# Data Pseudonymization & Key Management

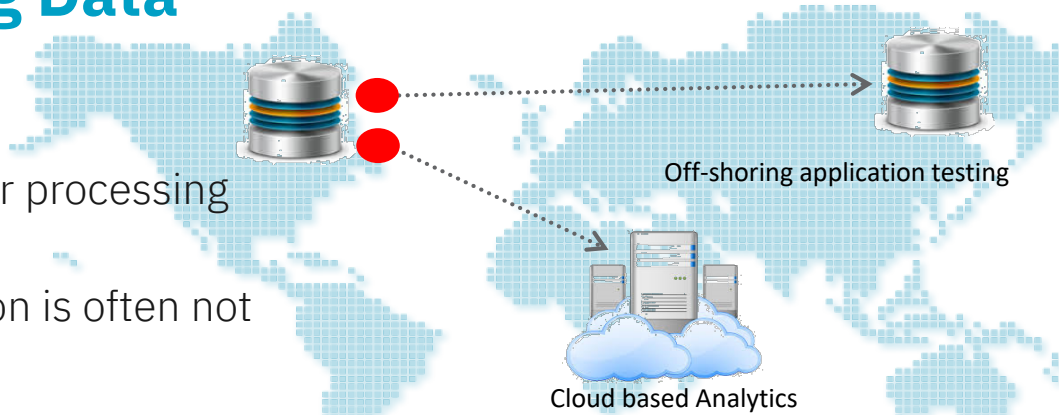
## From complex systems to simple solutions

Anja Lehmann

IBM Research – Zurich

IBM

# Motivation | De-Sensitizing Data




- Outsourcing of data for remote storage or processing
- Shipping of sensitive/personal information is often not desired or allowed
  - Personal data cannot be moved across borders or used outside of original purpose (GDPR!)
  - Cloud environment / external service provider are not fully trusted
- How to protect sensitive data & preserve utility ?

standard encryption “destroys” any structure & information

| Name | Post Code | Date of Birth | Balance | IBAN |
|---|---|---|---|---|
| Alice Doe | 64289 | 21.08.1978 | 52.650 | CH56 0483 5084 1385 0100 0 |
| Ks3tcqu15 | ahBd3js | o2eh2UAb | 9SXbjW3 | ny4uB5Na3dh7fWpik3yLFpk |

# Motivation | De-Sensitizing Data

This talk: pseudonymization

How to de-sensitize *primary* identifiers

- Utility-preserving de-sensitization

| Name | Post Code | Date of Birth | Balance | IBAN |
|---|---|---|---|---|
| Alice Doe | 64289 | 21.08.1978 | 52.650,77 | CH56 0483 5084 1385 0100 0 |
| | suppress ⇩ | generalize ⇩ | add noise ⇩ | |
| ?? | 64*** | 15.08.1978 | 53.012,62 | ?? |

- Many non-cryptographic approaches
  - Suppression, generalization, adding noise, swapping data

- Property-preserving encryption
  - Order-preserving / searchable / deterministic encryption
  - Preserved property comes for price of reduced (sometimes no) security!

# **Pseudonymization** (aka Tokenization)

- Pseudonymization = replacing a primary identifier with random-looking substitute
  - Mandated e.g., by Payment Card Industry Data Security Standard (PCI DSS)

| Name | Post Code | Date of Birth | Balance | IBAN |
|---|---|---|---|---|
| Alice Doe | 64289 | 21.08.1978 | 52.650,77 | CH56 0483 5084 1385 0100 0 |

- Requirements: pseudonymization must be
  - Consistent, i.e., referential integrity must be preserved
  - Dynamic, i.e., data can be pseudonymized on the fly
  - Secure, i.e., infeasible to determine **uid** from **nym**

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |

| UserID | Credit |
|---|---|
| Alice | € 599 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 599 |

# **Pseudonymization** (aka Tokenization)

- Pseudonymization = replacing a primary identifier with random-looking substitute
  - Mandated e.g., by Payment Card Industry Data Security Standard (PCI DSS)

| Name | Post Code | Date of Birth | Balance | IBAN |
|---|---|---|---|---|
| Alice Doe | 64289 | 21.08.1978 | 52.650,77 | CH56 0483 5084 1385 0100 0 |

- Requirements: pseudonymization must be
  - Consistent, i.e., referential integrity must be preserved **NO!**
  - Dynamic, i.e., data can be pseudonymized on the fly
  - Secure, i.e., infeasible to determine **uid** from **nym**

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |

| UserID | Credit |
|---|---|
| Alice | € 599 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 599 |

# Limitations of Standard Pseudonymization

- **Linkability is a privacy risk** – inference attacks allow re-identification

| Nym | Work |
|---|---|
| xH2ban6 | IBM Research |

| Nym | Education |
|---|---|
| xH2ban6 | PhD Cryptography |

| Nym | Nationality |
|---|---|
| xH2ban6 | Germany |

- Using context-specific pseudonyms better for privacy – but restrict usability
  - Decision which data is linkable upon pseudonymization
  - Unlinkable data cannot be linked afterwards → risk of losing too much information

| Nym | Work |
|---|---|
| xH2ban6 | IBM Research |

| Nym | Education |
|---|---|
| xH2ban6 | PhD Cryptography |

| Nym | Nationality |
|---|---|
| 75jQyl0 | Germany |


| Nym | Date of Birth |
|---|---|
| 75jQyl0 | 22.06.1981 |

Business | Private

# **Chameleon Pseudonyms |** Unlinkable Data Storage

| UserID | Date of Birth | Gender | Nationality |
|---|---|---|---|
| Alice | 22.06.1981 | female | Germany |
| Bob | 08.11.1976 | male | Italy |

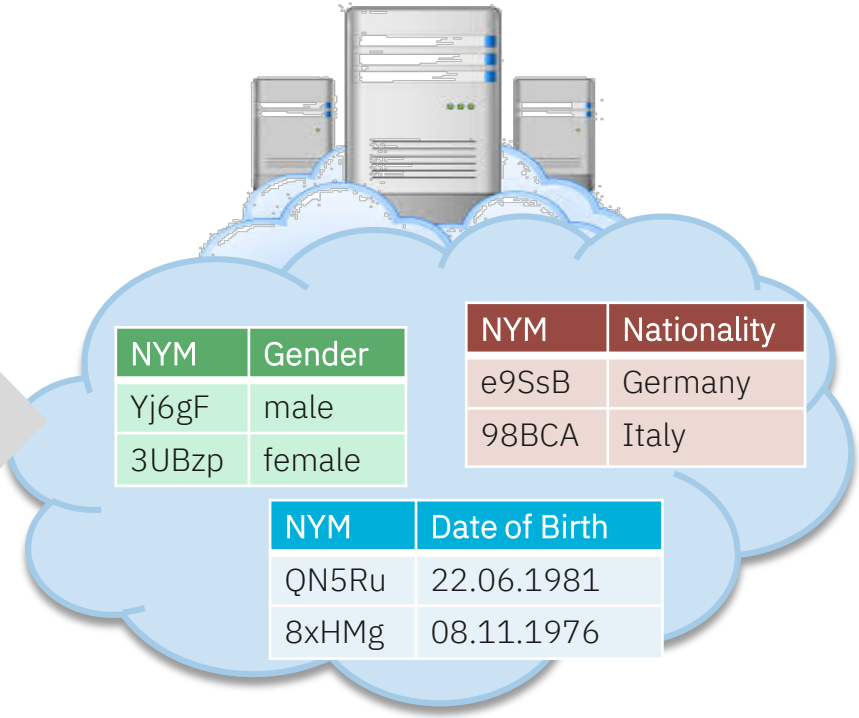
Converter

Pseudonym generation can be done **blindly**

| NYM | Date of Birth |
|---|---|
| QN5Ru | 22.06.1981 |

| NYM | Gender |
|---|---|
| 3UBzp | female |

| NYM | Nationality |
|---|---|
| e9SsB | Germany |



| NYM | Gender |
|---|---|
| Yj6gF | male |
| 3UBzp | female |

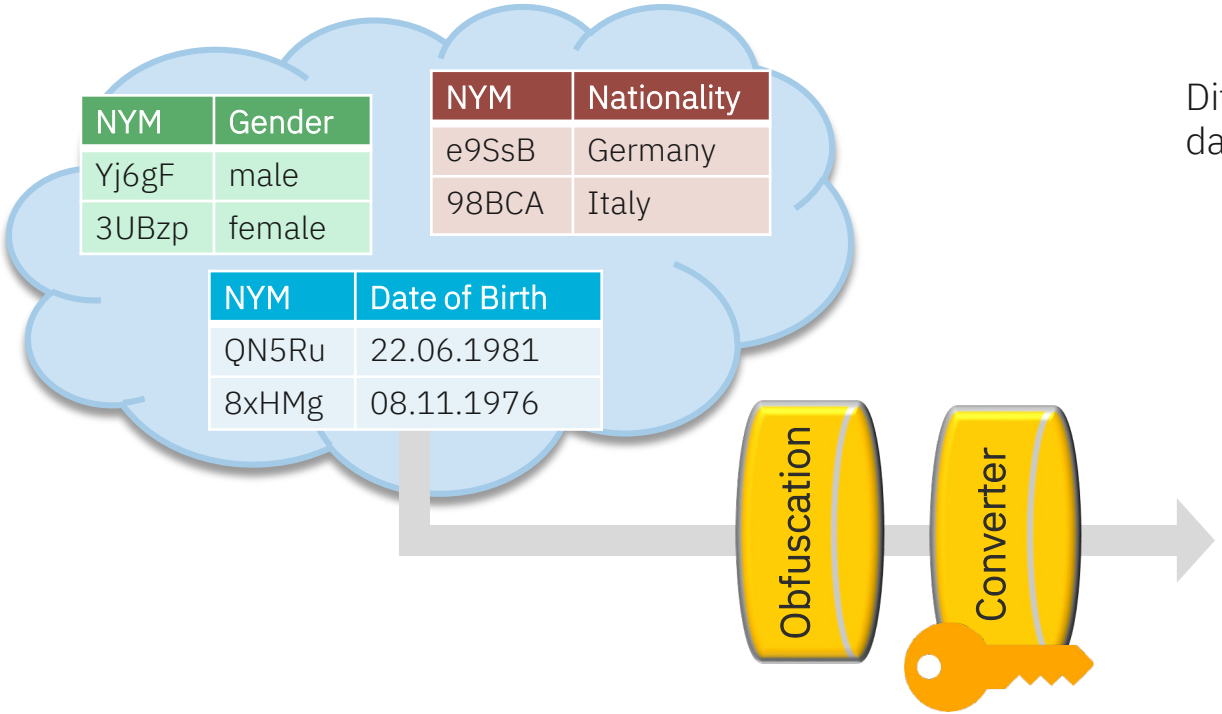
| NYM | Nationality |
|---|---|
| e9SsB | Germany |
| 98BCA | Italy |

