Registered Attribute-Based Encryption

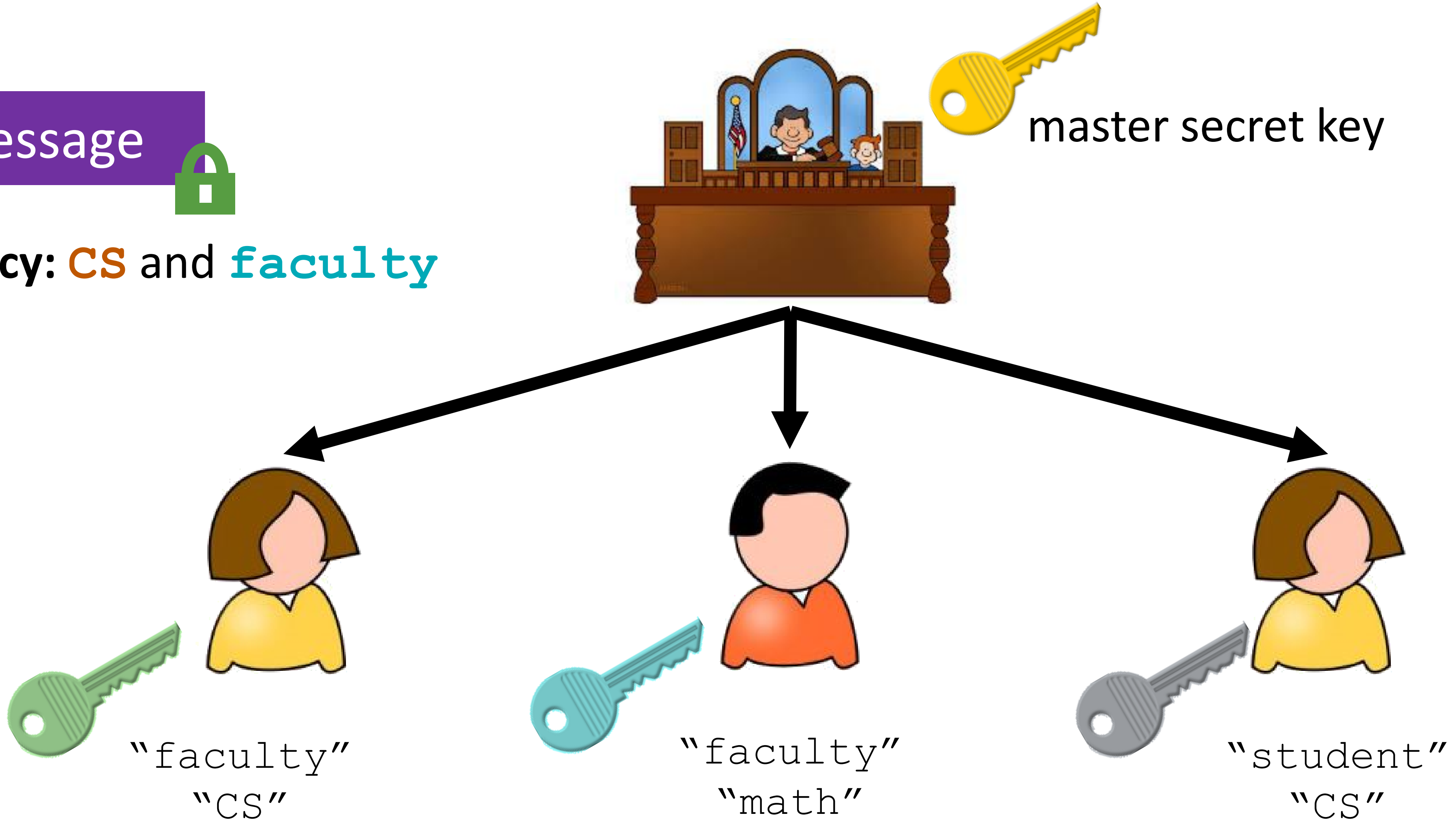
Susan Hohenberger, George Lu, Brent Waters, and David Wu

Attribute-Based Encryption

[SW05, GPSW06]

message

policy: CS and faculty

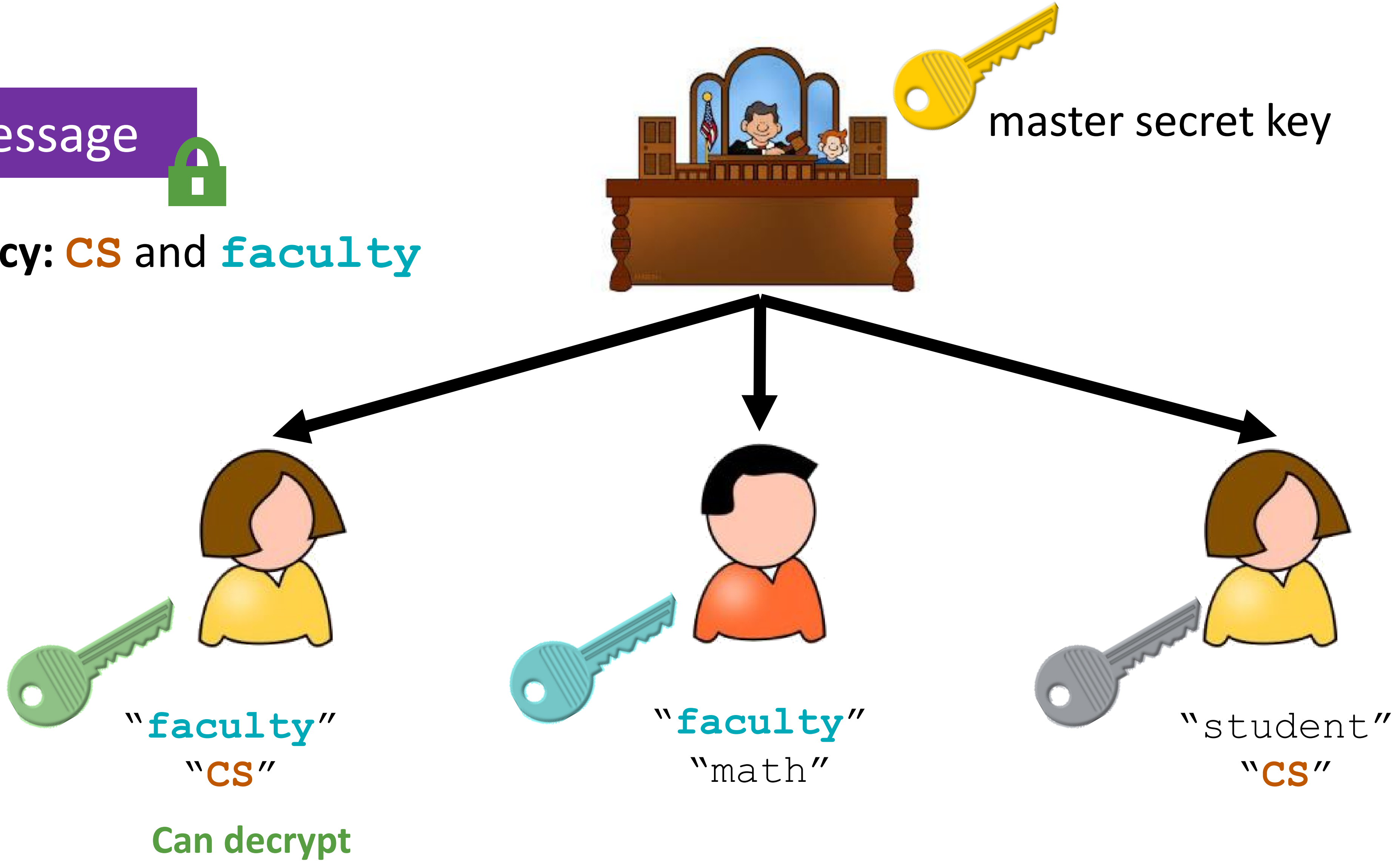


Attribute-Based Encryption

[SW05, GPSW06]

