Dynamic Credentials and Ciphertext Delegation for ABE

Amit Sahai, Hakan Seyalioglu, Brent Waters

Attribute-Based Encryption
[S-Waters 2005, GPSW'06, BSW'07]

Different users will have credentials (attributes).

Top Secret,
Forensics

Attribute-Based Encryption

[S-Waters 2005, GPSW'06, BSW'07]

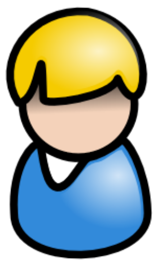
Different users will have credentials (attributes).

`Top Secret,`
`Forensics`

$$\mathsf{KeyGen}(MSK, S) \rightarrow K_S$$

Attribute set =
`Top Secret,`
`Forensics`

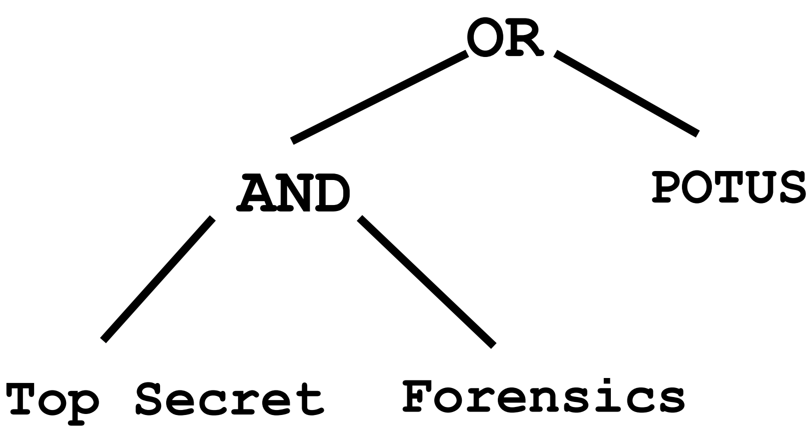
Attribute-Based Encryption



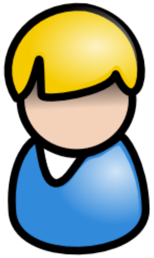
has a message, wants to send it to everyone authorized to receive it.

Encryption takes as input a policy.

$$\mathsf{Encrypt}(PK, M, P) \rightarrow C_P$$



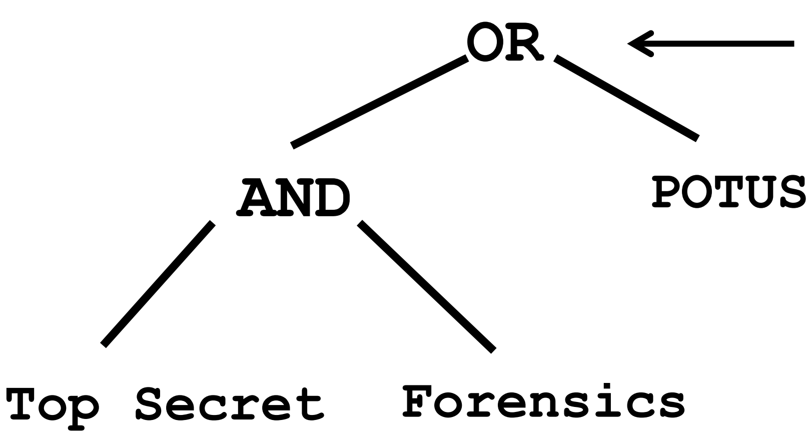
Attribute-Based Encryption



has a message, wants to send it to everyone authorized to receive it.

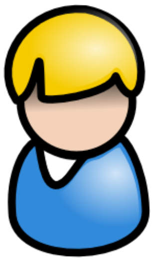
Encryption takes as input a policy.

$$\text{Encrypt}(PK, M, P) \rightarrow C_P$$



Top Secret, Forensics

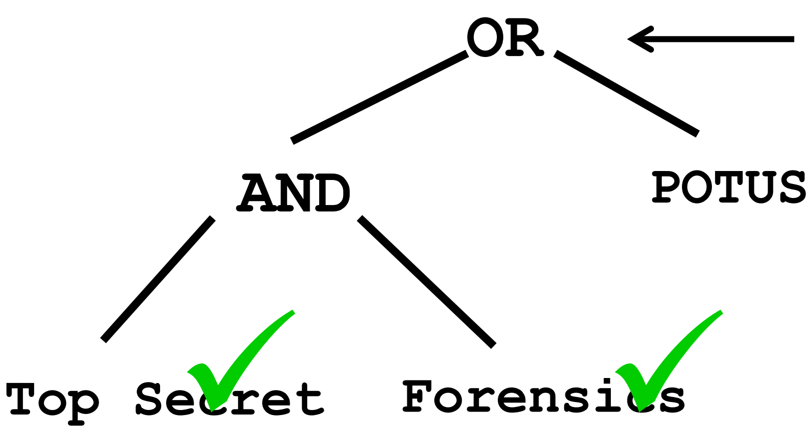
Attribute-Based Encryption



has a message, wants to send it to everyone authorized to receive it.

