Multi-Client Functional Encryption with Public Inputs and Strong Security

Ky Nguyen[1], Duong Hieu Phan[3], David Pointcheval[1,2]

Talk is given by: Robert Schädlich[1]

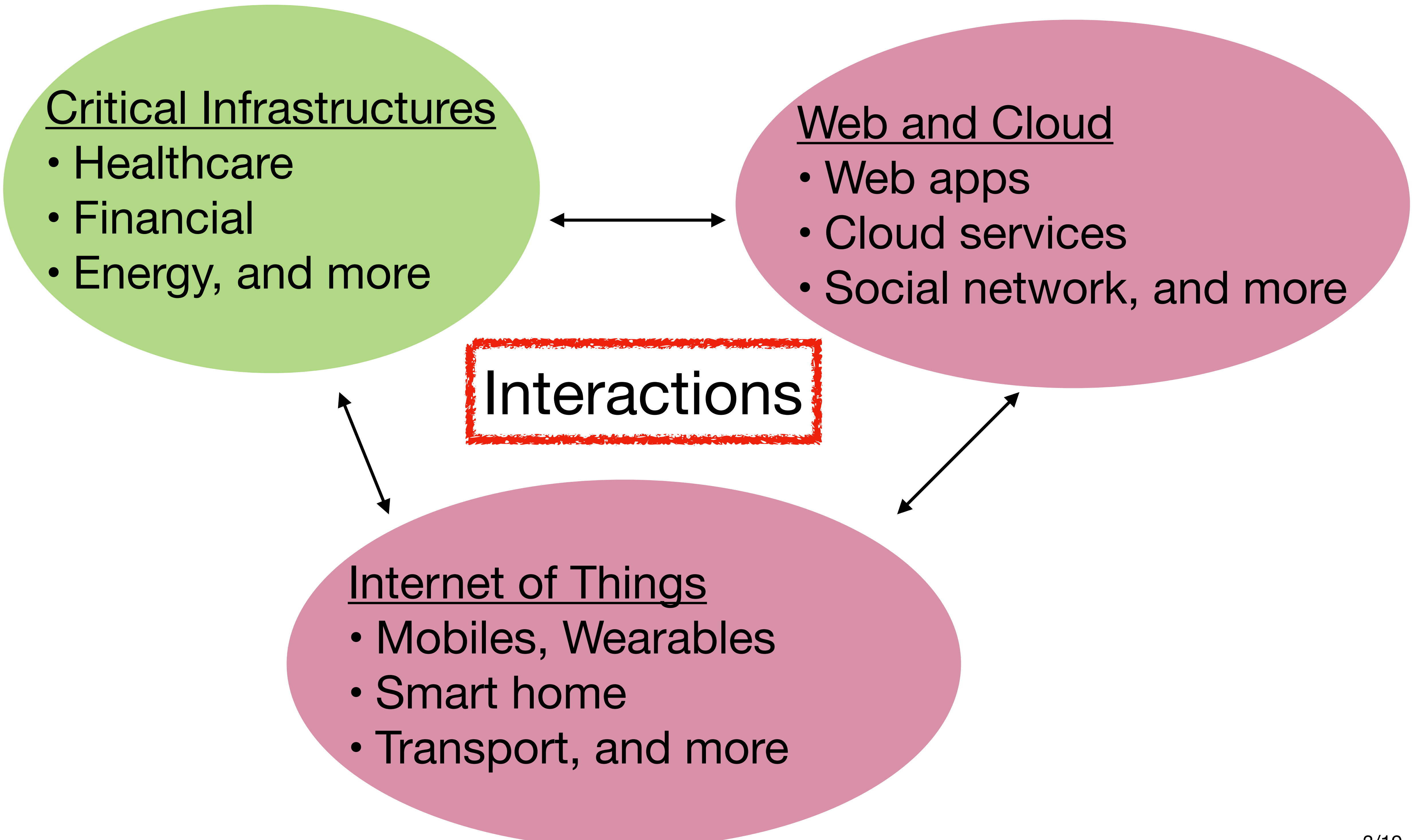
[1] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
[2] Cosmian, Paris, France
[3] LTCI, Telecom Paris, Institut Polytechnique de Paris, France

Outline

1. Motivation: Multiple Senders in Functional Encryption
2. Related Works
   a. Fine-Grained Access Control in Functional Encryption
   b. Revisiting Existing Security Notions
3. Our Contributions
4. Technical Highlights: Achieving Adaptive Security
5. Conclusion

Evolution of Information

Multiple sources, Complex Ecosystems

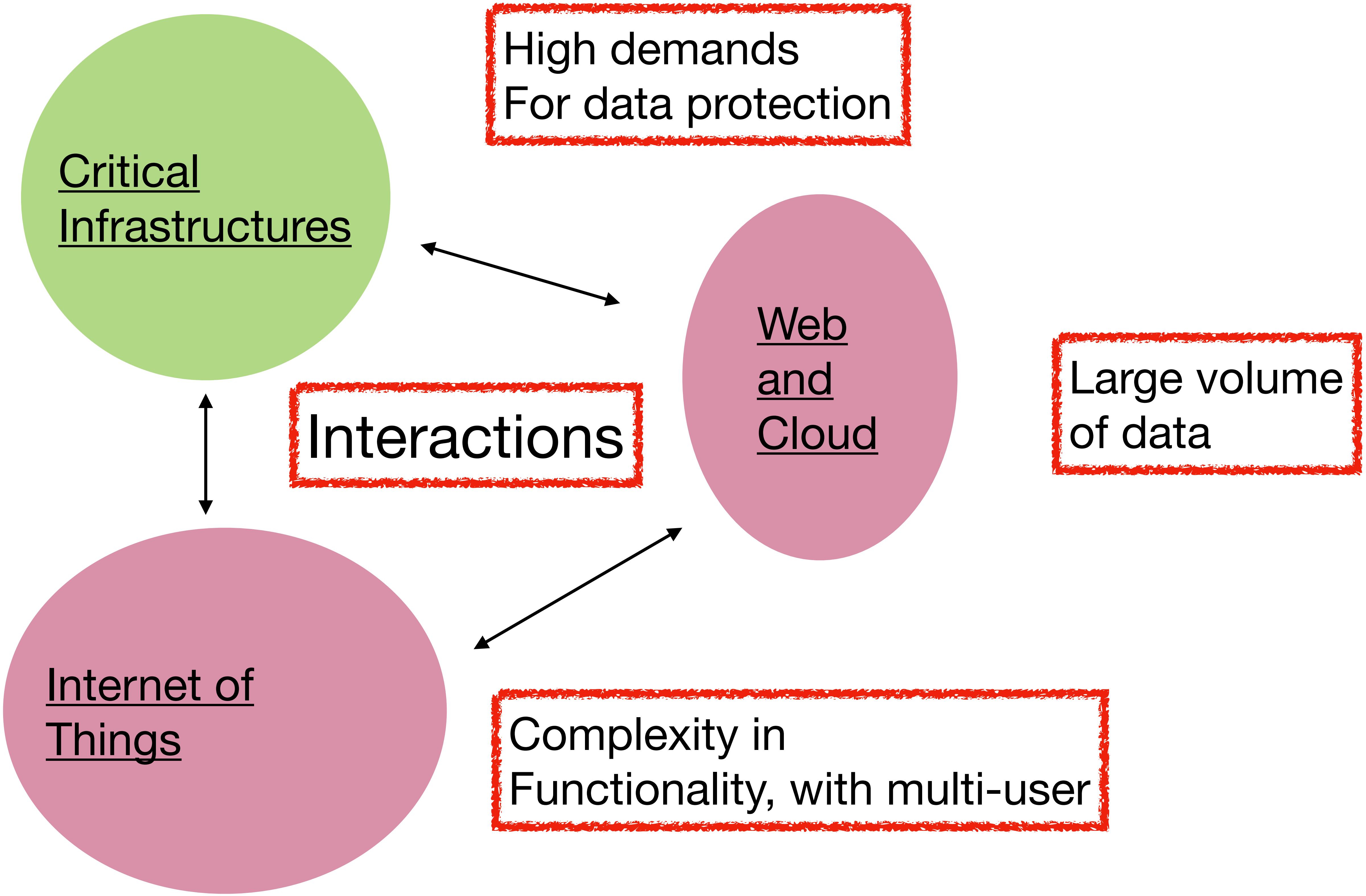


Evolution of Information

Multiple sources, Complex Ecosystems

Evolution of Information

Multiple sources, Complex Ecosystems



Evolution of Information

Multiple sources, Complex Ecosystems

Evolution of Information

Multiple sources, Complex Ecosystems

“Cryptographically”:
- Authentication
- Confidentiality
- Data Integrity

Critical Infrastructures

Web and Cloud

Interactions

+ Multiple Users

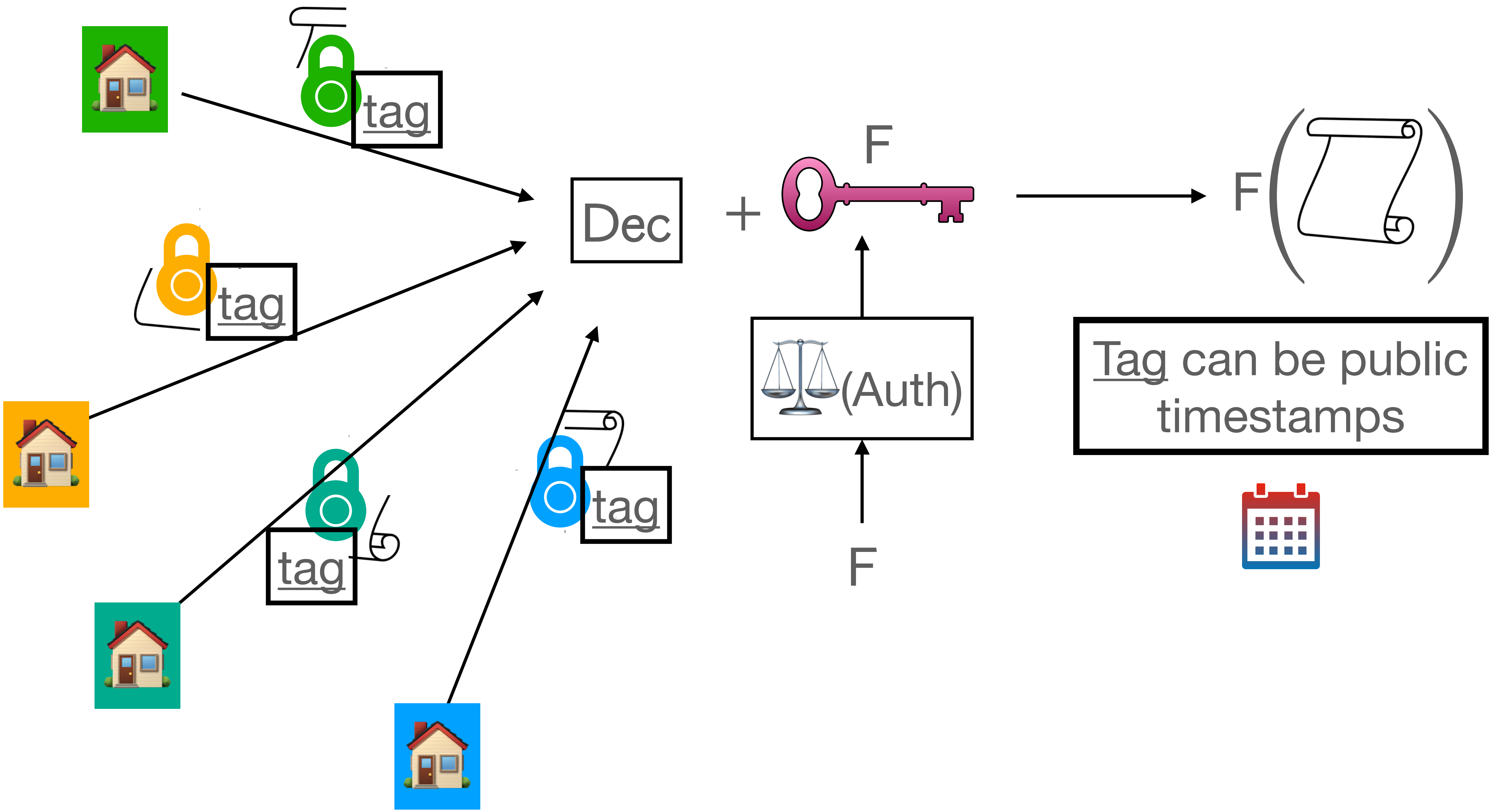
+ Fine-Grained Analysis

Internet of Things

One potential approach:

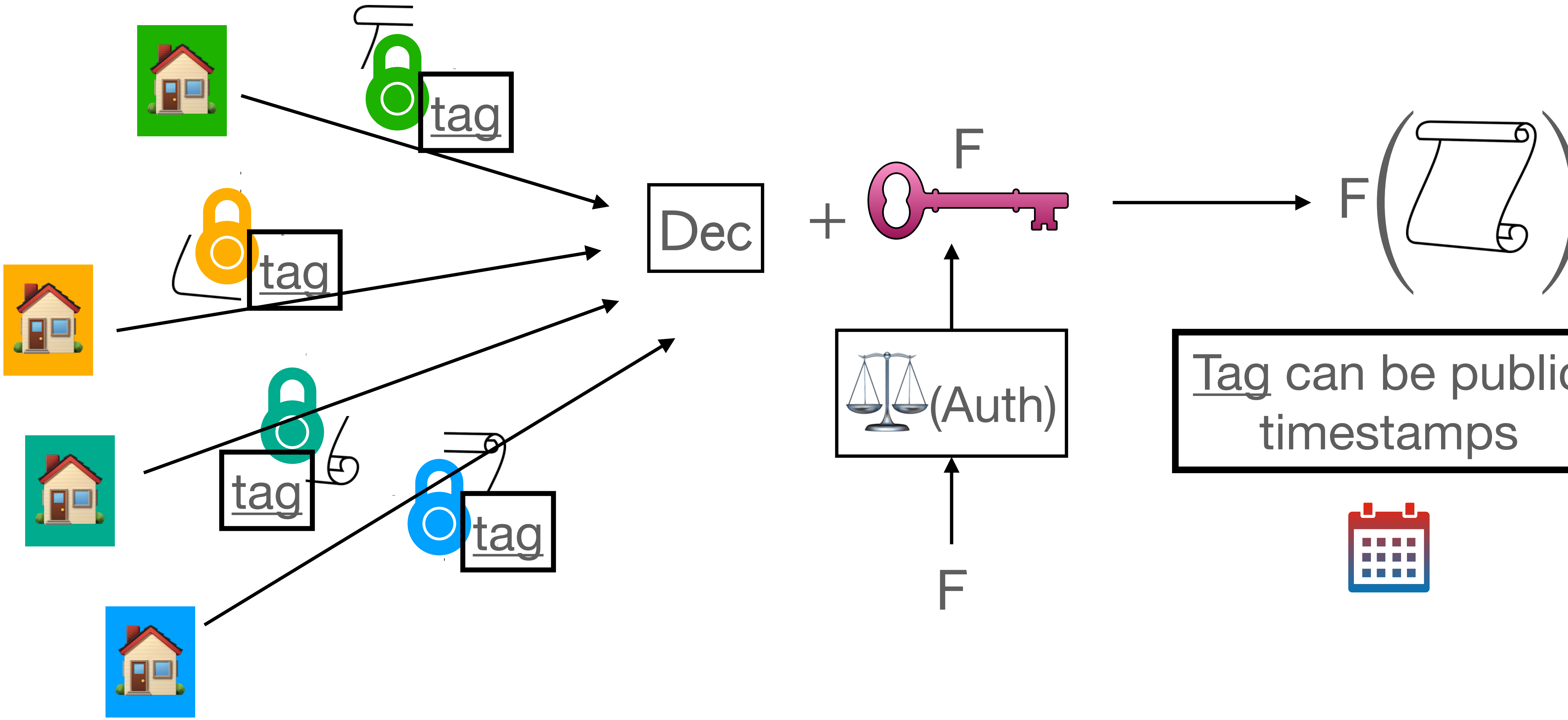
Sensitive Data are Encrypted
+ Fine-Grained Decryption

Multi-Client Functional Encryption (MCFE) [GGG+14,CDGPP18]



MCFE [GGG+14,CDGPP18]

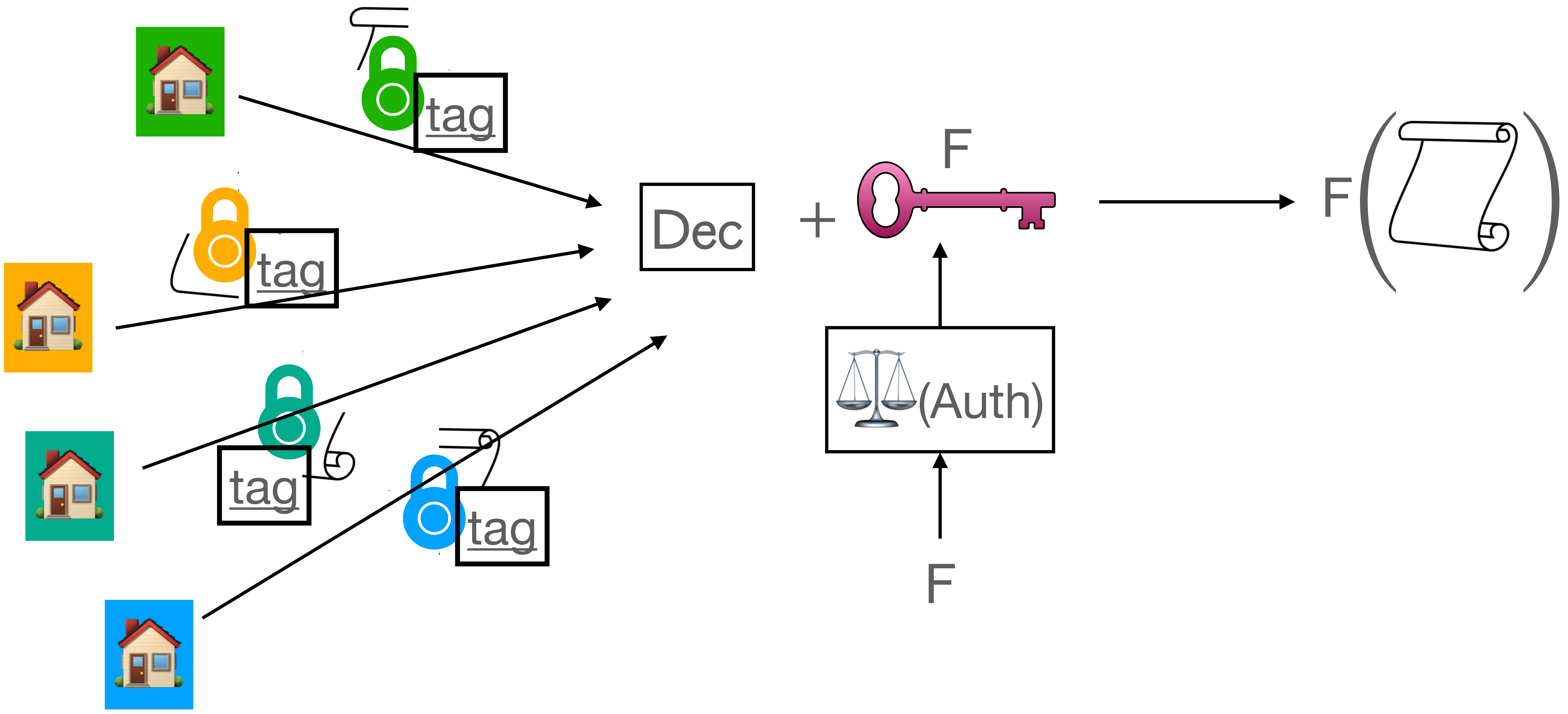
How to Encrypt?



Tag can be public
timestamps

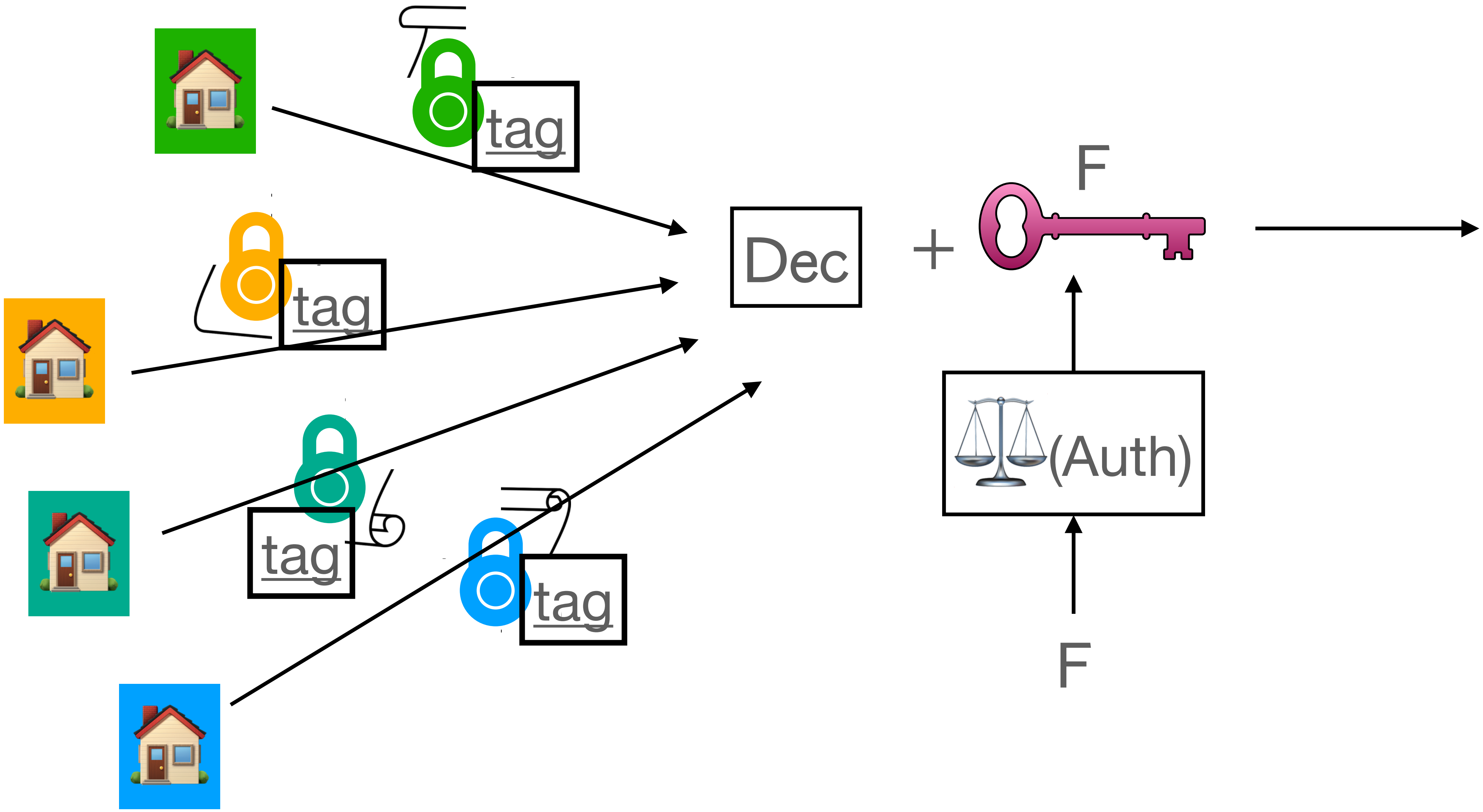
MCFE [GGG+14,CDGPP18]

How to Encrypt?