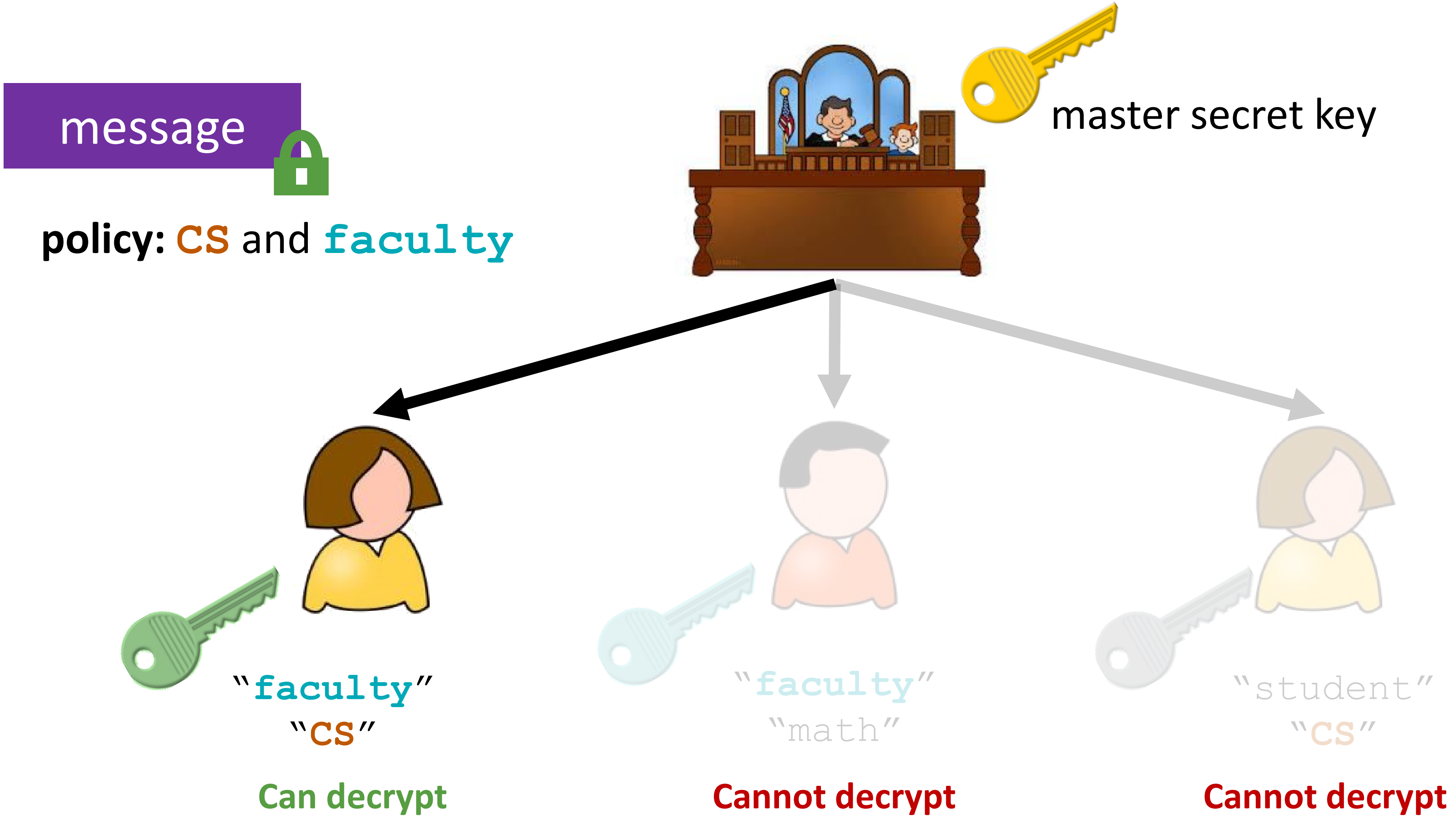
message

policy: CS and faculty



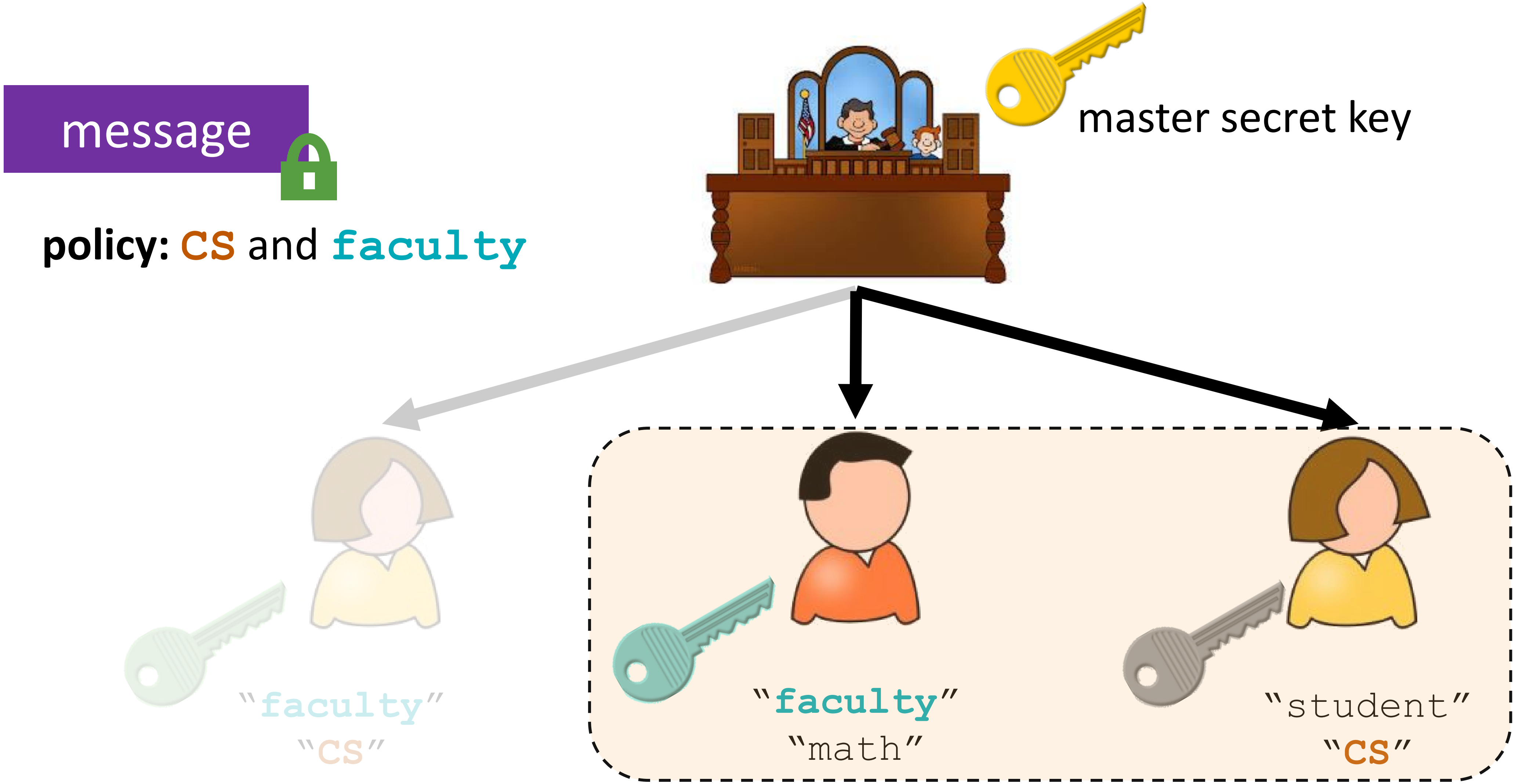
Attribute-Based Encryption

[SW05, GPSW06]



Attribute-Based Encryption

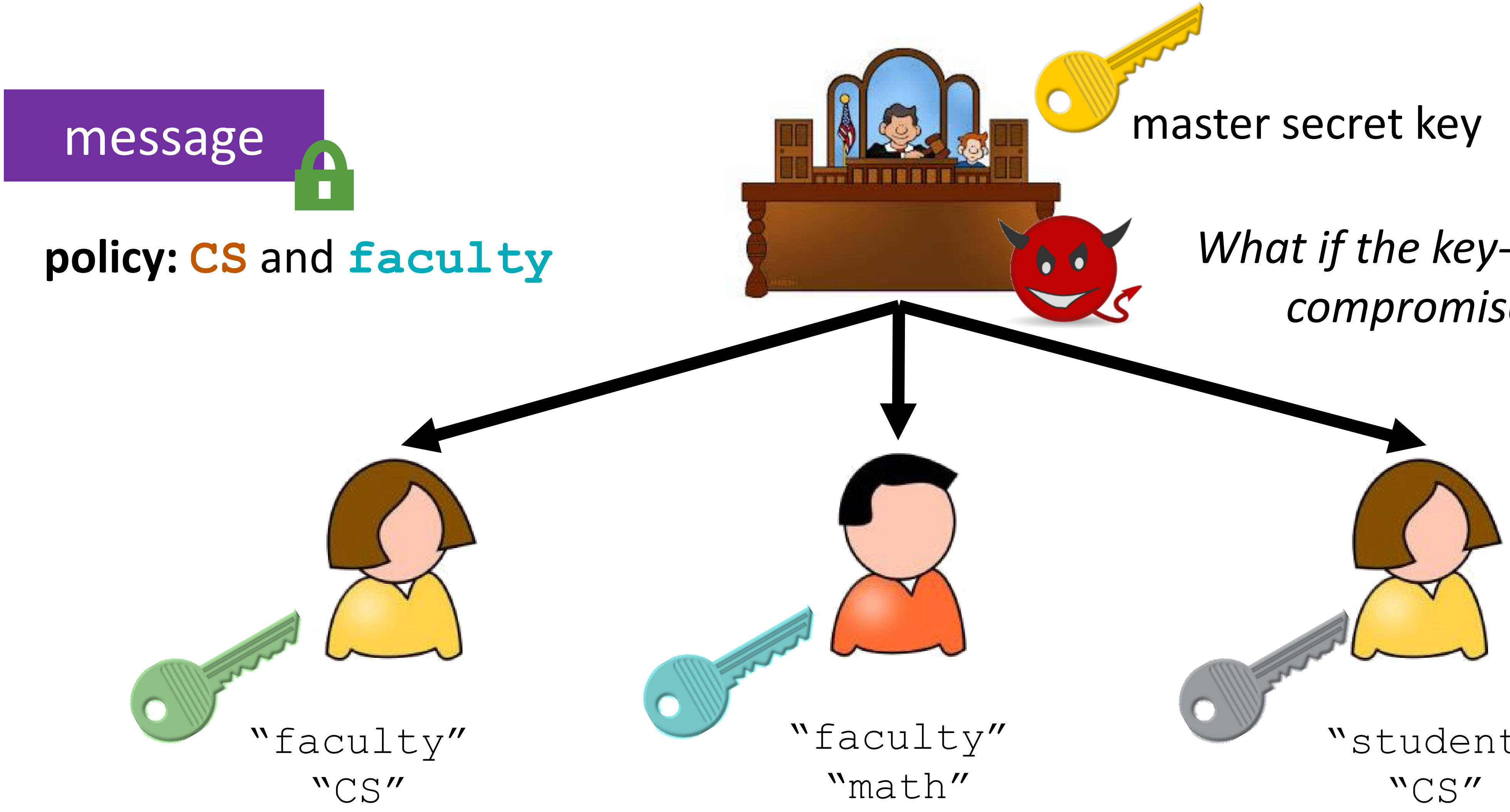
[SW05, GPSW06]



Users cannot collude to decrypt

Attribute-Based Encryption

[SW05, GPSW06]



What if the key-issuer is compromised?

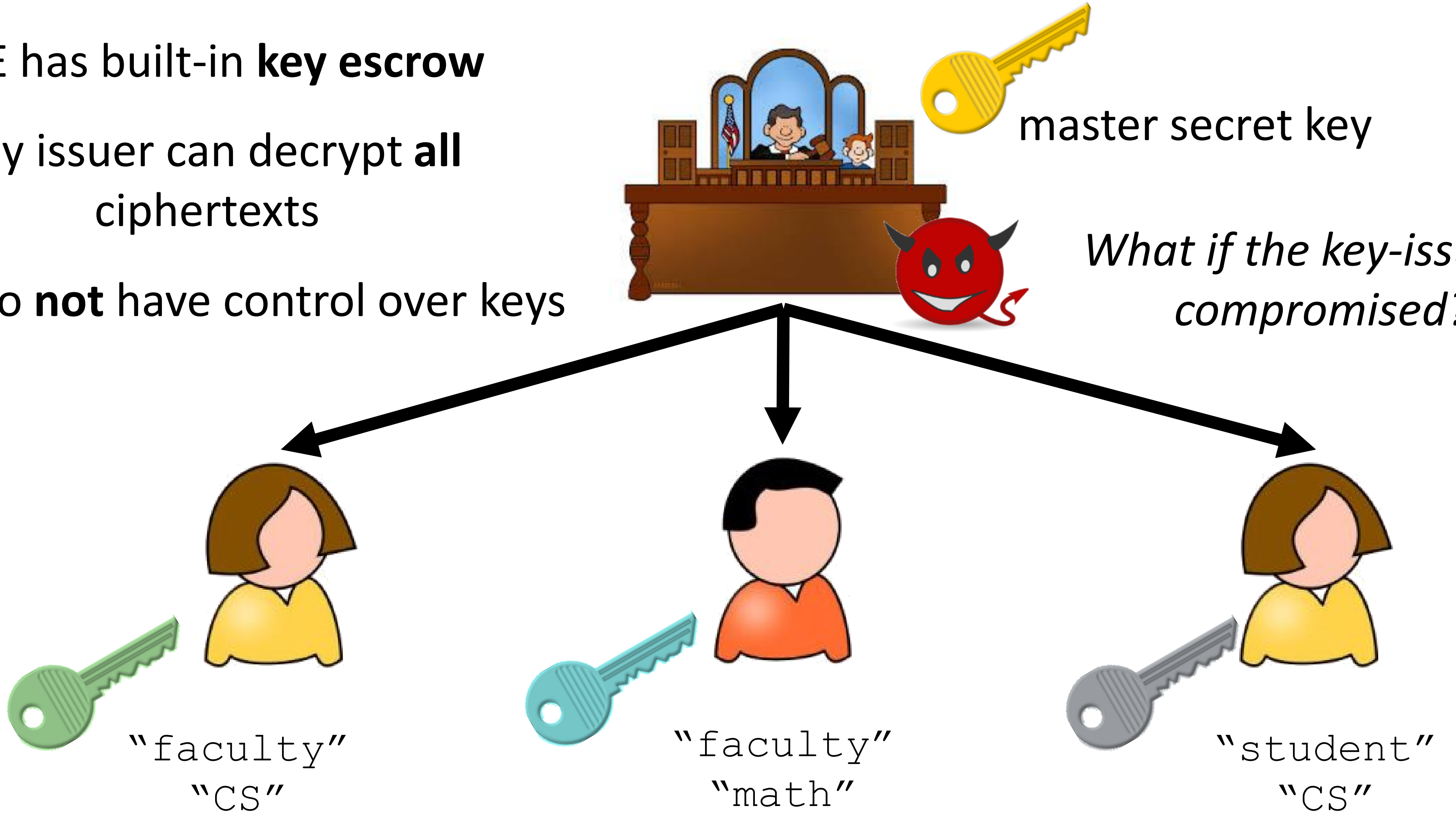
Attribute-Based Encryption

[SW05, GPSW06]

ABE has built-in **key escrow**

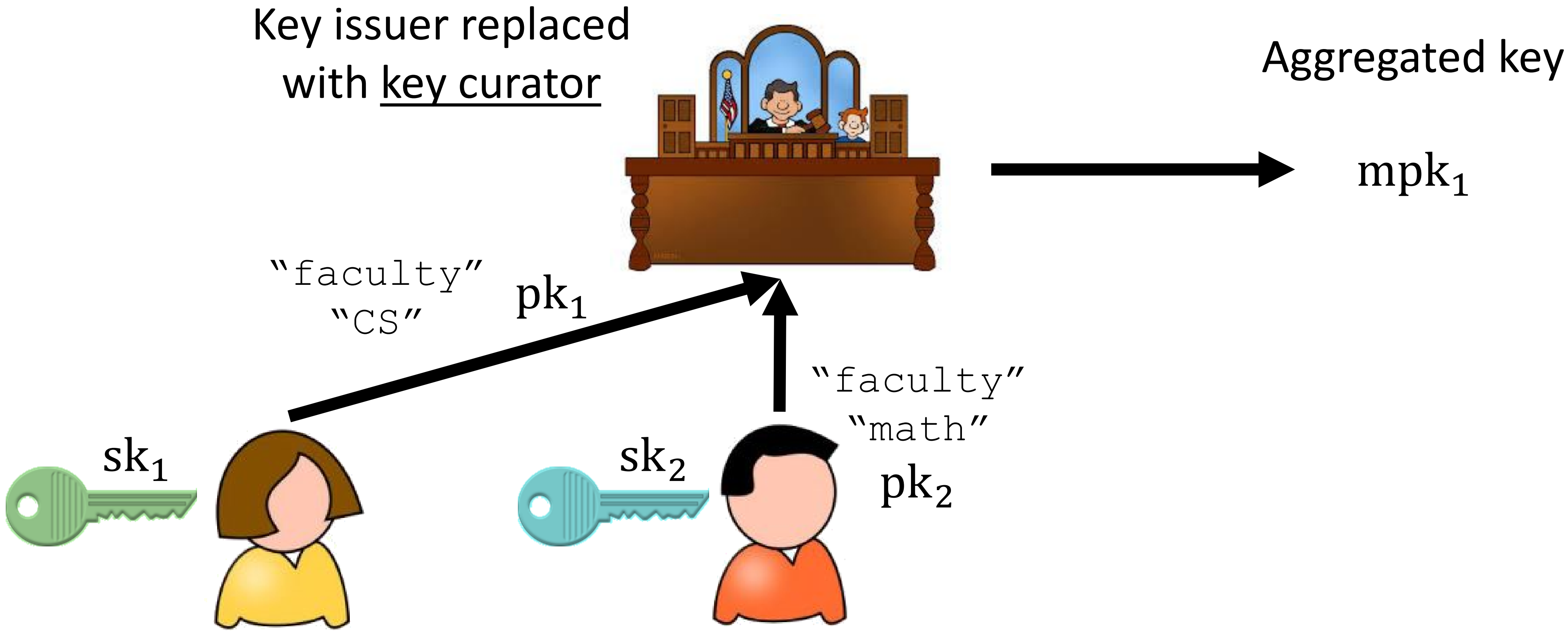
Key issuer can decrypt **all** ciphertexts

