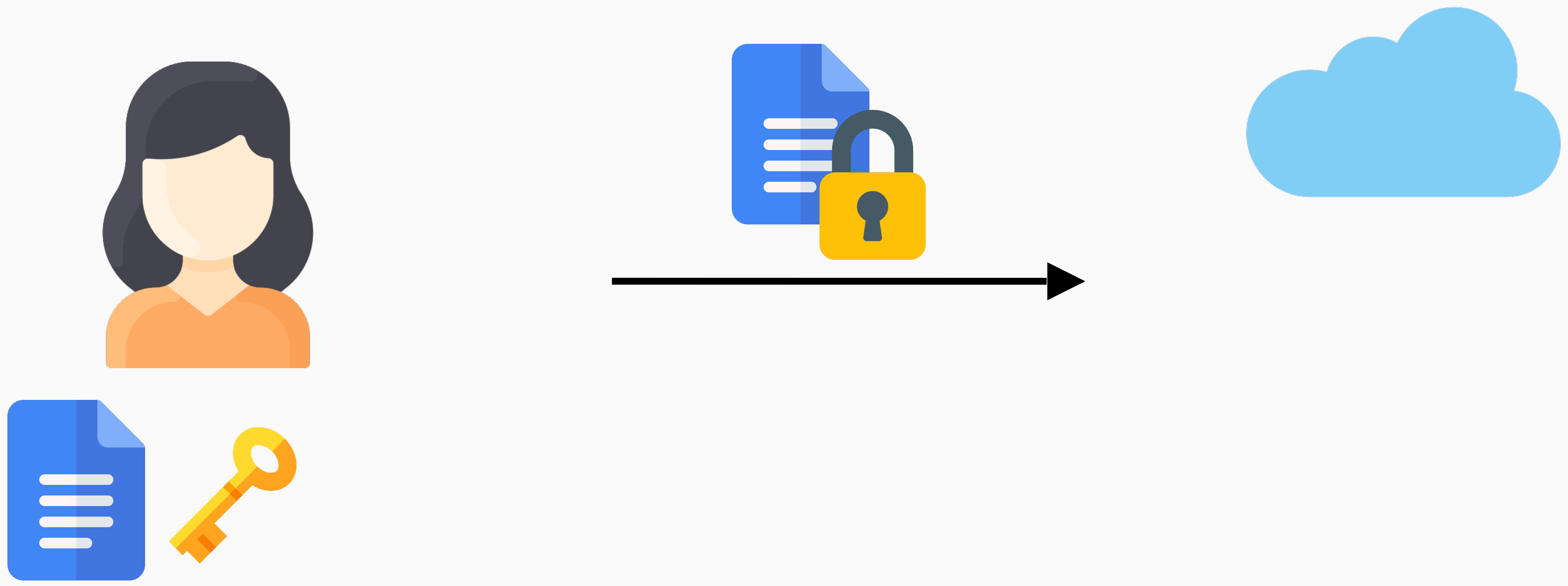
CCA Secure Updatable Encryption from Non-Mappable Group Actions

Jonas Meers, Doreen Riepel

June 13, 2024

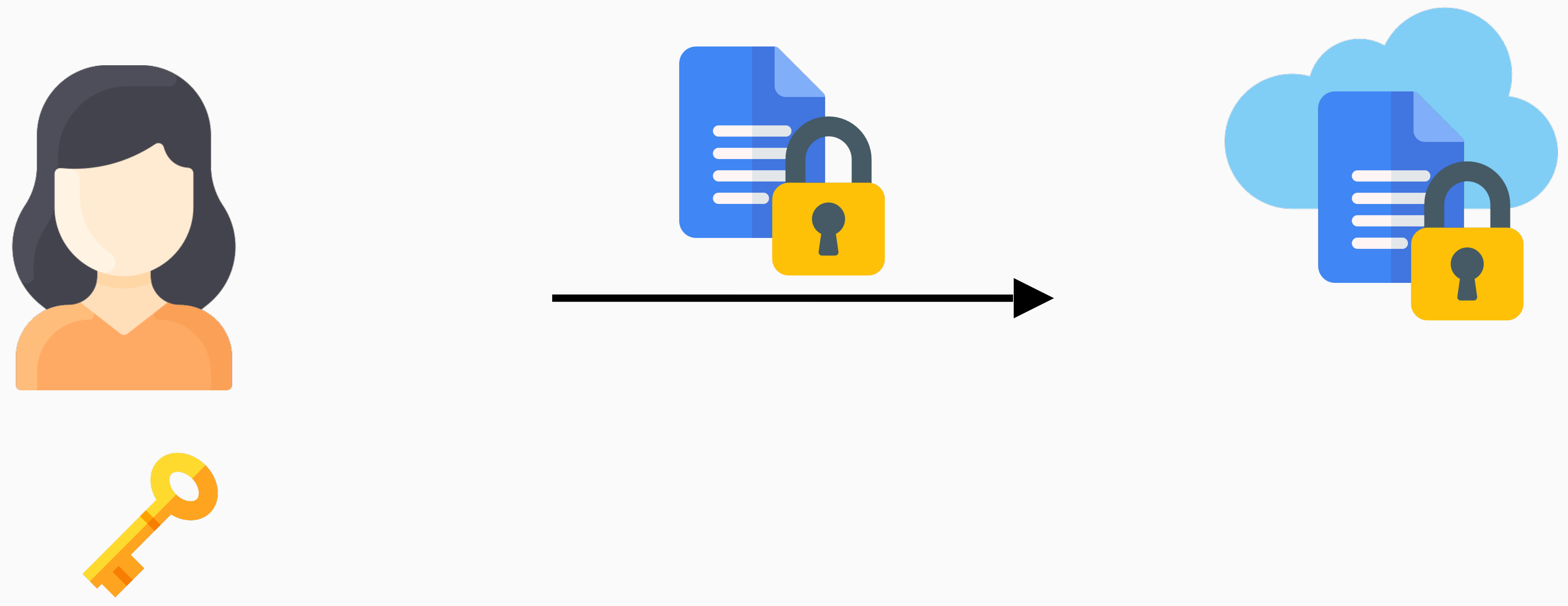
Motivation

Cloud Storage



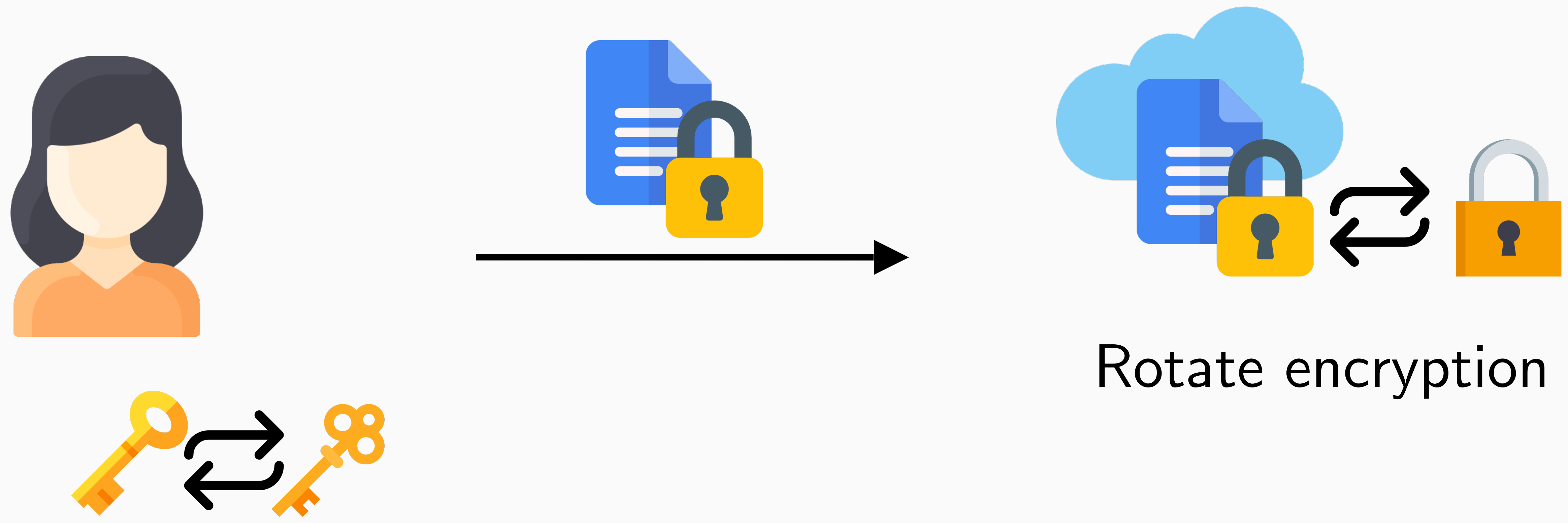
Motivation

Cloud Storage



Motivation

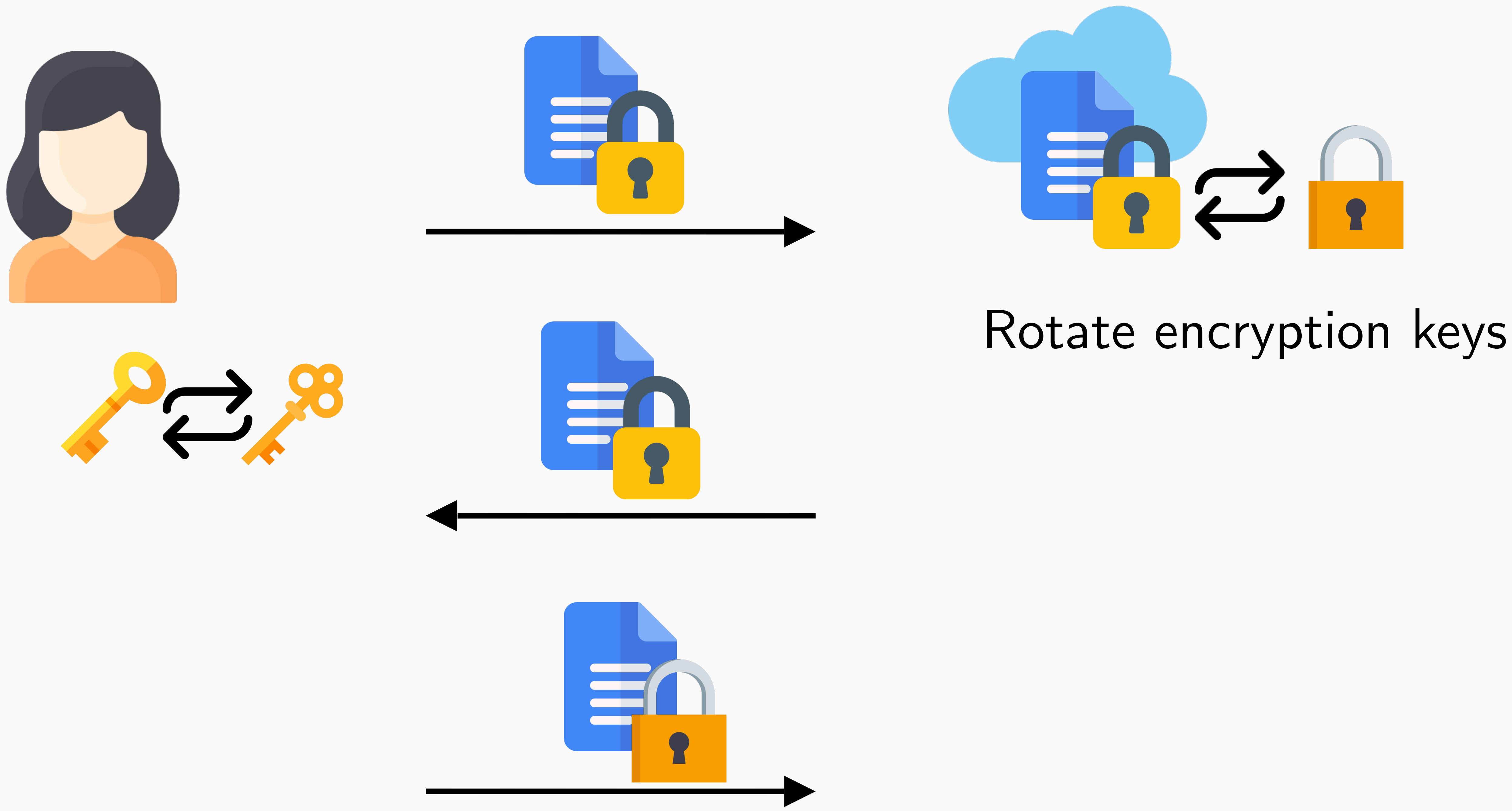
Cloud Storage



Rotate encryption keys

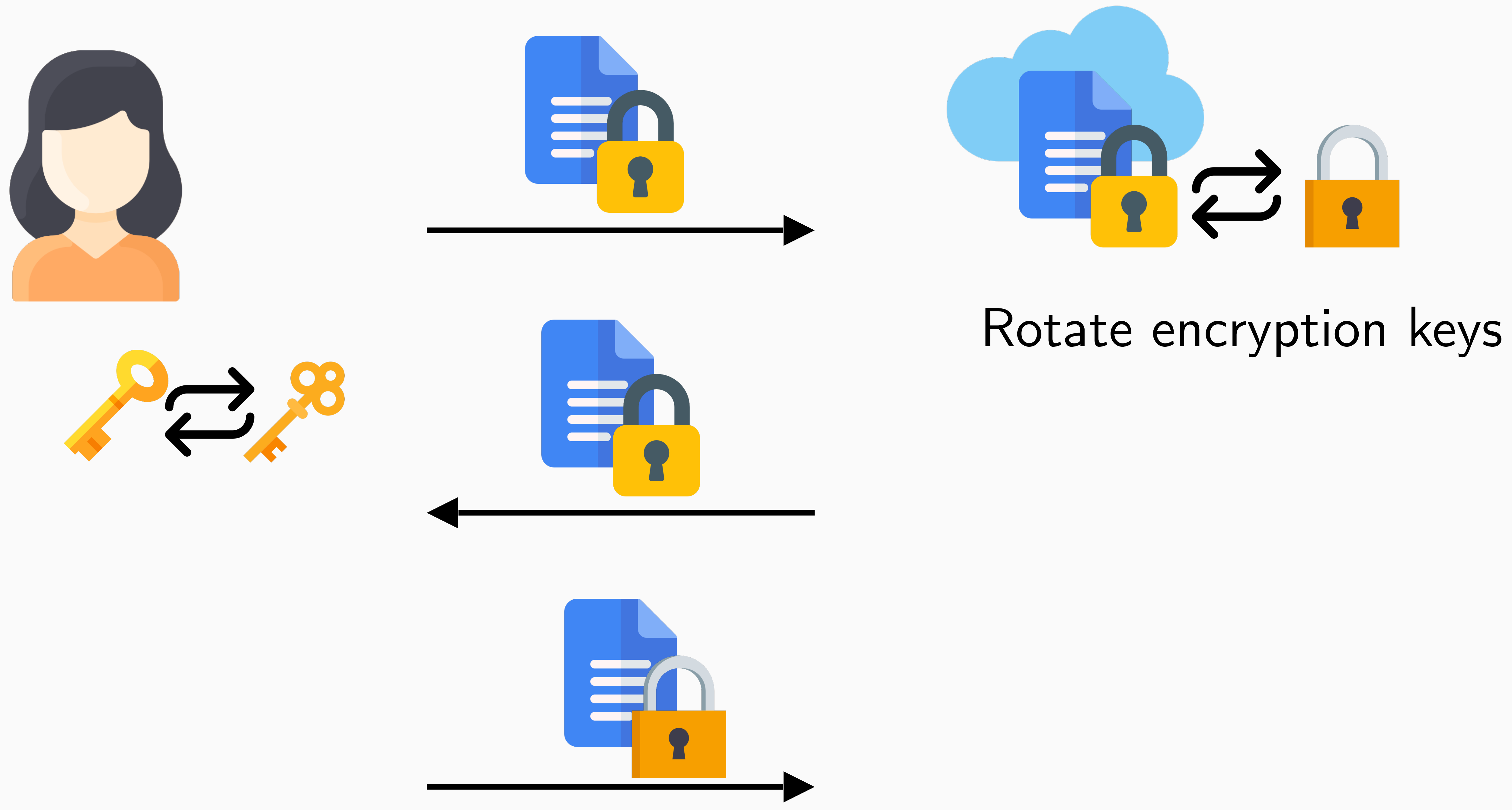
Motivation

Cloud Storage



Motivation

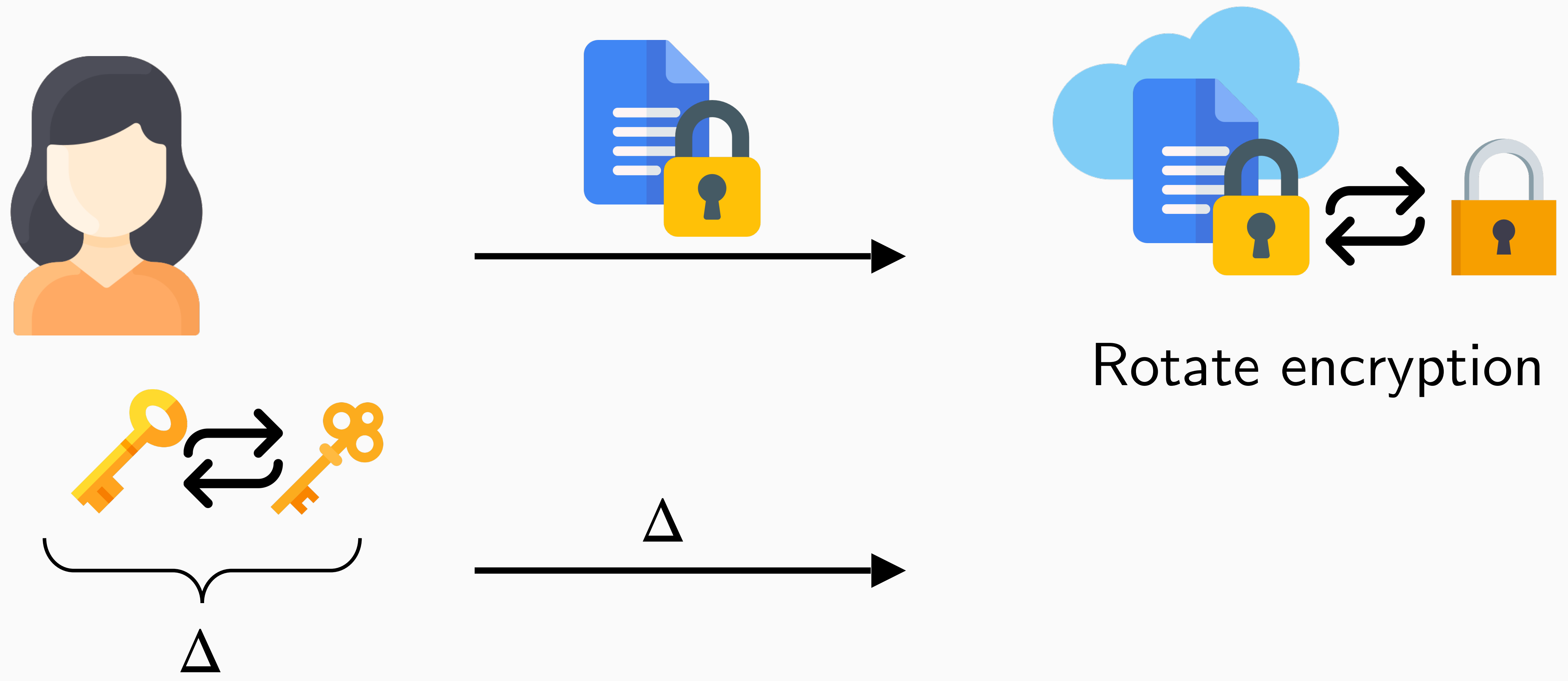
Cloud Storage



Inefficient!