Encryption takes as input a policy.

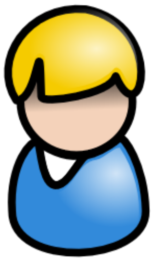
$$\text{Encrypt}(PK, M, P) \rightarrow C_P$$



can decrypt

Top Secret,
Forensics

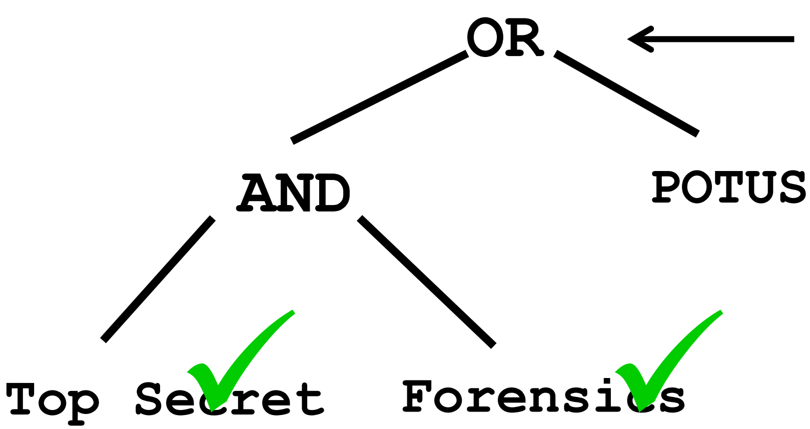
Attribute-Based Encryption



has a message, wants to send it to everyone authorized to receive it.

Encryption takes as input a policy.

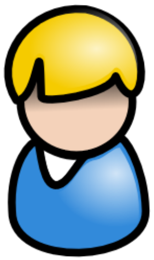
$$\text{Encrypt}(PK, M, P) \to C_P$$



← can decrypt

Top Secret, Forensics

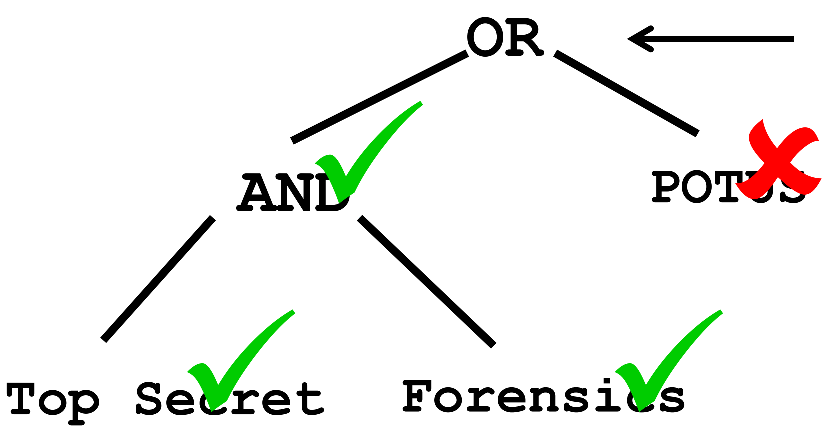
Attribute-Based Encryption



has a message, wants to send it to everyone authorized to receive it.

Encryption takes as input a policy.

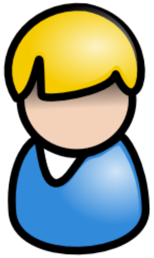
$$\mathsf{Encrypt}(PK, M, P) \rightarrow C_P$$



can decrypt

Top Secret, Forensics

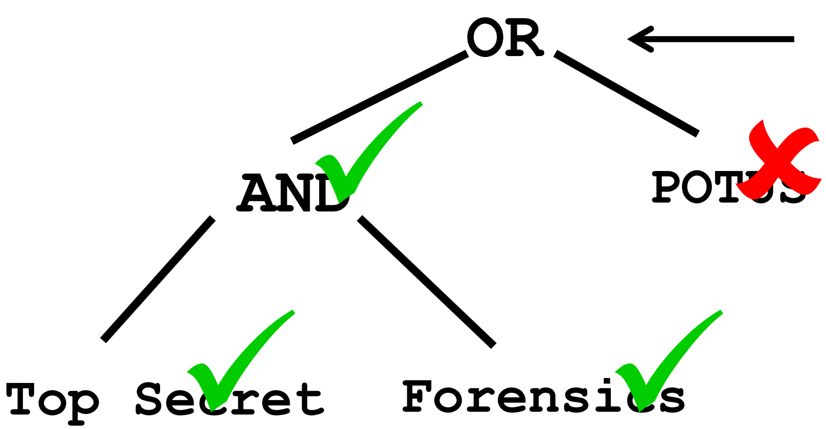
Attribute-Based Encryption



has a message, wants to send it to everyone authorized to receive it.

