Removing Trust Assumptions from Functional Encryption

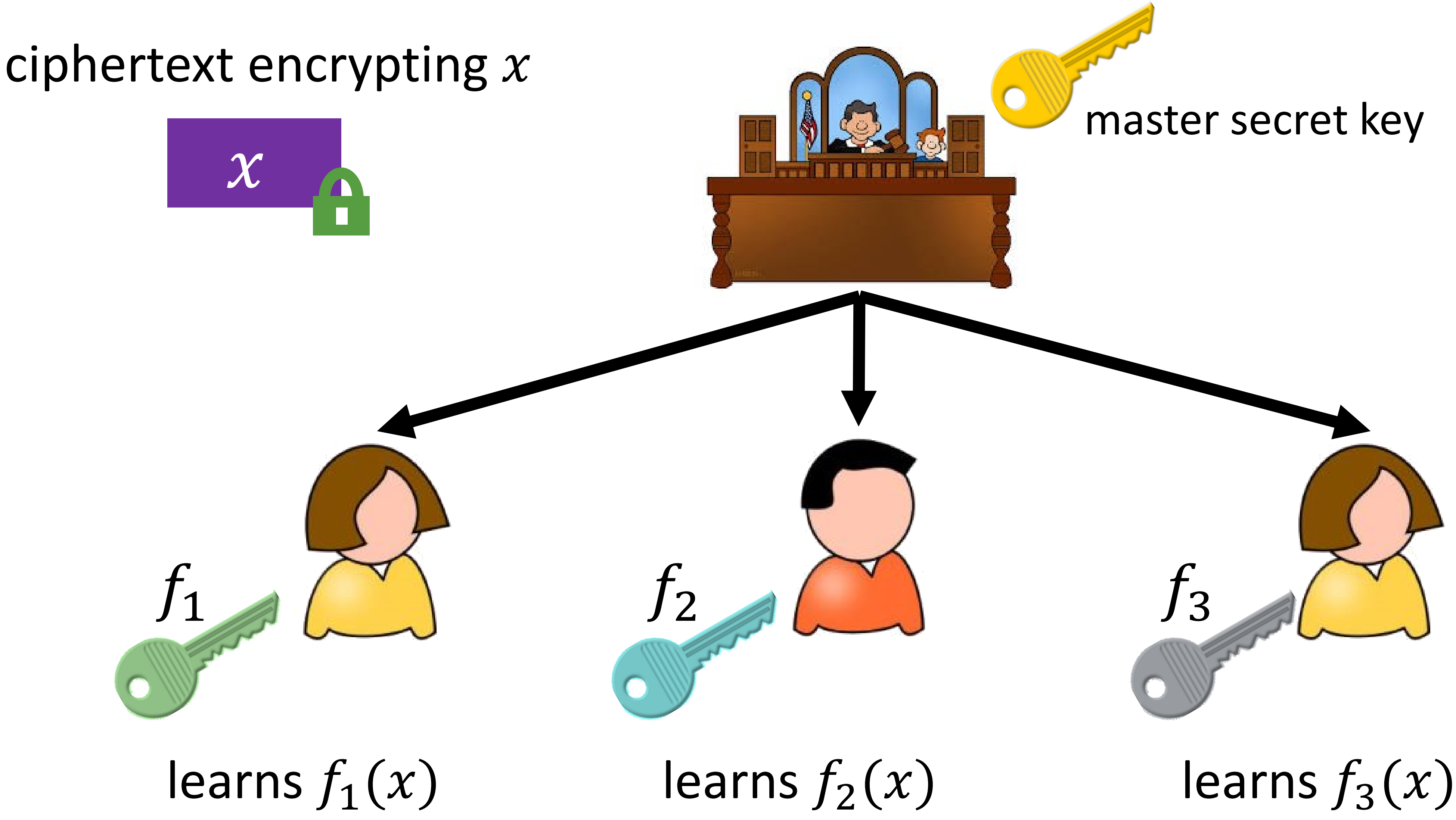
David Wu

December 2023

based on joint works with Cody Freitag, Rachit Garg, Susan Hohenberger, George Lu, and Brent Waters

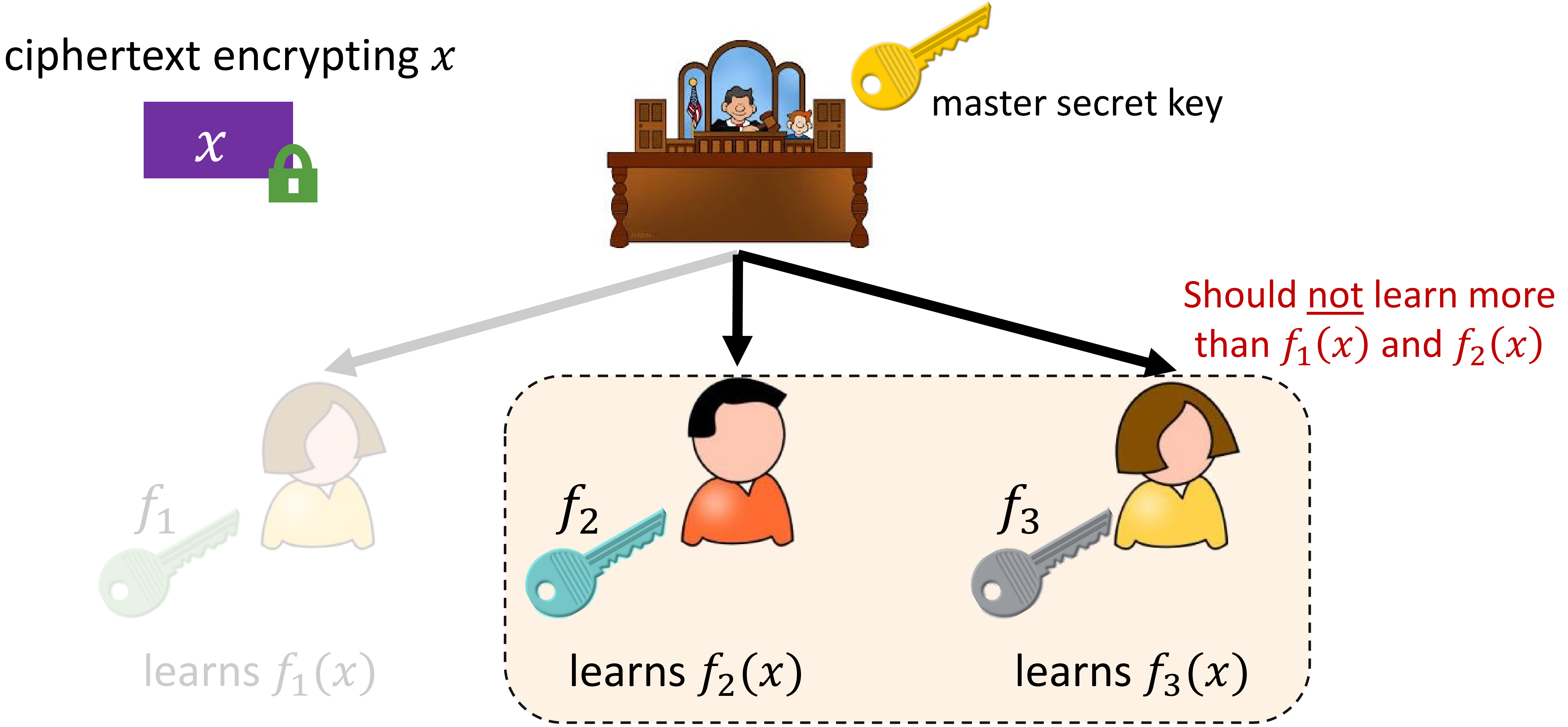
Functional Encryption (FE)

[SS10, O’N10, BSW11]



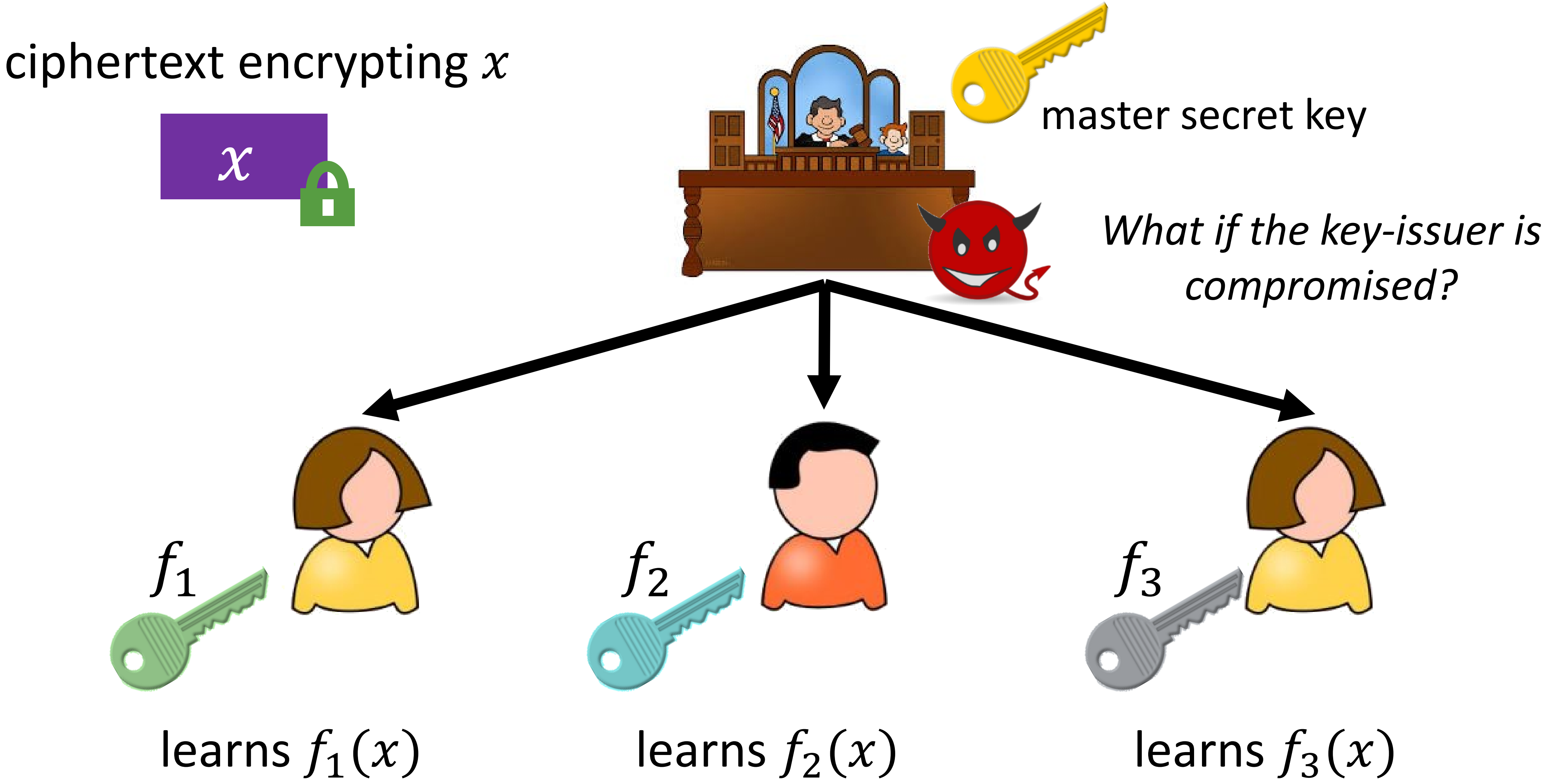
Functional Encryption (FE)

[SS10, O'N10, BSW11]



Functional Encryption (FE)

[SS10, O'N10, BSW11]



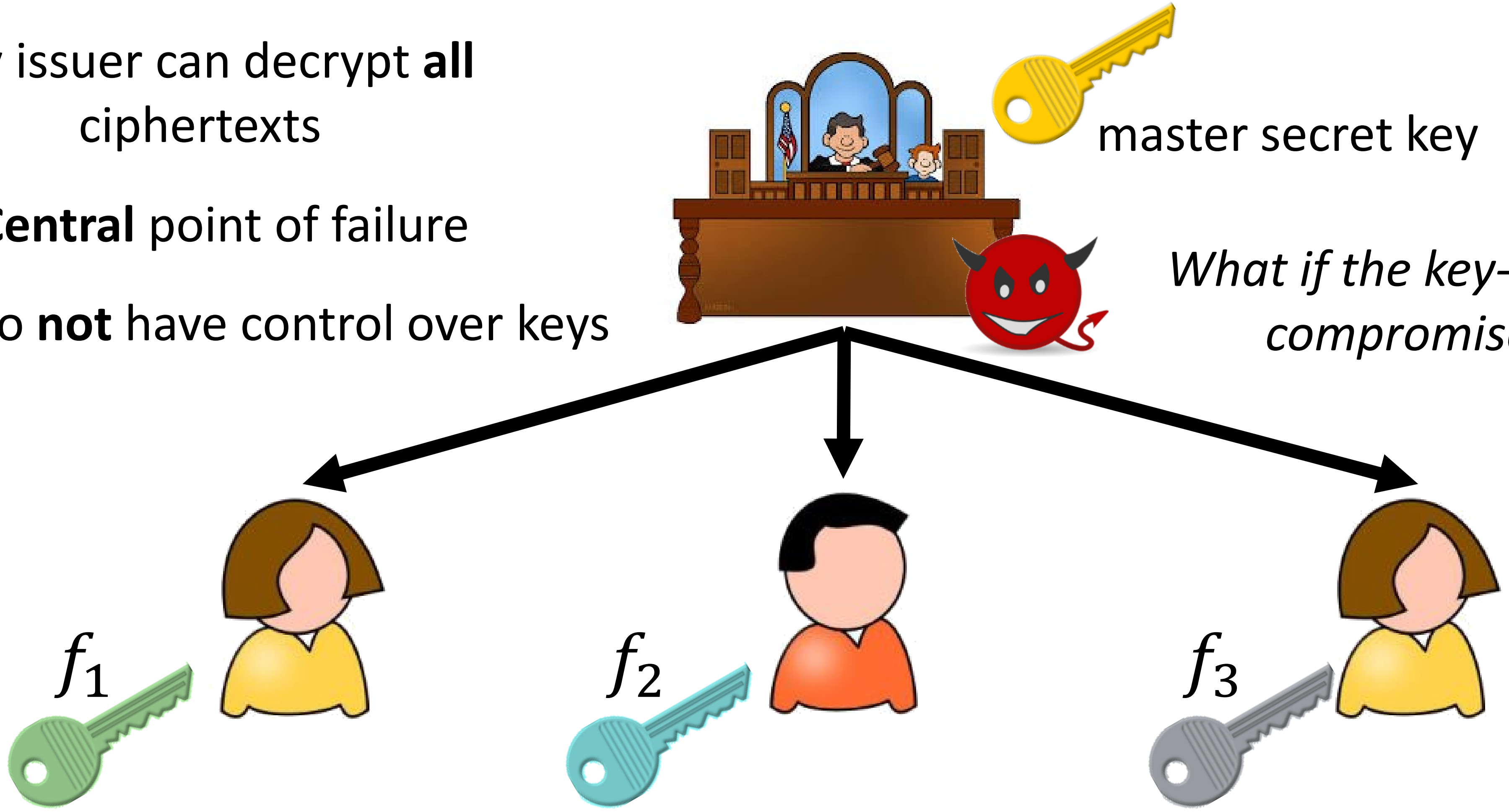
Functional Encryption (FE)