| NYM | Date of Birth |
|---|---|
| QN5Ru | 22.06.1981 |
| 8xHMg | 08.11.1976 |

Data is stored in unlinkable data snippets

# **Chameleon Pseudonyms |** Controlled Linkability

Only required sub-sets of the data are made linkable w.r.t. to application-specific pseudonym




Conversion can be done **blindly**

Different subsets remain unlinkable & data can be obfuscated before join

query: age & EU nationality

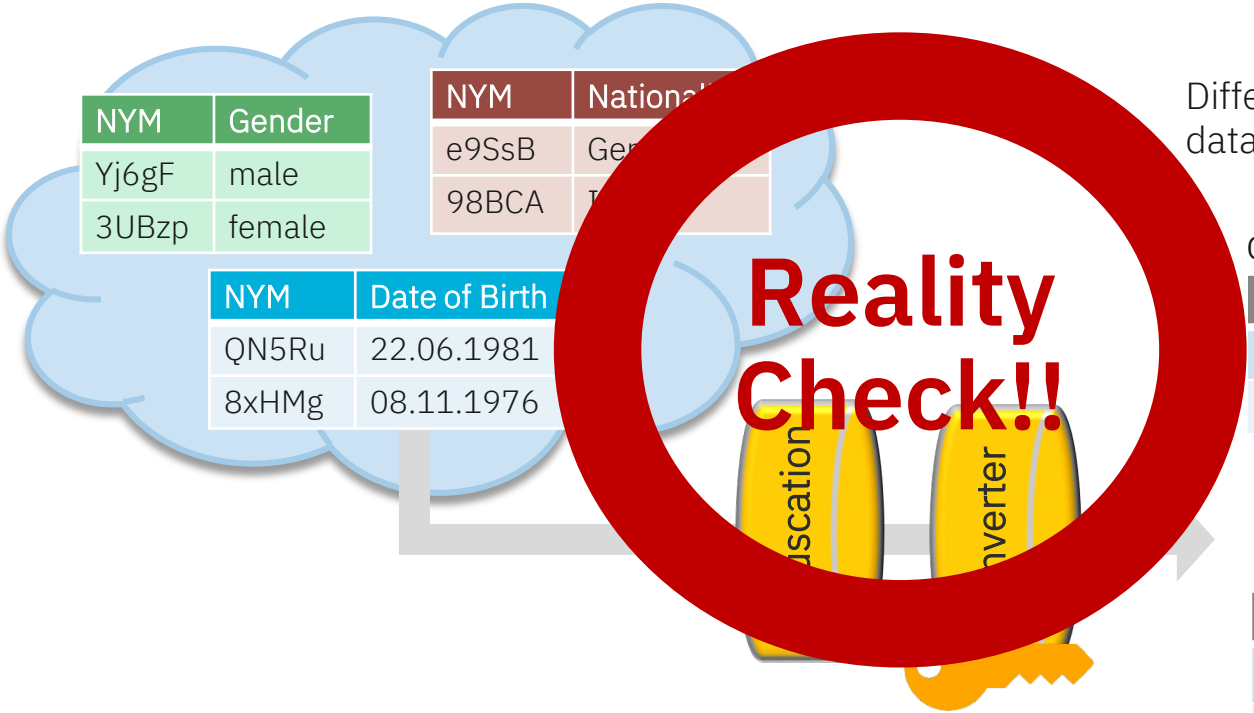
| NYM | Date of Birth | Nationality |
|---|---|---|
| GDA12 | **.**.1976 | EU |
| 0tU5r | **.**.1981 | EU |

query: gender & nationality

| NYM | Gender | Nationality |
|---|---|---|
| 4T3gq | female | Germany |
| kOLc6 | male | Italy |

# **Chameleon Pseudonyms |** Controlled Linkability

Only required sub-sets of the data are made linkable w.r.t. to application-specific pseudonym




Different subsets remain unlinkable & data can be obfuscated before join

query: age & EU nationality

| NYM | Date of Birth | Nationality |
|---|---|---|
| GDA12 | **.**.1976 | EU |
| 0tU5r | **.**.1981 | EU |

query: gender & nationality

| NYM | Gender | Nationality |
|---|---|---|
| 4T3gq | female | Germany |
| kOLc6 | male | Italy |

Conversion can be done **blindly**

# **Pseudonymization |** Current Solution

- One-way functions, e.g. Hash function $H(uid) \rightarrow nym$
  - Often reversibility is undesirable (stronger regulations apply for re-identifiable data)


  - Unkeyed functions vulnerable to offline attacks:

$$H(Bob) = nym\ ?, H(Eve) = nym\ ?, \ldots$$

  - Pseudonymization must be based on strong secret key, e.g., $H(key, uid) \rightarrow nym$

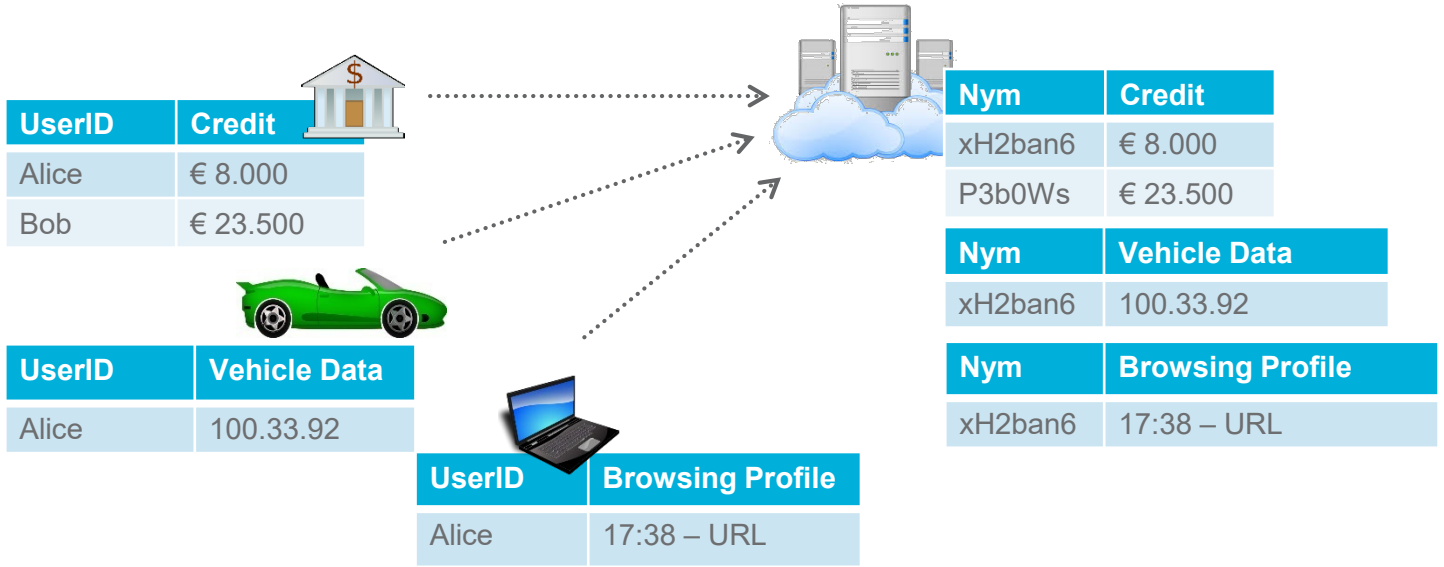
In theory: „Let K be a key“ In practice: Well, its not that easy ....

# ROADMAP

- Key Management for Pseudonymization
  - Oblivious Pseudonymization-as-a-Service
  - Key Rotation
- Updatable Encryption

# Distributed Pseudonymization & Consistency

- Pseudonymization from multiple sources while preserving consistency

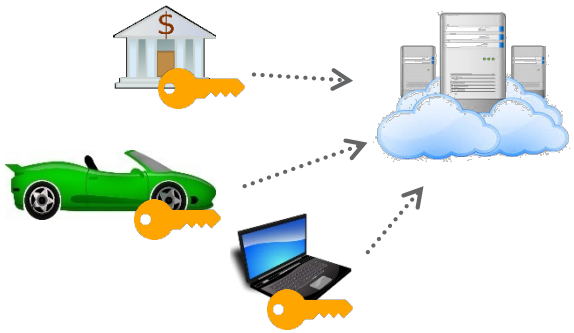



How to ensure consistent and secure pseudonymization when data is pushed by many, diverse entities ?

