Secure Keyless Multi-Party Storage Scheme

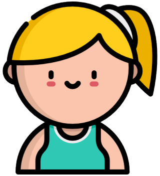
Pascal Lafourcade[1], Lola-Baie Mallordy[1,5], Charles Olivier-Anclin[1,2,4], and Léo Robert[3]

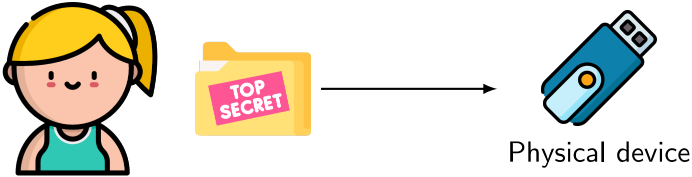
[1] Université Clermont Auvergne, LIMOS, CNRS, [2] be ys Pay, [3] Université de Picardie Jules Verne, Amiens, [4] LIFO, Université d'Orléans, INSA Centre Val de Loire, [5] Institut Polytechnique de Paris, École Polytechnique, LIX, Palaiseau, France & INRIA

August 22, 2024

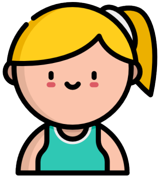
Outline

How to store a secret ?




TOP
SECRET
Physical device

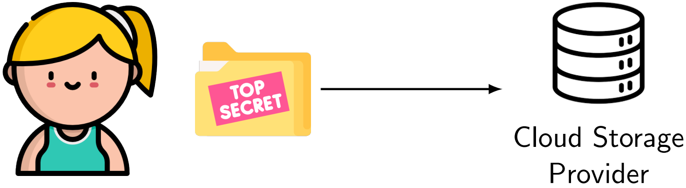
How to store a secret ?



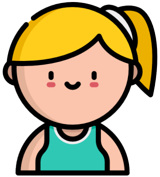
Secret lost!

Physical loss

How to store a secret ?



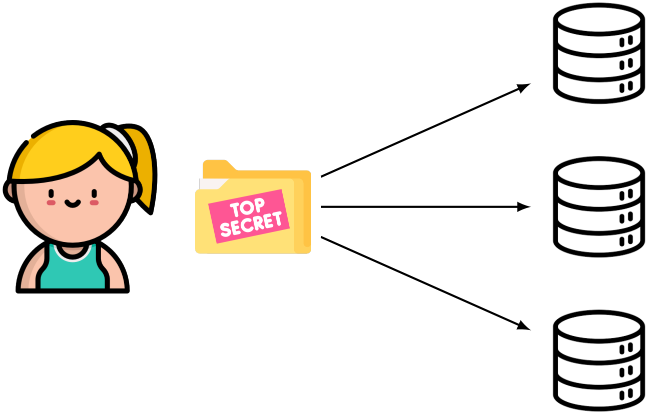
How to store a secret ?



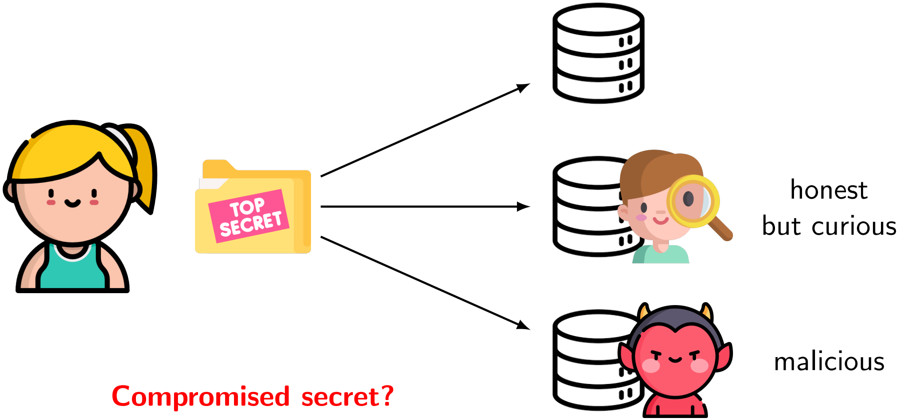
Secret lost!

