Formal analysis of a private access control protocol to a cloud storage

Mouhebeddine Berrima, Pascal Lafourcade, Matthieu Giraud, Narjes Ben Rajeb

Formal Analysis of a Private Access Control Protocol to a Cloud Storage

Mouhebeddine Berrima

LIP2, Université de Monastir, Monastir, Tunisia
E-mail: berrima.mouheb@gmail.com

Pascal Lafourcade

LIMOS, Université Clermont Auvergne, Aubière, France
E-mail: pascal.lafourcade@uca.fr

Matthieu Giraud

LIMOS, Université Clermont Auvergne, Aubière, France
E-mail: Matthieu.Giraud@uca.fr

Narjes Ben Rajeb

LIP2, Université de Tunis El-Manar, Tunis, Tunisia
E-mail: narjes.benrajeb@gmail.com

Abstract: Cloud storage provides an attractive solution for many organizations and enterprises due to its features such as scalability, availability and reduced costs. However, storing data in the cloud is challenging if we want to ensure data security and user privacy. To address these security issues cryptographic protocols are usually used. Such protocols rely on cryptographic primitives which have to guarantee some security properties such that data and user privacy or authentication. *Attribute-Based Signature* (ABS) and *Attribute-Based Encryption* (ABE) are very adapted for storing data on an untrusted remote entity. In this work, we enhance the security of cloud storage systems through a formal analysis of a cloud storage protocol based on ABS and ABE schemes. We clarify several ambiguities in the design of this protocol and model the protocol and its security properties with ProVerif an automatic tool for the verification of cryptographic protocols. We discover an unknown attack against user privacy in the Ruj et al. protocol. We propose a correction, and automatically prove the security of the corrected protocol with ProVerif.

Keywords: Cloud storage; formal methods; attribute based signature; attribute based encryption; data and user privacy.

Biographical notes: Mouhebeddine Berrima received his PhD in Computer Science at the University of El-Manar, Tunisa. He is an associate professor at the Department of Computer Science, University of Monastir, Tunisa. His research interests include formal verification of security protocols, algebraic specifications of programs and data structures and security in cloud computing.

Pascal Lafourcade obtained his Phd laboratory LSV at ENS Cachan on verification of cryptographic protocols in presence of algebraic properties. Then he spent one year at the ETH Zurich in David Basin group, working on WSN. After he was during seven years Maitre de conference at Verimag developing automatic technique for verifying cryptographic primitives, and analyzing security protocols. Between 2013 and 2016 he held an industrial chair on Digital Trust at Clermont Ferrand at Laboratory LIMOS. Now he is an associate professor at the University Clermont Auvergne (Clermont-Ferrand, France). He is a member of the Networks and Protocols Team of the LIMOS Laboratory.

Matthieu Giraud received his Master in Cryptography in 2016 from Université de Bordeaux, France. Then he spent six months in the cryptography department of Thales Security and Communications (Gennevilliers, France) to work on Searchable Encryption. Since he is a PhD student under the supervision of Pascal Lafourcade at Laboratory LIMOS of University Clermont Auvergne. He studies the security of data storage in the Cloud.

Narjes Ben Rajeb holds an Engineering degree in Computer Science from the Faculty of Sciences of Tunis, a PHD degree in Computer Science from Henri Poincaré University at Nancy, and an Habilitation for conducting research in Computer Science from the Faculty of Sciences of Tunis. Currently, she is a full Professor at the National Institute of Applied Sciences and Technology at Tunis. Her research interests include formal methods for specification and verification and security protocols.

1 Introduction

Cloud storage refers to data storage services hosted over Internet. The cloud users store data online, so other authorized users can access them from any location via the Internet. Due to the features of cloud storage such as large storage capacity, high availability, scalability and reduced costs, many applications manipulating sensitive data are using a cloud storage. However, sharing sensitive data on a third party through a public network brings some security challenges. In particular, there are concerned with the privacy of users and data. Protecting privacy in clouds is more difficult than in traditional environments, because sensitive data may be disseminated and stored over many external locations managed by external service providers (Wang et al., 2010), and both cloud and their user can be malicious (Mulazzani et al., 2011; Zhang et al., 2012). User privacy is required in many applications where users have a right of privacy when storing their sensitive information like financial or health data (Tang et al., 2012). There are two important privacy requirements when a user stores data on the cloud: *anonymity* and *unlinkability*. The ISO/IEC standard (governmental organisations, 2009) define anonymity as the property ensuring that a user may use a service or a resource without disclosing his (or her) identity. However, preserving the anonymity property may still release information about a user by allowing an adversary to track several uses of a resource by the same user. Such information might allow an adversary to deduce or at least restrict the possible identities of a user. Therefore, the unlinkability property is required, ensuring that the different uses of a service or a resource for the same user should not be linked by an adversary. On the other hand, the Cloud Service Provider (CSP) must authenticate the user to be sure that he has the right to store data on the cloud, moreover this authentication must be done without reveal any information about his identity. As a concrete example, consider an employer in an institution learning about a corruption while going through some records. He decides to send the records in question to the authority for combating corruption using a cloud storage service while retaining her anonymity, but with a proof that she indeed has access to these records. The challenge of privacy-preserving authentication of users is one of the main concern in cloud security. There are few cryptographic schemes which can be used in these situations like anonymous authentication (Cramer et al., 1994), group signatures (Chaum and Van Heyst, 1991), ring signatures (Rivest et al., 2001), mesh signatures (Boyen, 2007) and Attribute-Based Signature (ABS) (Maji et al., 2008). In a cloud environment, group signature can not be used because it assumes the preexistence of a group which is difficult in the cloud. Ring signature is not feasible not only because of a large number of users, but also the signer cannot know who these people are. Also the mesh signature can not be used since does not provide a way to convince that a message was endorsed by a single user, and so, can be endorsed by many users colluding together. ABS is the most convenient cryptographic scheme for privacy-preserving authentication in the cloud. In ABS, a signer possessing a set of attributes can sign a message with a claim predicate that is satisfied by his attributes. The verifier can only check if the message is signed by an authorized user without knowing any information about its identity.

Data Privacy has been also gained research interest because only authorized users have access to sensitive data on the cloud. Data must be protected when transmitted to CSP and during the storage. The protection is against the unauthorized users as well as the CSP since the cloud is often assumed to be honest and curious, which means that the cloud can be interested by the user's information, but can not modify it (Li et al., 2010; Yu et al., 2010). To ensure data privacy, several works propose the storage of data in encrypted form. Thus, if the storage is compromised, then the leaked information should be protected or at least limited. Identity-based cryptography is not feasible in this situation because the inability of users to share their encrypted data at a fine-grained level. *Attribute-Based Encryption* (ABE) (Sahai and Waters, 2005) solves the problem of fine-grained access control. There are two complimentary forms of ABE defined by (Goyal et al., 2006): Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and Key-Policy Attribute-Based Encryption (KP-ABE). In a CP-ABE system, the users have given a set of attributes and the data are encrypted under an access policy described as a Boolean formula. Only users having the attributes satisfying the access policy can decrypt the ciphertext. In a KP-ABE system, the situation is reversed: users are associated with access policies and ciphertexts are encrypted with sets of attributes.

Many approaches and techniques have been proposed to deal with security and privacy in cloud storage (Huang et al., 2017; Hosseinzadeh et al., 2017; Grover and Kaur, 2016; Giraud et al., 2017). In addition, some works have been interested in security for mobile cloud computing and Internet of Things (IoT), e.g (Stergiou et al., 2016; Memos et al., 2017; Zkik et al., 2017; Ibtihal et al., 2017). Taking advantages of ABS and ABE has emerged as a widely accepted approach by the cloud security community (Dahshan and Elkassass, 2014; Wang et al., 2014; Belguith et al., 2016; Alsmirat et al., 2017; Shakunthala et al., 2017). The ABS is used to ensure the authentication while hiding anonymity, and the ABE allows a fine-grained access control to data. The cloud storage protocol proposed by Ruj et al. (Ruj et al., 2012) is among the pioneering works to use ABS and ABE. The protocol uses the SSH protocol to secure all the communication between the users and the cloud, and it supports reading and writing data stored in the cloud. However, the security protocols are error-prone and difficult to be evaluated using simulation techniques or conventional test. This has encouraged

the researchers to formally analyze security protocols (Kremer and Ryan, 2005; Cremers et al., 2009; Delaune et al., 2010; Chen et al., 2013; Bansal et al., 2013; Puys et al., 2016). To the best of our knowledge, there is no formal modeling yet of ABE or ABS schemes using symbolic formalism as applied π-calculus.