Users do **not** have control over keys



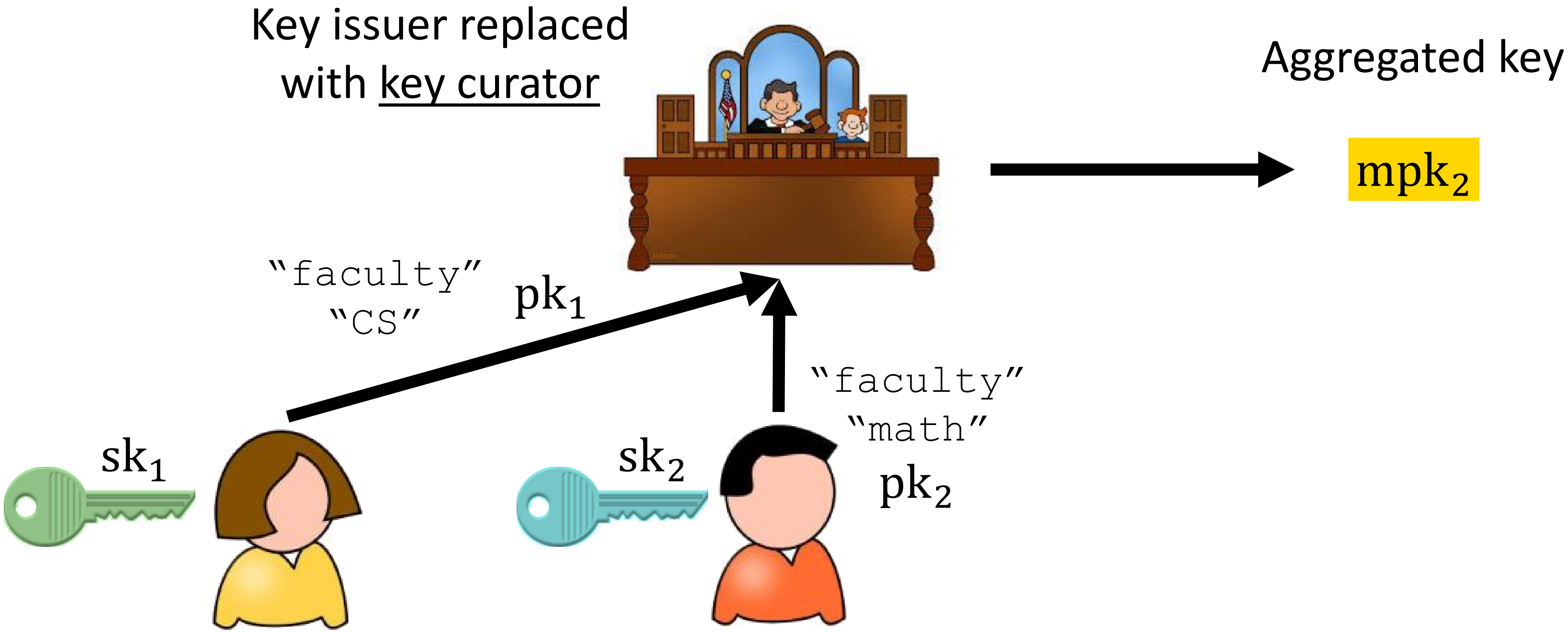
What if the key-issuer is compromised?

Registered ABE



Users chooses their <u>own</u> public/secret key

Registered ABE



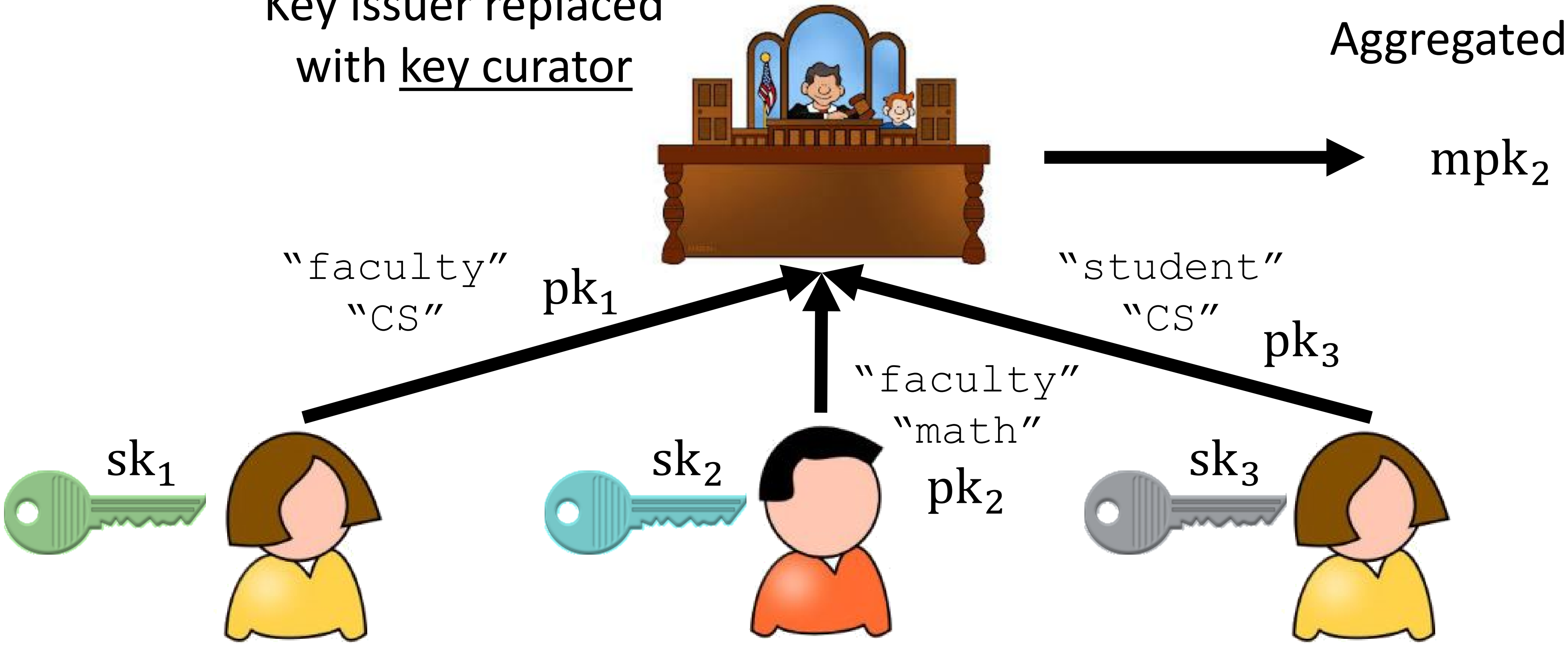
Users chooses their own public/secret key

As users join the system, public key is updated

Registered ABE

Key issuer replaced with key curator

Aggregated key



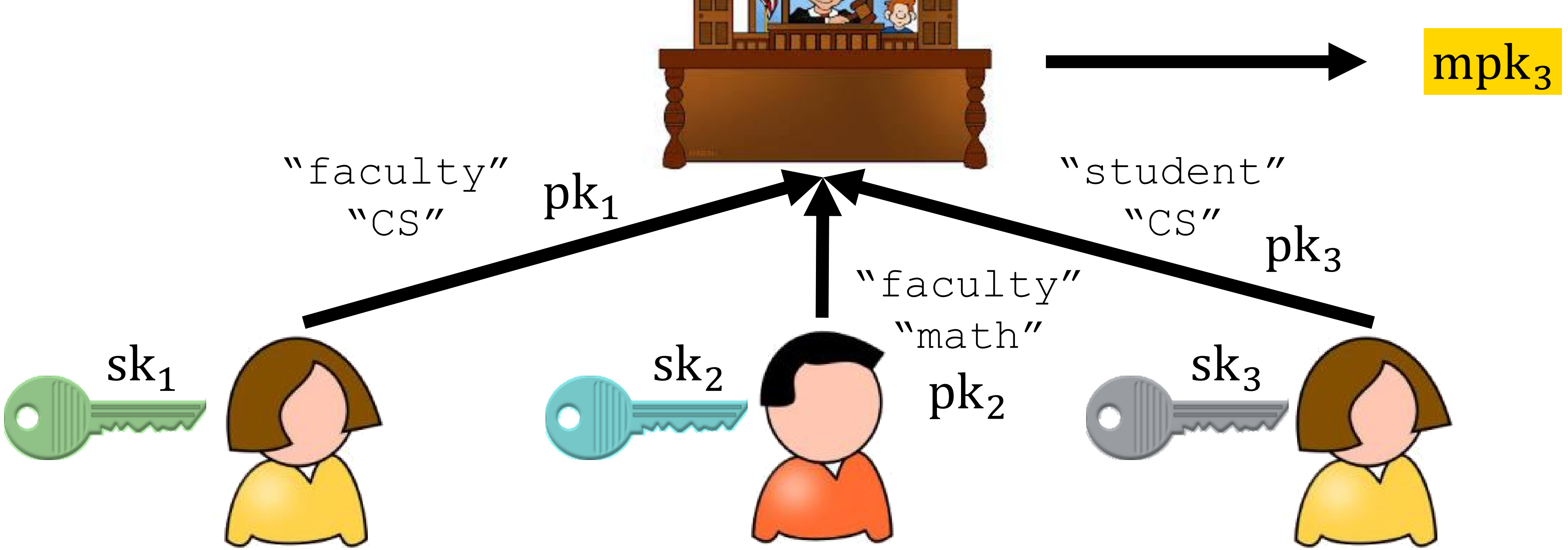
Users chooses their own public/secret key