# Distributed Pseudonymization | Existing Solutions

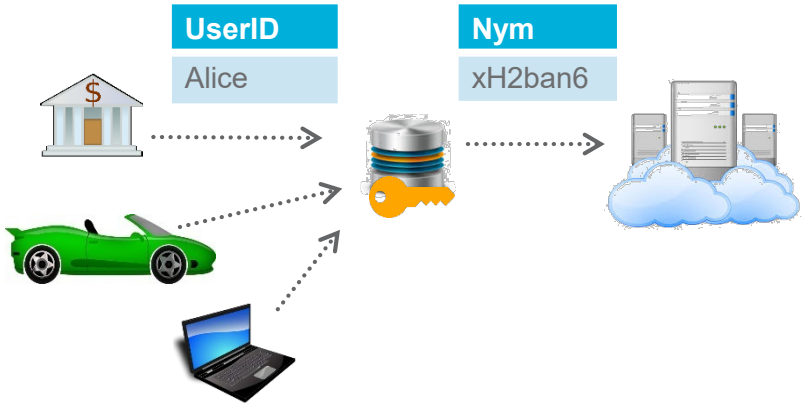
- Secure pseudonymization requires strong cryptographic keys

Key is distributed to all entities



- Replication of keys is security issue
- HSMs everywhere too expensive
- Key updates virtually impossible

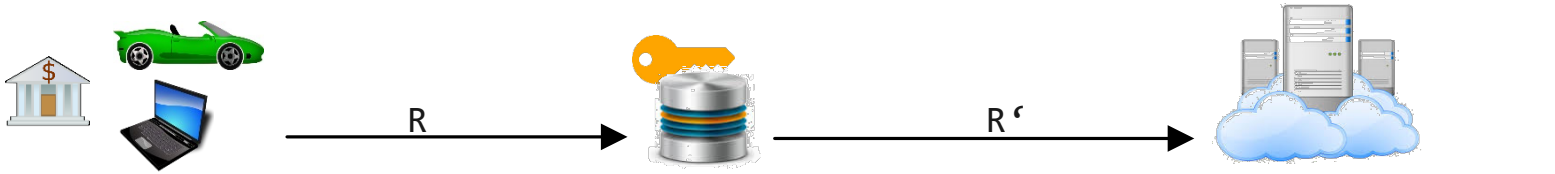
Central service (TTP)




- TTP knows the relation between UID & Nym
- TTP learns all metadata – which user interacts when & where & with whom
- TTP must be fully trusted → privacy risk itself!

# *Oblivious* Pseudonymization-as-a-Service

- Central OPaaS creates pseudonyms, but without becoming a privacy risk:




$$R = H(uid)^N \quad \text{for random nonce } N$$

$$R' = R^{key}$$

$$nym = R'^{\frac{1}{N}} = H(uid)^{key}$$

- Oblivious OPaaS blindly computes cryptographic pseudonyms:
  - OPaaS doesn‘t learn anything about input nor output
  - OPaaS generates pseudonyms in deterministic manner
- Simple key management – only OPaaS has to store a single secret key
- Compatible with key-update solution

# ROADMAP

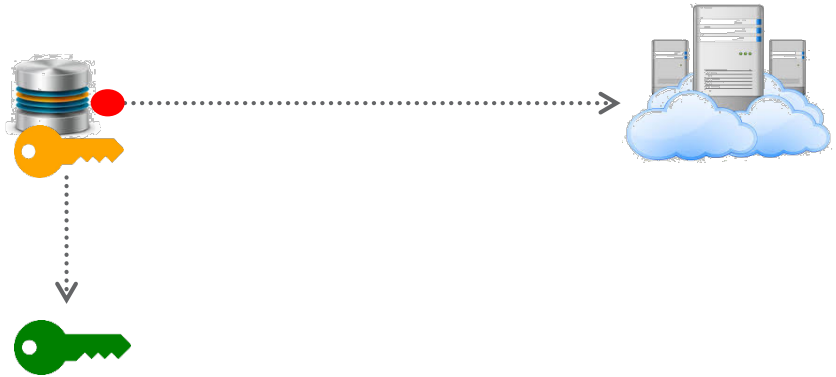
- Key Management for Pseudonymization
  - Oblivious Pseudonymization-as-a-Service
  - Key Rotation
- Updatable Encryption

# **Pseudonymization |** Key Rotation

- Key often must have high availability due to *dynamic* pseudonymization → security risk

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |

| UserID | Credit |
|---|---|
| Alice | € 599 |



| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 599 |

- Proactive security by periodically changing the secret key
  - Key rotation reduces risk & impact of key or data exposure
  - After key rotation any previously exposed data gets useless

- Key rotation often mandated in high-security environments and by PCI DSS

# **Pseudonymization |** Key Rotation

- Key update inherently destroys consistency

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |

| UserID | Credit |
|---|---|
| Alice | € 599 |

| UserID | Credit |
|---|---|
| Alice | € 125 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 599 |

| Nym | Credit |
|---|---|
| JqNs58k | € 125 |

- Outsourced data needs to be updated as well to ensure consistency after key update

- Existing solutions require “re-pseudonymization” of all data
  → Inefficient (download/local copy of all data, key often protected by HSM, …)

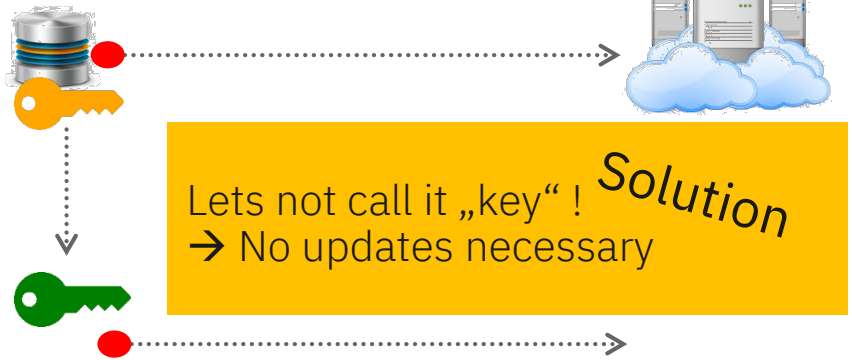
# Pseudonymization | Key Rotation

- Key update inherently destroys consistency

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |

| UserID | Credit |
|---|---|
| Alice | € 599 |

| UserID | Credit |
|---|---|
| Alice | € 125 |



Lets not call it „key“ !
→ No updates necessary

Solution

| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 599 |

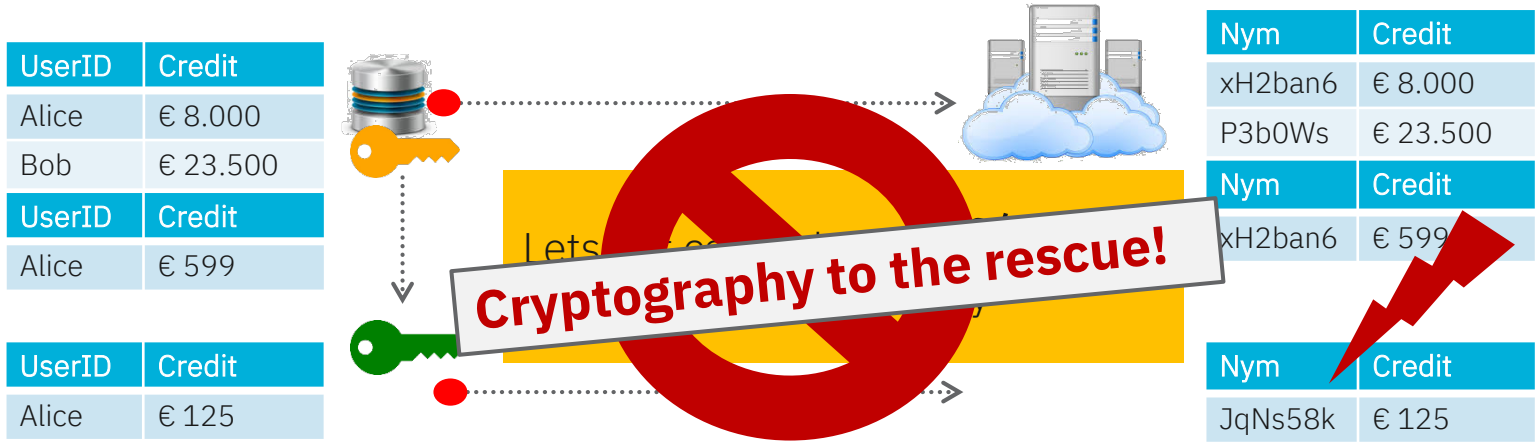
| Nym | Credit |
|---|---|
| JqNs58k | € 125 |

- Outsourced data needs to be updated as well to ensure consistency after key update

- Existing solutions require “re-pseudonymization” of all data
  → Inefficient (download/local copy of all data, key often protected by HSM, …)

# Pseudonymization | Key Rotation

- Key update inherently destroys consistency




- Outsourced data needs to be updated as well to ensure consistency after key update

- Existing solutions require “re-pseudonymization” of all data
  → Inefficient (download/local copy of all data, key often protected by HSM, …)

# Key-Evolving Pseudonymization

- Consistency-preserving updates

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |

| UserID | Credit |
|---|---|
| Alice | € 599 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 599 |

Key update generates
key & update tweak

# Key-Evolving Pseudonymization | Nym Updates

- Consistency-preserving updates

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |
| UserID | Credit |
| Alice | € 599 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |
| Nym | Credit |
| xH2ban6 | € 599 |