Single Point of Failure

Multi-Cloud Storage

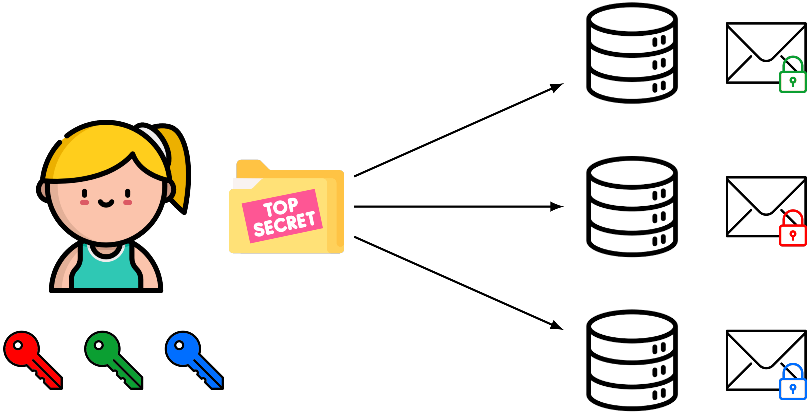


Dangers in multi-cloud storage – Trust issues

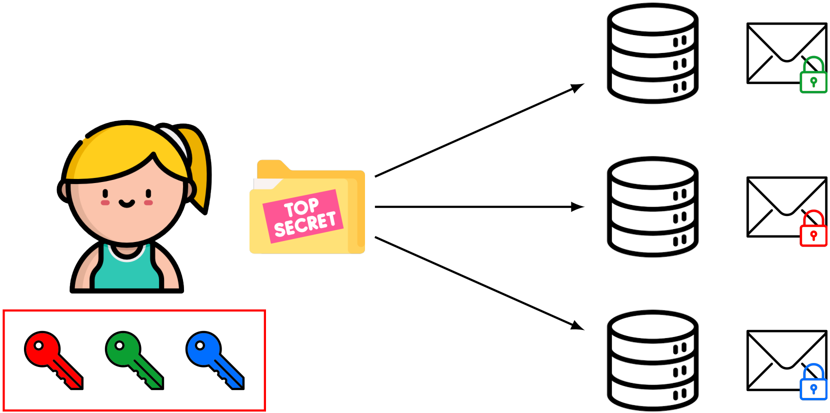


Compromised secret?

Dangers in multi-cloud storage – Key(s) management



Dangers in multi-cloud storage – Key(s) management

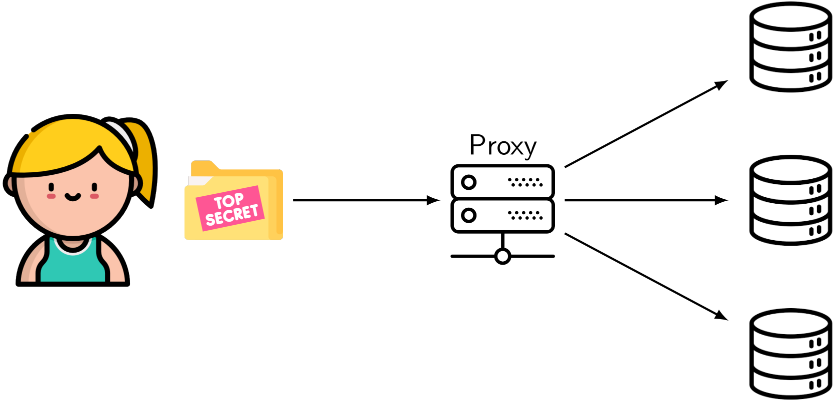


The new top secret

Expected properties for an ideal storage scheme

- The secret must stay confidential to its owner
- Any modification on the secret must be detected
- If the secret is corrupted, the user must know which provider(s) to blame
- Centralized authentication