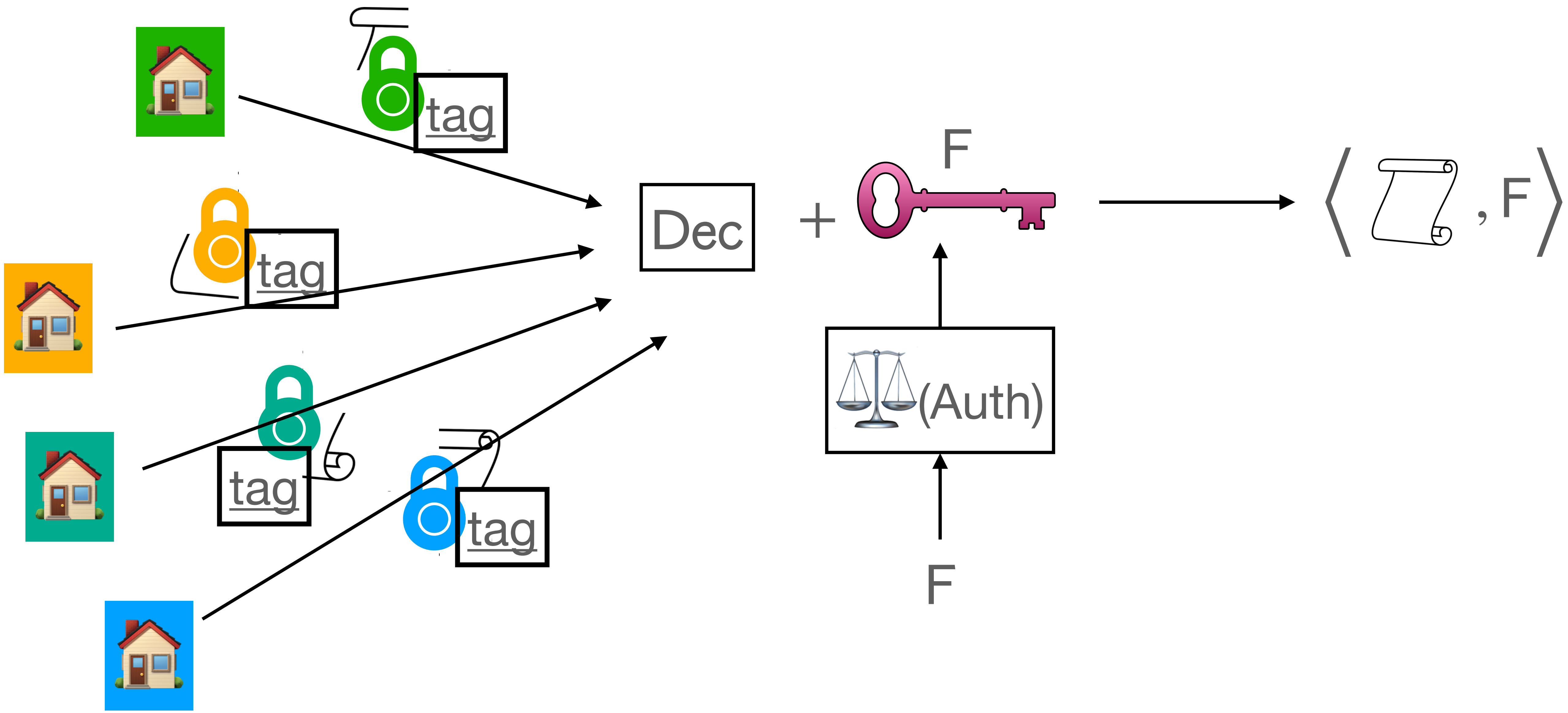
MCFE [GGG+14,CDGPP18]

How to Encrypt?



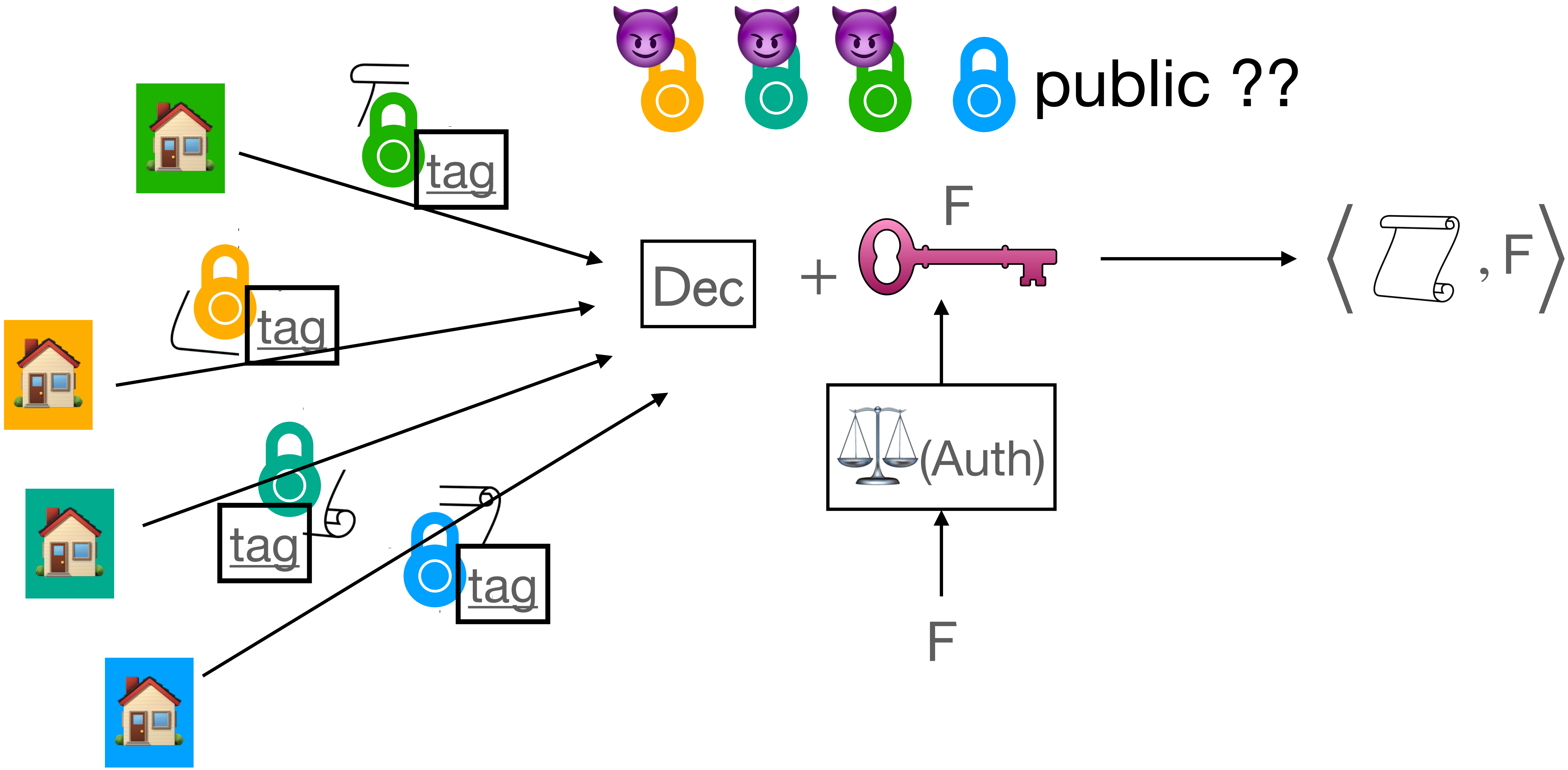
MCFE [GGG+14,CDGPP18]

How to Encrypt?



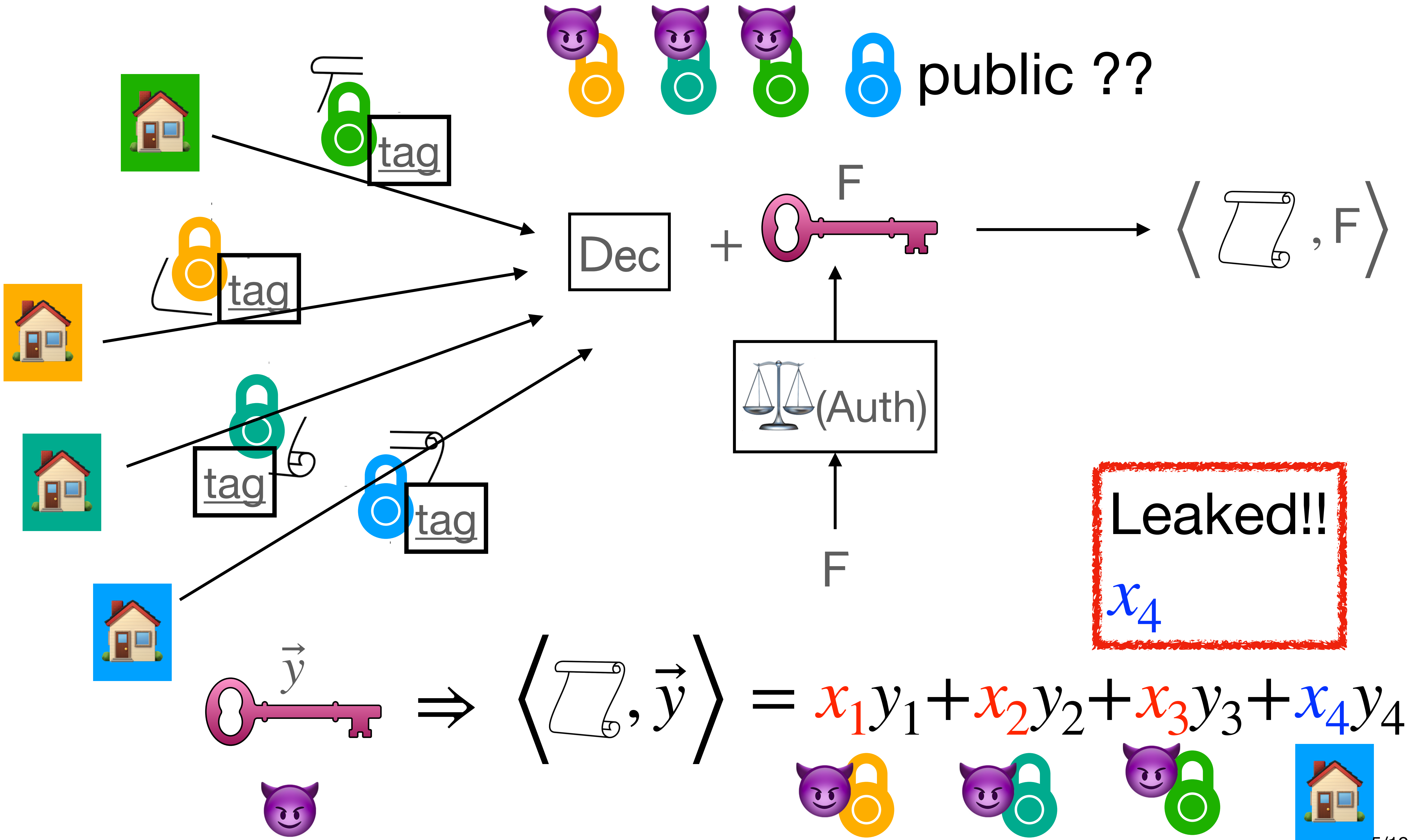
MCFE [GGG+14,CDGPP18]

How to Encrypt?



MCFE [GGG+14,CDGPP18]

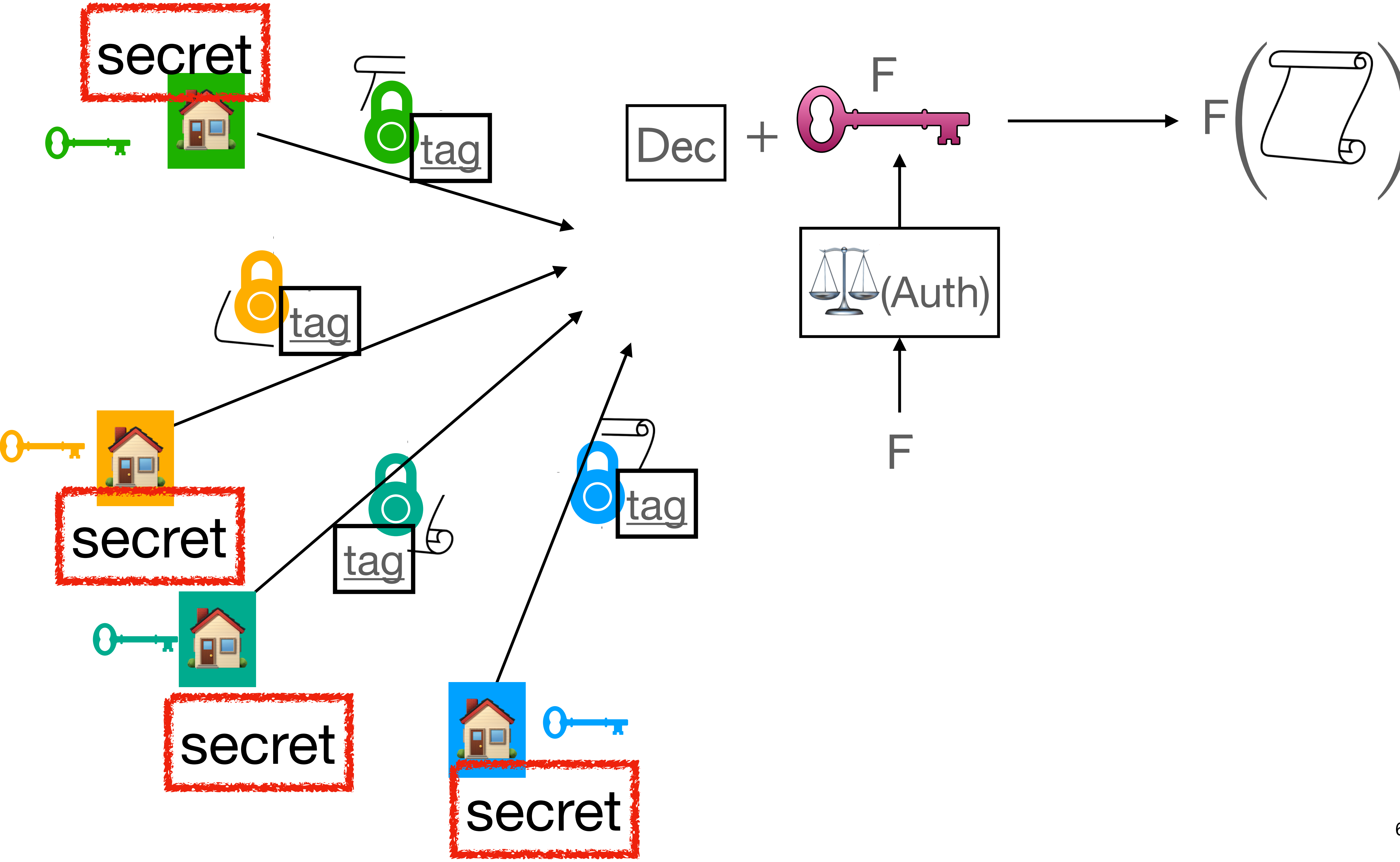
How to Encrypt?



$$\vec{y} \Rightarrow \langle \;\cdot\;, \vec{y} \rangle = x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4$$