Tweak updates pseudonyms into new key epoch

| Nym | Credit |
|---|---|
| JqNs58k | € 8.000 |
| 4GLu8W | € 23.500 |
| Nym | Credit |
| JqNs58k | € 599 |

| UserID | Credit |
|---|---|
| Alice | € 125 |

Updated & newly computed pseudonyms are consistent

| Nym | Credit |
|---|---|
| JqNs58k | € 125 |

# Key-Evolving Pseudonymization | Nym Updates

- Consistency-preserving updates

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |

| UserID | Credit |
|---|---|
| Alice | € 599 |

$$nym = H(uid)^{key}$$

| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 599 |

Tweak updates pseudonyms into new key epoch

$$\Delta = \frac{key'}{key}$$

| Nym | Credit |
|---|---|
| JqNs58k | € 8.000 |
| JqNs58k | € 599 |

$$nym' = nym^{\Delta} = H(uid)^{key'}$$

| UserID | Credit |
|---|---|
| Alice | € 125 |

$$nym' = H(uid)^{key'}$$

ted pseudonyms are consistent

| Nym | Credit |
|---|---|
| JqNs58k | € 125 |



[CCFL17] Updatable Tokenization: Formal Definitions and Provably Secure Constructions, FC 2017

# Key-Evolving Pseudonymization | Nym Updates

- Consistency-preserving updates

| UserID | Credit |
|---|---|
| Alice | € 8.000 |
| Bob | € 23.500 |

| UserID | Credit |
|---|---|
| Alice | € 599 |

$$nym = H(uid)^{key}$$

| Nym | Credit |
|---|---|
| xH2ban6 | € 8.000 |
| P3b0Ws | € 23.500 |

| Nym | Credit |
|---|---|
| xH2ban6 | € 599 |

Tweak updates pseudonyms into new pseudonyms

$$\Delta = \frac{key'}{key}$$

- Security model for key updates & adaptive corruptions
- Provably secure schemes

| Nym | Credit |
|---|---|
| JqNs58k | € 8.000 |
| JqNs58k | € 599 |

$$nym' = nym^{\Delta} = H(uid)^{key'}$$

| UserID | Credit |
|---|---|
| Alice | € 125 |

$$nym' = H(uid)^{key'}$$

ted pseudonyms are consistent

| Nym | Credit |
|---|---|
| JqNs58k | € 125 |



[CCFL17] Updatable Tokenization: Formal Definitions and Provably Secure Constructions, FC 2017

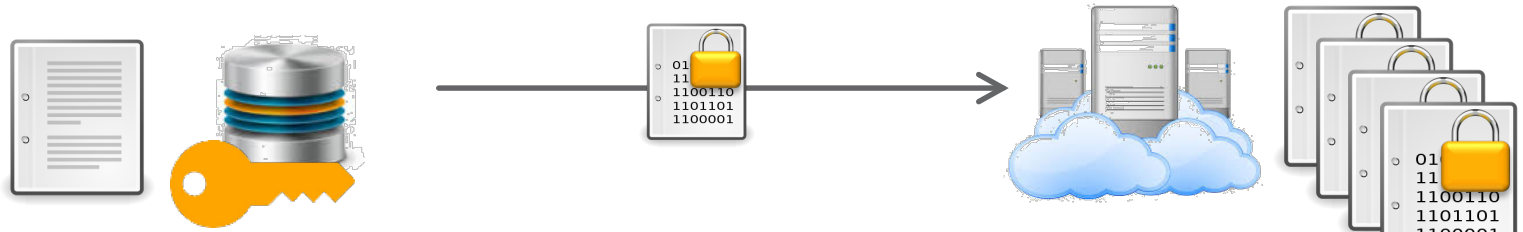
# ROADMAP

- Key Management for Pseudonymization
  - Oblivious Pseudonymization-as-a-Service
  - Key Rotation
- Updatable Encryption

  [LT18] Updatable Encryption with Post-Compromise Security, EC 2018

# **Motivation |** Outsourced Storage

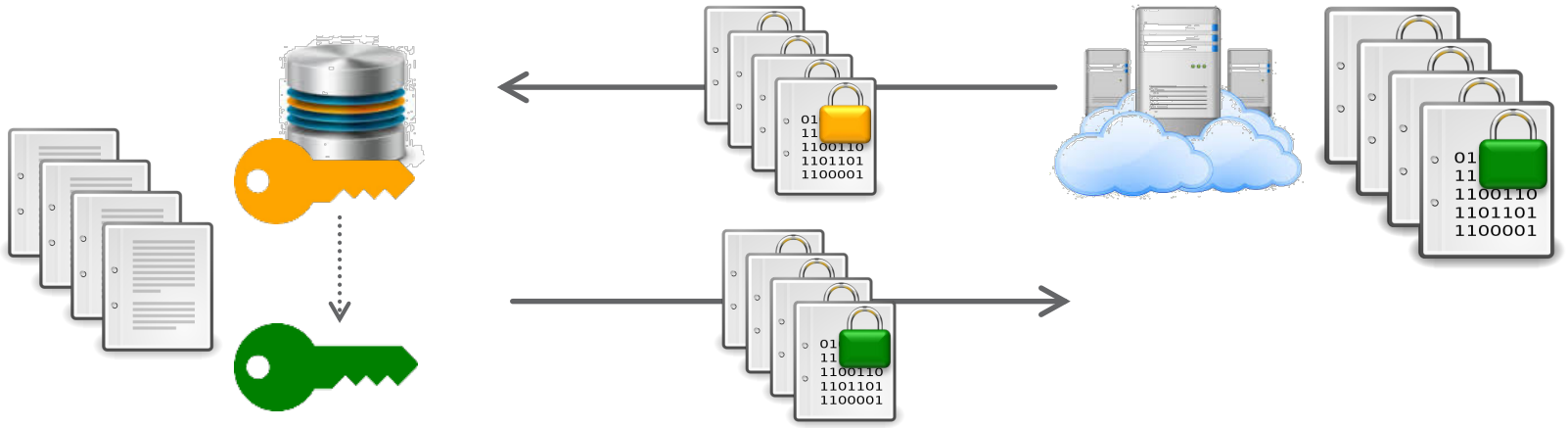
- Data owner stores encrypted data at (untrusted) data host




- Proactive security by periodically changing the secret key
  - Key rotation reduces risk & impact of key or data exposure
- Key rotation often mandated in high-security environments and by PCI DSS

# **Motivation |** Key Rotation

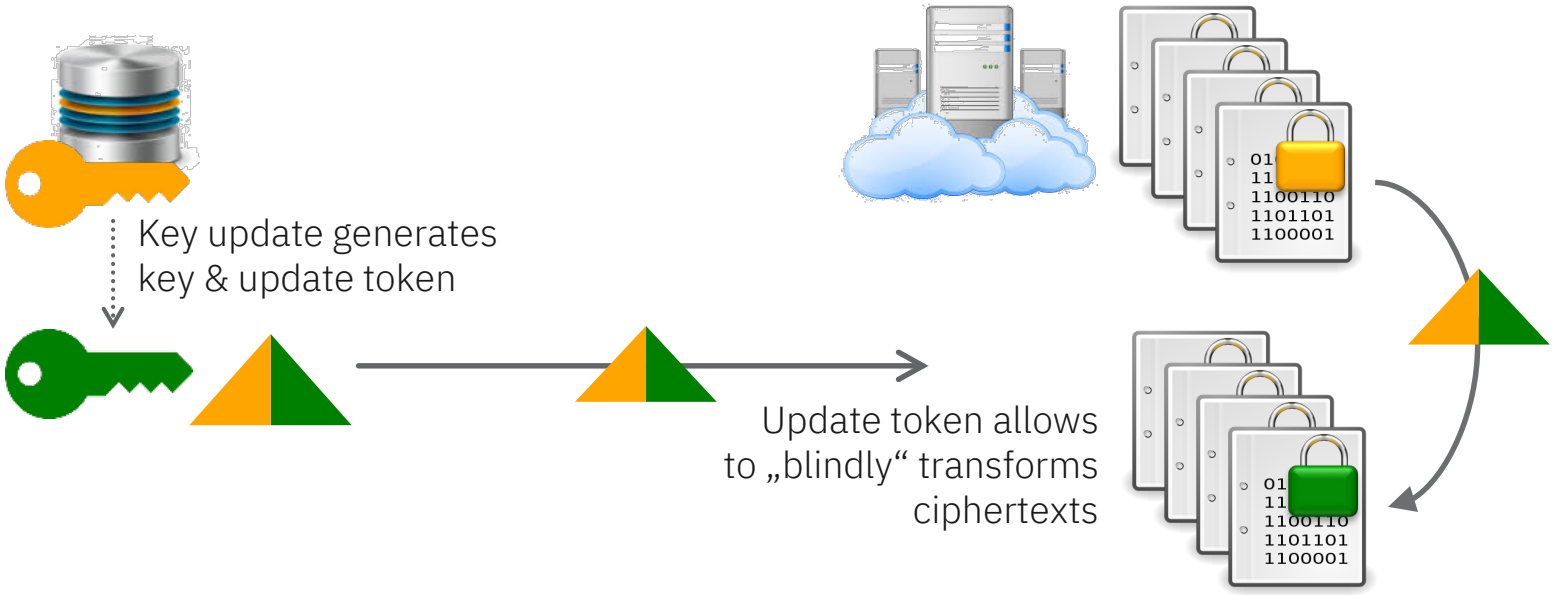
- How to update exiting ciphertexts to the new key?