Motivation

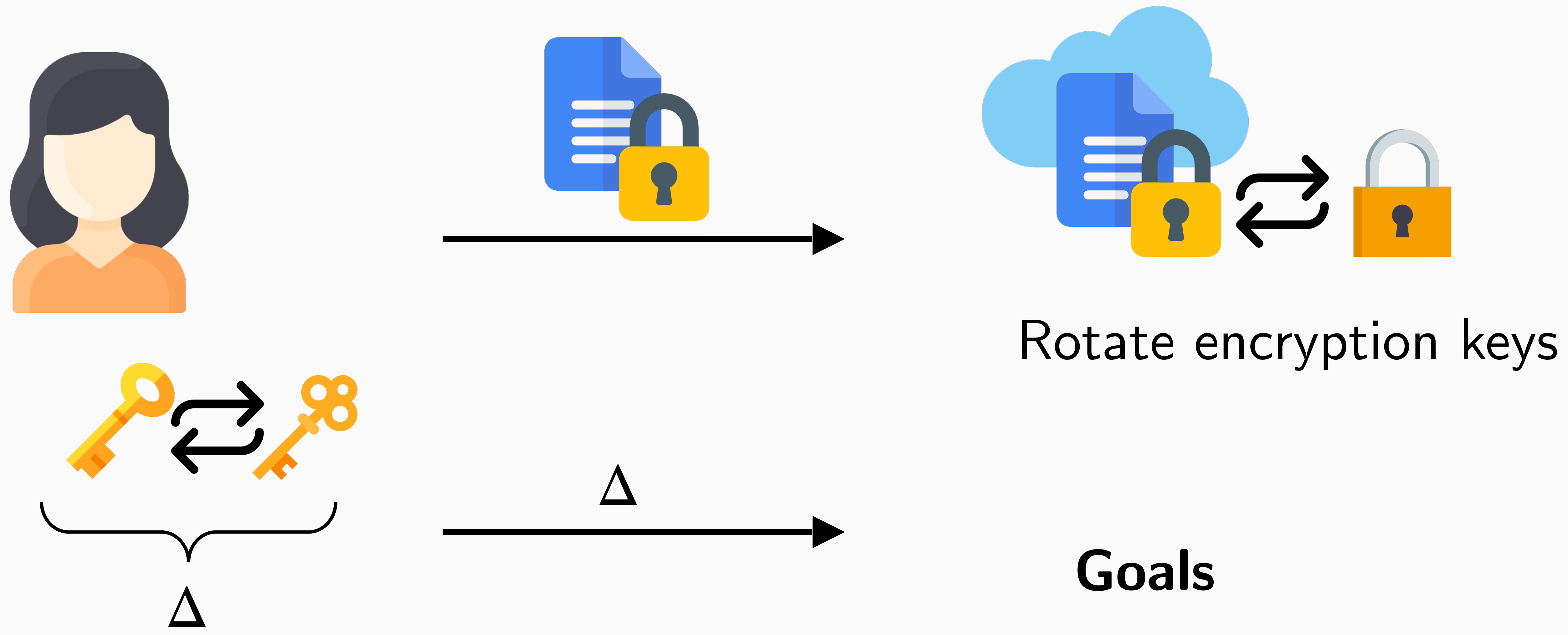
Cloud Storage



Rotate encryption keys

Motivation

Cloud Storage



Goals

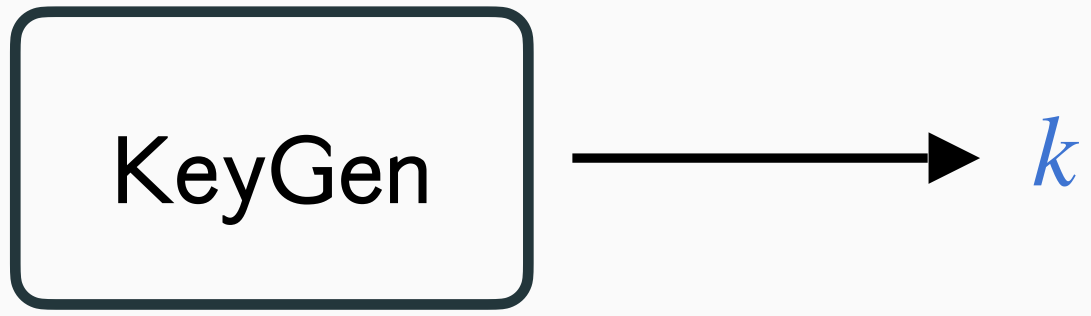
- **Confidentiality:** Cannot distinguish encryptions of two chosen messages
- **Integrity:** Cannot modify ciphertexts
- **Unlinkability:** Cannot tell which ciphertext an update was derived from
- **Forward secrecy:** Old ciphertext is secure even if current key leaks
- **Post-compromise security:** Old key does not help to decrypt updated ciphertext

Syntax

Updatable Encryption $\mathsf{UE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Upd})$

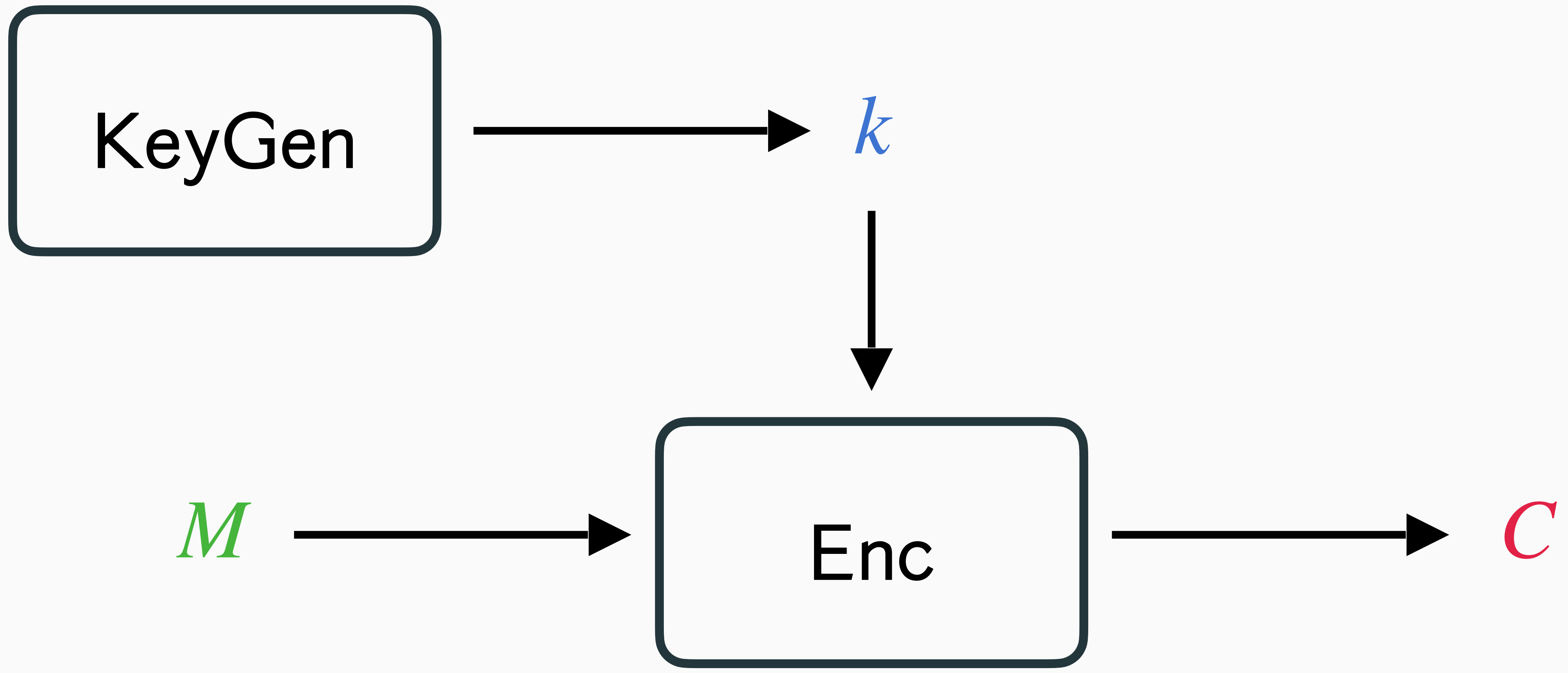
Syntax

Updatable Encryption $\mathsf{UE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Upd})$



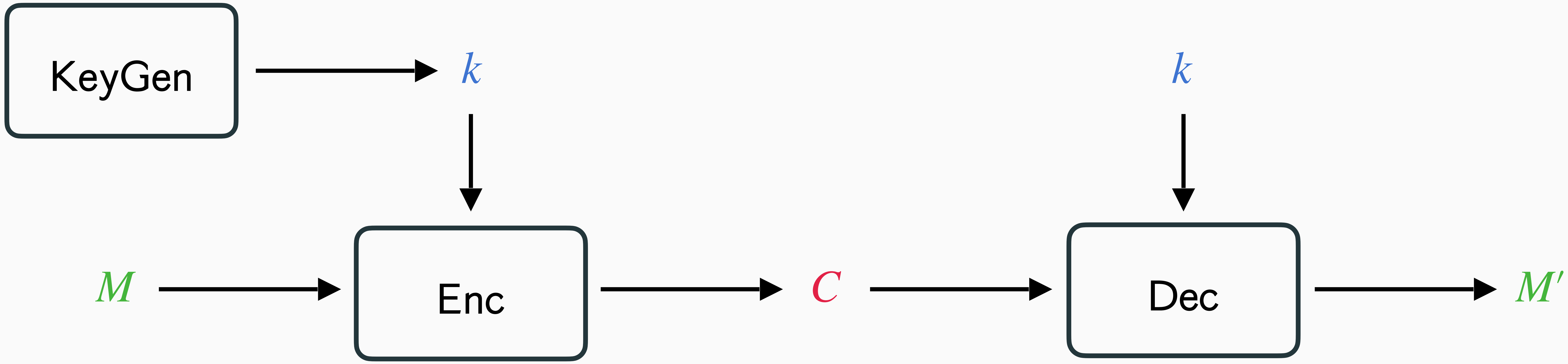
Syntax

Updatable Encryption $\mathsf{UE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Upd})$



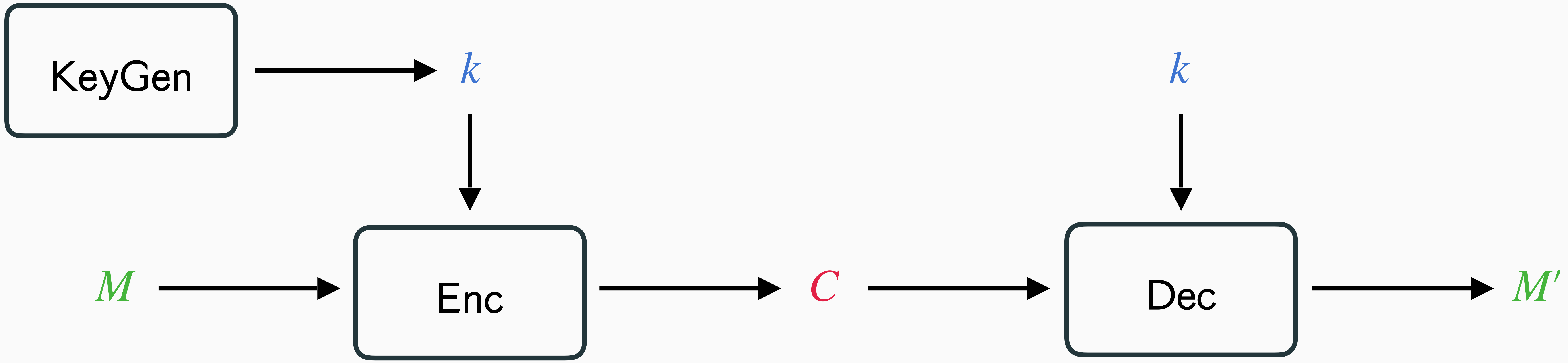
Syntax

Updatable Encryption $\mathsf{UE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Upd})$



Syntax

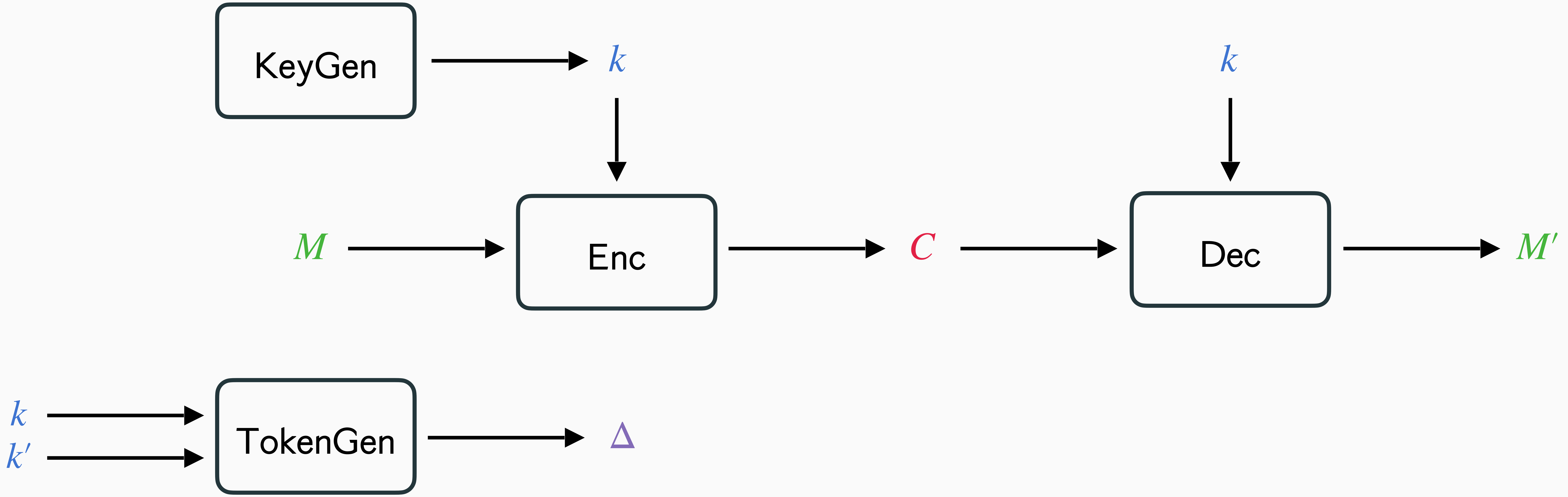
Updatable Encryption $\mathsf{UE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Upd})$



Correctness: $M = M'$

Syntax

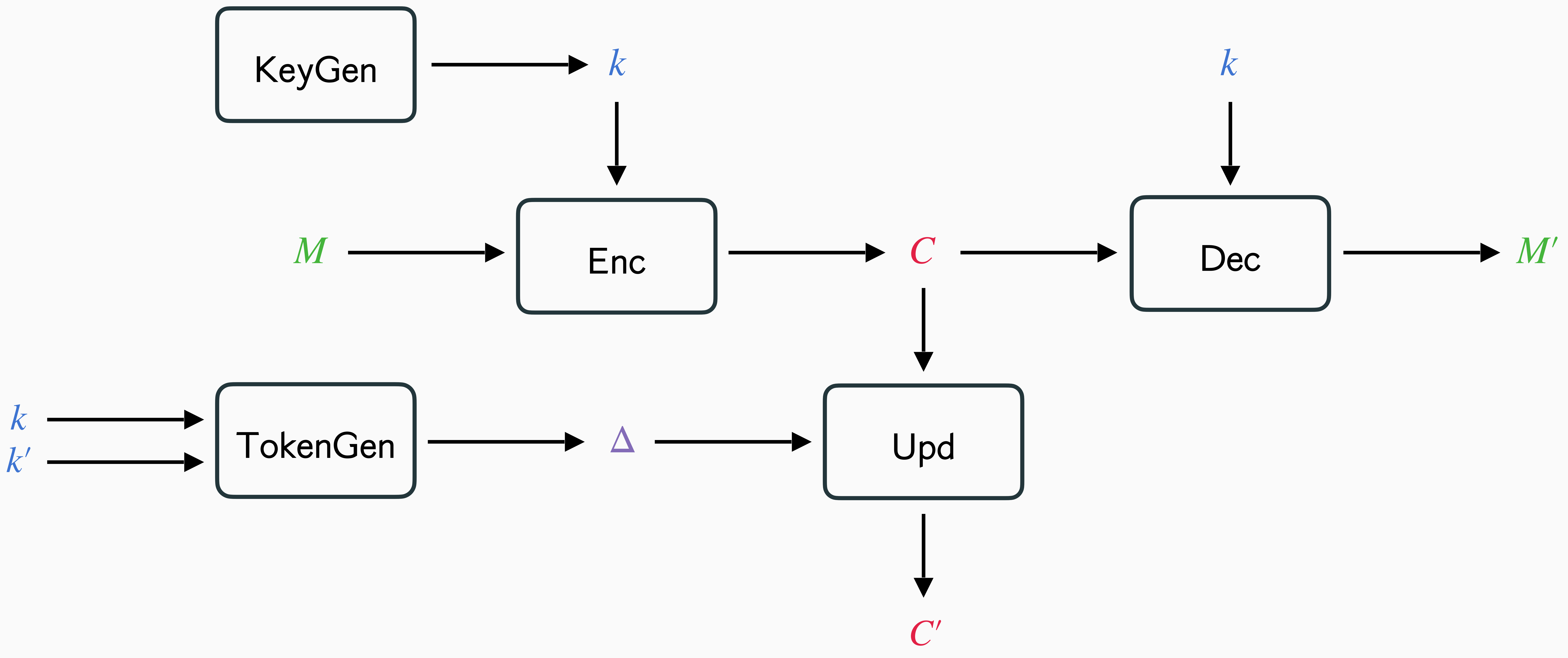
Updatable Encryption UE = (KeyGen, Enc, Dec, TokenGen, Upd)