Encryption takes as input a policy.

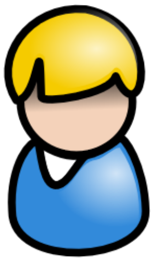
$$\text{Encrypt}(PK, M, P) \rightarrow C_P$$



can decrypt

Top Secret,
Forensics

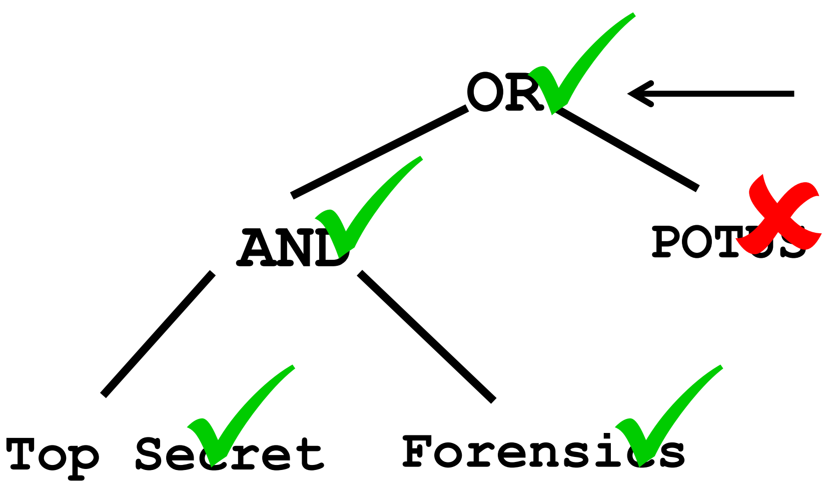
Attribute-Based Encryption



has a message, wants to send it to everyone authorized to receive it.

Encryption takes as input a policy.

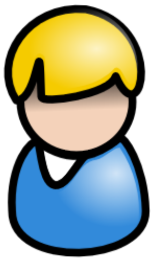
$$\text{Encrypt}(PK, M, P) \rightarrow C_P$$



can decrypt

Top Secret, Forensics

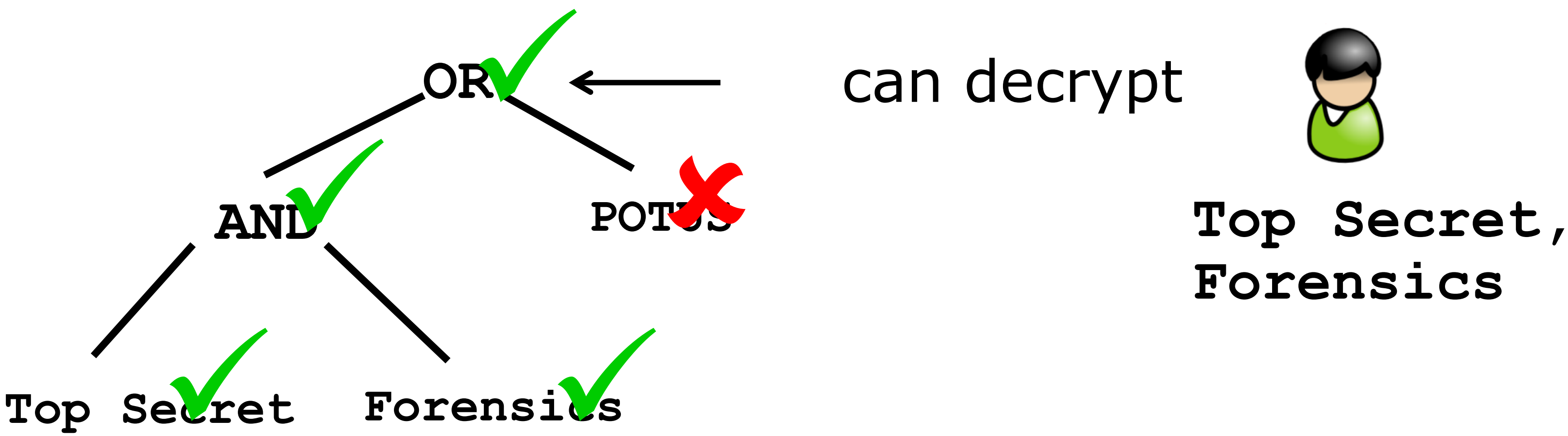
Attribute-Based Encryption



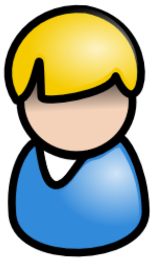
has a message, wants to send it to everyone authorized to receive it.