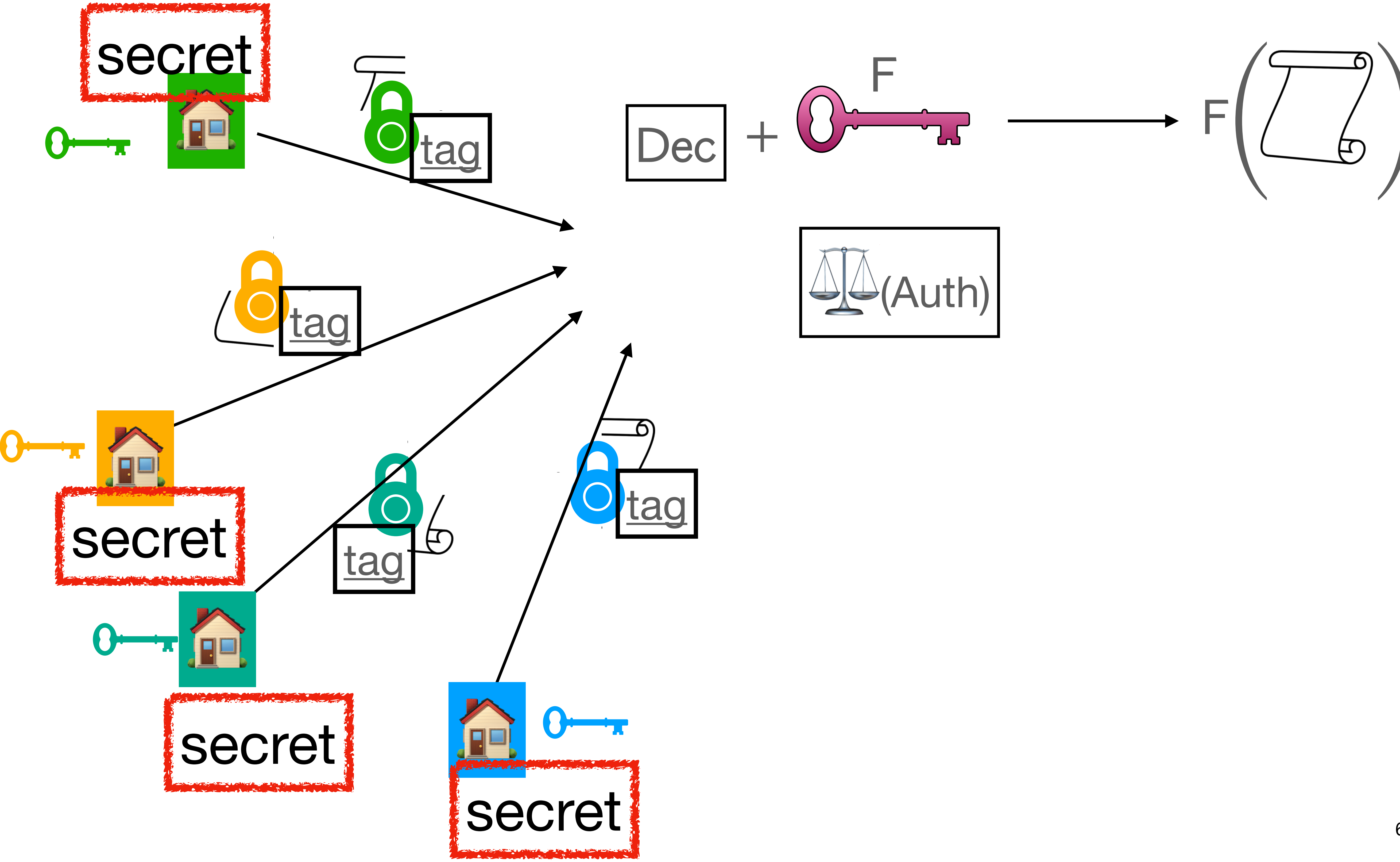
Changes in Model

MCFE [GGG+14,CDGPP18] - Secret Encryption Keys



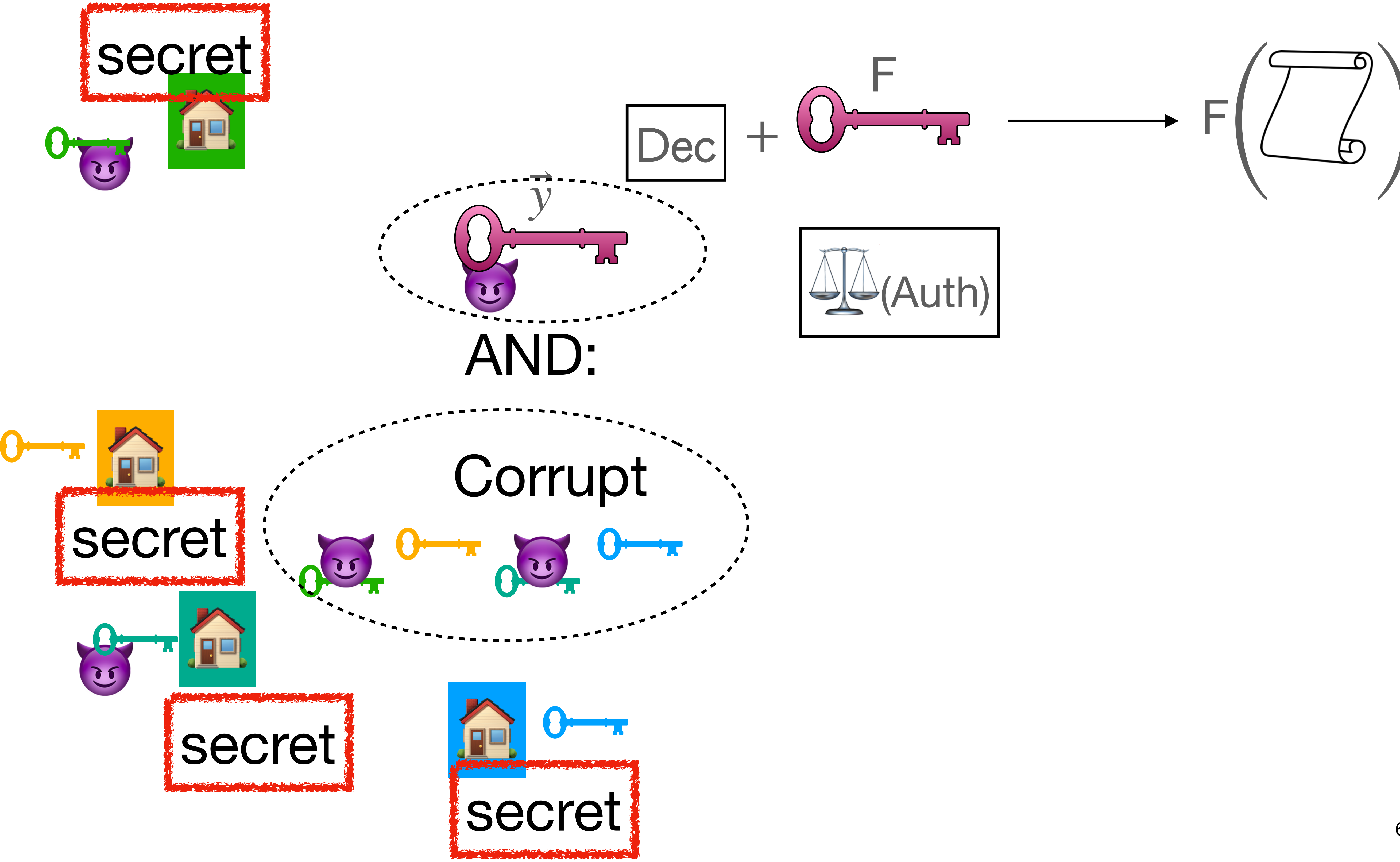
Changes in Model

MCFE [GGG+14,CDGPP18] - Secret Encryption Keys



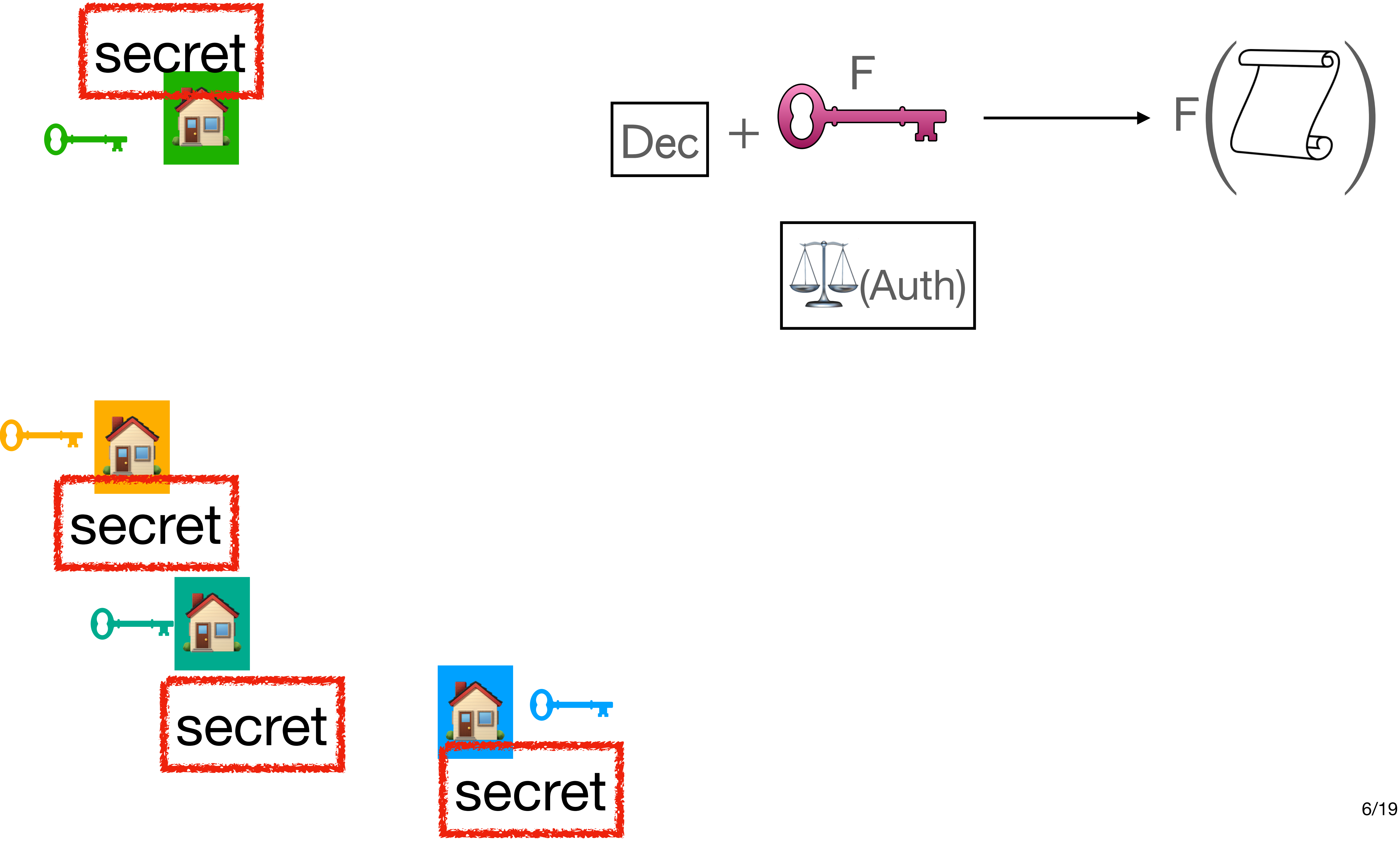
Changes in Model

MCFE [GGG+14,CDGPP18] - Secret Encryption Keys



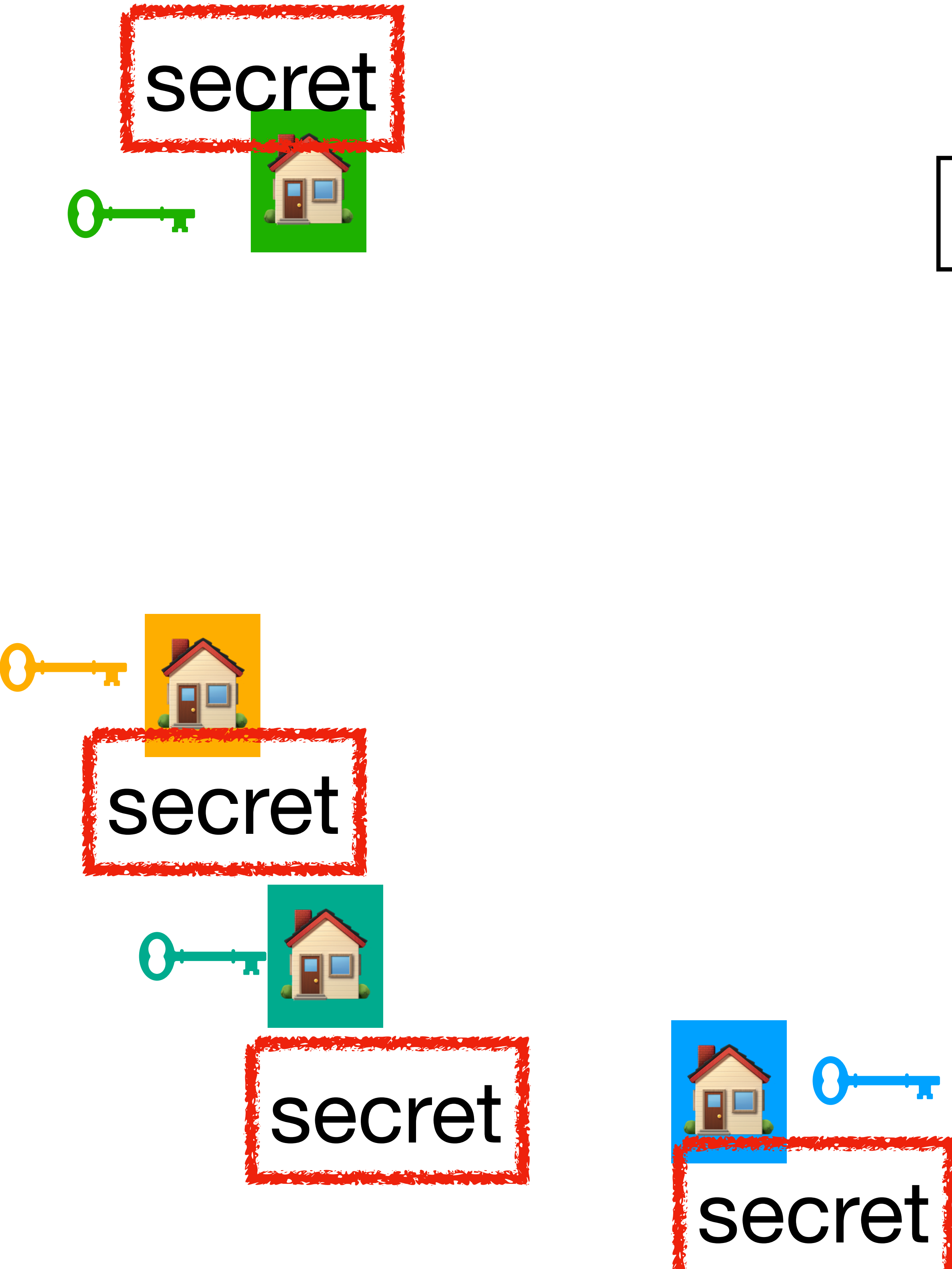
Changes in Model

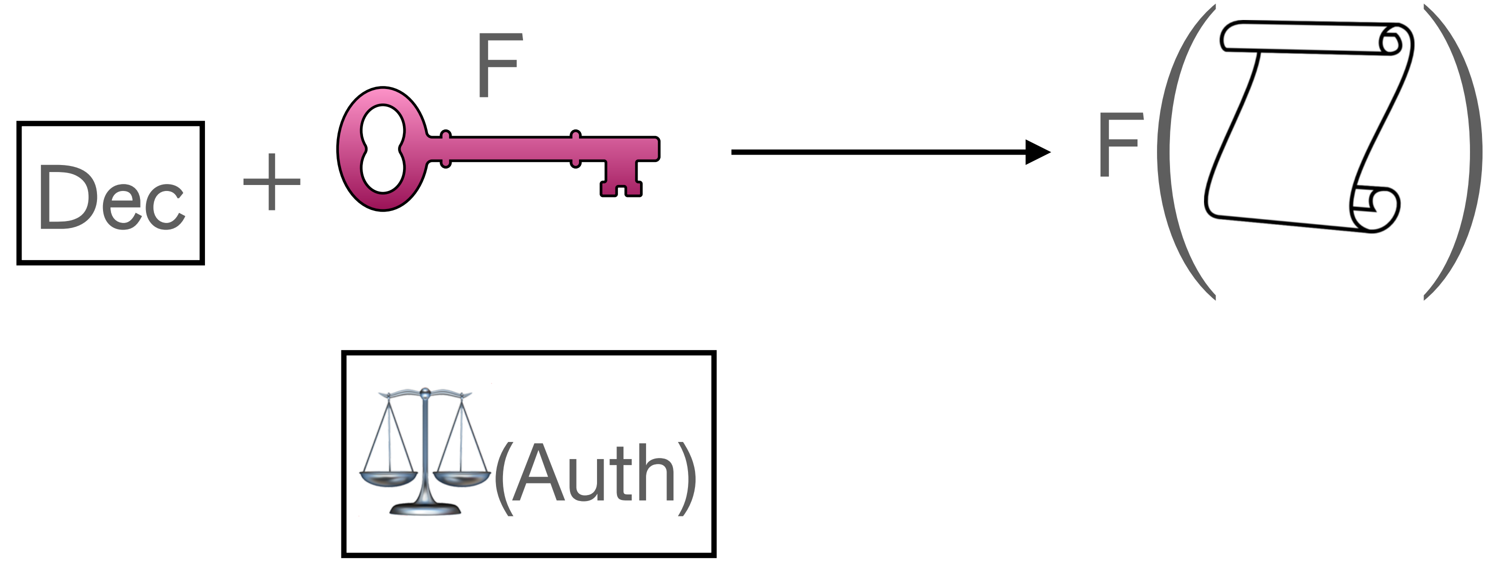
MCFE [GGG+14,CDGPP18] - Secret Encryption Keys



Changes in Model

MCFE [GGG+14,CDGPP18] - Secret Encryption Keys





[GGG+14,CDGPP18]
“vanilla” syntax

F( ) :

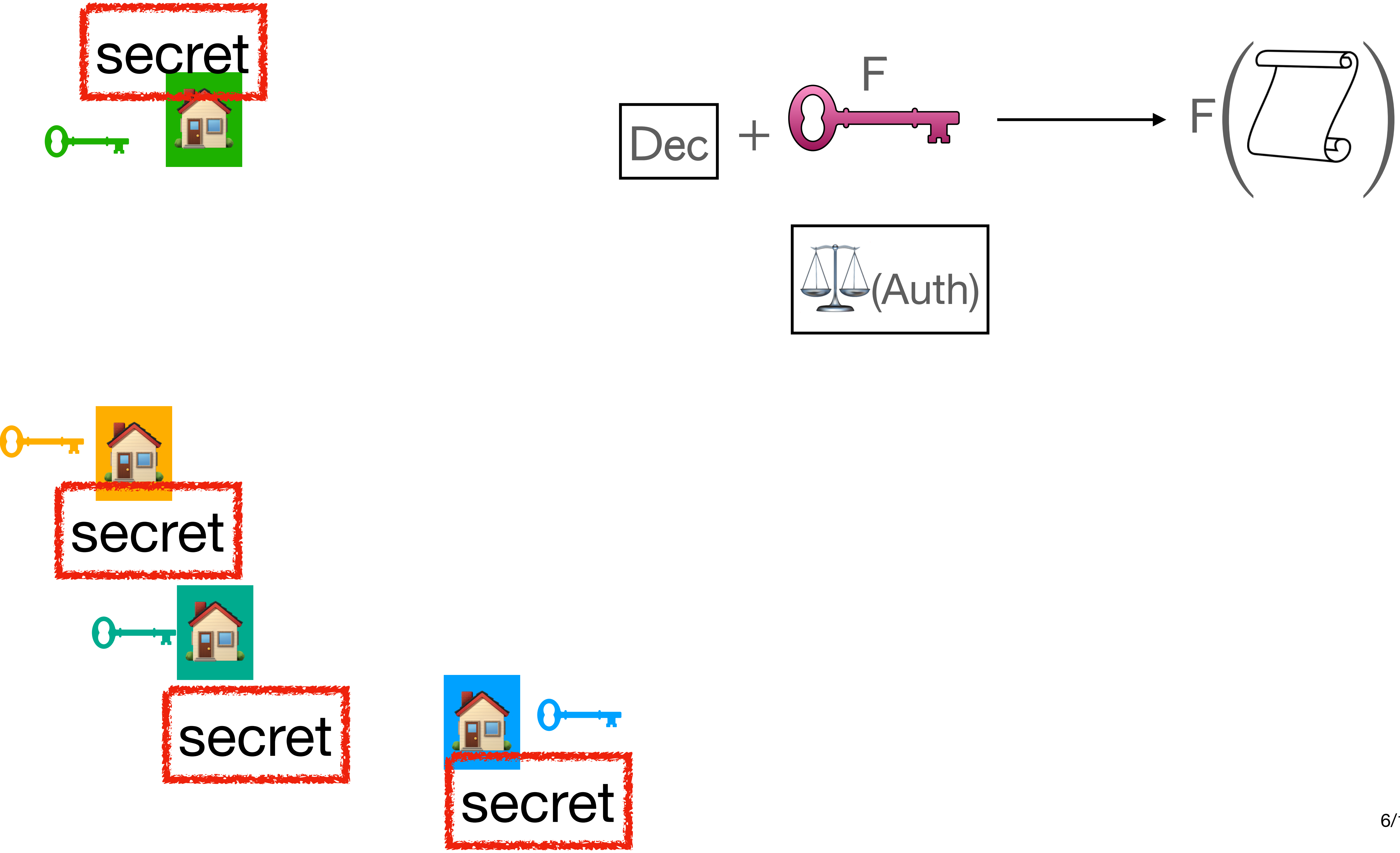
Parse Att, =

If Pol(Att) = True then

Else “Nothing”

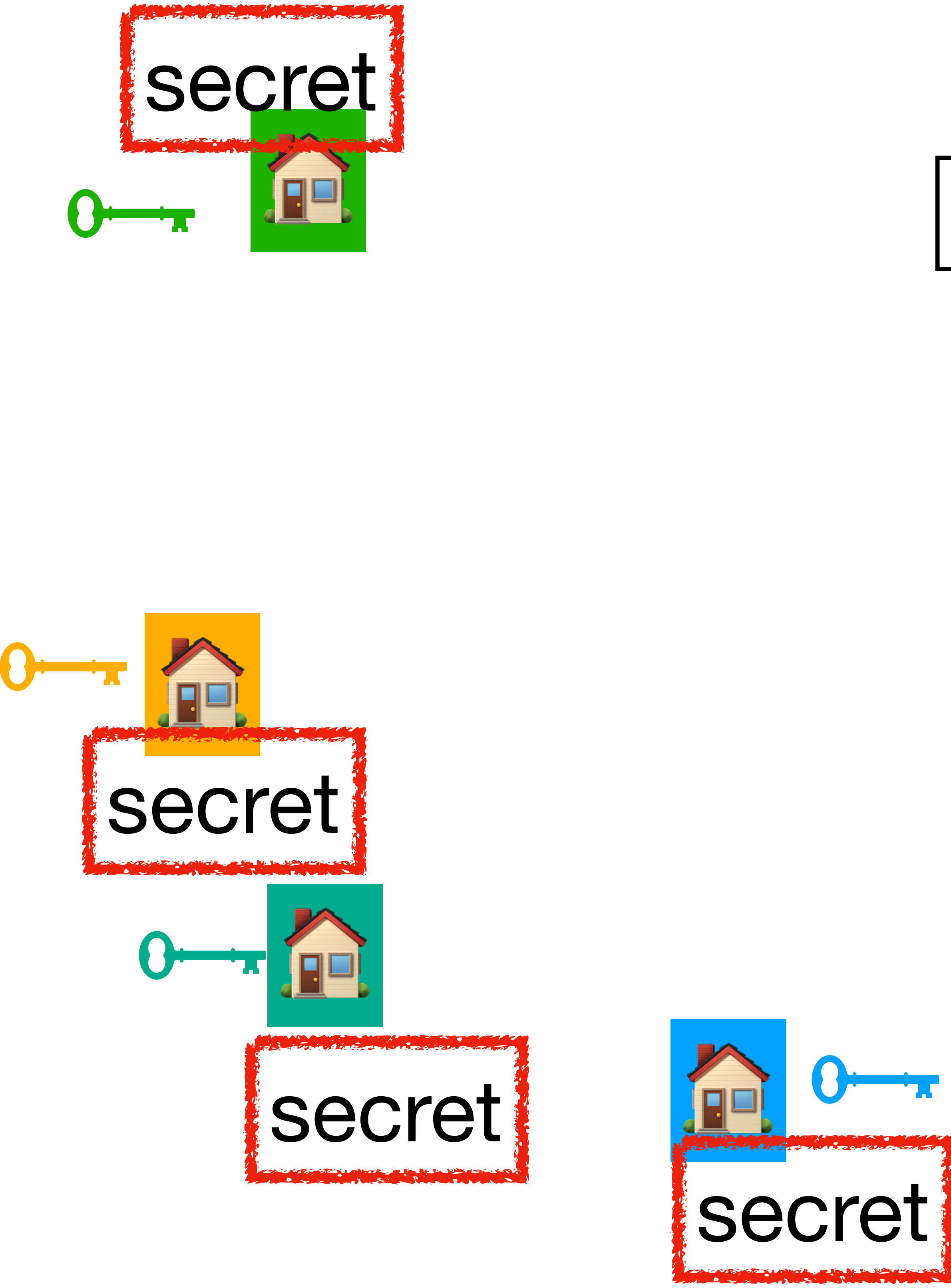
Changes in Model

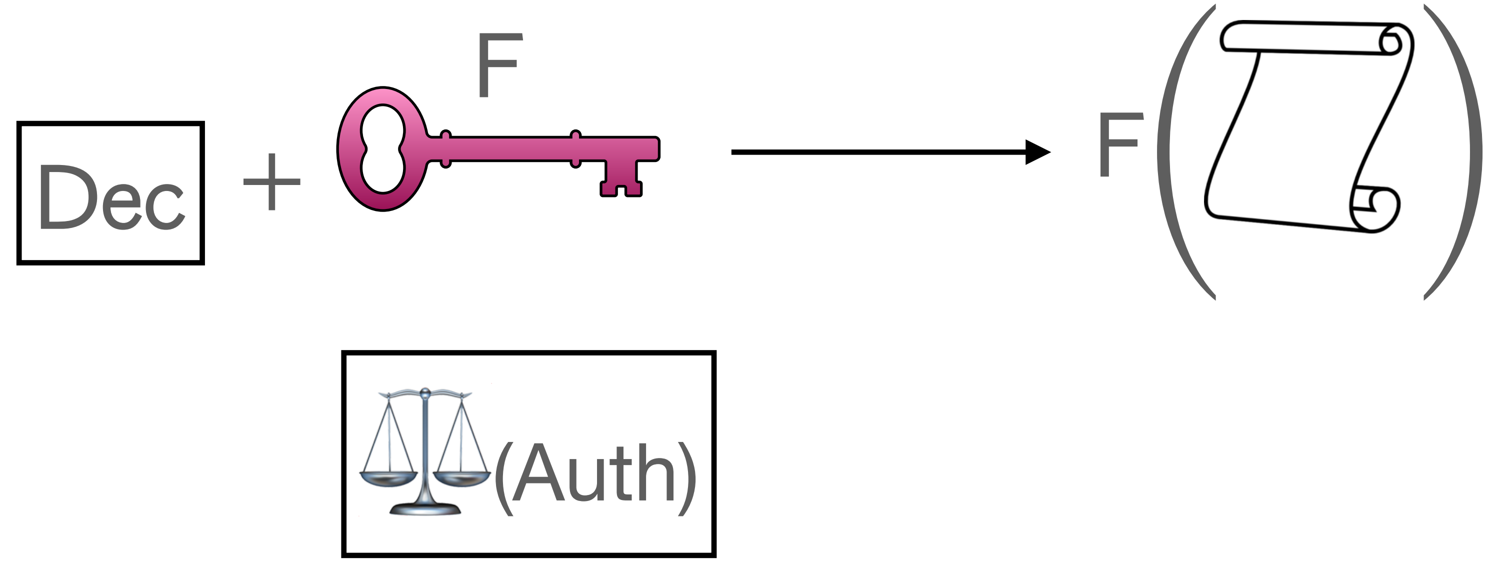
MCFE [GGG+14,CDGPP18] - Secret Encryption Keys