Correctness: $M = M'$

Syntax

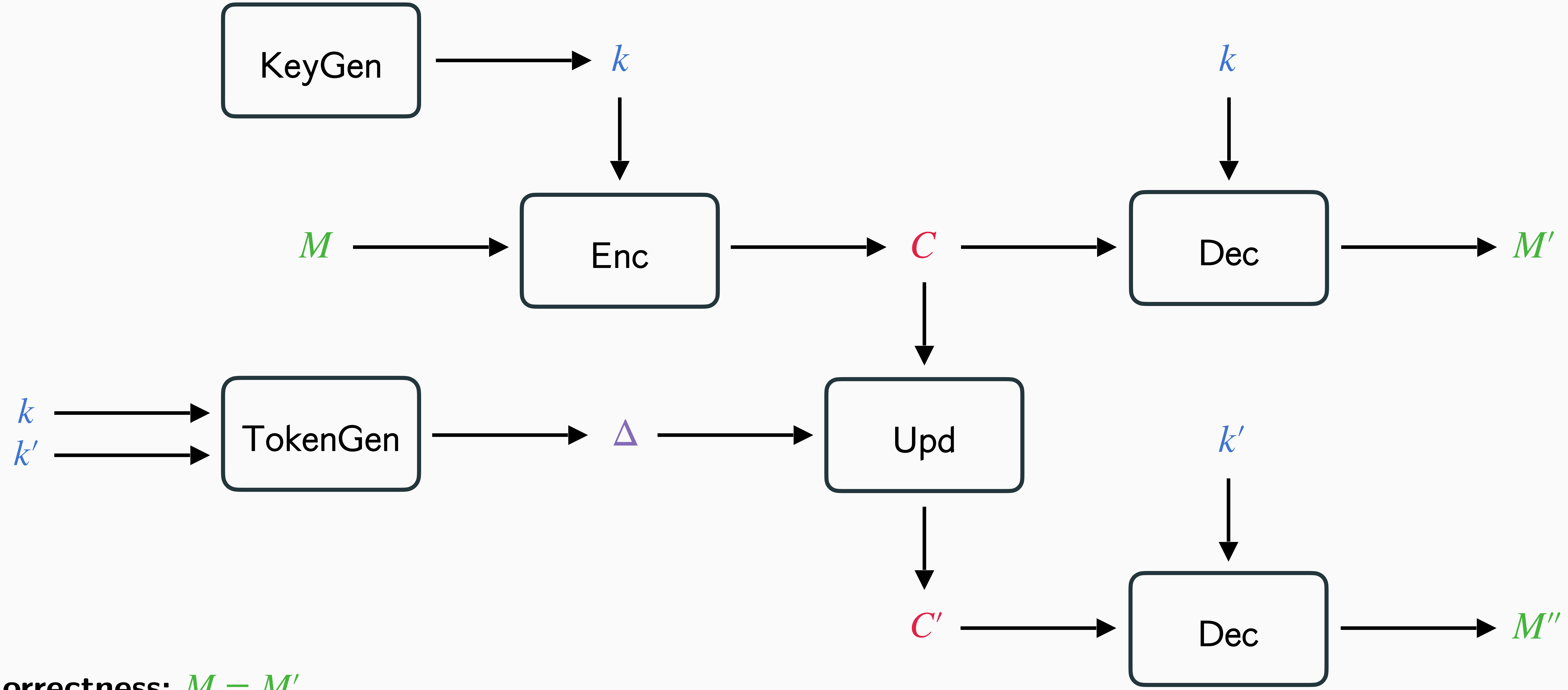
Updatable Encryption UE = (KeyGen, Enc, Dec, TokenGen, Upd)



Correctness: $M = M'$

Syntax

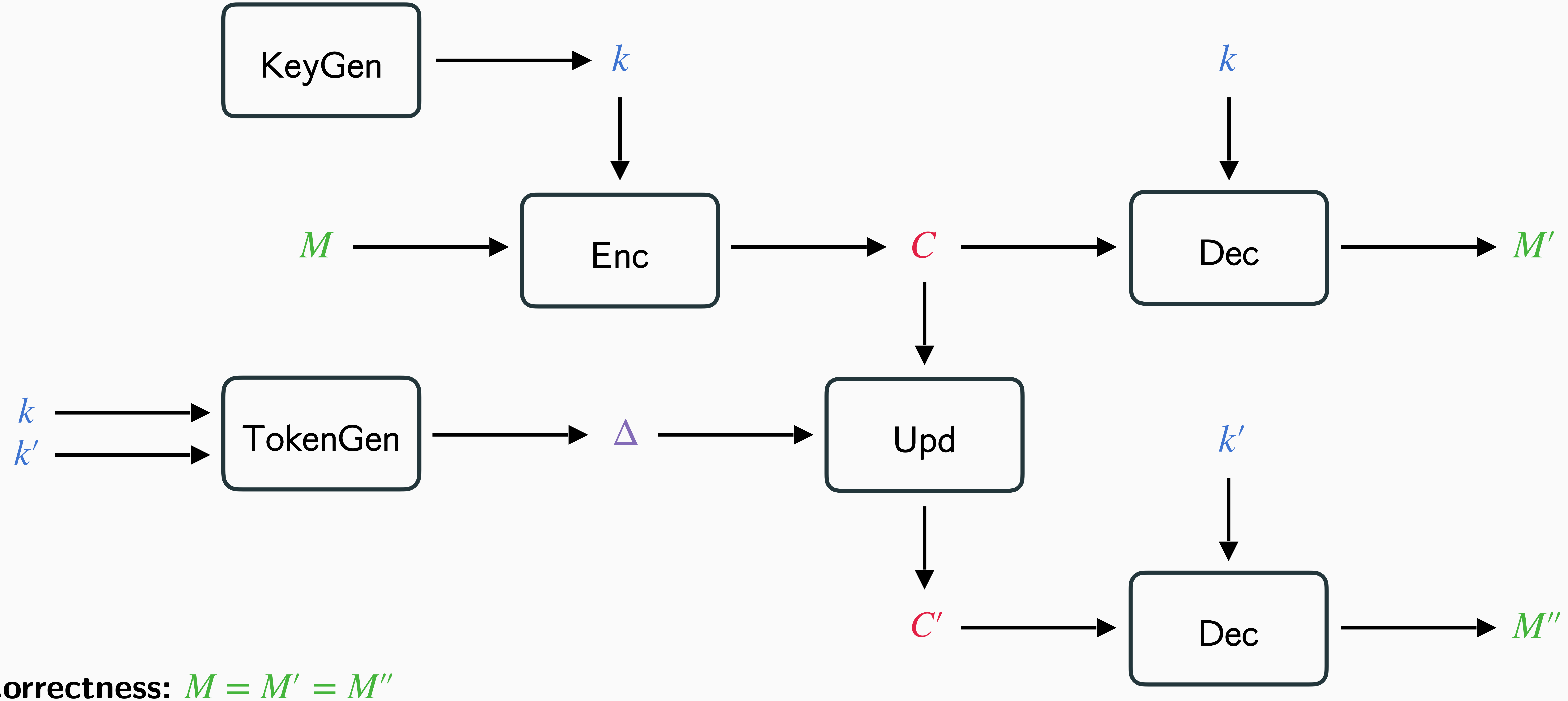
Updatable Encryption UE = (KeyGen, Enc, Dec, TokenGen, Upd)



Correctness: $M = M'$

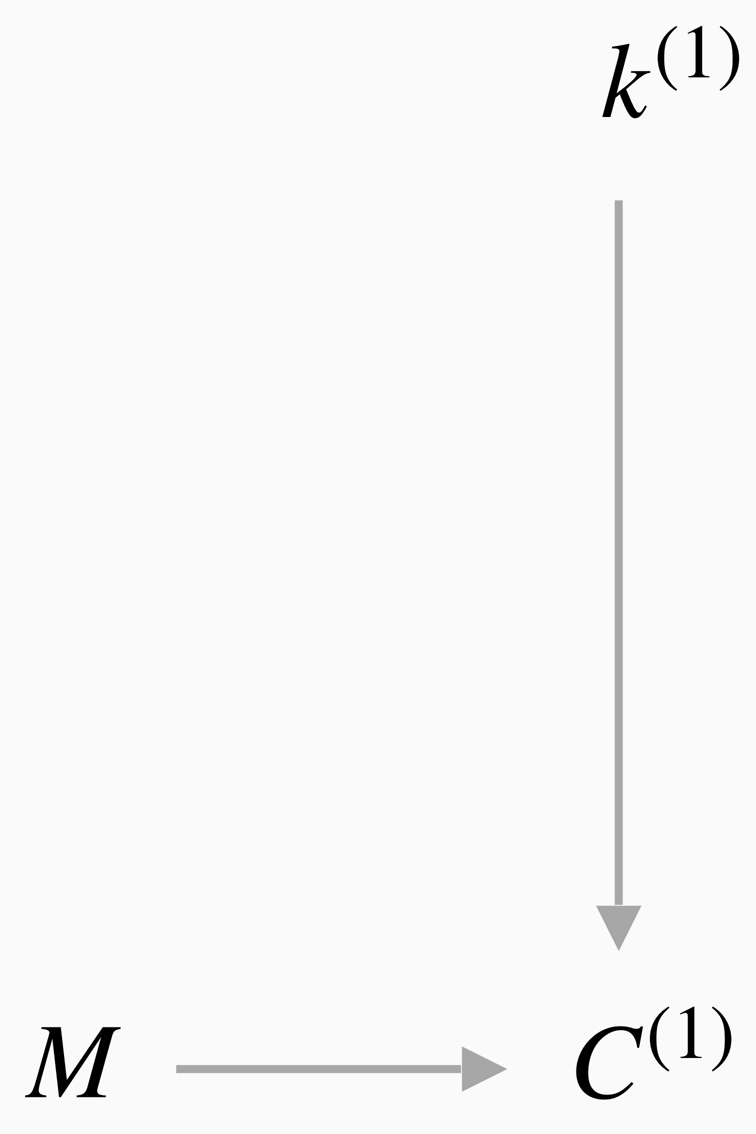
Syntax

Updatable Encryption UE = (KeyGen, Enc, Dec, TokenGen, Upd)