Contributions: In this work we enhance the security of cloud storage services based on the attribute-based cryptography, through a formal analysis of Ruj et al. (Ruj et al., 2012) protocol. We consider this protocol, which we call *RSN'12* protocol, because it uses both decentralized ABS and ABE schemes, which are more suitable for a large-scale environment, and it is widely accepted by the cloud security community.

- We model it in the applied π-calculus (Abadi and Fournet, 2001), which allows us to model various cryptographic primitives with some equational theories. We use ProVerif tool (Blanchet et al., 2001) which is one of the most used and efficient tools to analyze cryptographic protocols (Puys and Lafourcade, 2015; Cremers et al., 2009). For sake of simplicity, we consider one attribute in our symbolic modeling of the ABE and ABS schemes, which can support more than one policy for signature and encryption.
- We formalize and verify the fundamental security properties of the protocol. In writing mode, we verify the writer authentication and writer privacy which is expressed by the anonymity of writer's identity and unlinkability, that is a user who stores data on the cloud. While in reading mode, we check the required property that is data privacy. We show that the unlinkability of a writer is not satisfied against an attack in which the adversary delays the messages of some writers. Then, we propose a fix, which prevents this attack.
- We also discuss some ambiguous aspects of RSN'12 protocol concerning the use of SSH protocol to secure connections, and the use of timestamps to prevent replay attacks.

A preliminary version of this paper appeared in (Berrima et al., 2017).

Outline: In Section 2, we give a description of SSH protocol, ABS and ABE schemes and RSN'12 protocol. In Section 3, we briefly introduce applied π-calculus and ProVerif tool. We model RSN'12 protocol in Section 4 and analyze the security properties in Section 5. Finally, we conclude in Section 6.

2 RSN'12 Protocol Description

Ruj et al. (Ruj et al., 2012) propose a protocol for reading and writing data stored in the cloud. The protocol is based on the decentralized approach of CP-ABE (Lewko and Waters, 2011) and ABS (Maji et al., 2008), where many authorities distribute secret keys associated to attribute. Using ABS the cloud verifies the authenticity of a user without knowing his identity before storing data. Using ABE only valid users are able to decrypt the stored data. The protocol makes the following assumptions:

- The CSP is honest-but-curious, i.e. the CSP is interested to view the user's information. So it tries to derive some information from the messages he learned during the execution of the protocol, but cannot modify the user's content.
- Users can have either read or write or both access to a file stored in the CSP.
- All the communication between participants are secured by SSH (Secure Shell) protocol.

Before presenting the protocol, we give some cryptographic background.

2.1 Cryptographic Background

The security of Ruj et al. protocol is based on the SSH protocol and two cryptographic schemes which are: *Attribute-Based Encryption* (ABE) and *Attribute-Based Signature* (ABS).

SSH protocol:. SSH (Secure Shell) is a widely deployed secure network protocol. Currently, SSH Version 2 is the recent specification for SSH. It is composed of three sub-protocols:

- *Transport Protocol*: It establishes session keys and ensures authentication of the server, confidentiality, and integrity of the communication. It prevents also replay attack using monotonically increasing sequence numbers.
- *Authentication Protocol*: It authenticates the user who is about to log in to the server.
- *Connection protocol*: It establishes different communication channels within an SSH session.

In SSH authentication sub-protocol there are three methods to authenticate the client (Ylonen and Lonvick, 2006). We consider "public key" method in which a user sends a signature created with his private key. The server must check that the key is a valid authenticator for the user and that the signature is valid.

Multi-authority ABS system:. ABS is a cryptographic scheme that allows a party, who possesses a set of attributes from an authority, to sign a message with a predicate, called *claim policy*, that is satisfied by his attributes. The signature reveals no more than the fact that a single user, whose his attributes satisfy the predicate, has signed the message. Moreover, the signature hides the attributes used to satisfy

the predicate. Then, ABS is used in applications requiring anonymous authentication. In a decentralized environment, there are multiple authorities for distributing attributes to users. These attribute authorities may not trust each other, nor even be aware of each other. However, a separate trusty authority is required to coordinate attribute authorities and set up the various public parameters of the ABS. The security properties of an ABS scheme are:

- *Perfect privacy*: The signature does not leak which attributes were used to generate it, nor any other user's identifying information.
- *Collusion-resistance*: Different parties cannot pool together their attributes to sign a message with a claim predicate which none of them satisfy alone.
- *Unforgeability*: Ensures that a signature was endorsed by one party who satisfies the condition described in the claim predicate, i.e. a party that does not possess the expected attributes, cannot generate a valid signature.

We give the formal definition of ABS. Let $\mathbb{A}$ be the universe of possible attributes. A claim predicate over $\mathbb{A}$ is a monotone Boolean function, whose input attributes are associated with attributes of $\mathbb{A}$.

Multi-Authority ABS (Maji et al., 2008): A multi-authority ABS scheme consists of the following algorithms:

- TSetup: Runs by the signature authority to produce a trusty public key TPK and trusty secret key TSK. The authority publishes TPK and stores TSK.
- TRegister: When a user with identity *uid* registers with the signature authority, the authority runs $TRegister(TSK, uid)$ which outputs a public user-token τ. The authority gives τ to the user.
- ASetup: An attribute authority who wishes to issue attributes runs $ASetup(TPK)$ which outputs an attribute-authority public key APK and an attribute-authority secret key ASK. The attribute authority publishes APK and stores ASK.
- AttrGen: When an attribute authority needs to issue an attribute $u \in \mathbb{A}$ to a user *uid*, first it obtains (from the user) her user-token τ, and runs the token verification algorithm $TokenVerify(TPK, uid, \tau)$. If the token is verified, then it runs $AttrGen(ASK, \tau, u)$ which outputs an attribute key K_u. The attribute authority gives K_u to the user.
- Sign: A user signs a message m with a claim-predicate Υ, only if there is a set of attributes $\mathcal{A}$ such that $\Upsilon(\mathcal{A}) = 1$, the user has obtained a set of keys $\{K_u \mid u \in \mathcal{A}\}$ from the attribute authorities. Then the signature σ can be generated using

$$Sign(TPK, \{APK_{auth(u)} \mid u \in \mathbb{A}_\Upsilon\}, \tau, \{K_u \mid u \in \mathcal{A}\}, m, \Upsilon).$$

  Here $auth(u)$ stands for the authority who owns the attribute u, and $\mathbb{A}_\Upsilon$ is the set of attributes appearing in Υ. (m, Υ, σ) can be given out for verification.
- Verify: To verify a signature σ on a message m with a claim-predicate Υ, a user runs

$$Verify(TPK, \{APK_{auth(u)} \mid u \in \mathbb{A}_\Upsilon\}, m, \Upsilon, \sigma)$$

  which outputs a Boolean value, *accept* or $reject$.

Multi-authority ABE system:. CP-ABE encryption allows encrypting data under an *access policy* expressed as a Boolean formula over a set of attributes, so that the decryption succeeds only for the users possessing the attributes which satisfy the access policy. Decryption is performed by using secret attribute keys, associated with the attributes of the user, and issued by an attribute authority. In the Multi-Authority CP-ABE system (Lewko and Waters, 2011), any party can act as an authority by creating a public key and issuing private keys to different users that reflect their attributes. There is no requirement for any global coordination or central authority other than the creation of an initial set of common public parameters. To ensure the property of collusion resistance, the authors use the concept of global identifiers to "link" secret attribute keys together that were issued to the same user by different authorities.

Multi-Authority CP-ABE (Lewko and Waters, 2011): A multi-authority Ciphertext-Policy Attribute-Based Encryption system is formed of the following five algorithms:

- GlobalSetup(λ) $\rightarrow$ GP: It takes in the security parameter λ and outputs global parameters GP.
- AuthoritySetup(GP) $\rightarrow$ (SK,PK): Each authority runs the authority setup algorithm with GP as input to produce its own secret key and public key pair, (SK, PK).
- $Encrypt(M, (A, \rho), GP, \{PK\}) \rightarrow CT$: It takes in a message M, an access matrix (A, ρ), the set of public keys for relevant authorities, and the global parameters. It outputs a ciphertext CT.
- $KeyGen(GID, GP, i, SK) \rightarrow K_{i,GID}$: It takes in identity GID, the global parameters, an attribute i belonging to some authority, and the secret key SK for this authority. It produces a key $K_{i,GID}$ for this attribute, identity pair.

- $Decrypt(CT, GP, \{K_{i,GID}\}) \rightarrow M$: It takes in the global parameters, the ciphertext, and a collection of keys corresponding to attribute, identity pairs all with the same fixed identity GID. It outputs either the message M when the collection of attributes i satisfies the access matrix corresponding to the ciphertext. Otherwise, decryption fails.