Changes in Model

MCFE [GGG+14,CDGPP18] - Secret Encryption Keys





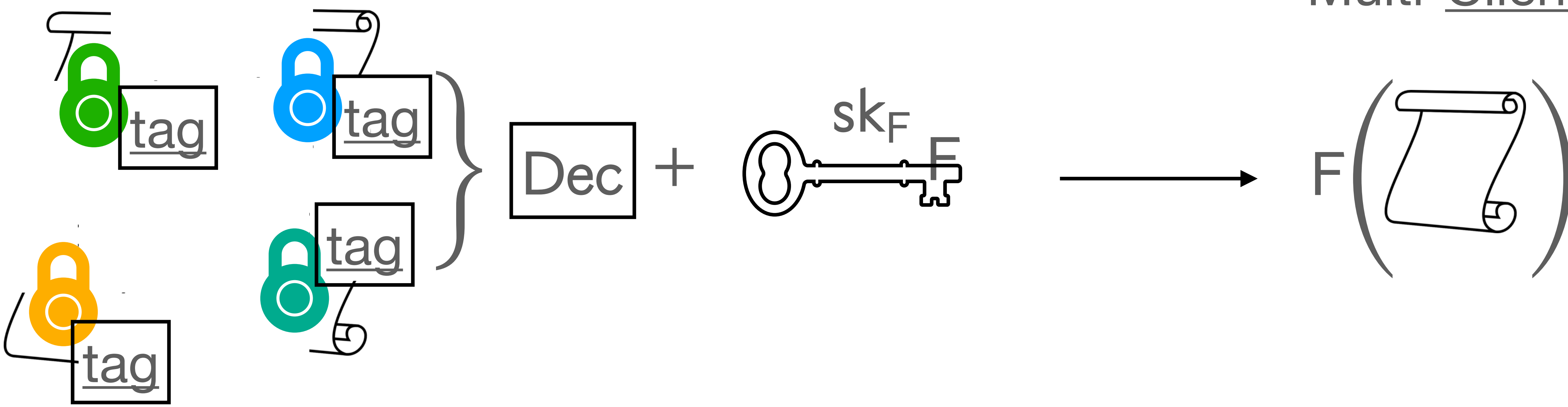
🤔**Syntactical:**

NOT allow public-Att

Motivation - Usage of Keys

MCFE [GGG+14, GKL+13,CDGPP18]

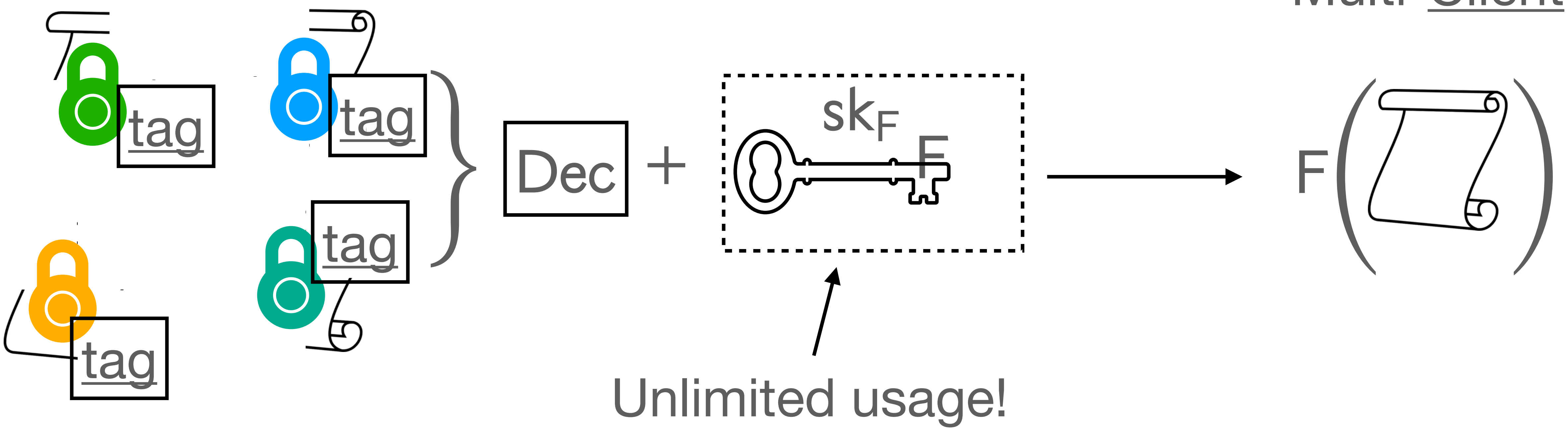
Multi-Client



Motivation - Usage of Keys

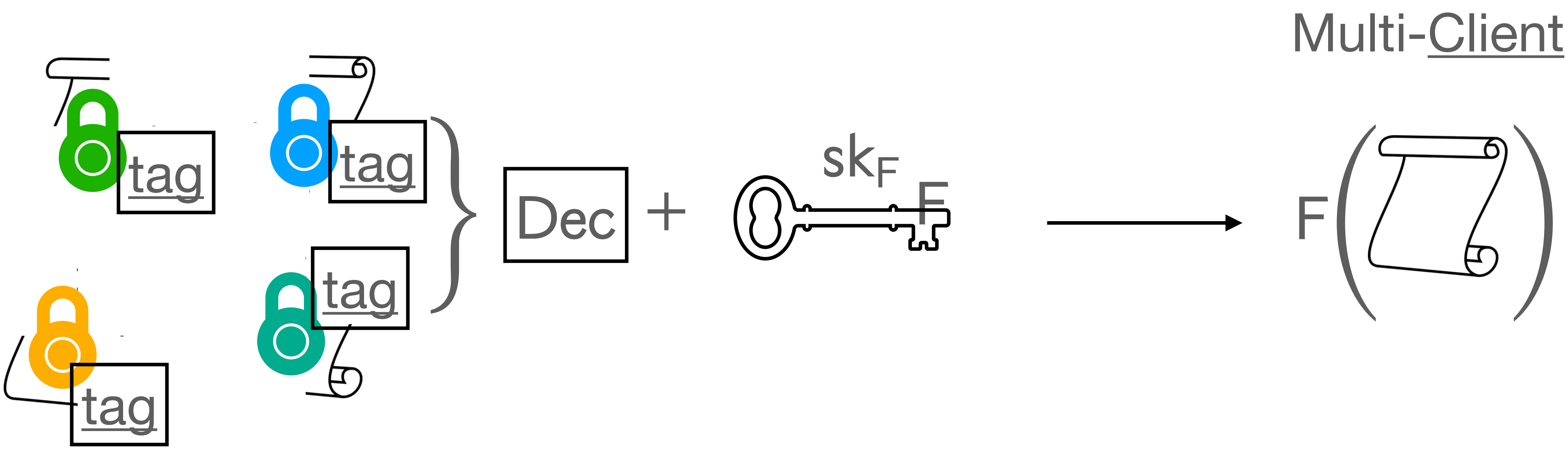
MCFE [GGG+14, GKL+13,CDGPP18]

Multi-Client



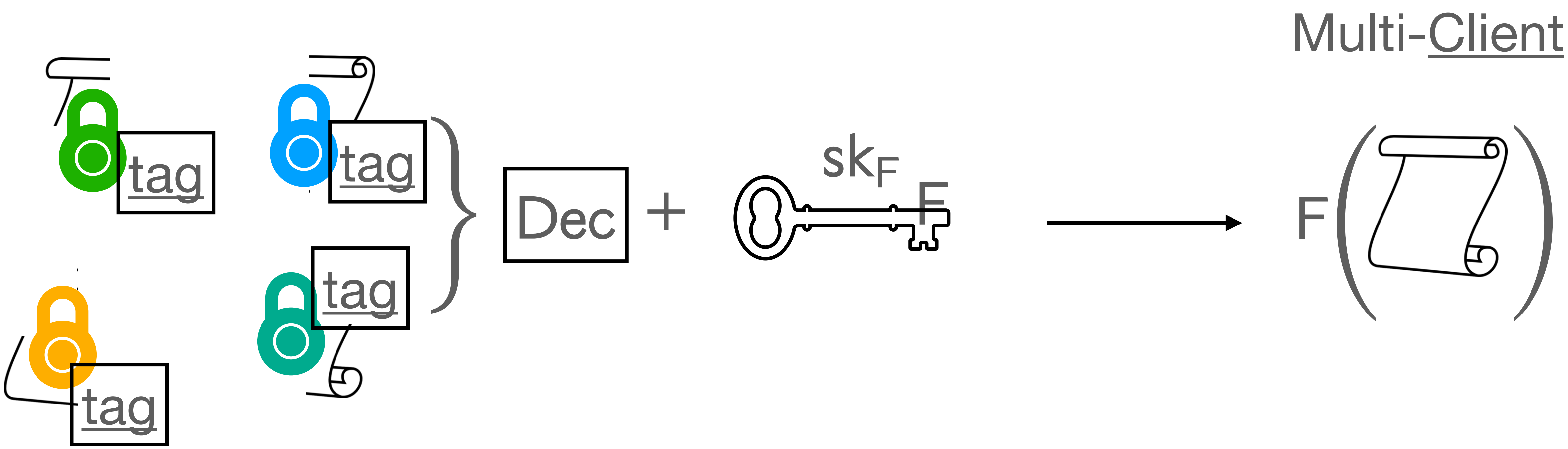
Motivation - Usage of Keys

MCFE [GGG+14, GKL+13,CDGPP18]



Motivation - Usage of Keys

MCFE [GGG+14, GKL+13,CDGPP18]



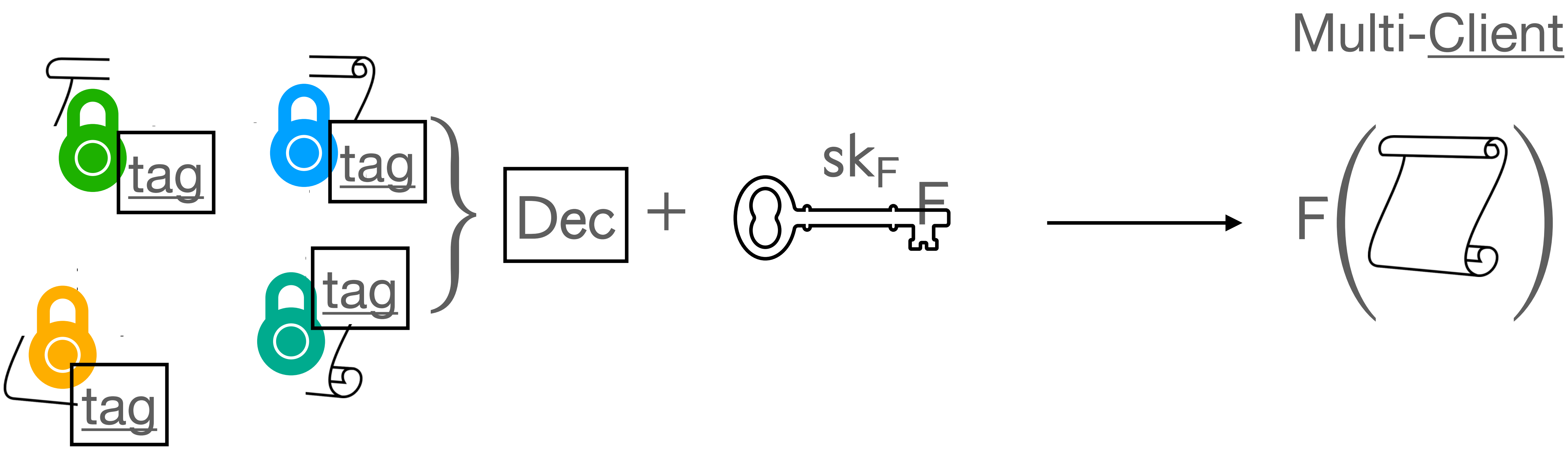
[ACGU20]: control keys
In *Multi-Input* FE [GGG+14]

Secret Encryption Keys

Only one encryptor

Motivation - Usage of Keys

MCFE [GGG+14, GKL+13,CDGPP18]



🤔Efficiency

Related Works

Multi-Client with Fine-Grained Access Control [NPP22]

Related Works

Multi-Client with Fine-Grained Access Control [NPP22]

✓ **First definitions for MCFE with key control**

Related Works

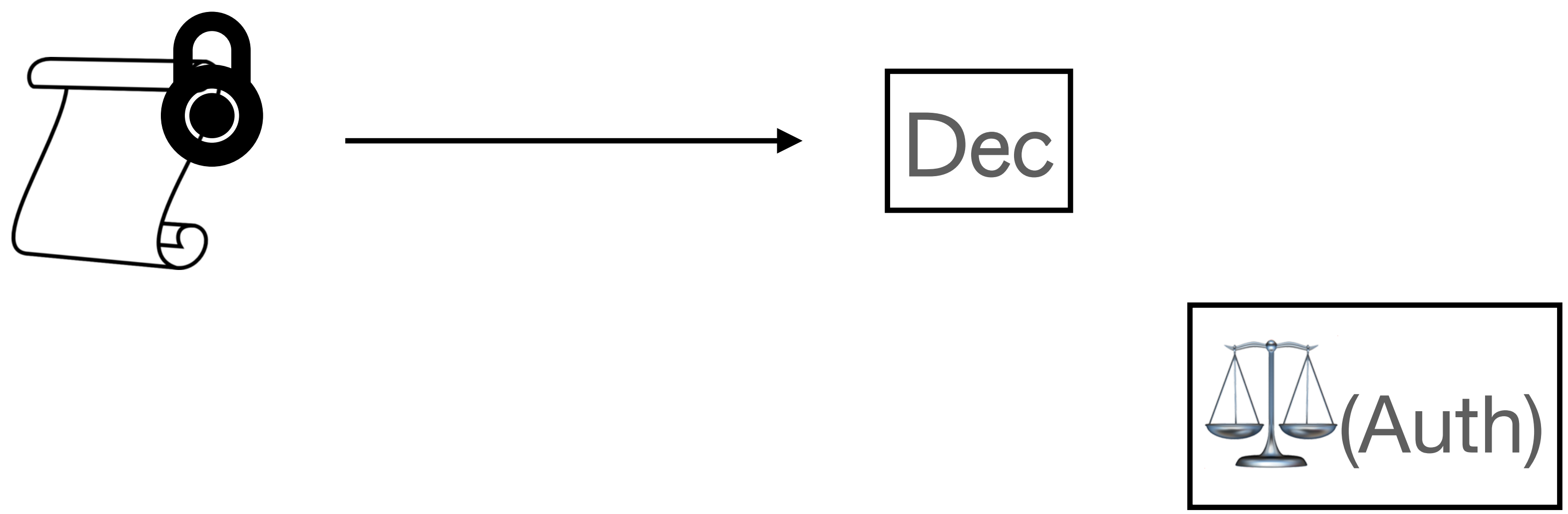
Multi-Client with Fine-Grained Access Control [NPP22]

✓ First definitions for MCFE with key control

✓ Concrete MCFE for scalar Inner Products with Linear Secret Sharing for key controlling

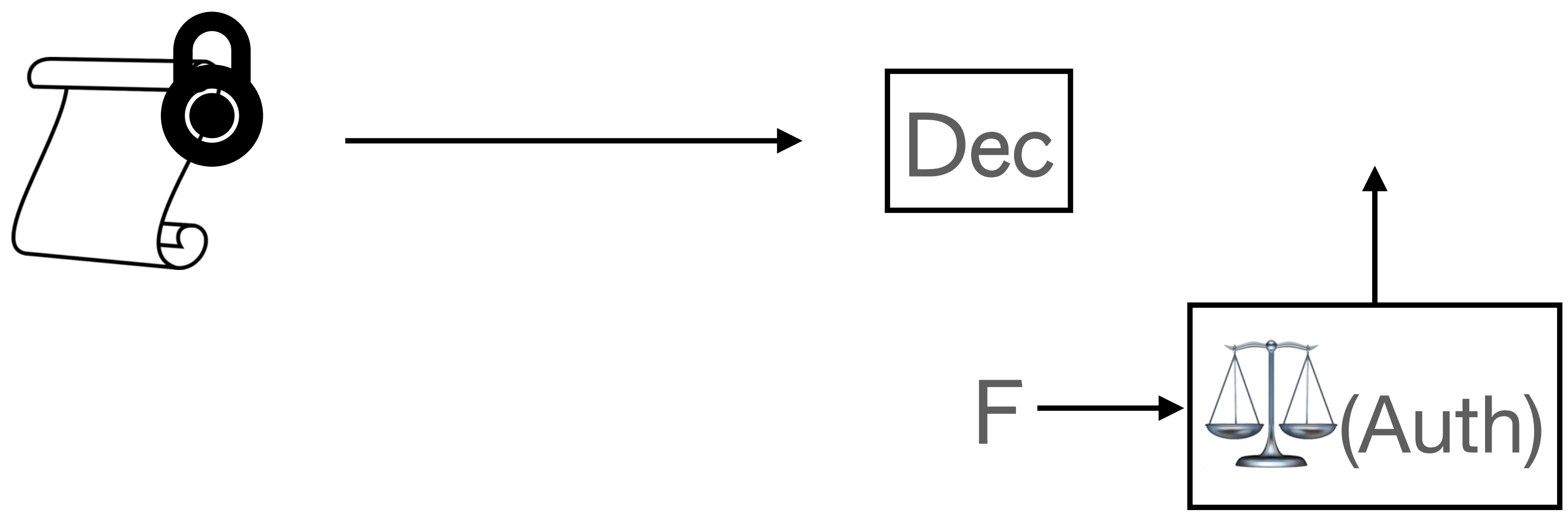
Motivation - Security Model

How do we define a “secure” FE ?



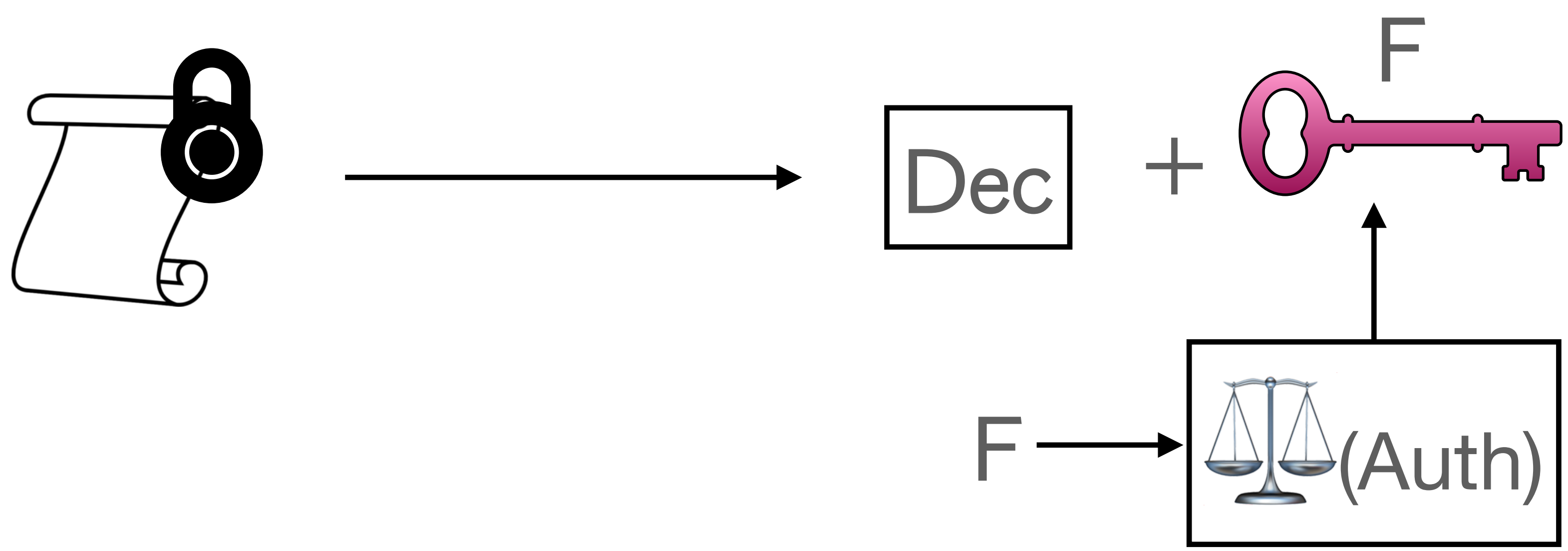
Motivation - Security Model

How do we define a “secure” FE ?



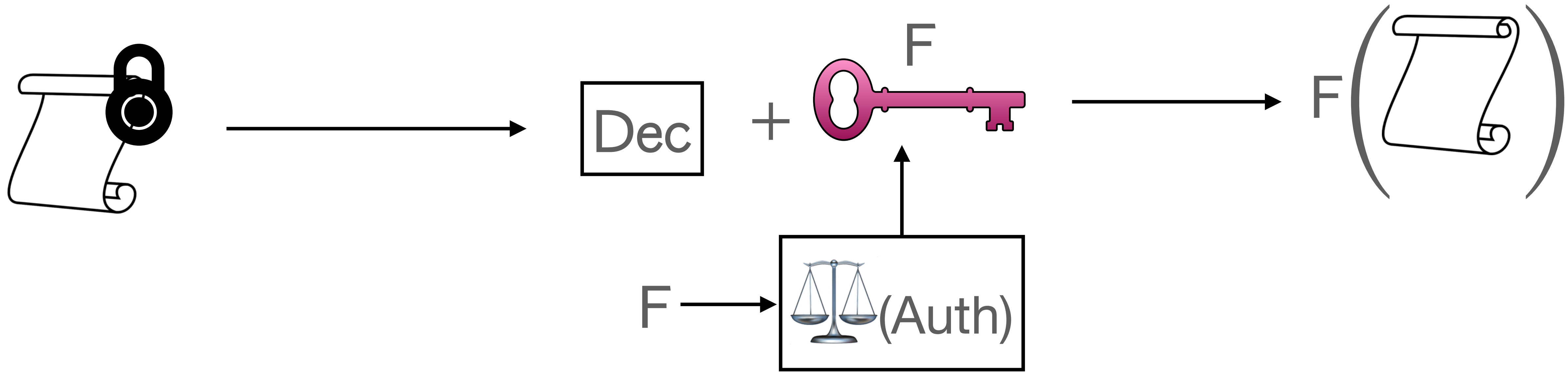
Motivation - Security Model

How do we define a “secure” FE ?