- Standard symmetric encryption → download all ciphertext & re-encrypt from scratch
- Inefficient: down&upload of all ciphertexts, symmetric key often protected by hardware

# **Motivation |** Updatable Encryption

- Proposed by Boneh et al. [BLMR13]: ciphertexts can be updated w/o secret key




- Update operation of ciphertexts is shifted to (untrusted) data host w/o harming security

# **Updatable Encryption |** State-of-the-Art

Ciphertext-Independent

$\mathrm{UE.setup}(\lambda) \to k_0$
$\mathrm{UE.enc}(k_e, m) \to C_e$
$\mathrm{UE.dec}(k_e, C_e) \to m$

$\mathrm{UE.next}(k_e) \to (k_{e+1}, \Delta_{e+1})$
$\mathrm{UE.upd}(\Delta_{e+1,}\, C_e) \to C_{e+1}$

- BLMR13: high level idea & scheme, no security definitions

- **Our work: formal definitions & secure schemes for ciphertext-independent setting**

Ciphertext-Dependent

$\mathrm{UE.setup}(\lambda) \to k_0$
$\mathrm{UE.enc}(k_e, m) \to C_e$
$\mathrm{UE.dec}(k_e, C_e) \to m$

$\mathrm{UE.next}(k_e) \to k_{e+1}$
$\mathrm{UE.token}(k_e, k_{e+1}, C_e) \to \Delta_{C,e+1}$
$\mathrm{UE.upd}(\Delta_{C,e+1,}\, C_e) \to C_{e+1}$

- BLMR15: partial definitions & new scheme

# Updatable Encryption | State-of-the-Art

Ciphertext-Independent

$\mathrm{UE.setup}(\lambda) \to k_0$

$\mathrm{UE.enc}(k_e, m) \to C_e$

$\mathrm{UE.dec}(k_e, C_e) \to m$

$\mathrm{UE.next}(k_e) \to (k_{e+1}, \Delta_{e+1})$

$\mathrm{UE.upd}(\Delta_{e+1}, C_e) \to C_{e+1}$

- Fine-grained control of updates
- Less efficient: requires down&upload of (parts of) all ciphertexts & one token generation per ciphertext
- Less convenient: update requires coordination

Ciphertext-Dependent

$\mathrm{UE.setup}(\lambda) \to k_0$

$\mathrm{UE.enc}(k_e, m) \to C_e$

$\mathrm{UE.dec}(k_e, C_e) \to m$

$\mathrm{UE.next}(k_e) \to k_{e+1}$

$\mathrm{UE.token}(k_e, k_{e+1}, C_e) \to \Delta_{C,e+1}$

$\mathrm{UE.upd}(\Delta_{C,e+1}, C_e) \to C_{e+1}$

- BLMR13: high level idea & scheme, no security definitions
- BLMR15: partial definitions & new scheme
- **Our work: formal definitions & secure schemes for ciphertext-independent setting**

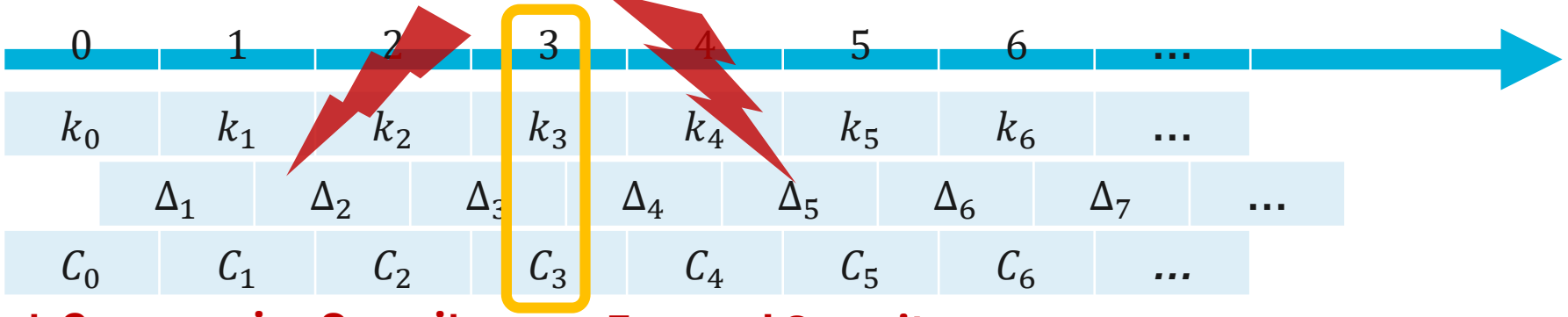
# **Updatable Encryption |** Sequential Setting


$k_0$

$C_0$

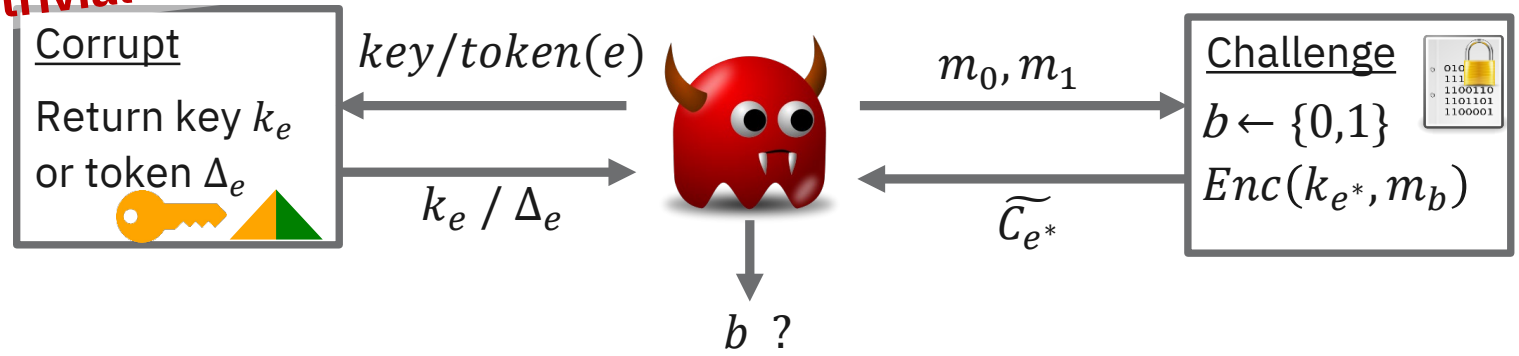
- Our work: strictly sequential setting
- Previous works: adaptions of proxy re-encryption definition
  - Allows re-encryptions across arbitrary epochs (back & forward)
  - No notion of time → hard to grasp *when* key corruptions are allowed

# Updatable Encryption | Security

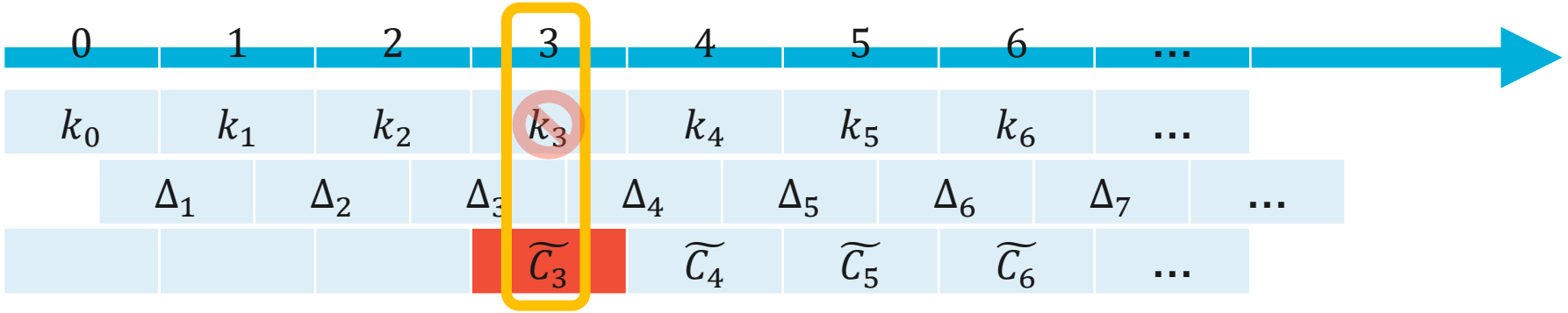



**Post-Compromise Security + Forward Security = IND-ENC**

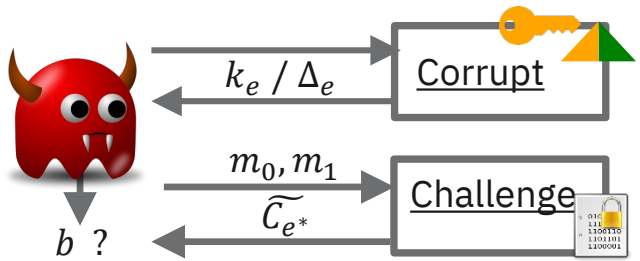
**No "trivial" corruptions**



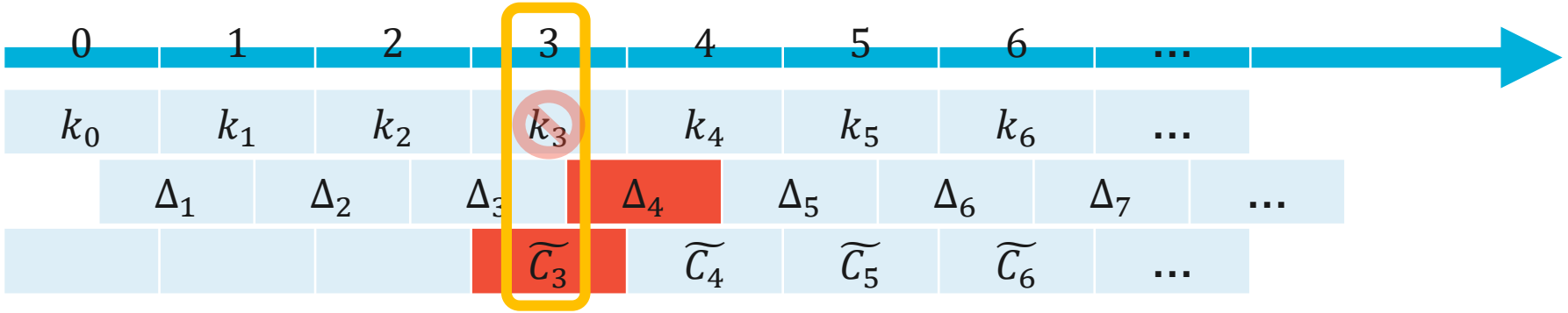
# Updatable Encryption | Capturing Trivial Wins