2.2 RSN'12 Protocol

The protocol involves a user who may be a writer or a reader or both, a trusty authority (TA) registering users, one or more Key Distribution Center (KDC) issuing the secrets keys associated with users' attributes, and the CSP. TA and KDCs are trusted entities, CSP is a semi-trusted entity. However, some users may be malicious and thus are considered as untrusted entities. The protocol is composed of three sub-protocols.

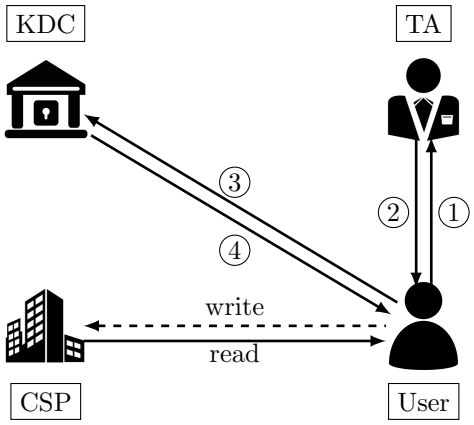

Figure 1: RSN'12 protocol.

Registering and getting attribute secret keys:. In a first phase, a user gets attribute secret keys from the KDCs by presenting his token obtained from the TA:

- The user presents his identity to the TA, for instance a federal government (① in Figure 1).
- The TA registers the user if he is eligible and gives him a token as described in ABS scheme (② in Figure 1). The TA embeds a random value in the token. This random value will be incorporated in the attribute secret keys for signing to prevent collusion of the users.
- If a user presents the token to one or more KDCs then she receives secret attribute keys for signing and decryption (③ in Figure 1). The KDC checks the validity of the token using the TA's public key, and sends the corresponding keys for signing and decryption (④ in Figure 1).

Writing on the cloud:. To store a message MSG on the cloud, the user proceeds as follows:

- The user creates an access policy $\mathcal{X}$ containing all required fields, and encrypts the message MSG under $\mathcal{X}$ as $C = Encrypt(MSG, \mathcal{X})$ (① in Figure 2).
- Then he calculates the message $C_1 = \mathcal{H}(C)\|\tau$ where $\mathcal{H}$ is a hash function, τ is a timestamp and $\|$ is the concatenation operation. The timestamp is used to prevent the user to use stale message back with a valid signature when his attributes have been revoked. Next, he generates the signature σ of C_1 with a claim policy $\mathcal{Y}$ (② in Figure 2).
- Finally, the user sends $c = (C, \tau, \sigma, \mathcal{Y})$ to the CSP (③ in Figure 2). Then CSP verifies, using $Verify$ algorithm, if the message $\mathcal{H}(C)\|\tau$ was signed by a user satisfying the claim policy $\mathcal{Y}$.

Reading from the cloud:. A user can access at any time to the data and requests a ciphertext, then the CSP sends the requested ciphertext using SSH.

Note that, the authors do not propose any revocation model, but it is still possible to incorporate a revocation model. The protocol specification is clear but contains some ambiguities. Next, we discuss these minor problems and explain how to fix them.

Timestamps: They are used to prevent the writing when the attributes and keys of a writer have been revoked, since a timestamp informs of the time when the message was created. However a writer signs its message along with a timestamp generated by himself. Then a verifier cannot really be sure, since the signer may include an arbitrary timestamp. Then in order to address this problem, we recommend the use of a trusted timestamping described in RFC 3161 (Adams et al., 2001), where the writer sends the hash of its message (① in Figure 3) to a trusted third party called TSA-Time-Stamping Authority (② in Figure 3). Then, the TSA concatenates a timestamp τ to the hash and calculates the hash of this concatenation. This hash is then digitally signed with the private key of TSA to give the signature σ (③ in Figure 3). Then, the writers sends to the CSP, the message, the timestamp and the signature (④ in Figure 3). This role of TSA can be ensured by the trusted entity TA in the RSN'12 protocol.

Writer and SSH connection to the CSP: In writing access, the protocol uses SSH connection between users and the CSP which is assumed to be semi-trusted. However, when establishing SSH connection the CSP knows the user's identity following the execution of SSH authentication sub-protocol (Ylonen and Lonvick, 2006), which compromises the user privacy against the CSP. This ambiguity can be easily addressed by configuring the SSH server of the CSP to allow a user to login without any user authentication.

Reader and SSH connection to the CSP: In reading access the SSH connection is useless because the

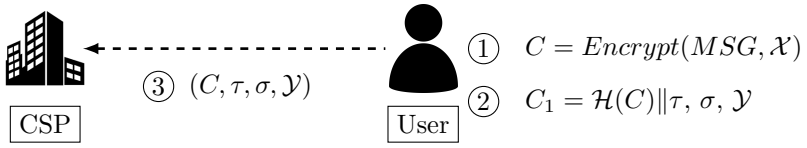

Figure 2: Writing on the cloud.

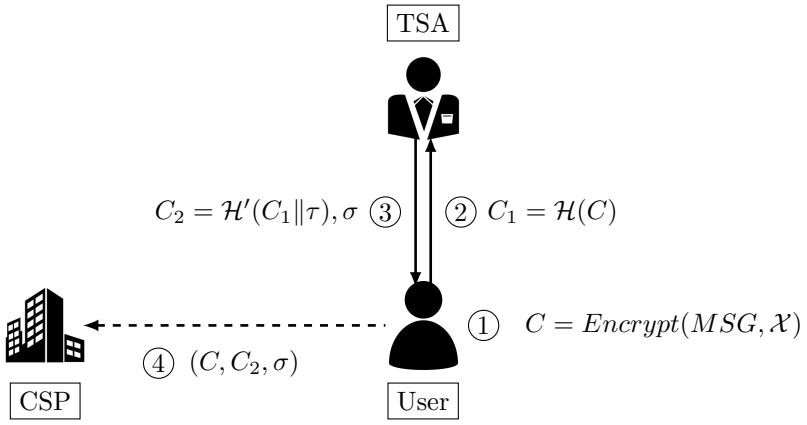

Figure 3: Writing on the cloud with a TSA.

messages are encrypted using ABE and only authorized users can decrypt them. Hence, we can drop the SSH connection between a reader and the CSP, thus helping to reduce the performance overhead due to a large number of SSH connections.

2.3 Computational complexity

The computational complexity of the protocol relies on the practical instantiations used for ABS and ABE schemes. The instantiation of ABS scheme in (Maji et al., 2008) requires cyclic groups of prime order equipped with bilinear pairing. Signing a message needs $2w + l(1 + 2t) + 3$ exponentiation, where l and t are the dimensions of the monotone span program representing the claim policy and w is the minimum number of attributes to satisfy the claim policy. Checking a signature requires $(l + 2t)$ pairing operation. Note that there is a probabilistic verification requiring only $l + 4$ pairings, at the cost of additional exponentiations and a very small probability of false positive (Maji et al., 2008). The probabilistic verification is more efficiency because pairing operation is most expensive operation.

The instantiation of ABE scheme in (Lewko and Waters, 2011) works in composite bilinear groups. The encryption takes $5m$ exponentiations and 1 pairing operation, with m is the number of attributes. The decryption takes $2m + 1$ pairing operation.

3 Applied π-calculus

We give a brief overview of the applied π-calculus, a language for describing concurrent processes and their interactions and handling cryptographic primitives (Abadi and Fournet, 2001).

3.1 Syntax and Semantics

Given a signature Σ, an infinite set of names, and an infinite set of variables, the set of terms are defined by

$L, M, N, T, U, V ::=$	terms
$k, \ldots, n, \ldots, s$	names
x, y, z	variables
$f(M_1, \ldots, M_k)$	function application

An equational theory over a signature usually consists of a set of equations asserting the equality of cryptographic primitives. We usually use E to denote an equational theory. The notation $M =_E N$ means that the equation $M = N$ holds in the theory E. A protocol is modeled as a set of processes defined by the grammar described in Figure 4.

P,Q,R ::=	extended processes
0	null process
$P \mid Q$	parallel composition
!P	replication
new n; P	name restriction
if $M =_E$ then P else Q	conditional
in(c,x);P	message input
out(c,M);P	message output
$\{M/x\}$	active substitution

Figure 4: Grammar of extended processes.

The null process cannot perform any actions. A parallel composition $P|Q$ represents the combined behavior of P and Q executing in parallel. The replication $!P$ behaves as an infinite number of copies of P running in parallel. The process *new n; P* generates a new name n then behaves as P. The name n is bound, i.e. it is not accessible to the environment. The process $if\ M =_E N\ then\ P\ else\ Q$ behaves as P if $M =_E N$, otherwise it behaves as Q. The input process $in(c, x); P$ is ready to input a message on the channel c, then to run $P\{M/x\}$, i.e. to run P with the message M replaced by the formal parameter x. The output process $out(c, M); P$ is ready to output a message M on channel c, then to run P. The active substitution $\{M/x\}$ replaces the variable x with the term M. The active substitution $\{M/x\}$ typically appears when the term M has been sent to the environment. A process is closed if all variables are bound either by an input or by an active substitution. We use $C[\_]$ to denote a context (a process with a whole) and $C[P]$ is a process obtained by filling the whole with process P. An evaluation context is a context whose hole is not under a replication, a conditional, an input, or an output.

