STRĀTUMN

CRYPTO NIGHTS

CP-ABE

Ciphertext-Policy Attribute-Based Encryption

MOURAD BEJI

–

07.11.2018

Intro

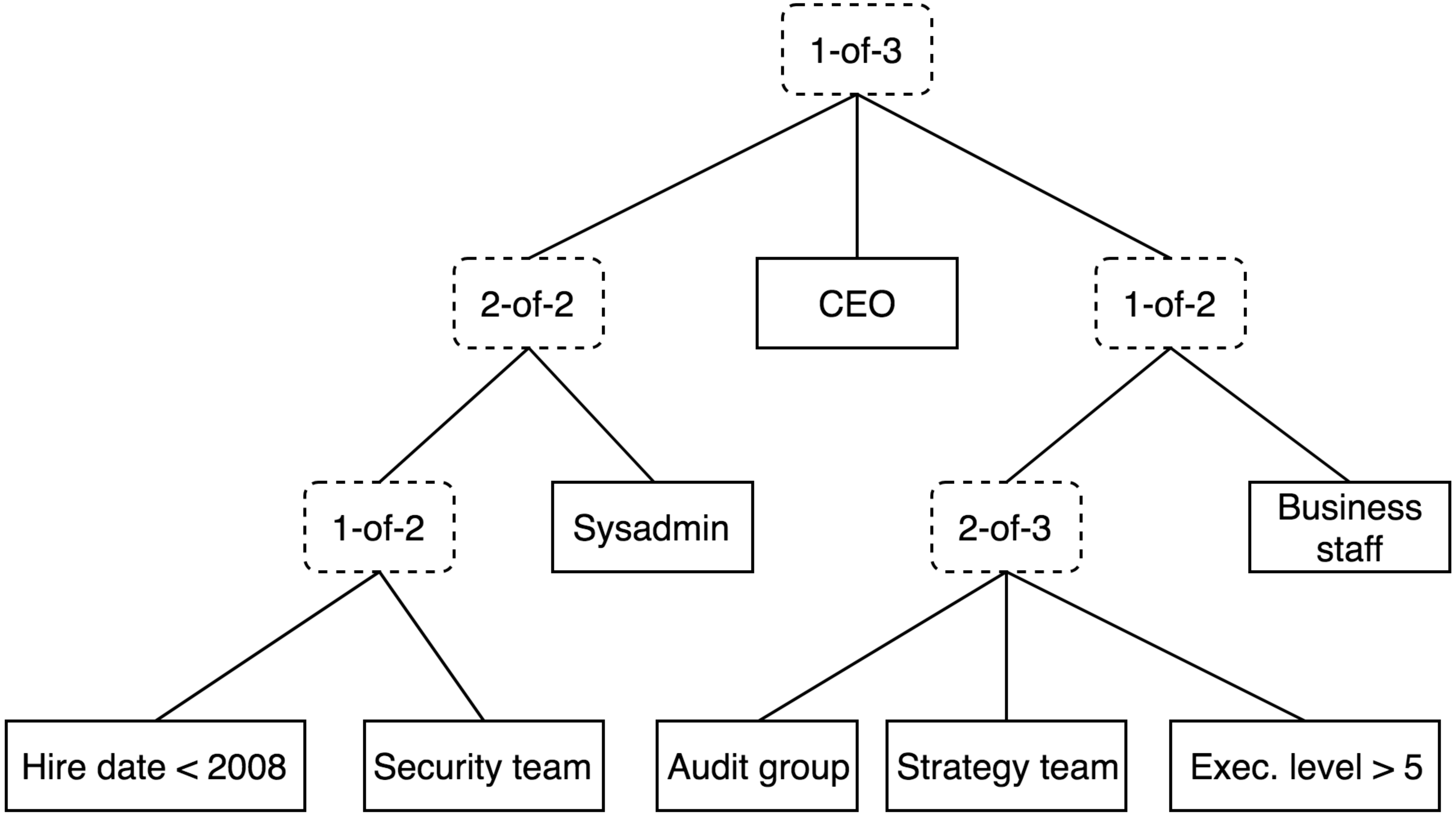
- define fine-grain access policies for data encryption and recovery
- usually the server that has the data controls the access to it
  - But more and more data is distributed (cloud) => untrusted/semi-trusted servers, higher chances of compromising data
  - p2p networks: no centralized server after system setup

We don't want to encrypt the file with an exhaustive list of all possible recipients

