OAuth and ABE based Authorization in Semi-Trusted Cloud Computing

Anuchart Tassanaviboon and Guang Gong

Department of Electrical and Computer Engineering
University of Waterloo
CANADA
<http://comsec.uwaterloo.ca/∼atassana, ggong>

Outline

- **Security** requirements in cloud environment
- Solutions & **challenges** in semi-trusted cloud computing (STCC)
- **Review** of OAuth and ABE based schemes
- **AAuth**: A new authenticated **authorization** scheme for securing STCC
- **Performance** evaluation and simulation
- Conclusions and remarks

Vulnerabilities

- **Web-interface flaws**, XML signature wrapping, legacy same origin policy, unsecured browser authentication
- **Leak virtual isolation**, side/covert channel, cross-tenant data access
- **Image insanity**, malicious/illegal images
- Limited **network control**, under-provisioning, limited QoS, new form of DoS
- **Weak access control**, weak credentials, weak tokens, coarse authorization
- **Lack of standards**, APIs, inter-operations

Review of Access Control (AAA)

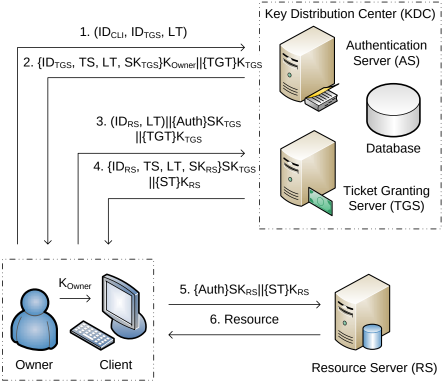
- **Typical** models
  - Centralized server
  - Client-server: Kerberos/Active Directory
  - HTTP: OpenID/OAuth
- **Cloud Computing ?**

- **Cloud** problems and challenges
  - Trust boundary is expanded to CSPs
  - CSPs are untrusted or semi-trusted
  - A shared trusted domain doesn't present
  - A single trusted domain is unscalable

AAA Adversary Models

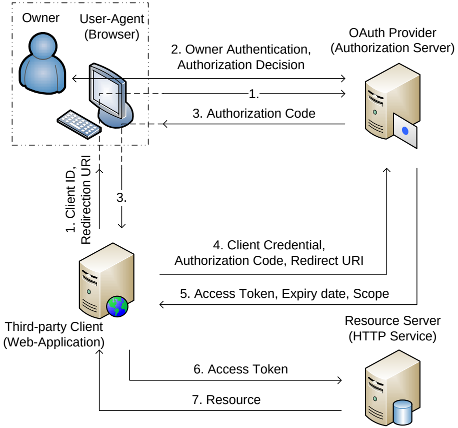
- An authorizer arbitrarily **grants** accesses
- **Cloud** servers reveal sensitive data
- Cloud servers **disobey the access** policies
- Weak tokens cause fabrication, replay attacks, etc.
- **Lock-in** vendors

Kerberos



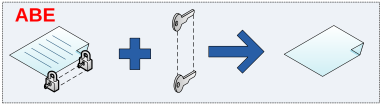
ID_{CLI}: Client ID ID_{RS}: Resource Server ID ID_{TGS}: TGS ID
TS: Timestamp LT: Lifetime
K_A: A's Secret key in KDC SK_B: CLI – B Session key between
TGT = (ID_{CLI}, ID_{TGS}, TS, LT, SK_{TGS}) Auth = (ID_{CLI}, TS)
ST = (ID_{CLI}, ID_{SR}, TS, LT, SK_{RS}) K_{Owner} = Hash(password, salt)