[SS10, O’N10, BSW11]

Key issuer can decrypt **all** ciphertexts

Central point of failure

Users do **not** have control over keys



master secret key

What if the key-issuer is compromised?

learns $f_1(x)$ learns $f_2(x)$ learns $f_3(x)$

Functional Encryption vs. Public-Key Encryption

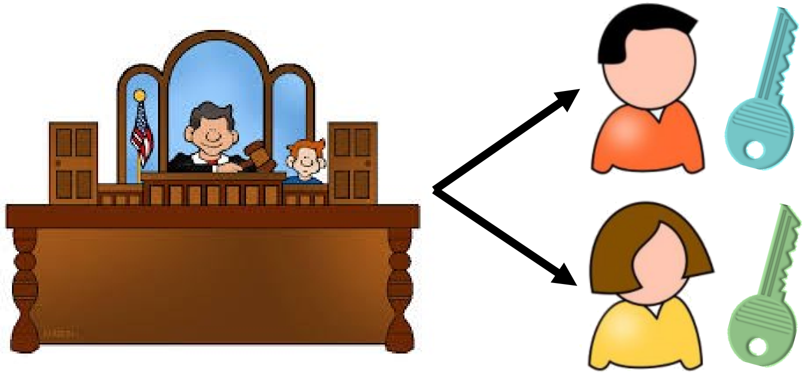
Public-key encryption is **decentralized**

Can we get the best of both worlds?

Every user generates their own key (no coordination or trust needed)

Does **not** support fine-grained decryption

Functional encryption is **centralized**



Central (trusted) authority generates individual keys

Supports **fine-grained** decryption capabilities

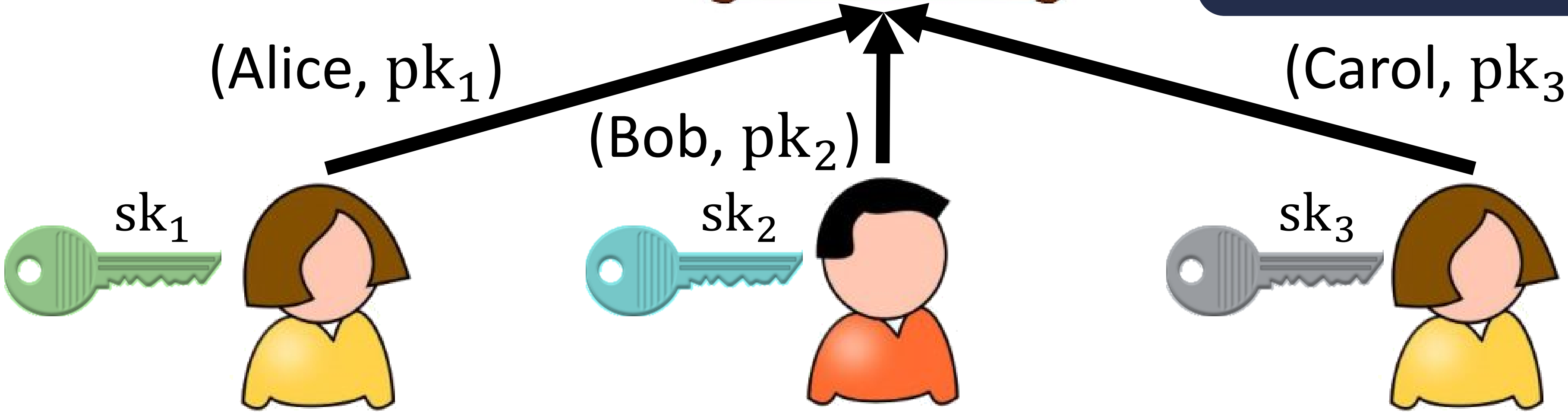
Registration-Based Encryption (RBE)

[GHMR18]

Key issuer replaced with **key curator**

Special case of **identity-based encryption** (IBE)

Decryption keys are associated with **identities**



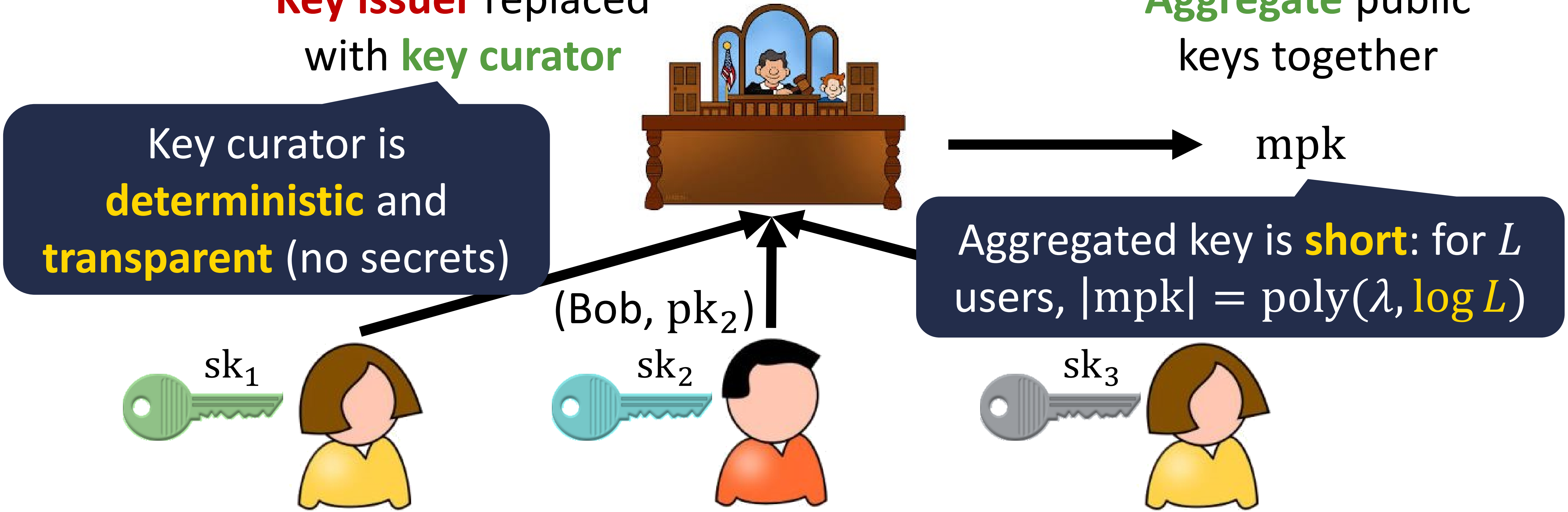
Users chooses their <u>own</u> public/secret key and **register** their public key with the curator

Registration-Based Encryption (RBE)

[GHMR18]

Key issuer replaced with **key curator**

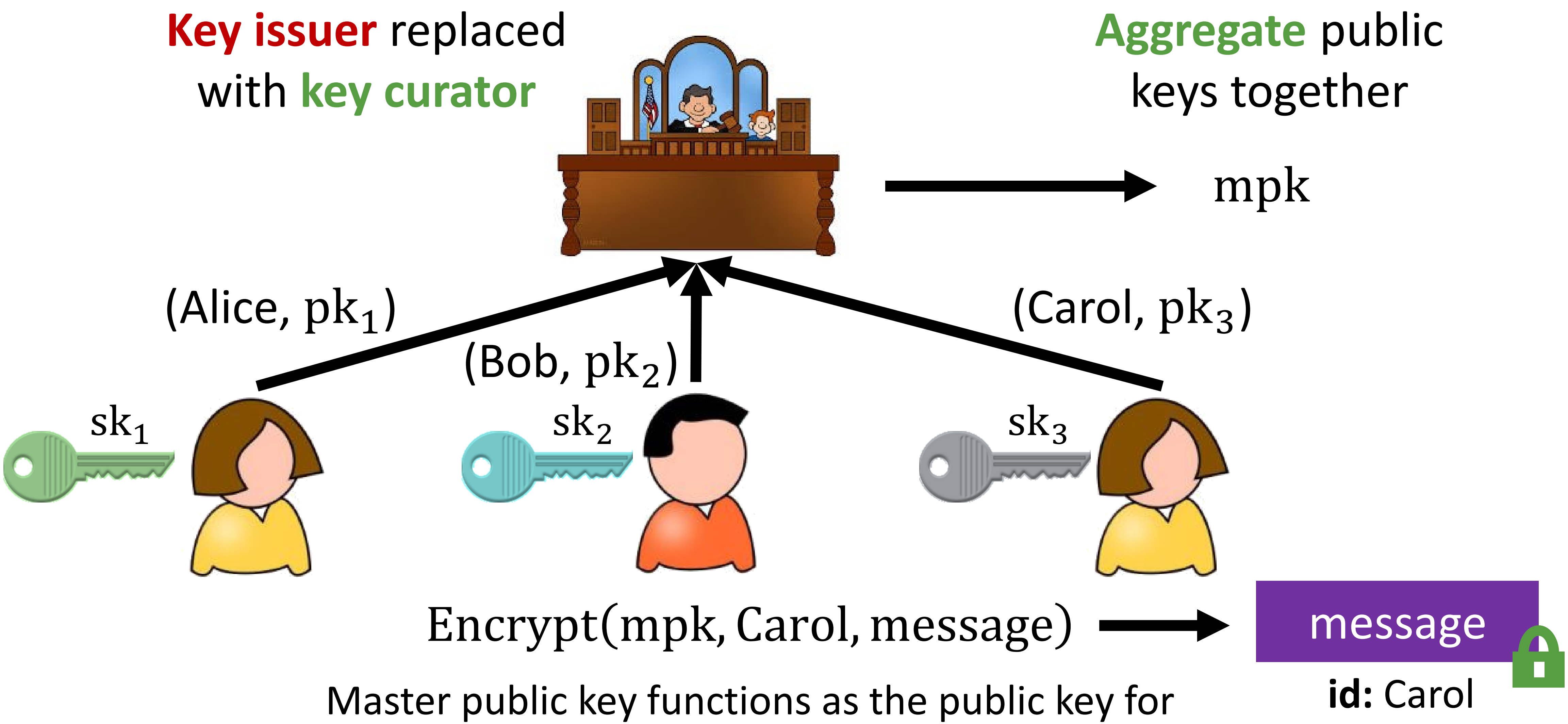
Aggregate public keys together



Users chooses their <u>own</u> public/secret key and **register** their public key with the curator

Registration-Based Encryption (RBE)

[GHMR18]



Master public key functions as the public key for an identity-based encryption scheme

Registration-Based Encryption (RBE)

[GHMR18]

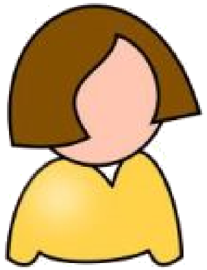
Key issuer replaced with **key curator**

Aggregate public keys together

$\longrightarrow \mathrm{mpk}$

$(\text{Alice}, \mathrm{pk})$

sk



hsk

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

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

Note: As users join, the master public key is updated, so users **occasionally** need to retrieve a new helper decryption key

$\#\text{ key updates per user} = \mathrm{poly}(\lambda, \log L)$

Registration-Based Encryption (RBE)

[GHMR18]

Key issuer replaced with **key curator**

Aggregate public keys together

$\longrightarrow$ mpk

- Initial 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]

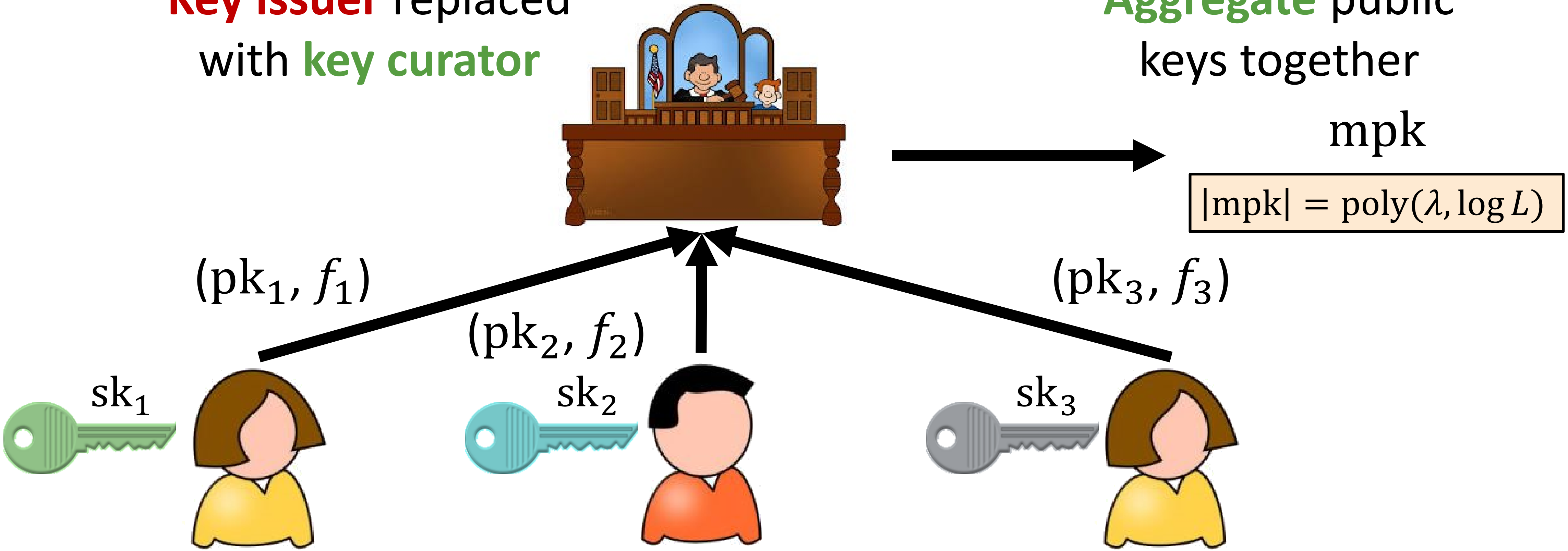
Can we construct RBE schemes that only need black-box use of cryptography?