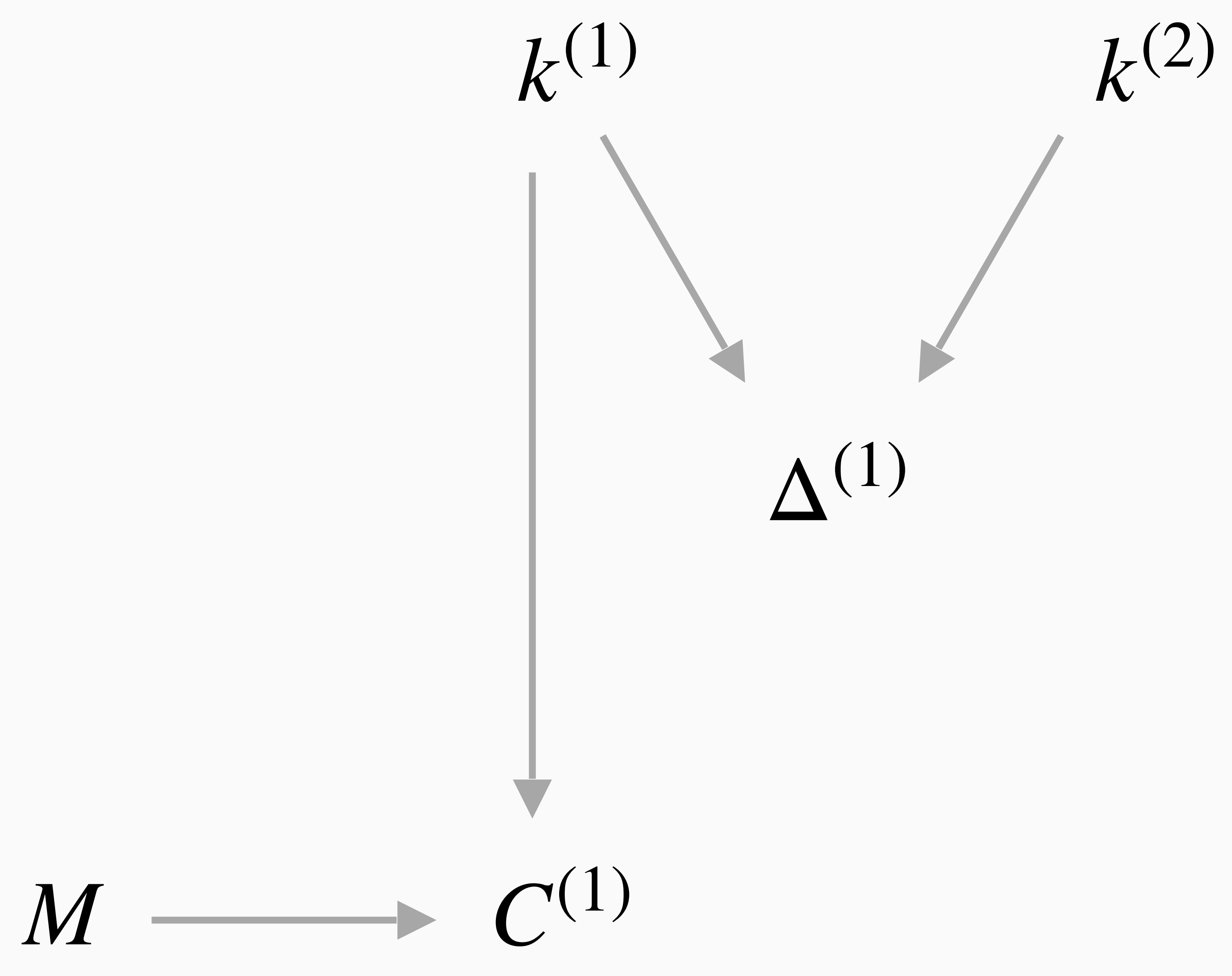


Correctness: $M = M' = M''$

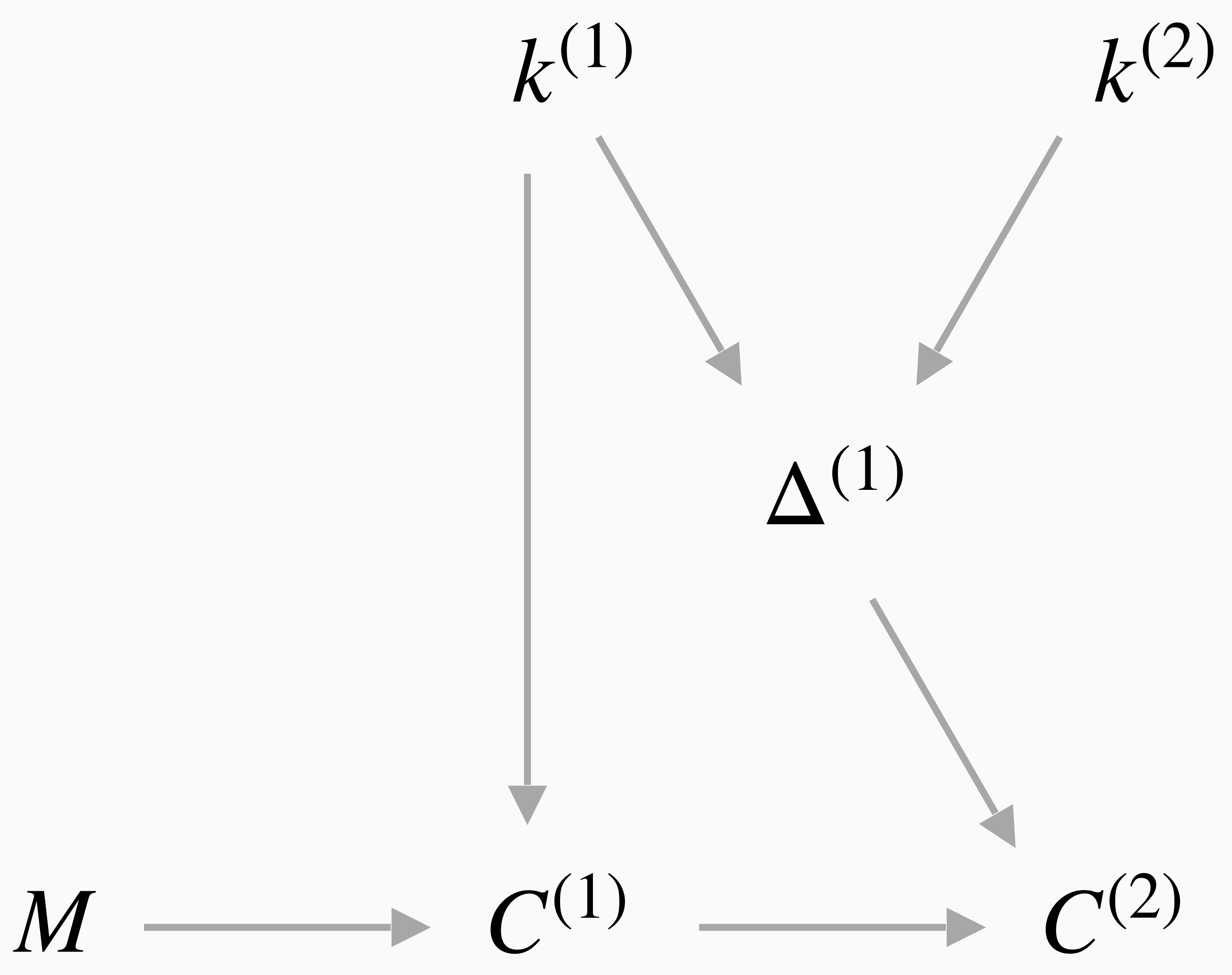
Modeling Security



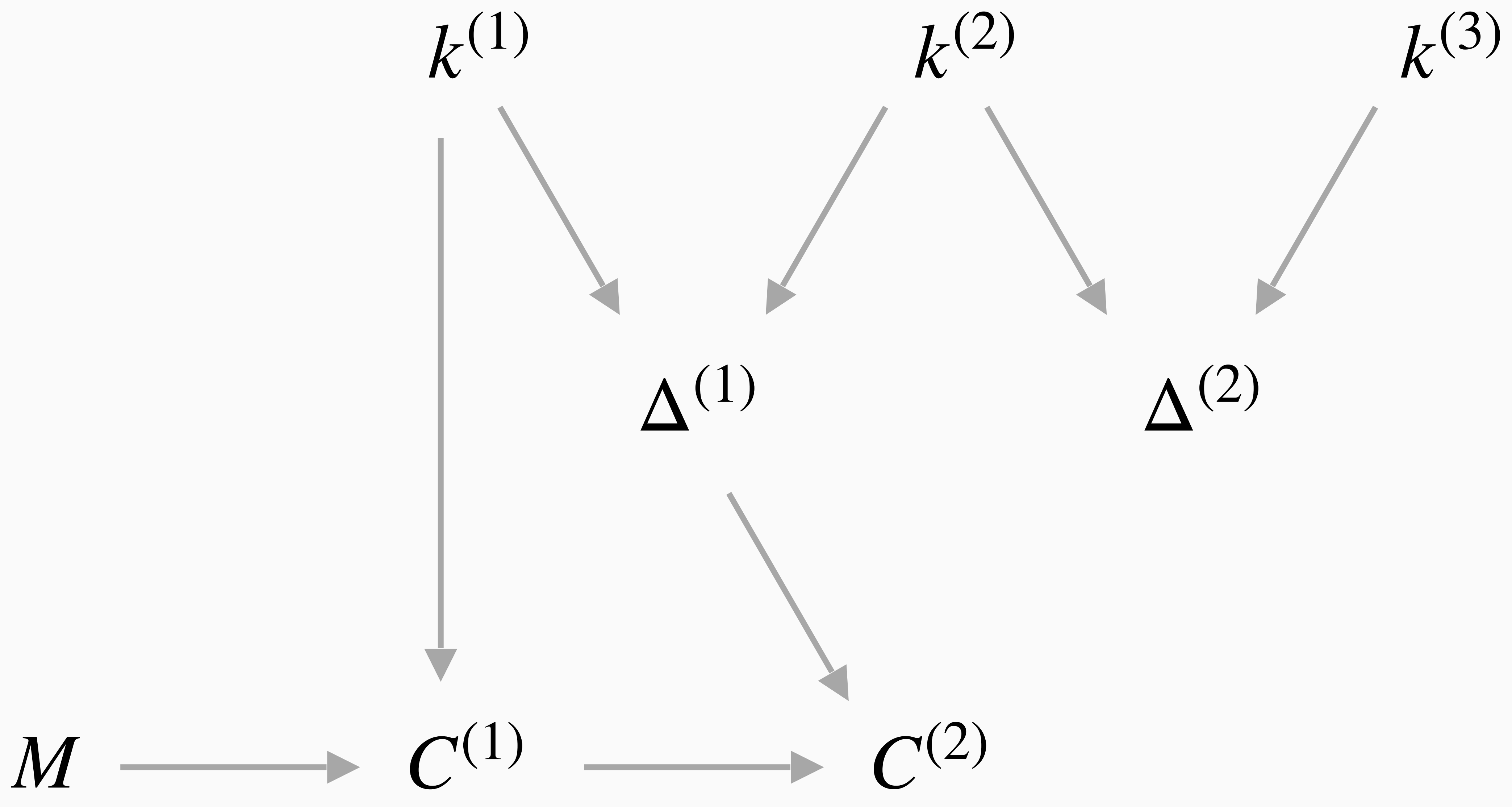
Modeling Security



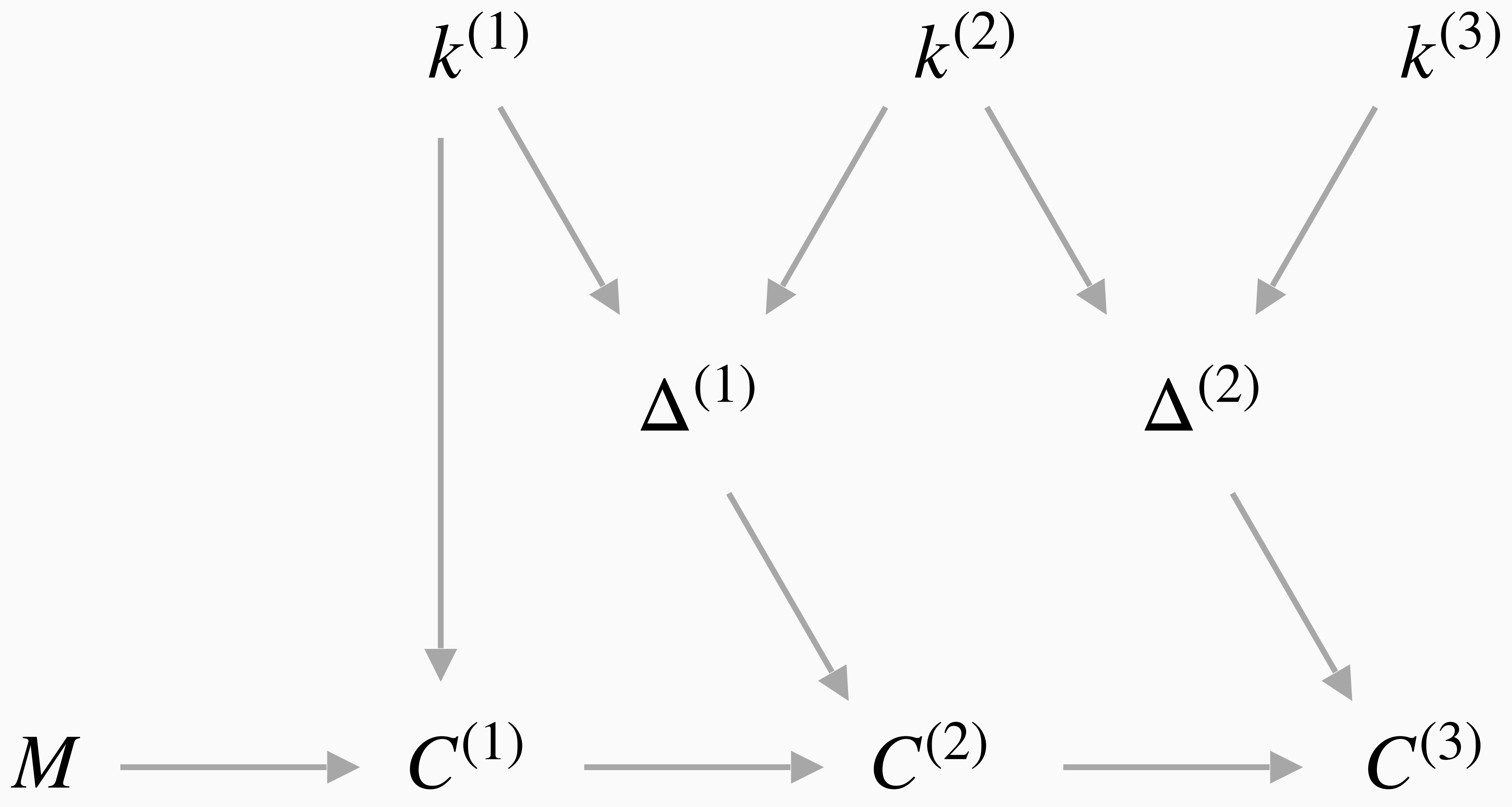
Modeling Security



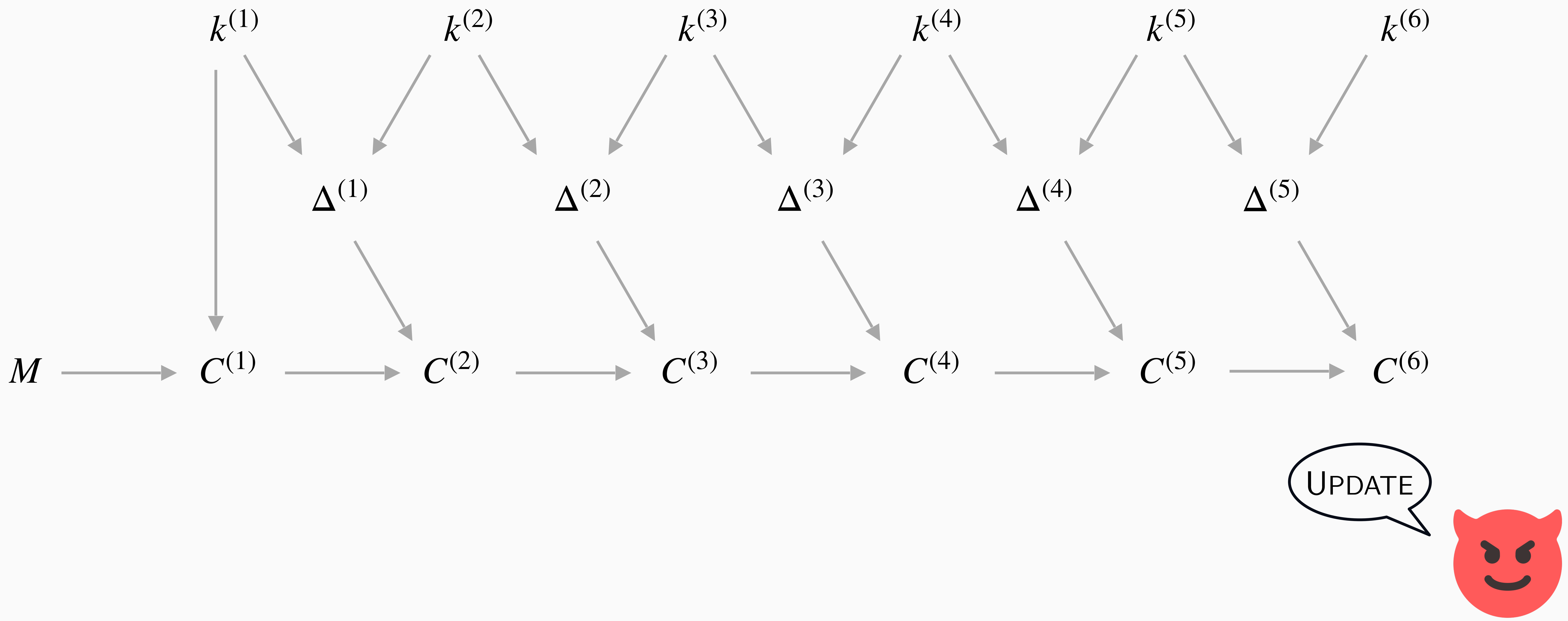
Modeling Security



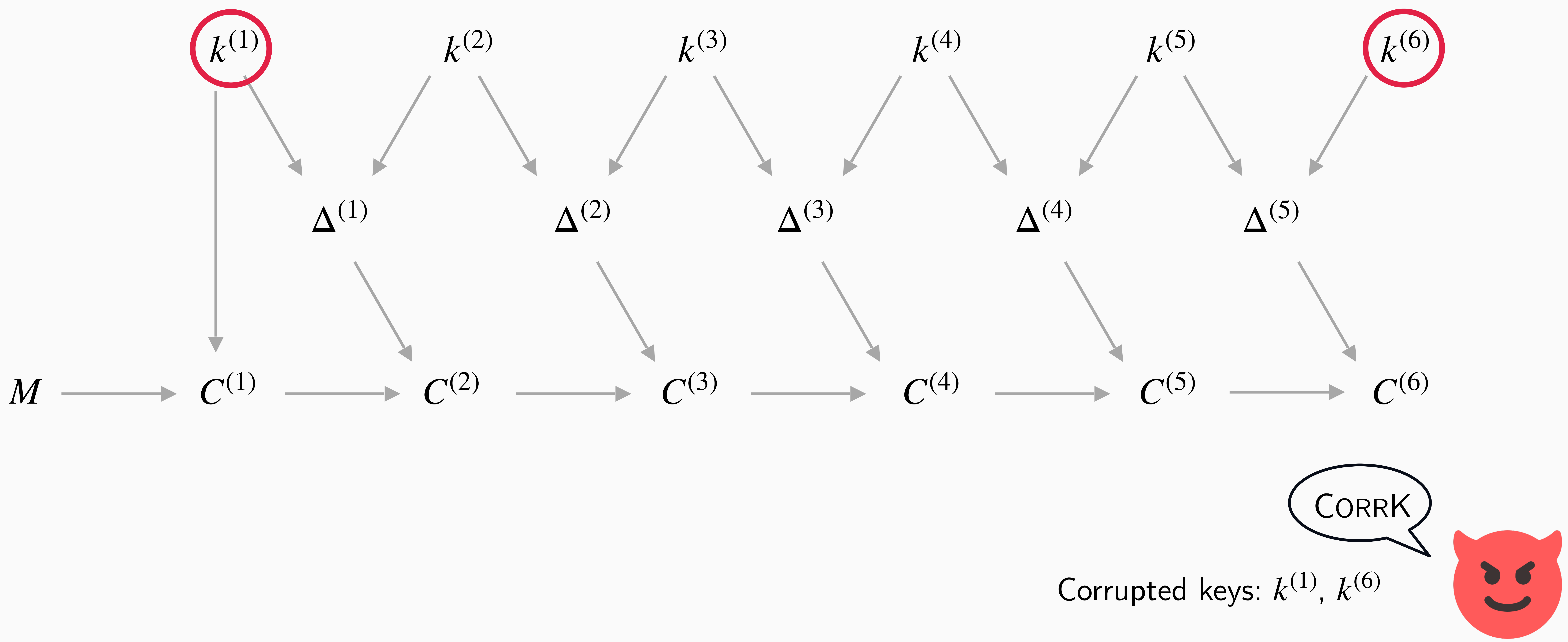
Modeling Security



Modeling Security

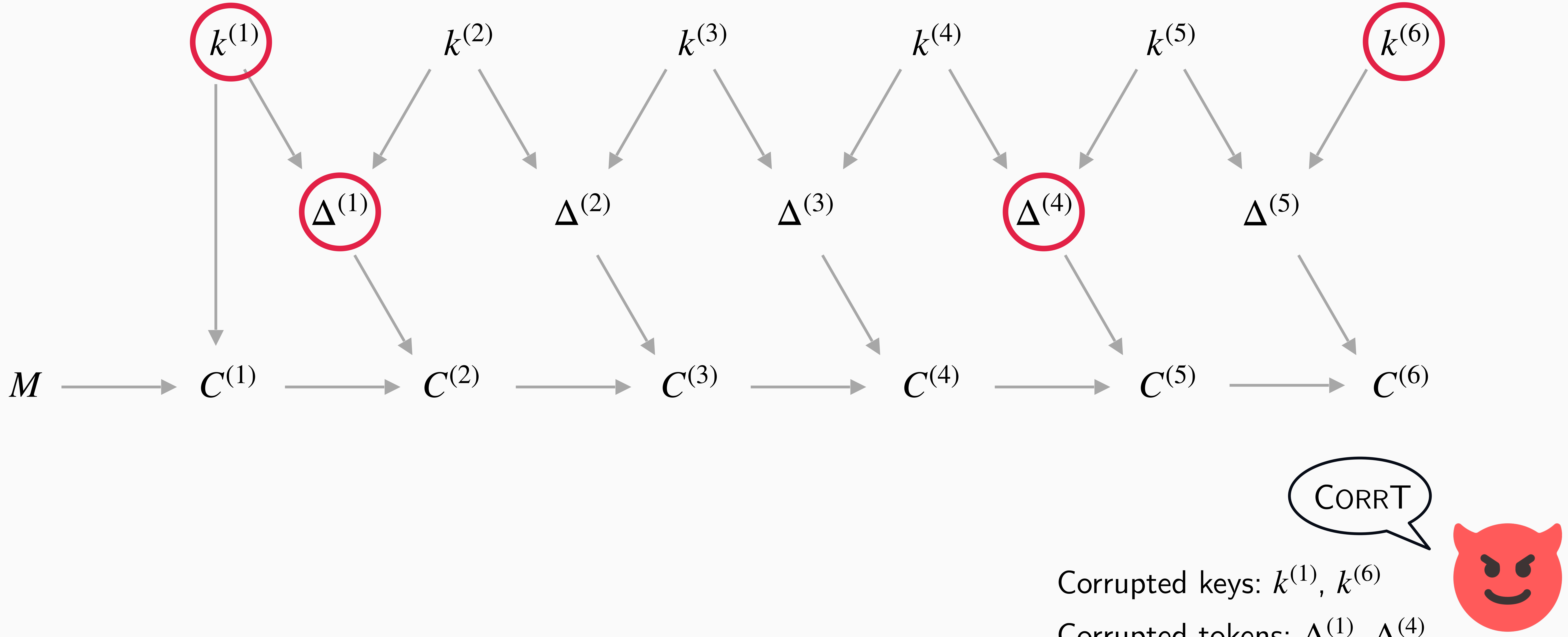


Modeling Security



Corrupted keys: $k^{(1)}$, $k^{(6)}$

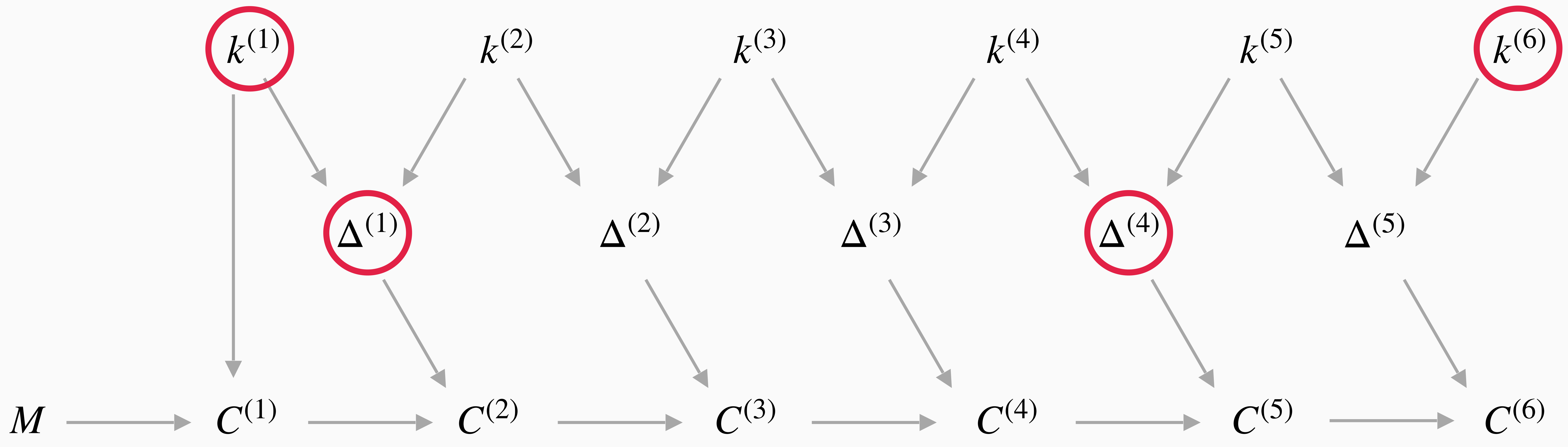
Modeling Security



Corrupted keys: $k^{(1)}$, $k^{(6)}$

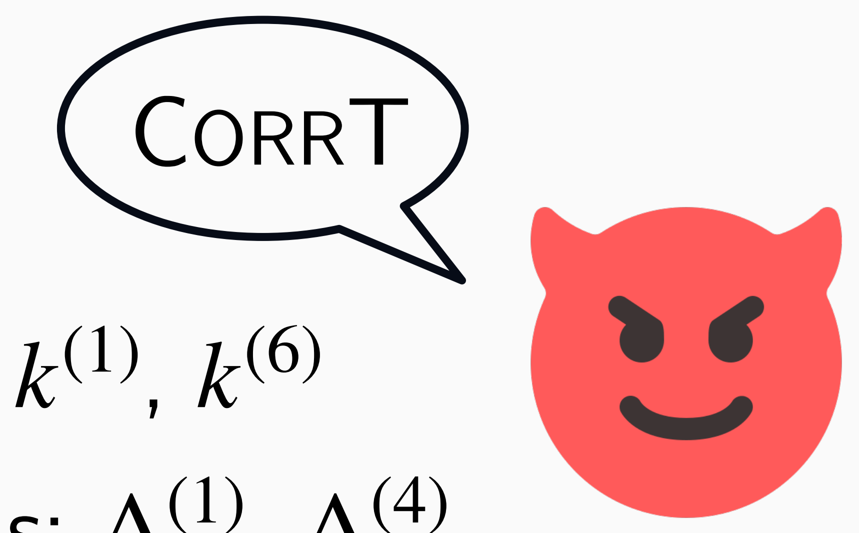
Corrupted tokens: $\Delta^{(1)}$, $\Delta^{(4)}$

Modeling Security



Exact definition depends on properties of the scheme

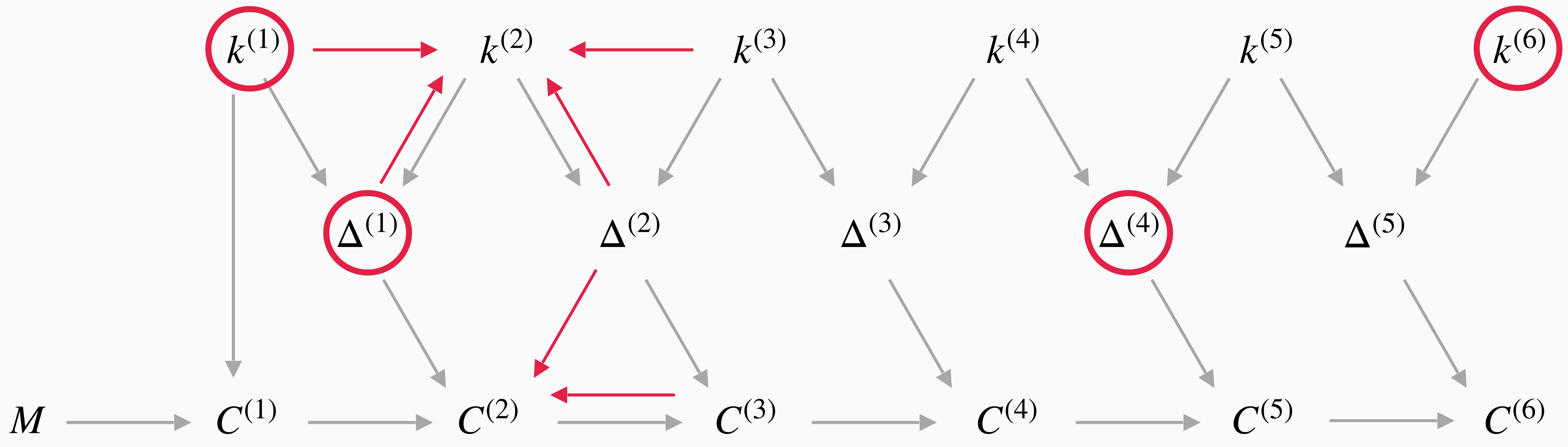