<u>Solution</u>: access control should be inherent to the cryptographic scheme

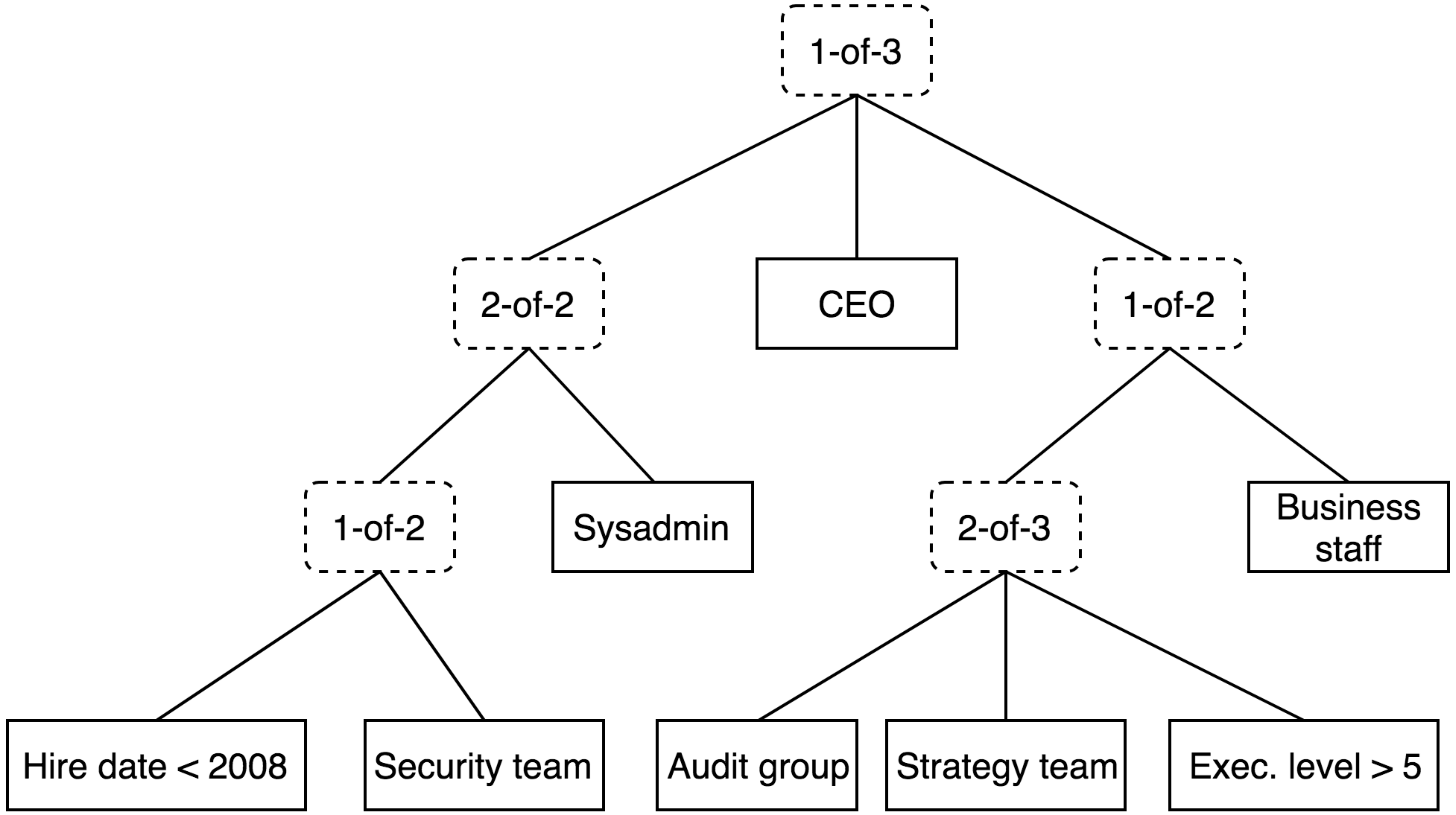
Access structure

Access structure



Access structure

((Hire date < 2008 **OR** security team) **AND** Sysadmin) **OR**
CEO **OR**
(**2 OF** (Executive level ≥ 5, Audit group, Strategy team) **AND** Business staff)



CP-ABE

A. Sahai, B. Waters

CP-ABE

- Each user has a secret key = f(attributes)
- Data is encrypted using an policy = access structure
- Descryption is possible if attributes verify the policy tree

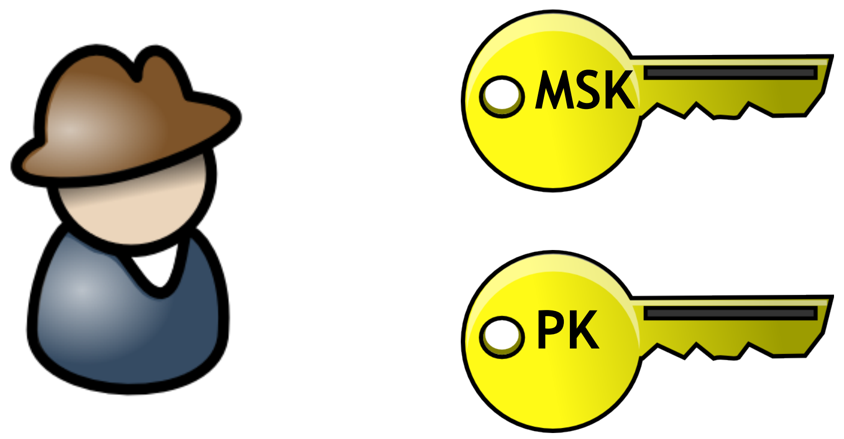
CP-ABE Algorithms

- Setup: $1^\lambda \to (PK, MSK)$
- KeyGen: $(MSK, \mathcal{S}) \to SK$
- Encrypt: $(PK,\ plaintext,\ \mathbb{A}) \to ciphertext$
- Decrypt: $(PK,\ ciphertext,\ SK) \to \begin{cases} plaintext \text{ if } \mathcal{S} \text{ verifies } \mathbb{A} \\ \perp \text{ otherwise} \end{cases}$

