Access Control Encryption for General Policies from Standard Assumptions

Sam Kim David J. Wu

Stanford University

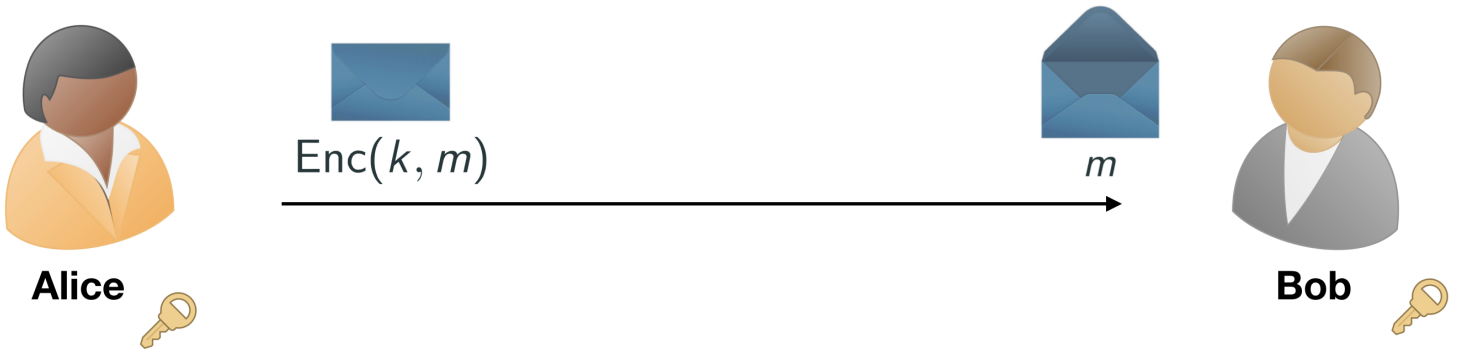
Symmetric Encryption

Alice

Bob

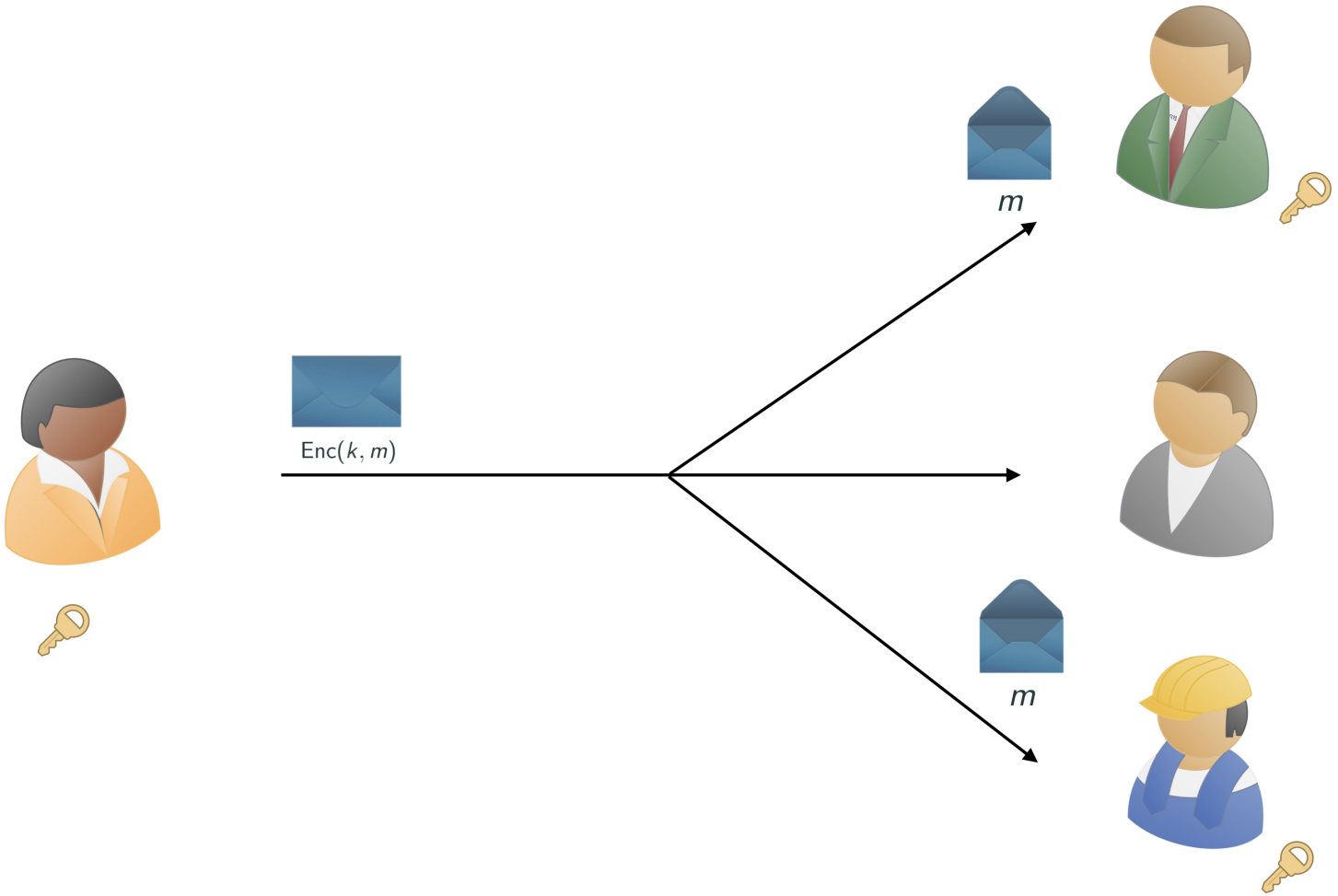
Symmetric Encryption

Symmetric Encryption

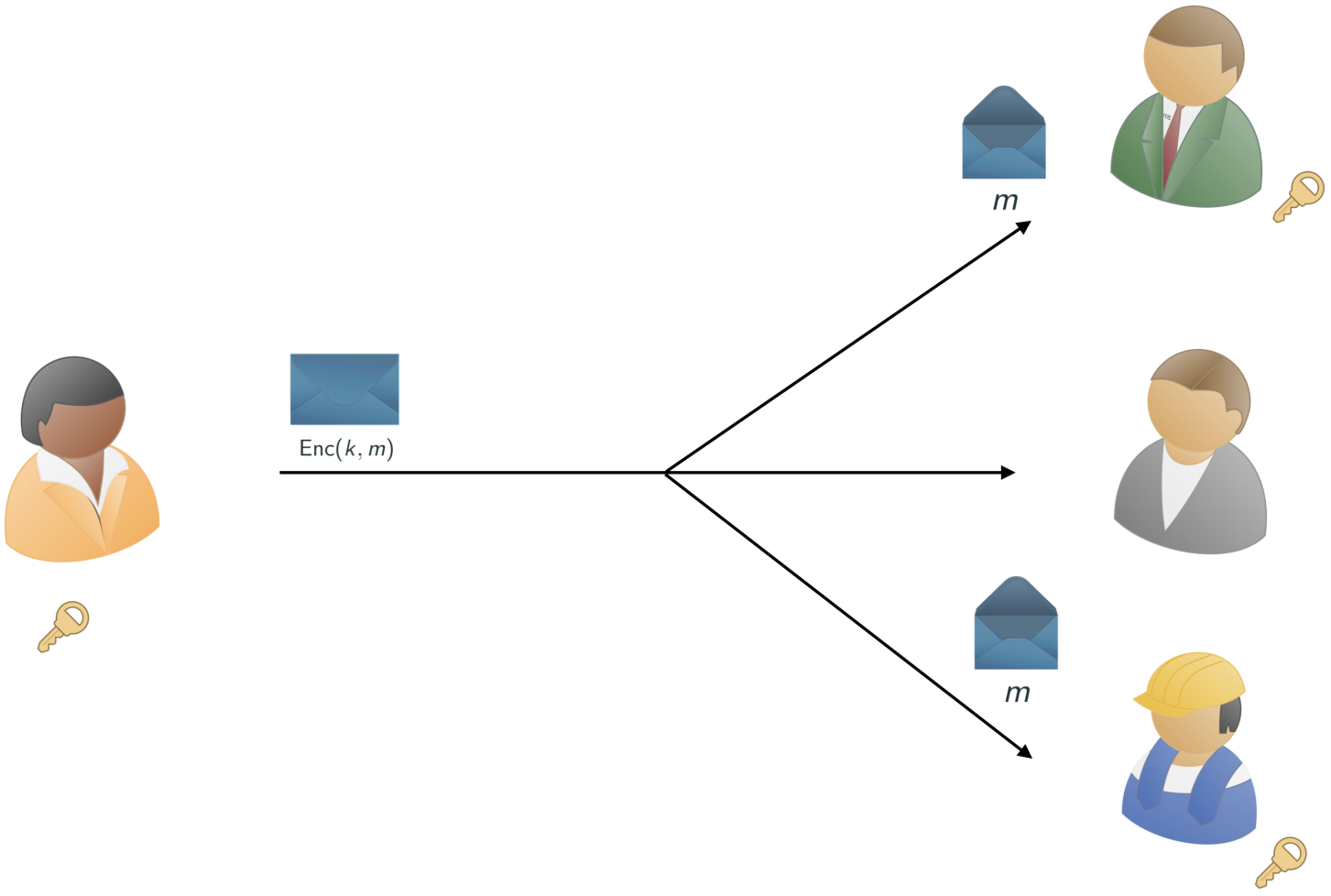


Symmetric Encryption

Symmetric Encryption



Symmetric Encryption



Must have many shared keys!

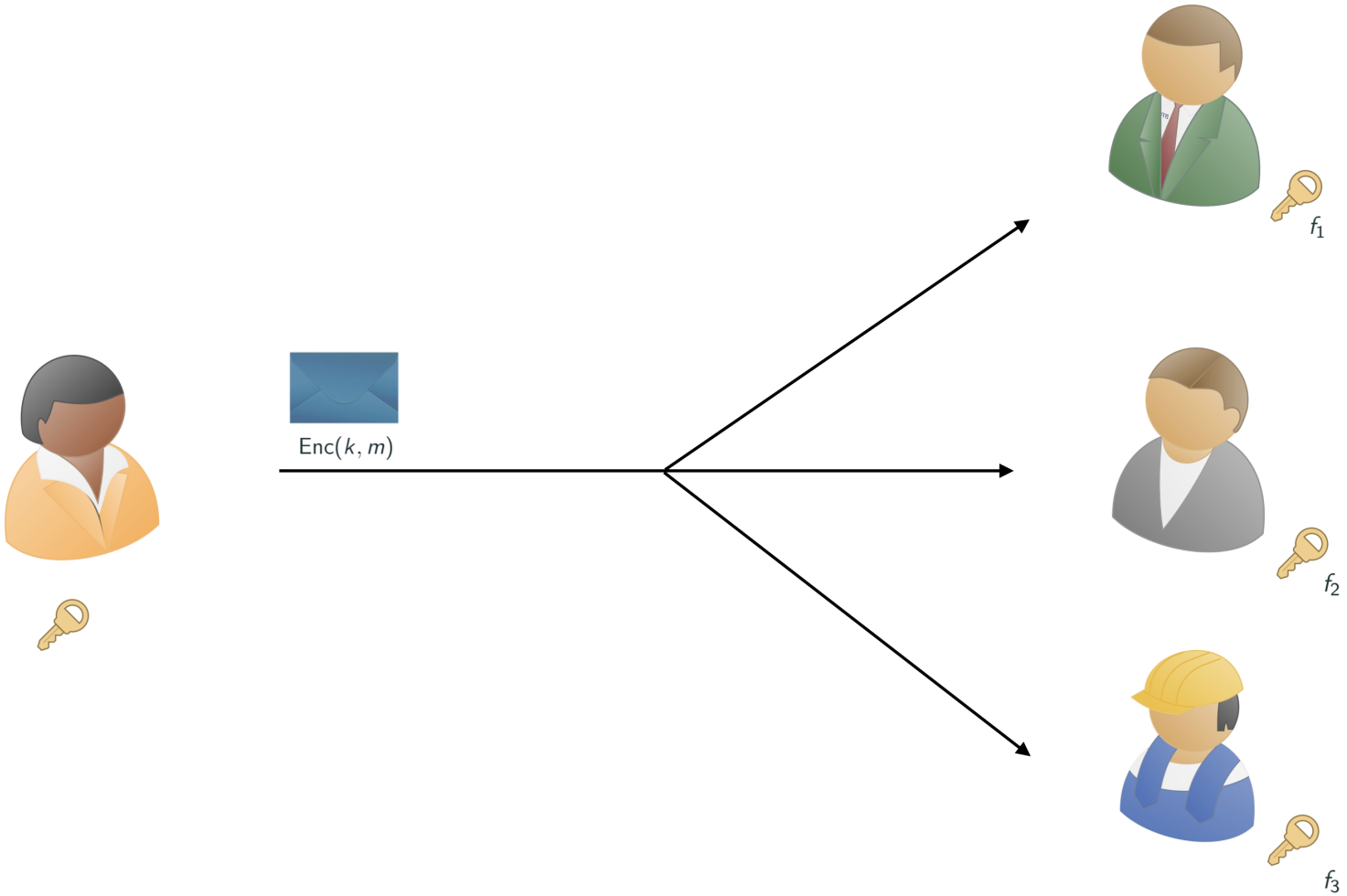
Functional Encryption [BSW11]