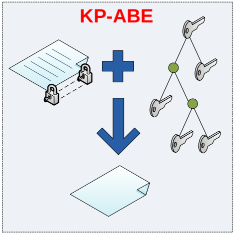
OAuth

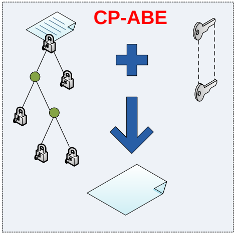


Owner
User-Agent
(Browser)
2. Owner Authentication,
Authorization Decision
1.
3. Authorization Code
OAuth Provider
(Authorization Server)
1. Client ID,
Redirection URI
3.
4. Client Credential,
Authorization Code, Redirect URI
5. Access Token, Expiry date, Scope
Third-party Client
(Web-Application)
Resource Server
(HTTP Service)
6. Access Token
7. Resource

Crypto Tool in the New Scheme: Cipher-Policy Attribute-Based Encryption (CP-ABE)







ABE (Sahai-Waters 05), KP-ABE (Goyal, et. al. 06), and CP-ABE (Bethencourt-Sahai-Waters 07). In order to adapt to our scheme, a **modified CP-ABE** will be introduced later.

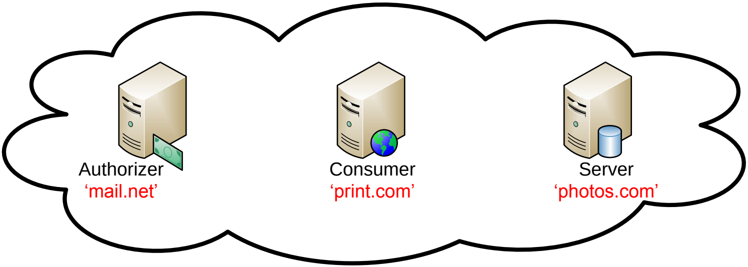
Design Goals

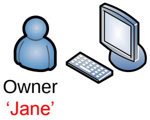
- **Data owners** contribute to token generation.
- Data is encrypted in an **end-to-end** fashion.
- **Policies** are enforced by cryptographic functions.
- **Token** knowledge is distributed among CSPs for reducing risks.
- Scheme is integrated with existing standards and cloud entities.

System Model

- **Data owner** (O): (owners for short) entities, i.e., end-users or software applications, who have resource ownerships and the right to grant access to protected data.
- **Cloud server** (S): (servers for short) cloud-storage or cloud-database providers that host protected data and provide basic data-services, i.e., read, write, and delete.
- **Consumers** (C): web or traditional applications service provider (ASPs) that use owners' data to provide services to the owners.
- **Authority** (AA): trusted organizations or agencies who legitimately define descriptive attributes to eligible consumers.
- **Authorizer** (AZ): the server who runs AAuth protocol, then issues ABE-based tokens to eligible consumers.

System Model (Example)





Confined attributes
[FILE-LOC=http://photos.com/2010/brunce/pic-1]
AND [OWNER=Jane@photos.net]
AND [SEC-CLASS=3]
AND [PERMIS=r]
AND [TIMESLOT=2011/06/27/13/**]
AND # Descriptive attributes
[(OWNERe@mail.net=Jane@mail.net) OR
[(NAME@authority.org=printer.com) AND
(SERVICE@authority.org = print) AND
(LOCAT@authority.org = canada) OR
(TRUST-LEV@authority.org = 3)]].

Pre-conditions and Adversary Model

- **Servers** are trusted to provide data-services properly but may be curious about sensitive information and prone to reveal data to ineligible parties.
- **The authorizer** may disobey owners' orders to issue tokens, or issue any arbitrary tokens to its conspirators.
- **Consumers** may try to get unauthorized files from honest servers by fabricating tokens to obtain unauthorized accesses, resubmitting previous tokens (replay attacks).
- **Internet users** may launch general network attacks on encrypted data or tokens. However, we assume that the communications among CSPs are secure and authentic under SSL/TLS secure channel.
- Adversaries do not have enough computing power to break cryptographic primitives.