The operational semantics of the applied π-calculus contains *structural equivalence*, *internal reduction* (denoted as $\rightarrow$) and *labelled reduction* (denoted as $\xrightarrow{\alpha}$). The structural equivalence defines the equivalence relations between two processes which differ only in structure. Internal reduction means that a process can execute an action without interaction with the environment, while labelled reduction means that a process interacts with the environment. Transition $A \xrightarrow{\alpha} B$ indicates that A executes α and then continues as B.

Many properties of security protocols are formalized in terms of observational equivalence between processes. To define it, we use $A \downarrow c$ to denote that process A can send a message on c.

Observational equivalence ($\approx$): As defined in (Abadi and Fournet, 2001), observational equivalence is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\mathcal{R}B$ implies:

1. if $A \downarrow a$ then $B \downarrow a$,
2. if $A \rightarrow^* A'$ then $B \rightarrow^* B'$ and $A\mathcal{R}B'$ for some B',
3. if $C[A]\mathcal{R}C[B]$ for closing evaluation context C.

Moreover, we consider an active adversary who is able to access to any message sent on public channel, and also to insert, delay and remove messages. In the applied π-calculus the sender's identity of a message is unknown, unless the identity is specified explicitly in the conveyed message.

3.2 ProVerif

ProVerif (Blanchet et al., 2001) is a tool for automatically proving cryptographic protocols. The input of the ProVerif is a protocol modeled in the applied π-calculus. It outputs whether this protocol satisfies or not a security property which is described as a query. ProVerif is sound but not complete, if ProVerif says that a property is satisfied, then the model really guarantees that property. However, if ProVerif detects a flaw, it might be a false attack due to the nonce approximation done in ProVerif.

ProVerif can analyze three types of security properties. The first one is the *reachability*. It allows evaluating the availability of a given term to an attacker. It is usually used to model the secrecy of a message. The second one is the *correspondence assertion* which is used to capture a relation between events. Formally, a correspondence property is an expression of the form $ev(M_1, \ldots, M_n) \Rightarrow ev'(N_1, \ldots, N_k)$, where M_i, N_j are terms with variables. The property holds if on every execution trace, each occurrence of $ev(M_1 \ldots, M_n)$ is preceded by an occurrence of $ev'(M_1, \ldots, M_n)$. To reason with correspondence the grammar of extended process of Figure 4 are enriched by events of the form $event\ e(M_1, \ldots, M_n); P$. These events mark important stages reached by the protocol but do not otherwise affect the behavior of the protocol. The third one is so-called *diff-equivalence*, which allows us to reason about complex properties that cannot be expressed as reachability or correspondence assertions. It is defined between two processes having the same structure but differs only in the choice of terms. Given two processes P and Q, diff-equivalence is written by the notion of *biprocess* that encodes both P and Q. Such a biprocess uses the construct $choice[M, M']$ to represent the terms that differ between P and Q. Thus, P uses the first component of the choice, M, while Q uses the second one, M'. Note that diff-equivalence is stronger than the observational equivalence defined in applied π-calculus (Definition 3.1). This means that in some cases ProVerif fails to prove diff-equivalence between two processes which are observationally equivalent in the sense of Definition 3.1.

4 Modeling in Applied π-calculus

We define a specific equational theory with abstraction for SSH protocol, ABS and ABE schemes. Then we model the protocol in the applied π-calculus from the viewpoint of four types of principals. In our abstraction of ABS and ABE schemes, for sake of simplicity, we consider one attribute and consequently, the protocol involves one KDC. However, our modeling of the claim and access policies can be extended to consider more than one attribute, one access policy, and one claim policy.

4.1 Equational Theory

We describe the equations modeling the cryptographic primitives used within RSN'12 protocol.

Standard cryptographic primitives:. SSH protocol includes standard primitives such as symmetric and asymmetric encryption, digital signature and concatenation of messages.
Symmetric encryption: A symmetric encryption scheme allows the sender to encrypt his message using a shared secret key `k`, whereas a symmetric decryption algorithm allows the receiver to decrypt the ciphertext using `k`. The functions `senc` and `sdec` model, respectively, encryption and decryption. The decryption of an encrypted message `senc(k,msg)` is modeled by the following equation:

```
sdec(k,senc(k,msg)) = msg
```

Asymmetric encryption: In such encryption scheme, each participant of a protocol possesses a pair of keys, one private only known by the owner and used to decrypt ciphertexts, and the other one public used to encrypt messages. To model this scheme, we consider a constant `sk` (function of arity 0) representing a private key and a function `pk` for generating public key corresponding to `sk`. The function `adec` models the decryption of an encrypted message `msg` represented by the term `aenc(sk,msg)` as described by the following equation:

```
adec(sk,aenc(pk(sk),msg))
```

Digital signature: It is used in the authentication of message and thus the authenticity of participants. *Non-message revealing digital* signature is a digital signature which does not allow getting the message from the signature. In a similar way to asymmetric encryption, digital signatures rely on a private and public keys. The private key serves for computing signatures and the public key for verifying those signatures. In order to model a non-message revealing digital signatures and their verifiable, in addition to the function `pk` from asymmetric encryption, we use function `sign` for constructing signatures and the function `checksign` to check the signature, and returning `true` only when the signature is correct.

```
checksign(pk(sk),sign(sk,msg),msg) = true
```

Concatenation: Algebraic data structures such as tuples of messages appear in many protocols and are supported by ProVerif. A tuple of length $n > 1$ is defined as $(M_1, \ldots, M_n)$ where $M_1, \ldots, M_n$ are terms. Given a tuple, it is possible to recover the i-th element.

ABS equational theory:. ABS scheme uses separate keys, constructed from the public parameters which are set up by TA. The equational theory modeling ABS scheme consists of five equations given in Figure 5.

The function `absPK` is used for deriving the public key of the TA from its private key. The term `absToken(TSK,base,uid)` denotes the token delivered by a TA with secret key `TSK`, to a user with identity `uid`. Any entity can check the validity of the token using `absTokenCheck` algorithm, and the public key `absPk(TSK)` as described by Equation (ABS-1). The

```
(ABS-1)absTokenCheck(absPk(TSK),uid,
          absToken(TSK,base,uid)) = true.
(ABS-2)absGetBase(absToken(TSK,base,uid))=base.
(ABS-3)absKeyCheck(absPk(TSK),absPkA(absPk(TSK),
       ASK),absSka(ASK,base,att),base,att)=true.
(ABS-4)absEval(ClaimP,x)=x.
(ABS-5)absSignCheck(absPk(TSK),APK,msg,ClaimP,
       absSign(absPk(TSK),APK,absToken(TSK,base,
               uid),skA,msg,CaimP))
       = absEval(ClaimP,absKeyCheck(absPk(TSK),
       APK,skA,base,att)).
```

Figure 5: ABS equational theory.

algorithm extraction of the random value `base` from a token is modeled by Equation (ABS-2). A KDC extracts `base` value to embedded it in the attribute secret keys of the users. This allows us to link the attribute secret keys to one user and consequently, to avoid the collusion of the users. An attribute secret key corresponding to attribute `att` is denoted by `absSka(ASK,base,att)` with `ASK` is the private key of the issuing KDC. It can be checked for correctness using `absKeyCheck` algorithm as described in the equation (ABS-3). Since it is possible to have many TA, and a KDC would issue attribute keys to a user for all the TA he wishes to work with, the public key of a KDC, denoted by `absPkA(absPk(TSK),ASK)`, is associated with its secret key `ASK` and the public `absPk(ASK)` of the TA with which the user works. The evaluation of the claim policy, which expressed by a Boolean function, is modeled by Equation (ABS-4). Given a Boolean function F, the term $absEval(F, x_1 \ldots, x_n)$ is equivalent to $F(x_1, \ldots, x_n)$. Hence, when considering more than one attribute, `absEval` will be used to represent the truth table of the Boolean function. The functions `absSign` and `absSignCheck` are employed to represent respectively the signature and verification procedures. As described in Equation (ABS-5), `absSignCheck` returns the truth value of claim policy evaluation, which is true only if the message was signed using a valid attribute secret key.