f_1

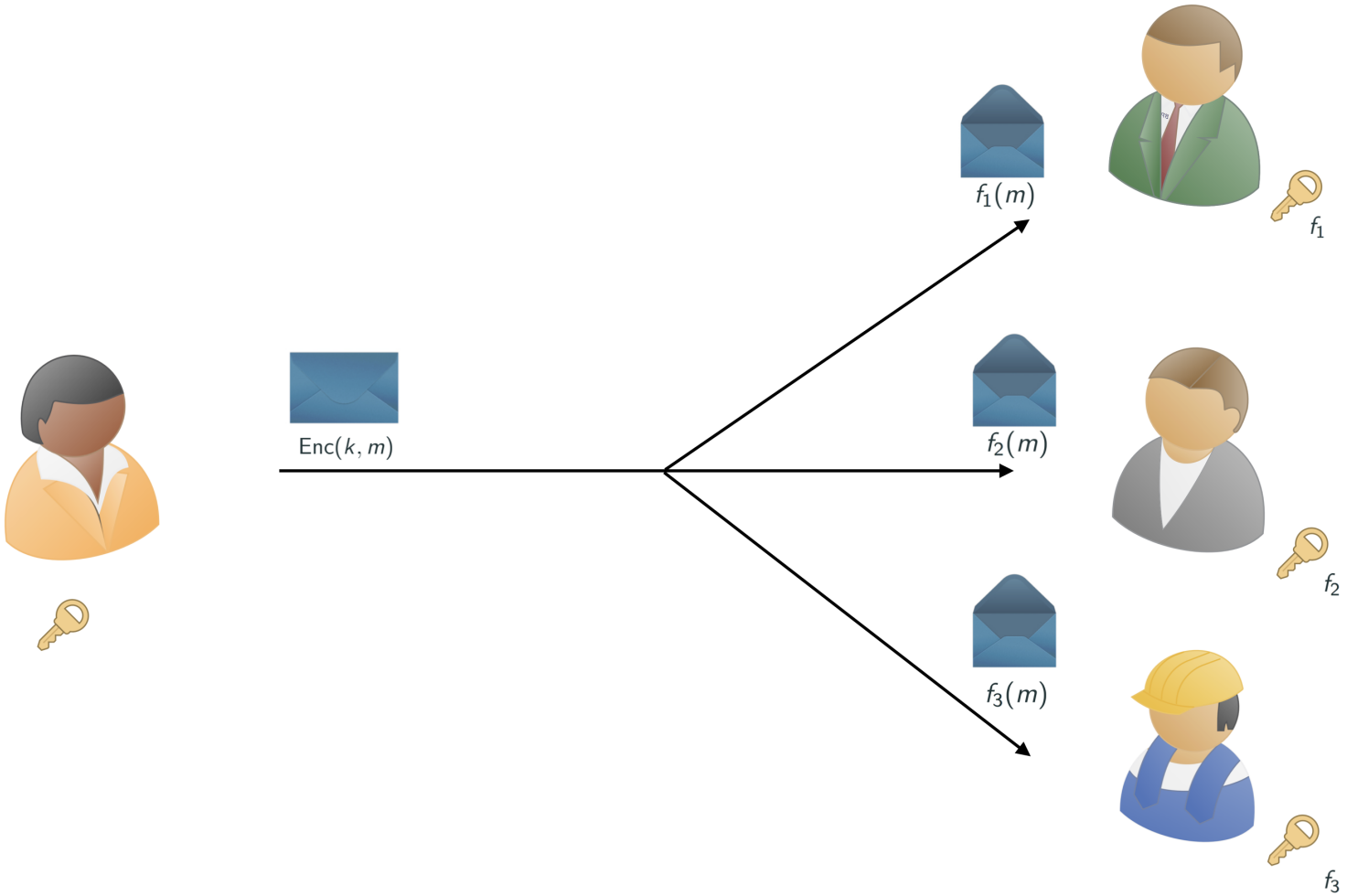
f_2

f_3

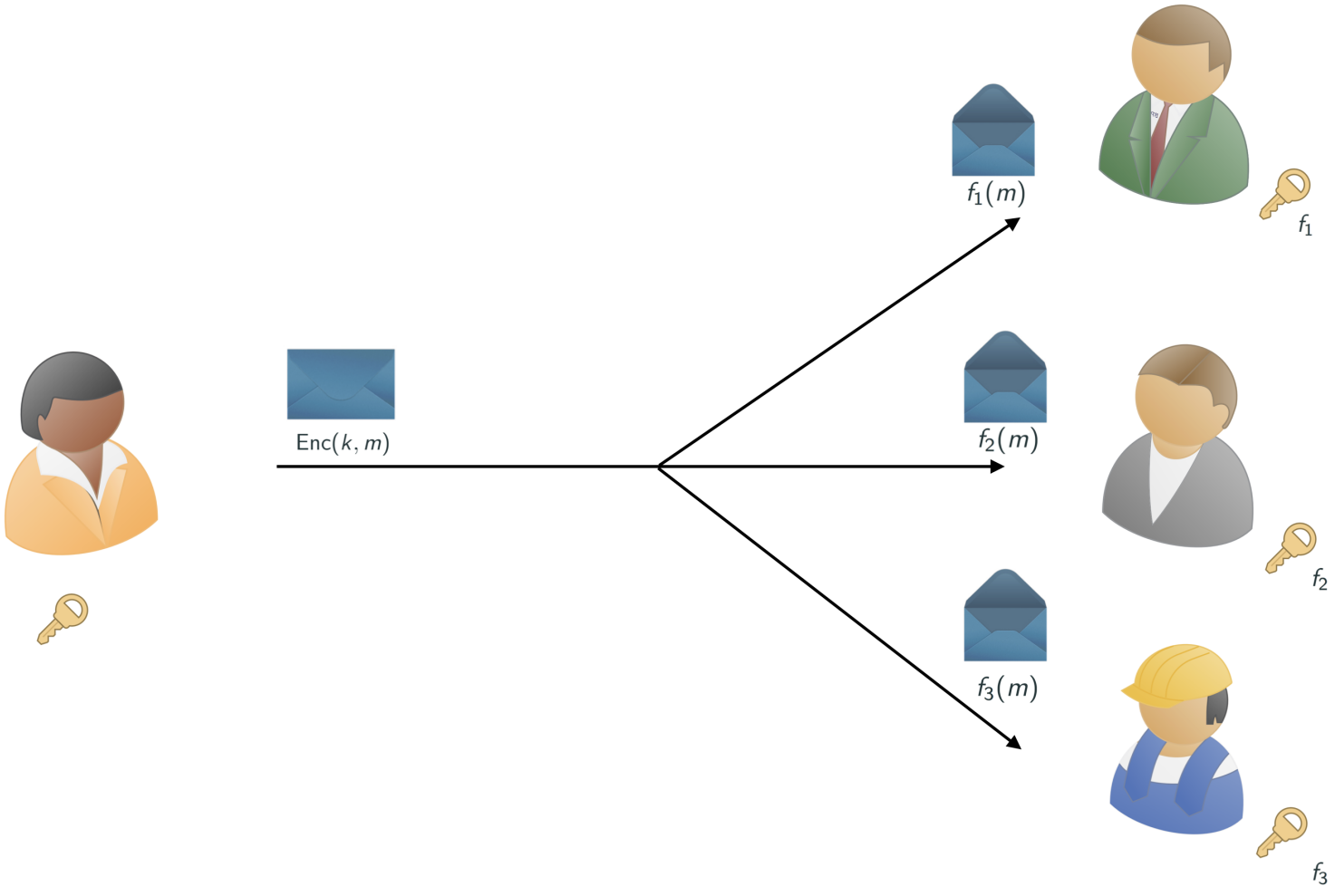
Functional Encryption [BSW11]



Functional Encryption [BSW11]



Functional Encryption [BSW11]



Functional Encryption: Fine-Grained Control of Decryptors

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Applicable to many real-world scenarios:

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Applicable to many real-world scenarios:

- Users allowed to work with data

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Applicable to many real-world scenarios:

- Users allowed to work with data
- Users not allowed to publicly reveal data

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Applicable to many real-world scenarios:

- Users allowed to work with data
- Users not allowed to publicly reveal data
- Want to prevent malicious senders

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Applicable to many real-world scenarios:

- Users allowed to work with data
- Users not allowed to publicly reveal data
- Want to prevent malicious senders
- Want to prevent mistakes or virus

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Applicable to many real-world scenarios:

- Users allowed to work with data
- Users not allowed to publicly reveal data
- Want to prevent malicious senders
- Want to prevent mistakes or virus

Clearly impossible without some additional hardware measures.

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Applicable to many real-world scenarios:

- Users allowed to work with data
- Users not allowed to publicly reveal data
- Want to prevent malicious senders
- Want to prevent mistakes or virus

Clearly impossible without some additional hardware measures.

[DHO16]: Access Control Encryption (ACE)

Access Control Encryption [DHO16]

Can we also have fine-grained control of encryptors?

Applicable to many real-world scenarios:

- Users allowed to work with data
- Users not allowed to publicly reveal data
- Want to prevent malicious senders
- Want to prevent mistakes or virus

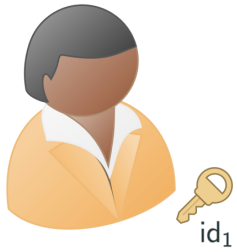
Clearly impossible without some additional hardware measures.