Encryption takes as input a policy.

$$\text{Encrypt}(PK, M, P) \rightarrow C_P$$



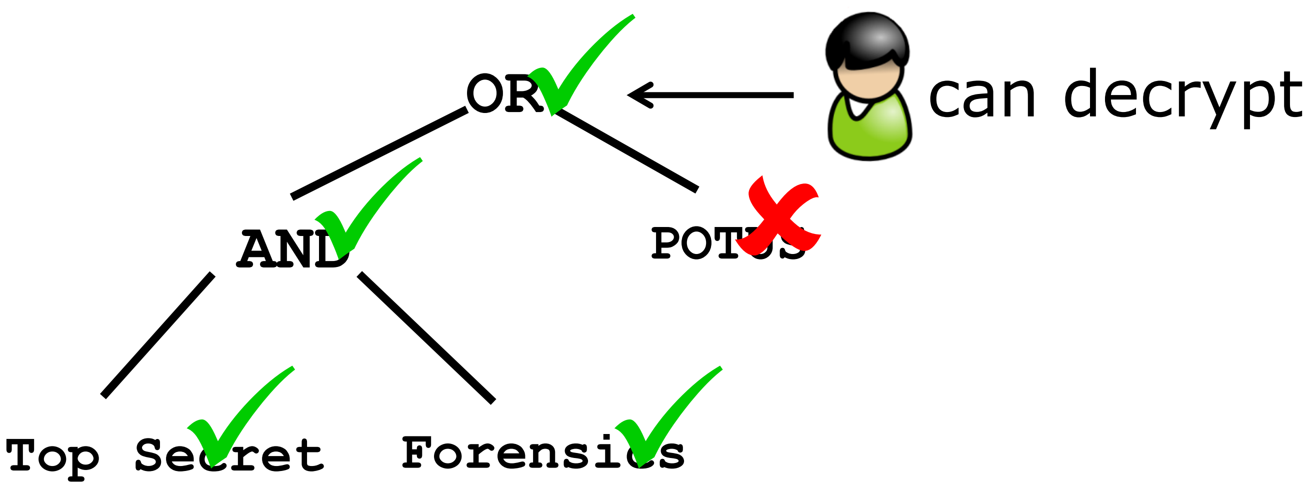
Attribute-Based Encryption



has a message, wants to send it to everyone authorized to receive it.

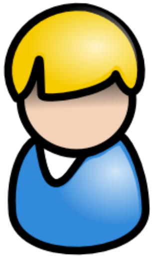
Encryption takes as input a policy.

$$\text{Encrypt}(PK, M, P) \rightarrow C_P$$



Top Secret, Forensics

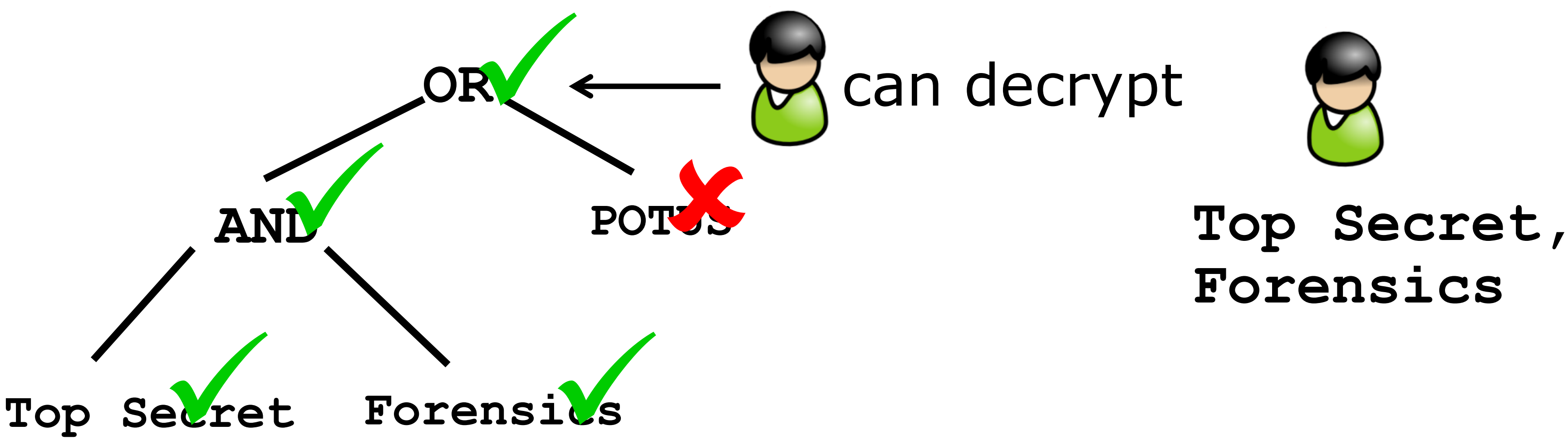
Attribute-Based Encryption



has a message, wants to send it to everyone authorized to receive it.