ABE equational theory:. As ABS scheme, ABE scheme uses separate keys, but does not require a central authority. The equational theory of ABE scheme is given in Figure 6.

```
(ABE-1)abeKeyCheck(abePk(ASK),
          abeSka(ASK,uid,att),uid,att)=true.
(ABE-2)abeEval(AccessP,msg,true)=msg.
(ABE-3)abeDec(abeEnc(abePk(ASK),AccessP,
          msg),abeSka(ASK,uid,att))
          = abeEval(AccessP,msg,
          abeCheckKey(abePk(ASK),abeSka(ASK,uid,
          att),uid,att)).
```

Figure 6: ABE equational theory.

In ABE scheme each KDC has a private key and a public key derived from its secret one using the function `abePk`. The evaluation of access policy and the verification of attribute secret key using respectively `abeEval` and `abeKeyCheck` are similar to ABS scheme as is shown by the equations (ABE-1) and (ABE-2). An attribute secret key associated to an attribute `att`, and delivered by a KDC with private key `ASK` to a user of identity `uid`, is denoted by the term `abeSka(ASK,uid,att)`. Here the `uid` of the user appears in the attribute keys because the technique preventing the collusion of users embeds the user's identifier `uid` into the attribute keys. The encryption algorithm, denoted by the function `abeEnc`, takes as input a plaintext, the access policy and the public key of the relevant authority, thus an encrypted message is denoted by a term like `abeEnc(abePk(ASK),AccessP,msg)`. The equation (ABE-3) describes the decryption algorithm `abeDec`. It outputs the message `msg` only when the attribute secret key corresponds to the attribute `att`.

We now model the role of participants in the protocol and their interactions. We start by modeling the role of SSH client and SSH server, then we describe the processes of RSN'12 protocol and its participants.

4.2 Modeling SSH Protocol

To model the transport sub-protocol, only the symmetric encryption is considered, i.e. we ignore the MAC and compression algorithms. Therefore, our abstraction of transport sub-protocol must establish a session key and ensures authentication of the server. A possible abstraction is based on an asymmetric encryption scheme in order to establish a fresh session key for a symmetric encryption scheme and authentication of the server. First, the client creates a session key, which is the symmetric session key. Then, the session key is encrypted with the server's public key and transmitted to the server. To decrypt, the server uses its private key to obtain the session key. Since in applied π-calculus it is difficult to model the sequential numbers, we use random nonce to prevent replay attacks in SSH connections as follows: For each message sent after SSH transport sub-protocol, the sender concatenates its message with a nonce previously received from the receiver. Then, the receiver can check if the message is fresh or not. For sake of clarity, we do not include this use of random value in the below process.

The processes of SSH client and SSH server are given in Figure 7 and 8 respectively. The process of SSH client (resp. SSH server) is split into two sub-process, Clt-SSH-tran (resp. Ser-SSH-tran) modeling transport sub-protocol and Clt-SSH-Auth (resp. Ser-SSH-Auth) modeling SSH authentication sub-protocol. This separation provides some modularity because the communication between a writer and the CSP involves only SSH transport sub-protocol. In the following processes, we use some classical syntactic sugar of ProVerif syntax for readability. The processes P and Q that appear in Figure 8 represent the rest of the server process.

```
Clt-SSH-Trans(pkS) ≜
new SK;
out(c,aenc(pkS,SK));
Clt-SSH-Auth(skClt) ≜
out(c,senc(SK,(sign(skClt,SK),pk(skClt)))).
```

Figure 7: SSH client.

```
Ser-SHH-Trans(skS) ≜
in(c,encSK);
let SK = adec(skS,encSK) in P.
Ser-SHH-Auth(pkClt) ≜
in(c,msg)
let (sig,pkC)= sdec(SK,msg) in
if checksign(pkClt,sig,SK)= true then Q.
```

Figure 8: SSH server.

Note that, in our modeling the legitimate users are identified by their public keys sent as a parameter of the Proverif processes. For instance, in `Ser-SSH-Auth(pkClt)` sub-protocol in Figure 8, the signature `sig` is checked with the key `pkClt` representing a valid authenticator for a legitimate user.

4.3 Modeling RSN'12 Protocol

We describe the main process modeling the RSN'12 protocol. It is specified as the parallel composition of the processes modeling the roles of writers, readers, TA, KDC and CSP.

The main process. It is specified in Figure 9. First, the fresh secret keys `skTA`, `skKDC` and `skCSP`, used respectively by the TA, the KDC, and the CSP for asymmetric encryption and signature are generated. Their corresponding public keys are then sent on public channels, i.e. they are made available to the adversary. Moreover, the fresh secret keys `TSK`, `absASK` and `abeASK` used in ABS and ABE schemes, are also generated and their corresponding public keys are published. The secret keys of writer `skW` and reader `skR` are made under replication to model an infinite number of writers and readers. The processes writer and reader are under replication, because one user may establish many sessions with the CSP. Since TA and KDC processes have the public key of the user as a parameter, they are instanced twice to interact with readers and writers. Moreover, in our modeling, the public keys of asymmetric encryption are used as identities of participants.

The writer process. The writer process given in Figure 10 models the role of a writer. The secret keys `KWTA`, `KWKDC` and `KWCL` are the secret shared keys established by SSH transport protocol respectively with the TA, the KDC and the CSP. After the receipt of a token form the TA, the writer sends a request to the

```
MainProcess ≜
new skTA;new skKDC;new skCSP;
let pkTA=pk(skTA) in
let pkKDC=pk(skKDC) in
let pkCSP=pk(skCSP) in
out(ch,pkTA); out(ch,pkKDC); out(ch,pkCSP);
new TSK; new absASK;
let TPK=absPk(TSK) in
let absAPK=absPkA(TPK,absASK) in
out(ch,TPK); out(ch,absAPK)
new abeASK;
let abeAPK=abePk(abeASK) in
out(ch,abeAPK);
!(new skW;let pkW=pk(skW) in out(ch,pkW);
!Writer(skW,TPK,absAPK,abeAPK,pkTA,pkKDC,pkCSP))|
!(new skR;let pkR=pk(skR) in out(ch,pkR);
!Reader(skR,pkTA,pkKDC)) |
!TA(skTA,TSK,pkW) |
!KDC(skKDC,TPK,absASK,abeASK,pkW) |
!TA(skTA,TSK,pkR) |
!KDC(skKDC,TPK,absASK,abeASK,pkR) |
!CSP(skCSP,TPK,absAPK)
```

Figure 9: Main process.

KDC to get attribute secret key for signing, this request is encoded by the pair `(token,write)`. Afterwards, the writer encrypts his message `msg` and signs it using the attribute secret key `absSkA`. Finally, it sends the signed message `sigMsg` to the CSP.

```
Writer(skW,TPK,absAPK,abeAPK,pkTA,pkKDC,pkCSP)≜
Clt-SSH-Trans(pkTA); (*SSH connection to TA*)
Clt-SSH-Auth(skW);
in(ch,encToken);
let token = sdec(KWTA,encToken) in
Clt-SSH-Trans(pkKDC);(*SSH connection to KDC*)
Clt-SSH-Auth(skW);
out(ch,senc(KWKDC,(token,write)))
in(ch,encAbsSkA);
let absSkA = sdec(KWKDC,encAbsSkA) in
let abeEncMsg = abeEnc(abeAPK,AccessP,msg) in
Clt-SSH-Trans(pkCSP); (*SSH connection to CSP*)
let sigMsg=absSign(TPK,absAPK,token,absSkA,
                        abeEncMsg,ClaimP) in
out(ch,senc(KWCSP,(abeEncMsg,sigMsg))).
```

Figure 10: Writer process.

The reader process. The role of a reader is modeled by reader process given in Figure 11. At first, a reader behaves as a writer by requesting a token from the TA and an attribute secret key for decryption from the KDC. Next, it has access to the CSP, without secure communication SSH, to read a message stored on the cloud. Finally, he decrypts the message read from the CSP using his attribute secret key `abeSkA`, and behaves as `RestOfReader` with the received message.

The TA process. The trusted authority process is given in Figure 12. After the establishment of the shared key for symmetric encryption `KWTA` by SSH transport protocol, and the authentication of the user by SSH authentication protocol which uses the process's parameter `pkClt` as authenticator, the TA generates a token and sends it to the user encrypted with `KWTA`.

```
Reader(skR,pkTA,pkKDC) ≜
Clt-SSH-Trans(pkTA); (*SSH connection to TA*)
Clt-SSH-Auth(skR);
in(ch,encToken);
let token = sdec(KWTA,encToken) in
Clt-SSH-Trans(pkKDC); (*SSH connection to KDC*)
Clt-SSH-Auth(skR);
out(ch,senc(KWKDC,(token,read)));
in(ch,encAbeSkA);
let abeSkA = sdec(KWKDC,encAbsSkA) in
in( c,encMsg); (*Reception of data from the CSP*)
let msg = abeDec(encMsg,abeSkA) in
RestOfReader.
```

Figure 11: Reader process.

```
TA(skTA,TSK,pkClt) ≜
Ser-SSH-Trans(skTA); (*SSH connection to a user*)
Ser-SSH-Auth(pkClt);
new base;
event DelivToken(pkClt);
out(ch,senc(KWTA,absToken(TSK,base,pkClt))).
```

Figure 12: Trusted Authority process.

