sensitive data include in the bundles without enforcing their privacy policies.

Using VMs in this way poses two main challenges: (i) how to assure that a visited host executes VM code faithfully and correctly; and (ii) how to implement a VM that protects confidentiality of sensitive data. We believe that using a Trusted Platform Module (TPM) is currently the most practical solution to address the first challenge. We assume that an operating system (OS) certified by a TPM executes correctly a VM code.

The second challenge, implementing a VM able to protect active bundles.

2.5.1. Structure of Active Bundles

An active bundle includes the following components –sensitive data, meta data, virtual machine.[8]

1) **Sensitive data:** It is a digital content that needs to be protected from privacy violations, data leaks, unauthorized dissemination, etc. The digital content can include documents, pieces of code, images, audio or video files, ex- personal identity information (PII).

2) **Metadata:** It describes the active bundle and its privacy policies. The meta data includes (but is not limited to) the following components:

a. **Provenance metadata**—Including an identifiers of its creator and owner, the creation date, identifier of the active bundle, identifiers of all visited hosts, as well as identifiers of all guardians with their access timestamps and possibly, their update timestamps if they performed any updates of sensitive data. The history of visits and updates is kept private (it could be used only for forensic investigations by authorities that obtain a court order).

b. **Integrity check metadata:** It includes the algorithm for checking the integrity of sensitive data as well as a hash calculated by the owner.

c. **Privacy policy metadata:** It includes access control policies for sensitive data of the bundle. For each subset of sensitive data, required minimal host's trust level.

d. **Dissemination control metadata:** It includes dissemination policies specifying who can disseminate the active bundle, and under what conditions (e.g., the active bundle could be disseminated at a specified time, or could be disseminated to a specific group of addressees—such as employees, suppliers, or a board of directors.)

e. **Life duration:** It specifies a date and time where the sensitive data must disappear.

f. **Security metadata:** It includes: (i) security server id, specifying the security server used in the process of bundle encryption and decryption low); (ii) encryption algorithm used by the VM of the bundle; (iii) trust server id, used to validate the trust level and the role of a host at which the bundle arrived; and (iv) trust level threshold, specifying the minimal trust level required by the VM to enable the active bundle.

g. **Other application-dependent and context- dependent components:** It includes information related to semantics (the application, the context, etc.) of sensitive data of the active bundle.

3) **Virtual machine (VM):** It manages and controls the program enclosed in an active bundle. The role of a VM is to guarantee the appropriate access control to sensitive data of the bundle (for example, disclosing to a guardian only the portion of sensitive data that the guardian is entitled to access). The VM also performs integrity checks for the bundle. Meta data and VM of any active bundle must be constructed in such a way that an attacker who removes the VM from an active bundle must not be able to access its sensitive data. For instance, by using metadata an attacker must not be able to extract, the address of the security server, that stores the decryption keys, and then obtain from the server any decryption key without using the VM of the active bundle. This can be achieved by a security server evaluates the trust level of the visited host and provides the decryption keys only to hosts that have a sufficient trust level for accessing the sensitive data included in the active bundle. A host that has a sufficient trust level is assumed to use the VM of an active bundle in order to access the bundle's sensitive data. If an active bundle is received by a destination host H with the trust level t, the VM of the bundle must ensure that H has access to only a portion of data for which t is a sufficient trust level. If only a portion of data is not to be seen by H, evaporation of the data not to be shown to H is performed by the VM. If no bundle data can be seen by H, apoptosis is triggered by the VM.

2.5.2. Operation of Active Bundles

The three basic operations performed by an active bundle are:

1) ***Evaporation:*** An active bundle asks for the host's trust level after arriving at a host. If the hosts' trust level is sufficient, then it is to allow access to all or portion of the bundle's data, Here the bundle's privacy policy is applied which is stored in its metadata. All data that the host is not allowed to access might be "evaporated" according as specified in the privacy policy. Evaporation of data diminishes the value of data.

2) ***Apoptosis:*** An active bundle may realize that its security or privacy is about to be compromised, e.g., it may discover that its self integrity check fails, or the trust level of its guardian is too low. In response, the bundle may choose to apoptosis i.e., perform a clean self-destruction. a complete self-destruction that leaves no trace usable for an attacker. Rules for activate an apoptosis are included in the privacy policies of the bundle's metadata. An active bundle might perform other operations.

3) ***Integrity self-check:*** *At* a new guardian, the bundle checks its integrity using the algorithm. algorithm specified in its metadata. It calculates the hash value of the active bundle. Then Compares the computed value to the value recorded within the bundle Meta data. If the values differ, the bundle performs apoptosis.

An active bundle is sent from a source host to a destination host. When bundle is arriving at a "foreign." host, an active bundle ascertains the host's trust level through a TTP. Using its disclosure policy, it decides whether the host may be eligible to access all or part of bundle's data, thus becoming a "guardian" for the data, and which portion of sensitive data can be revealed to it. The remaining data is not to be revealed. It might be *evaporated* as specified in the access control policies and protecting the data. We consider a different metrics for adaptive control of the degree of evaporation and trust-based metrics.