- Randomized or deterministic ciphertext updates
- Bi-directional and uni-/no-directional key updates
- CPA and (R)CCA security



Corrupted keys: $k^{(1)}$, $k^{(6)}$

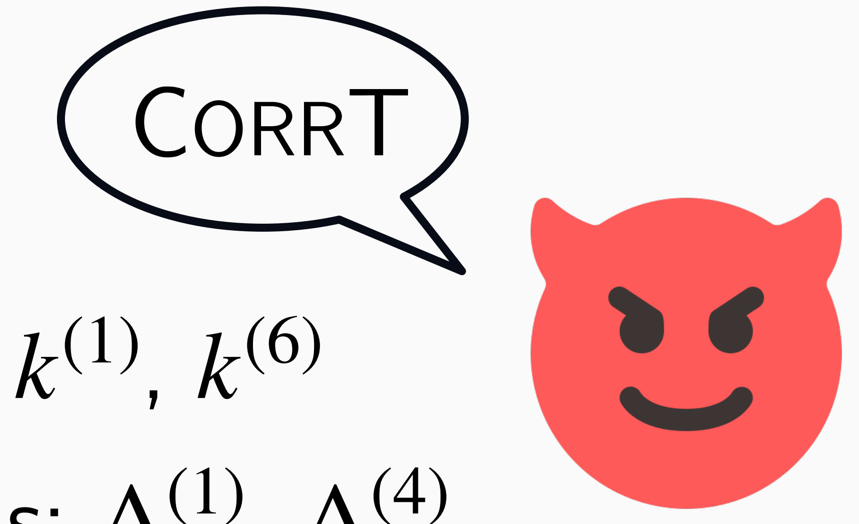
Corrupted tokens: $\Delta^{(1)}$, $\Delta^{(4)}$

Modeling Security



Exact definition depends on properties of the scheme

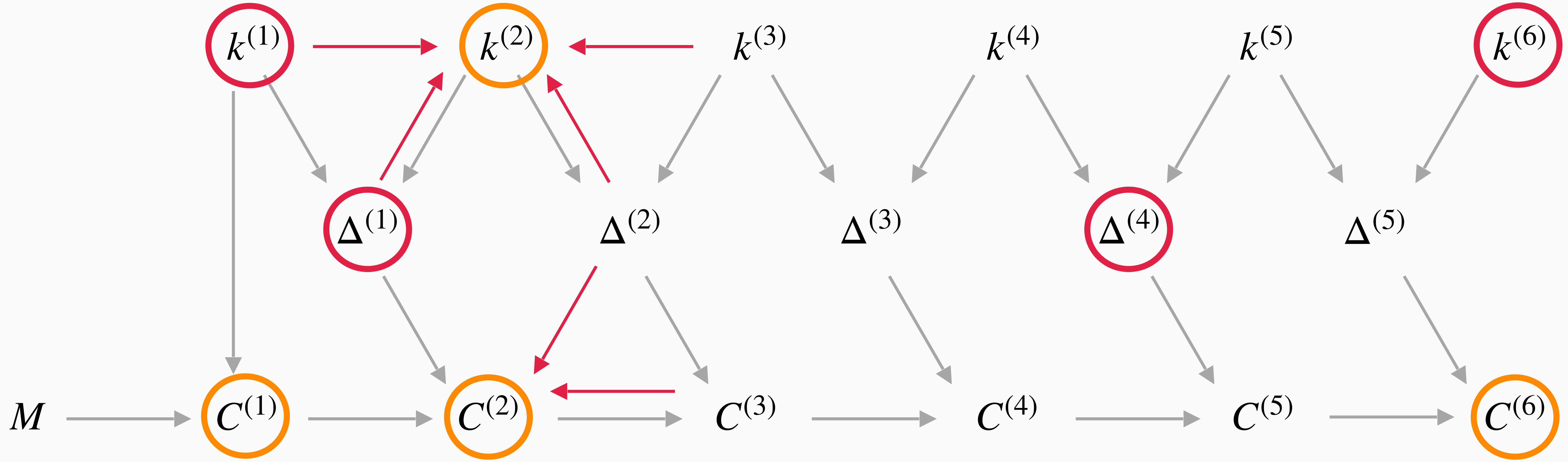
- Randomized or deterministic ciphertext updates
- Bi-directional and uni-/no-directional key updates
- CPA and (R)CCA security



Corrupted keys: $k^{(1)}$, $k^{(6)}$

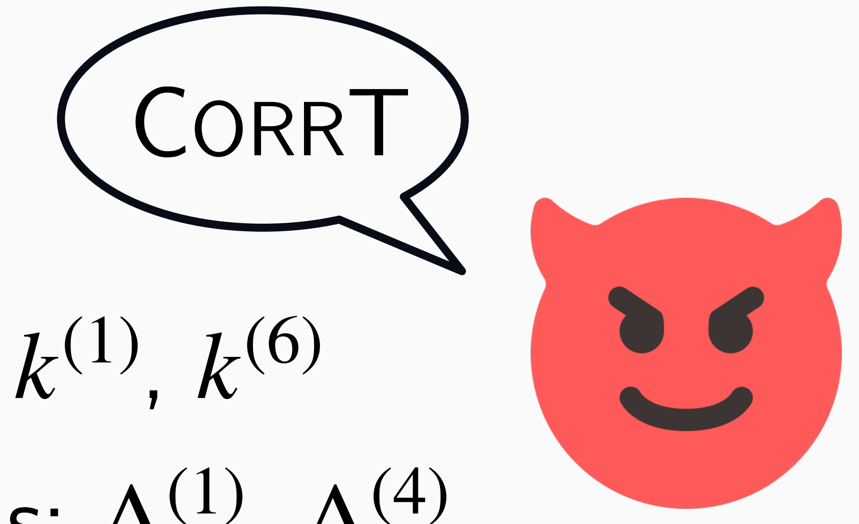
Corrupted tokens: $\Delta^{(1)}$, $\Delta^{(4)}$

Modeling Security



Exact definition depends on properties of the scheme

- Randomized or deterministic ciphertext updates
- Bi-directional and uni-/no-directional key updates
- CPA and (R)CCA security