Encryption takes as input a policy.

$$\mathsf{Encrypt}(PK, M, P) \rightarrow C_P$$



This work: Dynamic Credentials

Users’ credentials change over time

If a user’s credentials change, his old key is revoked and he is issued a new key

This work: Dynamic Credentials

Users' credentials change over time

- If a user's credentials change, his old key is revoked and he is issued a new key

(Usual) Framework to make this possible:

This work: Dynamic Credentials

Users’ credentials change over time

If a user’s credentials change, his old key is revoked and he is issued a new key

(Usual) Framework to make this possible:

- Periodic broadcasts by key authority

This work: Dynamic Credentials

Users' credentials change over time

If a user's credentials change, his old key is revoked and he is issued a new key

(Usual) Framework to make this possible:

- Periodic broadcasts by key authority
- Unrevoked keys can be updated and can decrypt data encrypted at new time

This work: Dynamic Credentials

Users’ credentials change over time

If a user’s credentials change, his old key is revoked and he is issued a new key

(Usual) Framework to make this possible:

- Periodic broadcasts by key authority
- Unrevoked keys can be updated and can decrypt data encrypted at new time

$$\text{Encrypt}(PK, M, P, t)$$

This work: Dynamic Credentials

Users' credentials change over time

If a user's credentials change, his old key is revoked and he is issued a new key

(Usual) Framework to make this possible:

- Periodic broadcasts by key authority
- Unrevoked keys can be updated and can decrypt data encrypted at new time

$$\text{Encrypt}(PK, M, P, t)$$

This work: Dynamic Credentials

Are the security concerns the same as standard revocation?

No: standard revocation is for *broadcast*: you only care about protecting the future

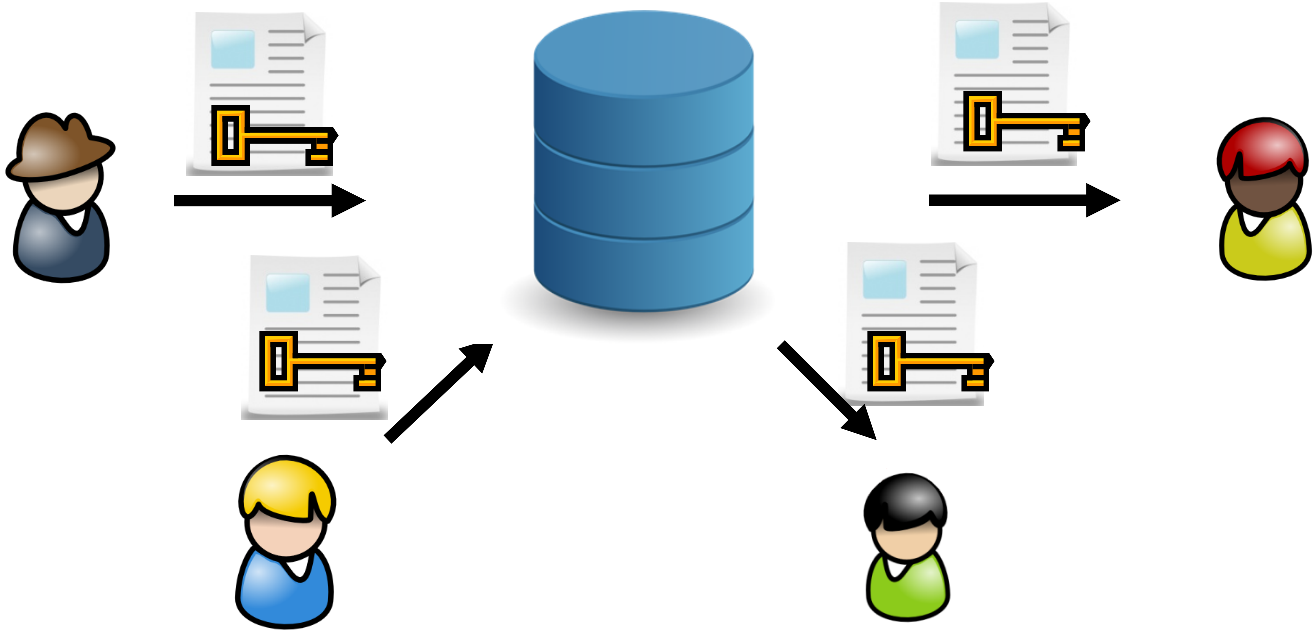
We illustrate with a motivating example:

Inspired by a wonderful conversation with Thomas King and Daniel Manchala (Xerox LA) Our thanks to them for inspiring this work!