Can we construct support more general policies (beyond identity-based encryption)?

Removing Trust from Functional Encryption

Key issuer replaced with **key curator**

Aggregate public keys together



Users chooses their own key and **register** the public key (together with **function f**) with the curator

Note: f could also be chosen by the key curator

Removing Trust from Functional Encryption

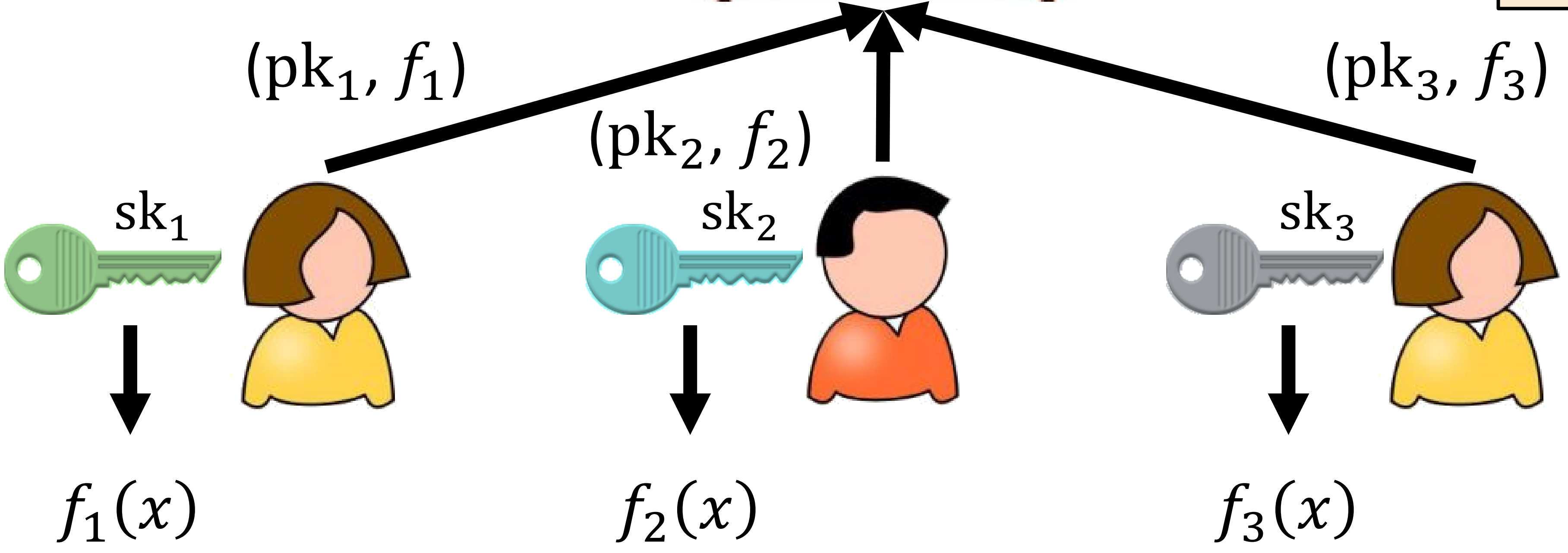
$\mathrm{Encrypt}(\mathrm{mpk}, x) \longrightarrow$ x

mpk is essentially a key for a functional encryption scheme

Aggregate public keys together

mpk

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



Registered Functional Encryption

Can we construct RBE schemes that only need black-box use of cryptography? **YES!**

Can we construct support more general policies (beyond identity-based encryption)? **YES!**

Registration-based encryption [GHMR18, GHMMRS19, GV20, CES21, DKLLMR23, GKMR23, ZZGQ23, FKP23]

Registered attribute-based encryption (ABE)

- Monotone Boolean formulas [HLWW23, ZZGQ23]
- Inner products [FFMMRV23, ZZGQ23]
- Arithmetic branching program [ZZGQ23]
- Boolean circuits [HLWW23, FWW23]

Lots of progress in this past year!

Distributed/flexible broadcast [BZ14, KMW23, FWW23, GLWW23]

Registered functional encryption

- Linear functions [DPY23]
- Boolean circuits [FFMMRV23, DPY23]

Underlined schemes only need **black-box** use of cryptography

Registered Functional Encryption

Can we construct RBE schemes that only need black-box use of cryptography? **Yes!**

Can we construct support more general policies (beyond identity-based encryption)? **Yes!**

Registration-based encryption [GHMR18, GHMMRS19, GV20, CES21, DKLLMR23, GKMR23, ZZGQ23, FKP23]

Registered attribute-based encryption (ABE)

- Monotone Boolean formulas [HLWW23, ZZGQ23]
- Inner products [FFMMRV23, ZZGQ23]
- Arithmetic branching program [ZZGQ23]
- Boolean circuits [HLWW23, FWW23]

This talk

Distributed/flexible broadcast [BZ14, KMW23, FWW23, GLWW23] **This talk**

Registered functional encryption

- Linear functions [DPY23]
- Boolean circuits [FFMMRV23, DPY23]

Lots of progress in this past year!

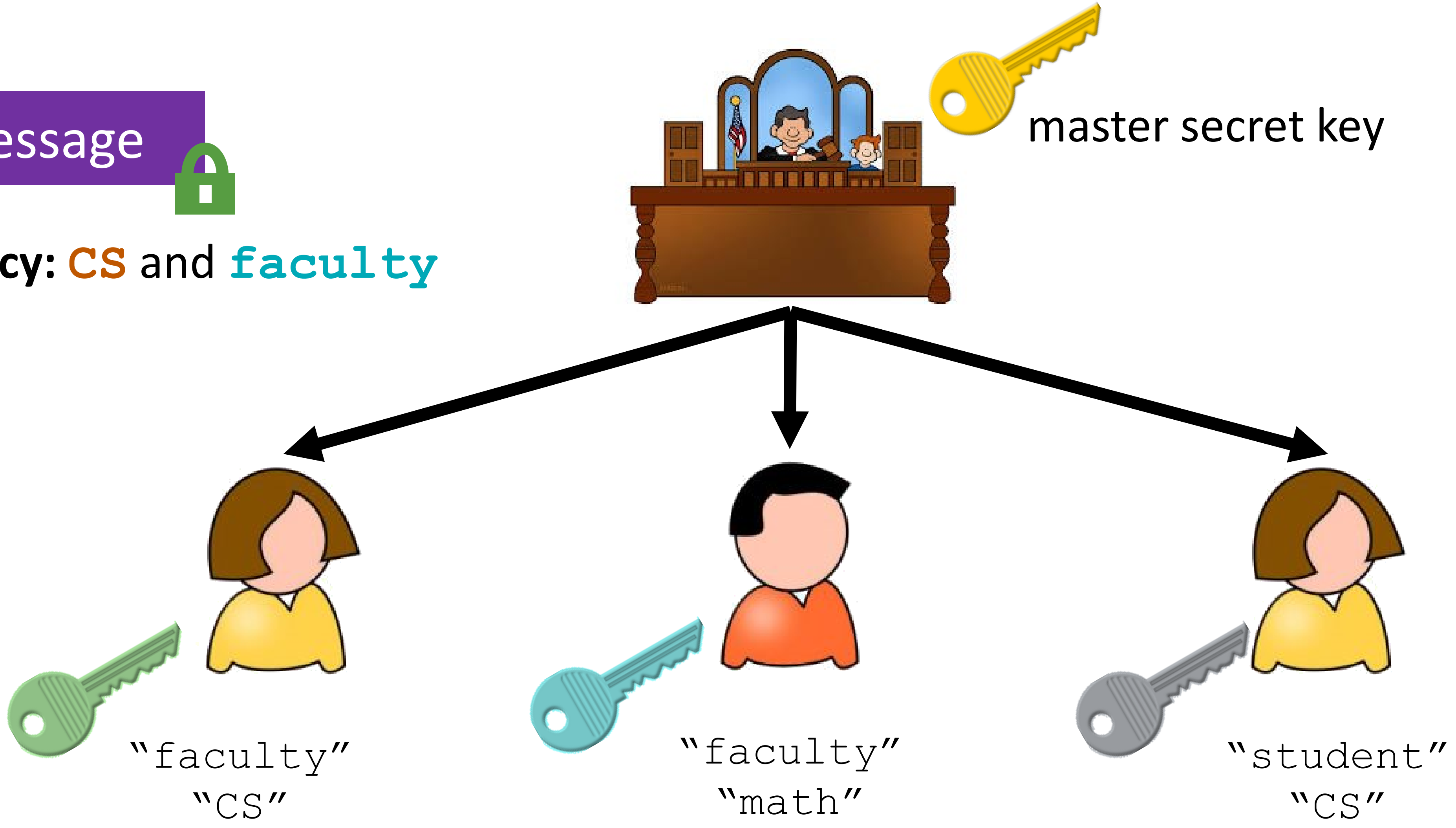
Underlined schemes only need **black-box** use of cryptography

Attribute-Based Encryption

[SW05, GPSW06]

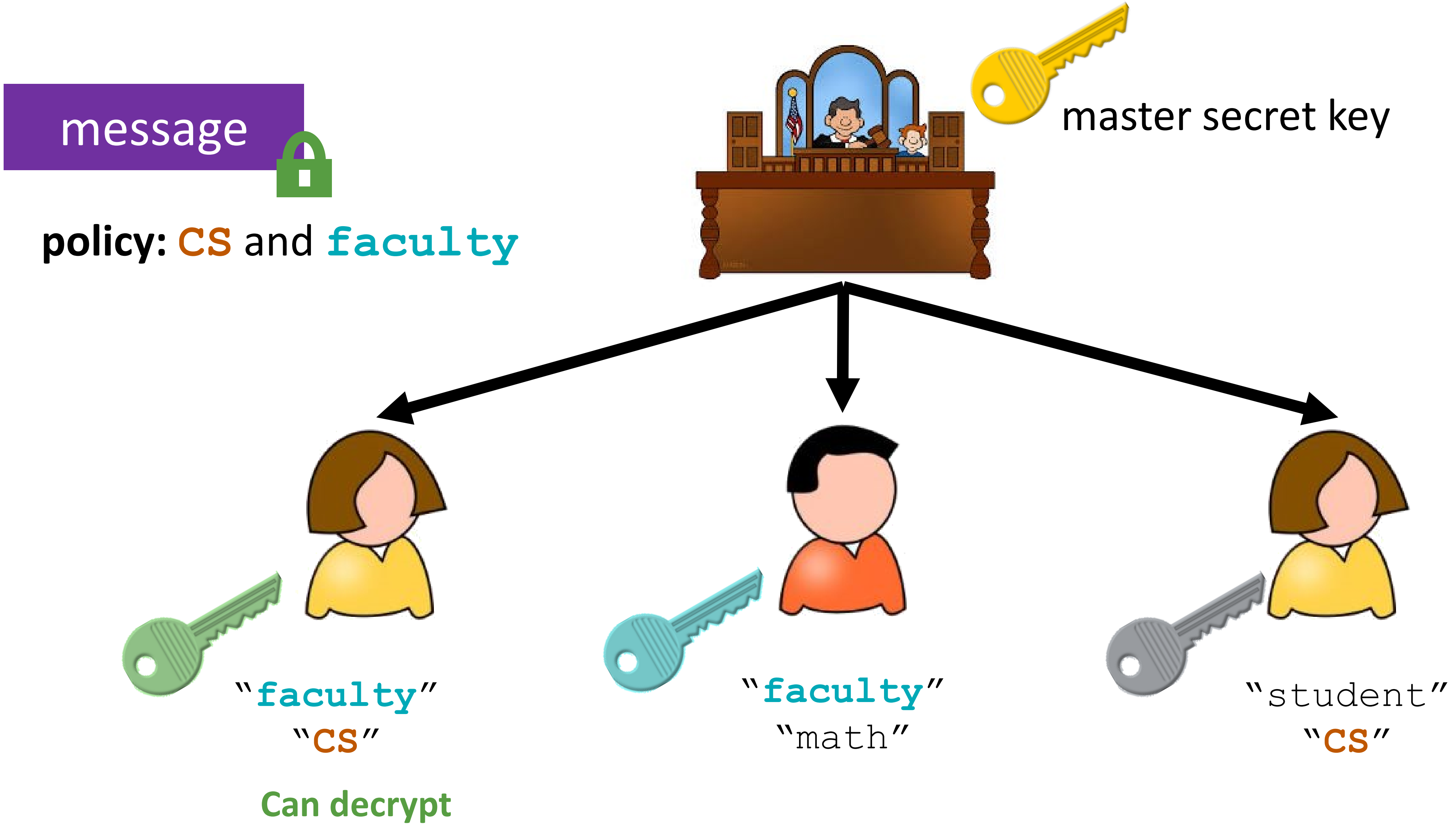
message

policy: `CS` and `faculty`



Attribute-Based Encryption

[SW05, GPSW06]