AAuth Components

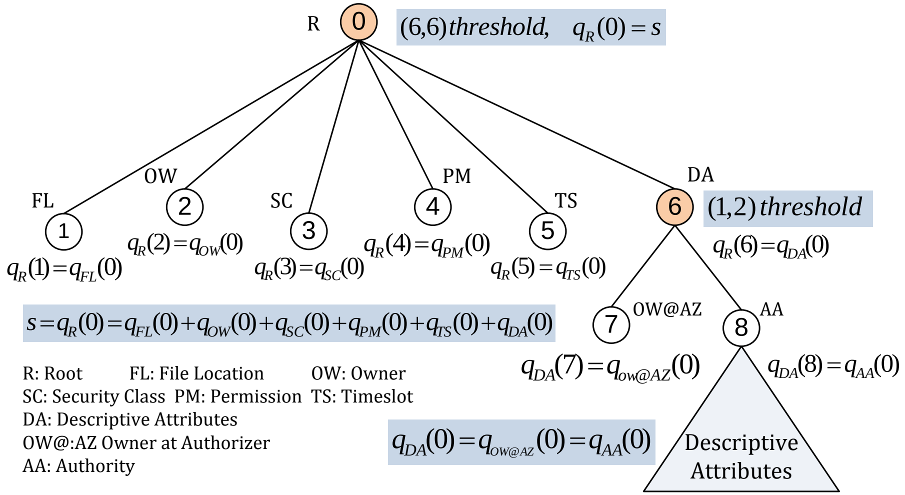
- Access Policy $\mathbb{A}$

$$\begin{aligned}\mathbb{A} =& \text{[FILE-LOC] AND}\\ & \text{[OWNER] AND}\\ & \text{[SEC-CLASS] AND}\\ & \text{[PERMIS] AND}\\ & \text{[TIMESLOT] AND}\\ & \text{[(OWNER@AUTHZ) OR}\\ & (\textit{Descriptive Boolean Algebra})].\end{aligned}$$

- Defined Attributes

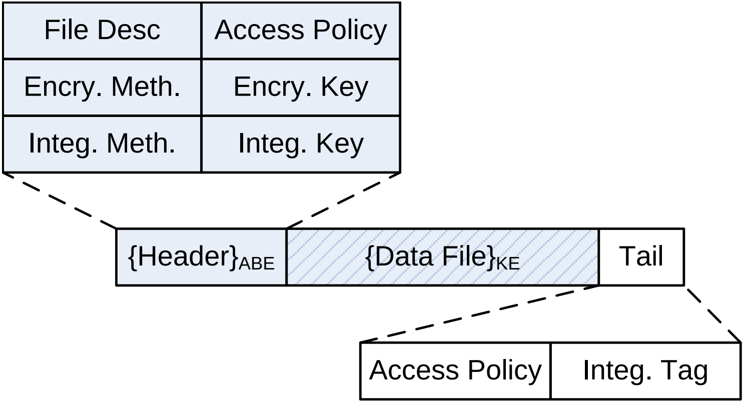
FILE-LOC = *URI*
OWNER = *ownerId*
PERMIS = $\langle r|w\rangle$
SEC-CLASS = $\langle 1 - 5\rangle$
TIMESLOT = *yyyy/mm/dd/hh/nn*

AAuth Components (Cont.): Access Tree τ



R: Root FL: File Location OW: Owner
SC: Security Class PM: Permission TS: Timeslot
DA: Descriptive Attributes
OW@:AZ Owner at Authorizer
AA: Authority

AAuth Components (Cont.): Archive File



Modified CP-ABE

- *Setup*(k)
  - **Authorizer**

System parameters Bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Generator g of group $\mathbb{G}_1$. Hash function $H : \{0,1\}^* \to \mathbb{G}_1$.

    Randomly selects $\beta \in \mathbb{Z}_p$.
    Master Secret Key: $MSK = \langle \beta \rangle$.
    Master Public Key: $MPK = \langle \mathbb{G}_1, g, h = g^{\beta}, f = g^{1/\beta} \rangle$.
  - **Owner**
    Randomly selects $\alpha \in \mathbb{Z}_p$.
    Owner Secret Key: $OSK = \langle g^{\alpha} \rangle$.
    Owner Public Key: $OPK = \langle e(g,g)^{\alpha} \rangle$.