The KDC process. The KDC process is given in Figure 13. When receiving a request from a user, the KDC checks the correctness of the token using the public key `pkClt` of the user, which was authenticated during SSH authentication protocol. If the token is valid, it issues an attribute secret key for encryption or signing following the value of `AccessMode`, which has two possible values: "write" or "read".

```
KDC(skKDC,TPK,absASK,abeASK,pkClt) ≜
Ser-SSH-Trans(skKDC);(*SSH connection to a user*)
Ser-SSH-Auth(pkClt);
in(ch,encToken);
let (token,AccessMode)=sdec(KWKDC,encToken) in
if absTokenCheck(TPK,pkClt,token)= true then
if AccessMode = write then
event DelivKeySign(pkClt);
out(ch,senc(KWKDC,absSka(absASK,
                    absGetBase(token),att)));
else if AccessMode = read then
out(ch,senc(KWKDC,abeSka(abeASK,pkClt,att)))
else 0.
```

Figure 13: Key Distribution Center process.

The CSP process. The CSP which is responsible for the storage of user data is modeled by the process in Figure 14. SSH connection without user authentication is established between writers. If the signature is valid with respect to the claim policy, the CSP stores the message that becomes immediately accessible by the readers. Since in reading mode, there is no secure communication between the reader and the CSP, in our modeling the CSP outputs the incoming messages from the writers on a public channel.

```
CSP(skCSP,TPK,absAPK) ≜
Ser-SSH-Trans(skCSP); (*SSH with a writer*)
in(ch,encMsg);
let (msg,sigMsg) = sdec(KWCSP,encMsg) in
if absSignCheck(TPK,absAPK,msg,ClaimP,sigMsg)=
true then
event AcceptSign;
out(ch,msg).
```

Figure 14: Cloud Server Porvider process.

5 Security Analysis

We analyze the security properties of the protocol, namely the authentication and privacy of a writer, and the confidentiality of the data. All proofs of our propositions are not presented because they are directly implied by our ProVerif codes.

5.1 Confidentiality

It means that a user without valid access policy cannot decrypt the data stored on the cloud. In applied π-calculus this property can be expressed as a secrecy property: it should be impossible for an adversary, interacting with the protocol and without valid attribute secret key, to learn a message which is encrypted and stored on the cloud.

Definition 5.1: Given an access policy AP, a cloud storage protocol ensures confidentiality if a secret message stored on the cloud by an honest writer is not deducible by an attacker without attribute secret key satisfying AP.

Proving secrecy property is expressed by the reachability notion. We request ProVerif to check that a private message, encrypted using a public access policy AP, cannot be deduced by the attacker. The valid attribute secret key of AP `abeSka(ASK,uid,att)`, which has been generated by the KDC, is not sent on a public channel to be not available to the adversary. ProVerif proves this property in less one minute.

Proposition 1: *RSN'12 protocol ensures the confidentiality property.*

This result confirms the fact that SSH communication between the CSP and a reader is useless for confidentiality, since our modeling does not use it and ProVerif is able to prove the secrecy of the message.

5.2 Writer Authentication

A user can only write in the cloud if he has the attribute validating the claim policy. Moreover, an invalid user cannot receive attribute from a KDC, if does not have the token from the TA. Authentication property can be captured as a correspondence assertion. To define the authentication of a writer, we annotate the protocol by the following events:

- `AcceptSign`: This event is placed inside the CSP's process and emitted if the signature is valid, i.e. `absCheckSign` returns true.
- `DelivKeySign(pkClt)`: This event is placed inside the KDC's process and emitted when the KDC issues an attribute secret key for signing to a user with identity `pkClt`.
- `DelivToken(pkClt)`: This event is placed inside the TA's process and emitted when the TA delivers a token to a user with identity `pkClt`.

Definition 5.2: A cloud storage protocol ensures the authentication of a writer with identity Id if for every execution trace of the protocol each occurrence of the event $AcceptSign$ is preceded by an occurrence of $DelivKeySign(Id)$ which is preceded by an occurrence of $DelivToken(Id)$.

This property can be expressed in ProVerif in terms of nested correspondence (Blanchet, 2009) which allows us to order events. ProVerif can automatically prove the corresponding nested correspondence in less one second: $event(acceptSign) \Rightarrow (event(DelivKeySign(pkwriter)) \Rightarrow event(DelivToken(pkwriter)))$

Proposition 2: *RSN'12 protocol satisfies the authentication of a writer.*

5.3 Writer Privacy

In the context of cloud storage, writer privacy is expressed by two properties; anonymity and unlinkability. The anonymity of a writer's identity is ensured if it is not possible for anyone, even the CSP, to learn the writer's identity of a stored message. Unlinkability means that no one can link the messages stored on the cloud, more precisely no one is able to decide if two messages were stored by the same writer, or not.

Anonymity:. A cloud storage system ensures anonymity if it keeps the writer's identity secret from everyone. Hence, anonymity can be formalized as a secrecy property: no one can deduce the identity of a writer who stores a message on the cloud. Since the identities of the writers are known values, anonymity is captured by the concept of *strong secrecy*. Strong secrecy means that the adversary cannot distinguish two instances of the same protocol with two different values of the secret. For the precise definition, we refer the reader to (Blanchet, 2004). In ProVerif, strong secrecy is expressed by diff-equivalence defined between processes that share the same structure and differ only in the choice of terms representing the secret values (Blanchet et al., 2008).

Definition 5.3: A cloud storage protocol ensures anonymity of a writer's identity if for any two writers with identities IdW_1, IdW_2 and for any message msg, an adversary cannot distinguish whether msg comes from IdW_1 or IdW_2.

We request to ProVerif to check if

$$C[Writer(IdW_1, msg)] \approx C[Writer(IdW_2, msg)].$$

with $C[\_]$ is an evaluation context modeling the whole cloud storage protocol as described in main process with a hole for a writer process, and the process $Writer(IdW, msg)$ models a writer with identity IdW storing a message msg on the cloud. ProVerif succeeds to prove this request in 3 seconds.

Proposition 3: *RSN'12 protocol preserves the anonymity of writer's identity.*

Unlinkability:. Informally, in cloud storage context, unlinkability holds when the different stored messages of the same writer cannot be linked by an attacker even a dishonest user (writer or reader). Thus, unlinkability can be viewed as the secrecy of link between writer and its messages stored on the cloud. The definition of unlinkability is similar to the definition of voter privacy in e-voting protocol (Kremer and Ryan, 2005) in the sense that we must consider at least two honest writers. To understand this assumption, consider the case where all the writers are dishonest except one, as the stored messages on the cloud are published by the CSP, the dishonest writers can collude and determine the message of the honest writer.

Definition 5.4: A cloud storage protocol ensures unlinkability if for any two writers with identities IdW_1, IdW_2 and for any two messages $msg1$ and $msg2$, an adversary cannot distinguish the situation in which IdW_1 stores $msg1$ and IdW_2 stores $msg2$ from the situation in which IdW_1 stores $msg2$ and IdW_2 stores $msg1$.

In applied π-calculus this definition can be formalized as the following equivalence:

$$C[Writer(IdW_1, msg1)|Writer(IdW_2, msg2)] \approx C[Writer(IdW_1, msg2)|Writer(IdW_2, msg1)] ,$$

where $C[\_]$ is an evaluation context modeling the whole protocol with a hole for two writers. In ProVerif, the above pair of processes can be expressed as single biprocess as follows:

$$C[Writer(IdW_1, choice[msg_1, msg_2])] \mid C[Writer(IdW_2, choice[msg_2, msg_1])] .$$

ProVerif finds an attack, in which a man-in-the-middle attacker selectively delays or delete some messages sent to the CSP by one writer until he can link a message to somebody.

Proposition 4: *RSN'12 protocol does not ensure unlinkability property.*

For this attack, we consider an attacker that is a semi-honest reader with valid attribute secret keys, who wants link the messages to a writer. In a real cloud storage environment, to achieve the attack, an attacker performs the following steps:

- Access to the CSP and memorize all the files stored in the cloud.
- Listen to the network and wait for a message sent to the CSP.
- When a new message MSG is sent, he identifies its sender IdW and blocks all the messages sent to CSP after the message MSG. He now has just to wait until MSG becomes available on the CSP, i.e. the CSP appends MSG to the previous files.
- Then, he can access the files and then learn MSG by comparing the current contents of files with the previous contents. Thus, he concludes that MSG was sent by a writer with identity IdW and can link a file to somebody.