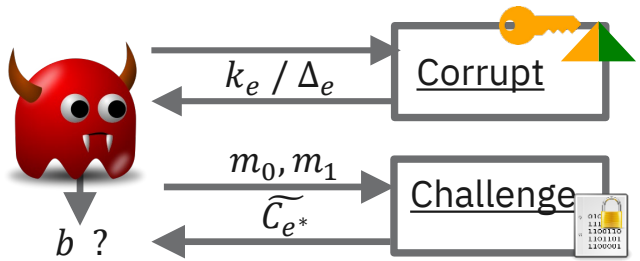

- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates



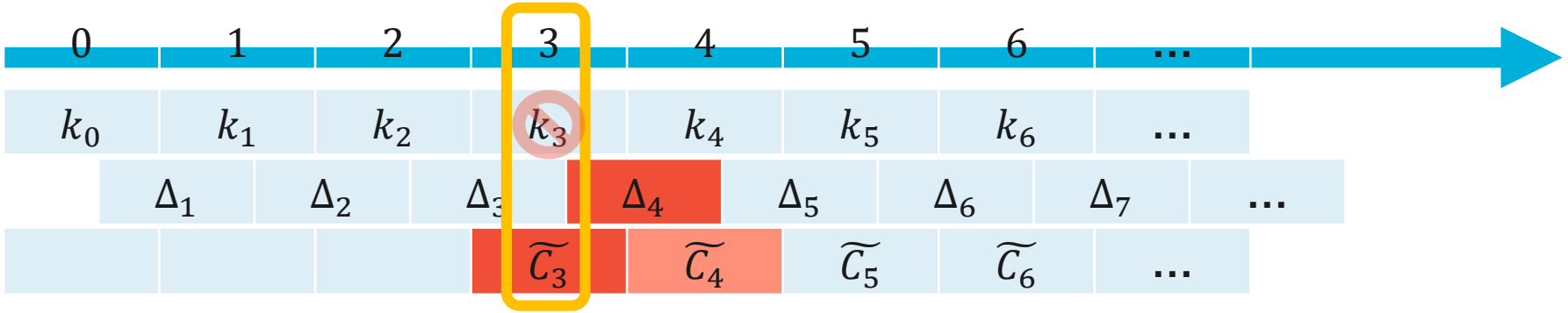
# Updatable Encryption | Capturing Trivial Wins



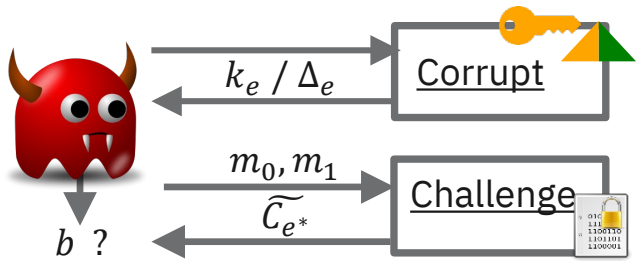

- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates



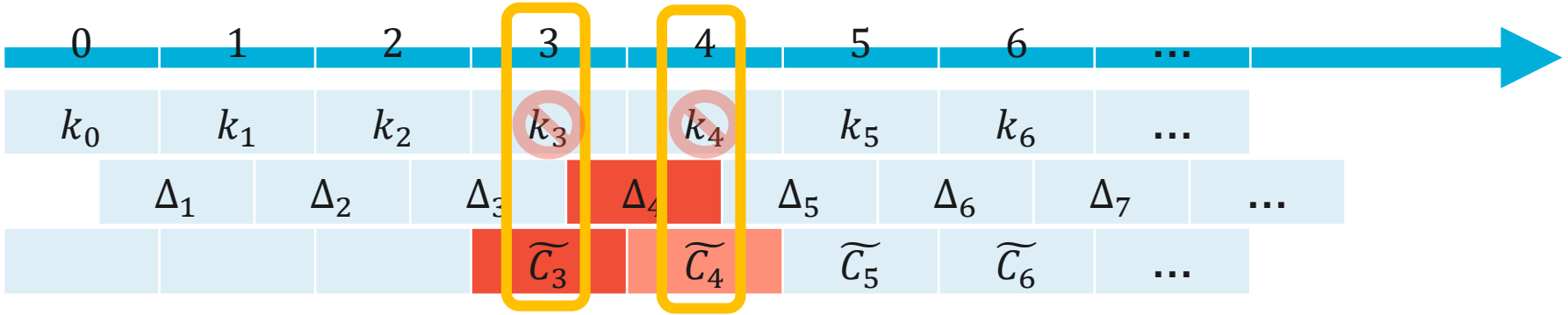
# Updatable Encryption | Capturing Trivial Wins



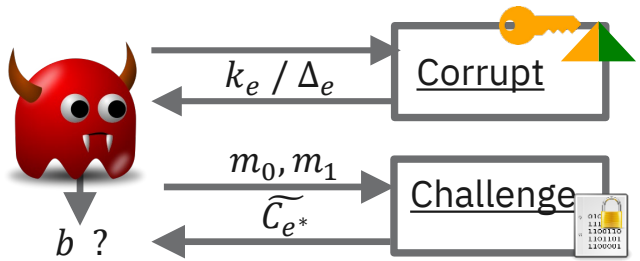

- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates



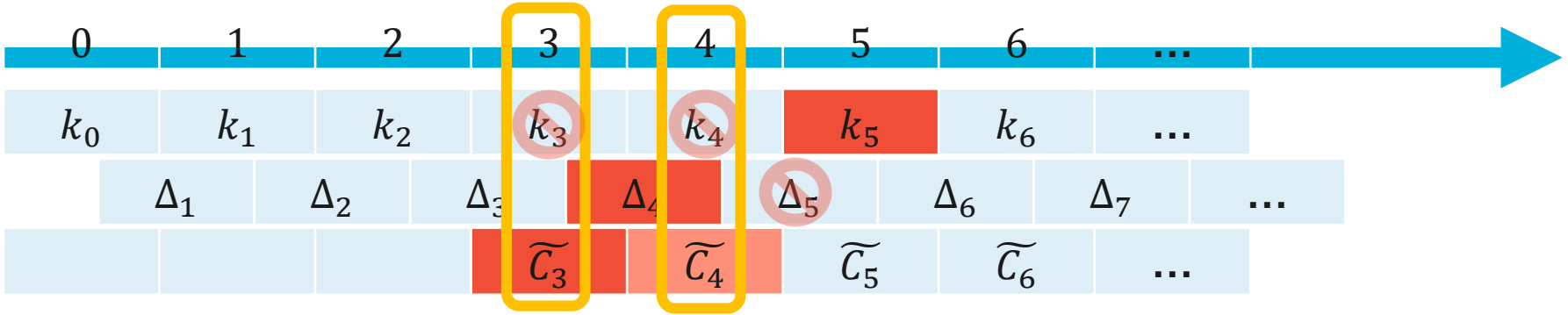
# **Updatable Encryption |** Capturing Trivial Wins



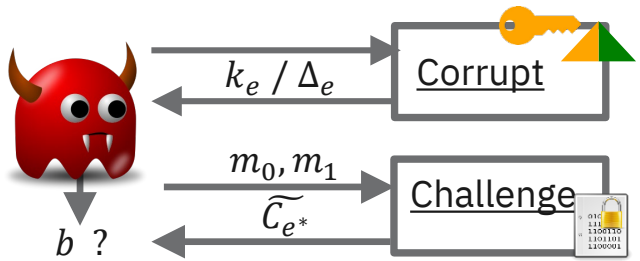

- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates



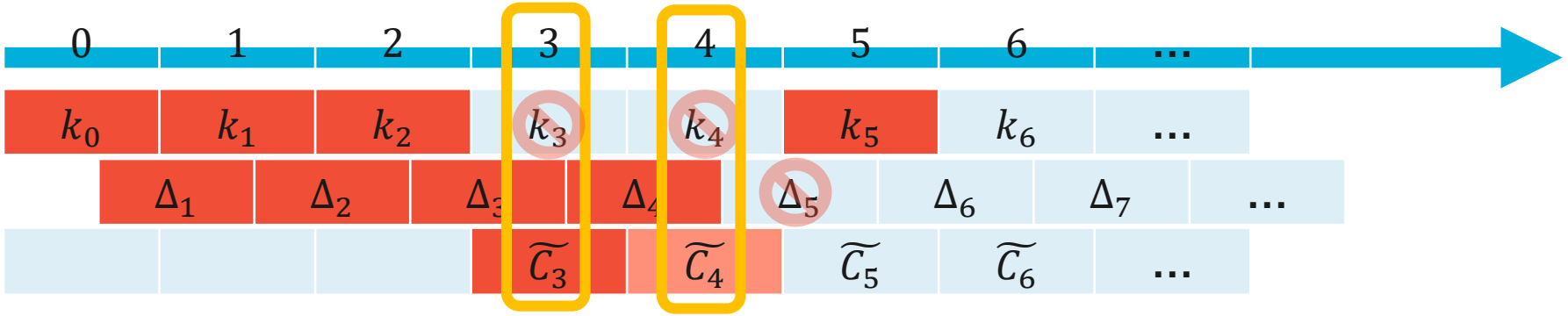
# **Updatable Encryption |** Capturing Trivial Wins