As users join the system, public key is updated

Registered ABE

Key issuer replaced with key curator

Aggregated key



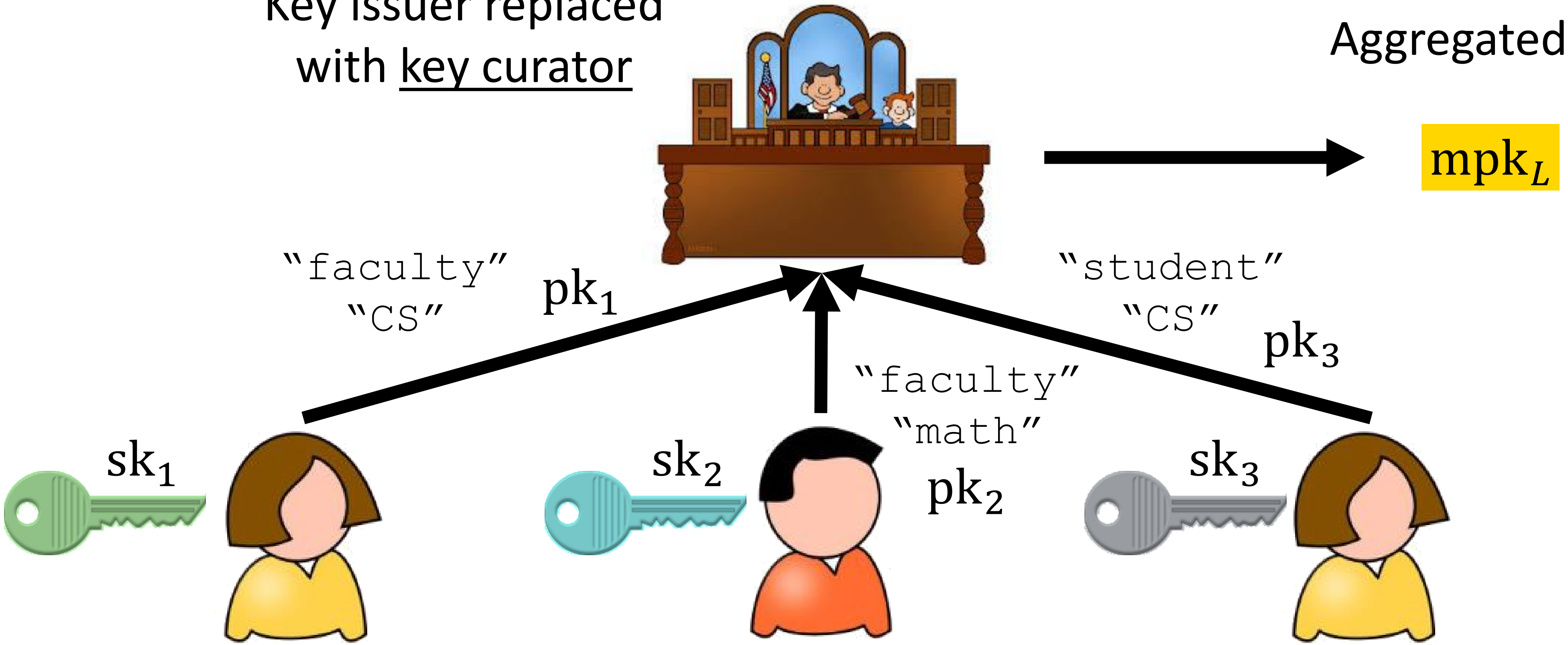
Users chooses their own public/secret key

As users join the system, public key is updated

Registered ABE

Key issuer replaced with <u>key curator</u>

Aggregated key



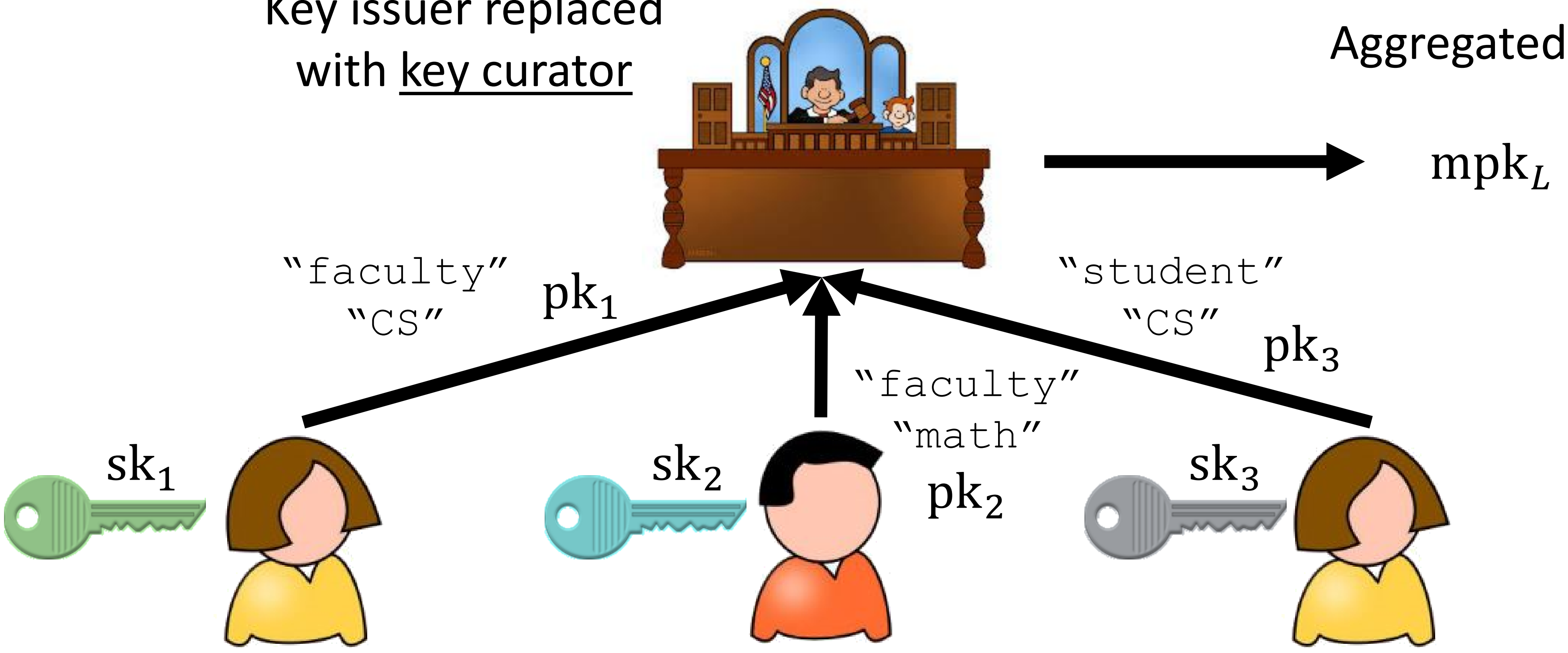
Users chooses their <u>own</u> public/secret key

As users join the system, public key is <u>updated</u>

Registered ABE

Key issuer replaced with <u>key curator</u>

Aggregated key



Users chooses their <u>own</u> public/secret key

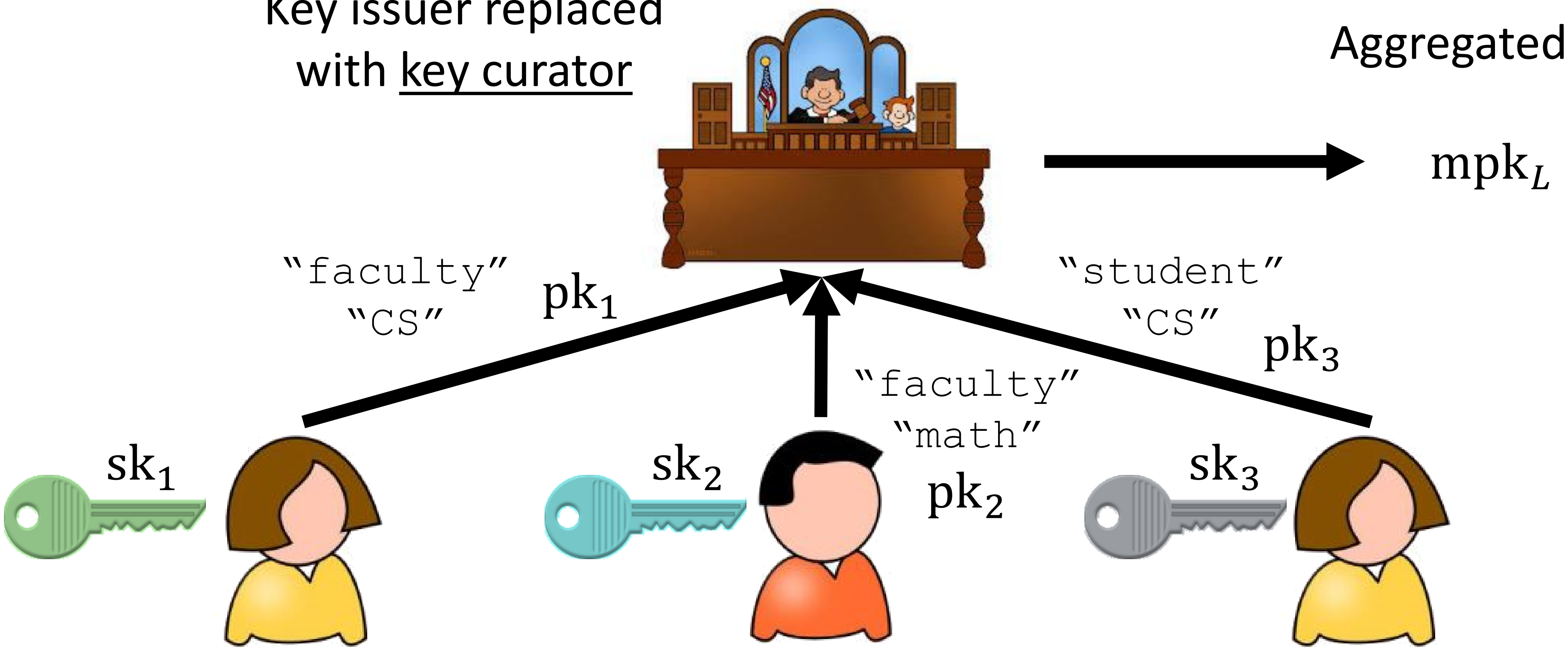
$|\mathrm{mpk}_L| = \mathrm{poly}(\lambda, \log L)$

Public key is *short*

Registered ABE

Key issuer replaced with <u>key curator</u>

Aggregated key



Users chooses their <u>own</u> public/secret key

master public key can be used to encrypt to policies, as in vanilla ABE

policy: `CS` and `faculty`

Registered ABE

Key issuer replaced
with <u>key curator</u>

Aggregated key

mpk_L

"faculty"
"CS"

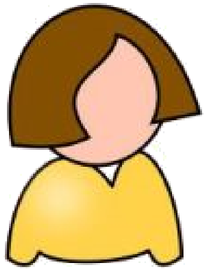
pk_1

To decrypt, user periodically
retrieves a helper decryption key hsk
(function of mpk and user's **public** key pk_1)

$|\mathrm{hsk}| = \mathrm{poly}(\lambda, \log L)$

$\#\ \mathrm{updates} = \mathrm{poly}(\lambda, \log L)$

sk_1



hsk_1

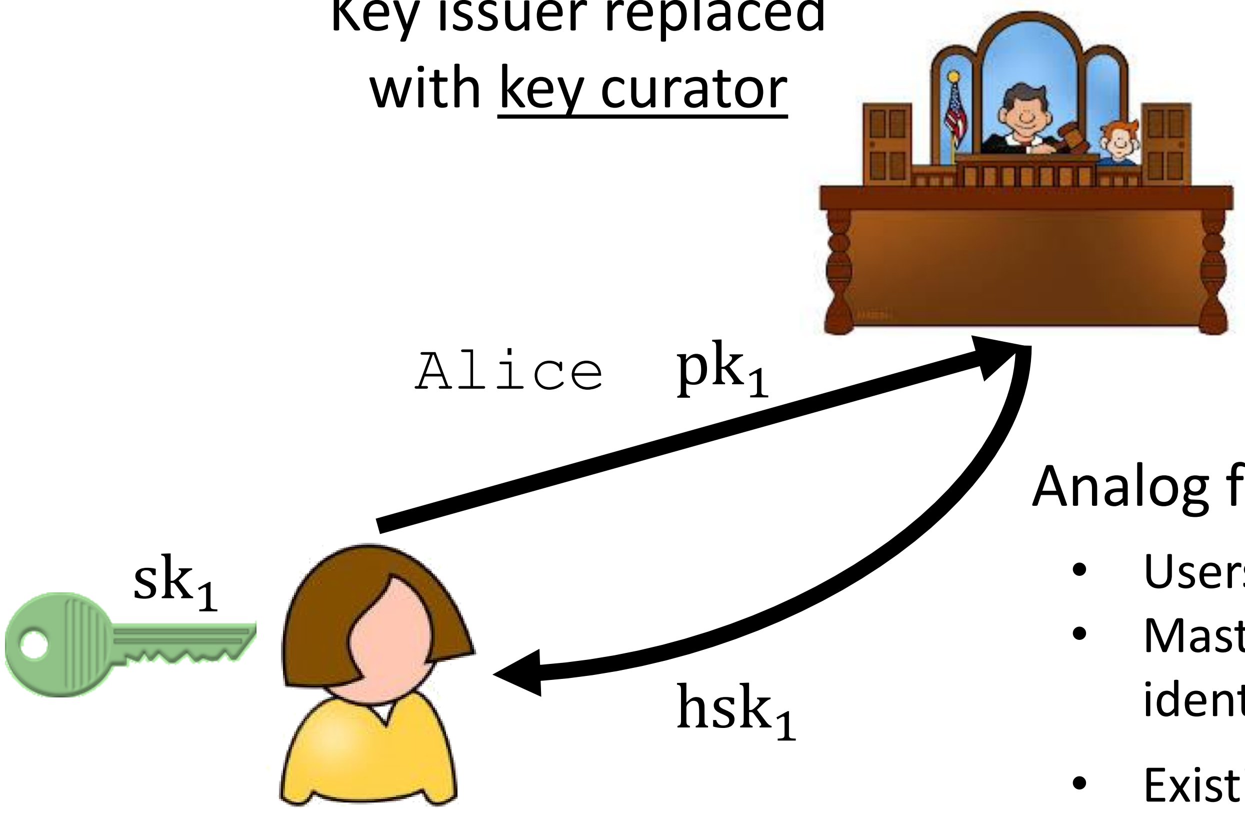
Users chooses their <u>own</u>
public/secret key