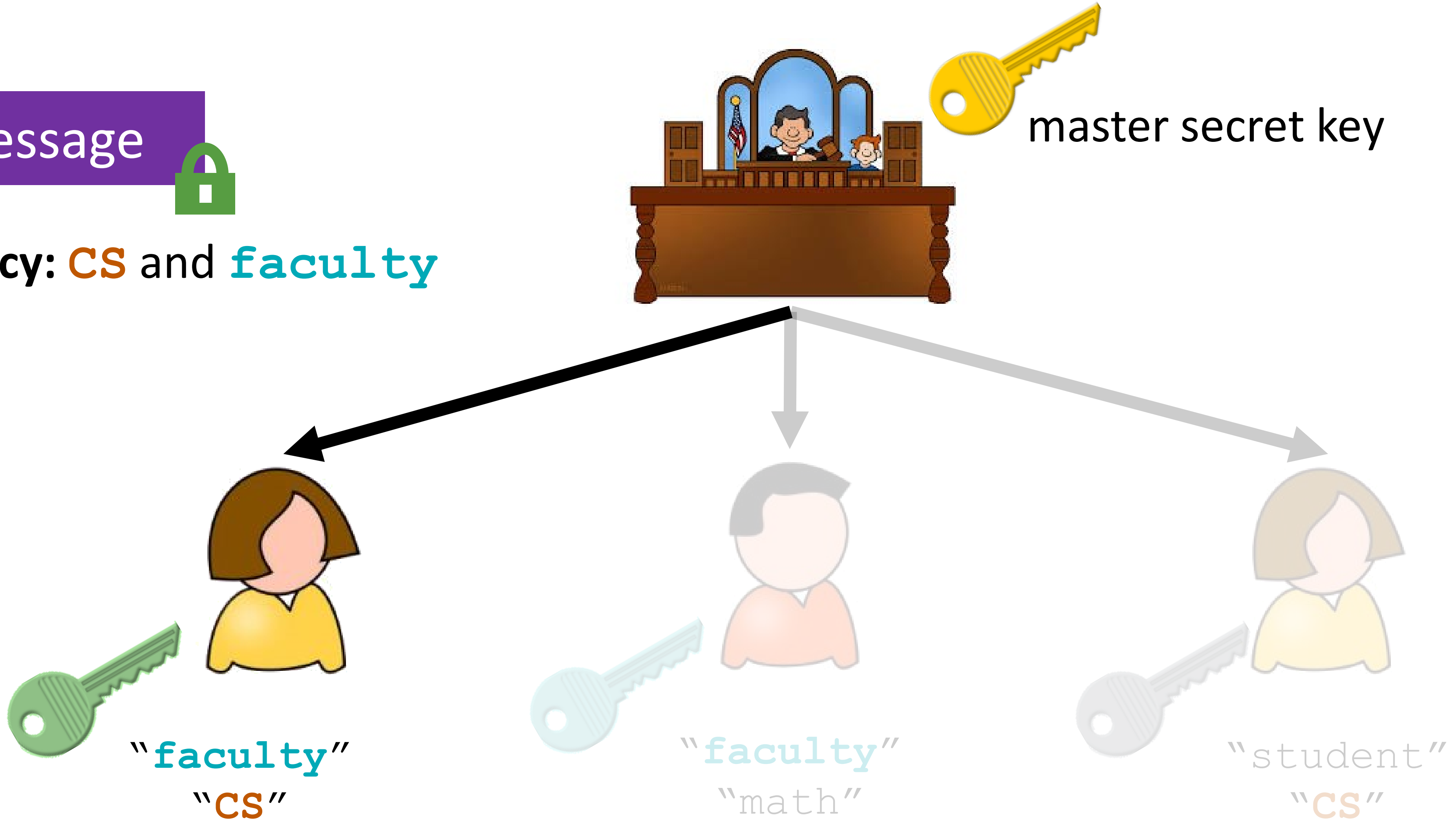


Attribute-Based Encryption

[SW05, GPSW06]

message

policy: CS and faculty



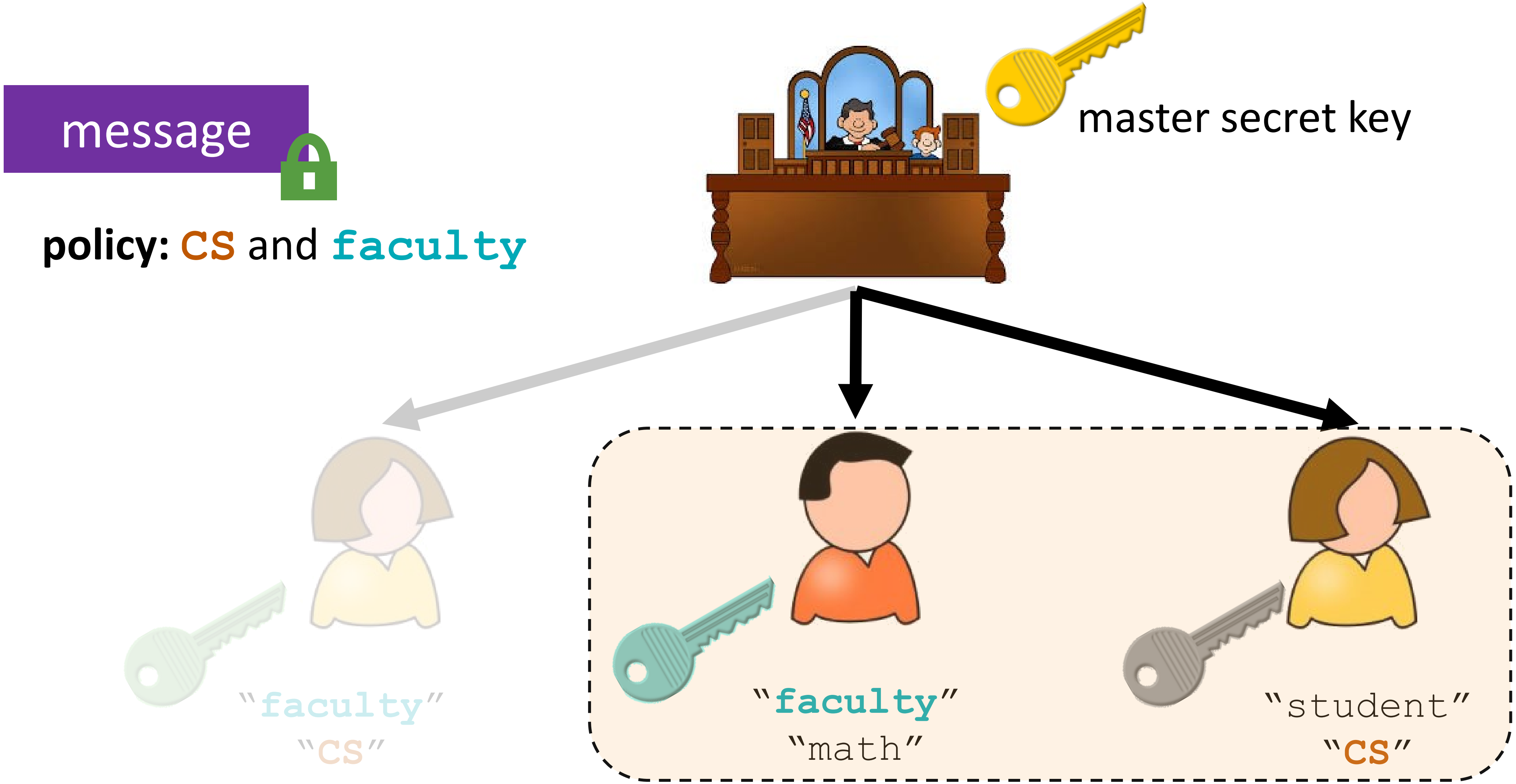
Can decrypt

Cannot decrypt

Cannot decrypt

Attribute-Based Encryption

[SW05, GPSW06]



Users cannot collude to decrypt

A Template for Building Registered ABE

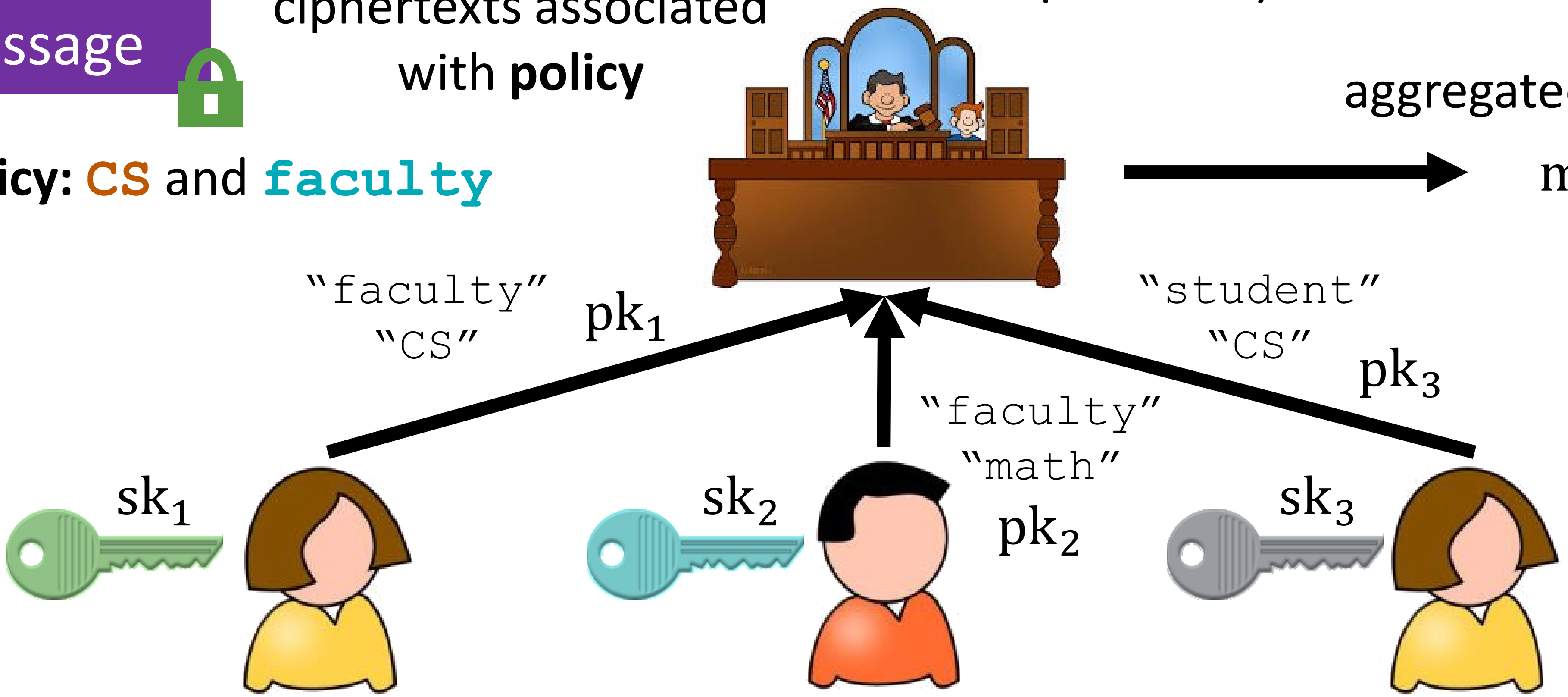
[HLWW23]

message

policy: **CS** and **faculty**

ciphertexts associated with **policy**

transparent key curator



aggregated public key

mpk

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

Users join the system by registering their public key along with a set of attributes

A Template for Building Registered ABE

[HLWW23]

Simplification: assume that all of the users register at the **same** time (rather than in an online fashion)

Slotted 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

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

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

A Template for Building Registered ABE

[HLWW23]

Simplification: assume that all of the users register at the same time (rather than in an online fashion)

Slotted 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

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

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

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

A Template for Building Registered ABE

[HLWW23]

Simplification: assume that all of the users register at the **same** time (rather than in an online fashion)

Slotted registered ABE:

Let L be the number of users

$$\boxed{\mathrm{pk}_1, S_1}\boxed{\mathrm{pk}_2, S_2}\boxed{\mathrm{pk}_3, S_3}\boxed{\mathrm{pk}_4, S_4}\boxed{\cdots}\boxed{\mathrm{pk}_L, S_L} \xrightarrow{\text{Aggregate}} \begin{matrix} \mathrm{mpk} \\ \mathrm{hsk}_1, \dots, \mathrm{hsk}_L \end{matrix}$$

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

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

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

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

Slotted Registered ABE to Registered ABE

[HLWW23]

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

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

Slotted scheme does *not* support online registration

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

Slotted Registered ABE to Registered ABE

[HLWW23]