We illustrate this attack in Figure 15.

Fixed protocol:. The previously discovered attack against unlinkability is based on the fact that an attacker can instantaneously access to the CSP to learn a message just after it was sent by a writer. To fix this problem, a solution is that the CSP simultaneously publishes at least two incoming messages from different persons. However, the messages are accessible from a file, so if the messages are written on the file in a deterministic order, for instance following the arriving time of the messages, the adversary can link a message with its writer by inspecting the order of the sent messages to the CSP on the network. Therefore, the CSP must write the incoming messages on the files in a non-deterministic way. The new role of the CSP is given in the Figure 16 and illustrated in Figure 17.

The synchronization command `sync 1` in the last line of the above process is introduced to synchronize CSP process. This means that the CSP process waits until the two `sync 1` are reached before publishing the received messages. Therefore, the outputs $out(ch, msg1)$ and $out(ch, msg2)$ of the two received messages are executed in parallel. This parallel execution captures the non-deterministic behavior of the writing of the messages on the file, because the semantic of a parallel composition $P \mid Q$ allows simultaneously and independently execution of P and Q. Moreover, in this case synchronization helps to automatically prove diff-equivalence by ProVerif, and hence the observational equivalence of applied π-calculus, because it allows swapping data between processes at the synchronization points. In fact, the diff-equivalence is stronger than observational equivalence. In particular,

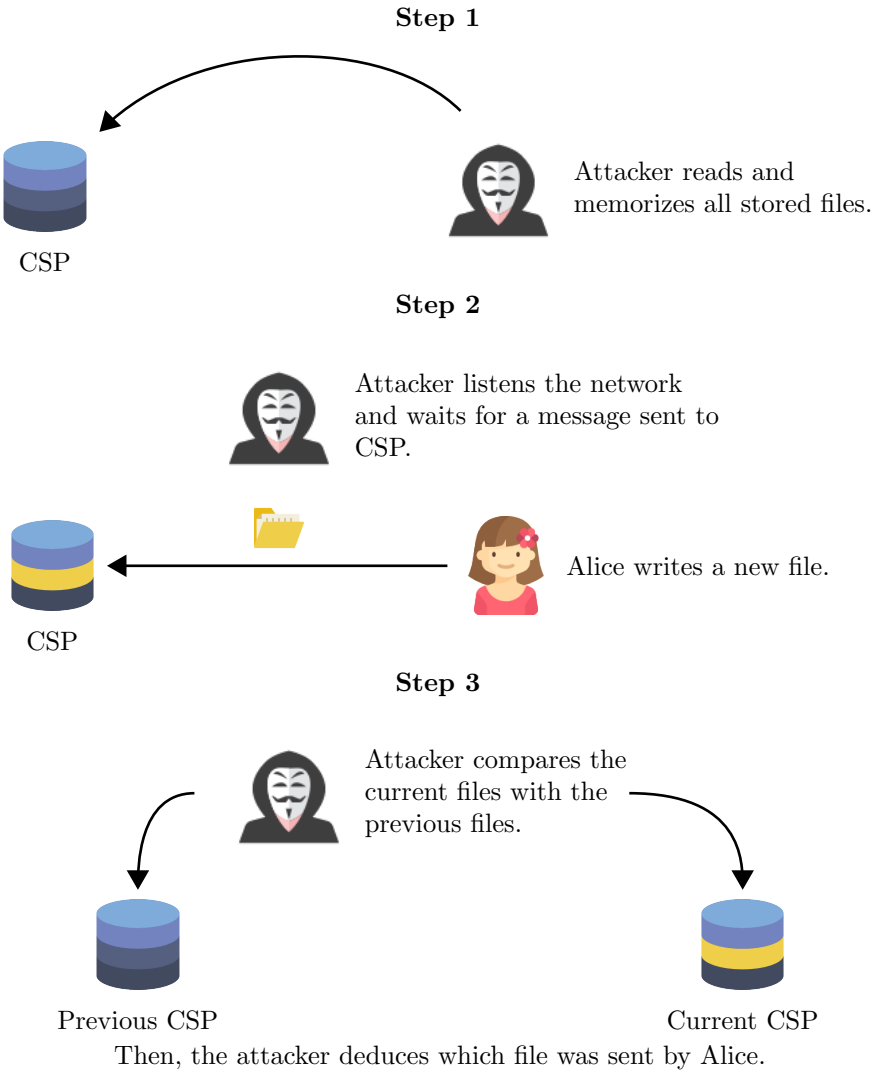

Figure 15: Attack on the unlinkability.

```
FixedCSP(skCSP,TPK,absAPK) ≜
Ser-SSH-Trans(skCSP);
in(ch,encMsg1);
let (msg1,sigMsg1) = sdec(KWCSP,encMsg1) in
if absSignCheck(TPK,APK,msg1,ClaimP,sigMsg1)=true
then in(ch,encMsg2);
let (msg2,sigMsg2) = sdec(KWCSP,encMsg2) in
if absSignCheck(TPK,APK,msg2,ClaimP,sigMsg2)=
true then
(sync 1; out(ch,msg1) | sync 1; out(ch,msg2))
```

Figure 16: Fixed Cloud Server Porvider process.

when proving equivalence between processes that contain several parallel components, e.g., $P \mid Q$ and $P' \mid Q'$, diff-equivalence requires that P is equivalent to P' and Q is equivalent to Q'. This constraint can be relaxed by swapping data between parallel processes at synchronization points. For more details, we refer the reader to (Blanchet and Smyth, 2016). Fortunately, ProVerif succeeds to prove observational equivalence with the new role of the CSP in 28 seconds, and therefore we can conclude the security of the fixed protocol.

Proposition 5: *The revisited RSN'12 protocol ensures unlinkability property.*

6 Conclusion

In this paper, we revisit the security of the protocol of Ruj et al. (Ruj et al., 2012). We propose a model for ABE and ABS in the well know applied π-calculus framework. We use ProVerif to prove automatically claimed security properties by the authors in the original paper. ProVerif helps us to discover a flaw in this protocol for the unlinkability property. We then give a correction and prove the security of the modified version with ProVerif.

The next step is to formalize and verify other properties such as accountability. A system is accountable when it is able to detect a malicious action and generates an undeniable evidence to identify the originator. It could be possible to extend *RSN'12* protocol by a revocation technique and then analyze the resistance against the replay attacks. Moreover, we will use our framework to model and analyze more

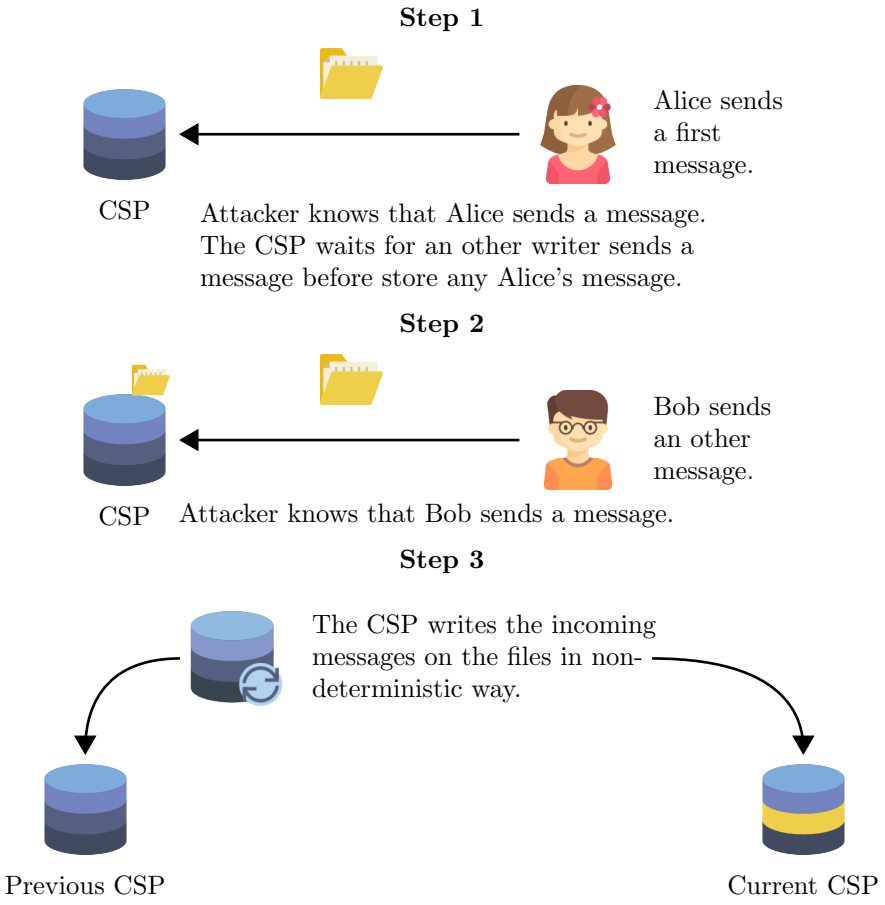

Figure 17: Fixed Cloud Server Provider process.

protocols using ABE and ABS (Yang and Jia, 2012, 2014; Dahshan and Elkassass, 2014; Belguith et al., 2016), in order to discover flaws or formally prove the security of these protocols.