Key curator maintains
no secrets

Registration-Based Encryption (RBE)

[GHMR18]

Key issuer replaced with <u>key curator</u>



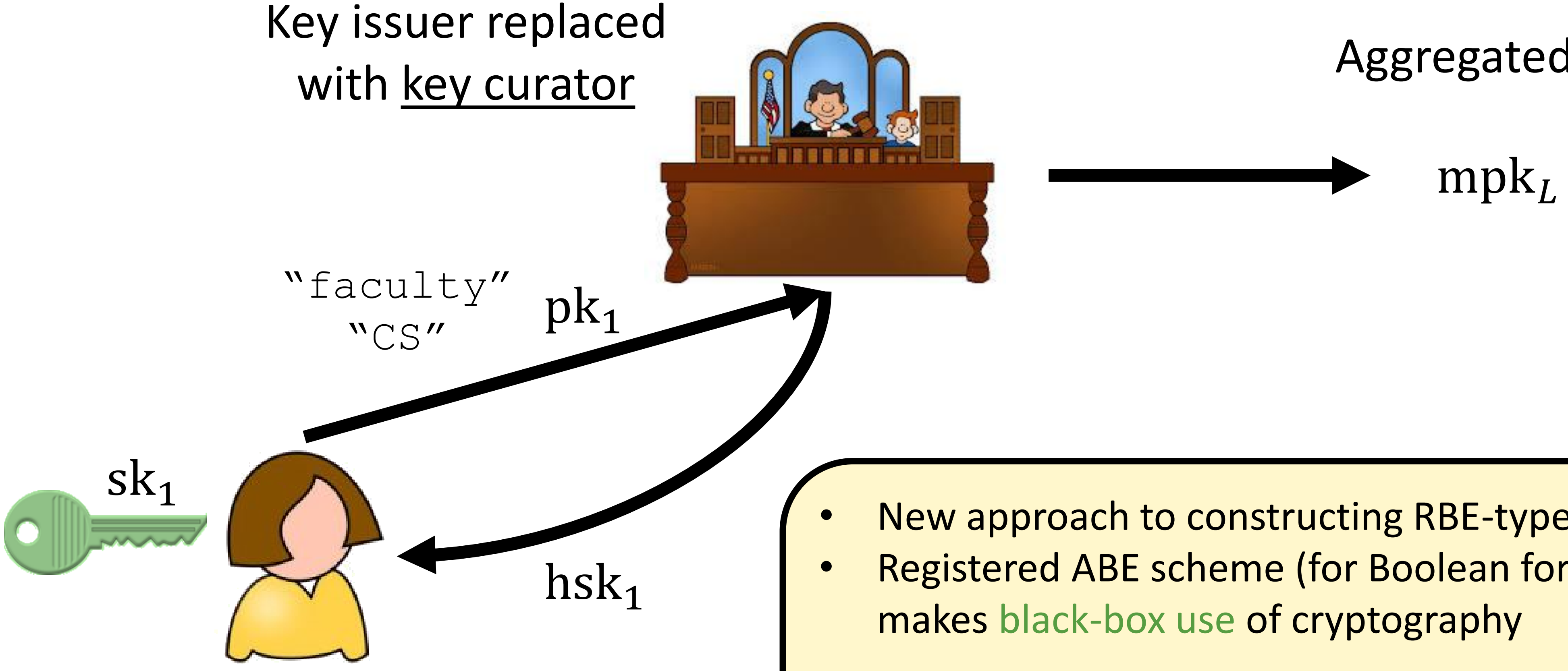
Aggregated key

mpk_L

Analog for *identity-based encryption* (IBE)

- Users register their public key along with an identity
- Master public key allows encryption to an arbitrary identity
- Existing constructions based on indistinguishability obfuscation or "hash garbling" (based on CDH, QR, LWE) – all require **non-black-box** use of cryptography
- High concrete efficiency costs: ciphertext is 4.5 TB for supporting 2 billion users [CES21]

This Work



- New approach to constructing RBE-type of primitives
- Registered ABE scheme (for Boolean formulas) only makes black-box use of cryptography

- Construction will need a (trusted) common reference string (CRS) and supports bounded number of users

Starting Point: A Slotted Scheme

Let L be the number of users

pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4	$\cdots$	pk_L, S_L

$$\xrightarrow{\text{Aggregate}} \begin{array}{c} \mathrm{mpk} \\ \mathrm{hsk}_1, \ldots, \mathrm{hsk}_L \end{array}$$

Each slot associated with a public key pk and a set of attributes S

$$|\mathrm{mpk}| = \mathrm{poly}(\lambda, |\mathcal{U}|, \log L)$$

$$|\mathrm{hsk}_i| = \mathrm{poly}(\lambda, |\mathcal{U}|, \log L)$$

λ: security parameter

$\mathcal{U}$: universe of attributes

For special case of IBE with identities of length ℓ, $|\mathcal{U}| = 2\ell$

Starting Point: A Slotted Scheme

Let L be the number of users

pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4	$\cdots$	pk_L, S_L

$\xrightarrow{\text{Aggregate}}$ mpk, $\mathrm{hsk}_1, \dots, \mathrm{hsk}_L$

Each slot associated with a public key pk and a set of attributes S

$$|\mathrm{mpk}| = \mathrm{poly}(\lambda, |\mathcal{U}|, \log L)$$

$$|\mathrm{hsk}_i| = \mathrm{poly}(\lambda, |\mathcal{U}|, \log L)$$

λ: security parameter

$\mathcal{U}$: universe of attributes

$\mathrm{Encrypt}(\mathrm{mpk}, P, m) \to \mathrm{ct}$

Encryption takes master public key and policy P (no slot)

$\mathrm{Decrypt}(\mathrm{sk}_i, \mathrm{hsk}_i, \mathrm{ct}) \to m$

Decryption takes secret key sk_i for some slot and the helper key hsk_i for that slot

Starting Point: A Slotted Scheme

Let L be the number of users

pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4	$\cdots$	pk_L, S_L

$\xrightarrow{\text{Aggregate}}$ mpk, $\mathrm{hsk}_1, \ldots, \mathrm{hsk}_L$

Each slot associated with a public key pk and a set of attributes S

$$|\mathrm{mpk}| = \mathrm{poly}(\lambda, |\mathcal{U}|, \log L)$$

$$|\mathrm{hsk}_i| = \mathrm{poly}(\lambda, |\mathcal{U}|, \log L)$$

λ: security parameter

$\mathcal{U}$: universe of attributes

$$\mathrm{Encrypt}(\mathrm{mpk}, P, m) \rightarrow \mathrm{ct}$$

$$\mathrm{Decrypt}(\mathrm{sk}_i, \mathrm{hsk}_i, \mathrm{ct}) \rightarrow m$$

Main difference with registered ABE:
Aggregate takes all L keys simultaneously

Constructing Slotted Registered ABE

Construction will rely on composite-order pairing groups

Let $\mathbb{G}$ be a group of order $N = p_1 p_2 p_3$ (composite order)

Scheme essentially operates in $\mathbb{G}_{p_1}$
(other subgroups used for randomization and security proof)

Pairing is an efficiently-computable bilinear map on $\mathbb{G}$:

$$e(g^x, g^y) = e(g, g)^{xy}$$

Multiplies exponents in the target group

Warm-Up: A Single-Slot Scheme

For simplicity: will describe scheme for conjunction policies

Generalizes to policies that can be described by linear secret sharing scheme

Scheme will rely on a common reference string (CRS)

General components: $Z = e(g, g)^{\alpha}$ $h = g^{\beta}$

g is generator for $\mathbb{G}_1$

Slot components: $A = g^{t}$ $B = g^{\alpha} h^{t}$

Attribute components: $U_w = g^{u_w}$ for each $w \in \mathcal{U}$

[Scheme described here does not have all the randomization needed for security – see paper for actual scheme]

Single-Slot Aggregation

	general		slot-specific		attribute
Common reference string:	$Z = e(g, g)^{\alpha}$	$h = g^{\beta}$	$A = g^t$	$B = g^{\alpha} h^t$	$U_w = g^{u_w}$

User's public/secret key: $\mathrm{sk} = r, \ \mathrm{pk} = g^r$ (ElGamal key)

Aggregated key: $\mathrm{pk}_1 = g^r$
(for 1 slot) $S_1 \subseteq |\mathcal{U}|$

General components: $Z = e(g, g)^{\alpha}$ $h = g^{\beta}$

Slot components: $\hat{T} = g^r$

Attribute components: $\hat{U}_w = 1$ if $w \in S_1$
$\hat{U}_w = U_w$ if $w \notin S_1$

mpk

Slot components: $A = g^t$ $B = g^{\alpha} h^t$

hsk_1

Single-Slot Ciphertext

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^\alpha \quad h = g^\beta$	$\hat{T} = g^r$	$\hat{U}_w = g^{u_w}$ for $w \notin S_1$

Encrypting message $\boldsymbol{\mu}$ to policy $\bigwedge_{i \in [\ell]} \boldsymbol{w_i}$:

Sample encryption randomness $s_1, \ldots, s_\ell \leftarrow \mathbb{Z}_N$ and let $s = s_1 + \cdots + s_\ell$

Sample $h_1, h_2 \leftarrow \mathbb{G}_{p_1}$ such that $h = h_1 h_2$

Message components: $C_1 = \mu \cdot Z^s \qquad C_2 = g^s$

Attribute components: $C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i} \qquad C_{4,i} = g^{\gamma_i}$

Slot components: $C_5 = h_1^s \hat{T}^{-\gamma_0} \qquad C_6 = g^{\gamma_0}$

$\gamma_0, \gamma_1, \ldots, \gamma_\ell$
additional blinding factors

Single-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^{\alpha}$ $\quad h = g^{\beta}$	$\hat{T} = g^{r}$	$\hat{U}_w = g^{u_w}$ for $w \notin S_1$
Helper key:		$A = g^{t}$ $\quad B = g^{\alpha} h^{t}$	
Ciphertext:	$C_1 = \mu \cdot Z^{s}$ $\quad C_2 = g^{s}$	$C_5 = h_1^{s} \hat{T}^{-\gamma_0}$ $\quad C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i}$ $\quad C_{4,i} = g^{\gamma_i}$

Goal: recover $Z^{s} = e(g,g)^{\alpha s}$

Observe: $e(B, C_2) = e\left(g^{\alpha} h^{t}, g^{s}\right) = e(g,g)^{\alpha s} e(h,g)^{st}$

Recall: $h = h_1 h_2$ so suffices to compute $e(h_1, g)^{st}$ and $e(h_2, g)^{st}$

Computing this requires knowledge of secret key for the slot

Computing this requires that attributes associated with the slot satisfy the policy

Single-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^{\alpha}$ $\quad h = g^{\beta}$	$\hat{T} = g^{r}$	$\hat{U}_w = g^{u_w}$ for $w \notin S_1$
Helper key:		$A = g^{t}$ $\quad B = g^{\alpha} h^{t}$	
Ciphertext:	$C_1 = \mu \cdot Z^{s}$ $\quad C_2 = g^{s}$	$C_5 = h_1^{s} \hat{T}^{-\gamma_0}$ $\quad C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i}$ $\quad C_{4,i} = g^{\gamma_i}$

Slot specific check: recover $e(h_1, g)^{st}$

Single-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^\alpha \quad h = g^\beta$	$\hat{T} = g^r$	$\hat{U}_w = g^{u_w}$ for $w \notin S_1$
Helper key:		$A = g^t \quad B = g^\alpha h^t$	
Ciphertext:	$C_1 = \mu \cdot Z^s \quad C_2 = g^s$	$C_5 = h_1^s \hat{T}^{-\gamma_0} \quad C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i} \quad C_{4,i} = g^{\gamma_i}$

Slot specific check: recover $e(h_1, g)^{st}$

$$e(C_5, A) = e\left(h_1^s \hat{T}^{-\gamma_0}, g^t\right) = e(h_1, g)^{st} e\left(\hat{T}, g\right)^{-\gamma_0 t} = e(h_1, g)^{st} e(g,g)^{-\gamma_0 r t}$$

$$e(C_6, A)^r = e\left(g^{\gamma_0}, g^t\right)^r = e(g,g)^{\gamma_0 r t}$$

Recall: r is the secret key

> Product of **three** quantities in the exponent – computing this requires knowledge of one of the exponents (namely, the secret key r)