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

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

Initially: all slots are empty

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

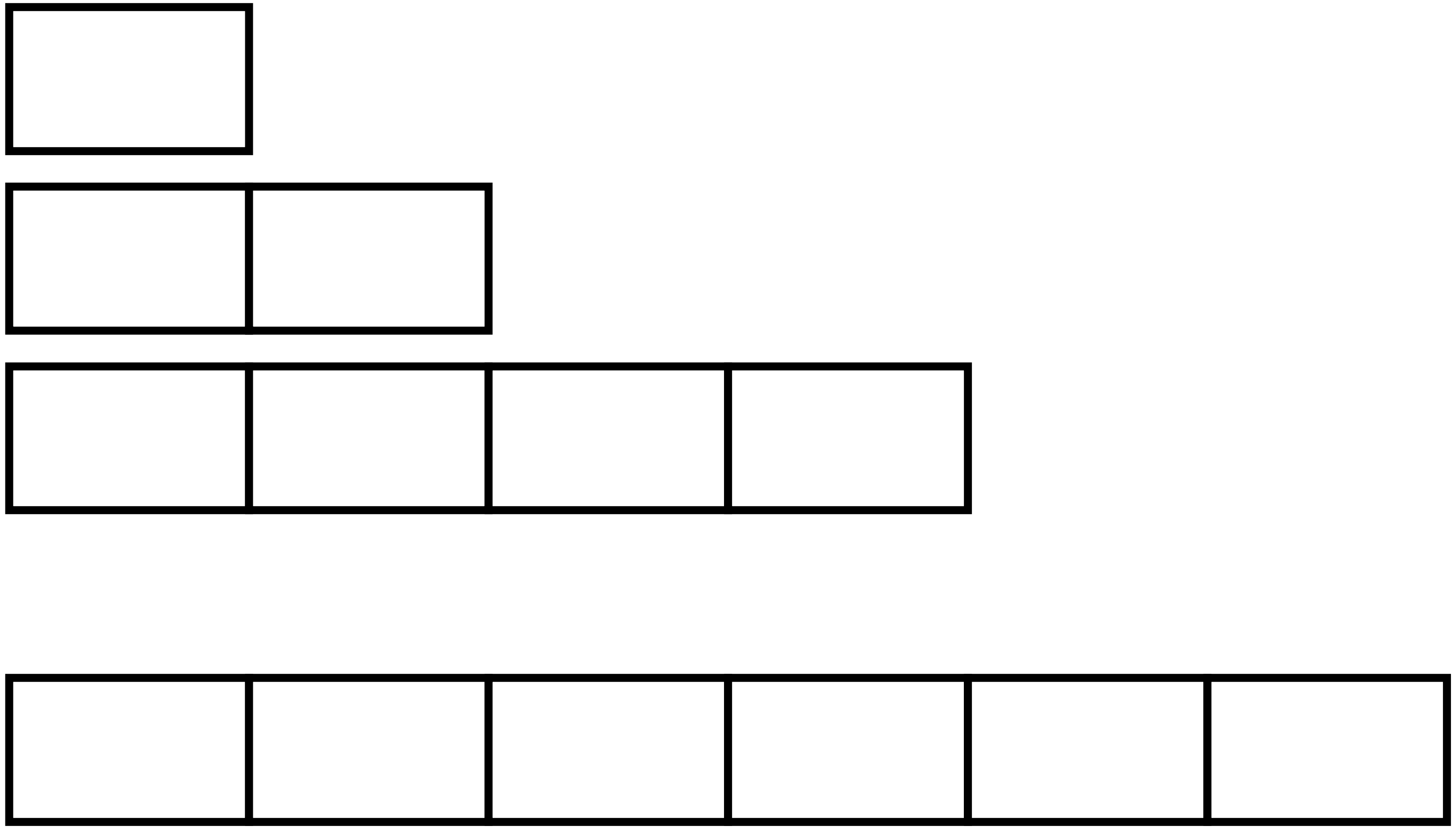
$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$\vdots$

$2^\ell = L$



Slotted Registered ABE to Registered ABE

[HLWW23]

Initially: all slots are empty

$\mathrm{mpk} = \perp$

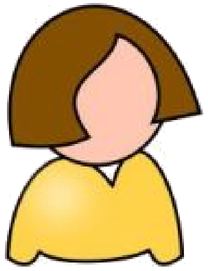
Solution: use "powers-of-two" approach (like [GHMR18])

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

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

Add key to each scheme with available slot

pk_1, S_1



Slotted Registered ABE to Registered ABE

[HLWW23]

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 | | | |

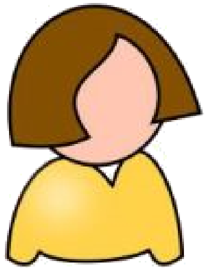
$\vdots$

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

Initially: all slots are empty

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

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



Slotted Registered ABE to Registered ABE

[HLWW23]

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 | $\xrightarrow{\text{all slots are full}}$ mpk_1

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

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

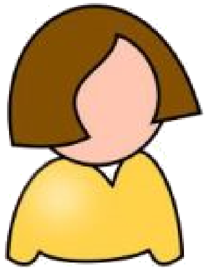
⋮

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

Initially: all slots are empty

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

$\uparrow$ pk_1, S_1



Slotted Registered ABE to Registered ABE

[HLWW23]

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

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

$2^0 = 1$ $\boxed{\mathrm{pk}_1, S_1}$ $\xrightarrow{\text{all slots are full}}$ mpk_1

$2^1 = 2$ $\boxed{\mathrm{pk}_1, S_1 \mid \mathrm{pk}_2, S_2}$

$2^2 = 4$ $\boxed{\mathrm{pk}_1, S_1 \mid \mathrm{pk}_2, S_2 \mid \quad \mid \quad}$

$\vdots$

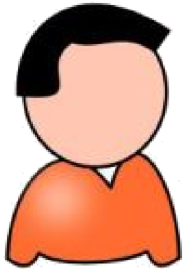
$2^\ell = L$ $\boxed{\mathrm{pk}_1, S_1 \mid \mathrm{pk}_2, S_2 \mid \quad \mid \quad \mid \quad \mid \quad}$

Initially: all slots are empty

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

Add key to each scheme with available slot

pk_2, S_2

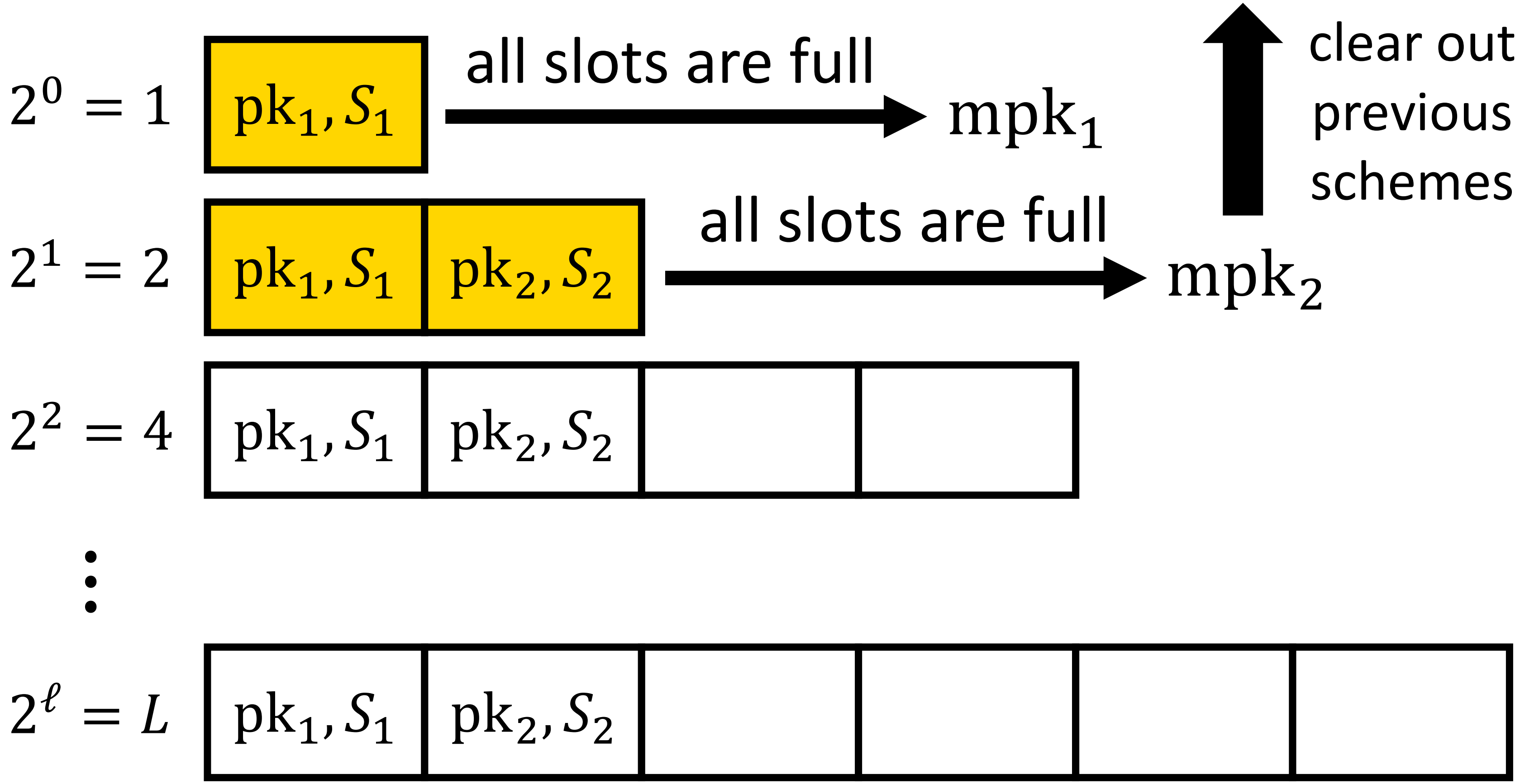


Slotted Registered ABE to Registered ABE

[HLWW23]

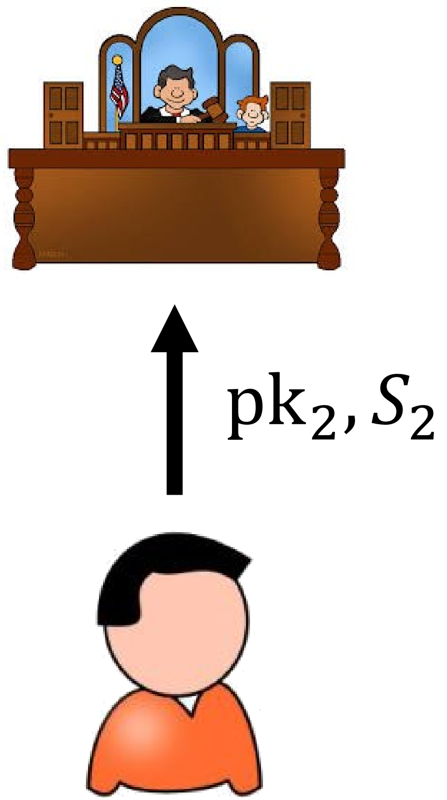
Solution: use “powers-of-two” approach (like [GHMR18])

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



Initially: all slots are empty

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



Slotted Registered ABE to Registered ABE

[HLWW23]

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				
$2^2 = 4$	pk_1, S_1	pk_2, S_2				
$\vdots$						
$2^\ell = L$	pk_1, S_1	pk_2, S_2				

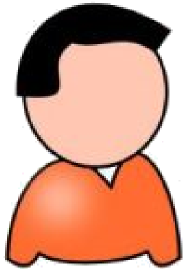
all slots are full $\longrightarrow \mathrm{mpk}_2$

clear out previous schemes

Initially: all slots are empty

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

pk_2, S_2



Slotted Registered ABE to Registered ABE

[HLWW23]

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				
$2^2 = 4$	pk_1, S_1	pk_2, S_2				
$\vdots$						
$2^\ell = L$	pk_1, S_1	pk_2, S_2				

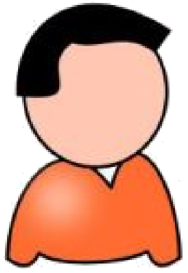
$\xrightarrow{\text{all slots are full}} \mathrm{mpk}_2$

clear out previous schemes

Initially: all slots are empty

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

pk_2, S_2



Slotted Registered ABE to Registered ABE

[HLWW23]

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

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

Initially: all slots are empty

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

$2^0 = 1$

pk_3, S_3

$2^1 = 2$

pk_1, S_1	pk_2, S_2

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

$2^2 = 4$

pk_1, S_1	pk_2, S_2	pk_3, S_3	

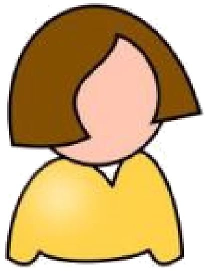
$\vdots$

$2^\ell = L$

pk_1, S_1	pk_2, S_2	pk_3, S_3			

Add key to each scheme with available slot

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



Slotted Registered ABE to Registered ABE

[HLWW23]

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			

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

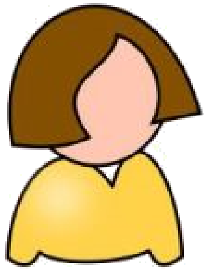
$\xrightarrow{\text{all slots are full}} \mathrm{mpk}_2$

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

[HLWW23]

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			

all slots are full $\longrightarrow \mathrm{mpk}_1$

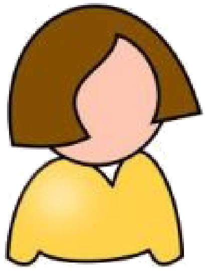
all slots are full $\longrightarrow \mathrm{mpk}_2$