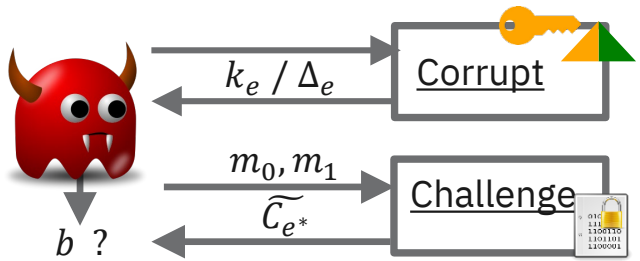

- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates



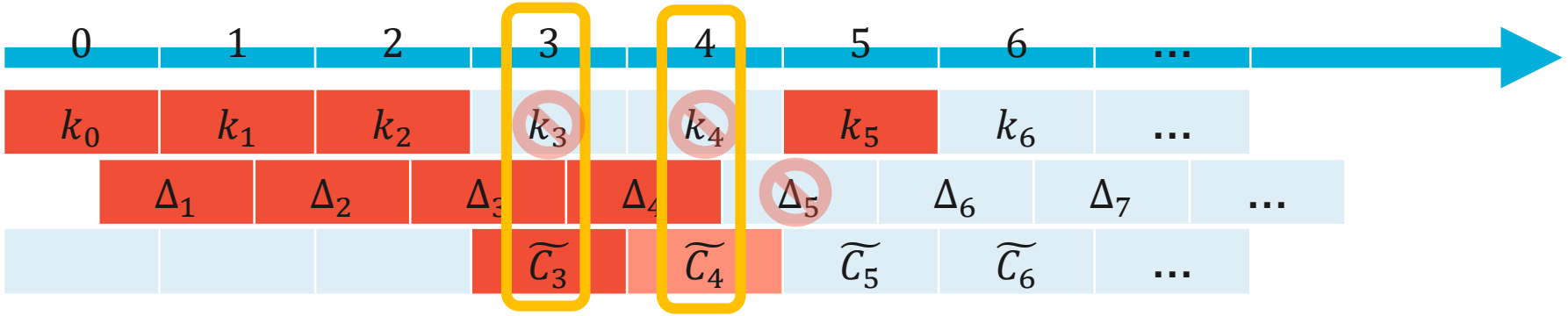
# Updatable Encryption | Capturing Trivial Wins



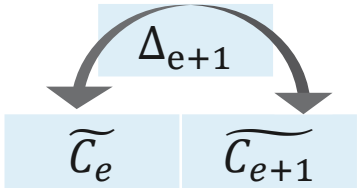

- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates

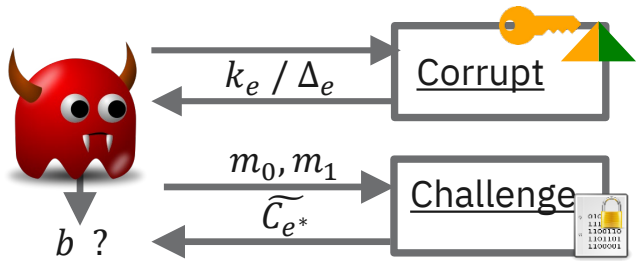


# Updatable Encryption | Capturing Trivial Wins

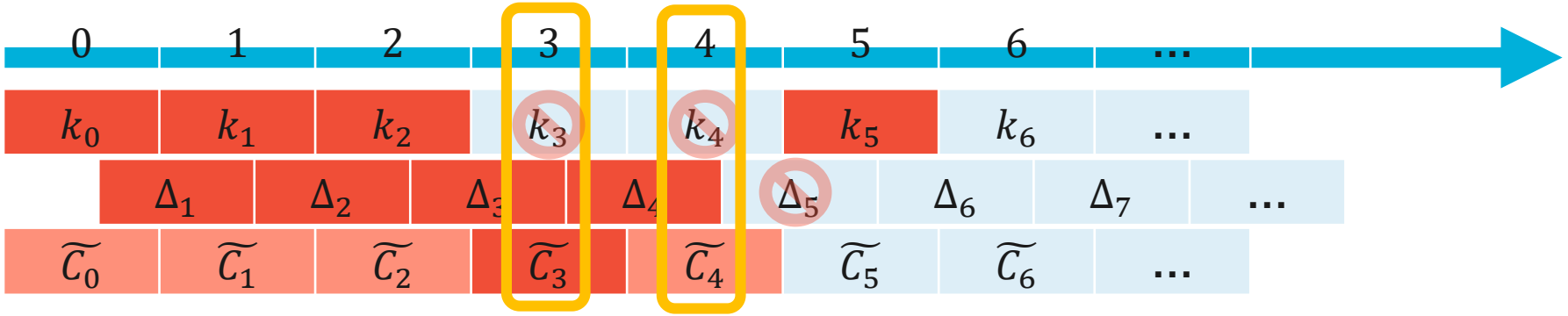



- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates
  - Real: **bi**directional ciphertext-updates

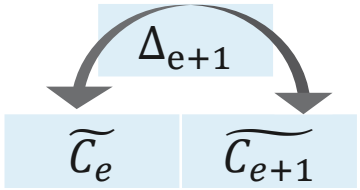


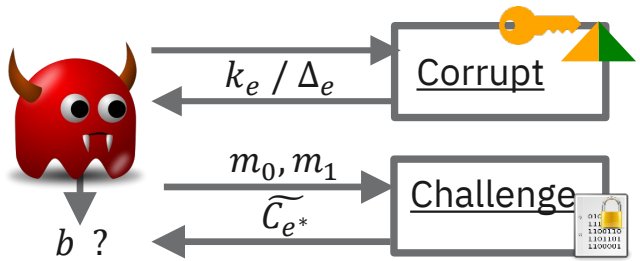



# Updatable Encryption | Capturing Trivial Wins

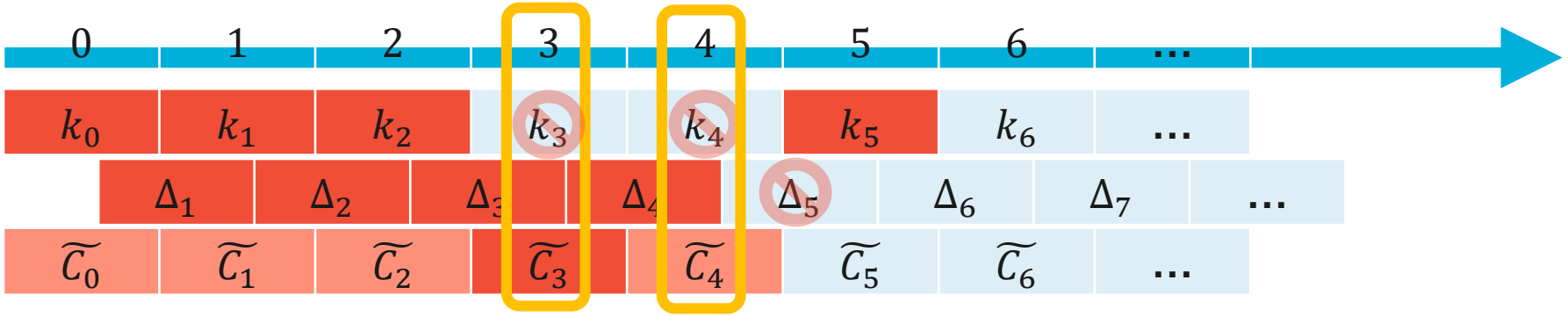



- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates
  - Real: **bi**directional ciphertext-updates

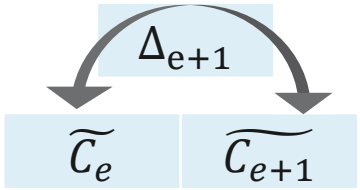


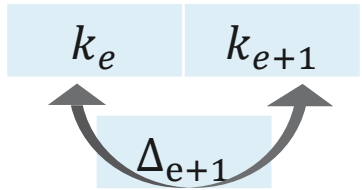



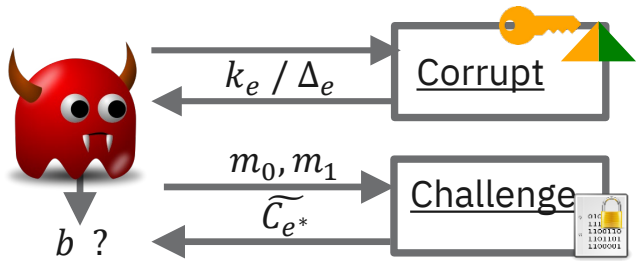
# Updatable Encryption | Capturing Trivial Wins



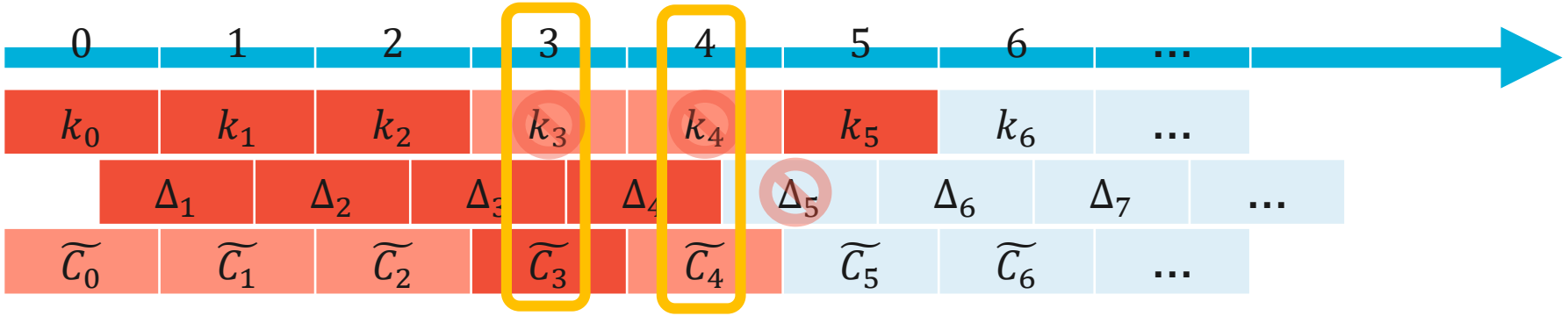

- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates
  - Real: **bi**directional ciphertext **& key-updates**