Single-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^\alpha \quad h = g^\beta$	$\hat{T} = g^r$	$\hat{U}_w = g^{u_w}$ for $w \notin S_1$
Helper key:		$A = g^t \quad B = g^\alpha h^t$	
Ciphertext:	$C_1 = \mu \cdot Z^s \quad C_2 = g^s$	$C_5 = h_1^s \hat{T}^{-\gamma_0} \quad C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i} \quad C_{4,i} = g^{\gamma_i}$

Attribute check: recover $e(h_2, g)^{st}$

If $w_i \in S$, then $U_w = 1$ and $C_{3,i} = h_2^{s_i}$

$$\prod_{i \in [\ell]} C_{3,i} = \prod_{i \in [\ell]} h_2^{s_i} = h_2^{\sum_{i \in [\ell]} s_i} = h_2^s$$

If $w_i \notin S$, then $h_2^{s_i}$ is blinded by $U_{w_i}^{-\gamma_i} = g^{-u_{w_i}\gamma_i}$ and pairing with g^t produces a term $g^{-u_{w_i}\gamma_i t}$

$$e(h_2^s, A) = e\left(h_2^s, g^t\right) = e(h_2, g)^{st}$$

Single-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^{\alpha}$ $h = g^{\beta}$	$\hat{T} = g^{r}$	$\hat{U}_w = g^{u_w}$ for $w \notin S_1$
Helper key:		$A = g^{t}$ $B = g^{\alpha} h^{t}$	
Ciphertext:	$C_1 = \mu \cdot Z^{s}$ $C_2 = g^{s}$	$C_5 = h_1^{s} \hat{T}^{-\gamma_0}$ $C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i}$ $C_{4,i} = g^{\gamma_i}$

Goal: recover $Z^{s} = e(g,g)^{\alpha s}$

Observe: $e(B, C_2) = e\left(g^{\alpha} h^{t}, g^{s}\right) = e(g,g)^{\alpha s} e(h,g)^{st}$

Recall: $h = h_1 h_2$ so suffices to compute $e(h_1, g)^{st}$ and $e(h_2, g)^{st}$

Slot specific check: recover $e(h_1, g)^{st}$

Attribute check: recover $e(h_2, g)^{st}$

$\Rightarrow$ Recover $e(h, g)^{st}$

Extending to Multiple Slots

	general	slot-specific	attribute
Common reference string:	$Z = e(g,g)^\alpha \quad h = g^\beta$	$A = g^t \quad B_1 = g^\alpha h^t$	$U_w = g^w$

Idea: replicate components for each slot

Extending to Multiple Slots

general | slot-specific | attribute

Common reference string: $Z = e(g,g)^{\alpha}$ $\quad h = g^{\beta}$ $\quad A_1 = g^{t_1}$ $\quad B_1 = g^{\alpha} h^{t_1}$ $\quad U_{w,1} = g^{u_{w,1}}$

$A_2 = g^{t_2}$ $\quad B_2 = g^{\alpha} h^{t_2}$ $\quad U_{w,2} = g^{u_{w,2}}$

$\vdots$ $\quad \vdots$

$A_L = g^{t_L}$ $\quad B_L = g^{\alpha} h^{t_L}$ $\quad U_{w,L} = g^{u_{w,L}}$

Idea: replicate components for each slot

Multi-Slot Aggregation

	general	slot-specific	attribute
Common reference string:	$Z = e(g,g)^\alpha \quad h = g^\beta$	$A_i = g^{t_i} \quad B_i = g^\alpha h^{t_i}$	$U_{w,i} = g^{u_{w,i}}$

User's public/secret keys: $\mathrm{pk}_1 = g^{r_1}, \ldots, \mathrm{pk}_L = g^{r_L}$

Single slot setting:

Slot components:

$\hat{T} = g^r$

Attribute components:

$\hat{U}_w = 1 \quad$ if $w \in S$

$\hat{U}_w = U_w \quad$ if $w \notin S$

$$\boxed{g^{r_1}}\boxed{\cdots}\boxed{g^{r_L}} \Rightarrow \hat{T} = \prod_{i \in [L]} g^{r_i}$$

$$\boxed{\hat{U}_{w,1}}\boxed{\cdots}\boxed{\hat{U}_{w,L}} \Rightarrow \hat{U} = \prod_{i \in [L]} \hat{U}_{w,i}$$

Aggregate by multiplying across slots

(Similar to vector commitments [CF13])

Multi-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^{\alpha}$ $h = g^{\beta}$	$\hat{T} = \prod_{i \in [L]} g^{r_i}$	$\hat{U}_w = \prod_{w \notin S_i} g^{u_{w,i}}$
Ciphertext:	$C_1 = \mu \cdot Z^s$ $C_2 = g^s$	$C_5 = h_1^s \hat{T}^{-\gamma_0}$ $C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i}$ $C_{4,i} = g^{\gamma_i}$

Ciphertext structure is **unchanged**

Goal: recover $Z^s = e(g,g)^{\alpha s}$

Observe: $e(B_i, C_2) = e\left(g^{\alpha} h^{t_i}, g^s\right) = e(g,g)^{\alpha s} e(h,g)^{s t_i}$

Recall: $h = h_1 h_2$ so suffices to compute $e(h_1, g)^{s t_i}$ and $e(h_2, g)^{s t_i}$

Recall: $B_i = g^{\alpha} h^{t_i}$

Multi-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^{\alpha}$ $h = g^{\beta}$	$\hat{T} = \prod_{i \in [L]} g^{r_i}$	$\hat{U}_w = \prod_{w \notin S_i} g^{u_{w,i}}$
Ciphertext:	$C_1 = \mu \cdot Z^s$ $C_2 = g^s$	$C_5 = h_1^s \hat{T}^{-\gamma_0}$ $C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i}$ $C_{4,i} = g^{\gamma_i}$

Ciphertext structure is **unchanged**

Slot specific check: recover $e(h_1, g)^{st_i}$

Consider previous decryption equation ($A_i = g^{t_i}$):

$$e(C_5, A) = e\left(h_1^s \hat{T}^{-\gamma_0}, g^{t_i}\right) = e(h_1, g)^{st_i} e\left(\hat{T}, g\right)^{-\gamma_0 t_i}$$

$$= e(h_1, g)^{st_i} e(g,g)^{-\gamma_0 r_i t_i} \prod_{j \neq i} e(g,g)^{-\gamma_0 r_j t_i}$$

"single-slot component" "cross-terms"

Multi-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^{\alpha}$ $\quad h = g^{\beta}$	$\hat{T} = \prod_{i \in [L]} g^{r_i}$	$\hat{U}_w = \prod_{w \notin S_i} g^{u_{w,i}}$
Ciphertext:	$C_1 = \mu \cdot Z^s$ $\quad C_2 = g^s$	$C_5 = h_1^s \hat{T}^{-\gamma_0}$ $\quad C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i}$ $\quad C_{4,i} = g^{\gamma_i}$

$$e(h_1, g)^{st} \quad \boxed{e(g,g)^{-\gamma_0 r_i t_i}} \quad \boxed{\prod_{j \neq i} e(g,g)^{-\gamma_0 r_j t_i}}$$

User computes
$e(C_6, A_i)^{r_i} = e\left(g^{\gamma_0}, g^{t_i}\right)^{r_i} = e(g,g)^{\gamma_0 r_i t_i}$

User does *not* know r_j for $j \neq i$

Approach: Include "cross term component" as the helper decryption key

$$\hat{V}_i = \prod_{j \neq i} A_i^{r_j} = \prod_{j \neq i} g^{r_j t_i} \Longrightarrow e\left(g^{\gamma_0}, \hat{V}_i\right) = \prod_{j \neq i} g^{\gamma_0 r_j t_i}$$

Multi-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^\alpha \quad h = g^\beta$	$\hat{T} = \prod_{i \in [L]} g^{r_i}$	$\hat{U}_w = \prod_{w \notin S_i} g^{u_{w,i}}$
Ciphertext:	$C_1 = \mu \cdot Z^s \quad C_2 = g^s$	$C_5 = h_1^s \hat{T}^{-\gamma_0} \quad C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i} \quad C_{4,i} = g^{\gamma_i}$

Approach: Include "cross term component" as the helper decryption key

$$\hat{V}_i = \prod_{j \neq i} A_i^{r_j} = \prod_{j \neq i} g^{r_j t_i} \Longrightarrow e\left(g^{\gamma_0}, \hat{V}_i\right) = \prod_{j \neq i} g^{\gamma_0 r_j t_i}$$

At registration time, each user (who knows r_j) will *additionally* compute

$$V_{j,i} = A_i^{r_j} = g^{r_j t_i} \quad \text{for all} \quad i \neq j$$

Recall: $A_i = g^{t_i}$ is part of the CRS

Key-curator can then compute cross-term

$$\hat{V}_i = \prod_{j \neq i} V_{j,i}$$

Multi-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^\alpha \quad h = g^\beta$	$\hat{T} = \prod_{i \in [L]} g^{r_i}$	$\hat{U}_w = \prod_{w \notin S_i} g^{u_{w,i}}$
Ciphertext:	$C_1 = \mu \cdot Z^s \quad C_2 = g^s$	$C_5 = h_1^s \hat{T}^{-\gamma_0} \quad C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i} \quad C_{4,i} = g^{\gamma_i}$

Ciphertext structure is **unchanged**

Attribute check: recover $e(h_2, g)^{st_i}$

Can use a similar approach: for each $w \in \mathcal{U}$, include a cross-term $\widehat{W}_{i,w}$

Multi-Slot Decryption

	general	slot-specific	attribute
Master public key:	$Z = e(g,g)^{\alpha}$ $h = g^{\beta}$	$\hat{T} = \prod_{i \in [L]} g^{r_i}$	$\hat{U}_w = \prod_{w \notin S_i} g^{u_{w,i}}$
Ciphertext:	$C_1 = \mu \cdot Z^s$ $C_2 = g^s$	$C_5 = h_1^s \hat{T}^{-\gamma_0}$ $C_6 = g^{\gamma_0}$	$C_{3,i} = h_2^{s_i} \hat{U}_{w_i}^{-\gamma_i}$ $C_{4,i} = g^{\gamma_i}$

Helper decryption key $\mathbf{hsk}_i$ (for slot i):

$A_i = g^{t_i}$ $B_i = g^{\alpha} h^{t_i}$ **(same as single-slot setting)**

$\hat{V}_i$ **(cross-terms for slot-specific components)**

$\widehat{W}_{i,w}$ for each $w \in \mathcal{U}$ **(cross-terms for attribute components)**

$|\mathrm{hsk}_i| = \mathrm{poly}(\lambda, |\mathcal{U}|)$ *independent* of L

Slotted Scheme from Pairings

Let L be the number of users

pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4	$\cdots$	pk_L, S_L

$\xrightarrow{\text{Aggregate}}$

$$\mathrm{mpk}$$
$$\mathrm{hsk}_1, \ldots, \mathrm{hsk}_L$$

Each slot associated with a <u>public key</u> pk and a set of attributes S

$$|\mathrm{mpk}| = \mathrm{poly}(\lambda, |\mathcal{U}|)$$

$$|\mathrm{hsk}_i| = \mathrm{poly}(\lambda, |\mathcal{U}|)$$

λ: security parameter

$\mathcal{U}$: universe of attributes

$$\mathrm{Encrypt}(\mathrm{mpk}, P, m) \to \mathrm{ct}$$

$$\mathrm{Decrypt}(\mathrm{sk}_i, \mathrm{hsk}_i, \mathrm{ct}) \to m$$

Security relies on assumptions over composite-order pairing groups
[see paper for details]

Slotted Registered ABE to Registered ABE

Let L be the number of users

pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4	$\cdots$	pk_L, S_L

$$\xrightarrow{\text{Aggregate}} \begin{array}{c} \mathrm{mpk} \\ \mathrm{hsk}_1, \ldots, \mathrm{hsk}_L \end{array}$$

Slotted scheme does *not* support online registration

Solution: use "powers-of-two" approach (like [GHMR18])