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

[HLWW23]

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 | pk_4, S_4 |

$\vdots$

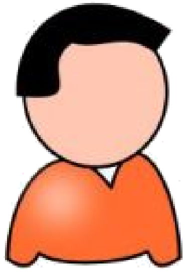
$2^\ell = L$: | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | pk_4, S_4 | | |

Initially: all slots are empty

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

Add key to each scheme with available slot

$\uparrow \mathrm{pk}_4, S_4$



Slotted Registered ABE to Registered ABE

[HLWW23]

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 $\rightarrow \mathrm{mpk}_1$

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

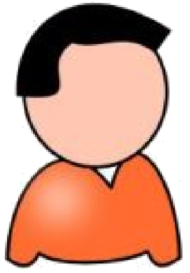
$2^2 = 4$ | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | pk_4, S_4 | all slots are full $\rightarrow \mathrm{mpk}_3$

$\vdots$

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

$\uparrow$ clear out previous schemes

$\uparrow$ pk_4, S_4



Slotted Registered ABE to Registered ABE

[HLWW23]

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

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

Initially: all slots are empty

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

$2^0 = 1$

$2^1 = 2$

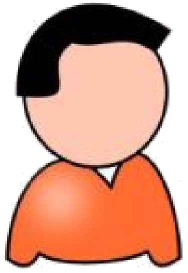
$2^2 = 4$ | pk_1, S_1 | pk_2, S_2 | pk_3, S_3 | pk_4, S_4 $\xrightarrow{\text{all slots are full}}$ mpk_3

$\vdots$

$2^\ell = L$ | 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

[HLWW23]

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

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

Initially: all slots are empty

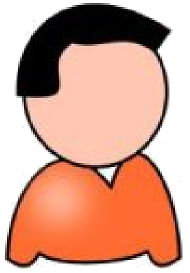
$\mathrm{mpk} = (\mathrm{mpk}_3)$

$2^0 = 1$						
$2^1 = 2$						
$2^2 = 4$	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

[HLWW23]

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

[HLWW23]

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

Constructing Slotted Registered ABE

Construction will rely on (composite-order) pairing groups $(\mathbb{G}, \mathbb{G}_T)$

Pairing is an **efficiently-computable** bilinear map $e: \mathbb{G} \to \mathbb{G}_T$ from $\mathbb{G}$ to $\mathbb{G}_T$:

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

Multiplies exponents in the ***target group***

Outline of Slotted Registered ABE

[HLWW23]

Scheme will rely on a **structured** common reference string (CRS)

Slot components: each slot $i \in [L]$ will have a set of associated group elements (denoted A_i)

A_1	A_2	A_3	A_4	$\cdots$	A_L

Attribute components: each attribute $w \in \mathcal{U}$ will have a group element U_w

User's individual public/secret key is an ElGamal key-pair

$$\mathrm{sk} = r, \quad \mathrm{pk} = g^r$$

Aggregated public key is just the product of every user's public key:

$$\mathrm{mpk} = \prod_{i \in [L]} g^{r_i}$$

Similar aggregation for attribute components

Outline of Slotted Registered ABE

[HLWW23]

Scheme will rely on a **structured** common reference string (CRS)

Slot components: each slot $i \in [L]$ will have a set of associated group elements (denoted A_i)

A_1	A_2	A_3	A_4	$\cdots$	A_L

Attribute components: each attribute $w \in \mathcal{U}$ will have a group element U_w

Decryption enforces the following two requirements:

Slot requirement: Decrypter know a secret key associated with the public key for some slot i^*

Attribute requirement: Attributes associated with slot i^* satisfy the decryption policy

In the construction, message is "blinded" by $v_1 v_2$, where v_1 can be computed with knowledge of a secret key associated with a slot i^* and v_2 can be computed if the attributes for slot i^* satisfy the policy

Outline of Slotted Registered ABE

[HLWW23]

Scheme will rely on a **structured** common reference string (CRS)

Slot components: each slot $i \in [L]$ will have a set of associated group elements (denoted A_i)

A_1	A_2	A_3	A_4	$\cdots$	A_L

Attribute components: each attribute $w \in \mathcal{U}$ will have a group element U_w

ving two requirements:

v a secret key associated with the public key for some slot i^*

s associated with slot i^* satisfy the decryption policy

Need to be careful to defend against collusions

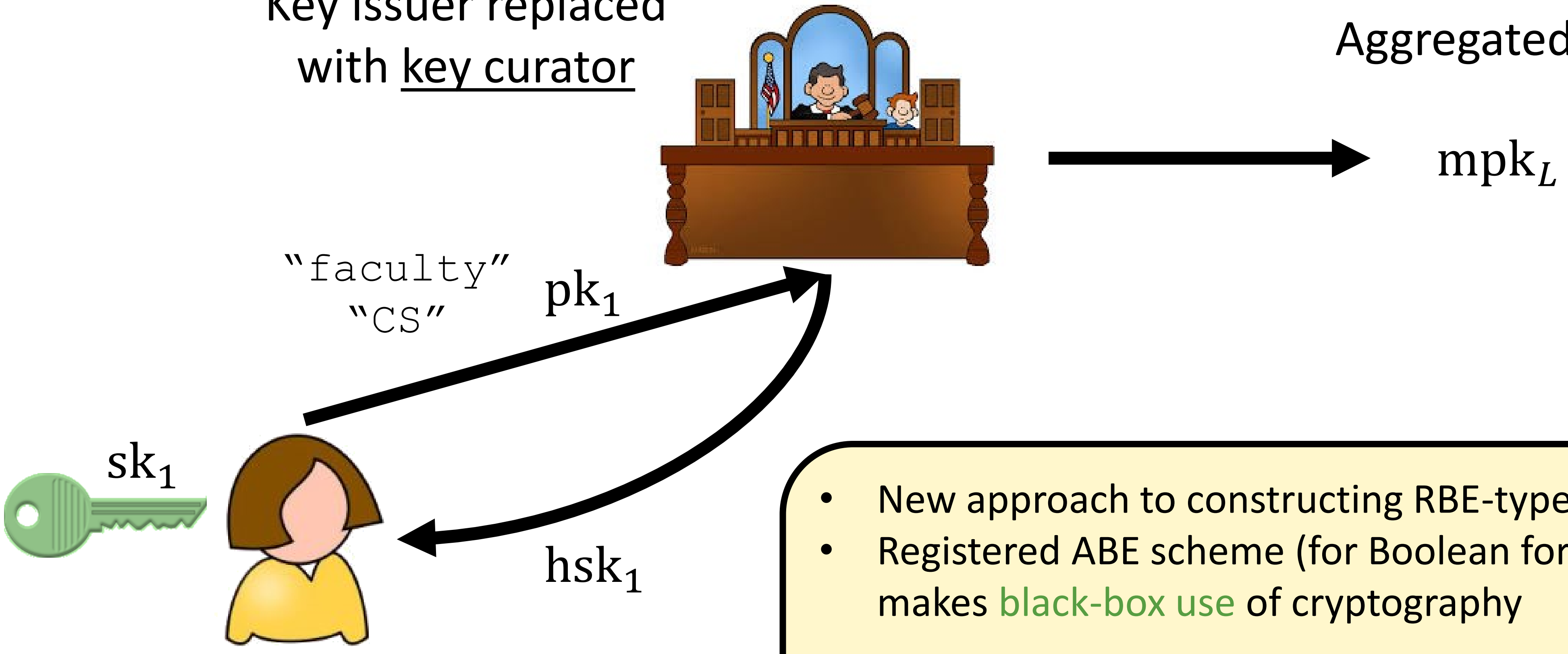
[see paper for details]

In the construction, message is "blinded" by $v_1 v_2$, where v_1 can be computed with knowledge of a secret key associated with a slot i^* and v_2 can be computed if the attributes for slot i^* satisfy the policy

Registered ABE Summary

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

Aggregated key



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

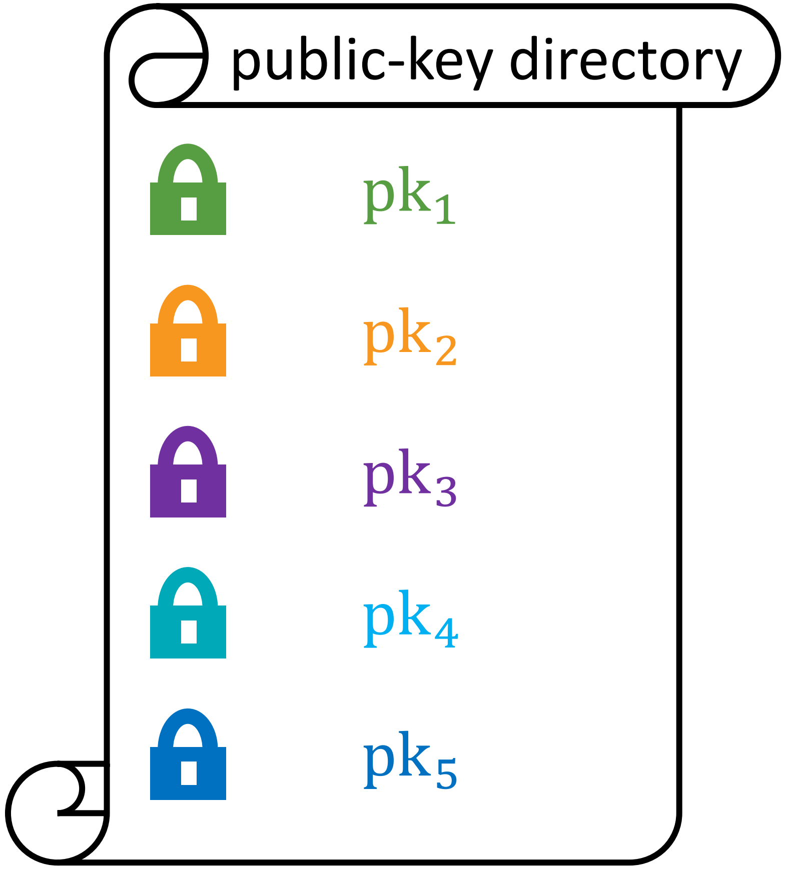
- New approach to constructing RBE-type of primitives
- Registered ABE scheme (for Boolean formulas) only makes black-box use of cryptography

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

An Application to Broadcast Encryption

[FWW23]

Registered ABE is a useful building block for other **trustless** cryptographic systems



Independent, user-generated keys

Suppose we want to encrypt a message to $\{\mathrm{pk}_1, \mathrm{pk}_3, \mathrm{pk}_4\}$

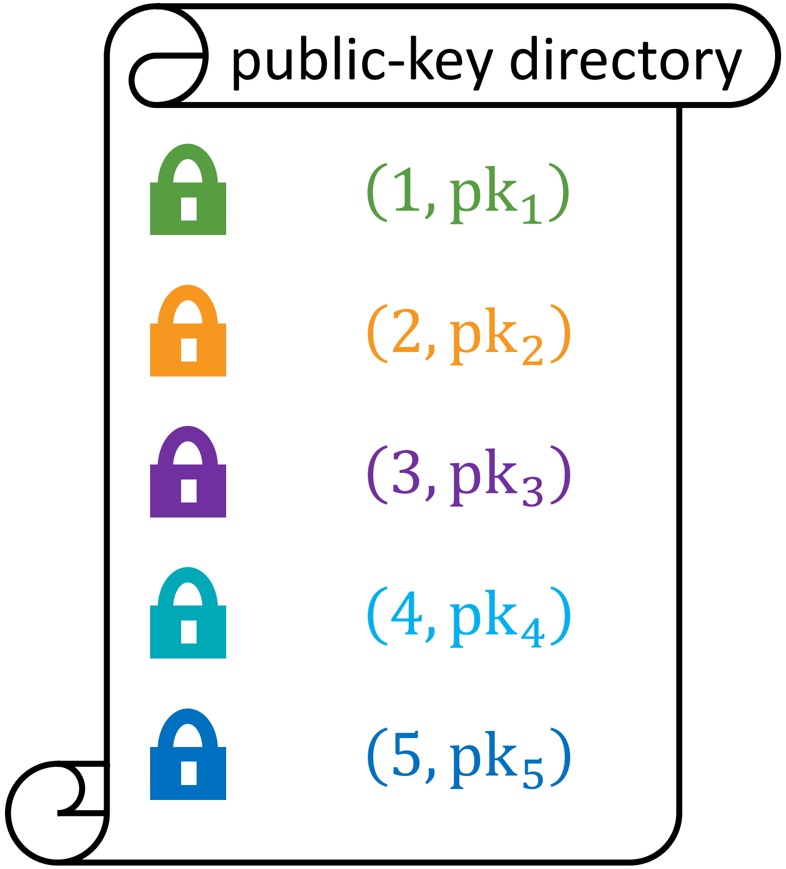
Public-key encryption: ciphertext size grows with the size of the set

Broadcast encryption: achieve *sublinear* ciphertext size, but requires central authority

An Application to Broadcast Encryption

[FWW23]

Distributed broadcast encryption [BZ14]



Each user chooses its own public key, and each key has a **unique** index

$$\mathrm{Encrypt}(\mathrm{pp}, \{\mathrm{pk}_i\}_{i \in S}, m) \to \mathrm{ct}$$

Can encrypt a message m to any set of public keys

Efficiency: $|\mathrm{ct}| = |m| + \mathrm{poly}(\lambda, \log|S|)$

$$\mathrm{Decrypt}(\mathrm{pp}, \{\mathrm{pk}_i\}_{i \in S}, \mathrm{sk}, \mathrm{ct}) \to m$$