Modified CP-ABE: $Encrypt(MPK, m, \tau)$

- Randomly selects $s \in \mathbb{Z}_p$.
- Construct access tree τ according to $q_R(0) = s$ and an access policy $\mathbb{A}$.
- Let Y be the leave nodes in τ:

Ciphertext: $CT = \langle \tau, \tilde{C} = m \cdot e(g,g)^{\alpha s}, C = h^s,$
$\forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)} \rangle.$

Modified CP-ABE: $KeyGen(MSK, OSK, \omega)$

Assume that an attribute set $\omega = \omega' \cup \omega''$ where w ω' =confined attributes, and ω''= descriptive attribute.

- **Authorizer:** $r \in_R \mathbb{Z}_p$, and selects a set $\{r_i \in_R \mathbb{Z}_p \mid i \in \omega'\}$, i.e., responds for confined attributes.
- **Authority:** selects $\{r_j \in_R \mathbb{Z}_p \mid j \in \omega''\}$, descriptive attributes
- **Owner:** $a \in_R \mathbb{Z}_p$.
- With ElGamal-like masking, the authorizer, the authority, and the owner jointly compute a private key for the consumer
  Private key:

$$SK = \langle D = g^{(\alpha+ra)/\beta}, D_k = g^{ra} \cdot H(k)^{r_k}, D'_k = g^{r_k}\rangle, \forall k \in \omega\rangle.$$

Modified CP-ABE: *Delegate*$(SK, \tilde{\omega})$

- Given a secret key SK for an attribute set ω.
- Let $\tilde{\omega} \supseteq \omega$ denote a new attribute set.
- Random value $\tilde{r}$ and random set $\{\tilde{r}_l \mid \forall l \in \tilde{\omega}\}$.
- A consumer creates a new private key $\widetilde{SK}$ for the attribute set $\tilde{\omega}$:

$$\widetilde{SK} = \langle \tilde{D} = D \cdot f^{\tilde{r}}, \forall l \in \tilde{\omega} \ : \ \tilde{D}_l = D_l \cdot g^{\tilde{r}} \cdot H(l)^{\tilde{r}_l}, \tilde{D}'_l = D'_l \cdot g^{\tilde{r}_l} \rangle.$$

Modified CP-ABE: *Decrypt*(CT, SK)

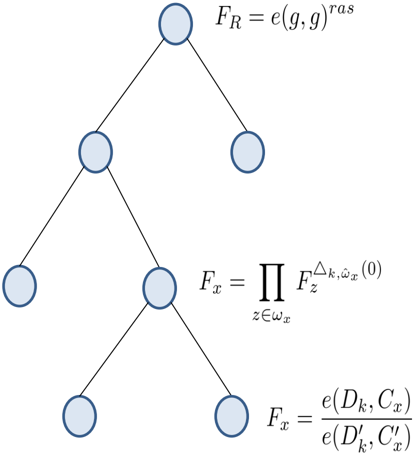
- Recursively computes from the root node R of access tree τ by using node algorithm $DecryptNode(CT, SK, x)$:

$$F_R = DecryptNode(CT, SK, R) = e(g, g)^{ra \cdot q_R(0)} = e(g, g)^{ras}$$

- If the tree τ is satisfied by ω then decryption can be computed by:

$$\begin{aligned} & Decrypt(CT, SK) \\ & = \tilde{C}/(e(C, D)/F_R)\tilde{C}/(e(h^s, g^{(\alpha+ra)/\beta})/e(g, g)^{ras}) \\ & = m \end{aligned}$$

A Diagram of *DecryptNode*(CT, SK, x)