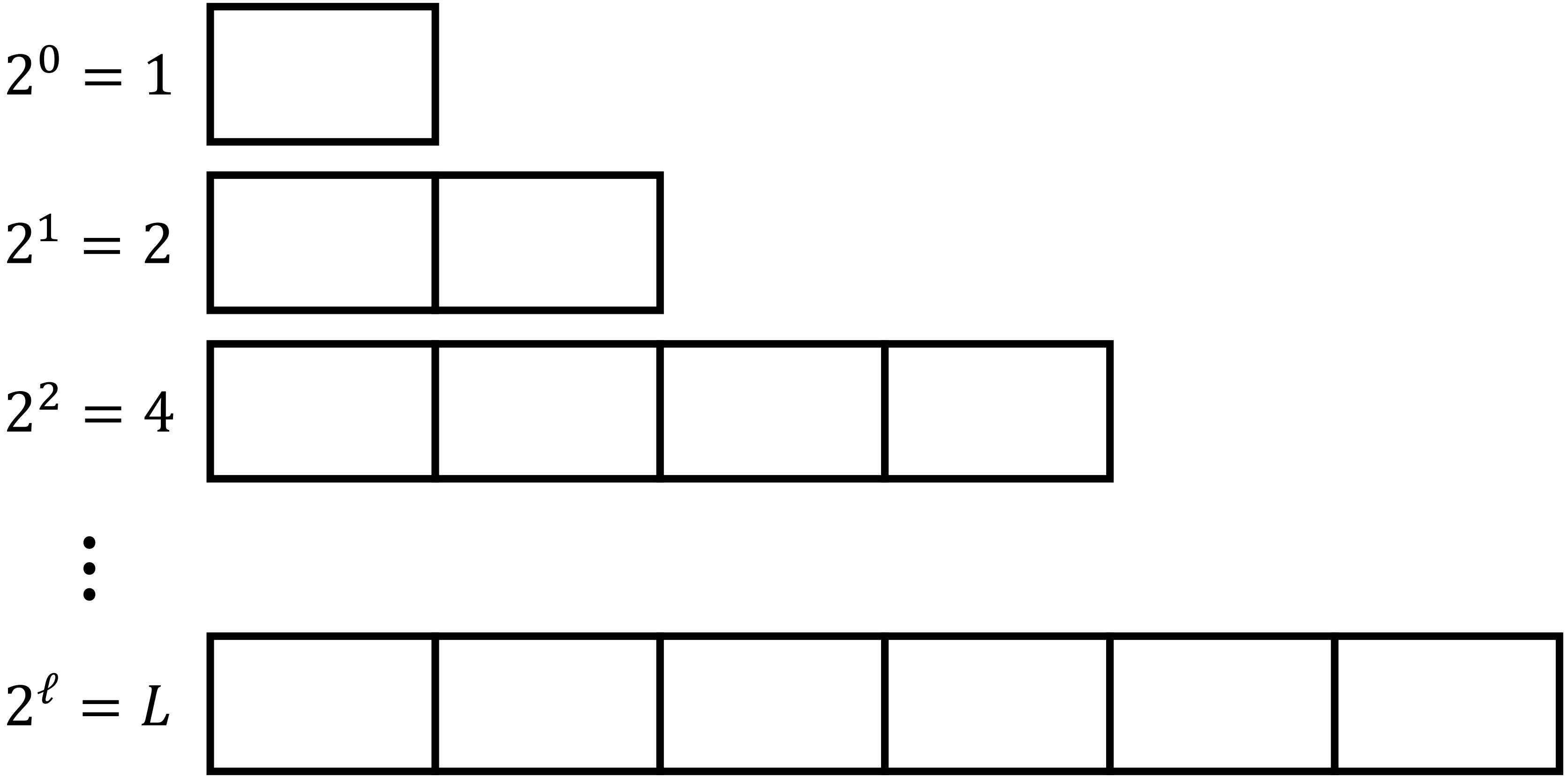
Slotted Registered ABE to Registered ABE

Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

Initially: all slots are empty

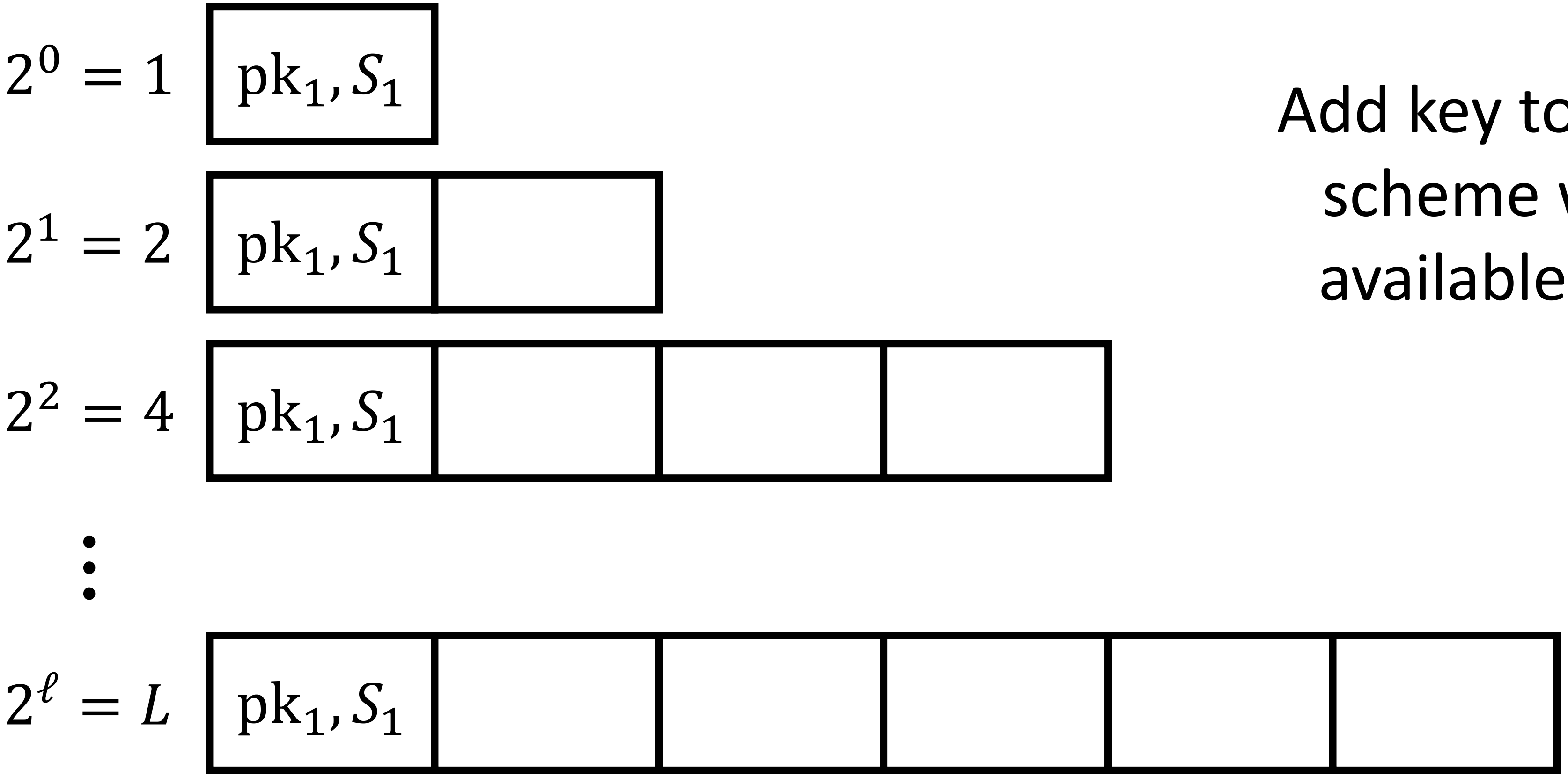
$$\mathrm{mpk} = \perp$$



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

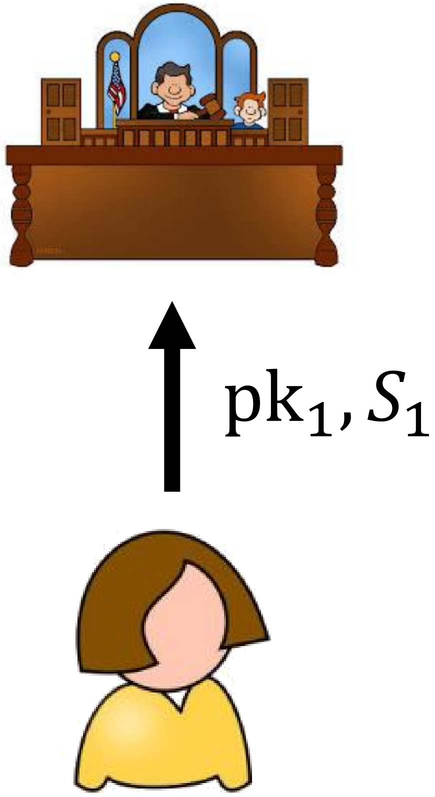
To support $L = 2^\ell$ users: maintain ℓ slotted schemes



Initially: all slots are empty

$$\mathrm{mpk} = \perp$$

Add key to each scheme with available slot



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$ | pk_1, S_1 | all slots are full $\longrightarrow \mathrm{mpk}_1$

$2^1 = 2$ | pk_1, S_1 | |

$2^2 = 4$ | pk_1, S_1 | | | |

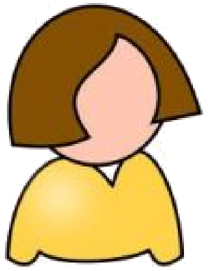
$\vdots$

$2^\ell = L$ | pk_1, S_1 | | | | | |

Initially: all slots are empty

$$\mathrm{mpk} = \perp$$

$\uparrow \mathrm{pk}_1, S_1$



Slotted Registered ABE to Registered ABE

Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

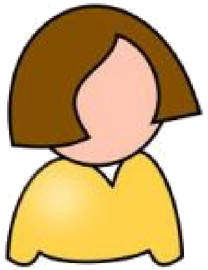
Initially: all slots are empty

$\mathrm{mpk} = (\mathrm{mpk}_1)$

$2^0 = 1$	pk_1, S_1					
$2^1 = 2$	pk_1, S_1					
$2^2 = 4$	pk_1, S_1					
$\vdots$						
$2^\ell = L$	pk_1, S_1					

$\xrightarrow{\text{all slots are full}} \mathrm{mpk}_1$

$\uparrow \mathrm{pk}_1, S_1$



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$ | pk_1, S_1 | $\xrightarrow{\text{all slots are full}} \mathrm{mpk}_1$

$2^1 = 2$ | pk_1, S_1 | pk_2, S_2 |

$2^2 = 4$ | pk_1, S_1 | pk_2, S_2 | | |

$\vdots$

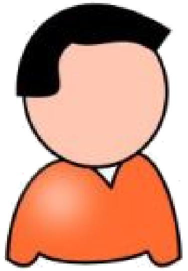
$2^\ell = L$ | pk_1, S_1 | pk_2, S_2 | | | | |

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_1)$$

Add key to each scheme with available slot

$\uparrow \mathrm{pk}_2, S_2$



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$ | pk_1, S_1 | all slots are full → mpk_1

$2^1 = 2$ | pk_1, S_1 | pk_2, S_2 | all slots are full → mpk_2

clear out previous schemes

$2^2 = 4$ | pk_1, S_1 | pk_2, S_2 | | |

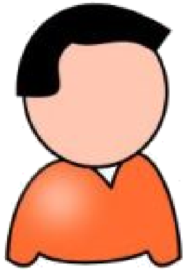
$\vdots$

$2^\ell = L$ | pk_1, S_1 | pk_2, S_2 | | | | |

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_1)$$

pk_2, S_2



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$	

$2^1 = 2$ | pk_1, S_1 | pk_2, S_2 $\xrightarrow{\text{all slots are full}}$ mpk_2

clear out previous schemes

$2^2 = 4$ | pk_1, S_1 | pk_2, S_2 | | |

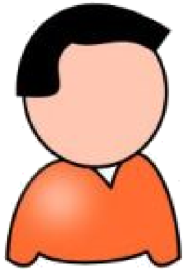
$\vdots$

$2^\ell = L$ | pk_1, S_1 | pk_2, S_2 | | | | |

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_1)$$

pk_2, S_2



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$

$2^1 = 2$

pk_1, S_1	pk_2, S_2

$\xrightarrow{\text{all slots are full}}$ mpk_2

clear out previous schemes

$2^2 = 4$

pk_1, S_1	pk_2, S_2		

⋮

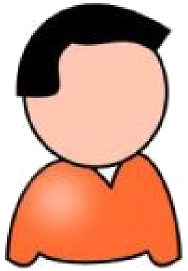
$2^\ell = L$

pk_1, S_1	pk_2, S_2				

Initially: all slots are empty

$\mathrm{mpk} = (\mathrm{mpk}_2)$

pk_2, S_2



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$	pk_3, S_3					
$2^1 = 2$	pk_1, S_1	pk_2, S_2				
$2^2 = 4$	pk_1, S_1	pk_2, S_2	pk_3, S_3			
$\vdots$						
$2^\ell = L$	pk_1, S_1	pk_2, S_2	pk_3, S_3			