Any secret key associated with broadcast set can decrypt

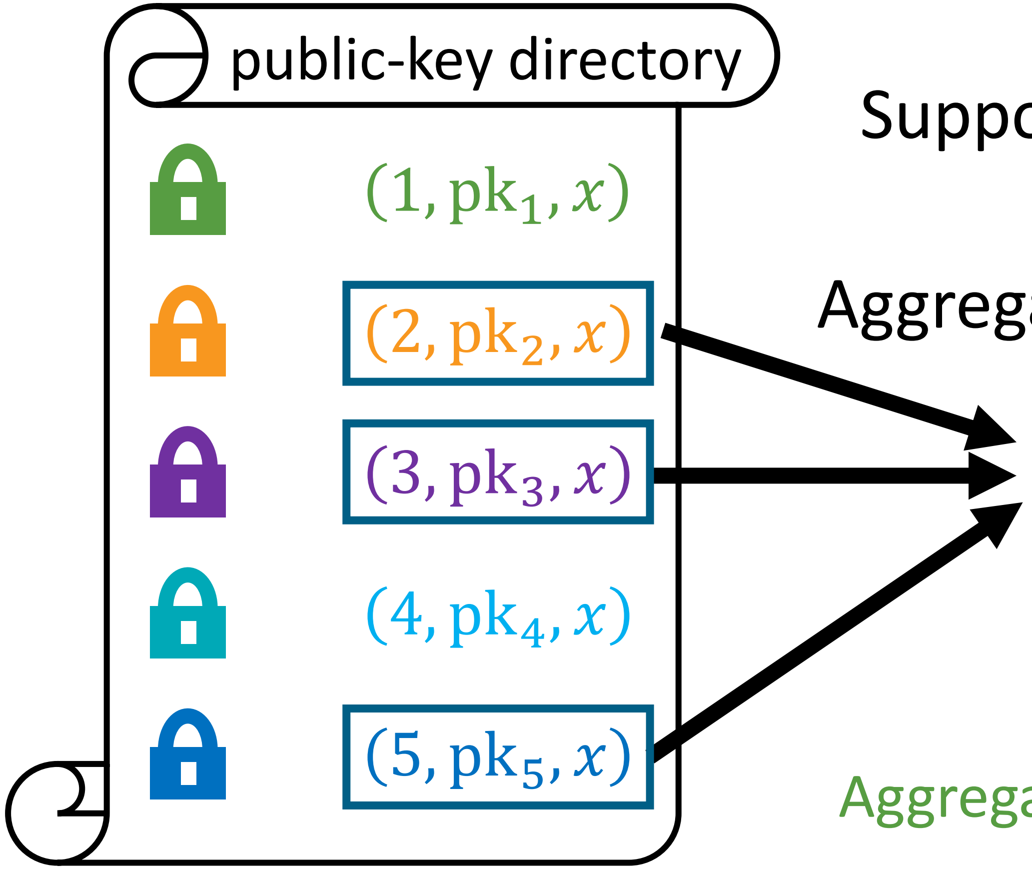
Decryption does requires knowledge of public keys in broadcast set

Distributed Broadcast from Slotted Registered ABE

[FWW23]

Consider a registered ABE scheme with a single dummy attribute x

Public key for an index i is a key for **slot i** with **attribute** x



Suppose we want to encrypt to a set $S = \{2,3,5\}$

Aggregate

$\mathrm{mpk} \longrightarrow \mathrm{Encrypt}(\mathrm{mpk}, x, P)$

Encrypt with respect to mpk to policy P that accepts x

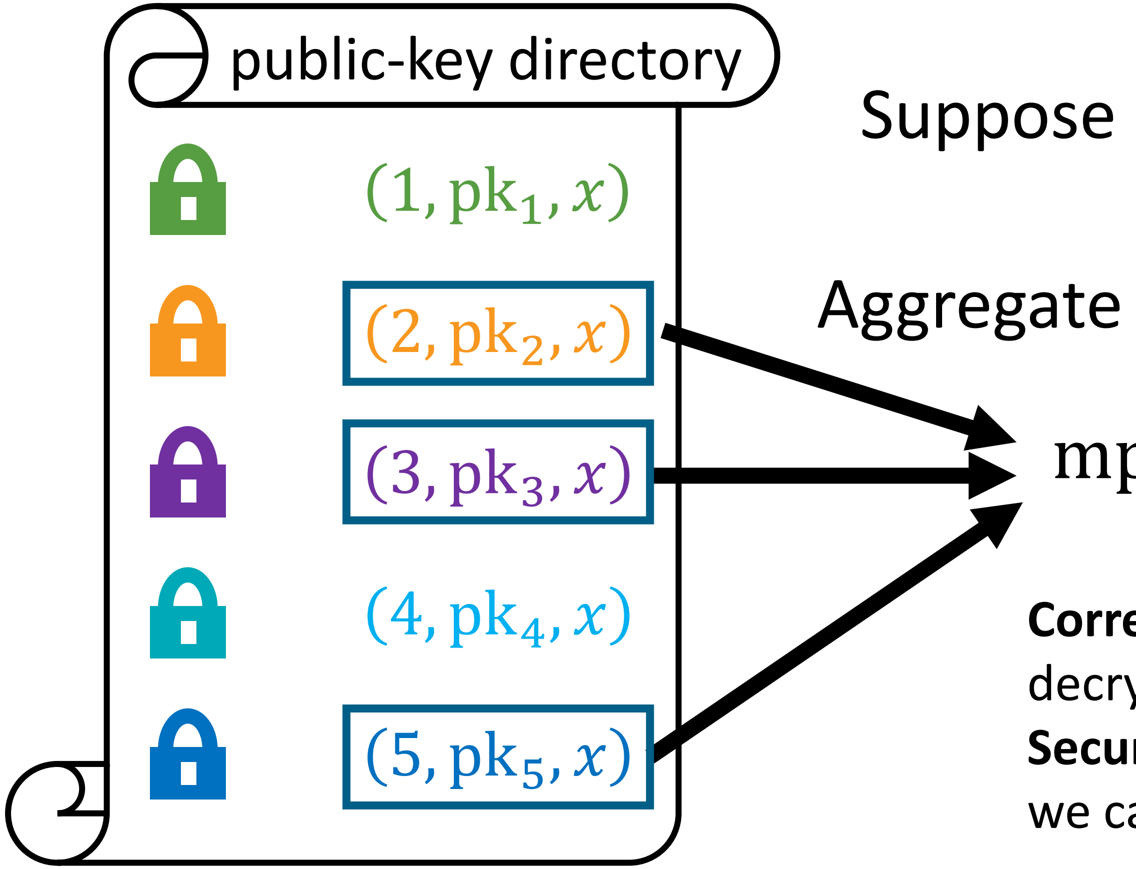
Aggregate public keys using slotted registered ABE scheme

Distributed Broadcast from Slotted Registered ABE

[FWW23]

Consider a registered ABE scheme with a single dummy attribute x

Public key for an index i is a key for **slot i** with **attribute** x



Suppose we want to encrypt to a set $S = \{2,3,5\}$

Encrypt with respect to mpk to policy P that accepts x

$\mathrm{mpk} \longrightarrow \mathrm{Encrypt}(\mathrm{mpk}, x, P)$

Correctness: If $i \in S$, then (i, pk_i, x) was aggregated in mpk so decryption is possible using sk_i

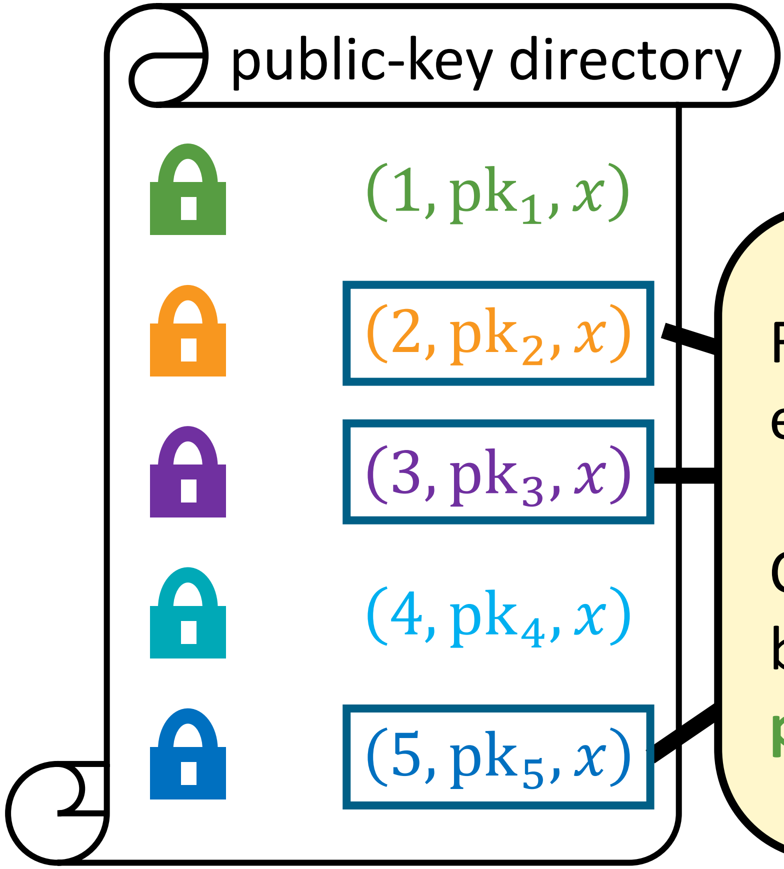
Security: If $i \notin S$, then (i, pk_i, x) was **not** aggregated in mpk so we can appeal to security of registered ABE

Distributed Broadcast from Slotted Registered ABE

[FWW23]

Consider a registered ABE scheme with a single dummy attribute x

Public key for an index i is a key for **slot i** with **attribute** x



Suppose we want to encrypt to a set $S = \{2,3,5\}$

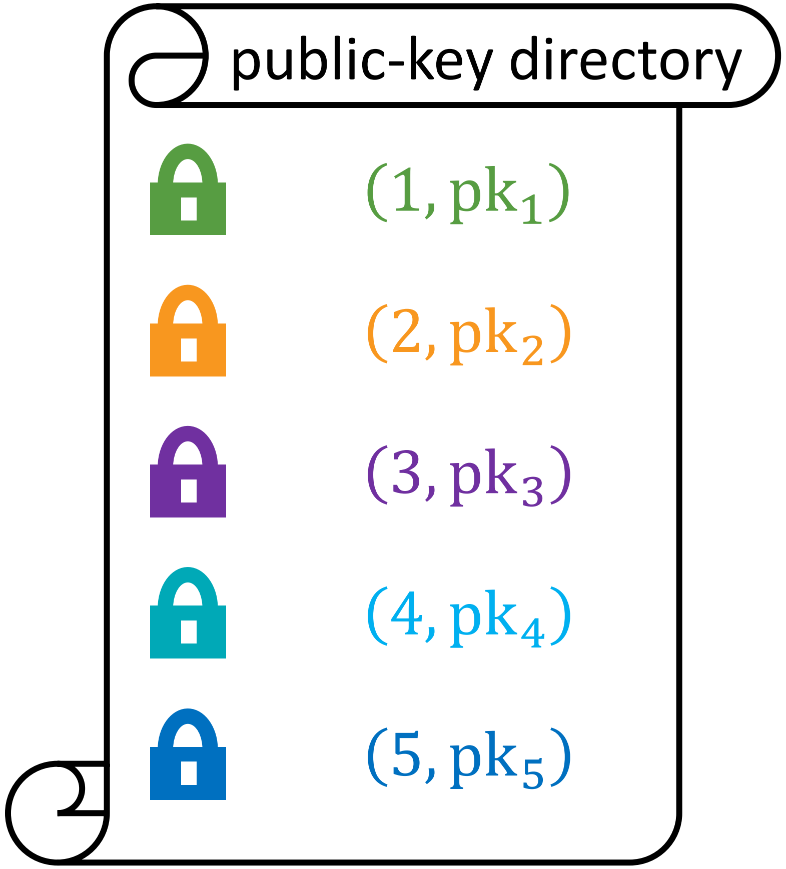
Registered ABE [HLWW23] + compiler ⇒ distributed broadcast encryption from **composite-order** pairing groups

Concurrent work [KMW23]: show how to adapt a centralized broadcast encryption scheme into a distributed one from **prime-order** pairing groups

Flexible Broadcast Encryption

[FWW23, GLWW23]

Distributed broadcast encryption still requires **some** coordination

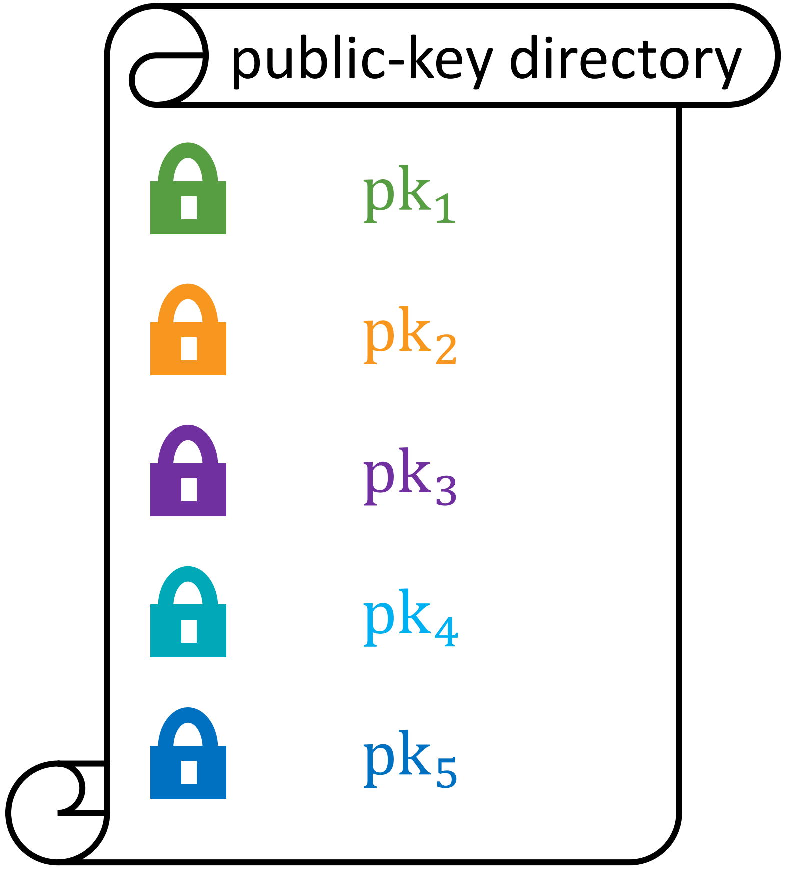


Users have to generate public keys for **distinct** slots (for correctness), so public-key directory needs to be **centralized**