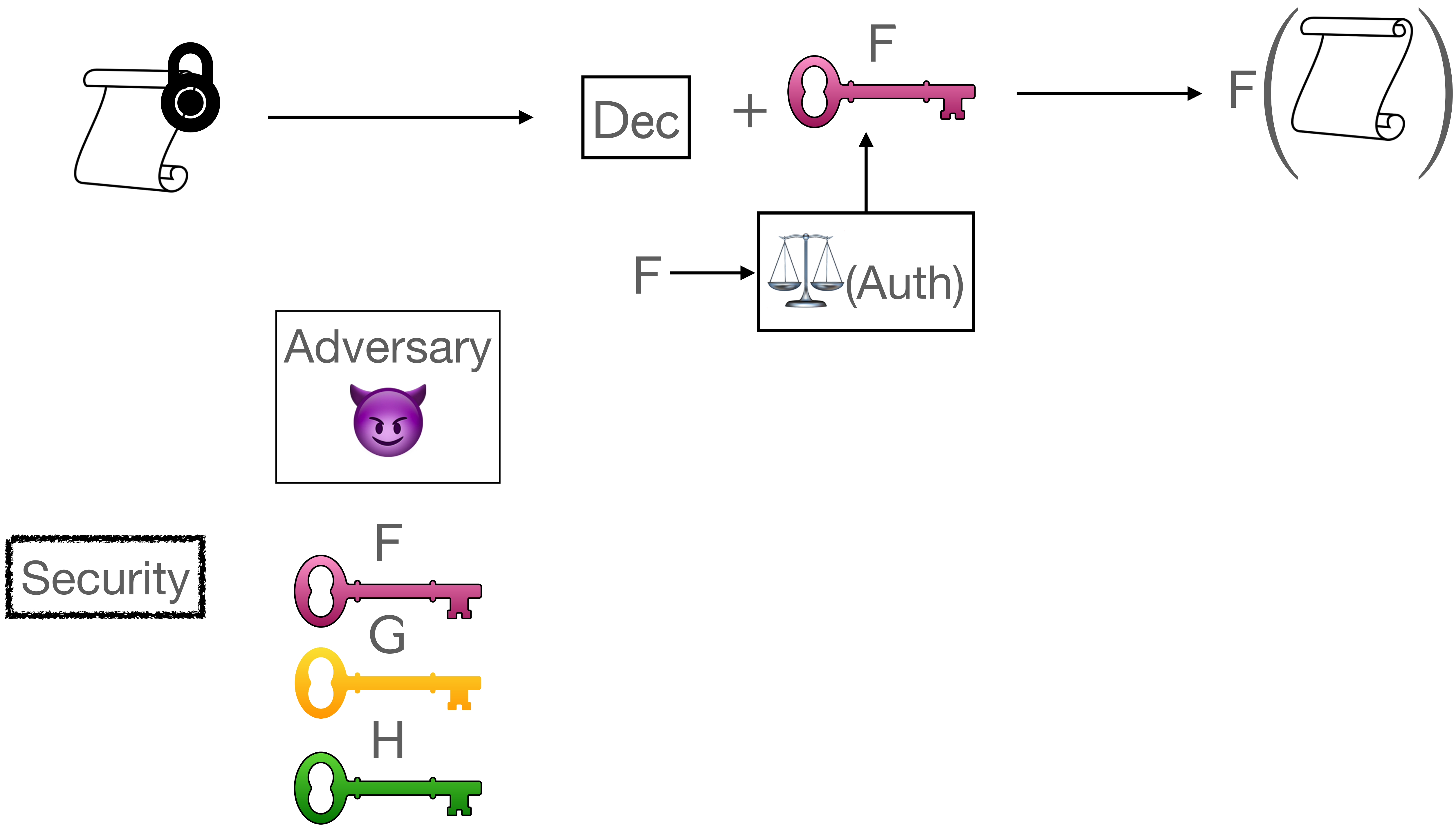
Motivation - Security Model

How do we define a “secure” FE ?



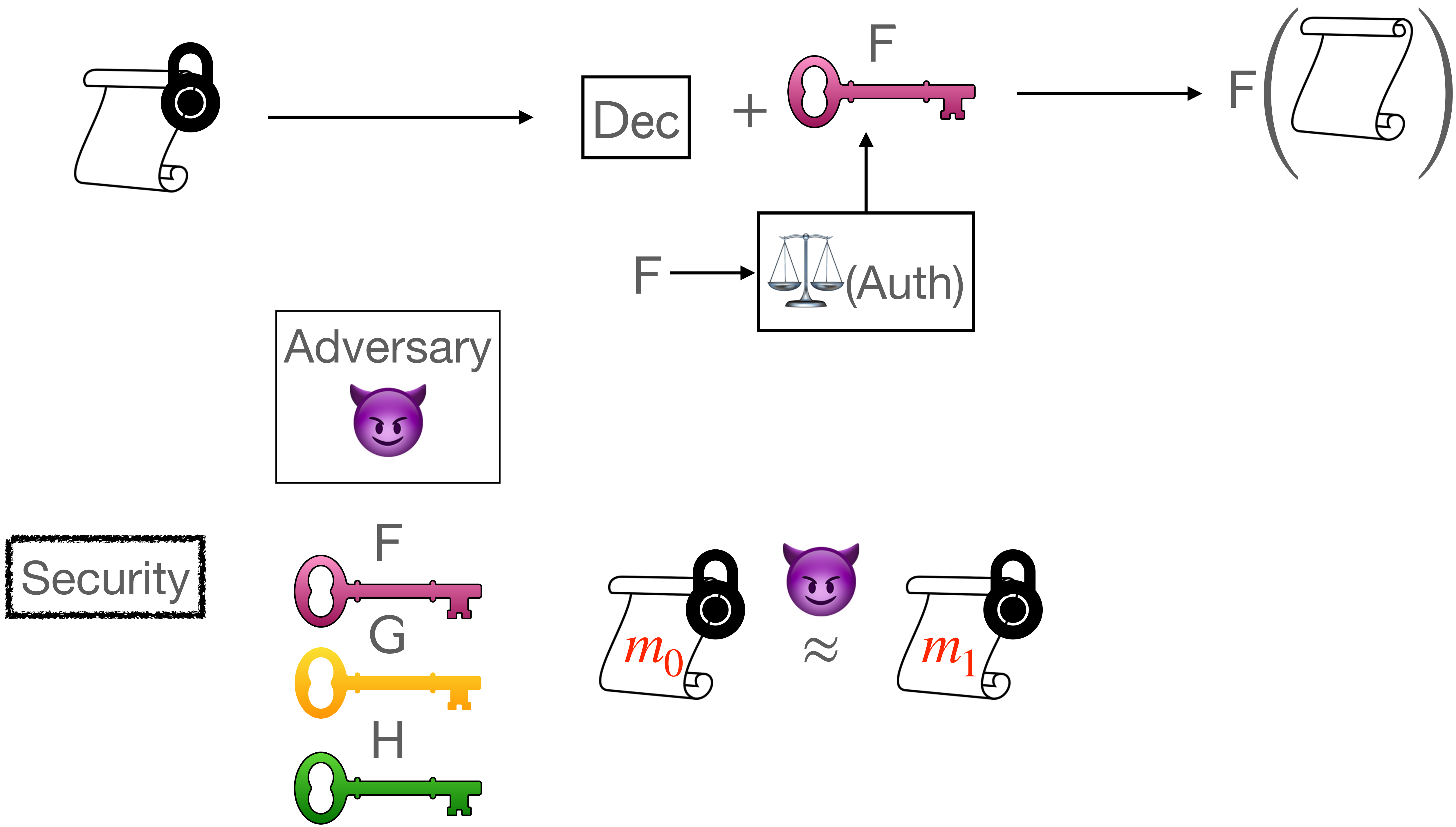
Motivation - Security Model

How do we define a “secure” FE ?



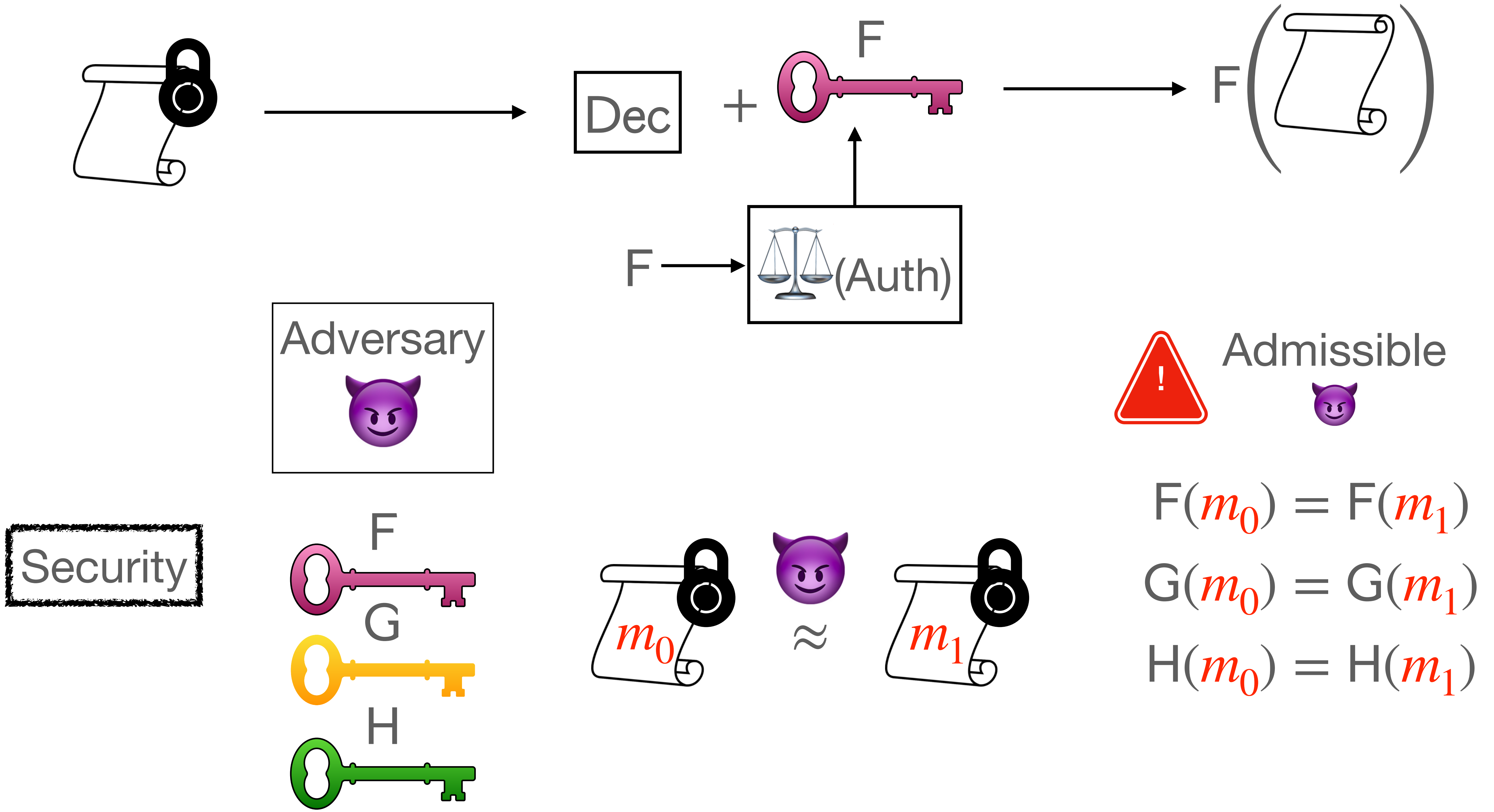
Motivation - Security Model

How do we define a “secure” FE ?



Motivation - Security Model

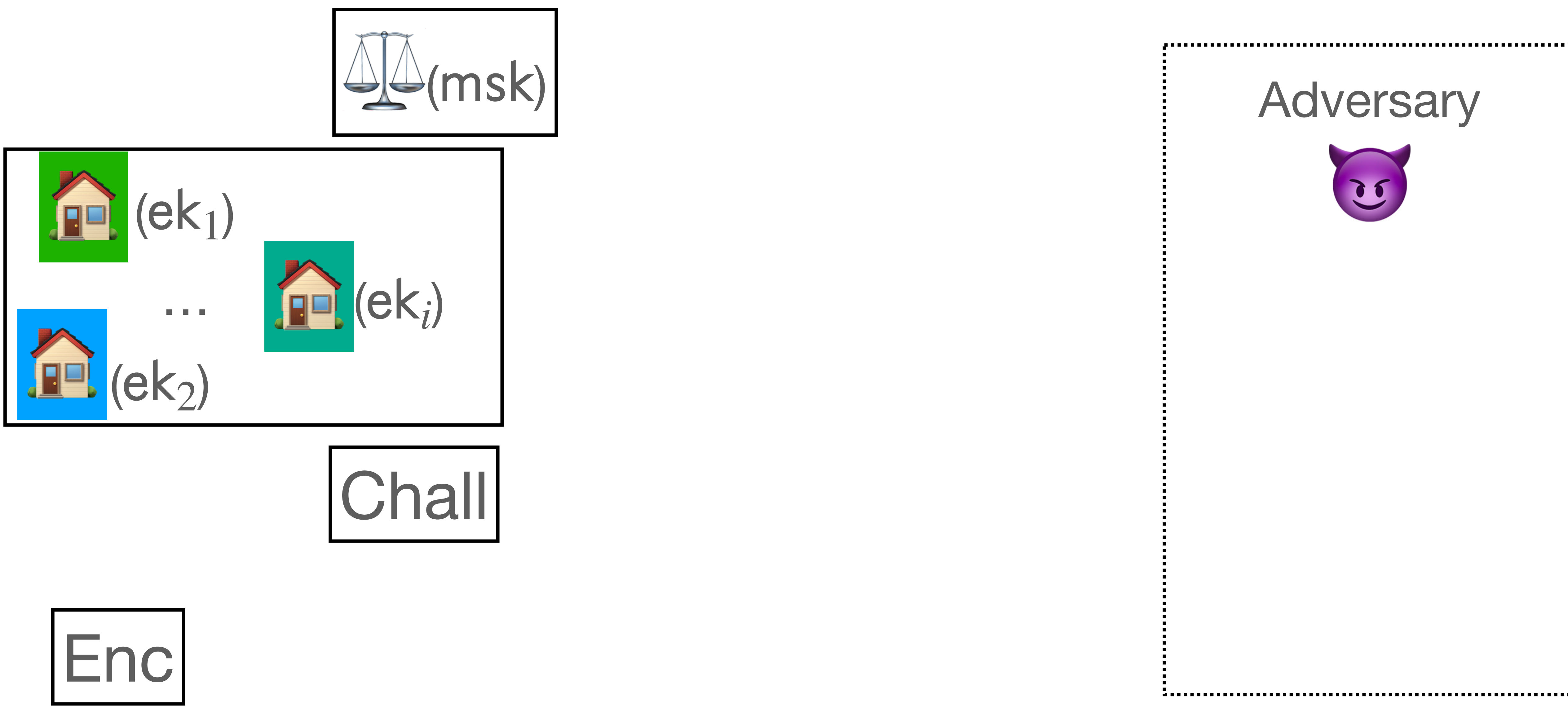
How do we define a “secure” FE ?



Motivation - Security Model

How do we define a “secure” MCFE ?

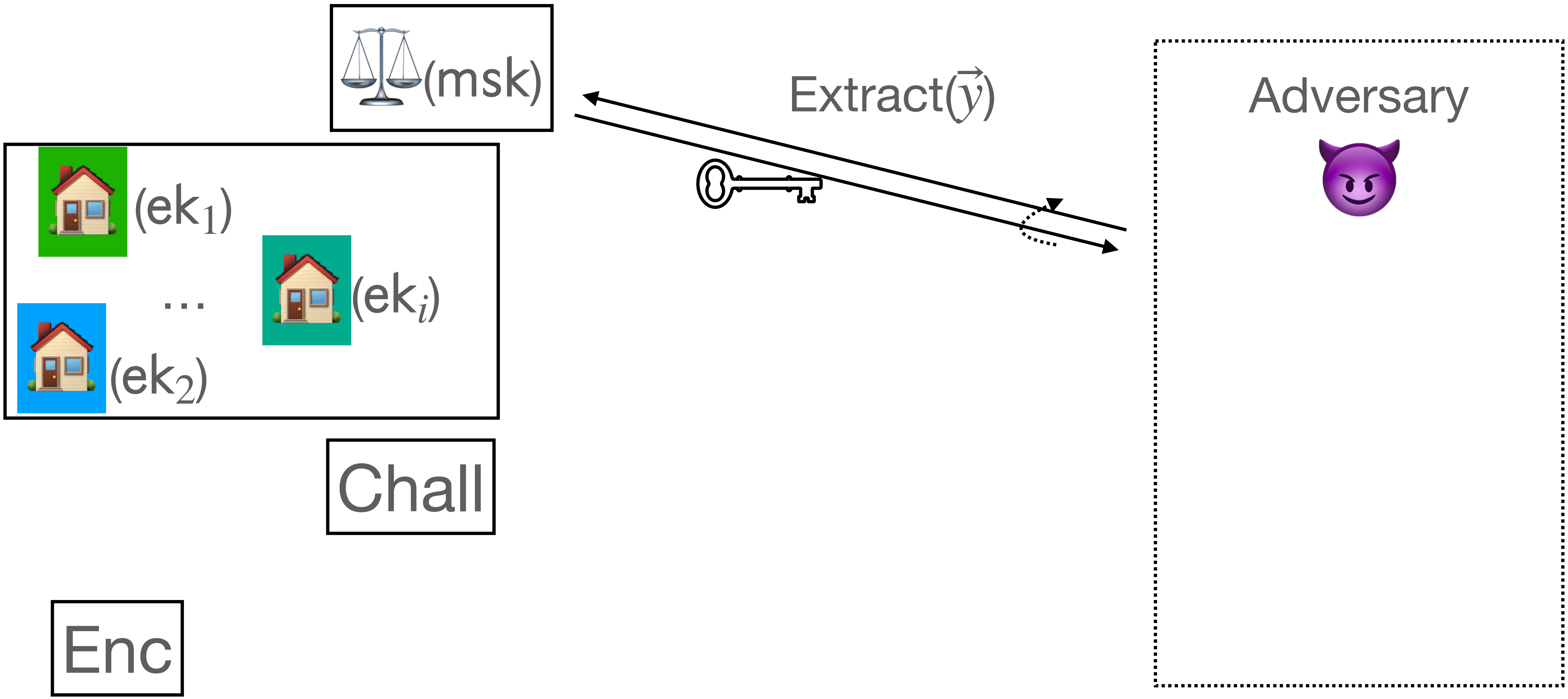
Challenger: $b \xleftarrow{\$} \{0,1\}$



Motivation - Security Model

How do we define a "secure" MCFE ?

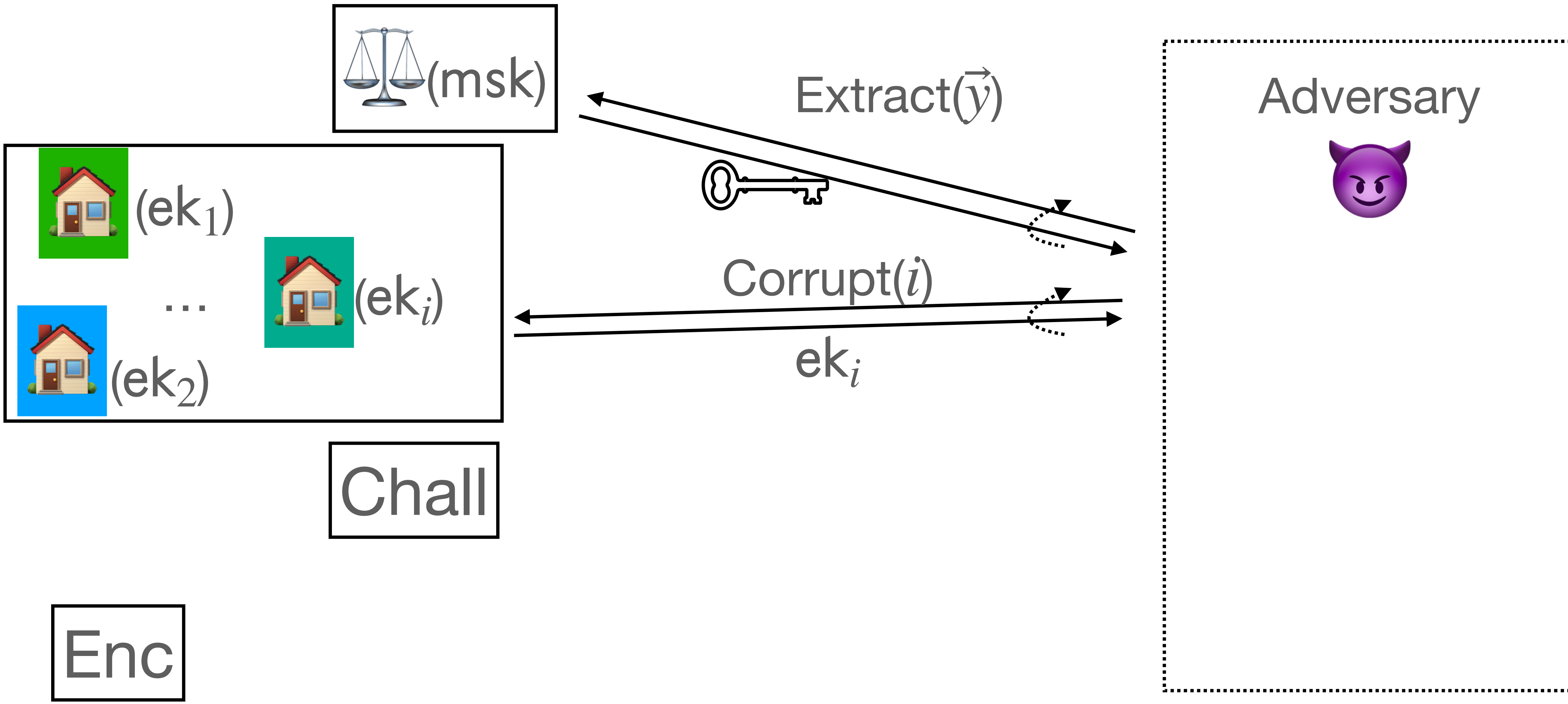
Challenger: $b \xleftarrow{\$} \{0,1\}$



Motivation - Security Model

How do we define a “secure” MCFE ?