[DHO16]: Access Control Encryption (ACE)

Each ciphertext is processed by a Sanitizer

Access Control Encryption [DHO16]

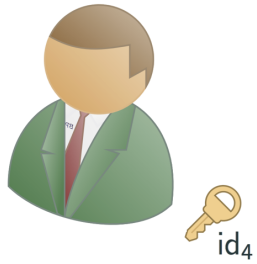
Encryptors

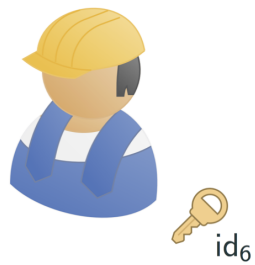


Fixed Predicate: $\pi(\mathrm{id}_i, \mathrm{id}_j) \to \{0, 1\}$

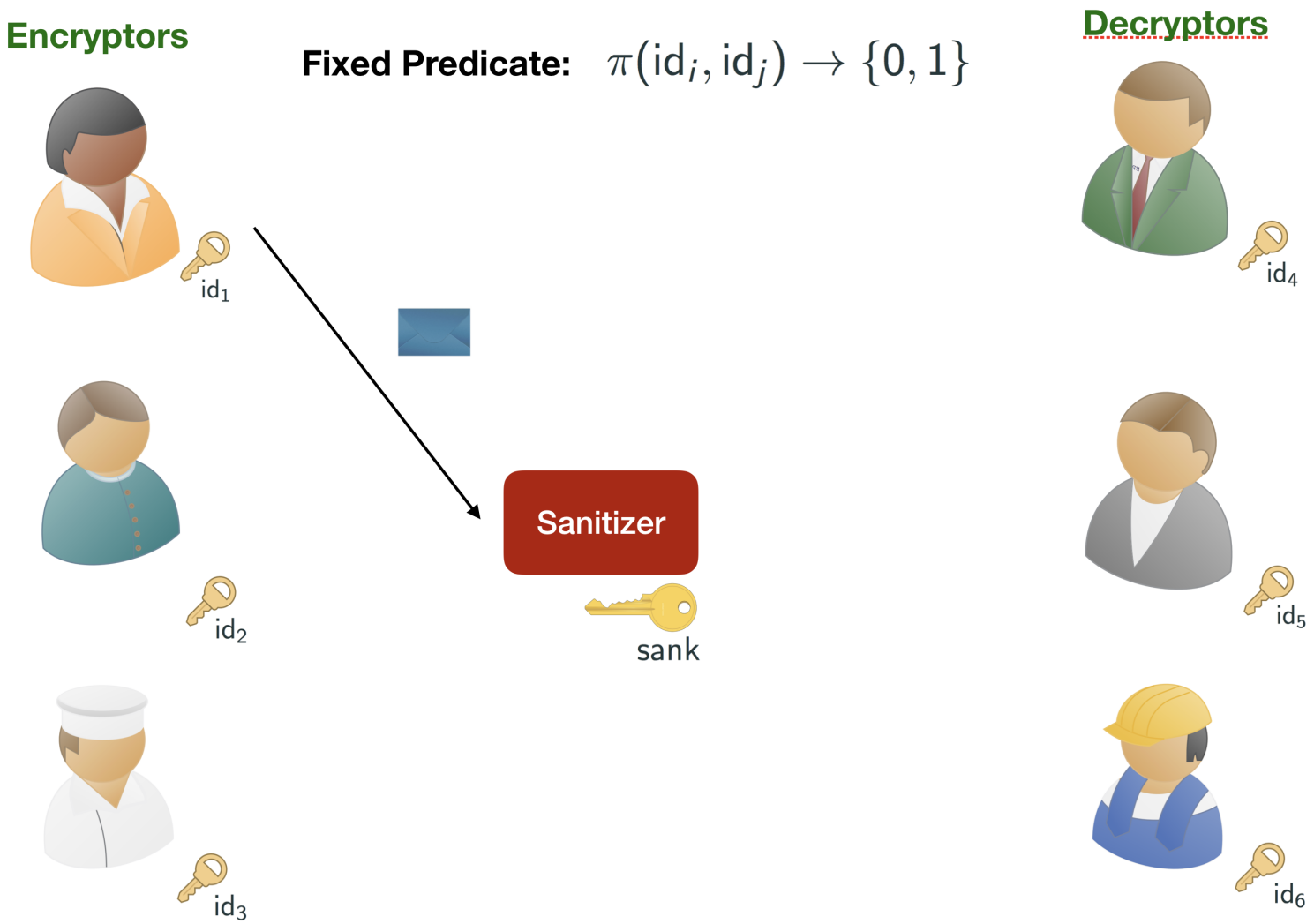
Sanitizer

Decryptors

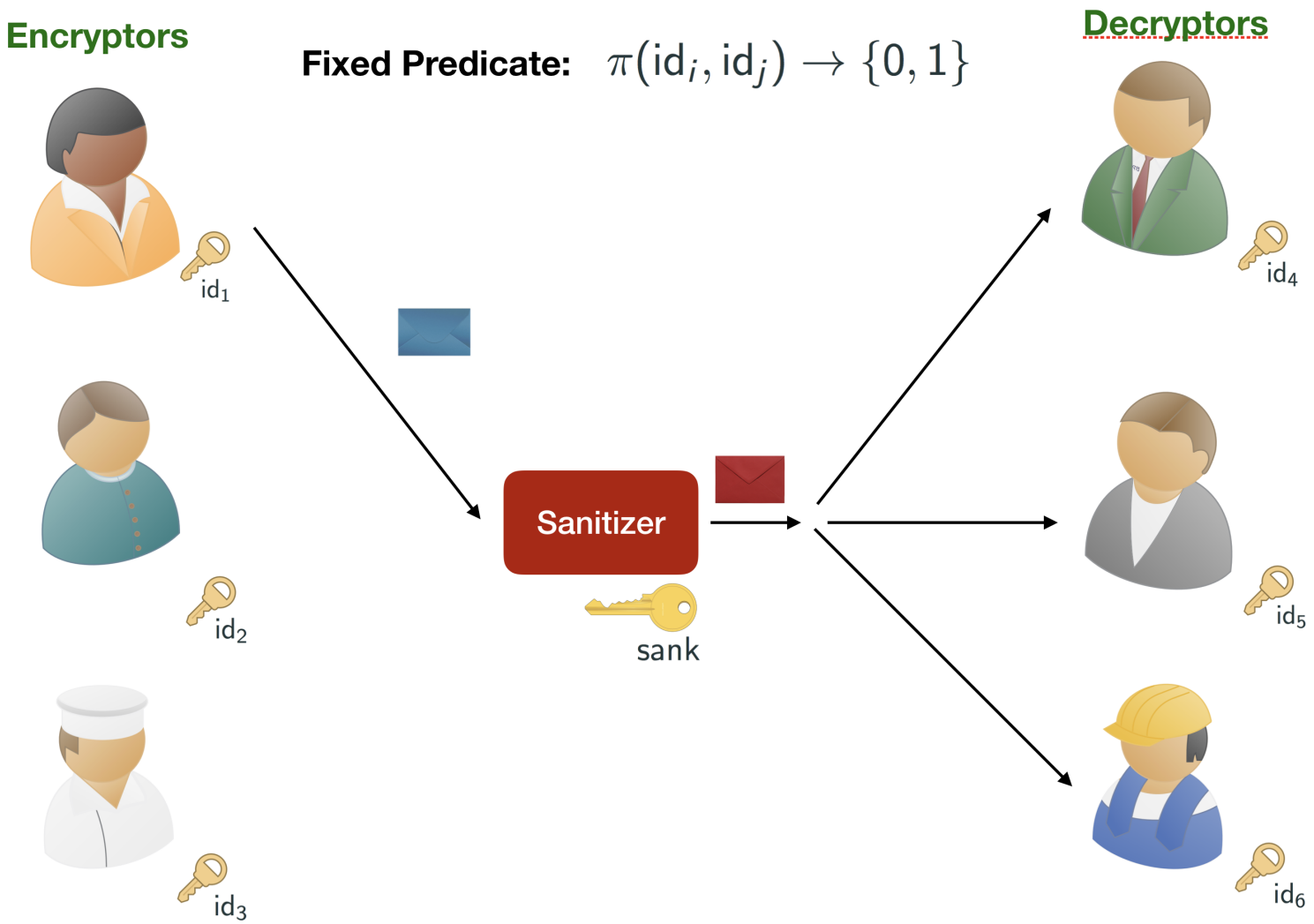




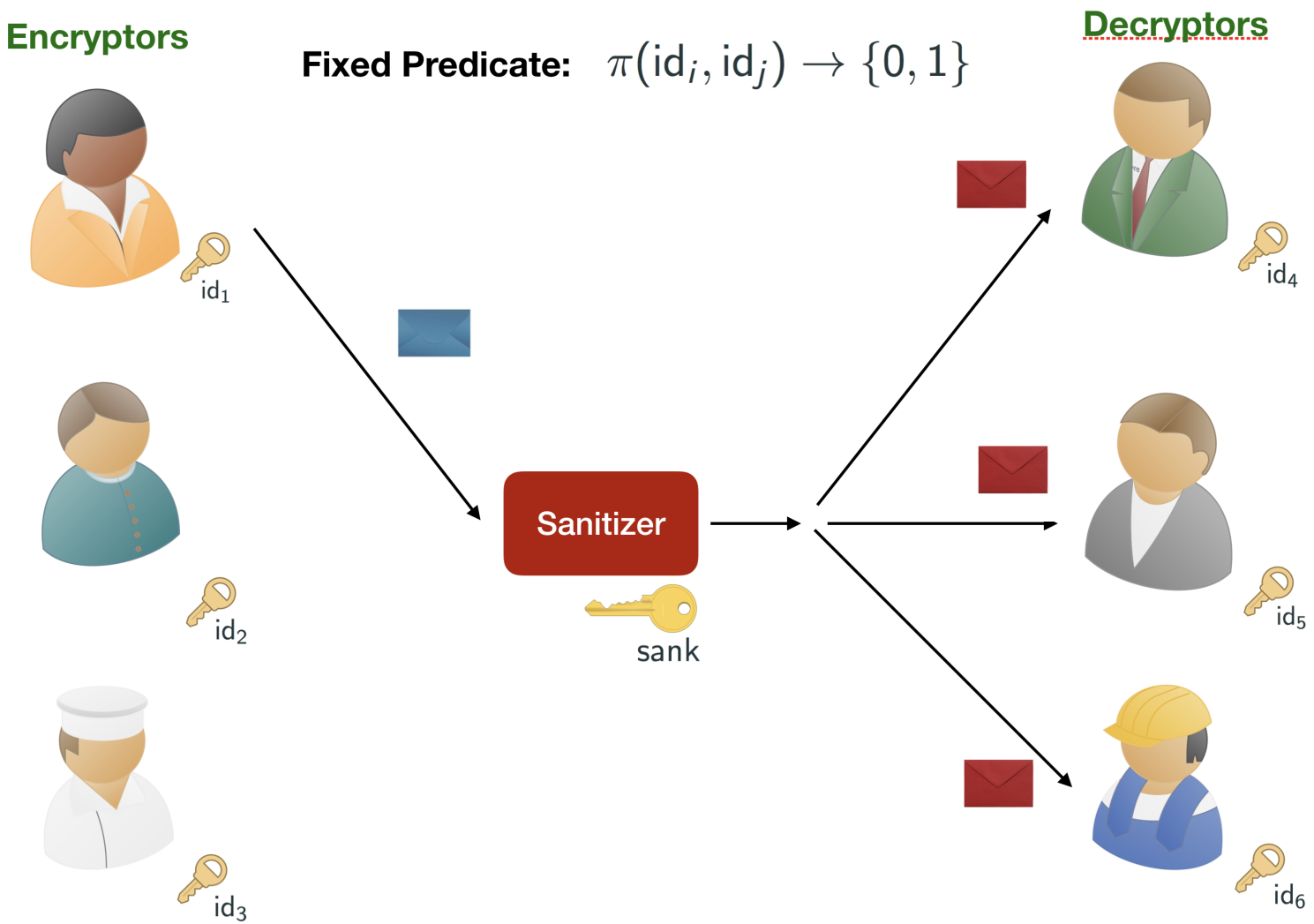
Access Control Encryption [DHO16]