CP-ABE Algorithms

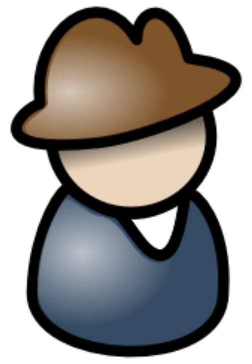
- Setup: $1^\lambda \to (PK, MSK)$
- KeyGen: $(MSK, \mathcal{S}) \to SK$
- Encrypt: $(PK, plaintext, \mathbb{A}) \to ciphertext$
- Decrypt: $(PK, ciphertext, SK) \to \begin{cases} plaintext \text{ if } \mathcal{S} \text{ verifies } \mathbb{A} \\ \perp \text{ otherwise} \end{cases}$
- Delegate $(SK, \tilde{\mathcal{S}}) \to \tilde{SK} \quad if \quad \tilde{\mathcal{S}} \subset \mathcal{S}$

CP-ABE Protocol - Setup

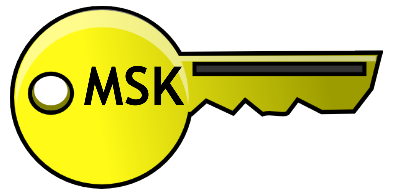


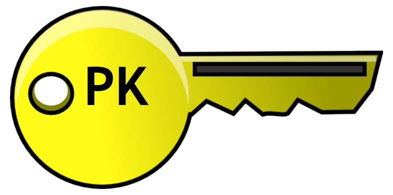
Key issuer

CP-ABE Protocol - Setup

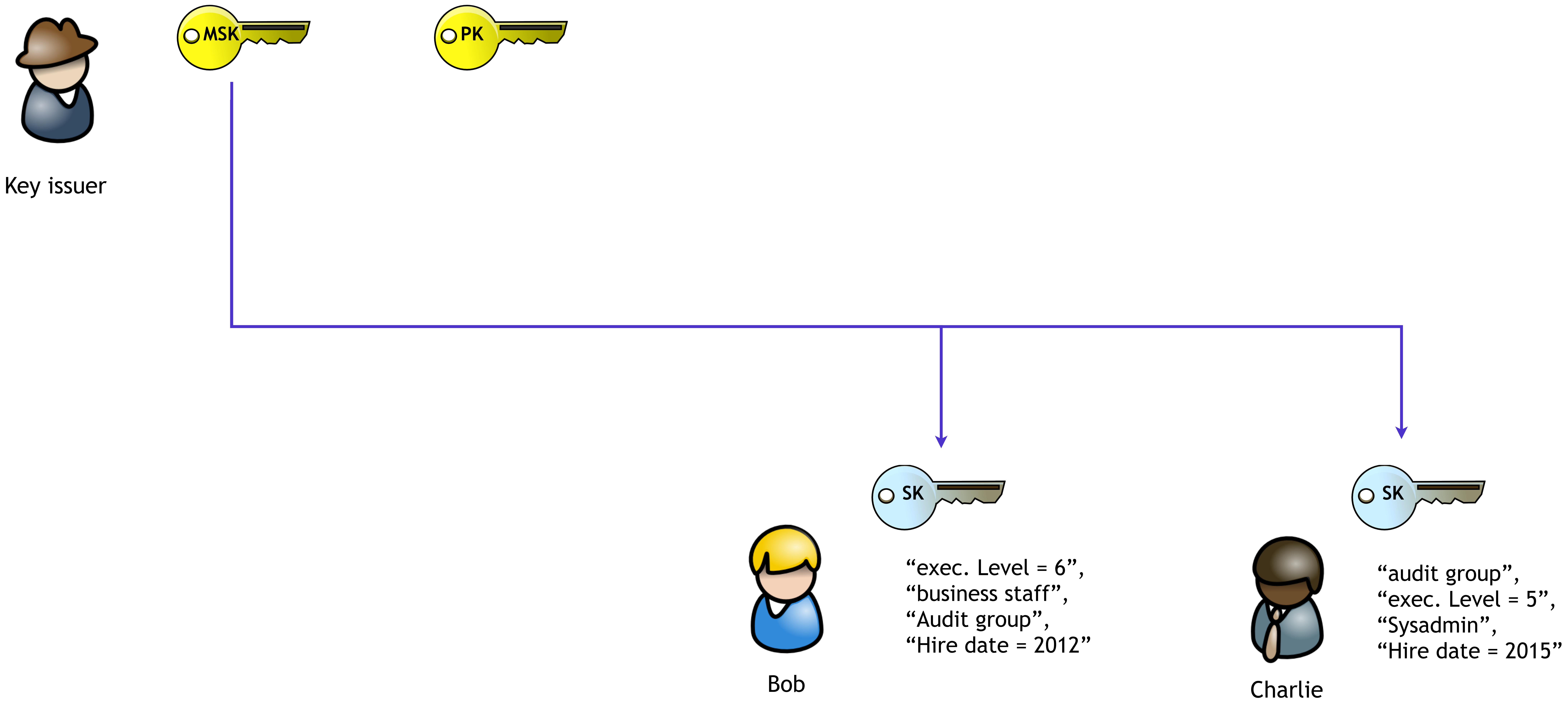


Key issuer