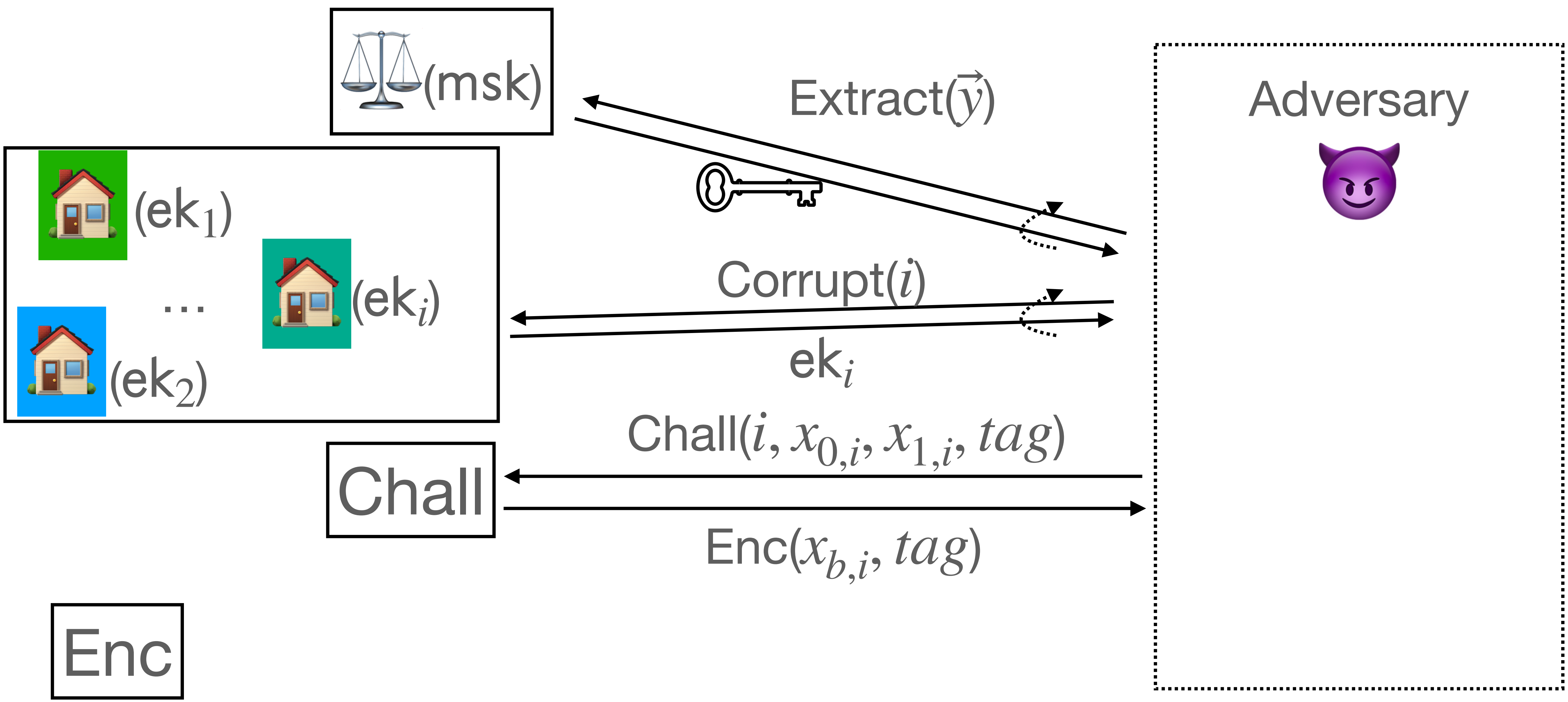
Challenger: $b \xleftarrow{\$} \{0,1\}$



Motivation - Security Model

How do we define a “secure” MCFE ?

Challenger: $b \xleftarrow{\$} \{0,1\}$

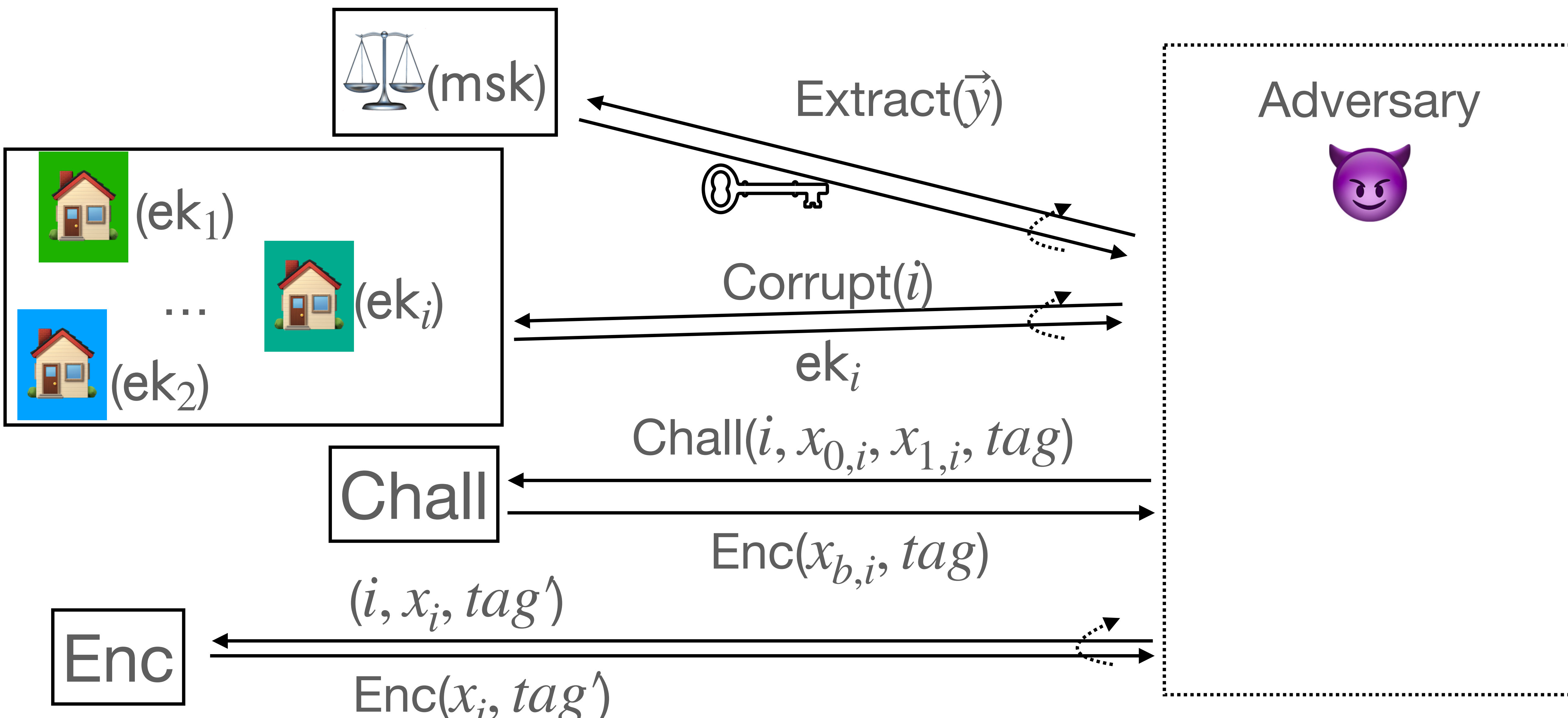


Secret Keys: $\boxed{\text{Enc}}$ +1 chall security ⇒ multi-chall security

Motivation - Security Model

How do we define a “secure” MCFE ?

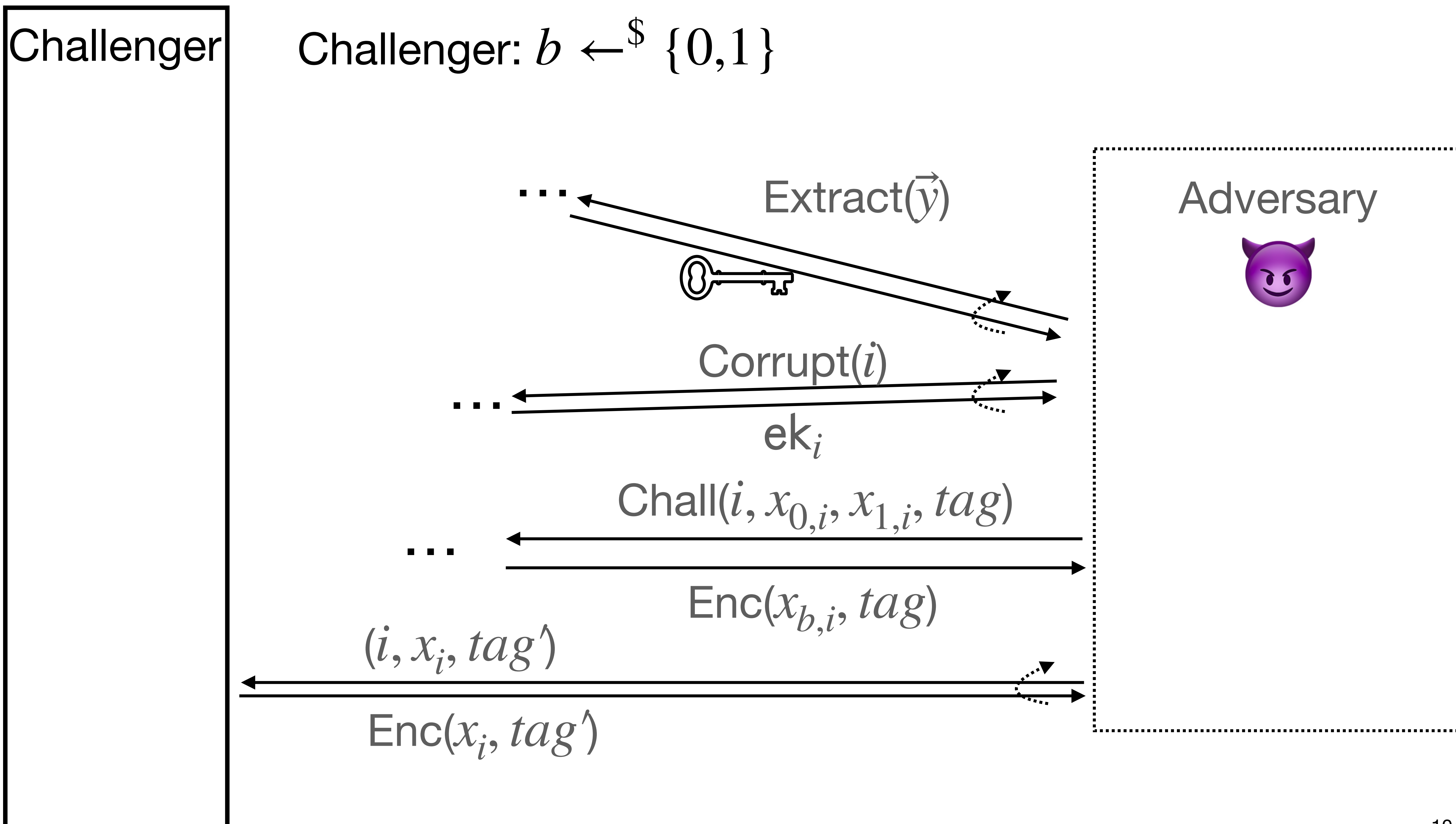
Challenger: $b \xleftarrow{\$} \{0,1\}$



Secret Keys: $\boxed{\text{Enc}}$ +1 chall security $\Rightarrow$ multi-chall security

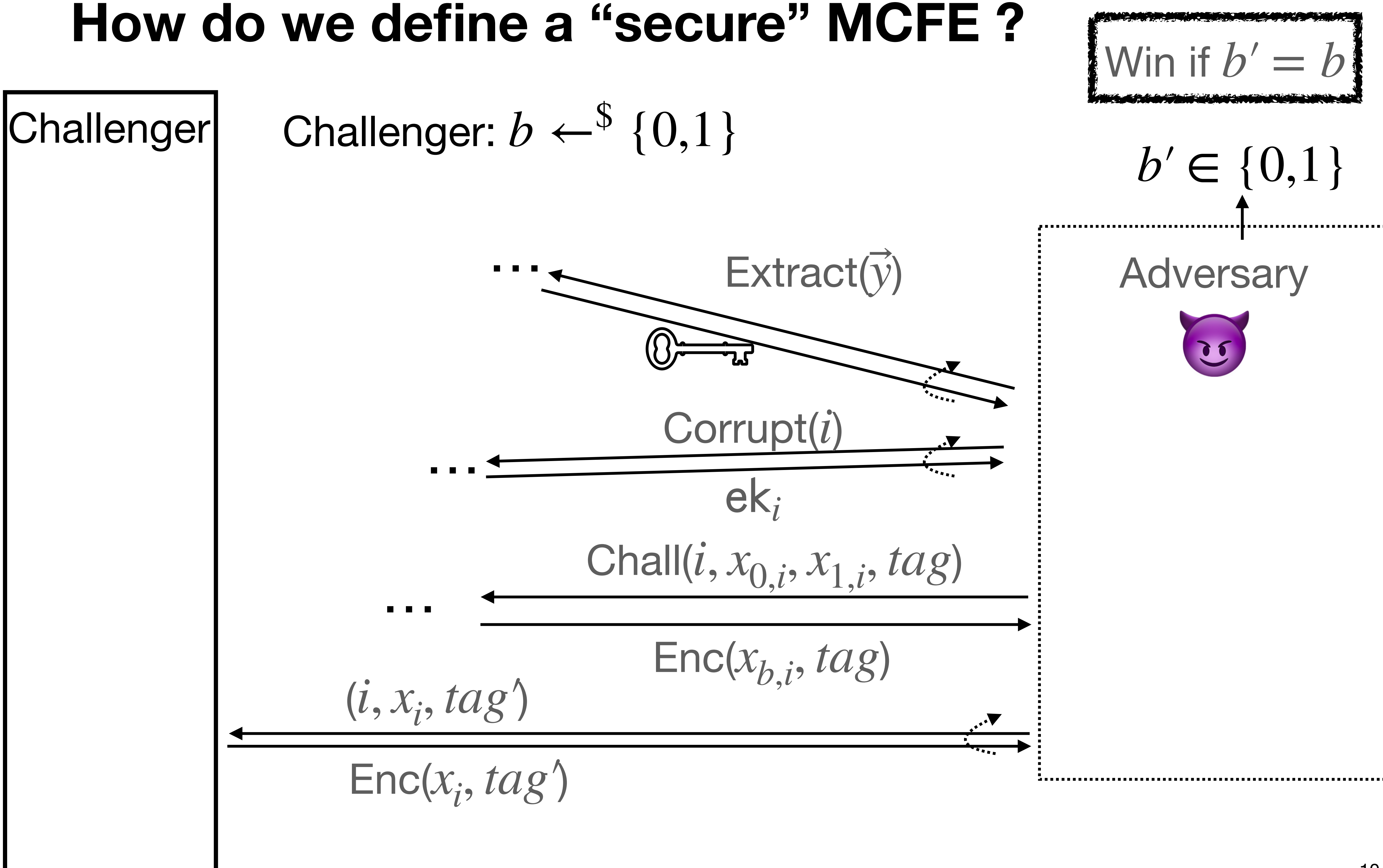
Motivation - Security Model

How do we define a “secure” MCFE ?



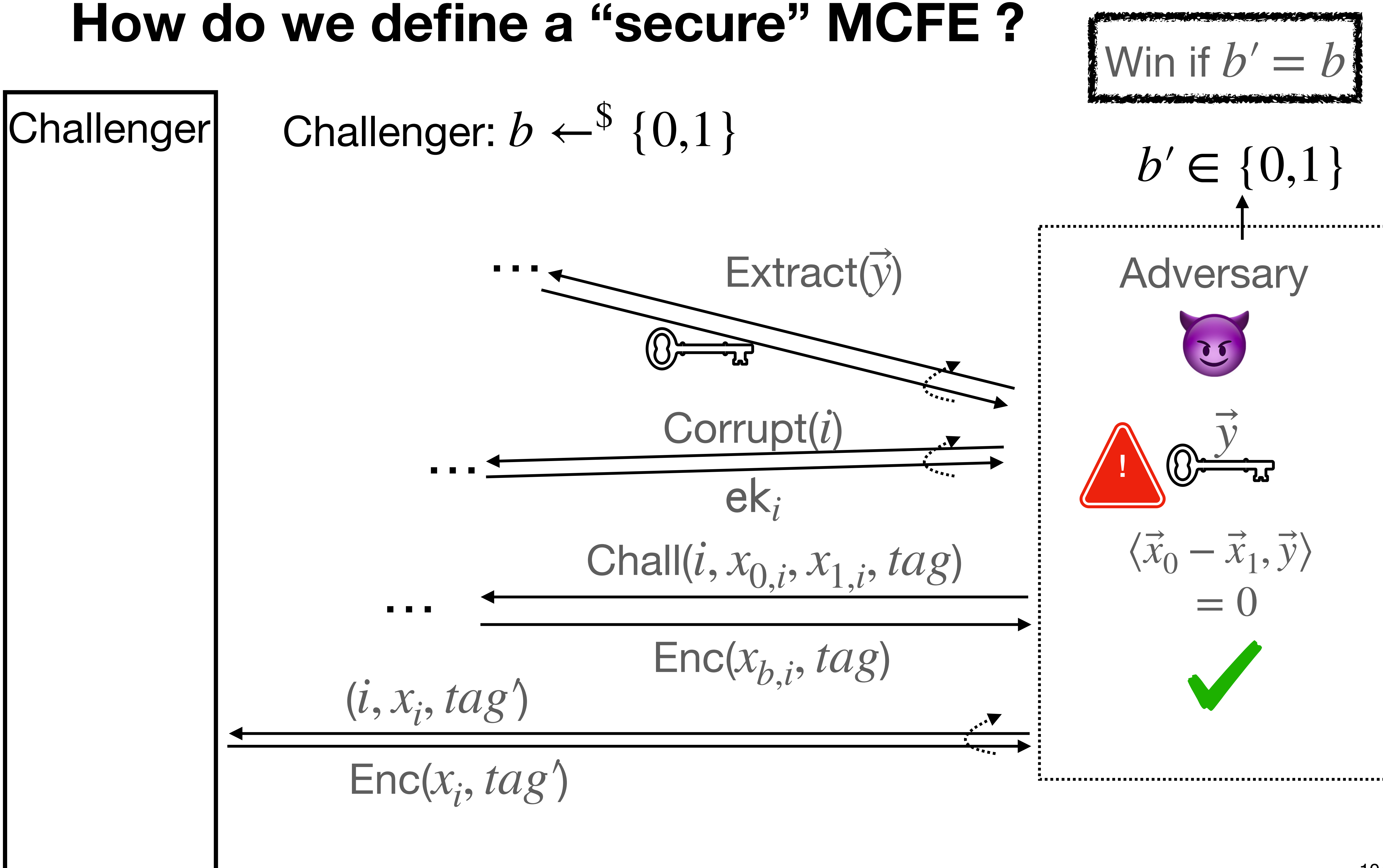
Motivation - Security Model

How do we define a “secure” MCFE ?



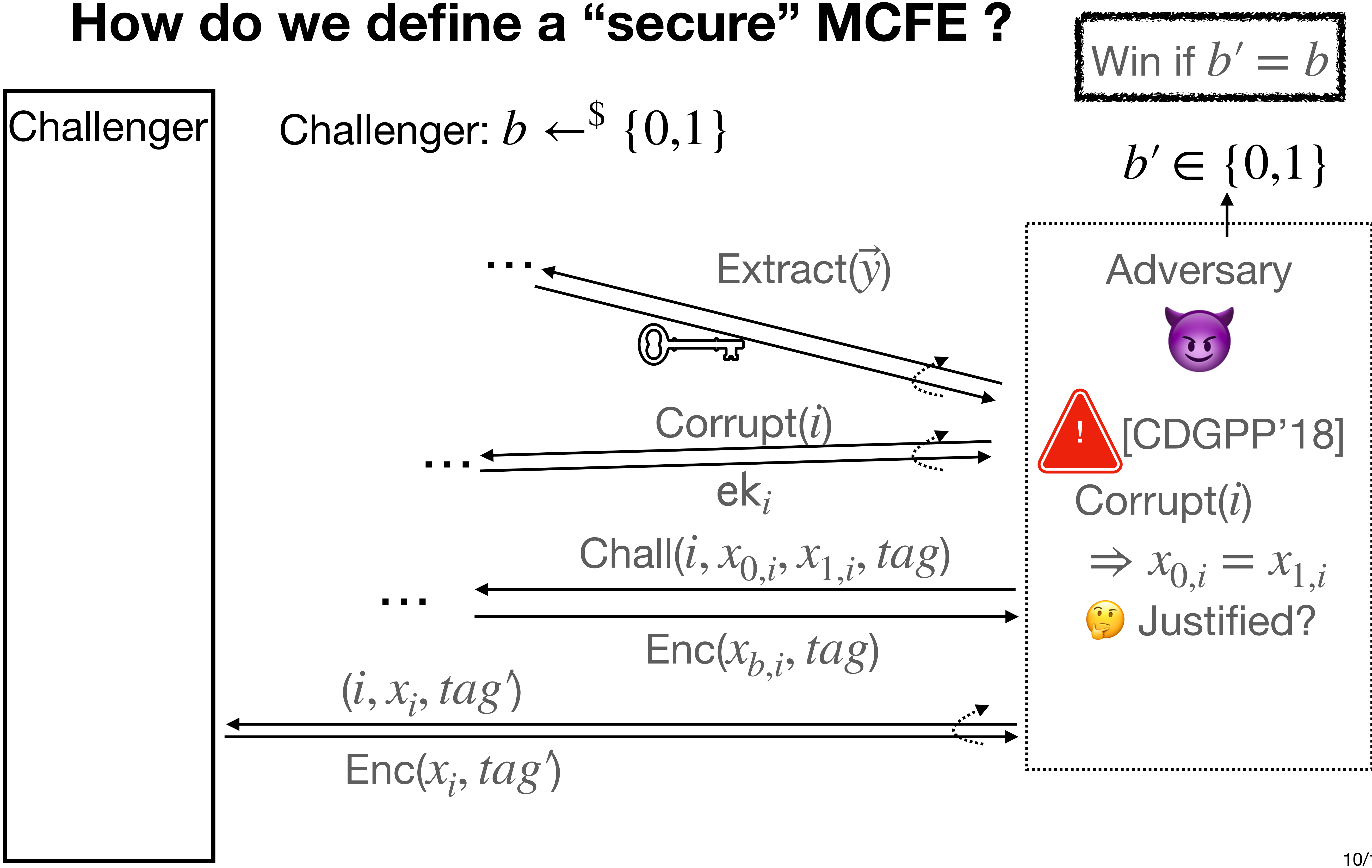
Motivation - Security Model

How do we define a “secure” MCFE ?



Motivation - Security Model

How do we define a “secure” MCFE ?



Motivation - Security Model

A“secure” MCFE - the case of Inner Products

Adversary 😈

⚠️ [CDGPP’18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

🤔 Justified?