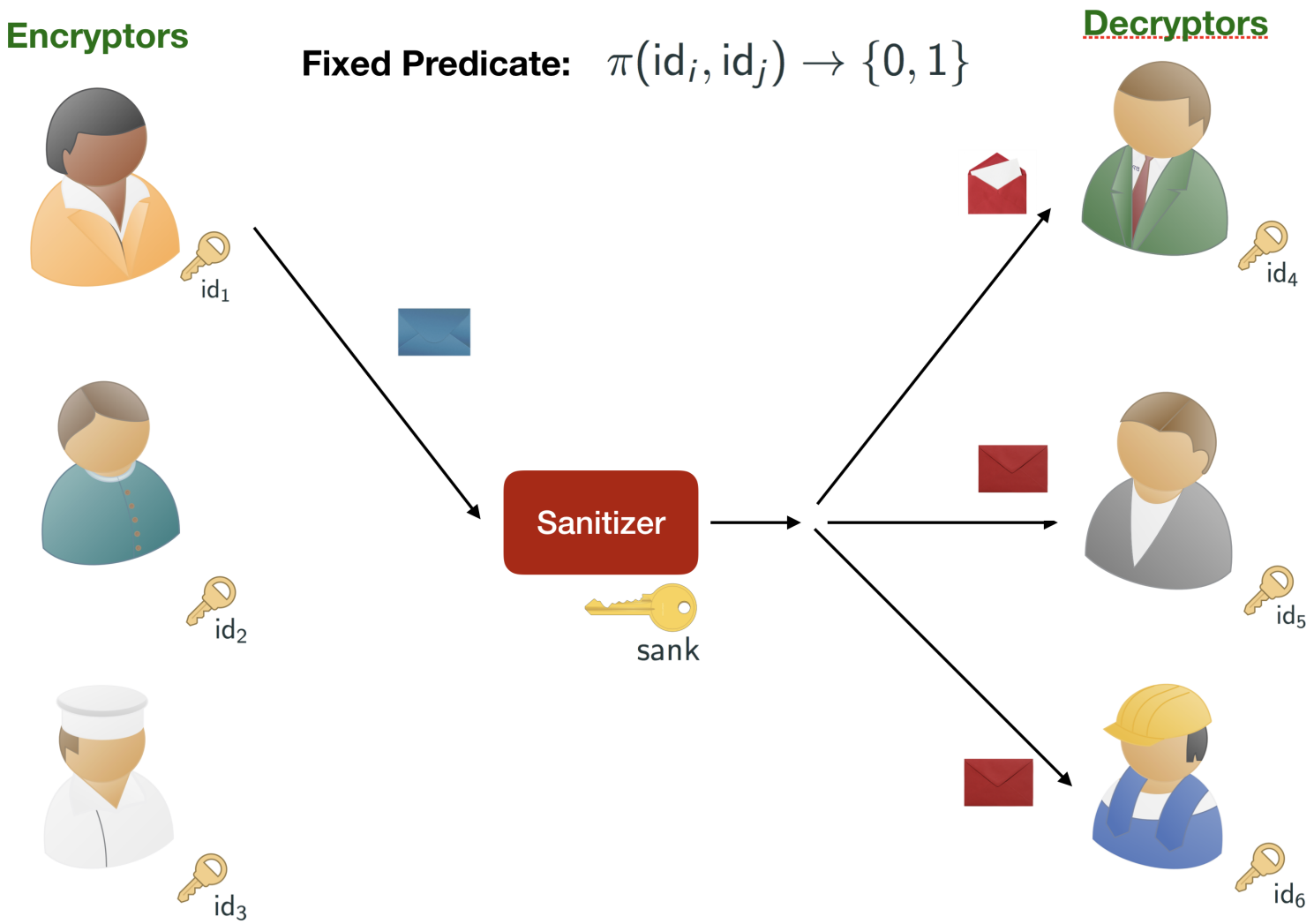
Access Control Encryption [DHO16]



Access Control Encryption [DHO16]

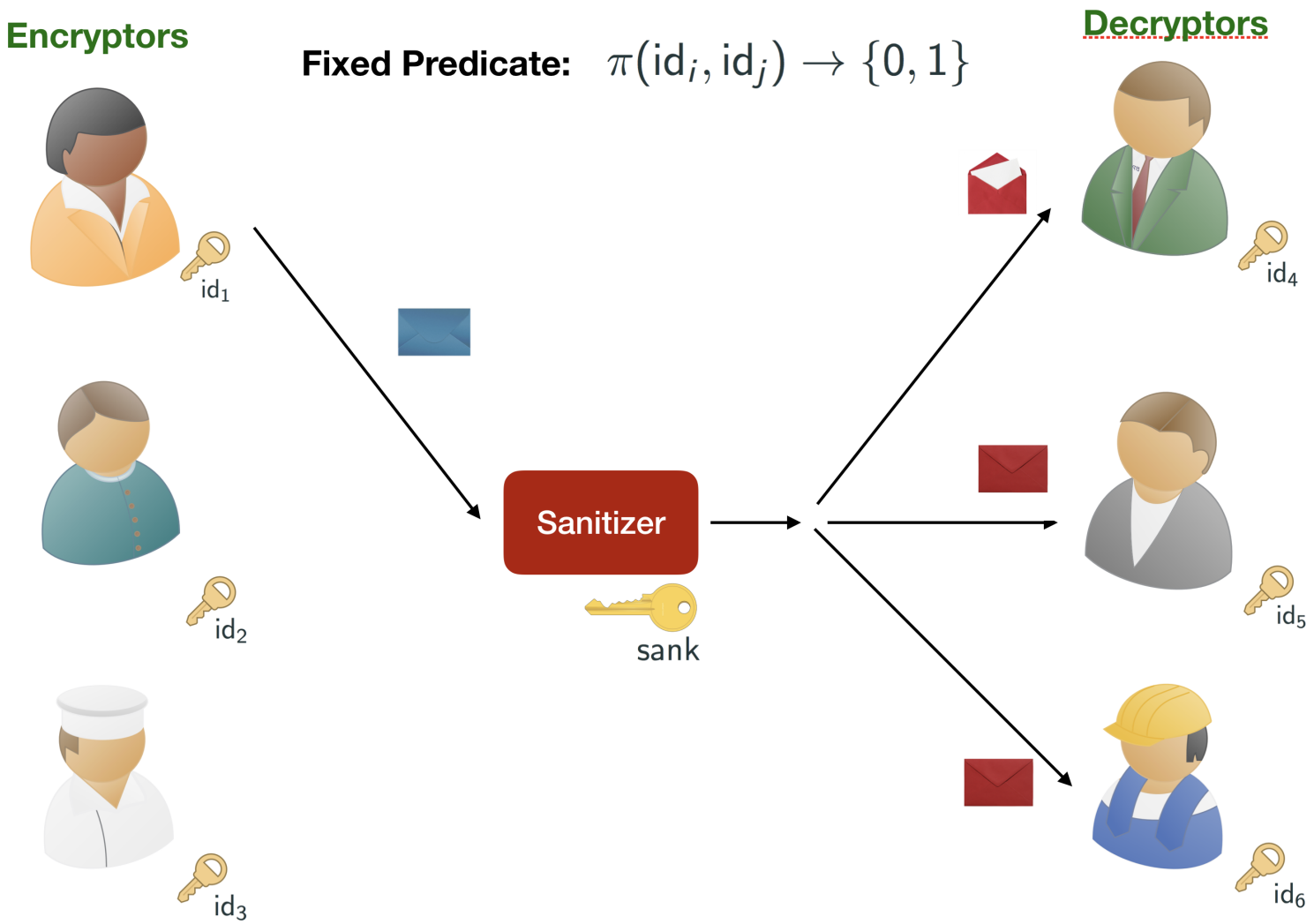


Access Control Encryption [DHO16]



Correctness: User id_i's ciphertext decryptable by user id_j iff $\pi(\mathrm{id}_i, \mathrm{id}_j) = 1$.

Access Control Encryption [DHO16]



General Goal: Minimize what Sanitizer learns!

No-Read Security

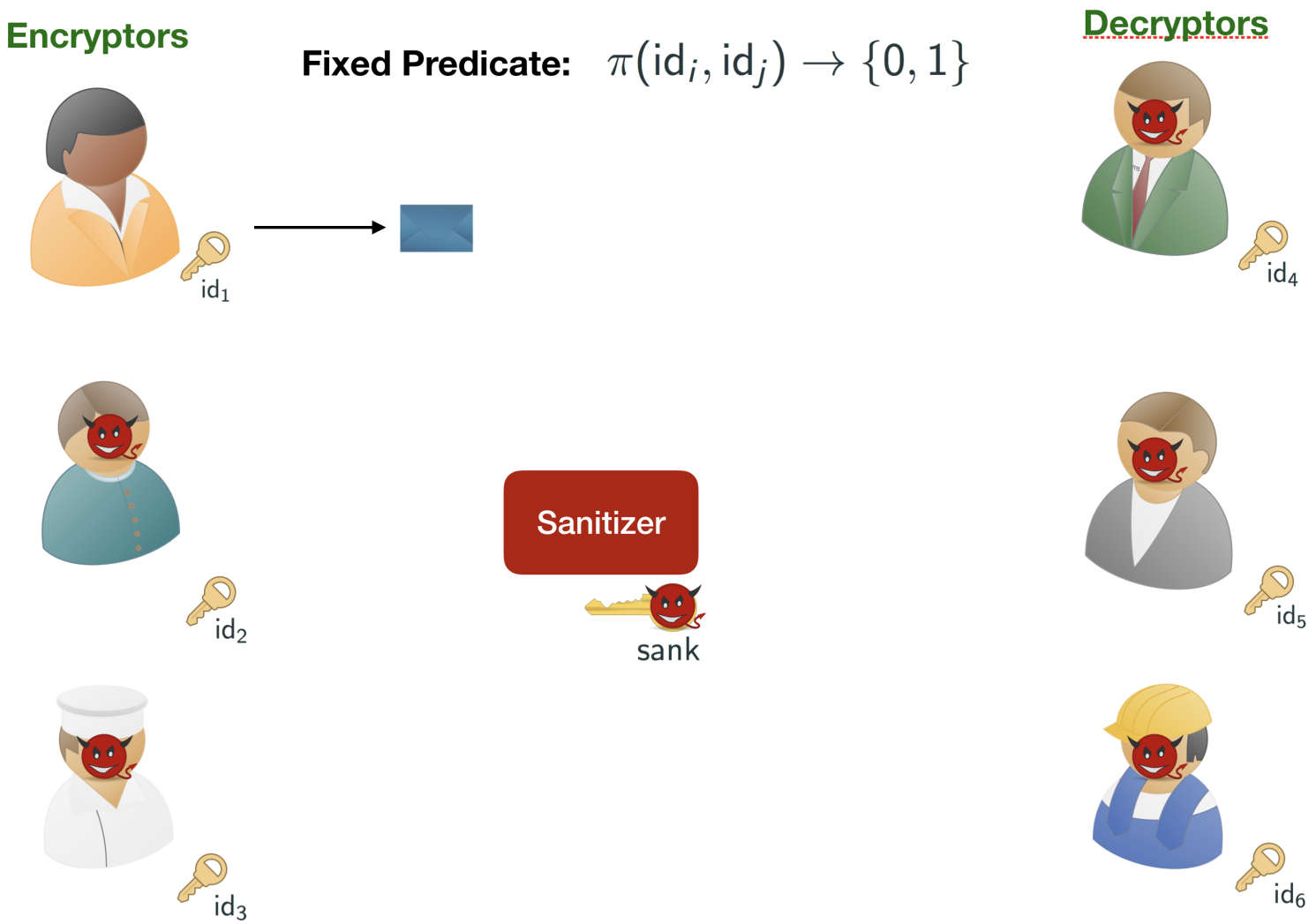
Encryptors

Fixed Predicate: $\pi(\mathrm{id}_i, \mathrm{id}_j) \rightarrow \{0, 1\}$

Sanitizer

Decryptors

No-Read Security



Requirement: Without a predicate-satisfying key, adversary learns no information.