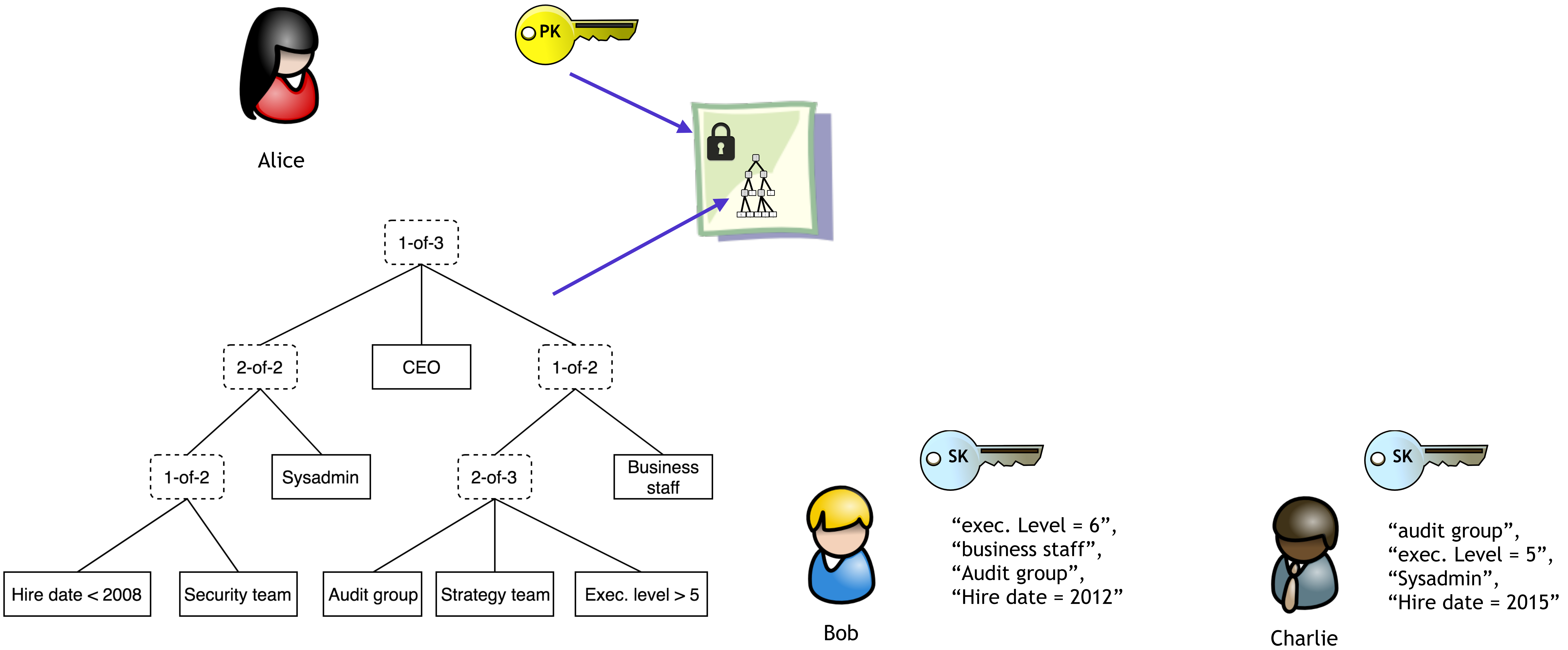




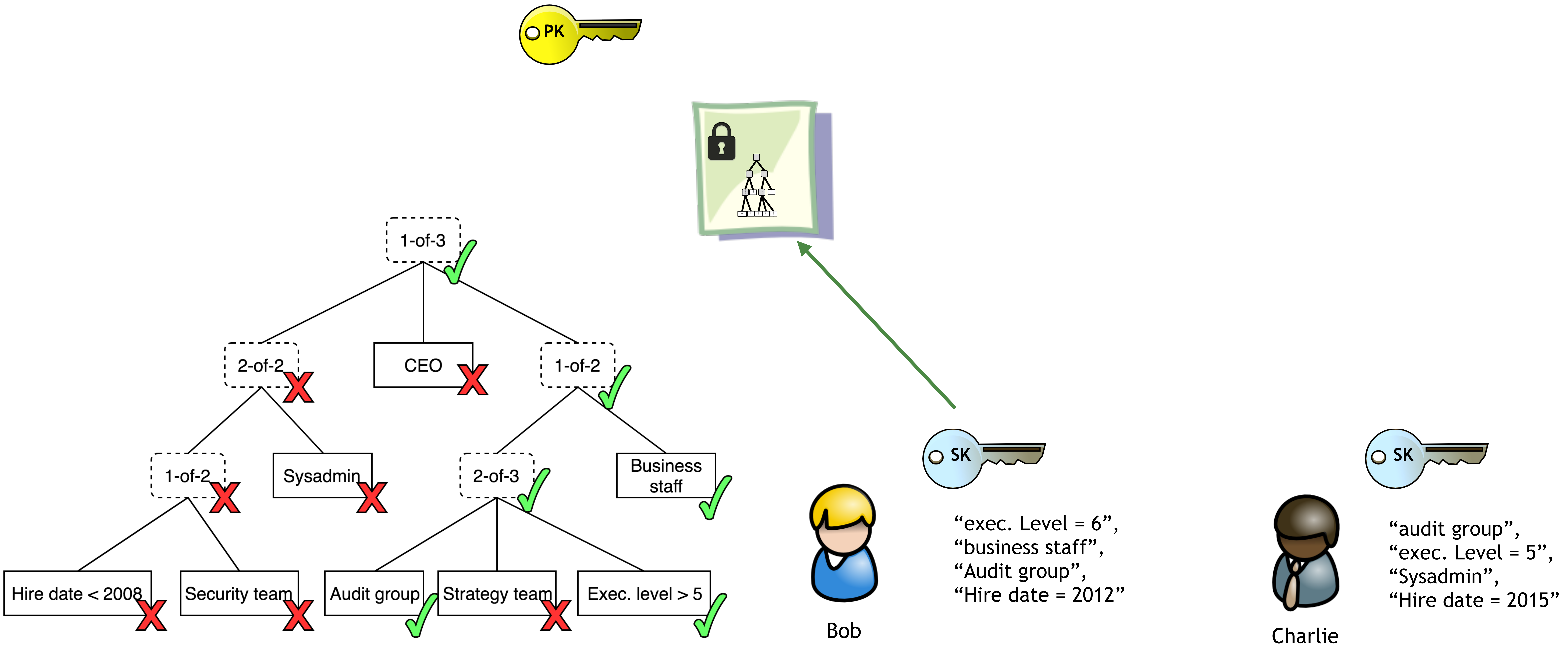
CP-ABE Protocol - Key Generation



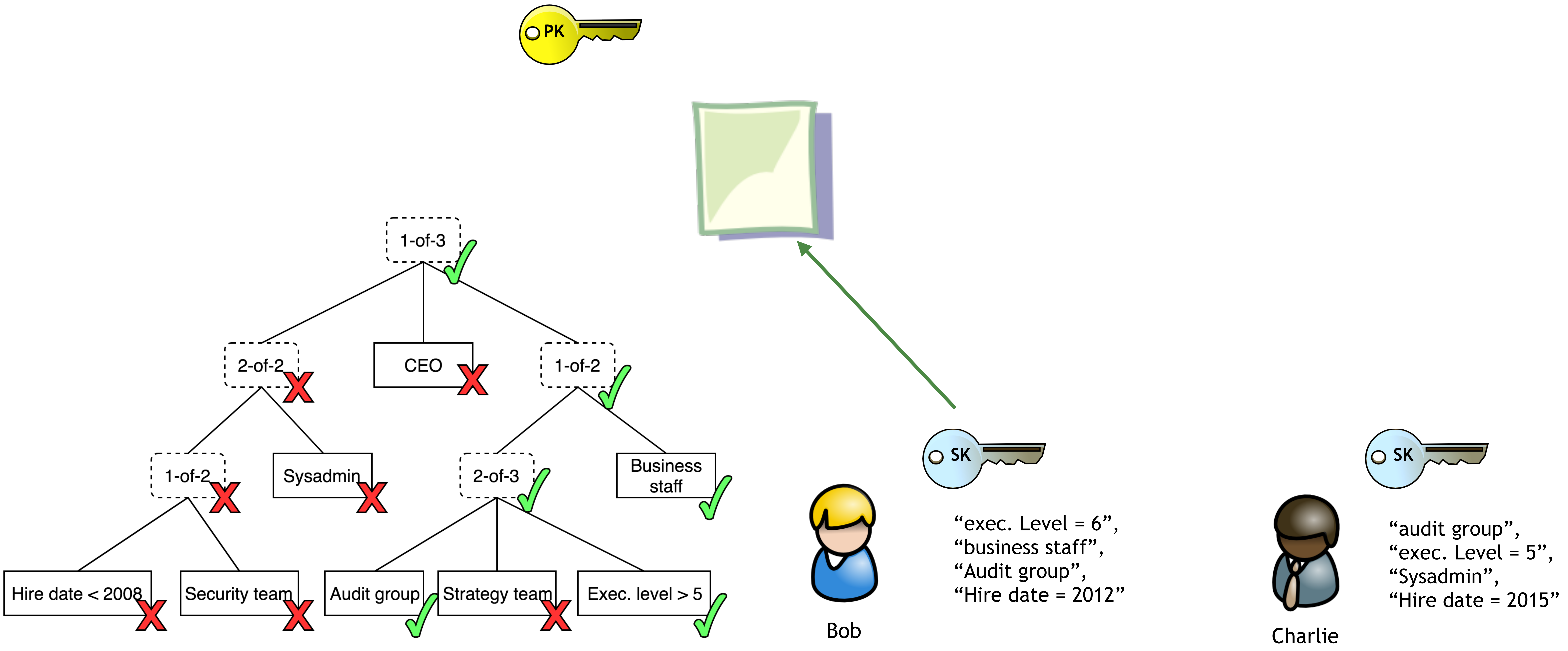
CP-ABE Protocol - Encryption



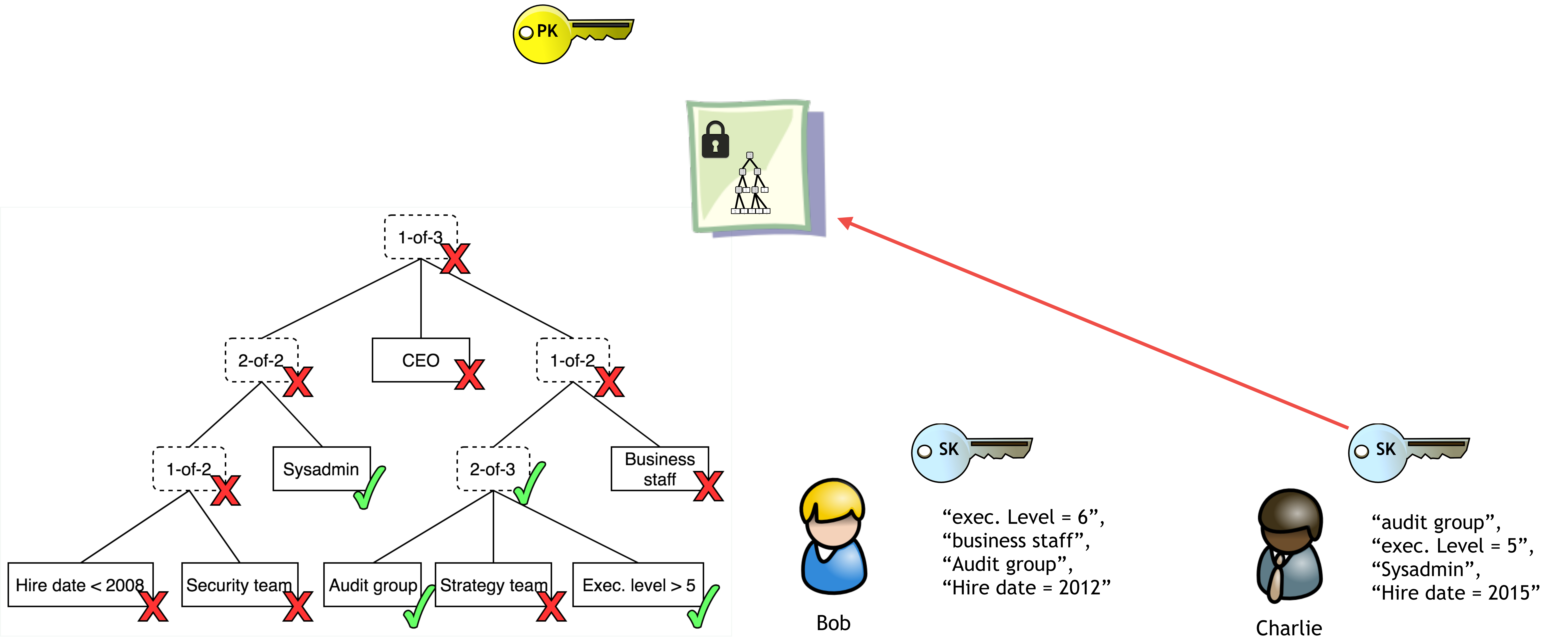
CP-ABE Protocol - Decryption



CP-ABE Protocol - Decryption



CP-ABE Protocol - Decryption



Cryptography

- CP-ABE is based on Pairing Based Cryptography on elliptic curves
- Chosen Ciphertext Attack secure
- Collusion resistance:
  If multiple users collude, they should be able to decrypt a ciphertext if and only if at least one of the users could decrypt it on his own.
  solution: randomizing users secret keys so that they cannot be combined.