Motivation - Security Model

A“secure” MCFE - the case of Inner Products

[CDGPP’18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

—> Follow-ups with same constraint: [ABKW19],[ABG19],[LT19], [CDG+20],[AGT21],[SV23]

Adversary 😈

⚠️ [CDGPP’18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

🤔 Justified?

Motivation - Security Model

A“secure” MCFE - the case of Inner Products

[CDGPP’18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

—> Follow-ups with same constraint: [ABKW19],[ABG19],[LT19], [CDG+20],[AGT21],[SV23]

Adversary 😈

⚠️ [CDGPP’18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

🤔 Justified?

From [CDGPP18]:

MCFE is Secret Encryption Keys

$\Rightarrow$ Usually deterministic

$\Rightarrow$ $\boxed{\text{Corrupt}(i) \Rightarrow x_{0,i} = x_{1,i}}$

Motivation - Security Model

A“secure” MCFE - the case of Inner Products

[CDGPP’18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

—> Follow-ups with same constraint: [ABKW19],[ABG19],[LT19], [CDG+20],[AGT21],[SV23]

Adversary

😈

⚠️ [CDGPP’18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

🤔 Justified?

Motivation - Security Model

A"secure" MCFE - the case of Inner Products

[CDGPP'18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

—> Follow-ups with same constraint: [ABKW19],[ABG19],[LT19], [CDG+20],[AGT21],[SV23]

Adversary 😈

⚠️ [CDGPP'18]

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

🤔 Justified?

✗

🤔Modeling Problem:

Why can't we do better?

(*I.e.*, weaker condition

$\Rightarrow$ considering stronger attacker

$\Rightarrow$ stronger security guarantee)

Related Works

Improved Security Notion for (D)MCFE

Optimal Security Notion for Decentralized Multi-Client Functional Encryption [NPP'23]

Computing scalar inner products:

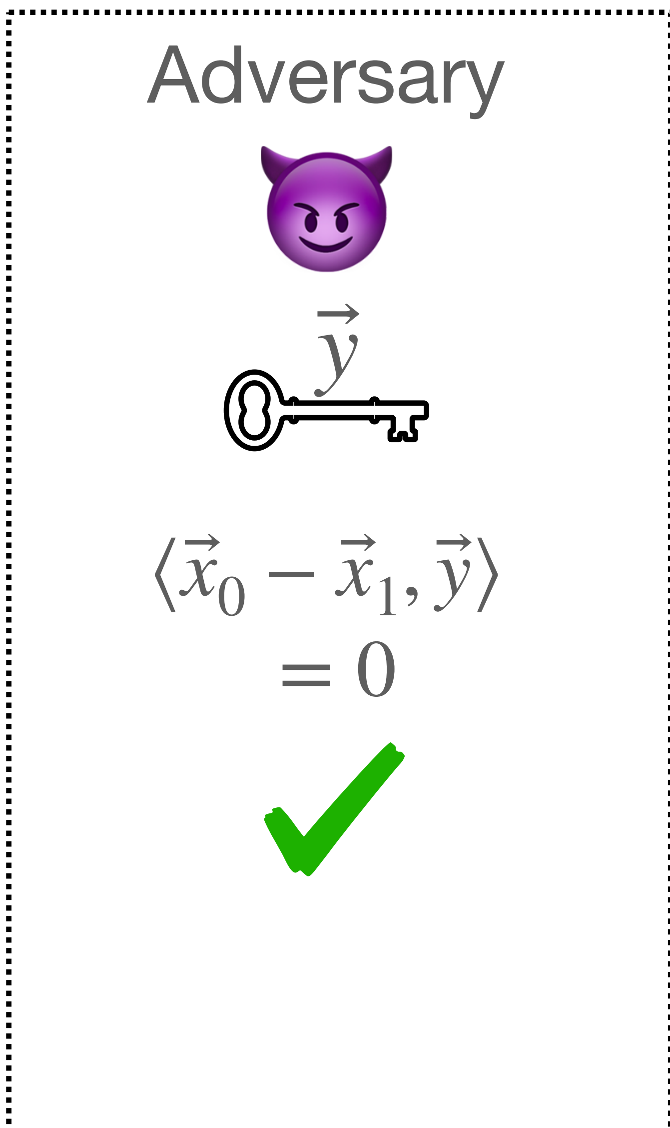
F(·) =

$$\langle \cdot , param \rangle$$

Related Works

Improved Security Notion for (D)MCFE

Optimal Security Notion for Decentralized Multi-Client Functional Encryption [NPP'23]



Computing scalar inner products:

$F(\cdot) = \langle \cdot \, , \, param \rangle$

Related Works

Improved Security Notion for (D)MCFE

Optimal Security Notion for Decentralized Multi-Client Functional Encryption [NPP'23]

Adversary

😈

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

or $y_i = 0$

Computing scalar inner products:

F(·) =

$\langle \cdot , param \rangle$

Related Works

Improved Security Notion for (D)MCFE

Optimal Security Notion for Decentralized Multi-Client Functional Encryption [NPP'23]

Adversary 😈

Corrupt(i)

$\Rightarrow x_{0,i} = x_{1,i}$

or $y_i = 0$

+ Optimality proof : this is the *least* we require 🎉

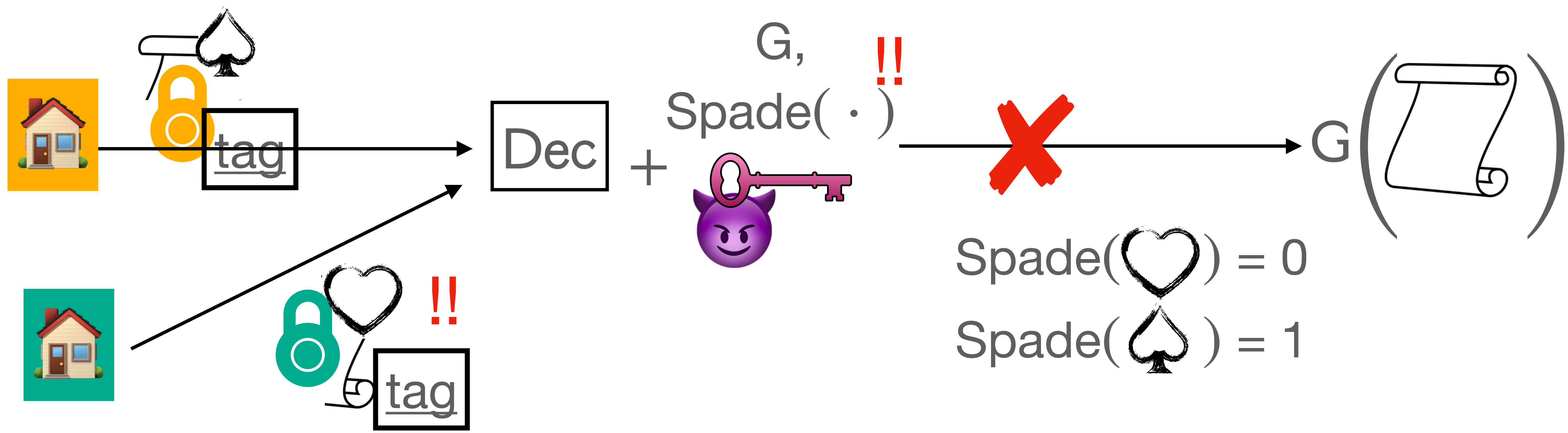
Computing scalar inner products:

F(·) =

$\langle \cdot , param \rangle$

Research Question - Combining All Together

Controlling Keys (*as per* [NPP22]) in the Optimal Setting (*as per* [NPP23])



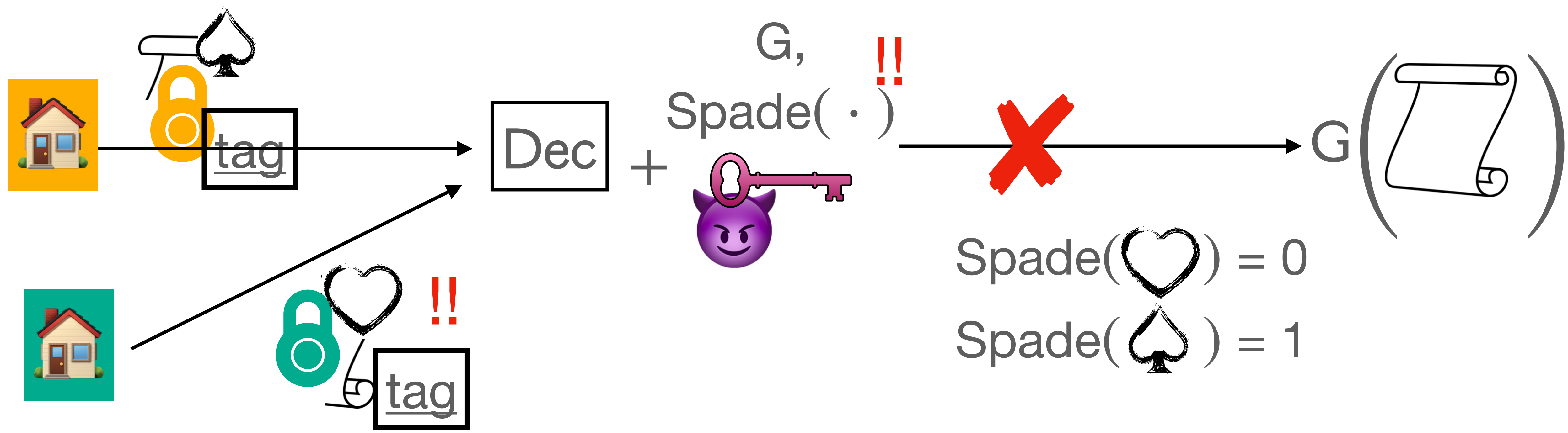
🤔 **public-key FE [BSW11], secret-key MCFE [GGG+14, CDG+18]**

Syntactical Problem:

NOT allow public-Att

Research Question - Combining All Together

Controlling Keys (*as per* [NPP22]) in the Optimal Setting (*as per* [NPP23])



Tag, ♤, ♡ : can be public

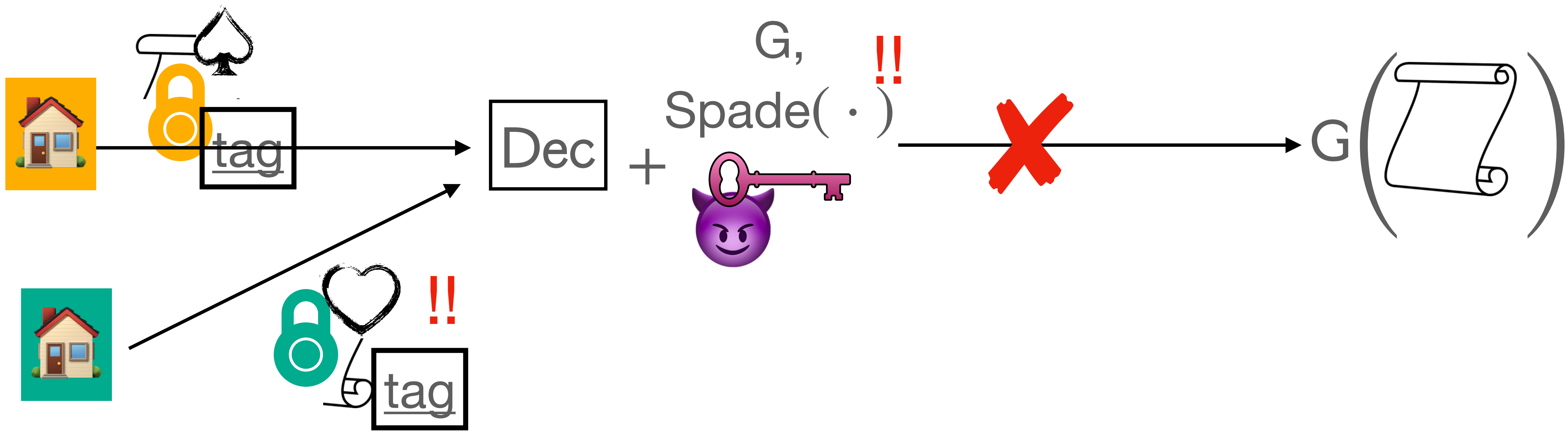
🤔 **<u>public-key FE [BSW11], secret-key MCFE [GGG+14, CDG+18]</u>**

<u>Syntactical Problem:</u>

NOT allow public-Att

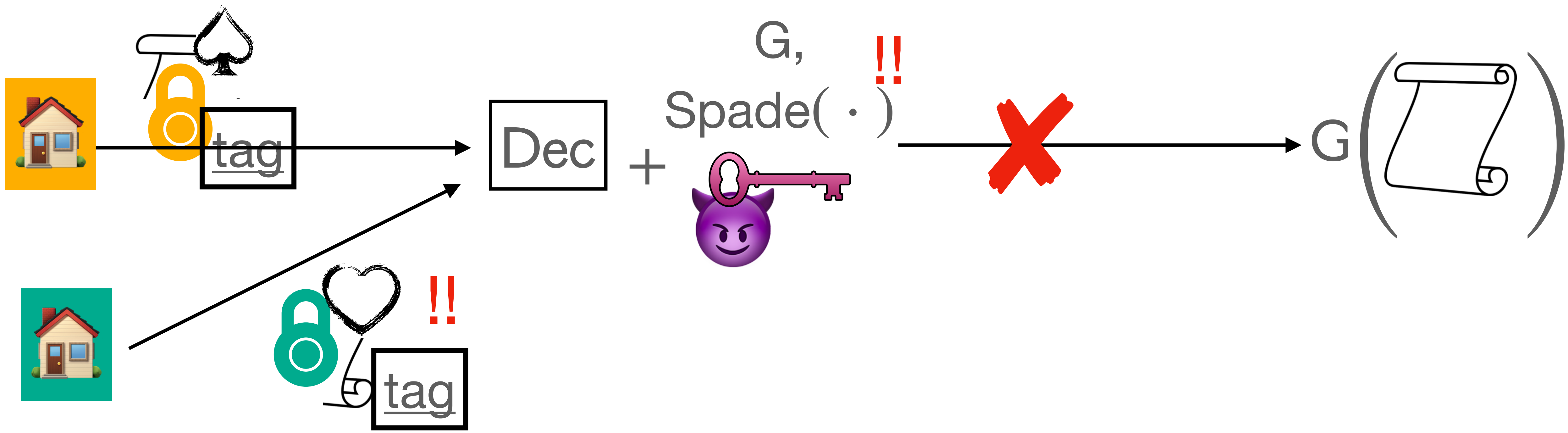
Our Result

Controlling Keys in the Optimal Setting



Our Result

Controlling Keys in the Optimal Setting

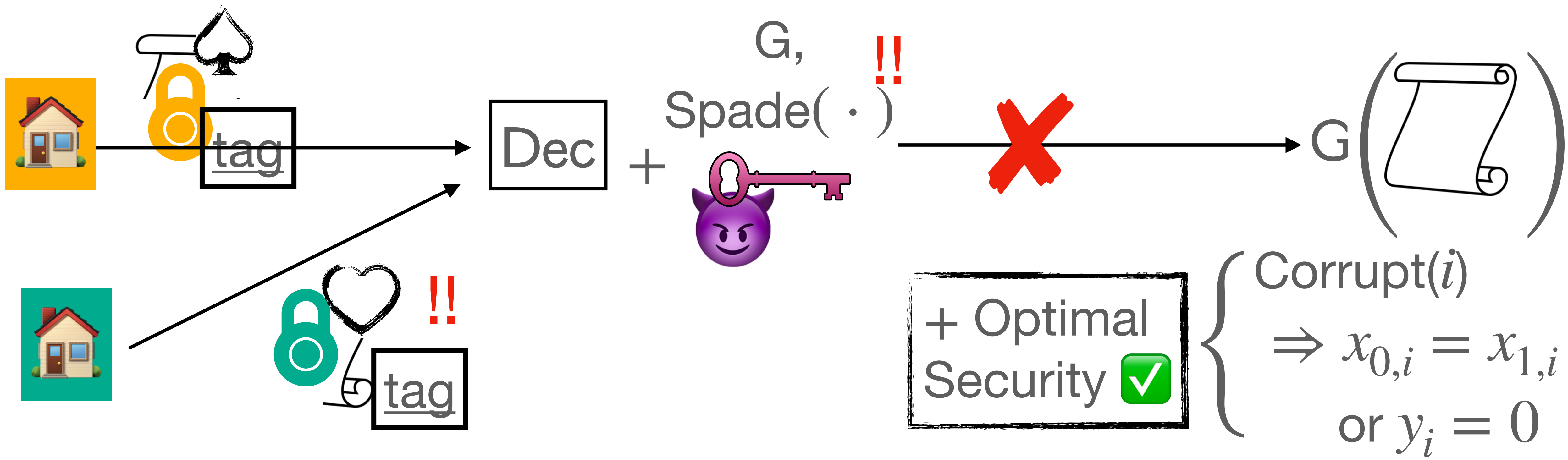


Refine Syntax of MCFE:

With public-inputs

Our Result

Controlling Keys in the Optimal Setting

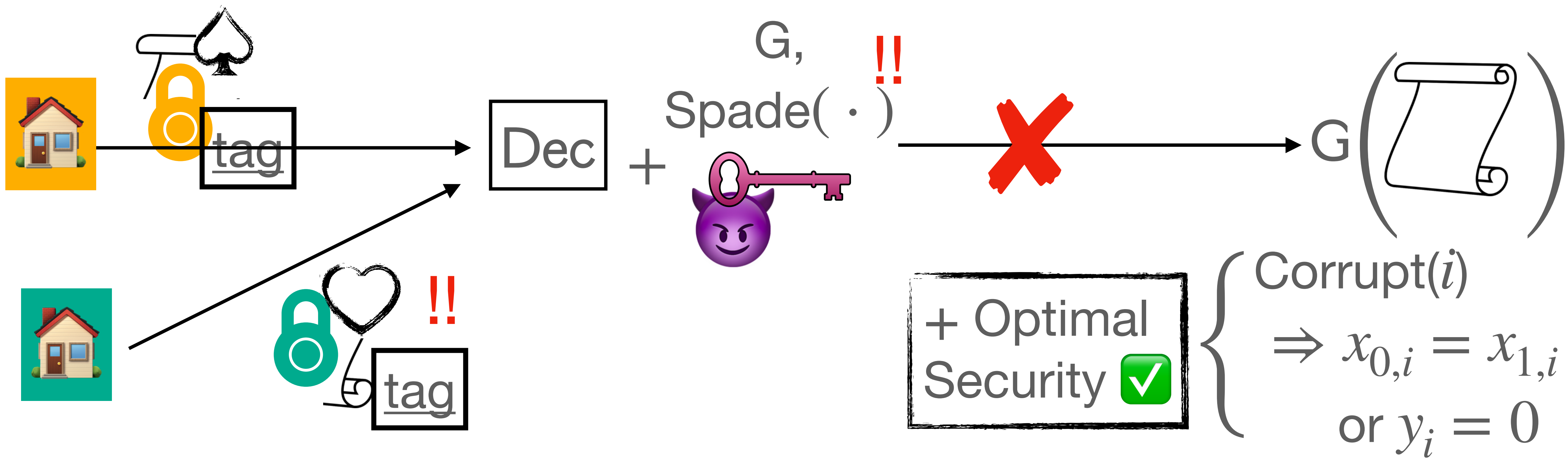


✔ **Refine Syntax of MCFE:**

With public-inputs

Our Result

Controlling Keys in the Optimal Setting



<u>Refine Syntax of MCFE:</u>

With public-inputs

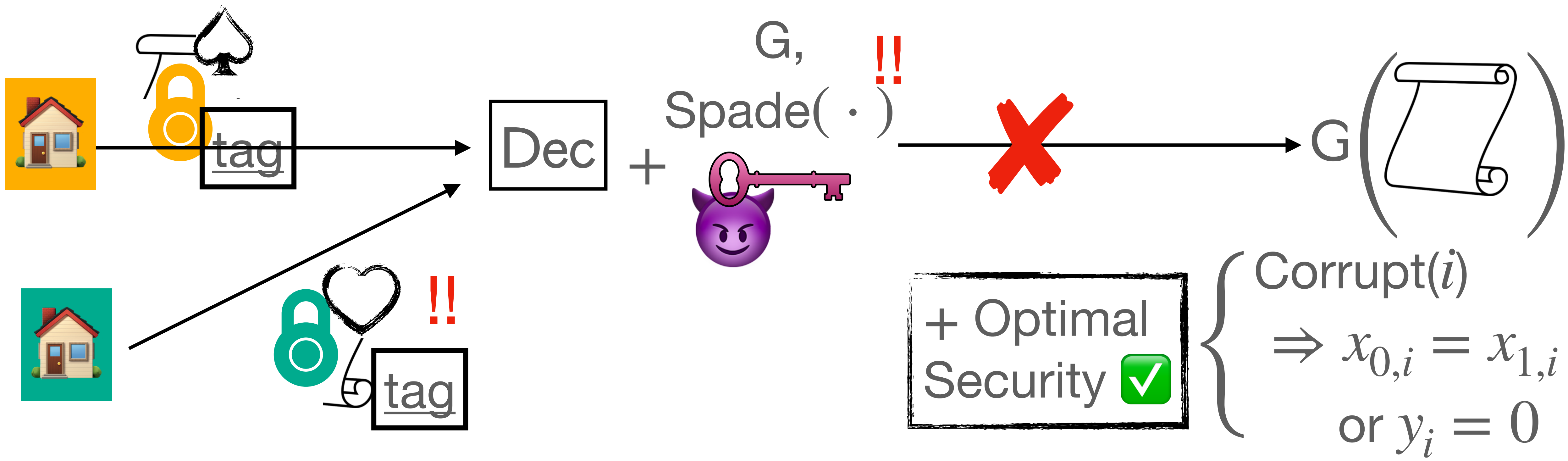
<u>Identify Implications:</u>

MCFE + optimal security

$\Rightarrow$ public-key FE

Our Result

Controlling Keys in the Optimal Setting



✓ **Refine Syntax of MCFE:**
With public-inputs

✓ **Identify Implications:**
MCFE + optimal security

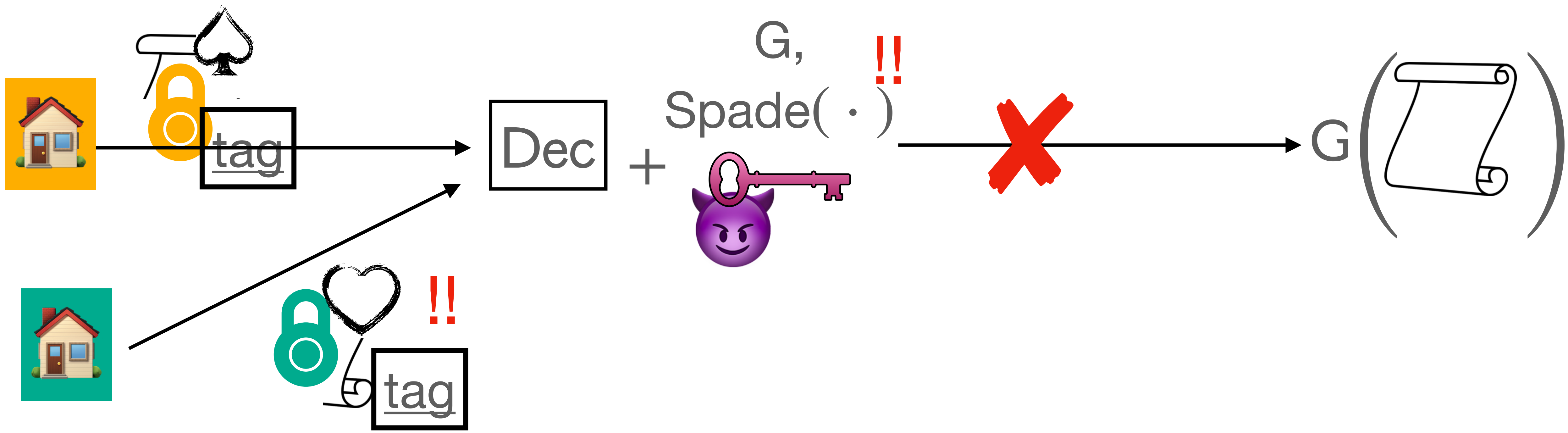
$\Rightarrow$ public-key FE

$\Longrightarrow$

Key-Policy
Attribute-based
Encryption,
Public- or private-att 🎉

Our Result

Controlling Keys in the Optimal Setting



✓ **Refine Syntax of MCFE:**

With public-inputs

✓ **Identify Implications:**

MCFE + optimal security

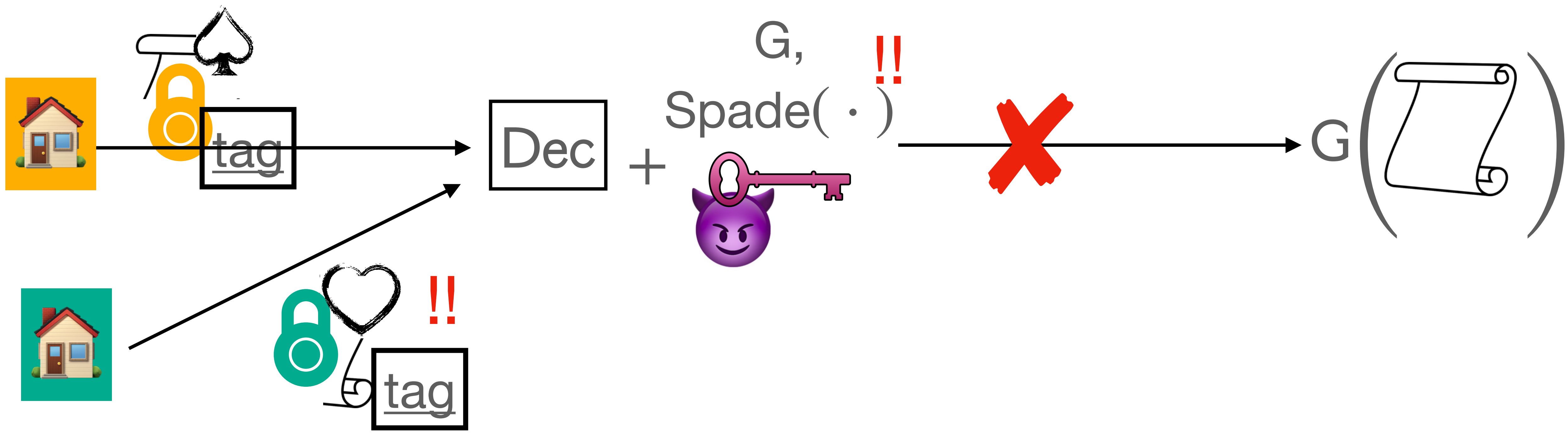
⇒ public-key FE

$\Longrightarrow$

Key-Policy
Attribute-based
Encryption,
Public- or private-att 🎉

Our Result

Controlling Keys in the Optimal Setting



Refine Syntax of MCFE:

With public-inputs

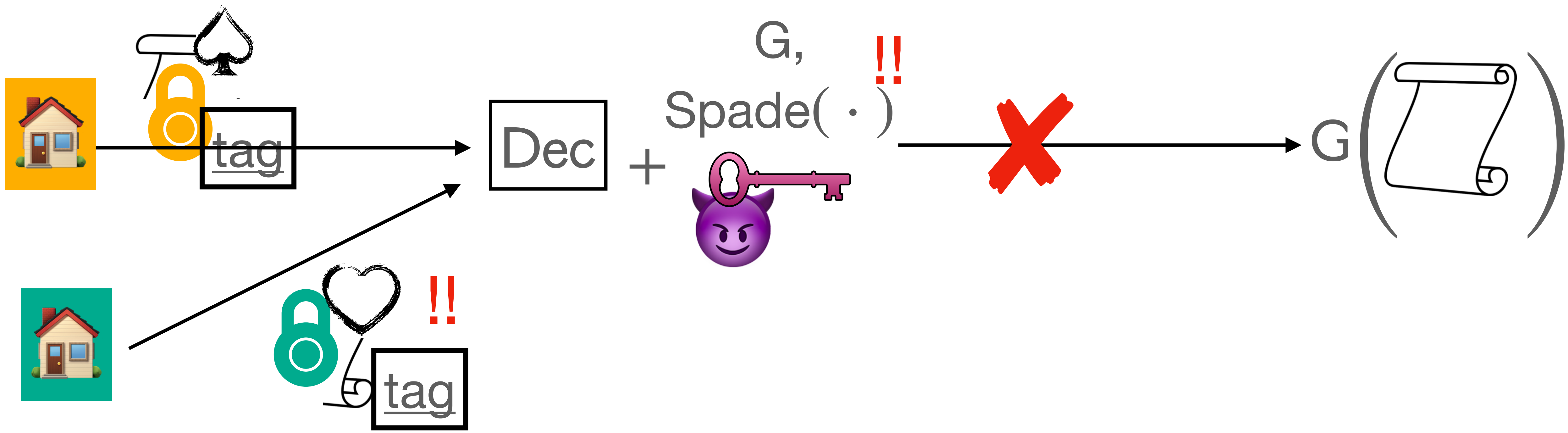
Identify Implications:

MCFE + optimal security

⇒ public-key FE

Our Result

Controlling Keys in the Optimal Setting



Refine Syntax of MCFE:
With public-inputs

Identify Implications:
MCFE + optimal security
⇒ public-key FE

Concret MCFE for Inner Products from Pairings in ROM, Stronger Security improving [NPP22,NPP23,ATY23]

Recall: The Admissibility from [NPP23]

Concrete conditions for IP, MCFE, one subvector/client

For all $(x_{0,i}, x_{1,i})_i$,

for all $\vec{y}$

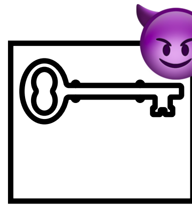
👼 honest i: $\boxed{x_{0,i}}$ $\boxed{x_{1,i}}$

😈 corrupted i: $x_{0,i}$ $x_{1,i}$

$$\vec{y} = \left(y_i \right)_i$$

(1) $$\sum_{\text{👼 honest } i} \left\langle \boxed{x_{0,i}}, y_i \right\rangle = \sum_{\text{👼 honest } i} \left\langle \boxed{x_{1,i}}, y_i \right\rangle$$

MCFE

(2)  😈(ek) $\left\langle x_{0,i}, y_i \right\rangle = \left\langle x_{1,i}, y_i \right\rangle$

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

🥳Our Work!

MCFE: Pub-Inputs,
Allowing repetitions $\vec{x}_i$

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

🥳Our Work!

MCFE: Pub-Inputs,
Allowing repetitions $\vec{x}_i$

#clients
$n = 1$

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

Our Work!

MCFE: Pub-Inputs,
Allowing repetitions $\vec{x}_i$

#clients
$n = 1$

Strong Admissiblity:

(ek) $\left\langle \vec{x}_0 , \vec{y} \right\rangle = \left\langle \vec{x}_1 , \vec{y} \right\rangle$

On Security: Admissiblity All Over Again

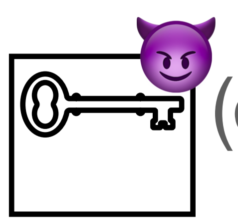
Implications of Pub-Inputs & Strong Admissibility

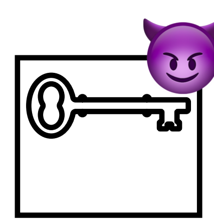
🥳Our Work!

MCFE: Pub-Inputs,
Allowing repetitions $\vec{x}_i$

#clients
$n = 1$

Strong Admissiblity:

(ek) $\left\langle \vec{x}_0 , \vec{y} \right\rangle = \left\langle \vec{x}_1 , \vec{y} \right\rangle$

+ Static Corruption (ek)

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

🥳Our Work!

MCFE: Pub-Inputs,
Allowing repetitions $\vec{x}_i$

#clients $n = 1$

Strong Admissiblity:

Admissibility in Public-key FE !!

(ek) $\left\langle \vec{x}_0, \vec{y} \right\rangle = \left\langle \vec{x}_1, \vec{y} \right\rangle$

+ Static Corruption (ek)

Becomes pk !!

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

🥳Our Work!

MCFE: Pub-Inputs, Allowing repetitions $\vec{x}_i$ $\Rightarrow$ Public-key FE With Pub-Inputs

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

🥳Our Work!

MCFE: Pub-Inputs,
Allowing repetitions $\vec{x}_i$

$\Rightarrow$

Public-key FE
With Pub-Inputs

Access control in
Public inputs

Covering syntactically
Non-pol/att-hiding
ABE !!

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

🥳Our Work!

MCFE: Pub-Inputs, Allowing repetitions $\vec{x}_i$ $\Rightarrow$ Public-key FE With Pub-Inputs

$\Downarrow$

Public-key ABE With or w/out Hiding properties

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

🥳Our Work!

<u>MCFE</u>: Pub-Inputs,
Allowing repetitions $\vec{x}_i$

$\Rightarrow$

<u>Public-key FE</u>
With Pub-Inputs

Fix one tag for all Enc

$\Downarrow$

<u>Public-key ABE</u>
With or w/out
Hiding properties

Repetitions on $\vec{x}_i$
Covers basic MIFE

On Security: Admissiblity All Over Again

Implications of Pub-Inputs & Strong Admissibility

🥳Our Work!

<u>MCFE</u>: Pub-Inputs, Allowing repetitions $\vec{x}_i$ $\Rightarrow$ <u>Public-key FE</u> With Pub-Inputs

$\Downarrow$

<u>MIFE:</u> "vanilla" Same functions on $\vec{x}_i$

$\Downarrow$

<u>Public-key ABE</u> With or w/out Hiding properties

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

In a nutshell: With Formal Changes, Guesses are Free

Selective

simulator : guess $(\vec{x}_i^{(0)}, \vec{x}_i^{(1)})$ from 😈

$$\mathrm{View}_0^{sel} \equiv \mathrm{View}_1^{sel} \equiv \ldots \equiv \underline{\mathrm{View}_k^{sel}}$$

$\leftarrow$

$\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

$\leftarrow$

$\cdots$

$\rightarrow$

Adversary 😈

Start $\Rightarrow$ Guessing Challenges $\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

In a nutshell: With Formal Changes, Guesses are Free

Selective

simulator : guess $(\vec{x}_i^{(0)}, \vec{x}_i^{(1)})$ from 😈

$\text{View}_0^{sel} \equiv \text{View}_1^{sel} \equiv \ldots \equiv \underline{\text{View}_k^{sel}}$

In $\underline{\text{View}_k^{sel}}$: Some critical step in the proof

$\longleftarrow$

$\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

$\longleftarrow$

…

$\longrightarrow$

Adversary

😈

Start $\Rightarrow$ Guessing Challenges $\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

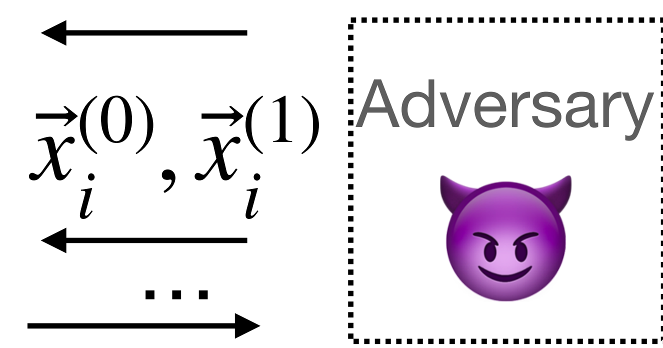
In a nutshell: **With Formal Changes, Guesses are Free**

Selective

simulator : guess $(\vec{x}_i^{(0)}, \vec{x}_i^{(1)})$ from 😈

$$\text{View}_0^{sel} \equiv \text{View}_1^{sel} \equiv \ldots \equiv \underline{\text{View}_k^{sel}}$$

In $\underline{\text{View}_k^{sel}}$: Some critical step in the proof



Start

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

<u>In a nutshell:</u> With Formal Changes, Guesses are Free

All is <u>formal</u> basis changes in *Dual Pairing Vector Spaces*

<u>Selective</u>

simulator : guess $(\vec{x}_i^{(0)}, \vec{x}_i^{(1)})$ from 😈

$$\mathrm{View}_0^{sel} \equiv \mathrm{View}_1^{sel} \equiv \ldots \equiv \underline{\mathrm{View}_k^{sel}}$$

Perfectly Perfectly Perfectly

In $\underline{\mathrm{View}_k^{sel}}$: Some critical step in the proof

$\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

…

Adversary

😈

Start

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

<u>In a nutshell:</u> With Formal Changes, Guesses are Free

All is <u>formal</u> basis changes
in *Dual Pairing Vector Spaces*

<u>Selective</u>

simulator : guess $(\vec{x}_i^{(0)}, \vec{x}_i^{(1)})$ from 😈

$\text{View}_0^{sel} \equiv \text{View}_1^{sel} \equiv \ldots \equiv \underline{\text{View}_k^{sel}}$

Perfectly Perfectly Perfectly

In $\underline{\text{View}_k^{sel}}$: Some critical step in the proof

$\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

...

Adversary
😈

Start $\Rightarrow$ Loss factor over choices of $\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$ in prob of correct

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

In a nutshell: With Formal Changes, Guesses are Free

Selective

simulator : guess $(\vec{x}_i^{(0)}, \vec{x}_i^{(1)})$ from 😈

$\text{View}_0^{sel} \equiv \text{View}_1^{sel} \equiv \ldots \equiv \underline{\text{View}_k^{sel}}$

Perfectly Perfectly Perfectly

In $\underline{\text{View}_k^{sel}}$: Some critical step in the proof

$\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

…

Adversary

😈

Start $\Rightarrow$ Loss factor over choices of $\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$ in prob of correct

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

<u>In a nutshell:</u> With Formal Changes, Guesses are Free

$$0 \leq \mathit{diff}(\mathsf{View}_0^{sel}; \mathsf{View}_k^{sel}) \leq \sum_i \mathit{diff}(\mathsf{View}_i^{sel}; \mathsf{View}_{i+1}^{sel}) = 0$$

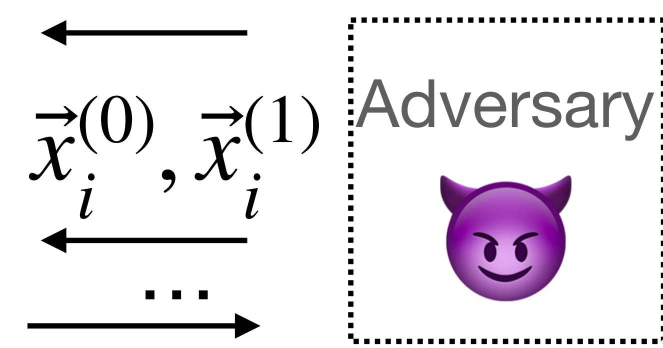
<u>Selective</u>

simulator : guess $(\vec{x}_i^{(0)}, \vec{x}_i^{(1)})$ from 😈

$$\mathsf{View}_0^{sel} \equiv \mathsf{View}_1^{sel} \equiv \ldots \equiv \underline{\mathsf{View}_k^{sel}}$$

Perfectly Perfectly Perfectly

In $\underline{\mathsf{View}_k^{sel}}$: Some critical step in the proof



$\boxed{\text{Start}} \Rightarrow$ $\boxed{\text{Loss factor over choices of } \vec{x}_i^{(0)}, \vec{x}_i^{(1)} \text{ in prob of correct}}$ $\times\, \mathit{diff}(\mathsf{View}_0; \mathsf{View}_k) = 0$

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

In a nutshell: With Formal Changes, Guesses are Free

Selective

simulator : guess $(\vec{x}_i^{(0)}, \vec{x}_i^{(1)})$ from 😈

$\text{View}_0^{sel} \equiv \text{View}_1^{sel} \equiv \ldots \equiv \underline{\text{View}_k^{sel}}$

Perfectly Perfectly Perfectly

In $\underline{\text{View}_k^{sel}}$: Some critical step in the proof

$\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

…

Adversary

😈

$\boxed{\text{Start}} \Rightarrow \boxed{\text{Loss factor over choices of } \vec{x}_i^{(0)}, \vec{x}_i^{(1)} \text{ in prob of correct}} \times \mathit{diff}(\text{View}_0; \text{View}_k) = 0$

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

In a nutshell: With Formal Changes, Guesses are Free

Adaptive

Simulator: no guesses

$$\text{View}_0^{sel} \equiv \text{View}_1^{sel} \equiv \ldots \equiv \underline{\text{View}_k^{sel}}$$

Perfectly Perfectly Perfectly

In $\underline{\text{View}_k^{sel}}$: Some critical step in the proof

$\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

...

Adversary 😈

$\boxed{\text{Start}} \Rightarrow$ $\boxed{\text{Loss factor over choices of } \vec{x}_i^{(0)}, \vec{x}_i^{(1)} \text{ in prob of correct}}$ $\times\, diff(\text{View}_0; \text{View}_k) = 0$

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

In a nutshell: With Formal Changes, Guesses are Free

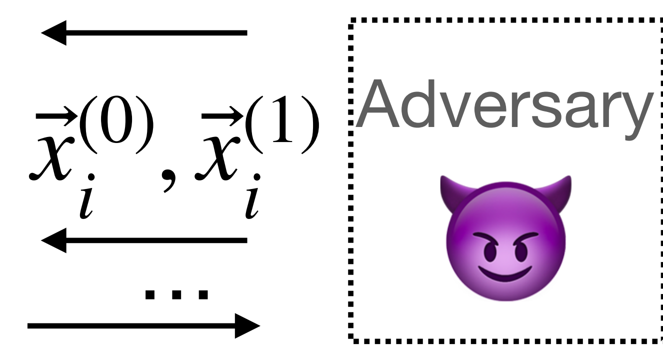
Adaptive

Simulator: no guesses

$$\text{View}_0^{sel} \equiv \text{View}_1^{sel} \equiv \ldots \equiv \underline{\text{View}_k^{sel}}$$

Perfectly Perfectly Perfectly

In $\underline{\text{View}_k^{sel}}$: Some critical step in the proof



Start

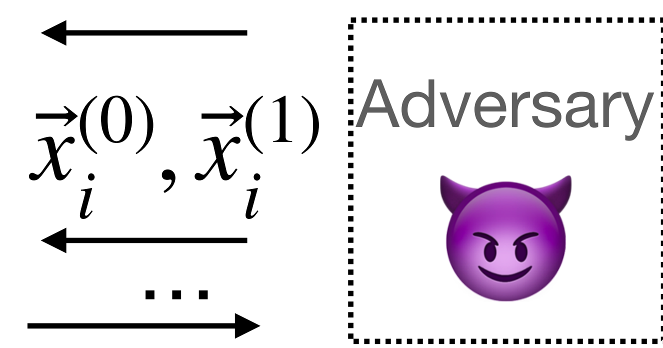
New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

In a nutshell: With Formal Changes, Guesses are Free

Adaptive

Simulator: no guesses

Perfectly Perfectly Perfectly



Start

New Techniques: Dual Paring Vector Spaces with Complexity Leveraging

In a nutshell: With Formal Changes, Guesses are Free

Adaptive

Simulator: no guesses

$$\text{View}_0 \equiv \text{View}_1 \equiv \ldots \equiv \underline{\text{View}_k}$$

Perfectly Perfectly Perfectly

In $\underline{\text{View}_k}$: Some critical step in the proof

$\vec{x}_i^{(0)}, \vec{x}_i^{(1)}$

…

Adversary

😈

Conclusion

Our Results:

- Definitional framework of MCFE with public inputs
- Implications from *secret-key* MCFE to *public-key* FE/KP-ABE
- Constructions with sub-vectors *per* client and strong security

Conclusion

Our Results:

- Definitional framework of MCFE with public inputs
- Implications from *secret-key* MCFE to *public-key* FE/KP-ABE
- Constructions with sub-vectors *per* client and strong security

Improving [NPP22,NPP23]

Conclusion

Our Results:

- Definitional framework of MCFE with public inputs
- Implications from *secret-key* MCFE to *public-key* FE/KP-ABE
- Constructions with sub-vectors *per* client and strong security

Improving [NPP22,NPP23]

Comparison to security of
Recent works such as [ATY23]

Conclusion

Our Results:

- Definitional framework of MCFE with public inputs
- Implications from *secret-key* MCFE to *public-key* FE/KP-ABE
- Constructions with sub-vectors *per* client and strong security

Improving [NPP22,NPP23]

Comparison to security of
Recent works such as [ATY23]

1. Stronger Computation? E.g.

Unbounded inner products.

2. Generalized primitives? E.g.

Dynamic and/or decentralized.

3. Lattice-based constructions?

Conclusion

Our Results:

- Definitional framework of MCFE with public inputs
- Implications from *secret-key* MCFE to *public-key* FE/KP-ABE
- Constructions with sub-vectors *per* client and strong security

Improving [NPP22,NPP23]

Comparison to security of
Recent works such as [ATY23]

1. *Stronger Computation? E.g.*

   Unbounded inner products.

2. *Generalized primitives? E.g.*

   Dynamic and/or decentralized.

3. *Lattice-based constructions?*