Centralization in a multi-cloud setting

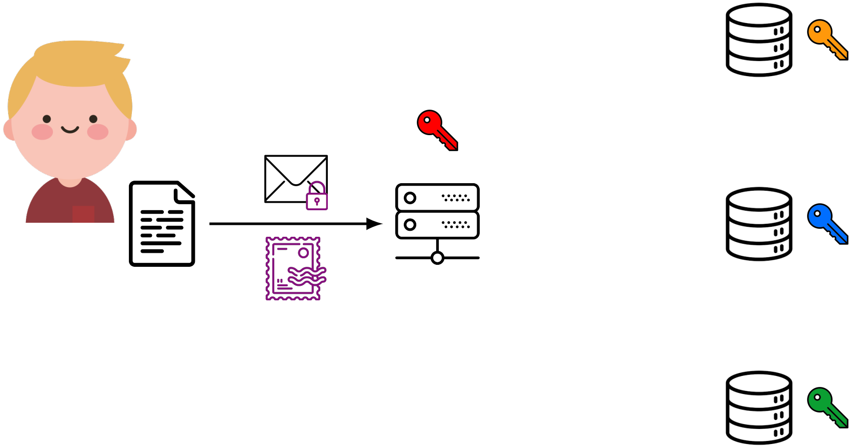


State of the art

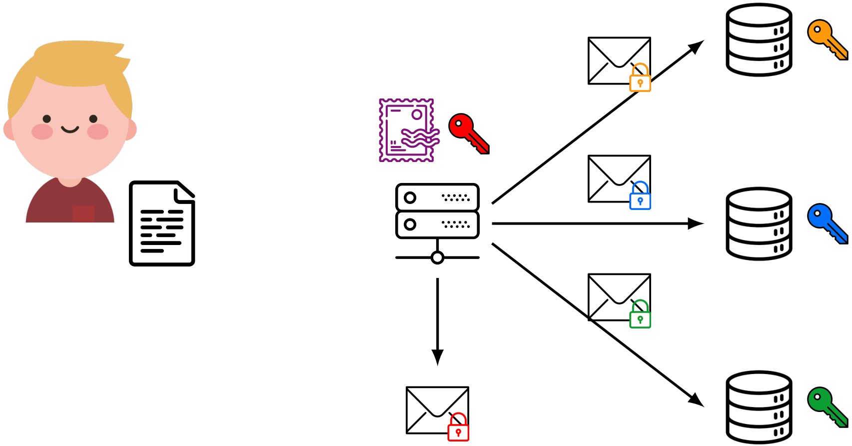
Multi-cloud Protocols	Confidential w.r.t. proxy	Providers collusion	Proxy collusion	Keyless
E. Stefanov et al. 2013	–	✗	–	✗
R. D. Pietro et al. 2017	✗	✗	✗	✗
M. Leila et al. 2020	✗	✗	✗	✗
A. Niknia et al. 2021	–	✓	–	✓
A. N. Bessani et al. 2013	–	✗	–	✓
M. Sulochana et al. 2015	✗	✗	✗	✗
E. N. Witanto et al. 2023	✗	✓	✗	✗
KAPRE	✓	✓	✗	✓
KAME	✓	✓	✓	✓