Flexible Broadcast Encryption

[FWW23, GLWW23]

Distributed broadcast encryption still requires **some** coordination



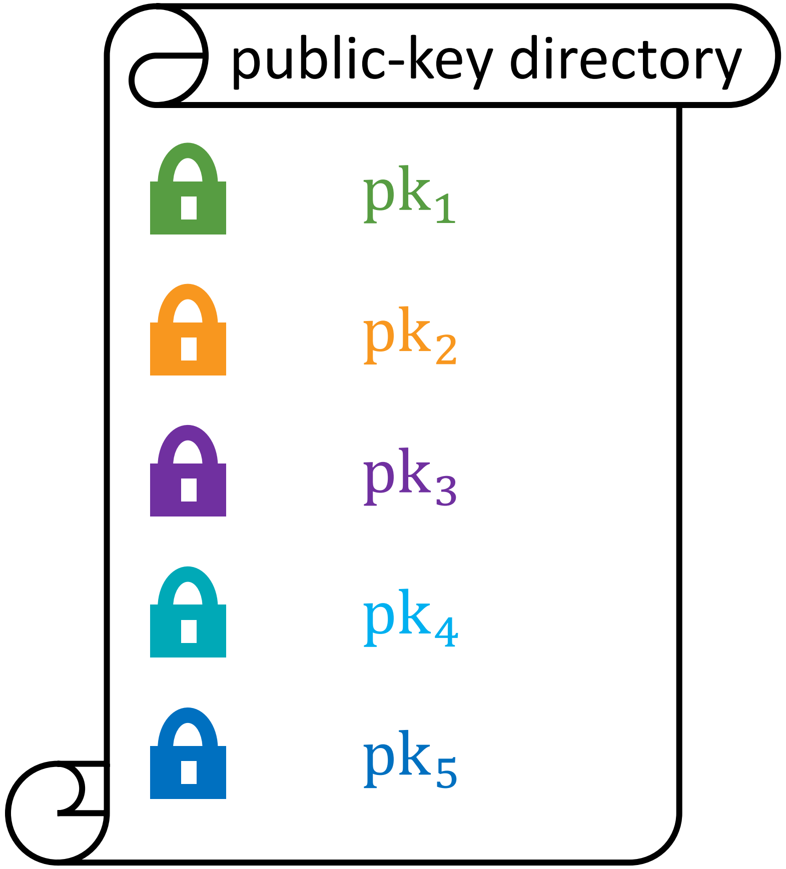
Users have to generate public keys for **distinct** slots (for correctness), so public-key directory needs to be **centralized**

Flexible broadcast encryption: no notion of slots, can encrypt to an *arbitrary* set of public keys

Flexible Broadcast Encryption

[FWW23, GLWW23]

Distributed broadcast encryption still requires **some** coordination



$$\mathrm{Encrypt}(\mathrm{pp}, \{\mathrm{pk}_i\}_{i \in S}, m) \to \mathrm{ct}$$

Can encrypt a message m to any set of public keys

Efficiency: $|\mathrm{ct}| = |m| + \mathrm{poly}(\lambda, \log|S|)$

$$\mathrm{Decrypt}(\mathrm{pp}, \{\mathrm{pk}_i\}_{i \in S}, \mathrm{sk}, \mathrm{ct}) \to m$$

Any secret key associated with broadcast set can decrypt

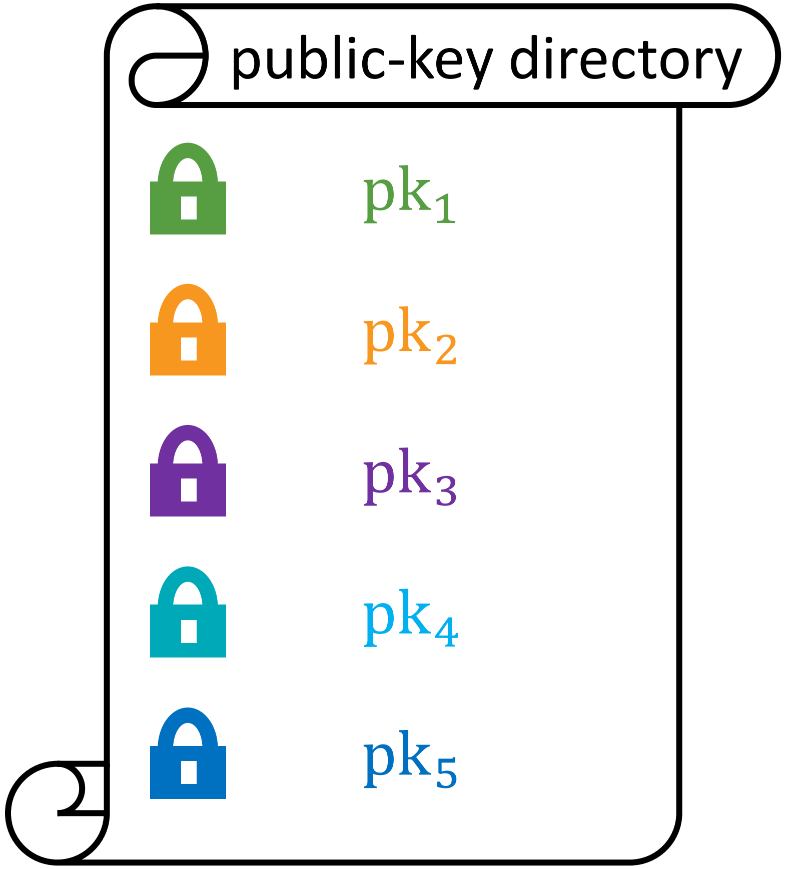
Decryption does requires knowledge of public keys in broadcast set

Flexible Broadcast Encryption

[FWW23, GLWW23]

Distributed broadcast encryptic … **some** coordination

public parameters



$$\mathrm{Encrypt}(\mathrm{pp}, \{\mathrm{pk}_i\}_{i \in S}, m) \to \mathrm{ct}$$

Can encrypt a message m to any set of public keys

Efficiency: $|\mathrm{ct}| = |m| + \mathrm{poly}(\lambda, \log|S|)$

$$\mathrm{Decrypt}(\mathrm{pp}, \{\mathrm{pk}_i\}_{i \in S}, \mathrm{sk}, \mathrm{ct}) \to m$$

Any secret key associated with broadcast set can decrypt

Decryption does requires knowledge of public keys in broadcast set

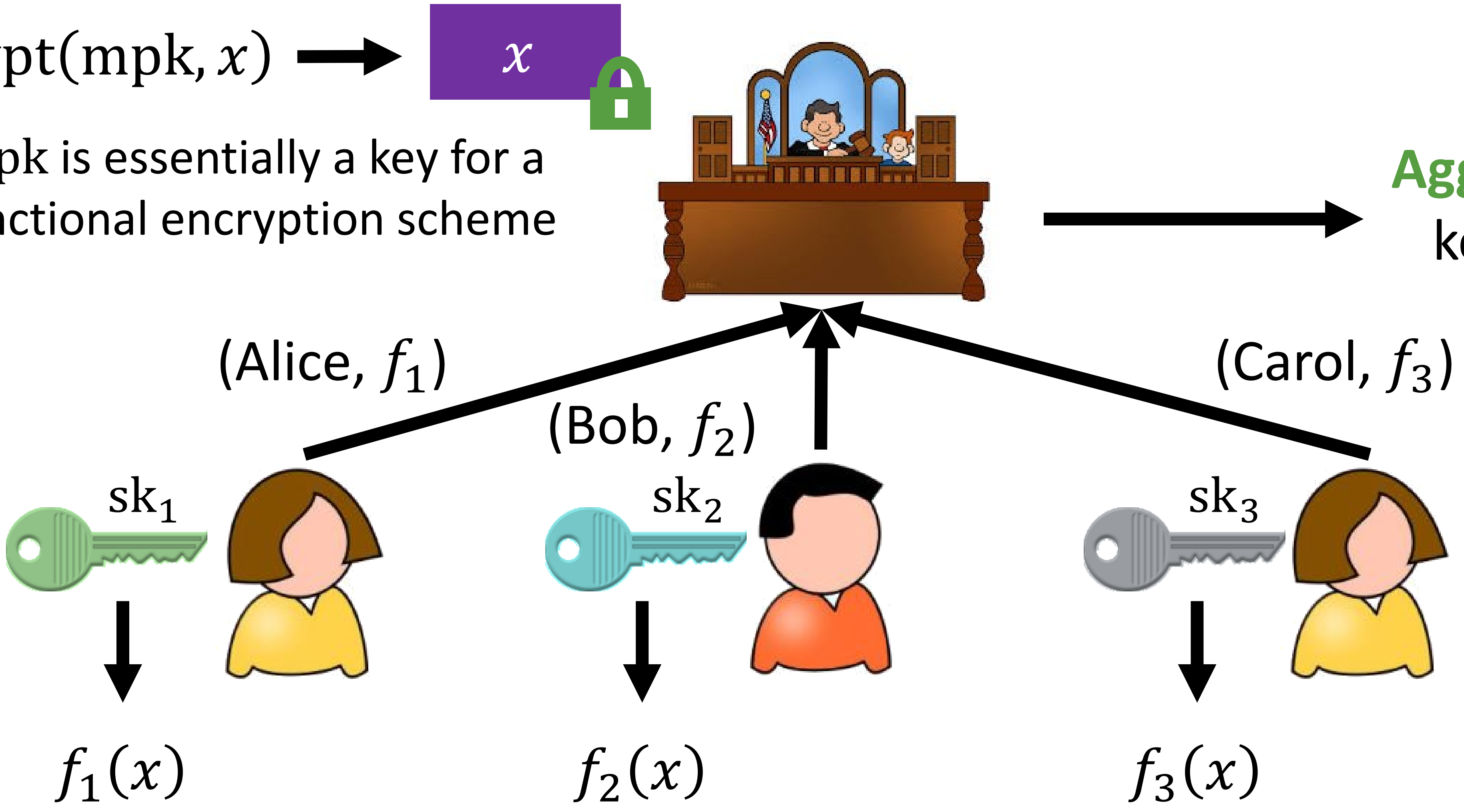
[GLWW23]: **distributed** broadcast encryption ⇒ **flexible** broadcast encryption

Removing Trust from Functional Encryption

$\text{Encrypt}(\text{mpk}, x) \longrightarrow$

mpk is essentially a key for a functional encryption scheme

Aggregate public keys together



Goal: Support capabilities of functional encryption **without** a trusted authority

Open Problems

Schemes with **short** CRS or **unstructured** CRS without non-black-box use of cryptography

- Existing constructions have long structured CRS (typically quadratic in the number of users)

Lattice-based constructions of registered FE (and special cases of FE)

- Registration-based encryption known from LWE [DKLLMR23]
- Registered ABE for circuits known from evasive LWE (via witness encryption) [FWW23]

Key revocation and verifiability

- Defending against possibly malicious adversaries

Improve concrete efficiency for registered FE schemes

- Current bottlenecks include large CRS and large public keys

Thank you!

References

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional Encryption: Definitions and Challenges. TCC 2011.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation. CRYPTO 2014.

[CES21] Kelong Cong, Karim Eldefrawy, and Nigel P. Smart. Optimizing Registration Based Encryption. IMACC 2021.

[DKLLMR23] Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient Laconic Cryptography from Learning with Errors. EUROCRYPT 2023.

[DPY23] Pratish Datta, Tapas Pal, and Shota Yamada. Registered FE Beyond Predicates: (Attribute-Based) Linear Functions and More. 2023.

[FFMMRV23] Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (Inner-Product) Functional Encryption. ASIACRYPT 2023.

[FKP23] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo Commitments: Registration-Based Encryption and Key-Value Map Commitments for Large Spaces. ASIACRYPT 2023.

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to Use (Plain) Witness Encryption: Registered ABE, Flexible Broadcast, and More. CRYPTO 2023.

References

[GHMMRS19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-Based Encryption from Standard Assumptions. PKC 2019.

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-Based Encryption: Removing Private-Key Generator from IBE. TCC 2018.

[GKMR23] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient Registration-Based Encryption. ACM CCS 2023.

[GLWW23] Rachit Garg, George Lu, Brent Waters, and David J. Wu. Realizing Flexible Broadcast Encryption: How to Broadcast to a Public-Key Directory. ACM CCS 2023.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. ACM CCS 2006

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable Registration-Based Encryption. CRYPTO 2020.

[HLWW23] Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered Attribute-Based Encryption. EUROCRYPT 2023.

[KMW23] Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed Broadcast Encryption from Bilinear Groups. ASIACRYPT 2023.

References

[O'N10] Adam O'Neill. Definitional Issues in Functional Encryption. 2010.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-Free Encryption: Functional Encryption with Public Keys. ACM CCS 2010.

[SW05] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. EUROCRYPT 2005.

[ZZGQ23] Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered ABE via Predicate Encodings. ASIACRYPT 2023.