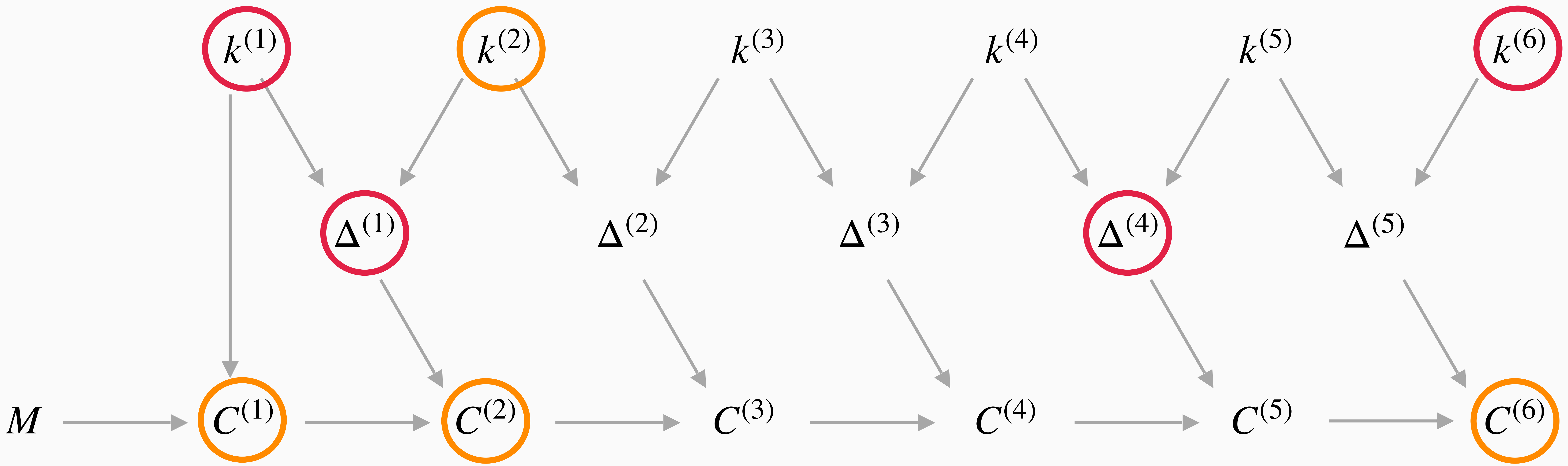


Corrupted keys: $k^{(1)}$, $k^{(6)}$

Corrupted tokens: $\Delta^{(1)}$, $\Delta^{(4)}$

Inferred knowledge: $k^{(2)}$, $M^{(1)}$, $M^{(2)}$, $M^{(6)}$

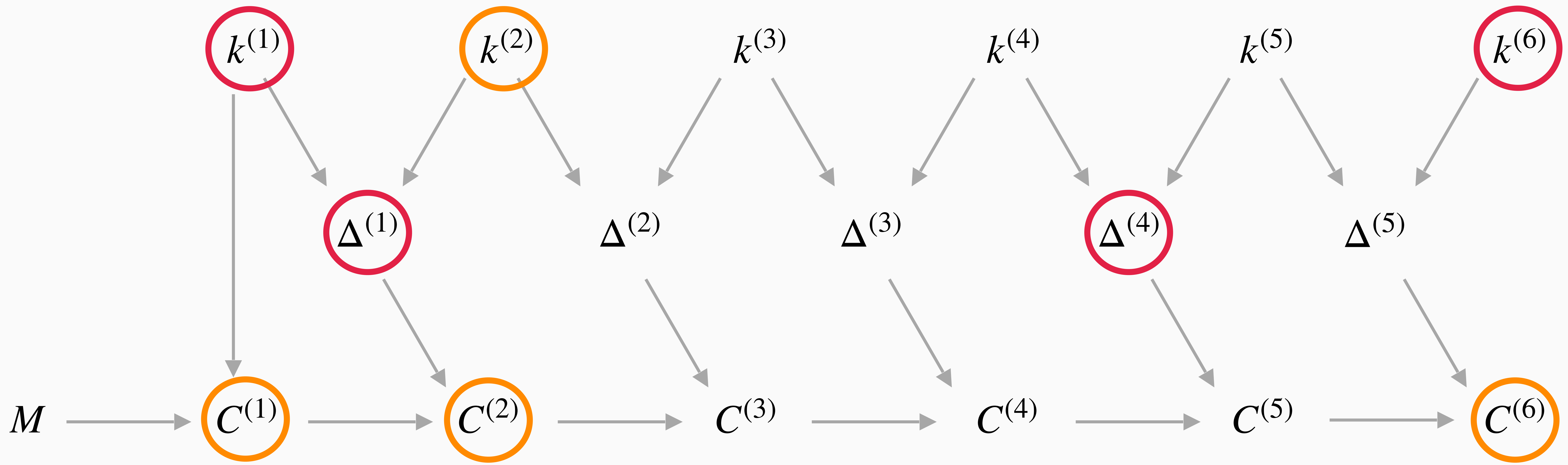
Modeling Security

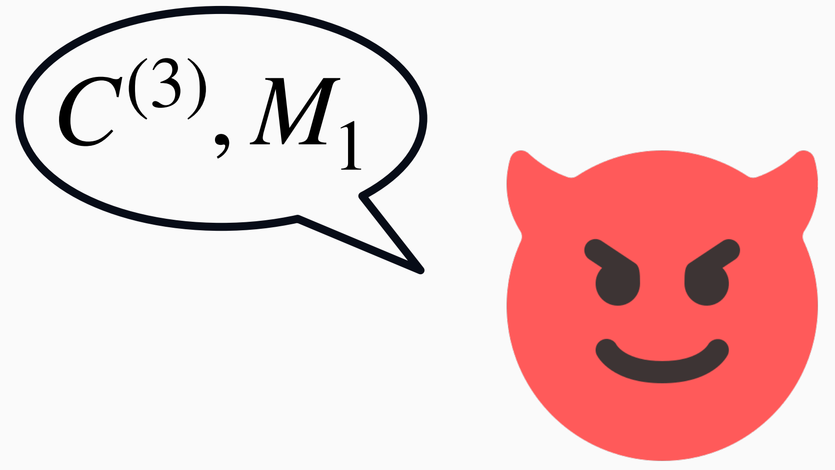


(det)IND-UE-CPA

- Distinguish updated ciphertext from encryption of a new message

Modeling Security

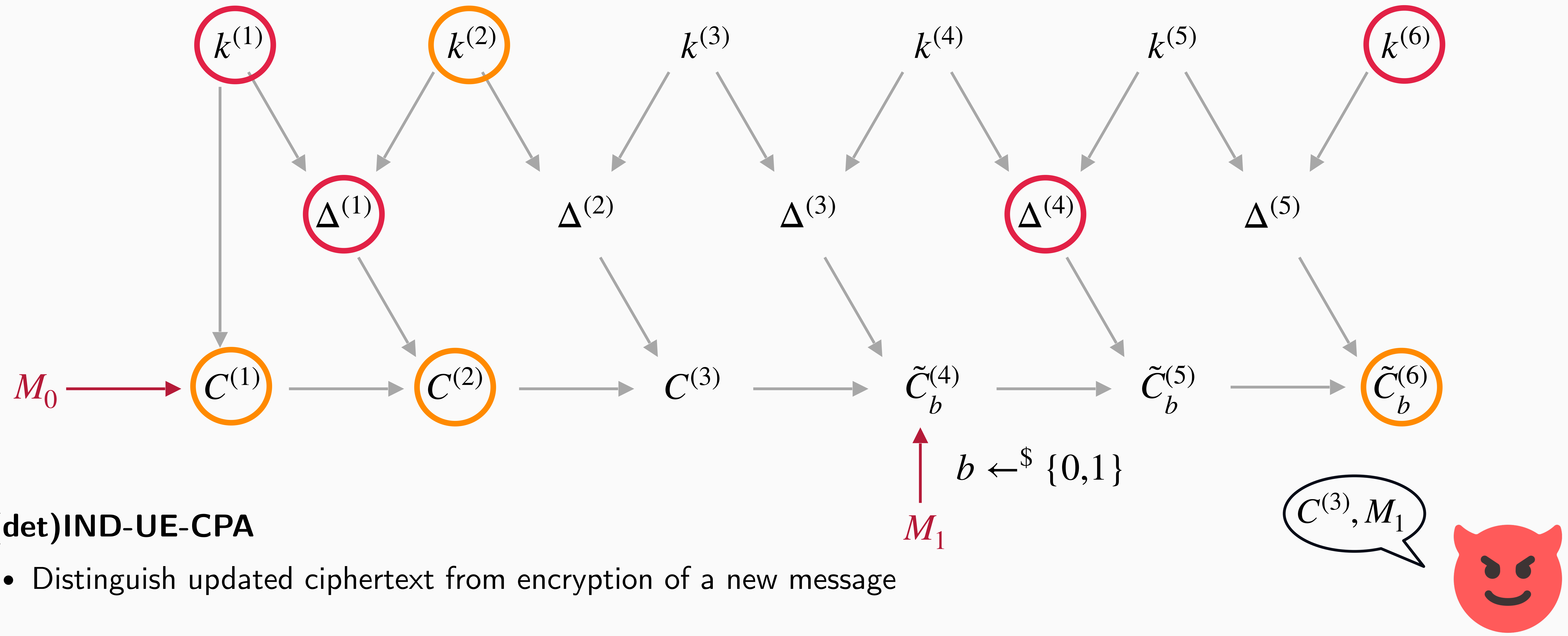




(det)IND-UE-CPA

- Distinguish updated ciphertext from encryption of a new message

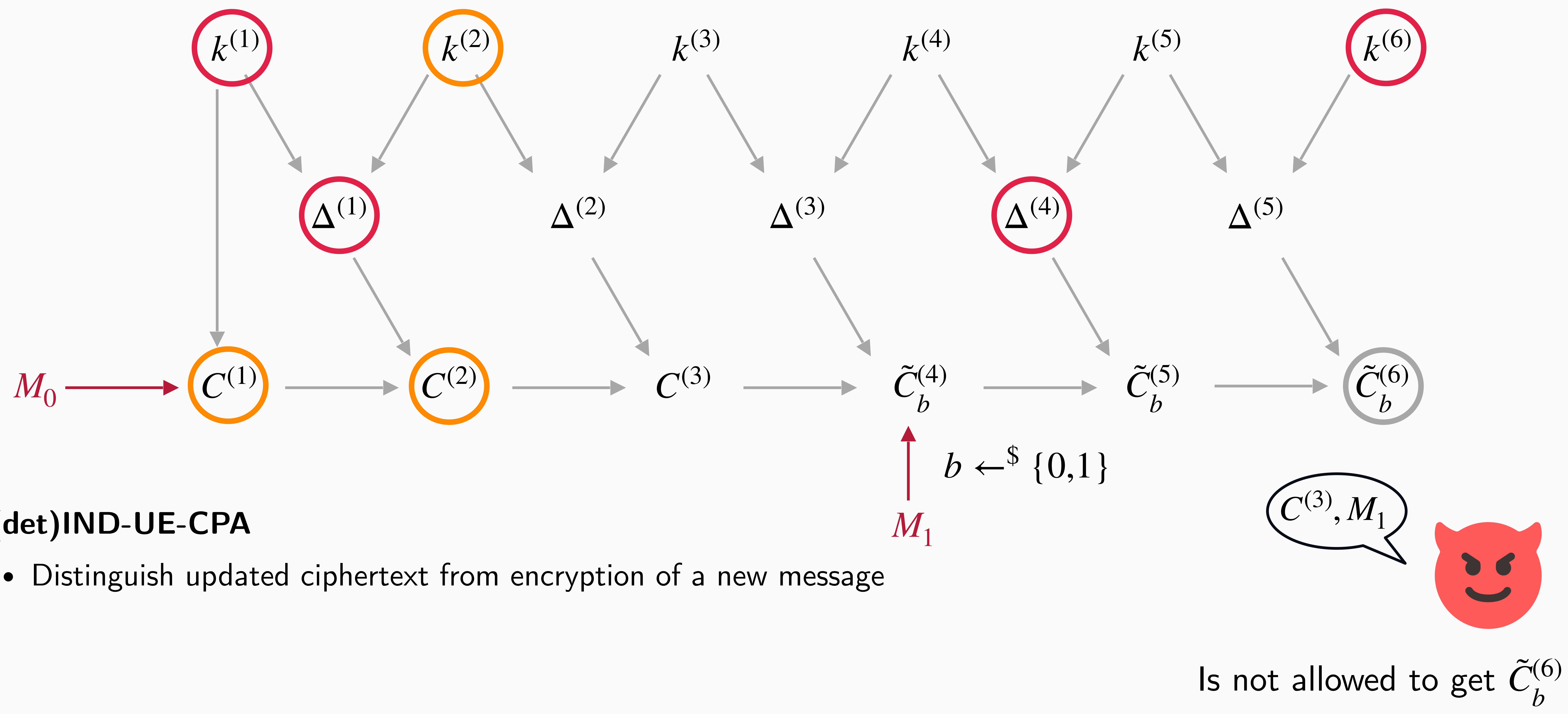
Modeling Security



(det)IND-UE-CPA

- Distinguish updated ciphertext from encryption of a new message

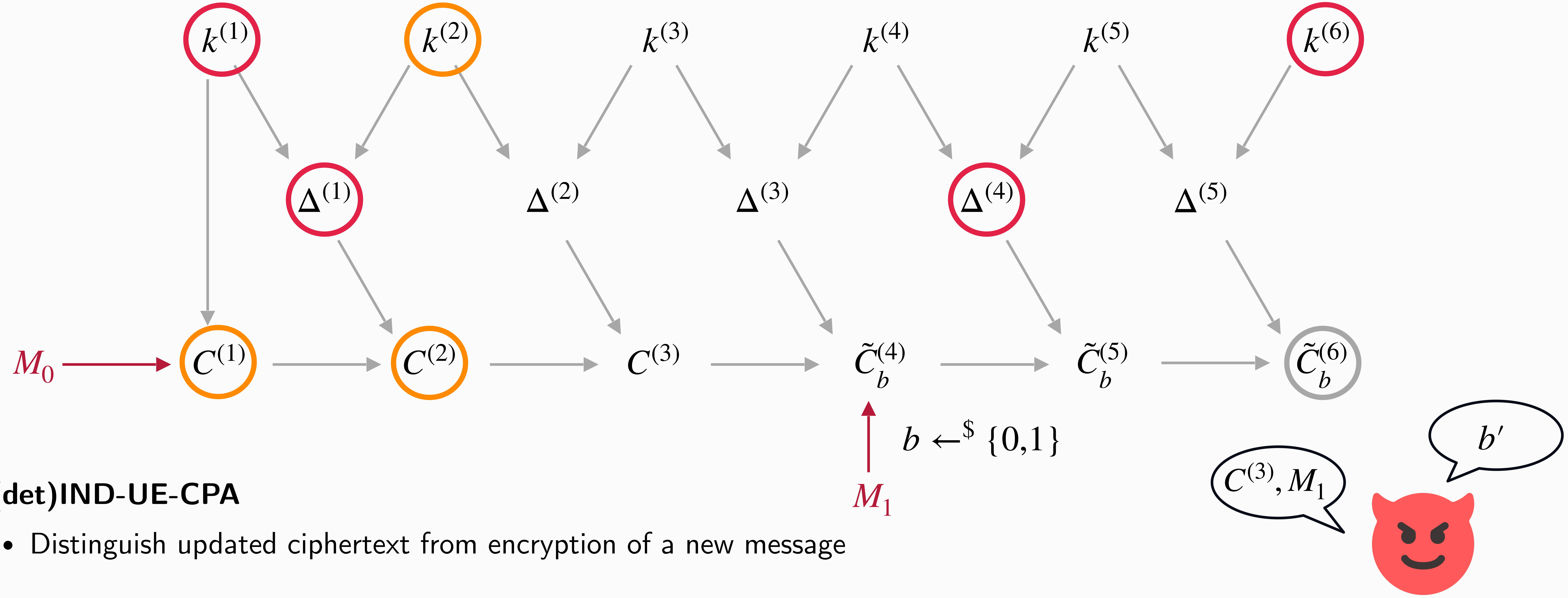
Modeling Security



(det)IND-UE-CPA

- Distinguish updated ciphertext from encryption of a new message

Modeling Security



(det)IND-UE-CPA

- Distinguish updated ciphertext from encryption of a new message

Is not allowed to get $\tilde{C}_b^{(6)}$

Cryptographic Group Actions

Definition: Group Action

Let $(\mathscr{G}, \cdot\,)$ be a group with identity element e and $\mathscr{X}$ a set. A group action is a map

$$\star : \mathscr{G} \times \mathscr{X} \to \mathscr{X}$$

which satisfies

1. Identity: $e \star x = x$ for all $x \in \mathscr{X}$
2. Compatibility: $(g \cdot h) \star x = g \star (h \star x)$ for all $g, h \in \mathscr{G},\ x \in \mathscr{X}$

Cryptographic Group Actions

Definition: Group Action

Let $(\mathcal{G}, \cdot)$ be a group with identity element e and $\mathcal{X}$ a set. A group action is a map

$$\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$$

which satisfies

1. Identity: $e \star x = x$ for all $x \in \mathcal{X}$
2. Compatibility: $(g \cdot h) \star x = g \star (h \star x)$ for all $g, h \in \mathcal{G},\ x \in \mathcal{X}$

Computational Problems

- DLOG: given $(x, g \star x)$ for $g \xleftarrow{\$} \mathcal{G}$, compute g.
- CDH: given $(x,\ g \star x,\ h \star x)$ for $g, h \xleftarrow{\$} \mathcal{G}$, compute $gh \star x$.
- DDH: given $(x,\ g \star x,\ h \star x,\ z)$, decide whether $z = gh \star x$ or random

Cryptographic Group Actions

Definition: Group Action

Let $(\mathcal{G}, \cdot)$ be a group with identity element e and $\mathcal{X}$ a set. A group action is a map

$$\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$$

which satisfies

1. Identity: $e \star x = x$ for all $x \in \mathcal{X}$
2. Compatibility: $(g \cdot h) \star x = g \star (h \star x)$ for all $g, h \in \mathcal{G},\ x \in \mathcal{X}$

Computational Problems

- DLOG: given $(x, g \star x)$ for $g \xleftarrow{\$} \mathcal{G}$, compute g.
- CDH: given $(x,\ g \star x,\ h \star x)$ for $g, h \xleftarrow{\$} \mathcal{G}$, compute $gh \star x$.
- DDH: given $(x,\ g \star x,\ h \star x,\ z)$, decide whether $z = gh \star x$ or random

CSIDH [AC:CLMPR18]

$\mathcal{G}$ = isogenies between elliptic curves

$\mathcal{X}$ = supersingular elliptic curves over $\mathbb{F}_p$

Group Action UE Scheme

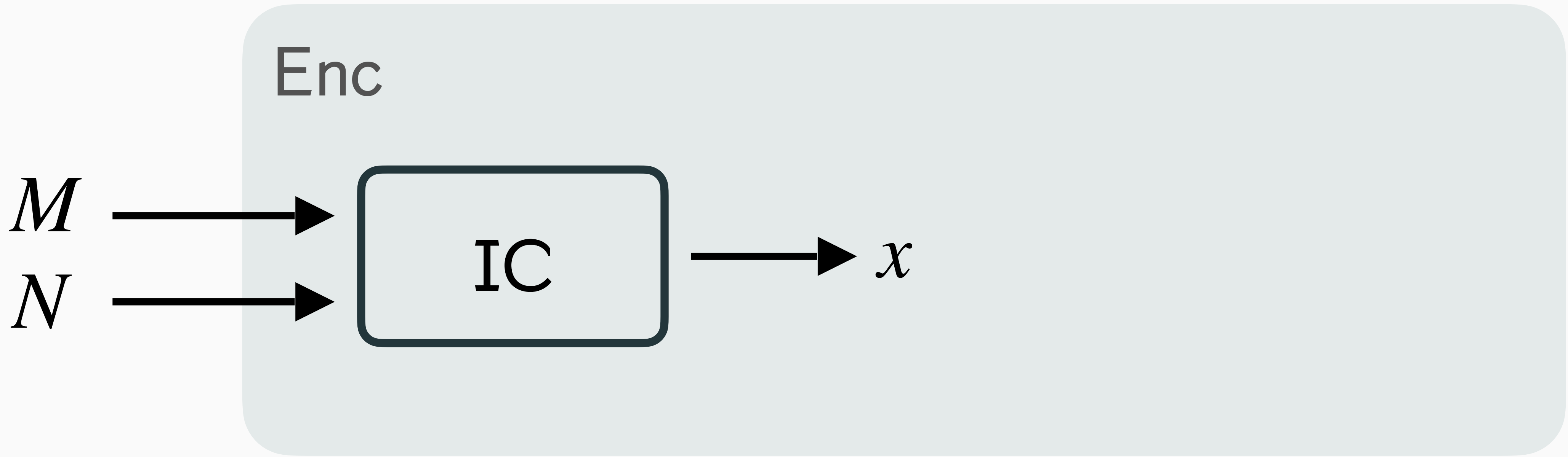
GAINE [PQCRYPTO:LerRom24]

- Adaptation of SHINE [C:BDGJ20] to group actions
- Key $k \in \mathcal{G}$, ideal cipher $\mathsf{IC} : \{0,1\}^{\ell} \times \{0,1\}^{\lambda} \to \mathcal{X}$ maps message and random nonce to the set ("mappable")

Group Action UE Scheme

GAINE [PQCRYPTO:LerRom24]

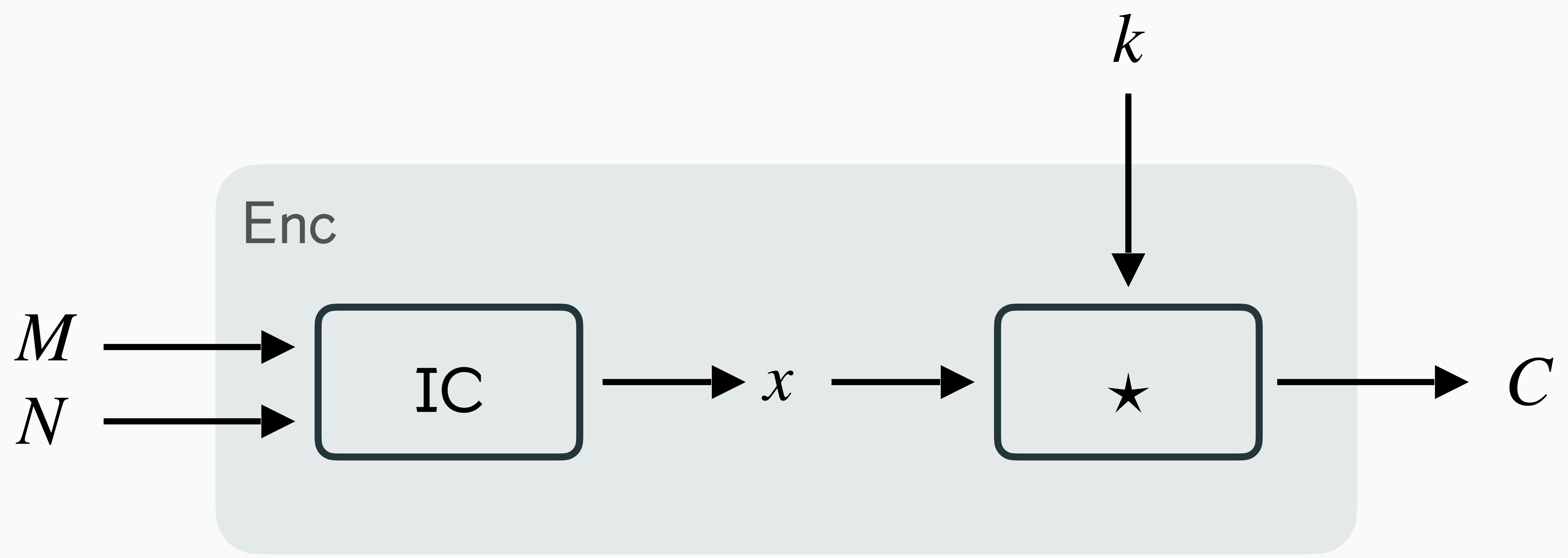
- Adaptation of SHINE [C:BDGJ20] to group actions
- Key $k \in \mathcal{G}$, ideal cipher $\mathsf{IC} : \{0,1\}^{\ell} \times \{0,1\}^{\lambda} \to \mathcal{X}$ maps message and random nonce to the set ("mappable")