Motivation

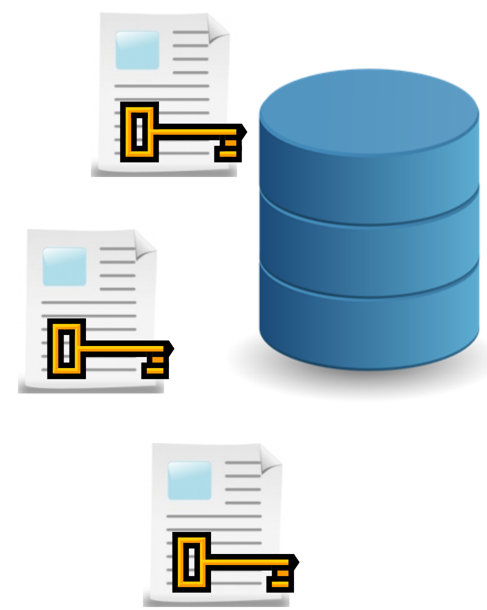
Setting:
Company with ABE based access control

Normally, employee only accesses files he needs (enforced by access logs).



Motivation

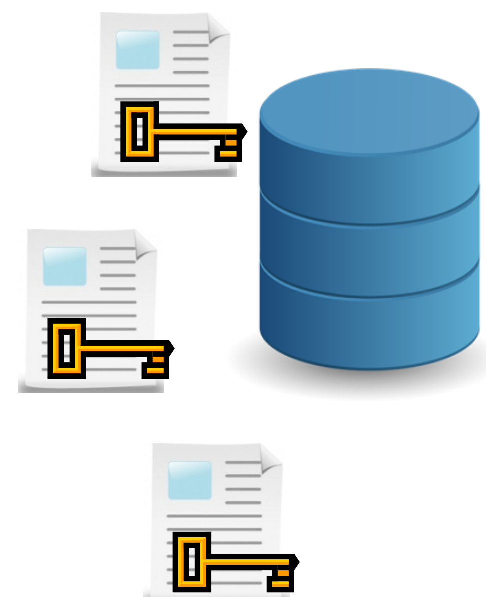
Employee Termination:
Employee’s key is revoked. Standard guarantee: he can’t access files added in the future.



Motivation

Employee Termination:
Employee’s key is revoked. Standard guarantee: he can’t access files added in the future.

Problem: He hacks into server and uses old key to decrypt old files that he didn’t download earlier.



Motivation

Employee Termination:
Employee's key is revoked. Standard guarantee: he can't access files added in the future.

Motivation

Employee Termination:
Employee’s key is revoked. Standard guarantee: he can’t access files added in the future.

Problem: He hacks into server and uses old key to decrypt old files that he didn’t download earlier.

Motivation

Employee Termination:
Employee’s key is revoked. Standard guarantee: he can’t access files added in the future.

Problem: He hacks into server and uses old key to decrypt old files that he didn’t download earlier.

Serious problem: balance between strict security and ease of use:
Necessitates broader access policies,
with countermeasures against misuse of privilege.

Preventing access to old files, even if they match old access policy, is important security concern.

Motivation

What security property do we need?

Motivation

What security property do we need?

After termination, employee should not be able to access anything he doesn't already have.

Motivation

What security property do we need?

After termination, employee should not be able to access anything he doesn't already have.

This breaks down into two guarantees.

Two Security Guarantees

1. Files encrypted in the past.

Two Security Guarantees

1. Files encrypted in the past.

How can we protect old files that the employee could access with his old key in the past?

Looked at only to a limited extent in the past for

Two Security Guarantees

1. Files encrypted in the past.

How can we protect old files that the employee could access with his old key in the past?

First time considered to the best of our knowledge

Looked at only to a limited extent in the past for

Two Security Guarantees

1. Files encrypted in the past.

How can we protect old files that the employee could access with his old key in the past?

First time considered to the best of our knowledge

2. Files added to system in future

Looked at only to a limited extent in the past for

Two Security Guarantees

1. Files encrypted in the past.

How can we protect old files that the employee could access with his old key in the past?

First time considered to the best of our knowledge

2. Files added to system in future

Looked at only to a limited extent in the past for

Two Security Guarantees

1. Files encrypted in the past.

How can we protect old files that the employee could access with his old key in the past?

First time considered to the best of our knowledge

2. Files added to system in future

Looked at only to a limited extent in the past for IBE/ABE [Boldyreva-Goyal-Kumar’08]

Only weak notions of security achieved.