Implementation

OpenABE project by Zeutro, written in C++

https://www.zeutro.com

https://github.com/zeutro/openabe

```
1. mourad@Fortress: /tmp/test (zsh)
➜  test echo The plaintext data > file.txt
➜  test oabe_setup -s CP
writing 497 bytes to mpk.cpabe
writing 149 bytes to msk.cpabe
➜  test oabe_keygen -s CP -i "business_team | hire_date = 2012 | exec_level = 42" -o bob
functional key input: business_team | hire_date = 2012 | exec_level = 42
➜  test oabe_keygen -s CP -i "exec_level = 50 | hire_date = 2007 | security_team" -o charlie
functional key input: exec_level = 50 | hire_date = 2007 | security_team
➜  test oabe_enc -s CP -i file.txt -o ciphertext -e "((hire_date < 2008 OR business_team) AND exec_level > 9000) OR security_team"
input file: file.txt
encryption functional input: ((hire_date < 2008 OR business_team) AND exec_level > 9000) OR security_team
➜  test oabe_dec -s CP  -k bob.key -i ciphertext.cpabe -o bob_dec.txt
ciphertext: ciphertext.cpabe
user's SK file: bob.key
abe/zcontextcca.cpp:decrypt:613: 'Error occurred during decryption'
caught exception: Error occurred during decryption
➜  test oabe_dec -s CP  -k charlie.key -i ciphertext.cpabe -o charlie_dec.txt
ciphertext: ciphertext.cpabe
user's SK file: charlie.key
➜  test cat charlie_dec.txt
The plaintext data
```