Group Action UE Scheme

GAINE [PQCRYPTO:LerRom24]

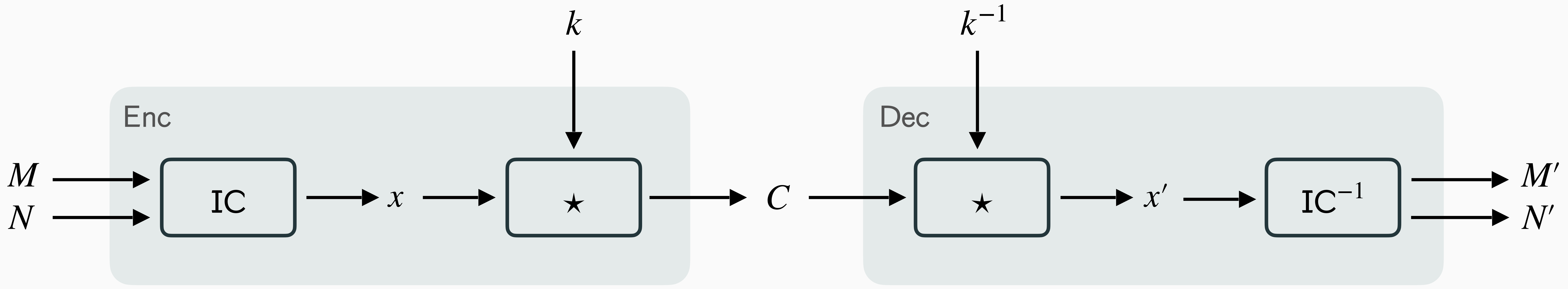
- Adaptation of SHINE [C:BDGJ20] to group actions
- Key $k \in \mathscr{G}$, ideal cipher $\mathsf{IC} : \{0,1\}^{\ell} \times \{0,1\}^{\lambda} \to \mathscr{X}$ maps message and random nonce to the set ("mappable")



Group Action UE Scheme

GAINE [PQCRYPTO:LerRom24]

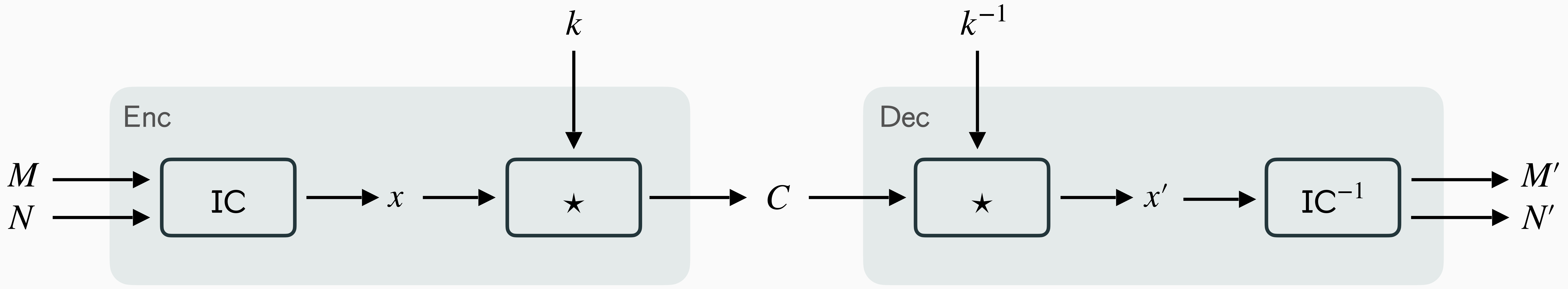
- Adaptation of SHINE [C:BDGJ20] to group actions
- Key $k \in \mathcal{G}$, ideal cipher $\mathsf{IC} : \{0,1\}^{\ell} \times \{0,1\}^{\lambda} \to \mathcal{X}$ maps message and random nonce to the set ("mappable")



Group Action UE Scheme

GAINE [PQCRYPTO:LerRom24]

- Adaptation of SHINE [C:BDGJ20] to group actions
- Key $k \in \mathcal{G}$, ideal cipher IC : $\{0,1\}^{\ell} \times \{0,1\}^{\lambda} \to \mathcal{X}$ maps message and random nonce to the set ("mappable")



But: For CSIDH we do not know how to map into $\mathcal{X}$ [EPRINT:BBDFGKMPSSTVVWZ22,EPRINT:MulMurPin22].

Our Schemes

Scheme 1: BIN-UE

Message space $\mathscr{M} = \{0,1\}^n \setminus \{0^n, 1^n\}$, $M = (m_1, \ldots, m_n)$ for some $n > 1$

Key space $\mathscr{K} = \mathscr{G}^n$, need some "ordering" for set elements in $\mathscr{X}$

Scheme 1: BIN-UE

Message space $\mathscr{M} = \{0,1\}^n \backslash \{0^n, 1^n\}$, $M = (m_1, \ldots, m_n)$ for some $n > 1$

Key space $\mathscr{K} = \mathscr{G}^n$, need some "ordering" for set elements in $\mathscr{X}$

Encryption

Pick $x_0, x_1 \xleftarrow{\$} \mathscr{X}$ s.t. $x_0 \prec x_1$
(can be done by sampling from $\mathscr{G}$)

$$\begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} \star \begin{pmatrix} x_{m_1} \\ x_{m_2} \\ \vdots \\ x_{m_n} \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix}$$

Scheme 1: BIN-UE

Message space $\mathscr{M} = \{0,1\}^n \setminus \{0^n, 1^n\}$, $M = (m_1, \ldots, m_n)$ for some $n > 1$

Key space $\mathscr{K} = \mathscr{G}^n$, need some "ordering" for set elements in $\mathscr{X}$

Encryption

Pick $x_0, x_1 \xleftarrow{\$} \mathscr{X}$ s.t. $x_0 \prec x_1$

(can be done by sampling from $\mathscr{G}$)

$$\begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} \star \begin{pmatrix} x_{m_1} \\ x_{m_2} \\ \vdots \\ x_{m_n} \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix}$$

Token Generation and Update

$$\overbrace{\begin{pmatrix} k'_1 \\ k'_2 \\ \vdots \\ k'_n \end{pmatrix} \cdot \begin{pmatrix} k_1^{-1} \\ k_2^{-1} \\ \vdots \\ k_n^{-1} \end{pmatrix}}^{\Delta}$$

Scheme 1: BIN-UE

Message space $\mathcal{M} = \{0,1\}^n \setminus \{0^n, 1^n\}$, $M = (m_1, \ldots, m_n)$ for some $n > 1$

Key space $\mathcal{K} = \mathcal{G}^n$, need some "ordering" for set elements in $\mathcal{X}$

Encryption

Pick $x_0, x_1 \xleftarrow{\$} \mathcal{X}$ s.t. $x_0 \prec x_1$

(can be done by sampling from $\mathcal{G}$)

$$\begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} \star \begin{pmatrix} x_{m_1} \\ x_{m_2} \\ \vdots \\ x_{m_n} \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix}$$

Token Generation and Update

$$\overbrace{\begin{pmatrix} k'_1 \\ k'_2 \\ \vdots \\ k'_n \end{pmatrix} \cdot \begin{pmatrix} k_1^{-1} \\ k_2^{-1} \\ \vdots \\ k_n^{-1} \end{pmatrix}}^{\Delta} \star \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix} = \begin{pmatrix} C'_1 \\ C'_2 \\ \vdots \\ C'_n \end{pmatrix}$$

Scheme 1: BIN-UE

Message space $\mathscr{M} = \{0,1\}^n \backslash \{0^n, 1^n\}$, $M = (m_1, \ldots, m_n)$ for some $n > 1$

Key space $\mathscr{K} = \mathscr{G}^n$, need some "ordering" for set elements in $\mathscr{X}$

Encryption

Pick $x_0, x_1 \xleftarrow{\$} \mathscr{X}$ s.t. $x_0 \prec x_1$

(can be done by sampling from $\mathscr{G}$)

$$\begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} \star \begin{pmatrix} x_{m_1} \\ x_{m_2} \\ \vdots \\ x_{m_n} \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix}$$

Token Generation and Update

$$\overbrace{\begin{pmatrix} k'_1 \\ k'_2 \\ \vdots \\ k'_n \end{pmatrix} \cdot \begin{pmatrix} k_1^{-1} \\ k_2^{-1} \\ \vdots \\ k_n^{-1} \end{pmatrix}}^{\Delta} \star \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix} = \begin{pmatrix} C'_1 \\ C'_2 \\ \vdots \\ C'_n \end{pmatrix}$$

Decryption

If $|\{x'_0, x'_1, \ldots x'_n\}| = 2$:

Parse bits of M

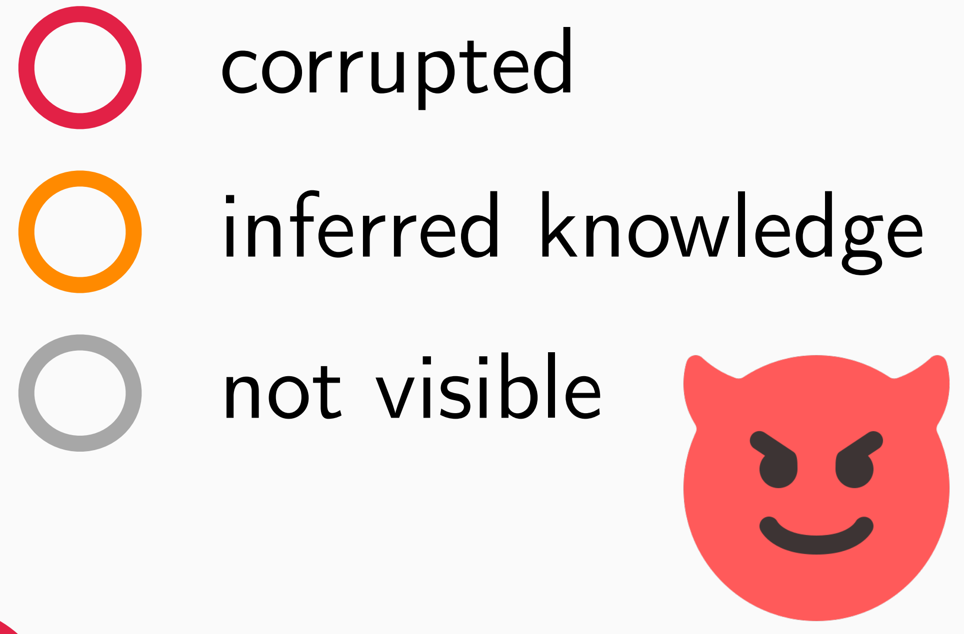
$$\begin{pmatrix} k_1'^{-1} \\ k_2'^{-1} \\ \vdots \\ k_n'^{-1} \end{pmatrix} \star \begin{pmatrix} C'_1 \\ C'_2 \\ \vdots \\ C'_n \end{pmatrix} = \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix}$$

CPA Security of BIN-UE

Firewall technique [C:BDGJ20]

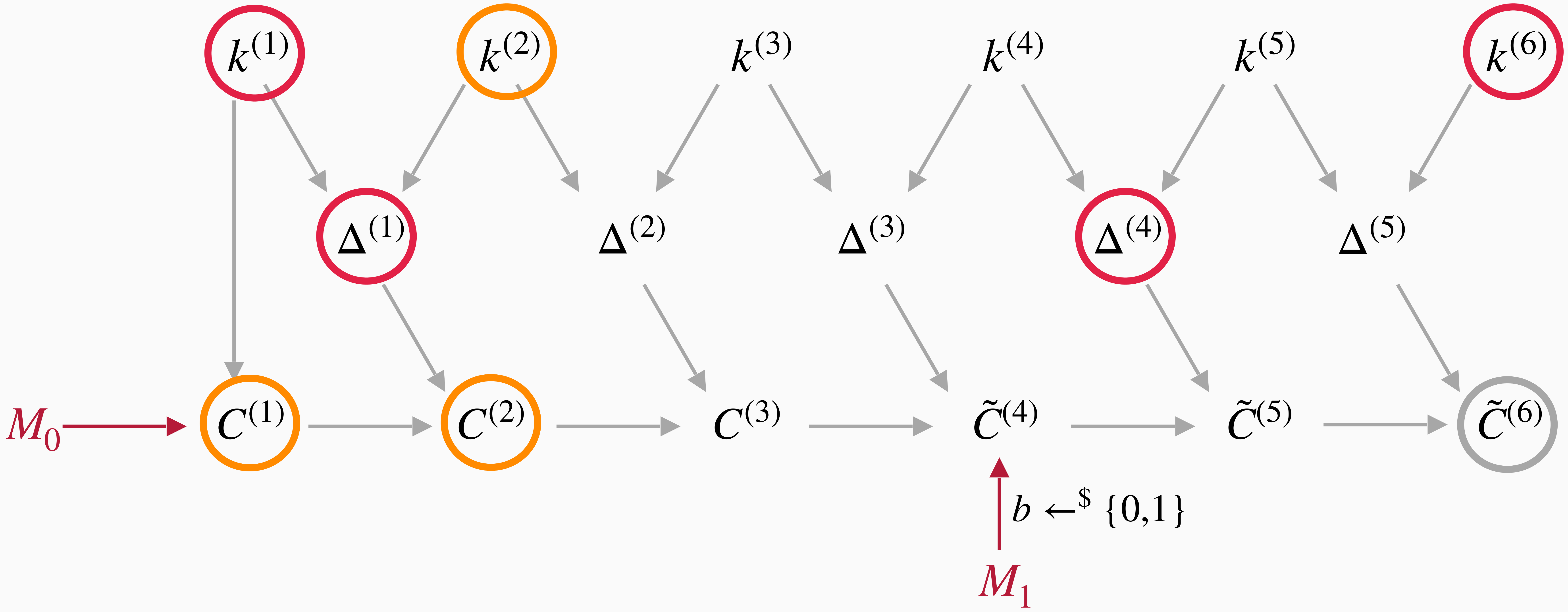
- Guess start and end of insulated regions (firewalls)
- Hybrid argument over those regions

CPA Security of BIN-UE



Firewall technique [C:BDGJ20]

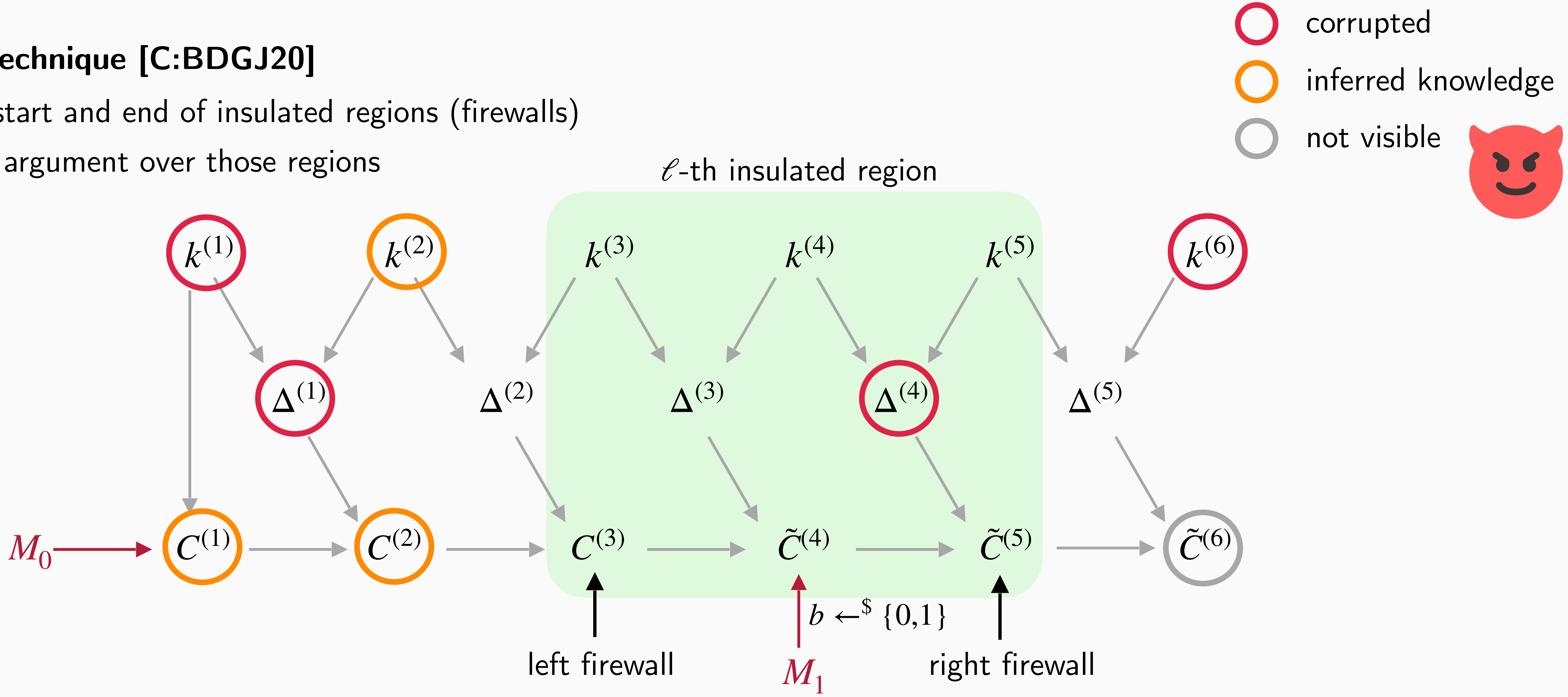
- Guess start and end of insulated regions (firewalls)
- Hybrid argument over those regions