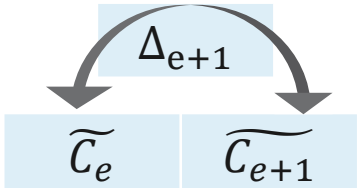



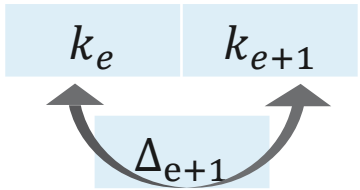



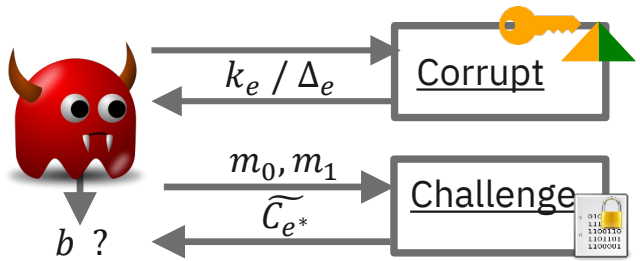
# Updatable Encryption | Capturing Trivial Wins



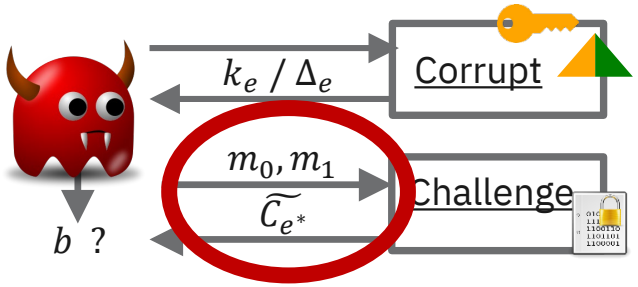

- Trivial win: secret key corruption in a challenge-equal epoch
- Capturing inferable information:
  - Ideal: **uni**directional ciphertext-updates
  - Real: **bi**directional ciphertext **& key-updates**



# Updatable Encryption | IND-ENC




- IND-ENC definition
  - Adaptive and retroactive key & token corruptions
  - Formalizes indirect knowledge of keys & challenge cipherexts
  - Covers CPA, post-compromise and forward security for fresh encryptions & updated ciphertexts

- IND-ENC is not sufficient: No guarantees about updated ciphertexts!
  - UE.upd$(\Delta_{e+1}, C_e) \rightarrow C_{e+1}$

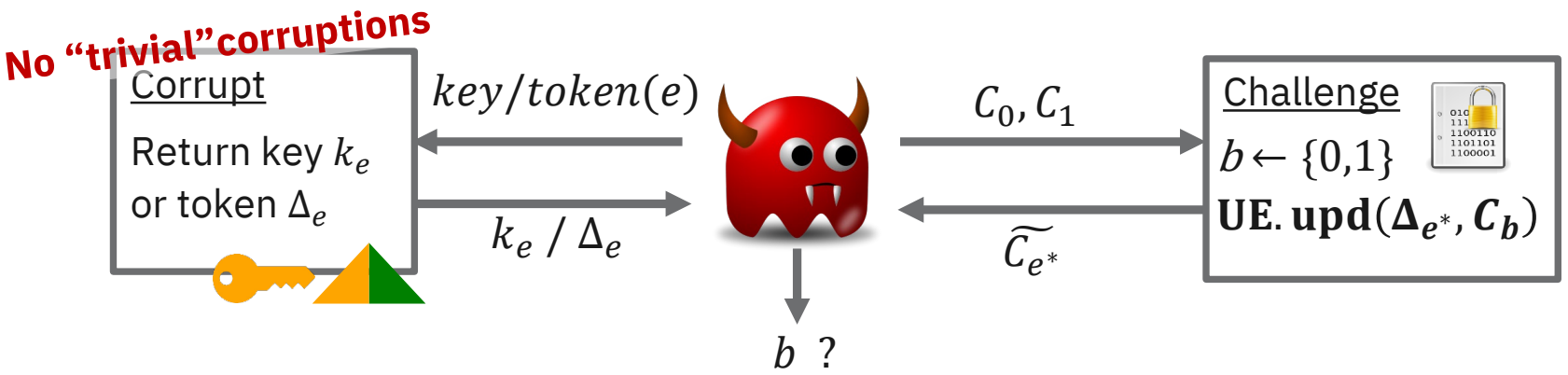
    could contain $C_e$, i.e. history of all old ciphertexts ( $C'_3 = C_3, (C_2, (C_1, (C_0)))$ )

    compromise of a single old key breaks security of updated ciphertexts

Mistake in our EC'18 paper

# Updatable Encryption | IND-UPD

- IND-UPD definition = Update Indistinguishability
  - Unlinkability of updated ciphertexts – no leakage through correlation attacks




IND-ENC = Secure Updatable Encryption

IND-ENC + IND-UPD = Strongly Secure Updatable Encryption

# Updatable Encryption | (In)Secure Schemes

Re-Randomizable Ciphertext-Independent Symmetric ElGamal

| | 2ENC (folklore) | XOR-KEM (EPRS17) | BLMR (BLMR13) | RISE |
|---|---|---|---|---|
| Enc | $Enc(k_e^o, Enc(k^i, m))$ | $(k_e \oplus x), Enc(x, m)$ | $PRF(k_e, N) \otimes m, N$ | |
| Tok $\Delta_{e+1}$ | $(k_e^o, k_{e+1}^o)$ | $k_e \oplus k_{e+1}$ | $k_e \oplus k_{e+1}$ | |
| IND-ENC | (with limitations) | | Key-homomorph PRF | DDH |
| IND-UPD | (with limitations) | | | DDH |

Key-homomorphic PRF: $PRF(k_1, N) \otimes PRF(k_2, N) = PRF(k_1 \oplus k_2, N)$

Also crucial building block in ReEnc [EPRS17] = ciphertext-*dependent* UE

Known instantiations either DL or lattice-based

# **Updatable Encryption |** Secure Construction (RISE)

RISE.setup$(\lambda)$: $x \xleftarrow{r} \mathbb{Z}_q^*$, set $k_0 \leftarrow (x, g^x)$, return $k_0$

RISE.enc$(k_e, m)$: parse $k_e = (x, y)$, $r \xleftarrow{r} \mathbb{Z}_q$, return $C_e \leftarrow (y^r, g^r m)$

RISE.dec$(k_e, C_e)$: parse $k_e = (x, y)$ and $C_e = (C_1, C_2)$, return $m' \leftarrow C_2 \cdot C_1^{-1/x}$

# **Updatable Encryption |** Secure Construction (RISE)

Token doesn't leak info about secret key
(but allows bidirectional key & ciphertext updates)

RISE.setup$(\lambda)$: $x \xleftarrow{r} \mathbb{Z}_q^*$, set $k_0 \leftarrow (x, g^x)$, return $k_0$

RISE.next$(k_e)$: parse $k_e = (x, y)$, draw $x' \xleftarrow{r} \mathbb{Z}_q^*$,
$k_{e+1} \leftarrow (x', g^{x'})$, $\Delta_{e+1} \leftarrow (x'/x, g^{x'})$ return $(k_{e+1}, \Delta_{e+1})$

RISE.enc$(k_e, m)$: parse $k_e = (x, y)$, $r \xleftarrow{r} \mathbb{Z}_q$, return $C_e \leftarrow (y^r, g^r m)$

ElGamal ciphertexts are key-private/anonymous

RISE.dec$(k_e, C_e)$: parse $k_e = (x, y)$ and $C_e = (C_1, C_2)$, return $m' \leftarrow C_2 \cdot C_1^{-1/x}$

RISE.upd$(\Delta_{e+1}, C_e)$: parse $\Delta_{e+1} = (\Delta, y')$ and $C_e = (C_1, C_2)$,
$C_1' \leftarrow C_1^{\Delta}$ return $C_{e+1} \leftarrow (C_1', C_2)$

Re-randomization → updated ciphertexts are unlinkable
(fresh & updated ones are indistinguishable)

# Updatable Encryption | Efficiency

n = number of ciphertexts

| Scheme | | Encryption | TokenGen | Update |
|---|---|---|---|---|
| BLMR | Only IND-ENC secure | 2 exp | 2 exp | 2n exp |
| **RISE** | | **2 exp** | **1 exp** | **2.5n exp** |
| ReEnc [EPRS17] | Ciphertext Dependent | 2 exp | 2n exp | 2n exp |

- RISE: CPA-secure **ciphertext-independent** updatable encryption
- ReEnc [EPRS17]: CCA & CTXT-secure & **ciphertext-dependent** updatable encryption

# Summary

## Data pseudonymization – de-sensitization of data

- Linkability crucial for utility, but also weakens privacy
- Schemes must use strong cryptographic keys → challenges for key management
  - Cryptographic solutions for key updates & oblivious pseudonymization
- Roadmap: Key Management → Chameleon Pseudonyms (solutions are compatible)

## Key rotation & updatable encryption

- Allow convenient updates of key-derived values
- Main challenge: sensible security models

# Thanks! Questions?

anj@zurich.ibm.com