Outline

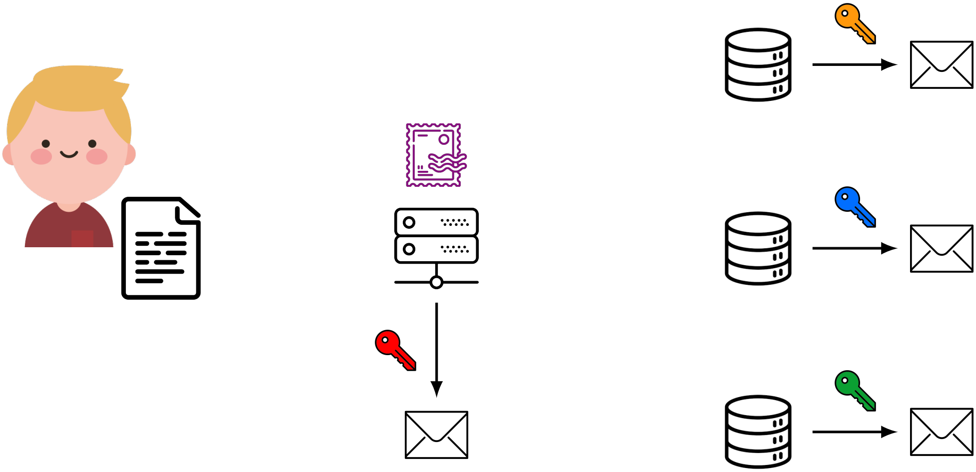
Upload – Transform



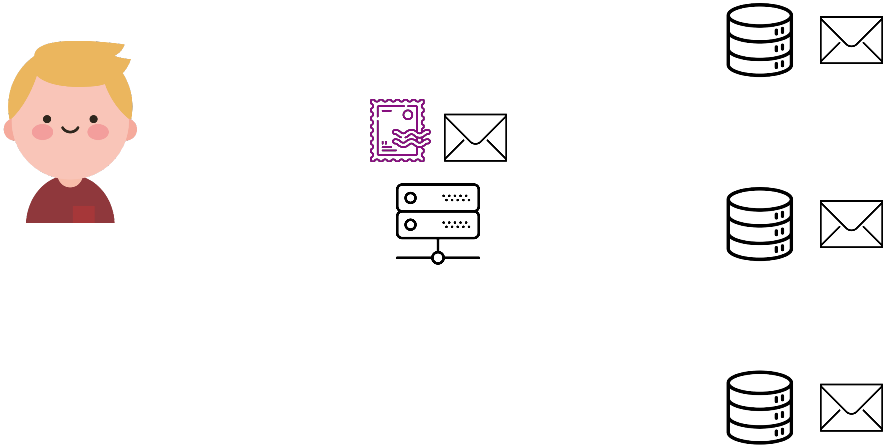
Upload – Distrib



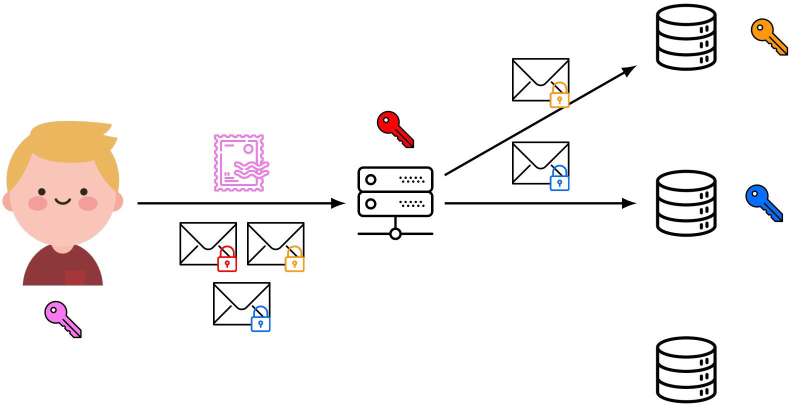
Upload – Open



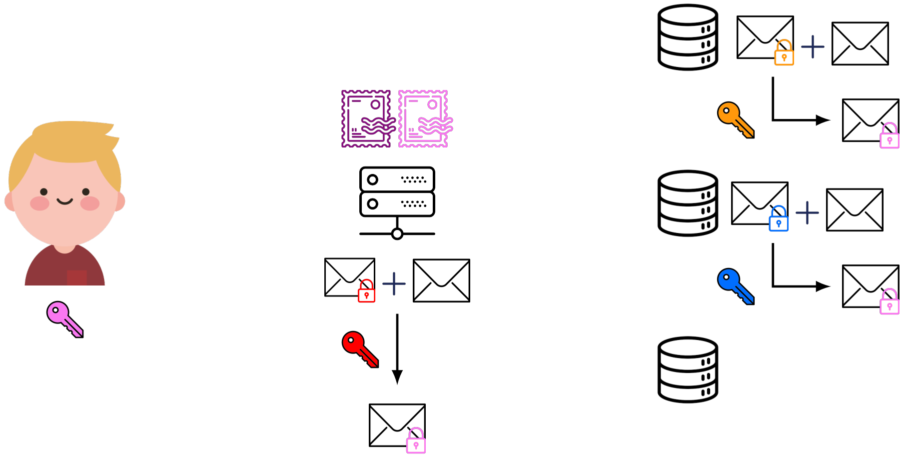
Upload – Final State



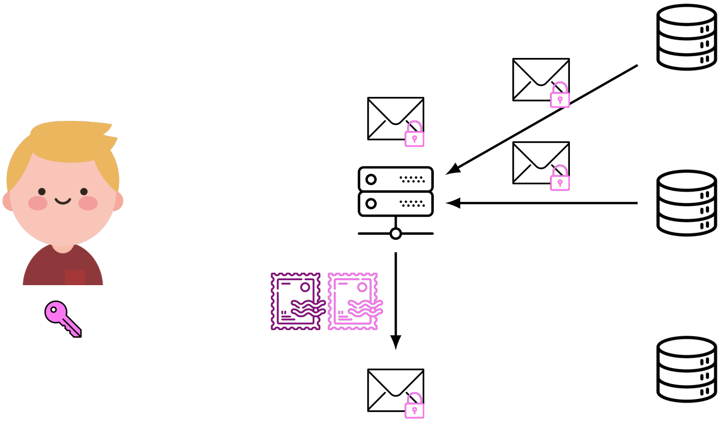
Download – Designate



Download – Hide