No-Write Security

Encryptors

id_1

id_2

id_3

Fixed Predicate: $\pi(id_i, id_j) \rightarrow \{0, 1\}$

Sanitizer

sank

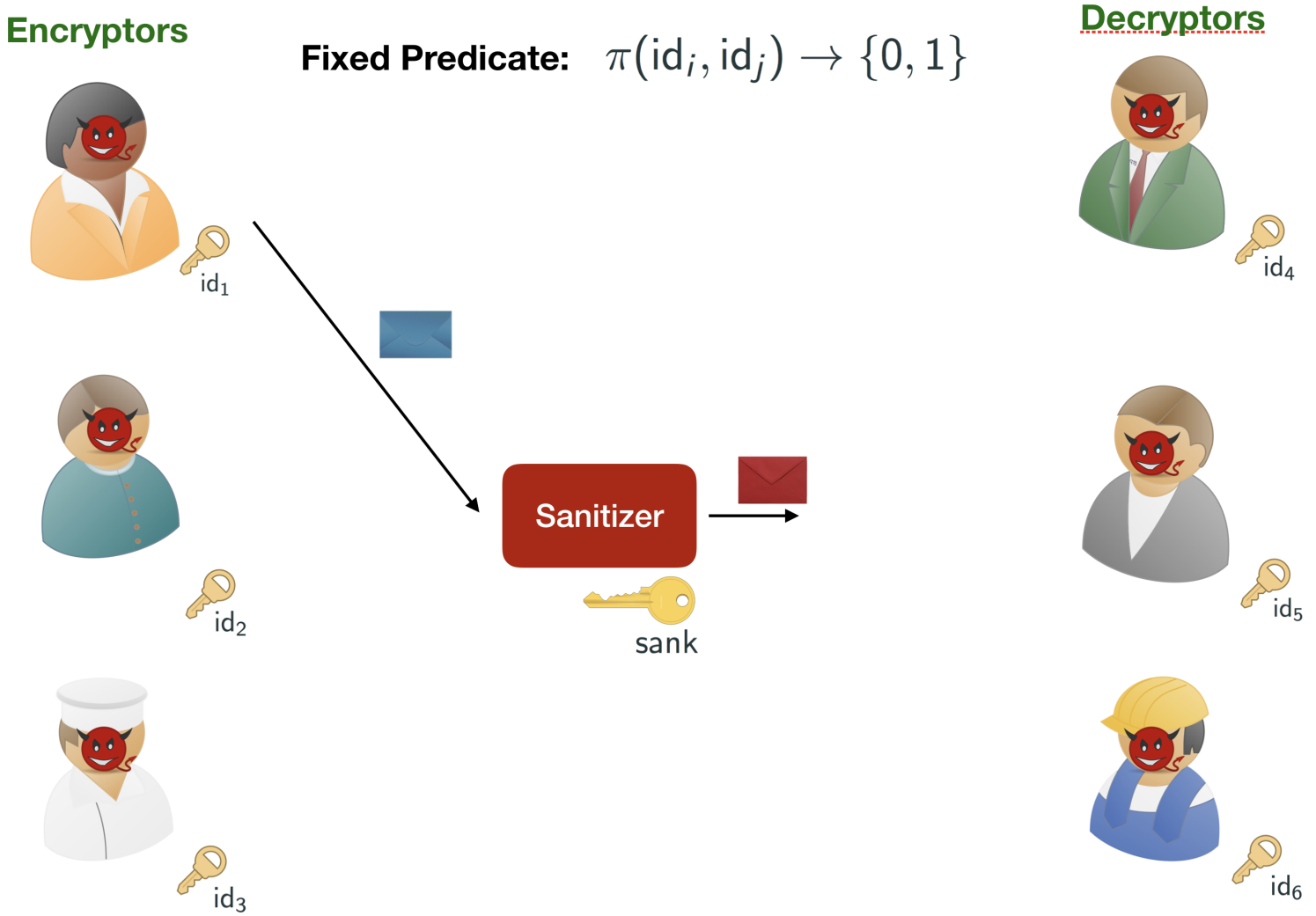
Decryptors

id_4

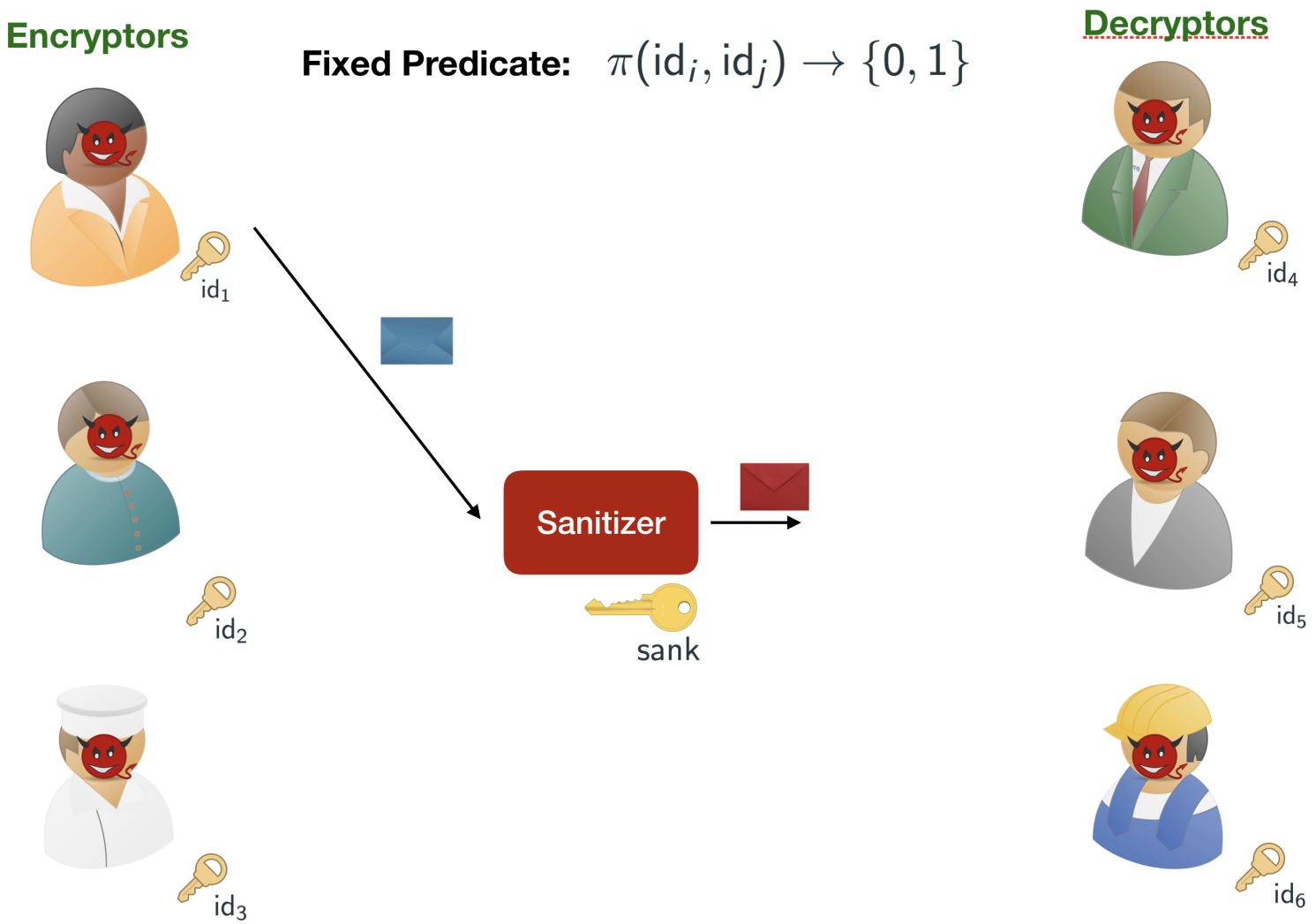
id_5

id_6

No-Write Security



No-Write Security



Requirement: Without a predicate-satisfying pair of keys, adversary cannot distinguish: honestly sanitized ctxt vs. randomly generated ctxt

Previous Works

Construction	Predicate	Ciphertext Size	Assumption
DHO16	arbitrary	$O(2^n)$	DDH or DCR
DHO16	arbitrary	poly(n)	iO

Previous Works

Construction	Predicate	Ciphertext Size	Assumption
DHO16	arbitrary	$O(2^n)$	DDH or DCR
DHO16	arbitrary	poly(n)	iO
FGKO17	restricted	poly(n)	SXDH

Previous Works

Construction	Predicate	Ciphertext Size	Assumption
DHO16	arbitrary	$O(2^n)$	DDH or DCR
DHO16	arbitrary	poly(n)	iO
FGKO17	restricted	poly(n)	SXDH
TZMT17	arbitrary	$O(2^n)$	LWE

Our Results

Construction	Predicate	Ciphertext Size	Assumption
DHO16	arbitrary	$O(2^n)$	DDH or DCR
DHO16	arbitrary	poly(n)	iO
FGKO17	restricted	poly(n)	SXDH
TZMT17	arbitrary	$O(2^n)$	LWE
This Work:	arbitrary	poly(n)	DDH, RSA, LWE

Starting Point: Basic Cert

Let's first achieve: No-Write Security

Starting Point: Basic Cert

Let's first achieve: No-Write Security

Simple Idea: Provide cert σ_{id} of id as encryption key

Starting Point: Basic Cert

Let's first achieve: No-Write Security

Simple Idea: Provide cert σ_{id} of id as encryption key

Encryptor can attach $(\mathsf{id}_i, \sigma_{\mathsf{id}_i})$ to message.

Starting Point: Basic Cert

Let's first achieve: No-Write Security

Simple Idea: Provide cert σ_{id} of id as encryption key

Encryptor can attach $(\mathsf{id}_i, \sigma_{\mathsf{id}_i})$ to message.

Sanitizer verifies $(\mathsf{id}_i, \sigma_{\mathsf{id}_i})$ and encrypt to all identities id_j for which