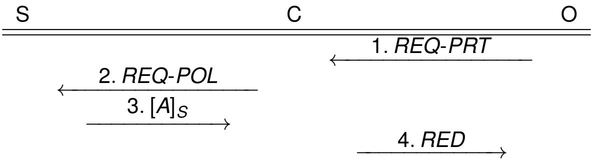
AAuth in a Nutshell

- **AAuth** extends OAuth to a cryptographic token system that its ABE-token is a private key associated with a set of attributes, and its protected resource (data file) is encrypted with an access policy constructed from an access structure over a public key.
- Due to the inapplicability of a single authority in large scale systems, our scheme divides attribute universe in two disjointed sets: **confined attributes** defined by owners to limit the lifetime and scope of tokens, and descriptive attributes defined by authority(s) to certify the characteristic of consumers.
- To allow **owners to contribute** to private-key generation, we separate the master key of CP-ABE to two parts: g^{α} for owners and β for a authorizer, then add another level of ElGamal-liked masks to conceal the master keys from each other during key generation.
- Resource servers have no affiliation in authorization but provide basic data-services: read, write, delete.

AAuth in a Nutshell (cont.)

- AAuth consists of **three off-line procedures**:
  - **Setup**: authorizer chooses a security parameter k and run CP-ABE algorithm $Setup(k)$.
  - **File encapsulation**: the owner encrypts and encapsulate data files into archives file and send to the sever by converting $\mathbb{A}$ to an access tree τ and using the CP-ABE Enc algorithm.
  - **File decapsulation**: consumer verifies the integrity of the archive file, then performs decapsulation of an archive file.
- **four on-line protocols**: service request, token request, file access, and timeslot synchronization, which will be presented in details.
- Optionally, AAuth can provide key delegation, policy change, and data update.

AAuth: Service Request Protocol



AAuth: Token Request Protocol

C O AZ AA

1. *RED*
2. *HTTP Form*
3. *Login*/*password*

allowed Att

4. *REQ-DES*1
5. $[\{\hat{D}_j\}]_{AA}$
6. $\{\hat{D}_i\}, [\{D_i'\}]_{AZ}, [\{\hat{D}_j\}]_{AA}, g^r$
7. $g^{(\alpha+ra)}$
8. *RED*

8. *RED*

9. $\{D_i\}, [\{D_i'\}]_{AZ}, \{D_j\}$
10. *REQ-DES*2
11. $\{D_j'\}$

1. RED=$[ID_C, \textit{RED-URI}]_C, [\mathbb{A}]_S$. 8. RED= $RED([\{g^{(\alpha+ra)/\beta}, \textit{Authz-Code}\}_C]_{AZ})$.

AAuth: File Access Protocol

Server S		Consumer C
	$\xleftarrow{1.\ \mathit{REQ\text{-}FILE}}$	
	$\xrightarrow{2.\ \mathit{Chall}}$	
	$\xleftarrow{3.\ \mathit{Resp}}$	
	$\xrightarrow{4.\ \mathit{Archive}}$	

AAuth: Time Slot Synchronization

Timeslot	0	1	$\cdots$	$n-1$	n
Random value, $\tilde{s}$	$\tilde{s}(0)$	$\tilde{s}(1)$	$\cdots$	$\tilde{s}(n-1)$	$\tilde{s}(n)$
Share, $q_{TS}(0)$	$q_{TS}(0,0)$	$q_{TS}(0,1) = q_{TS}(0,0) + \tilde{s}(1)$	$\cdots$	$q_{TS}(0,n-1)$	$q_{TS}(0,n) = q_{TS}(0,n-1) + \tilde{s}(n)$
Component, C_{ST}	$C_{ST}(0)$	$C_{ST}(1) = g^{q_{TS}(0,1)}$	$\cdots$	$C_{ST}(n-1)$	$C_{ST}(n) = g^{q_{TS}(0,n)}$
Component, C'_{ST}	$C_{ST}(0)$	$C'_{ST}(1) = H(Att_{ST}(1))^{q_{TS}(0,1)}$	$\cdots$	$C_{ST}(n-1)$	$C'_{ST}(n) = H(Att_{ST}(n))^{q_{TS}(0,n)}$
Component, C	$C(0)$	$C(1) = C(0) \cdot h^{\tilde{s}(1)}$	$\cdots$	$C(n-1)$	$C(n) = C(n-1) \cdot h^{\tilde{s}(n)}$
Component, $\tilde{C}$	$\tilde{C}(0)$	$\tilde{C}(1) = \tilde{C}(0) \cdot e(g,g)^{\alpha\tilde{s}(1)}$	$\cdots$	$\tilde{C}(n-1)$	$\tilde{C}(n) = \tilde{C}(n-1) \cdot e(g,g)^{\alpha\tilde{s}(n)}$
Secret mask, s	$s(0)$	$s(1) = s(0) + \tilde{s}(1)$	$\cdots$	$s(n-1)$	$s(n) = s(n-1) + \tilde{s}(n)$