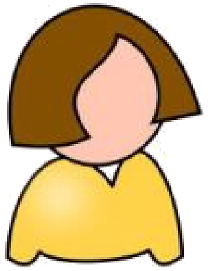
$\xrightarrow{\text{all slots are full}} \mathrm{mpk}_2$

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_2)$$

Add key to each scheme with available slot

$\uparrow \mathrm{pk}_3, S_3$



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$ | pk_3, S_3 | all slots are full $\longrightarrow \mathrm{mpk}_1$

$2^1 = 2$ | pk_1, S_1 | pk_2, S_2 | all slots are full $\longrightarrow \mathrm{mpk}_2$

$2^2 = 4$ | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | |

$\vdots$

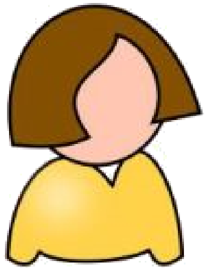
$2^\ell = L$ | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | | | |

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_2)$$

Add key to each scheme with available slot

$\uparrow \mathrm{pk}_3, S_3$



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$: | pk_3, S_3 | — all slots are full → mpk_1

$2^1 = 2$: | pk_1, S_1 | pk_2, S_2 | — all slots are full → mpk_2

$2^2 = 4$: | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | |

$\vdots$

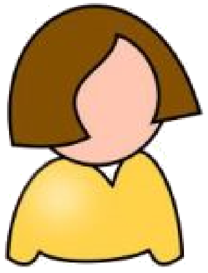
$2^\ell = L$: | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | | | |

Initially: all slots are empty

$\mathrm{mpk} = (\mathrm{mpk}_1, \mathrm{mpk}_2)$

Add key to each scheme with available slot

pk_3, S_3



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_1, \mathrm{mpk}_2)$$

$2^0 = 1$ | pk_3, S_3 — all slots are full $\longrightarrow \mathrm{mpk}_1$

$2^1 = 2$ | pk_1, S_1 | pk_2, S_2 — all slots are full $\longrightarrow \mathrm{mpk}_2$

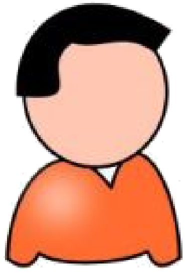
$2^2 = 4$ | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | pk_4, S_4

⋮

$2^\ell = L$ | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | pk_4, S_4 | |

Add key to each scheme with available slot

$\uparrow$ pk_4, S_4



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

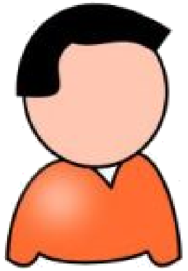
Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_1, \mathrm{mpk}_2)$$

$2^0 = 1$	pk_3, S_3					all slots are full	mpk_1
$2^1 = 2$	pk_1, S_1	pk_2, S_2				all slots are full	mpk_2
$2^2 = 4$	pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4		all slots are full	mpk_3
$\vdots$							
$2^\ell = L$	pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4			

clear out previous schemes

pk_4, S_4



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_1, \mathrm{mpk}_2)$$

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$ | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | pk_4, S_4 |

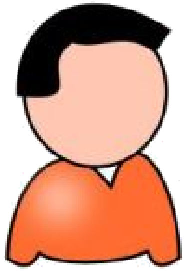
all slots are full $\longrightarrow$ mpk_3

$\vdots$

$2^\ell = L$ | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | pk_4, S_4 | | |

clear out previous schemes

pk_4, S_4



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

Initially: all slots are empty

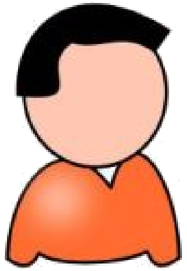
$\mathrm{mpk} = (\mathrm{mpk}_3)$

$2^0 = 1$						
$2^1 = 2$						
$2^2 = 4$	pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4		
$\vdots$						
$2^\ell = L$	pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4		

$\xrightarrow{\text{all slots are full}} \mathrm{mpk}_3$

clear out previous schemes

pk_4, S_4



Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$						
$2^1 = 2$						
$2^2 = 4$	pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4		
$\vdots$						
$2^\ell = L$	pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4		

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_3)$$

Ciphertext is an encryption to <u>each</u> public key

$\log L$ overhead

Slotted Registered ABE to Registered ABE

Solution: use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain ℓ slotted schemes

$2^0 = 1$						
$2^1 = 2$						
$2^2 = 4$	pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4		
$\vdots$						
$2^\ell = L$	pk_1, S_1	pk_2, S_2	pk_3, S_3	pk_4, S_4		

Initially: all slots are empty

$$\mathrm{mpk} = (\mathrm{mpk}_3)$$

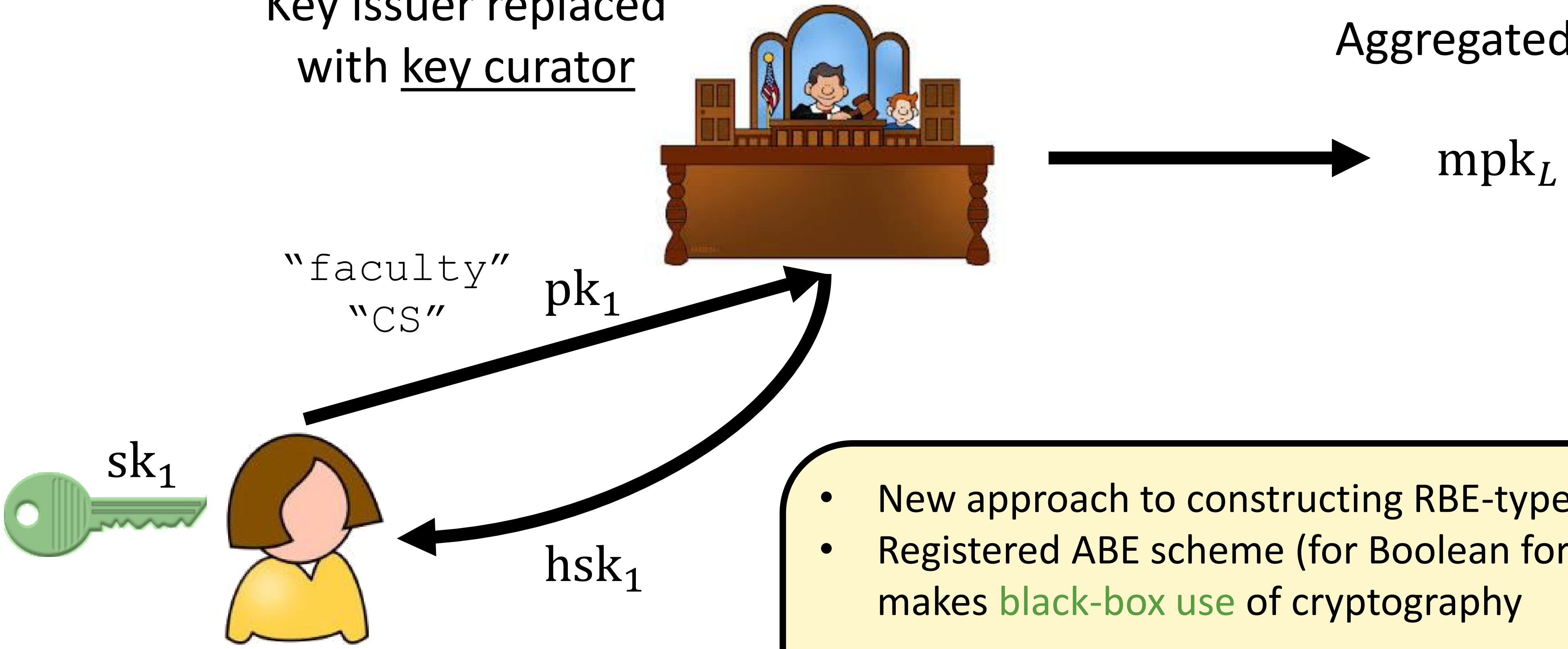
Update needed whenever user's key moves from scheme i to scheme $j > i$

At most $\ell = \log L$ updates

Registered ABE Summary

Key issuer replaced with <u>key curator</u>

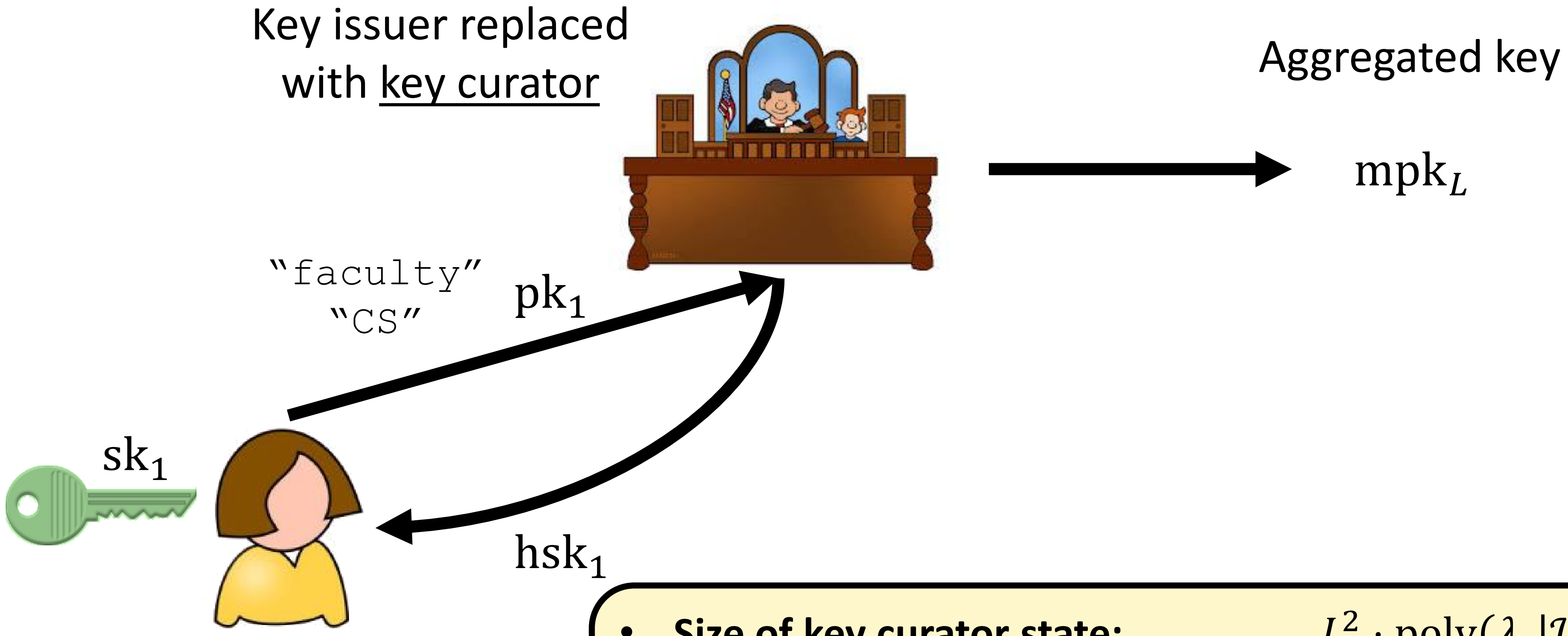
Aggregated key



Users chooses their <u>own</u> public/secret key

- New approach to constructing RBE-type of primitives
- Registered ABE scheme (for Boolean formulas) only makes black-box use of cryptography
- Construction will need a (trusted) common reference string (CRS) and supports bounded number of users

Registered ABE Summary



Users chooses their own public/secret key

- **Size of key curator state:** $L^2 \cdot \text{poly}(\lambda, |\mathcal{U}|, \log L)$
- **Key-generation/registration time:** $L \cdot \text{poly}(\lambda, |\mathcal{U}|, \log L)$
- **Public key, helper key size:** $|\mathcal{U}| \cdot \text{poly}(\lambda, \log L)$
- **Ciphertext size (for policy P):** $|P| \cdot \text{poly}(\lambda, \log L)$

Summary

This work: registered ABE for policies that can be based on linear secret sharing

- Only needs black-box use of cryptography
- Security based on composite-order bilinear map assumptions
- Supports *a priori* bounded number of users

Open questions:

- Registered ABE for general circuit policies
- Registered ABE for unbounded number of users
- Registered ABE with a *large* universe

Possible using indistinguishability obfuscation [see paper]

Registration-based model for *other* notions?

Thank you!