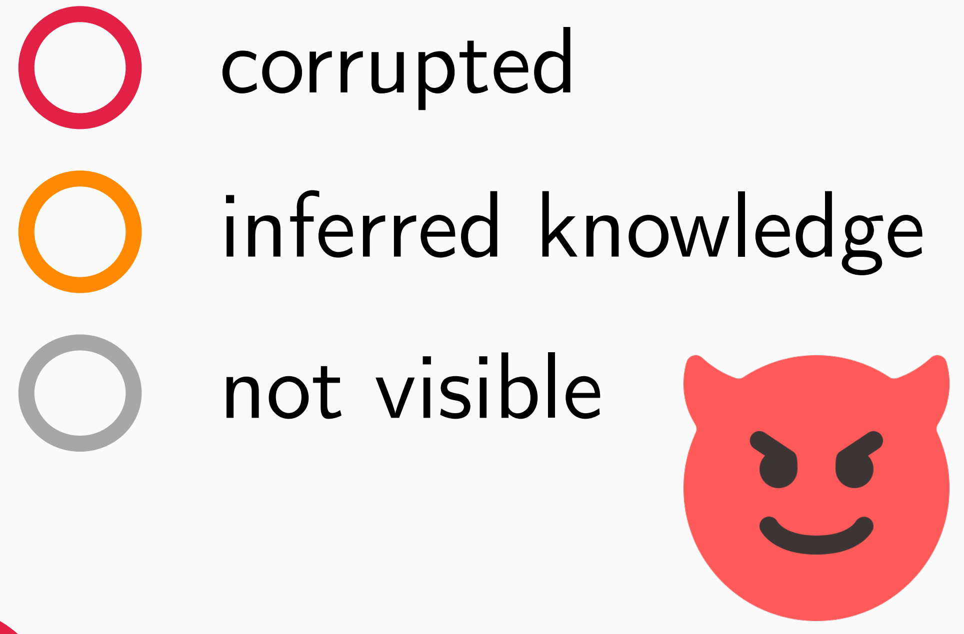
CPA Security of BIN-UE

Firewall technique [C:BDGJ20]

- Guess start and end of insulated regions (firewalls)
- Hybrid argument over those regions

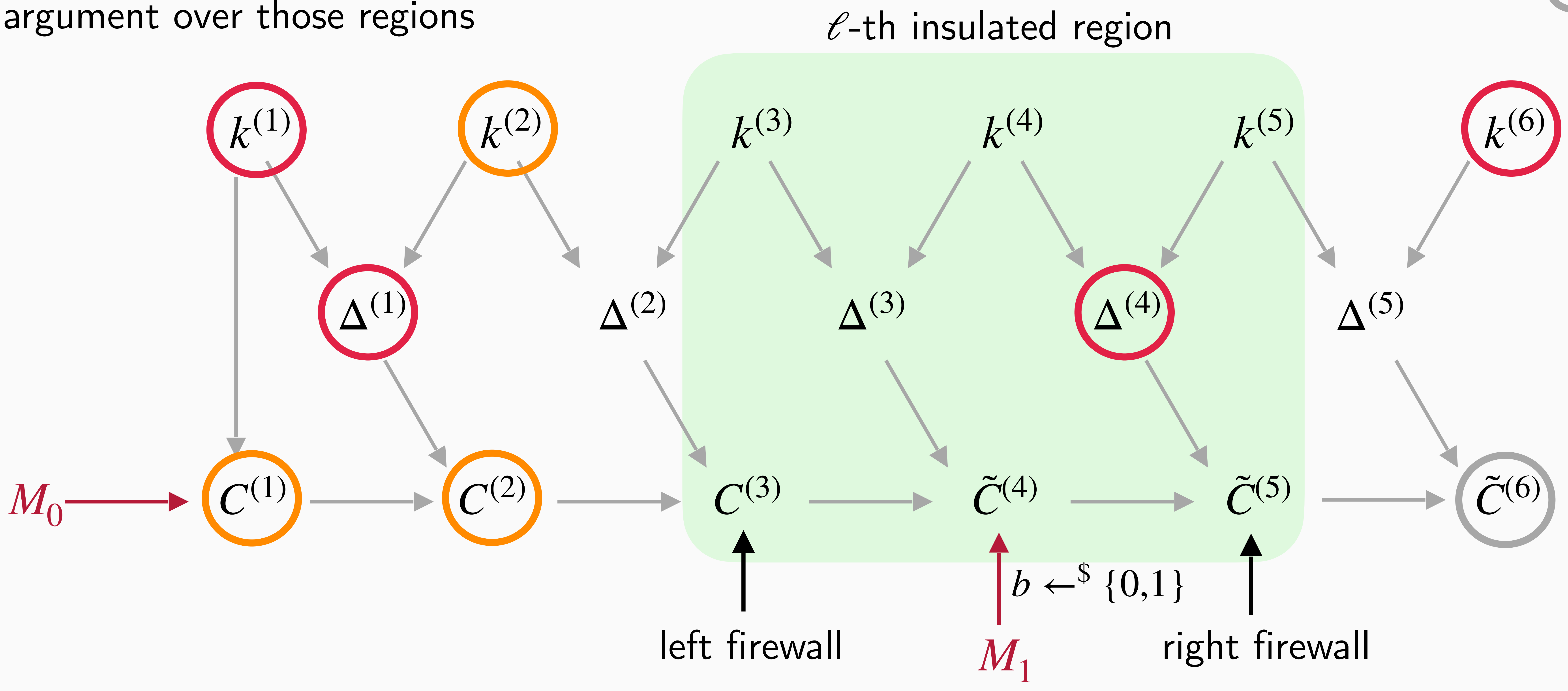


CPA Security of BIN-UE



Firewall technique [C:BDGJ20]

- Guess start and end of insulated regions (firewalls)
- Hybrid argument over those regions



Goal: replace $\tilde{C}_i^{(j)} = k_i^{(j)} \star x_{m_{b,i}}$ inside insulated regions with random elements from $\mathcal{X}$

- Use (multi-instance) group action DDH: given $(x, x_b, k \star x, u \star x_b)$, decide whether $u = k$ or random

Scheme 2: COM-UE

Observations

- BIN-UE (as most other UE schemes) is malleable
- It is randomness-recoverable and randomness-preserving

  $\Rightarrow$ x_0, x_1 are available to an adversary in the security game

Scheme 2: COM-UE

Observations

- BIN-UE (as most other UE schemes) is malleable
- It is randomness-recoverable and randomness-preserving

  $\Rightarrow$ x_0, x_1 are available to an adversary in the security game

COM-UE: **Tag-then-Encrypt**

- We define encryption as BIN-UE.Enc$(k, M \| T; r)$, where
  - $T = \mathsf{H}(M, r)$ using hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$
  - $r = (x_0, x_1)$ is the encryption randomness

Scheme 2: COM-UE

Observations

- BIN-UE (as most other UE schemes) is malleable
- It is randomness-recoverable and randomness-preserving

  $\Rightarrow$ x_0, x_1 are available to an adversary in the security game

COM-UE: **Tag-then-Encrypt**

- We define encryption as BIN-UE.Enc$(k, M \| T; r)$, where
  - $T = \mathsf{H}(M, r)$ using hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$
  - $r = (x_0, x_1)$ is the encryption randomness

Encryption

$$\begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \\ \hline k_{n+1} \\ \vdots \\ k_{n+\ell} \end{pmatrix} \star \begin{pmatrix} x_{m_1} \\ x_{m_2} \\ \vdots \\ x_{m_n} \\ \hline x_{t_1} \\ \vdots \\ x_{t_\ell} \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \\ \hline C_{n+1} \\ \vdots \\ C_{n+\ell} \end{pmatrix}$$

CCA Security of COM-UE

Ciphertext Integrity

- Should be hard to forge valid ciphertext without knowledge of k
- COM-UE binds the encryption randomness and message to the ciphertext using H to prevent malleability

CCA Security of COM-UE

Ciphertext Integrity

- Should be hard to forge valid ciphertext without knowledge of k
- COM-UE binds the encryption randomness and message to the ciphertext using H to prevent malleability

[C:BDGJ20]: IND-UE-CPA + INT-CTXT $\Rightarrow$ IND-UE-CCA

CCA Security of COM-UE

Ciphertext Integrity

- Should be hard to forge valid ciphertext without knowledge of k
- COM-UE binds the encryption randomness and message to the ciphertext using H to prevent malleability

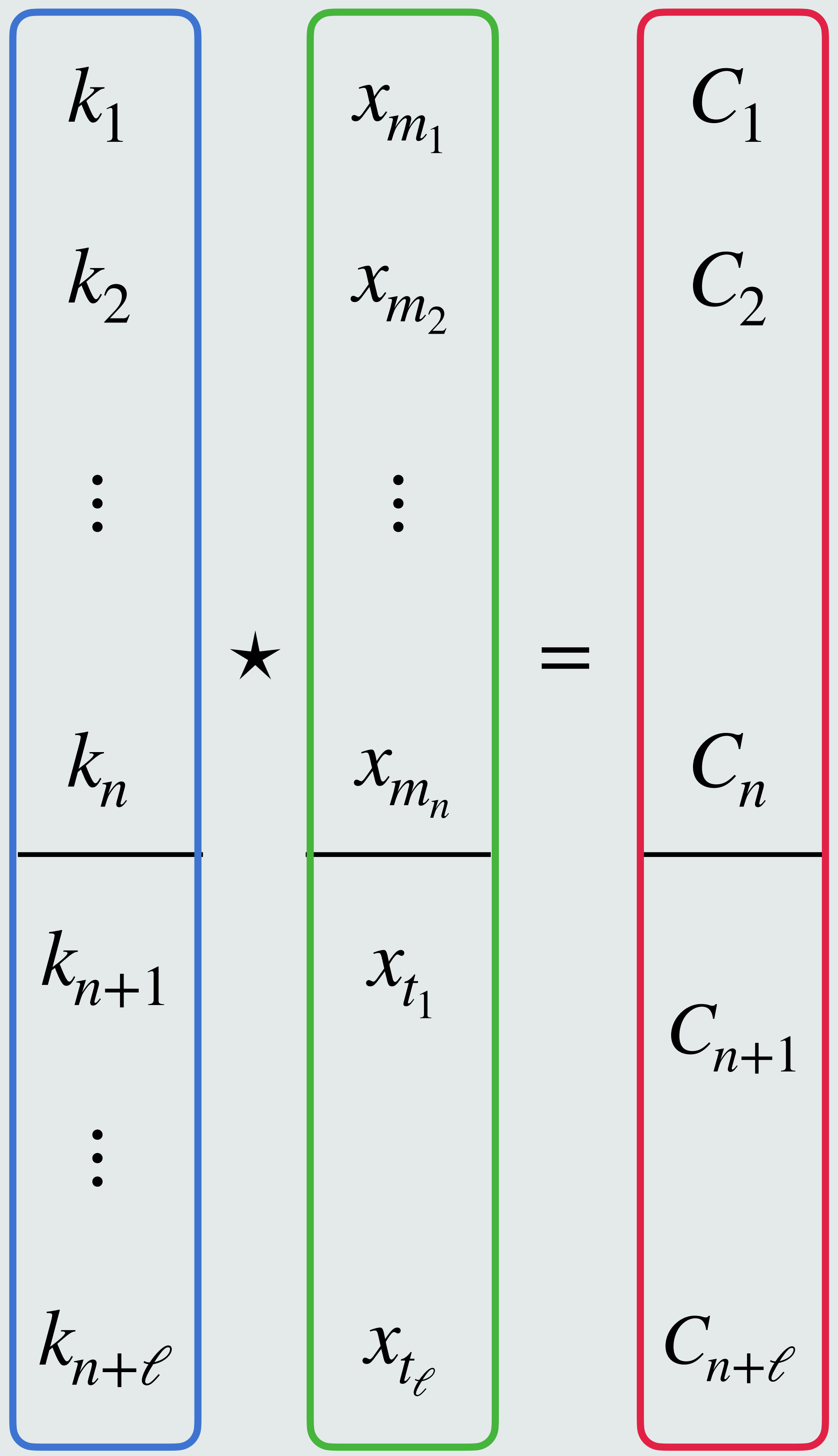
[C:BDGJ20]: $\underbrace{\textbf{IND-UE-CPA}}_{\text{same as for BIN-UE}} + \textbf{INT-CTXT} \Rightarrow \textbf{IND-UE-CCA}$

CCA Security of COM-UE

Tag-then-Encrypt

- We define encryption as BIN-UE.Enc$(k, M \| T; r)$, where $T = \mathsf{H}(M, r)$ using hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$ and encryption randomness $r = (x_0, x_1)$

Encryption



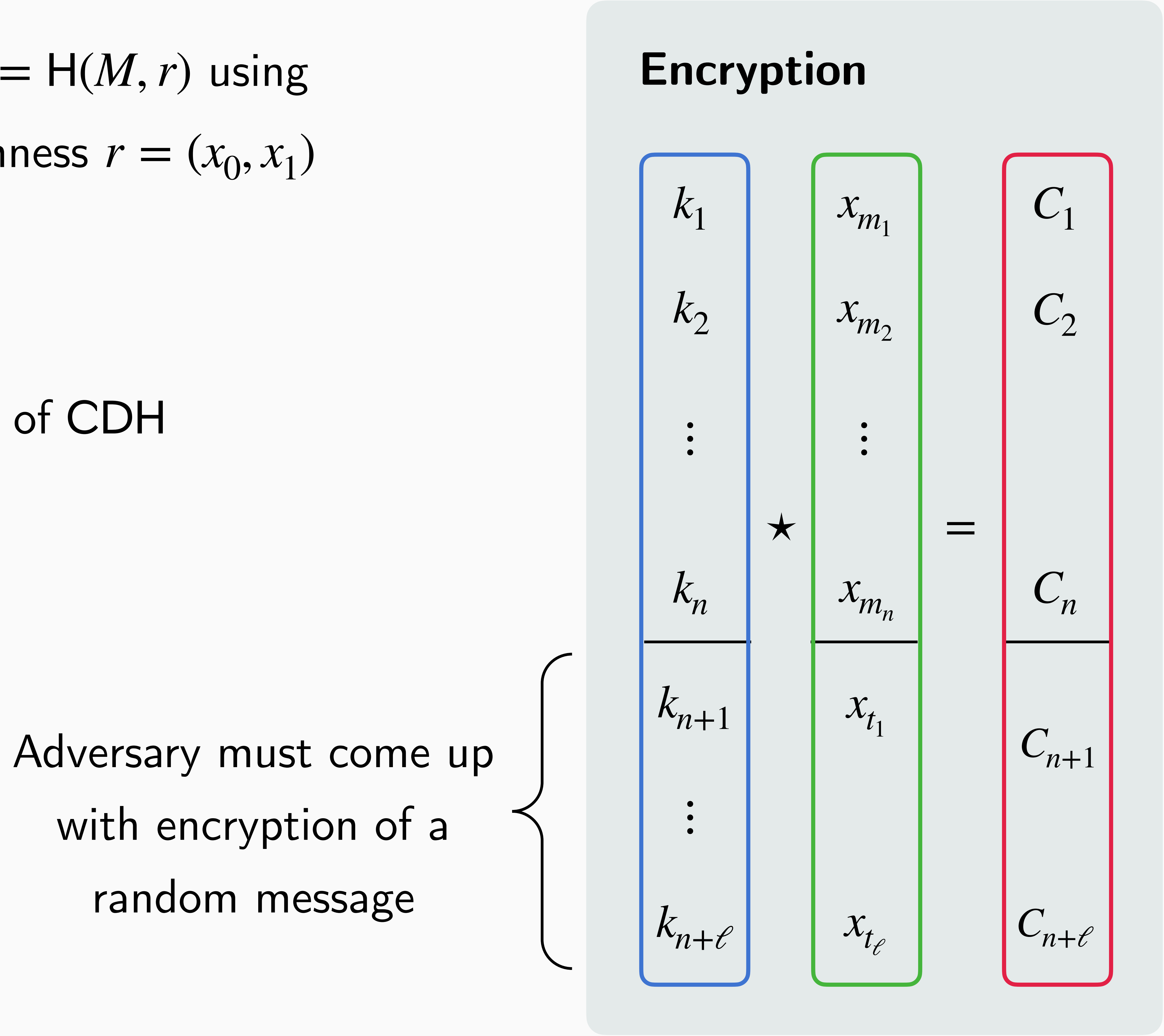
CCA Security of COM-UE

Tag-then-Encrypt

- We define encryption as BIN-UE.Enc$(k, M \| T; r)$, where $T = \mathsf{H}(M, r)$ using hash function $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ and encryption randomness $r = (x_0, x_1)$

Intuition for INT-CTXT

- Forging a ciphertext allows to solve a non-standard variant of CDH
- Embed the challenge by modeling H as a random oracle



Conclusion

Summary

- Updatable encryption from group actions requires some form of mappability
- Since CSIDH does not allow mapping into the set, we use a bit-wise approach
- BIN-UE achieves CPA security relying on (multi-instance) DDH for group actions
- COM-UE is the first CCA secure UE scheme based on post-quantum assumptions (in the algebraic/generic group action model)

Conclusion

Summary

- Updatable encryption from group actions requires some form of mappability
- Since CSIDH does not allow mapping into the set, we use a bit-wise approach
- BIN-UE achieves CPA security relying on (multi-instance) DDH for group actions
- COM-UE is the first CCA secure UE scheme based on post-quantum assumptions (in the algebraic/generic group action model)

ia.cr/2024/499

Conclusion

Summary

- Updatable encryption from group actions requires some form of mappability
- Since CSIDH does not allow mapping into the set, we use a bit-wise approach
- BIN-UE achieves CPA security relying on (multi-instance) DDH for group actions
- COM-UE is the first CCA secure UE scheme based on post-quantum assumptions (in the algebraic/generic group action model)

ia.cr/2024/499

Thank you!