$$\pi(\mathsf{id}_i, \mathsf{id}_j) = 1$$

Starting Point: Basic Cert

Encryptors

id_1
σ_{id_1}

id_2
σ_{id_2}

id_3
σ_{id_3}

Fixed Predicate: $\pi(\mathsf{id}_i, \mathsf{id}_j) \rightarrow \{0, 1\}$

Sanitizer

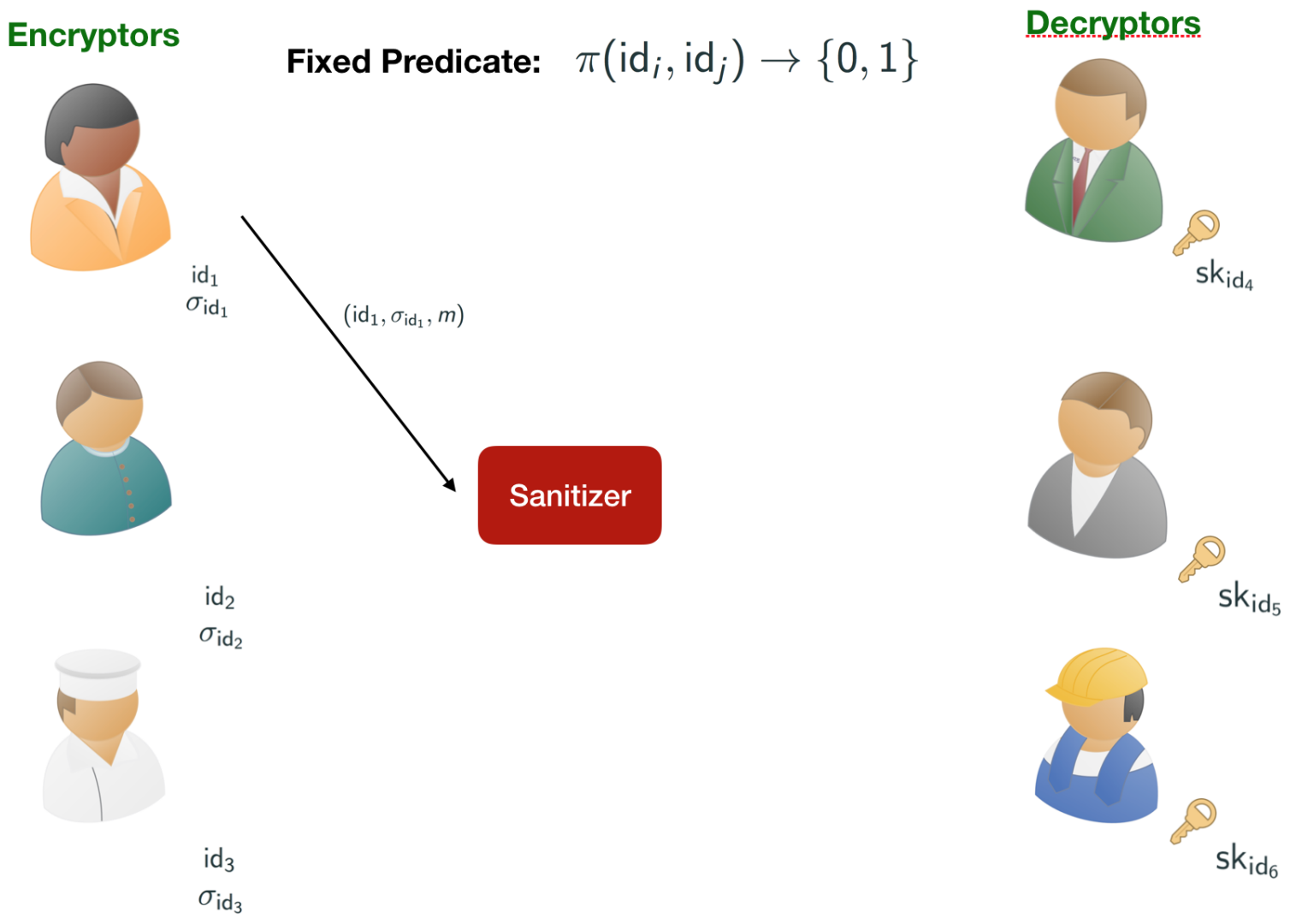
Decryptors

$\mathsf{sk}_{\mathsf{id}_4}$

$\mathsf{sk}_{\mathsf{id}_5}$

$\mathsf{sk}_{\mathsf{id}_6}$

Starting Point: Basic Cert



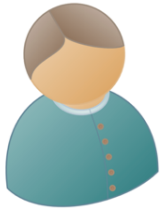
Starting Point: Basic Cert

Encryptors

Fixed Predicate: $\pi(\mathsf{id}_i, \mathsf{id}_j) \rightarrow \{0, 1\}$

Decryptors

id_1
σ_{id_1}



id_2
σ_{id_2}

id_3
σ_{id_3}

Sanitizer

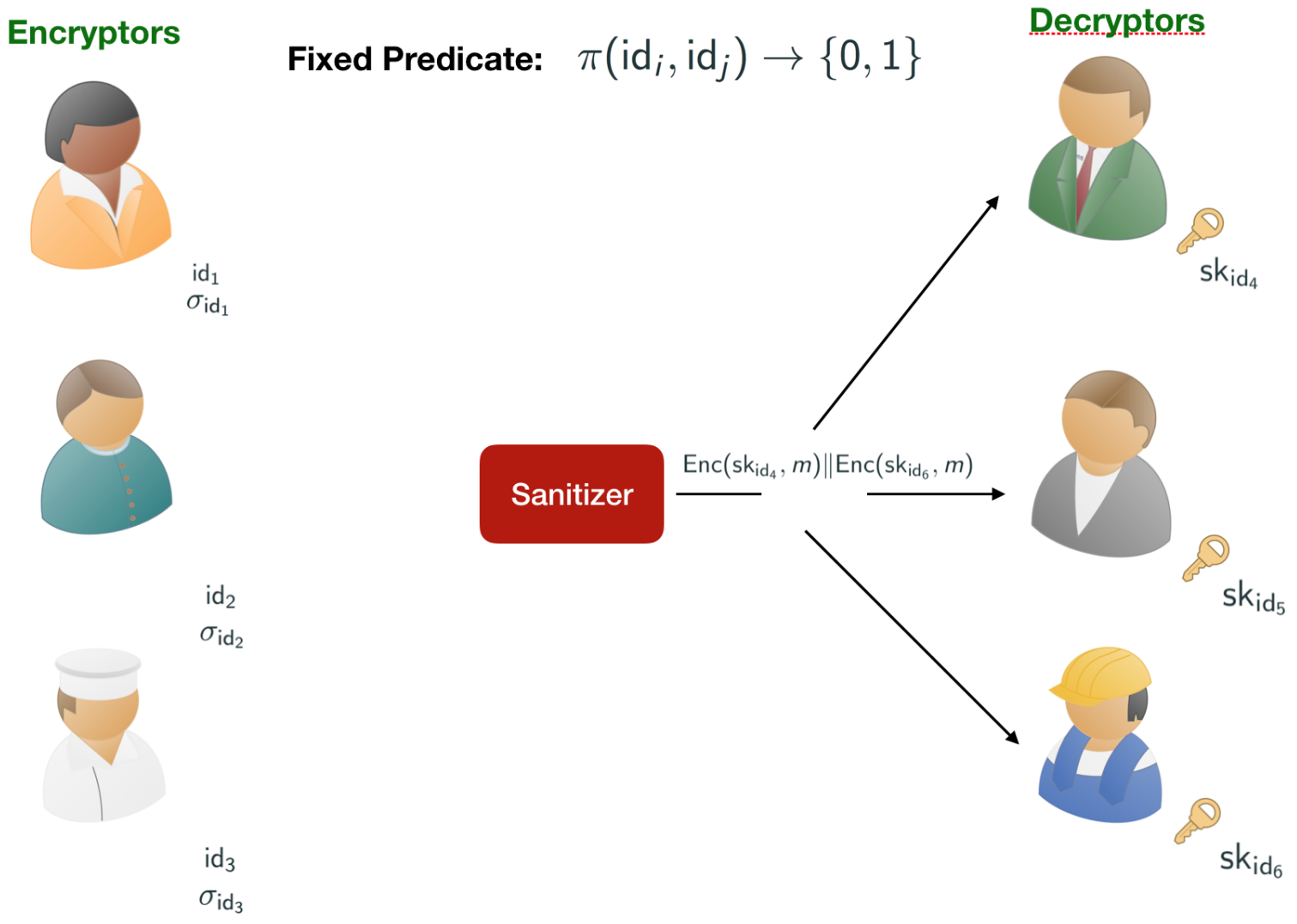
$\mathsf{Enc}(\mathsf{sk}_{\mathsf{id}_4}, m) \| \mathsf{Enc}(\mathsf{sk}_{\mathsf{id}_6}, m)$

$\mathsf{sk}_{\mathsf{id}_4}$

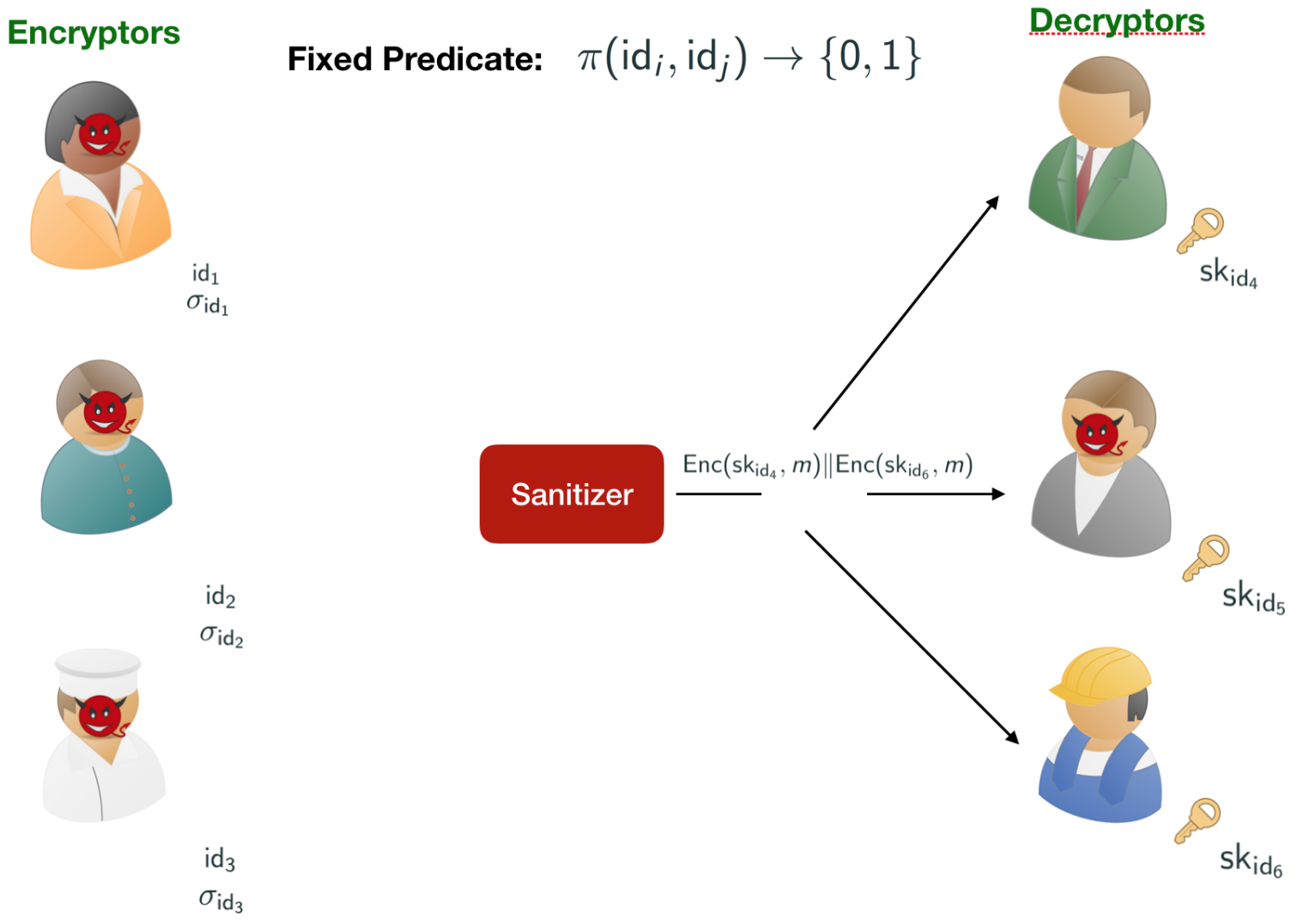
$\mathsf{sk}_{\mathsf{id}_5}$

$\mathsf{sk}_{\mathsf{id}_6}$

Starting Point: Basic Cert



Starting Point: Basic Cert



Compact Ciphertext: Predicate Encryption

Next Goal: Compact Ciphertext

Compact Ciphertext: Predicate Encryption

Next Goal: Compact Ciphertext

Useful Tool: Predicate Encryption [BW07,KSW08]

Compact Ciphertext: Predicate Encryption

Next Goal: Compact Ciphertext

Useful Tool: Predicate Encryption [BW07,KSW08]

Encryption algorithm takes in (attribute, message) pair

$$\mathsf{Enc}(x, m) \to \mathsf{ct}_{x,m}.$$

Decryption keys associated with function f: sk_f

$$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_{x,m}) = \begin{cases} m & f(x) = 1 \\ \perp & f(x) = 0 \end{cases}.$$

Compact Ciphertext: Predicate Encryption

Next Goal: Compact Ciphertext

Useful Tool: Predicate Encryption [BW07,KSW08]

Encryption algorithm takes in (attribute, message) pair

$$\mathsf{Enc}(x, m) \to \mathsf{ct}_{x,m}.$$

Decryption keys associated with function f: sk_f

$$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_{x,m}) = \begin{cases} m & f(x) = 1 \\ \perp & f(x) = 0 \end{cases}.$$

Security guarantee: (x, m) hidden if only given sk_f's for which $f(x) = 0$

Compact Ciphertext: Predicate Encryption

Next Goal: Compact Ciphertext

Useful Tool: Predicate Encryption [BW07,KSW08]

Encryption algorithm takes in (attribute, message) pair

$$\mathsf{Enc}(x, m) \to \mathsf{ct}_{x,m}.$$

Decryption keys associated with function f: sk_f

$$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_{x,m}) = \begin{cases} m & f(x) = 1 \\ \perp & f(x) = 0 \end{cases}.$$

Security guarantee: (x, m) hidden if only given sk_f's for which $f(x) = 0$

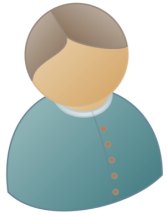
Idea:

- Sanitizer when given $(\mathsf{id}_i, \sigma_{\mathsf{id}_i}, m)$ encrypts $\mathsf{Enc}(\mathsf{id}_i, m)$
- Decryptor with id_j given $\mathsf{sk}_{\pi(\cdot,\mathsf{id}_j)}$