Performance

- KeyGen is linear in number of attributes associated with the key it is issuing
- Encryption is linear in number of leaf nodes in the policy tree
- Decryption perf depends on the particular policy tree shape, but roughly linear in the number of leaf nodes of the policy

Drawbacks

Revocation is complex:

- Many users have the same attributes, how do we exclude one of them ?
- Use expiration dates in the keys => continuous update of the SKs, not so fine-grained revocation and non retroactive revocation.

Drawbacks

Revocation is complex:

- Many users have the same attributes, how do we exclude one of them ?
- Use expiration dates in the keys => continuous update of the SKs, not so fine-grained revocation and non retroactive revocation.

Centralized key issuer(s):

- Need to be trusted with full power
- Need to protect the master key

Drawbacks

Revocation is complex:

- Many users have the same attributes, how do we exclude one of them ?
- Use expiration dates in the keys => continuous update of the SKs, not so fine-grained revocation and non retroactive revocation.

Centralized key issuer(s):

- Need to be trusted with full power
- Need to protect the master key

Based on discrete log => not quantum secure

Variants

Different types of ABE

- Ciphertext-Policy ABE: keys are bound to a set of attributes and the file is encrypted following an access policy
  Role based access control

- Key-Policy ABE: file has a set of attributes attached to it, keys are generated using an access policy
  Content based access control

KP vs CP: the intelligence (the one who actually defined the policy) is assumed to be with the key issuer vs encryptor

Different types of ABE

- Multi-Authority ABE: Different authorities, each one is responsible for a subset of all attributes
- IR-CP-ABE: Identity revokable CP-ABE (W. Wang et al. - 2017)
  - KeyGen: $(MSK, \mathcal{S}, ID) \rightarrow SK$
  - Encrypt: $(PK, plaintext, \mathbb{A}, \{ID_j\}) \rightarrow ciphertext$
    where $\{ID_j\}$ is the set of revoked IDs

Functional Encryption

generalization of public-key encryption in which possessing a secret key allows one to learn a function of what the ciphertext is encrypting.

- Setup: $1^\lambda \to (MK, PK)$
- KeyGen: $(MK, k) \to SK,$ where $k \in K$ the key space
- Enc: $(PK, m) \to c$
- Dec: $(SK, c) \to F(k, m)$ where F defined over (K, M)

Functional Encryption

Dec: $(SK,\ c) \rightarrow F(k, m)$ where F defined over (K, M)

$K = \{0,1\}^n$ n bit strings representing n boolean variables $\vec{z} = (z_1, \ldots, z_n)$