AAuth: Token Delegation

- The web site ‘printer.com’ can ask the website ‘poster.com’ to print a poster for a file ‘pic-1’ in the time slot ‘2011|06|27|13|**’

‘printer.example.com’

FILE-LOC = http://photos.com/2010/brunce/pic-1,
FILE-LOC = http://photos.com/2010/brunce/pic-2,
SEC-CLASS = 3, PERMIS=r,
/* current time slot */
TIMESLOT = 2011|06|27|13|**,
/* future time slot(s)*/
TIMESLOT = 2011|06|27|14|**.

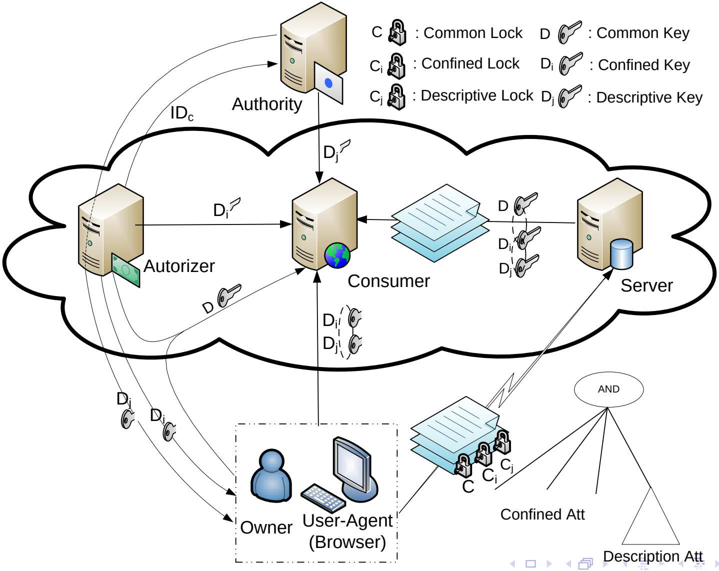
‘poster.com’

FILE-LOC = http://
photos.com/2010/brunce/pic-1,
SEC-CLASS = 3, PERMIS = r,
/* current time slot */
TIMESLOT = 2011|06|27|13|**.

Recap: The procedures and protocols in AAuth

AAuth	
Procedures/Protocols	Outputs
Setup procedure	1. A bilinear group $\mathbb{G}_1, \mathbb{G}_2$ 2. A bilinear map e 3. A generator g of $\mathbb{G}_1$ 4. hash function H
File encapsulation procedure	1. An access policy $\mathbb{A}$ from both confined and descriptive attributes 2. An access tree τ 3. An archive file
Service request protocol	An access policy $\mathbb{A}$
Token request protocol	An ABE-token
File access protocol	An archive file
File decapsulation procedure	1. A header in plaintext form 2. An integrity tag 3. A data file in plaintext form
Time slot synchronization protocol	1. Two ciphertext components 2. Two update values 3. A new time slot header

A Block Diagram of AAuth Authorization Scheme



Security Analysis

i With **end-to-end encryption** and signature, a cloud server cannot subvert the confidentiality and integrity of the data it is hosting.

ii With **end-to-end authorization**, the access policy is enforced by the encryption algorithm, not by a cloud server.

iii Without **cooperation** between owners and the authority, none of them can individually generates ABE-tokens.

iv Since owners can verify confined keys before combining, the **authorizer** cannot faked keys to owners.

v Separating keys to two parts, each of which is individually sent to consumer, to fabricate keys, owners face DLP while consumers face DBDH problems.

vi The scheme can prevent eavesdropping, active, MITM, off-line attacks form external adversaries.