Two Security Guarantees

1. Files encrypted in the past.

How can we protect old files that the employee could access with his old key in the past?

First time considered to the best of our knowledge

2. Files added to system in future

Looked at only to a limited extent in the past for IBE/ABE [Boldyreva-Goyal-Kumar'08]

Only weak notions of security achieved.

Main Result: First ABE scheme to address both of these problems simultaneously.

Two Security Guarantees

- Assume we have security for new files:

  $\text{Encrypt}(PK, M, P, t)$ can only be decrypted by users with secret key for time $\geq t$.

  (e.g., user with credential for time t+2 can decrypt)

- How can we get security for old files?

Solution ideas

Decrypting and Re-encrypting:
Every night, re-encrypt all files on server

$$\mathsf{Encrypt}(PK, M, P, t)$$

Solution ideas

Decrypting and Re-encrypting:
Every night, re-encrypt all files on server

$$\mathsf{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Solution ideas

Decrypting and Re-encrypting:
Every night, re-encrypt all files on server

$$\text{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Decrypt and re-encrypt for time t+1

$$\text{Encrypt}(PK, M, P, t+1)$$

Solution ideas

Decrypting and Re-encrypting:
Every night, re-encrypt all files on server

$$\text{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Decrypt and re-encrypt for time t+1

$$\text{Encrypt}(PK, M, P, t+1)$$

Problem: Maintenance requires master secret key. We do not want to trust the server with this.

Solution ideas

Decrypting and Re-encrypting:
Every night, re-encrypt all files on server

$$\text{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Decrypt and re-encrypt for time t+1

$$\text{Encrypt}(PK, M, P, t+1)$$

Problem: Maintenance requires master secret key. We do not want to trust the server with this.

Solution ideas

Overwrite Encryption:
Every night, re-encrypt all ciphertexts on server

$$\text{Encrypt}(PK, M, P, t)$$

Solution ideas

Overwrite Encryption:
Every night, re-encrypt all ciphertexts on server

$$\mathsf{Encrypt}(PK, M, P, t)$$

Encrypt the ciphertext at time t+1

Solution ideas

Overwrite Encryption:
Every night, re-encrypt all ciphertexts on server

$$\text{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Encrypt the ciphertext at time t+1

Problem: Overhead grows every night

Solution ideas

Overwrite Encryption:
Every night, re-encrypt all ciphertexts on server

$$\mathsf{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Encrypt the ciphertext at time t+1

$$\mathsf{Encrypt}(PK, \mathsf{Encrypt}(PK, M, P, t), P, t+1)$$

Problem: Overhead grows every night

Solution ideas

Overwrite Encryption:
Every night, re-encrypt all ciphertexts on server

$$\mathsf{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Encrypt the ciphertext at time t+1

$$\mathsf{Encrypt}(PK, \mathsf{Encrypt}(PK, M, P, t), P, t+1)$$

Problem: Overhead grows every night

Solution ideas

Overwrite Encryption:
Every night, re-encrypt all ciphertexts on server

We ask:
Can we allow server to “refresh” the encryption
without needing any secret keys,
and without growing the ciphertext?

$$\mathsf{Encrypt}(PK, \mathsf{Encrypt}(PK, M, P, t), P, t+1)$$

Problem: Overhead grows every night

Our Approach

Directly Refreshing Ciphertext:
Increment the time component using public data

$$\mathsf{Encrypt}(PK, M, P, t)$$

Our Approach

Directly Refreshing Ciphertext:
Increment the time component using public data

$$\mathsf{Encrypt}(PK, M, P, t)$$
$$\downarrow$$

We say such a scheme has *<u>Revocable Storage</u>*

Our Approach

Directly Refreshing Ciphertext:
Increment the time component using public data

$$\text{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

$$\text{Encrypt}(PK, M, P, t+1)$$

We say such a scheme has *Revocable Storage*

Our Approach

Directly Refreshing Ciphertext:
Increment the time component using public data

$$\mathsf{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Ciphertext update

$$\mathsf{Encrypt}(PK, M, P, t+1)$$

We say such a scheme has *Revocable Storage*

Our Approach

Directly Refreshing Ciphertext:
Increment the time component using public data

$$\text{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Ciphertext update

$$\text{Encrypt}(PK, M, P, t+1)$$

We

Note: new ciphertext is **more** restrictive than old ciphertext, so security is maintained.

Our Approach

Directly Refreshing Ciphertext:
Increment the time component using public data

$$\text{Encrypt}(PK, M, P, t)$$

$$\downarrow$$

Ciphertext update

$$\text{Encrypt}(PK, M, P, t+1)$$

We

Note: new ciphertext is **more** restrictive than old ciphertext, so security is maintained.