$M = (I, X)$ where $I =$ set of all poly-sized bool formulas over n variables

$i.e.m = (\phi,\ msg)$

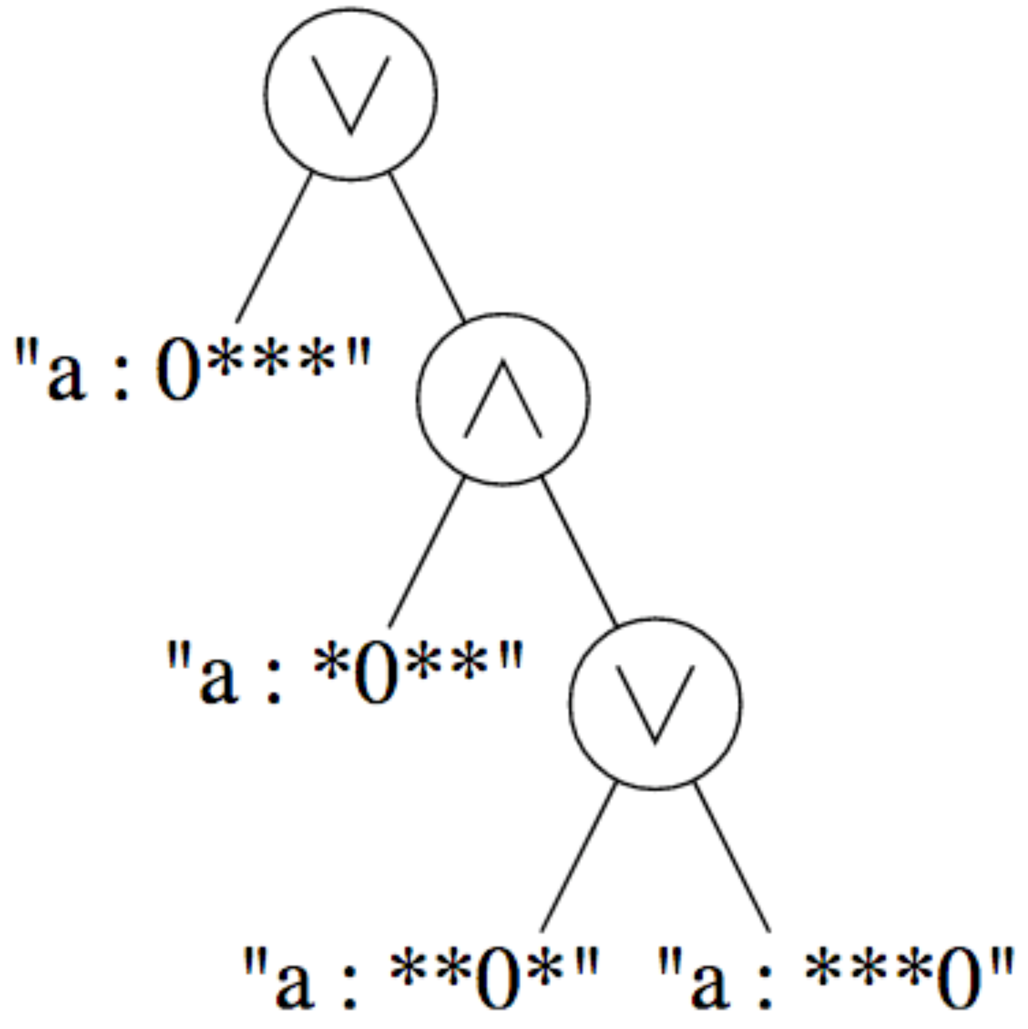
$$F(k, (\phi, msg)) = \begin{cases} m & if\ \phi(k) = 1 \\ \perp & otherwise \end{cases}$$

Questions ?

Appendices

Appendix I: Access structure

Policy tree implementing integer comparison "a < 11"



Appendix II: Cryptography

Setup

For a bilinear group $\mathbb{G}$ of prime order *p*, generator *g* and pairing *e* and a randomly chosen $\alpha, \beta \in \mathbb{Z}_p$

$$\begin{cases} MPK = \mathbb{G},\ g,\ h = g^{\beta},\ f = g^{1/\beta},\ e(g,g)^{\alpha} \\ MSK = \ \beta,\ g^{\alpha} \end{cases}$$

Appendix II: Cryptography

Key Generation

Pick random, $r \in \mathbb{Z}_p,\ r_j \in \mathbb{Z}_p,\ \forall j \in \mathcal{S}$

$$SK = g^{(\alpha+r)/\beta}, \forall j \in \mathcal{S} : g^r \cdot H(j)^{r_j}, g^{r_j}$$

Appendix II: Cryptography

Encryption

To encrypt a message M following an access tree $\mathcal{T}$,
for each node $x \in \mathcal{T}$, we create $q_x \in \mathbb{Z}_p[X]$ recursively s.t.

- $d_x = deg(q_x) = k_x - 1$ where k_x is the threshold value of the gate at node x
- $q_{root}(0) = s \in \mathbb{Z}_p$ chosen randomly, and set d_{root} other points randomly
- $for\ x \neq root,\ q_x(0) = q_{parent(x)}(index(x))$ and set d_x other points randomly

$$\boxed{ciphertext = (\mathcal{T},\ M \cdot e(g,g)^{\alpha s},\ h^s,\ \forall y \in Y : g^{q_y(0)}, H(att(y))^{q_y(0)})}$$

where Y is the set of leafs of the tree (the actual attributes)

Appendix II: Cryptography

Decryption

To decrypt a ciphertext $CT = (\mathcal{T}, \tilde{C}, C, \forall y \in Y : C_y, C'_y)$
with a secret key $SK = (d, \forall j \in \mathcal{S} : D_j, D'_j)$
we define recursively $DecryptNode(CT, SK, x)$

- if x leaf, $DecryptNode(CT, SK, x) = \begin{cases} \dfrac{e(D_{att(x)}, C_x)}{e(D'_{att(x)}, C'_x)} = \boxed{e(g,g)^{rq_x(0)}} & if\ att(x) \in \mathcal{S} \\ \perp \quad if\ att(x) \notin \mathcal{S} \end{cases}$

- Then for $x \in \mathcal{T}$, if we have a k_x-sized subset $\mathcal{S}_x$ of x's children that verify $F_z = DecryptNode(CT, SK, z) \neq \perp\ for\ z \in \mathcal{S}_x$

$$DecryptNode(CT, SK, x) = \prod_{z \in \mathcal{S}_x} F_z^{\Delta_{i,\mathcal{S}'_x}(0)} = e(g,g)^{rq_x(0)}$$

- if we go back to the root, we find $F_{root} = e(g,g)^{rs}$

$$\boxed{M = \tilde{C}/(e(C,D)/F_{root})} = \tilde{C}/\left(e\left(h^s, g^{(\alpha+r)/\beta}\right)/e(g,g)^{rs}\right)$$