References

Abadi, M. and Fournet, C. (2001). Mobile values, new names, and secure communication. In *ACM SIGPLAN Notices*, volume 36, pages 104–115. ACM.

Adams, C., Cain, P., Pinkas, D., and Zuccherato, R. (2001). Internet x.509 public key infrastructure time-stamp protocol (tsp). RFC 3161, RFC Editor.

Alsmirat, M. A., Jararweh, Y., Obaidat, I., and Gupta, B. B. (2017). Internet of surveillance: a cloud supported large-scale wireless surveillance system. *The Journal of Supercomputing*, 73(3):973–992.

Bansal, C., Bhargavan, K., Delignat-Lavaud, A., and Maffeis, S. (2013). Keys to the cloud: Formal analysis and concrete attacks on encrypted web storage. In *International Conference on Principles of Security and Trust*, pages 126–146. Springer.

Belguith, S., Kaaniche, N., Jemai, A., Laurent, M., and Attia, R. (2016). Pabac: a privacy preserving attribute based framework for fine grained access control in clouds. In *SECRYPT 2016: 13th International Conference on Security and Cryptography*, volume 4, pages 133–146. Scitepress.

Berrima, M., Lafourcade, P., Giraud, M., and Rajeb, N. B. (2017). Formal analyze of a private access control protocol to a cloud storage. In *SECRYPT 2017: 14th International Conference on Security and Cryptography*, volume 4, pages 495–500. SciTePress.

Blanchet, B. (2004). Automatic proof of strong secrecy for security protocols. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 86–100.

Blanchet, B. (2009). Automatic verification of correspondences for security protocols. *J. Comput. Secur.*, 17(4):363–434.

Blanchet, B., Abadi, M., and Fournet, C. (2008). Automated verification of selected equivalences for security protocols. *The Journal of Logic and Algebraic Programming*, 75(1):3–51.

Blanchet, B. et al. (2001). An efficient cryptographic protocol verifier based on prolog rules. In *CSFW*, volume 1.

Blanchet, B. and Smyth, B. (2016). Automated reasoning for equivalences in the applied pi calculus with barriers. In *CSF'16*, pages 310–324.

Boyen, X. (2007). Mesh signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 210–227. Springer.

Chaum, D. and Van Heyst, E. (1991). Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer.

Chen, X., Lenzini, G., Mauw, S., and Pang, J. (2013). Design and formal analysis of a group signature based electronic toll pricing system. *JoWUA*, 4(1):55–75.

Cramer, R., Damgård, I., and Schoenmakers, B. (1994). Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer.

Cremers, C. J. F., Lafourcade, P., and Nadeau, P. (2009). Comparing state spaces in automatic security protocol analysis. In *Formal to Practical Security - Papers Issued from the 2005-2008 French-Japanese Collaboration*, pages 70–94. Springer.

Dahshan, M. and Elkassass, S. (2014). Framework for securing data in cloud storage services. In *Security and Cryptography (SECRYPT), 2014 11th International Conference on*, pages 1–8. IEEE.

Delaune, S., Kremer, S., Ryan, M. D., and Steel, G. (2010). A formal analysis of authentication in the tpm. In *International Workshop on Formal Aspects in Security and Trust*, pages 111–125. Springer.

Giraud, M., Anzala-Yamajako, A., Bernard, O., and Lafourcade, P. (2017). Practical passive leakage-abuse attacks against symmetric searchable encryption. *IACR Cryptology ePrint Archive*, 2017:46.

governmental organisations (2009). Iso 15408-2: Common criteria for information technology security evaluation - part 2: Security functional components.

Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS'06*.

Grover, A. and Kaur, B. (2016). A framework for cloud data security. In *Computing, Communication and Automation (ICCCA), 2016 International Conference on*, pages 1199–1203. IEEE.

Hosseinzadeh, S., Laurén, S., Rauti, S., Hyrynsalmi, S., Conti, M., and Leppänen, V. (2017). Obfuscation and diversification for securing cloud computing. In *Enterprise Security*, pages 179–202. Springer.

Huang, H., Guo, S., Wu, J., and Li, J. (2017). Service chaining for hybrid network function. *IEEE Transactions on Cloud Computing*.

Ibtihal, M., Hassan, N., et al. (2017). Homomorphic encryption as a service for outsourced images in mobile cloud computing environment. *International Journal of Cloud Applications and Computing (IJCAC)*, 7(2):27–40.

Kremer, S. and Ryan, M. (2005). Analysis of an electronic voting protocol in the applied pi calculus. In *European Symposium on Programming*, pages 186–200.

Lewko, A. and Waters, B. (2011). Decentralizing attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 568–588. Springer.

Li, M., Yu, S., Ren, K., and Lou, W. (2010). Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings. In *International Conference on Security and Privacy in Communication Systems*, pages 89–106. Springer.

Maji, H. K., Prabhakaran, M., and Rosulek, M. (2008). Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. *IACR Cryptology ePrint Archive*, 2008:328.

Memos, V. A., Psannis, K. E., Ishibashi, Y., Kim, B.-G., and Gupta, B. (2017). An efficient algorithm for media-based surveillance system (eamsus) in iot smart city framework. *Future Generation Computer Systems*.

Mulazzani, M., Schrittwieser, S., Leithner, M., Huber, M., and Weippl, E. (2011). Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11.

Puys, M. and Lafourcade, P. (2015). Evaluations of cryptographic protocols: Verification tools dealing with algebraic properties. In *Foundations and Practice of Security - FPS 2015*. Springer.

Puys, M., Potet, M.-L., and Lafourcade, P. (2016). Formal analysis of security properties on the opc-ua scada protocol. In *International Conference on Computer Safety, Reliability, and Security*, pages 67–75. Springer.

Rivest, R. L., Shamir, A., and Tauman, Y. (2001). How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer.

Ruj, S., Stojmenovic, M., and Nayak, A. (2012). Privacy preserving access control with authentication for securing data in clouds. In *Cluster, Cloud and*

Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, pages 556–563. IEEE.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer.

Shakunthala, B., Kavya, N., KG, K. S., Sahana, C., and Shetraradder, K. G. (2017). Decentralized access control with anonymous authentication of data stored in clouds.

Stergiou, C., Psannis, K. E., Kim, B.-G., and Gupta, B. (2016). Secure integration of iot and cloud computing. *Future Generation Computer Systems*.

Tang, Z., Wang, X., Jia, L., Zhang, X., and Man, W. (2012). Study on data security of cloud computing. In *Engineering and Technology (S-CET), 2012 Spring Congress on*, pages 1–3. IEEE.

Wang, B. Y., Ming, J., Zhang, S. M., Jiang, H., and Luo, H. (2014). An access control method based on cp-abe and abs algorithm in cloud storage. In *Applied Mechanics and Materials*, volume 644, pages 1919–1922. Trans Tech Publ.

Wang, C., Ren, K., Lou, W., and Li, J. (2010). Toward publicly auditable secure cloud data storage services. *IEEE Network*, 24(4):19–24.

Yang, K. and Jia, X. (2012). Attributed-based access control for multi-authority systems in cloud storage. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 536–545. IEEE.

Yang, K. and Jia, X. (2014). Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE transactions on parallel and distributed systems*, 25(7):1735–1744.

Ylonen, T. and Lonvick, C. (2006). The secure shell (ssh) authentication protocol. RFC 4252, RFC Editor.

Yu, S., Wang, C., Ren, K., and Lou, W. (2010). Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*.

Zhang, G., Yang, Y., Zhang, X., Liu, C., and Chen, J. (2012). Key research issues for privacy protection and preservation in cloud computing. In *Second International Conference on Cloud and Green Computing, CGC'12*, pages 47–54.

Zkik, K., Orhanou, G., and El Hajji, S. (2017). Secure mobile multi cloud architecture for authentication and data storage. *International Journal of Cloud Applications and Computing (IJCAC)*, 7(2):62–76.