An active bundle may realize that its security is about to be compromised. E.g if its self-integrity check fails, or if the trust level of its host is too low. Then in response, bundle may choose to apoptosis, perform atomically a clean self-destruction. One that is complete and leaves no traces usable for an attacker.

3. ADVANTAGES OF THE PROPOSED APPROACH

The presented approach is one of the alternatives to using TTPs and it reduces the risks associated with the use of TTPs. The main advantages of the proposed approach are:

1) *No need for TTPs*. Since data exchange between its host and a bundle. its host is local to the host, it protects PII from man-in-the-middle, side channel and collaborative attacks.

2) *Authentication without disclosing unencrypted data*. This prevents unnecessary data disclosures.

3) *Protection of identity data from untrusted hosts*. If data reach an unintended destination or there are tampered with it, they self-destroy by apoptosis or by evaporation to prevent falling into wrong hands.

4. CONCLUSION

Important concerns for both the public and private sectors with the immense growth in the popularity of cloud computing is Privacy and security. Users having multiple identities in multiple service providers (SPs), multiple credentials, security repositories and multiple access permissions for different services provided by different SPs. There is a strong requirement for an efficient and effective Privacy preserving system .which is independent of TTPs. System should be able to identify users within enterprises and across the Web, and protects users PII. IDM is one of the core components in cloud privacy and security.

The proposed approach authentication without disclosing unencrypted data provides privacy and security provides by calculating the trust level, data self-destroy by apoptosis or by evaporation to prevent falling into wrong hands, for building IDM systems without using TTPs, using the active bundles scheme, computing predicate over encrypted data and multiparty computing. The solution enable the use the IDM application on untrusted hosts.

REFERENCES

[1] Protection of Identity Information in Cloud Computing without Trusted Third Party,Rohit Ranchal, Bharat Bhargava, Lotfi Ben Othmane, Leszek Lilien, Anya Kim, Myong Kang, Mark Linderman,29th IEEE International Symposium on Reliable Distributed System, 2010

[2] R. Gellman, ."Privacy in the Clouds: Risks to Privacy andConfidentiality from Cloud Computing,." World PrivacyForum, Feb.2009.Onlineat:http://www.worldprivacyforum.org/pdf/WPF_Cloud_Privacy_Report.pdf

[3] P. Angin, B. Bhargava, R. Ranchal, N. Singh, L. BenOthmane, L. Lilien, and M. Linderman, ."A User-Centric Approach for Privacy and Identity Management in CloudComputing,." Proc. 29th IEEE Intl. Symp. on ReliableDistributed Systems (SRDS), New Delhi , India, Nov. 2010.

[4] E. Shi, ."Evaluating Predicates over Encrypted Data,." Ph.D.Thesis. Carnegie Mellon University, Pittsburgh, PA. Oct.2008.

[5] M. Ben-Or, S. Goldwasser, and A.Wigderson. "Completeness theorems for non-cryptographic fault-tolerant distributed computation,." Proc. Twentieth Annual ACMSymposium on Theory of Computing. Chicago, IL. May 1988,pp.1-10.

[6] Active Bundles for Protecting Confidentiality ofSensitive Data Throughout Their Lifecycle.

[7] A. Gopalakrishnan, ."Cloud Computing Identity Management,." SETLabs Briefings, vol. 7, 2009. Online at:http://www.infosys.com/research.

[8] L. Ben Othmane, and L. Lilien, ."Protecting Privacy in Sensitive Data Dissemination with Active Bundles" Proc 7th Annual Conference on Privacy, Security & Trust (PST 2009), Saint John, New Brunswick, Canada, Aug 2009.

AUTHORS' BIOGRAPHY

Shikha, is perusing M.Tech 1st year in Computer Science & Engineering with **specialization in cyber security** at Centre for Advanced Studies under Dr. APJ Abdul Kalam Technical University (AKTU) Luck now UP, India.

She completed her B.Tech in Computer science & Engineering from IIMT Engineering College Meerut under Gautam Buddh Technical University.

Dr Vijay Tiwari, received his BTech (Electronics and Telecommunication) from the prestigious Military College of Telecommunication Engineering and Master of Engineering from Indian Institute of Science, Bangalore in 2002 and finally a PhD in 2012. The author has research interest in Computer Networks, Cyber Security, Wireless and Satellite networks. He has handled many technical projects since inception and is an expert in project management. The Author has published many research papers in reputed International Journals adequately indexed. He also participated in many reputed Scopus indexed international conferences. He is a member of Computer Society of India and has been awarded an Outstanding Reviewer certificate 2016, by Elsevier Journals for his contribution in Journal of Computer and Electrical Engineering. Author is an Editorial Member of International Journal of Research Studies in Electronics and Engineering, and American Research Journal for Electrical Engineering. He has contributed immensely to their quality and contents. Author is also a reviewer of Springer- Design Automation for Embedded Systems and is associated with other technical Journals of international repute.

Citation: *Shikha, V. K. Tiwari, (2018)" Security and Privacy of Identity Information in Cloud Computing",* ***International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 5(2), pp.7-16. DOI: http://dx.doi. org/10.20431/2349-4859.0502002***