Compact Ciphertext: Predicate Encryption

Encryptors

id_1
σ_{id_1}



id_2
σ_{id_2}

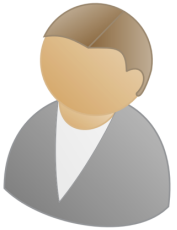
id_3
σ_{id_3}

Fixed Predicate: $\pi(\mathsf{id}_i, \mathsf{id}_j) \to \{0, 1\}$

Sanitizer

Decryptors

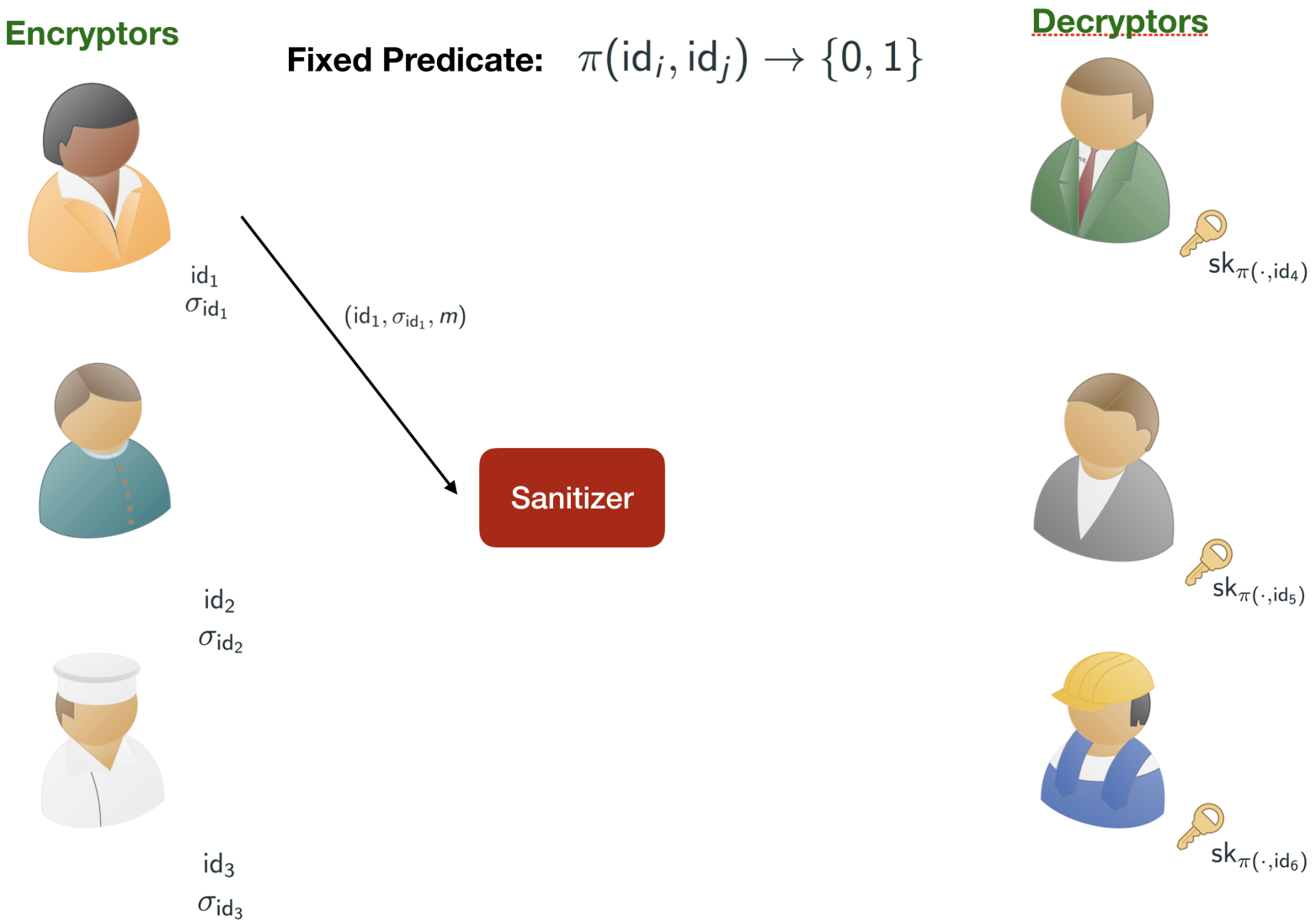
$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_4)}$



$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_5)}$

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_6)}$

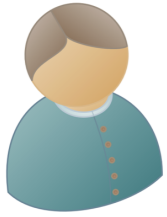
Compact Ciphertext: Predicate Encryption



Compact Ciphertext: Predicate Encryption

Encryptors

id_1
σ_{id_1}



id_2
σ_{id_2}

id_3
σ_{id_3}

Fixed Predicate: $\pi(\mathsf{id}_i, \mathsf{id}_j) \to \{0, 1\}$

$\mathsf{Enc}(\mathsf{id}_1, m)$

Sanitizer

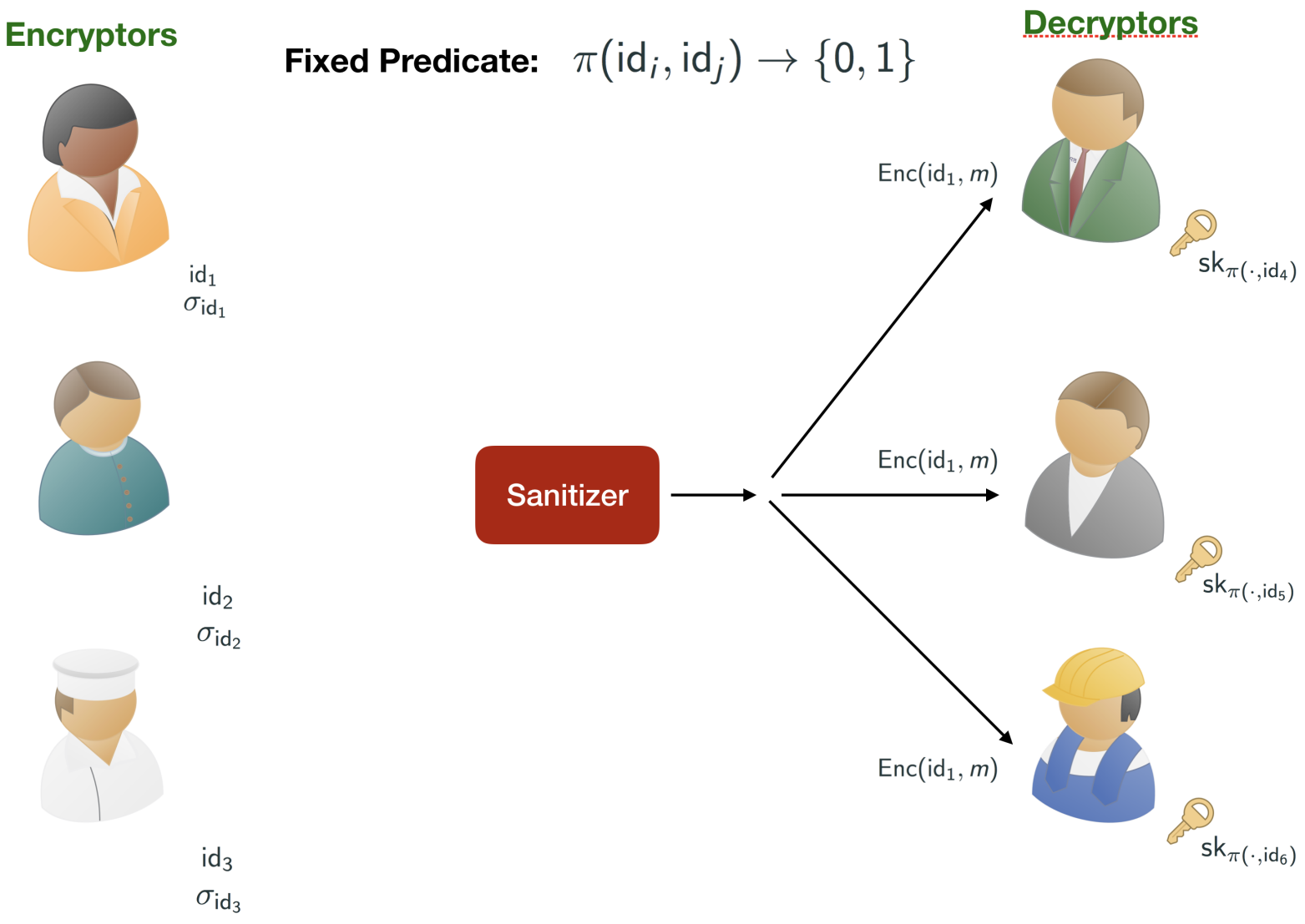
Decryptors

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_4)}$

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_5)}$

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_6)}$

Compact Ciphertext: Predicate Encryption



Hiding Information from Sanitizer

Next Goal: No-Read Security

Hiding Information from Sanitizer

Next Goal: No-Read Security

Problem: Sanitizer learns too much information!

Hiding Information from Sanitizer

Next Goal: No-Read Security

Problem: Sanitizer learns too much information!

Current task:

1. Check signature $(\mathsf{id}, \sigma_{\mathsf{id}})$
2. Encrypt $\mathsf{Enc}(\mathsf{id}, m)$

Hiding Information from Sanitizer

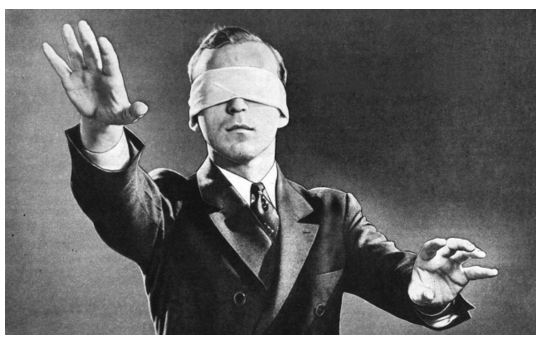
Next Goal: No-Read Security

Problem: Sanitizer learns too much information!

Current task:

1. Check signature $(\mathsf{id}, \sigma_{\mathsf{id}})$
2. Encrypt $\mathsf{Enc}(\mathsf{id}, m)$

Can the sanitizer perform its task blind-folded?



Hiding Information from Sanitizer

Let's use Functional Encryption [BSW11]!

Hiding Information from Sanitizer

Let's use Functional Encryption [BSW11]!

Bounded key Functional Encryption can be constructed from standard assumptions [SS10, GVW12, GKPVZ13, ...]

Hiding Information from Sanitizer

Let's use Functional Encryption [BSW11]!

Bounded key Functional Encryption can be constructed from standard assumptions [SS10, GVW12, GKPVZ13, ...]

Idea: Provide FE secret key sk_F where F defined as follows:

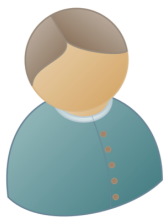
$F(\mathsf{id}, \sigma, m)$:

1. Check (id, σ)
2. If valid, then return $\mathsf{Enc}(\mathsf{id}, m)$

Hiding Information from Sanitizer

Encryptors

id_1
σ_{id_1}



id_2
σ_{id_2}

id_3
σ_{id_3}

Fixed Predicate: $\pi(\mathsf{id}_i, \mathsf{id}_j) \rightarrow \{0, 1\}$

Sanitizer

sk_F

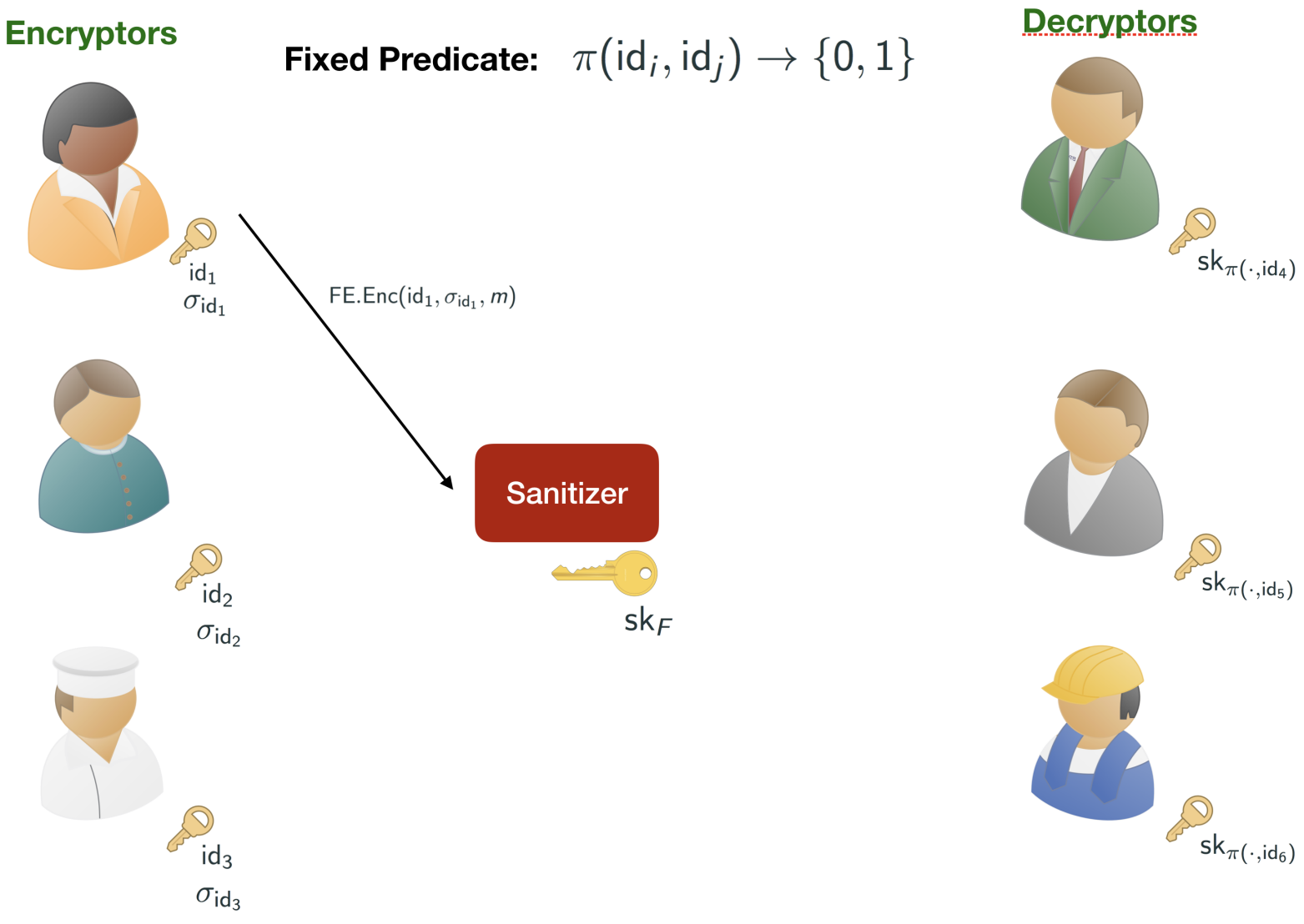
Decryptors

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_4)}$

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_5)}$

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_6)}$

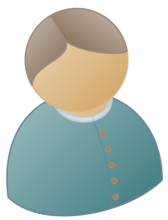
Hiding Information from Sanitizer



Hiding Information from Sanitizer

Encryptors

id_1
σ_{id_1}



id_2
σ_{id_2}

id_3
σ_{id_3}

Fixed Predicate: $\pi(\mathsf{id}_i, \mathsf{id}_j) \to \{0, 1\}$

Sanitizer

$\mathsf{PE.Enc}(\mathsf{id}_1, m)$

sk_F

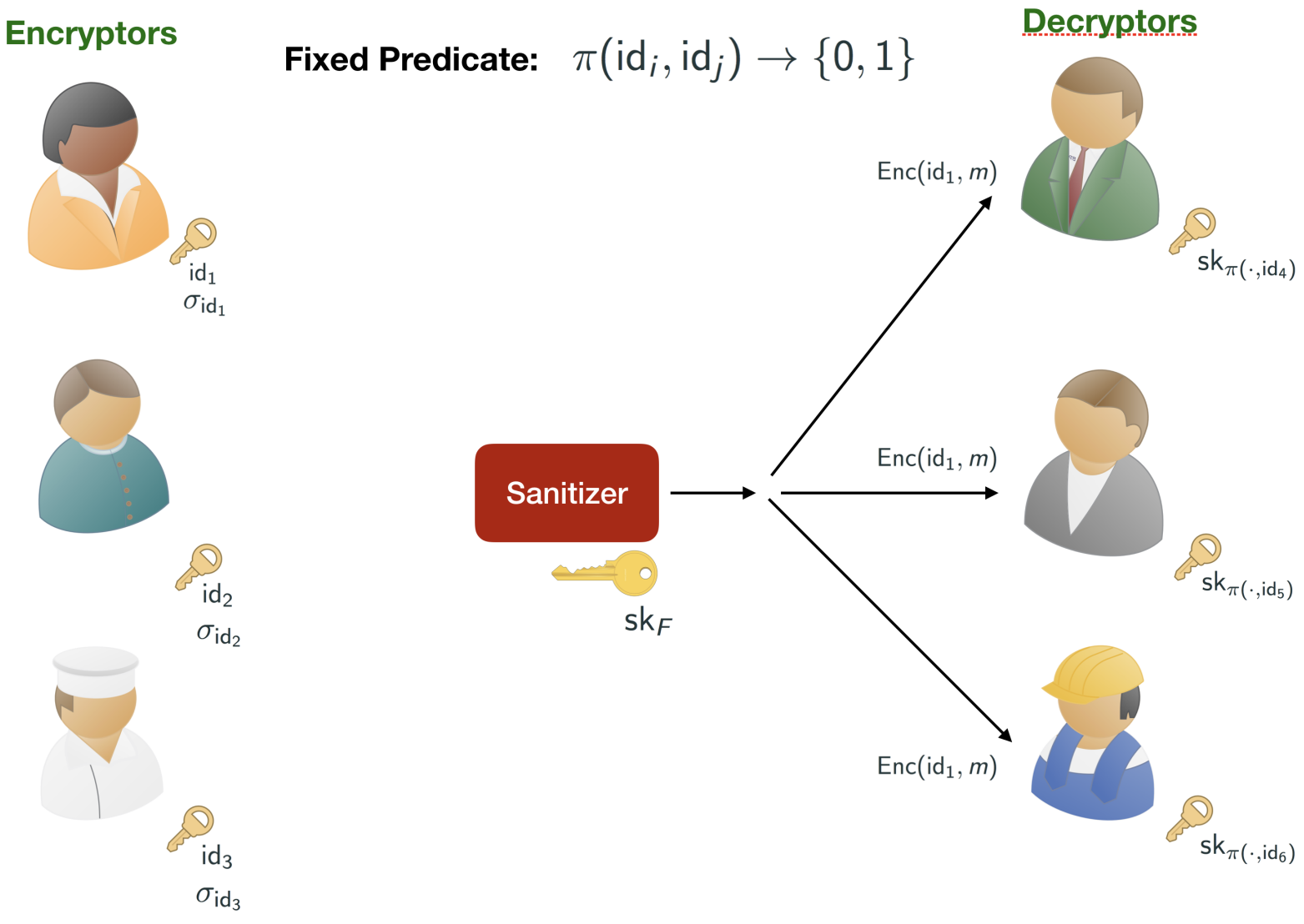
Decryptors

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_4)}$

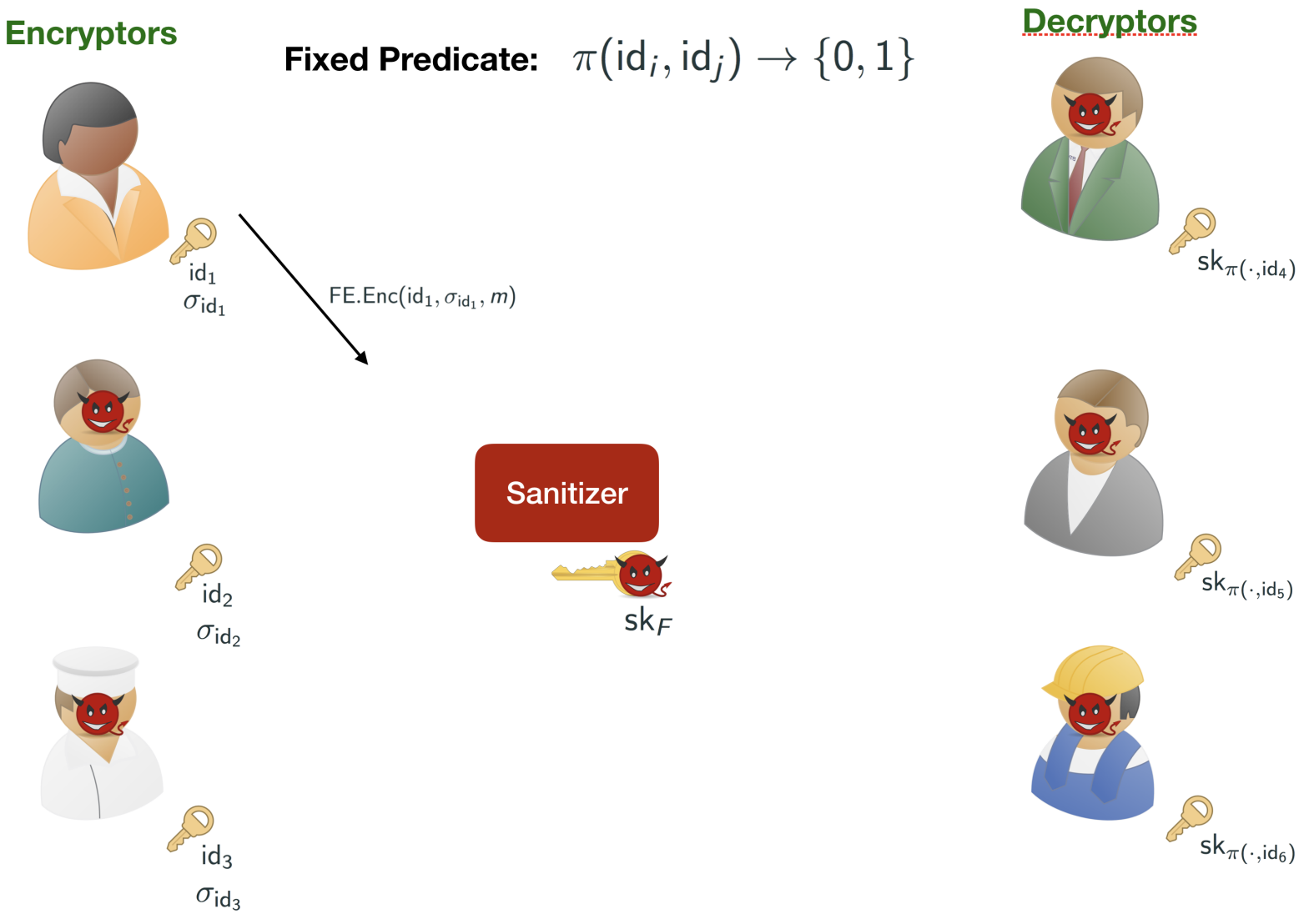
$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_5)}$

$\mathsf{sk}_{\pi(\cdot,\mathsf{id}_6)}$

Hiding Information from Sanitizer



Hiding Information from Sanitizer



Security Details

Requires a number of subtle details to argue security

Security Details

Requires a number of subtle details to argue security

Note: encryption is a randomized function!

Security Details

Requires a number of subtle details to argue security

Note: encryption is a randomized function!

Need to embed additional PRF keys as part of the message and FE keys to derive randomness

Security Details

Requires a number of subtle details to argue security

Note: encryption is a randomized function!

Need to embed additional PRF keys as part of the message and FE keys to derive randomness

Rely on FE for randomized functionalities [GJKS15, AW17]

Construction can be based on DDH + RSA.

Extensions

Additional extensions to ACE definition:

1. Dynamic policies

Extensions

Additional extensions to ACE definition:

1. Dynamic policies
2. Fine-grained sender policies

Extensions

Additional extensions to ACE definition:

1. Dynamic policies
2. Fine-grained sender policies
3. Beyond All-or-Nothing Decryption

Extensions

Additional extensions to ACE definition:

1. Dynamic policies
2. Fine-grained sender policies
3. Beyond All-or-Nothing Decryption

Provide modifications of our constructions can achieve these extended definition.

Conclusion

Future directions:

- Can we construct ACE from a single algebraic assumption?

Conclusion

Future directions:

- Can we construct ACE from a single algebraic assumption?
- Connection to other cryptographic primitives?

Conclusion

Future directions:

- Can we construct ACE from a single algebraic assumption?
- Connection to other cryptographic primitives?

Thanks!