- **On-line Cryptographic Cost**

	Signing	Verify	Exponent	Paring
Owner		1		6
Consumer		2		$2(\lvert I \bigcap L \rvert) + 1$
Authorizer	2		12	
Authority	1		$2\lvert I - 5 \rvert + 1$	
Server	1		$2\lvert L \rvert + 2$	

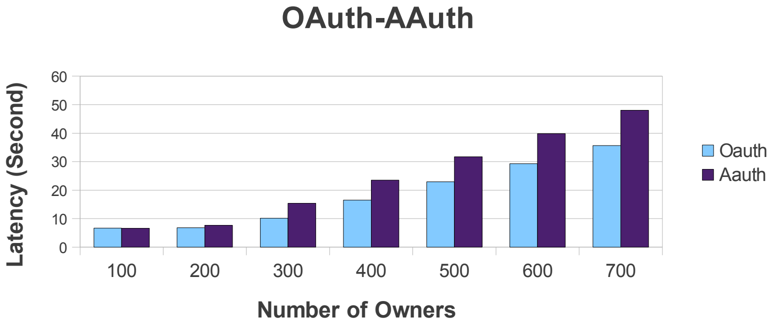
Performance Evaluation (Cont.)

- **Additional Communication Cost**

Protocol	Additional messages	Message flow
Service request	2	$C \rightarrow S$
Token request	2	$AZ \rightarrow AA$
	1	$O \rightarrow AZ$
	1	$C \rightarrow O$
	2	$C \rightarrow AA$
File access	—	

Simulations

- Tool: OMNet++
- Settings: the cloud network has a bandwidth at 400 packets/s, each owner continuously requests services in exponential distribution, each service request transfers three 256 KB-files as a dummy load, the number of owners (users) starts from 100 to 700.



Related Work

- Work on a cryptographic **storage** system
- Proof of Retrievability (POR) (Bowers, 09): a frame work on archival or backup files in cloud storage
- **Proxy** re-encryption and **lazy re-encryption** (Wang, et. al. 2010): Fine-grained and scalable access control in cloud computing that exploits KP-ABE to reduce complexity in key management and key distribution
- K2C (Zarandioon, 2011), a **scalable ABE-based** access hierarchies by combining KP-ABC and key-updating scheme and combining KP-ABE and signature scheme

Conclusions & Remarks

1. **ABE-tokens** for each authorization grant.
2. A **user-centric** system in which an owner controls the authorization system to protect her resources.
3. **End-to-end** cryptographic functions from an owner to a consumer.
4. A light-weight encryption for time slot **synchronization**.
5. No significant computation cost for users.
6. AAuth's cost is independent of the number of users in the system.
7. An acceptable increasing cost is compensated by achieving better security than OAuth.
8. **AAuth** is as secure as the original CP-ABE scheme and can resist both internal and external adversaries.

The comparison of Kerberos, OAuth, and AAuth

	Kerberos	OAuth	AAuth
Trust platform	Client	Browser	Browser
SSO	Yes	Yes	Yes
Key management	No	No	Integrated & distributed
Data-at-rest	Plaintext	Plaintext	Ciphertext
Policy mechanism	ACL / capabilities	ACL / capabilities	ABE attributes
Policy enforced by	server	server	ABE decryption
Token generation	AS & TGS	OAuth provider	Owner, Authorizer, and Authority(s)
Ext. attacks resisted by	Time synch.	SSL/TLS	multi SSL/TLS
Int. attacks resisted by	No	No	modified CP-ABE