Our Approach

More generally, for standard ABE:

$$\mathsf{Encrypt}(PK, M, P) = C$$

$$\mathsf{Delegate}(PK, C, P') \equiv \mathsf{Encrypt}(PK, M, P')$$

Our Approach

More generally, for standard ABE:

$$\mathsf{Encrypt}(PK, M, P) = C$$

$$\mathsf{Delegate}(PK, C, P') \equiv \mathsf{Encrypt}(PK, M, P')$$

where P' is a more restrictive policy than P.

We call this problem *Ciphertext Delegation.*

Our Approach

More generally, for standard ABE:

$$\mathsf{Encrypt}(PK, M, P) = C$$

$$\mathsf{Delegate}(PK, C, P') \equiv \mathsf{Encrypt}(PK, M, P')$$

where P’ is a more restrictive policy than P.

We call this problem *Ciphertext Delegation.*

Delegation

An example of **ciphertext delegation** in ABE [BSW07]:

Key Generation.

$$MSK = \alpha, \beta \leftarrow \mathbb{Z}_p$$

$$PK = g^{\beta}, g^{1/\beta}, e(g,g)^{\alpha}$$

Delegation

An example of **ciphertext delegation** in ABE [BSW07]:

Key Generation.

$$MSK = \alpha, \beta \leftarrow \mathbb{Z}_p$$

$$PK = g^{\beta}, g^{1/\beta}, e(g,g)^{\alpha}$$

Delegation

An example of **ciphertext delegation** in ABE [BSW07]:

Key Generation.

$$MSK = \alpha, \beta \leftarrow \mathbb{Z}_p$$

$$PK = g^{\beta}, g^{1/\beta}, e(g,g)^{\alpha}$$

Delegation

An example of **ciphertext delegation** in ABE [BSW07]:

Key Generation.

$$MSK = \alpha, \beta \leftarrow \mathbb{Z}_p$$

$$PK = g^{\beta}, g^{1/\beta}, e(g,g)^{\alpha}$$

(Only used in decryption)

Delegation

Encryption.

Take the ciphertext policy:
"Has `top secret (ts.)' and `accounting (ac.)' attributes"

$$s \leftarrow \mathbb{Z}_p$$

$$s = q_{ts.} + q_{ac.}$$

$$C = (Me(g,g)^{\alpha s} \quad ,$$

$$C_{ts.} = g^{q_{ts.}} \quad , \quad C'_{ts.} = H(ts.)^{q_{ts.}} \quad ,$$

$$C_{ac.} = g^{q_{ac.}} \quad , \quad C'_{ac.} = H(ac.)^{q_{ac.}})$$

Delegation

Can we delegate this to the policy:
"Has attributes `top secret (ts.)' and `accounting (ac.)' and `director (dir.)' "

We are given the ciphertext:

$$Me(g,g)^{\alpha s}$$

$$C_{ts.} = g^{q_{ts.}} \quad , \quad C'_{ts.} = H(ts.)^{q_{ts.}} \quad ,$$

$$C_{ac.} = g^{q_{ac.}} \quad , \quad C'_{ac.} = H(ac.)^{q_{ac.}}$$

where: $s = q_{ts.} + q_{ac.}$

and the public key: $g^{\beta}, g^{1/\beta}, e(g,g)^{\alpha}$

Delegation

Generate: $q_{dir.} \leftarrow \mathbb{Z}_p$

$$C = (Me(g,g)^{\alpha s} e(g,g)^{\alpha q_{dir.}} = Me(g,g)^{\alpha(s+q_{dir.})}$$

$$C_{ts.} = g^{q_{ts.}} \quad , \quad C'_{ts.} = H(ts.)^{q_{ts.}} \quad ,$$
$$C_{ac.} = g^{q_{ac.}} \quad , \quad C'_{ac.} = H(ac.)^{q_{ac.}}$$
$$C_{dir.} = g^{q_{dir.}} \quad , \quad C'_{dir.} = H(dir.)^{q_{dir.}})$$

Why is this a good ciphertext? $s' = s + q_{dir.}$

Plus: Use re-randomization to prevent subtle attacks.

Types of Delegation

We show most current ABE schemes support a variety of efficient ciphertext delegation ops:

- Increasing node thresholds
- Increasing node thresholds and adding nodes
- Deleting subtrees

We also conduct survey of delegation operations on LSSS matrix based schemes [GPSW06, Waters11, LOSTW10].

Conclusion

1. We define ciphertext delegation and give a number of efficient methods for ciphertext delegation.
2. We use ciphertext delegation to solve the problem of revocable storage.
3. We also construct fully secure ABE schemes that achieve revocation security vs. future encryptions.
4. We show how to combine these elements to achieve the first fully secure ABE schemes for dynamic credentials.