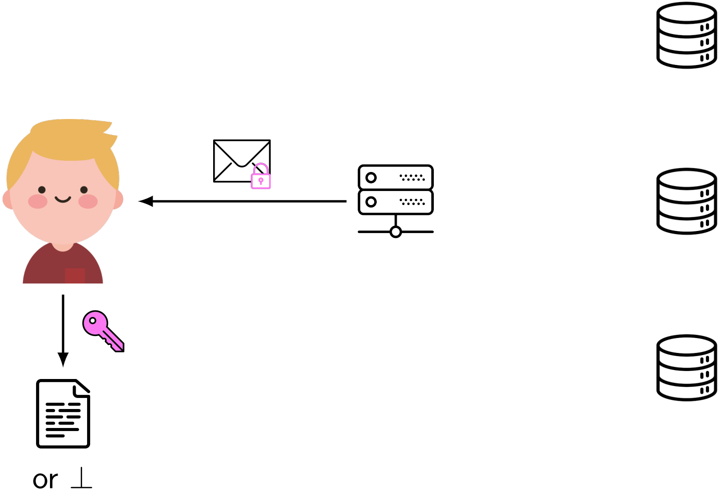


Download – Merge



or blame the culprit(s)!

Download – Recover

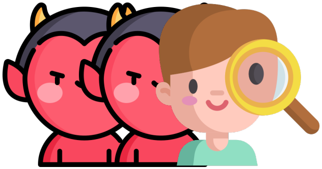


Outline

Adversary model

Proxy
Honest but curious

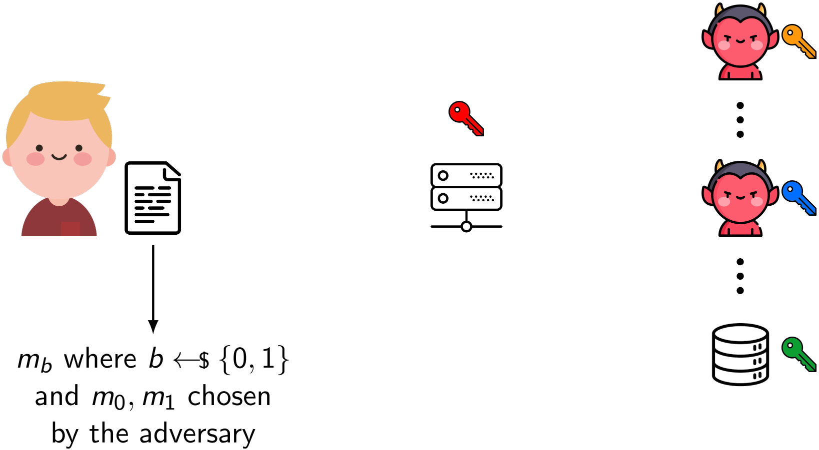
Servers
Malicious



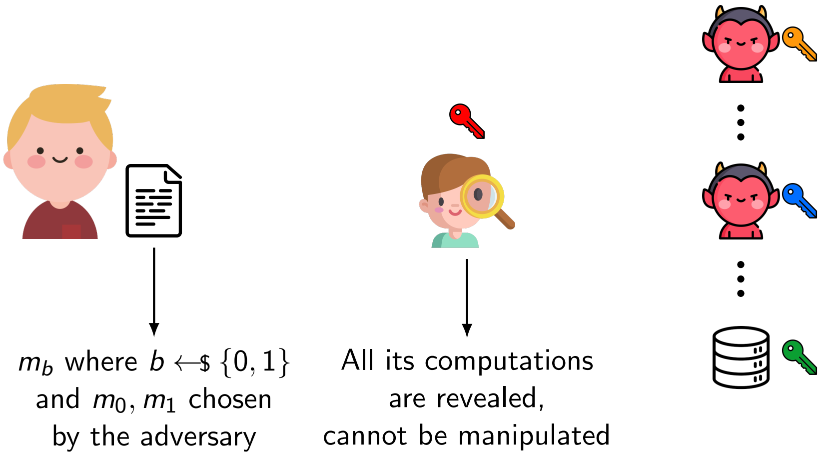
Collusion
of adversaries

$k-$providers secrecy

Guess the bit b ?

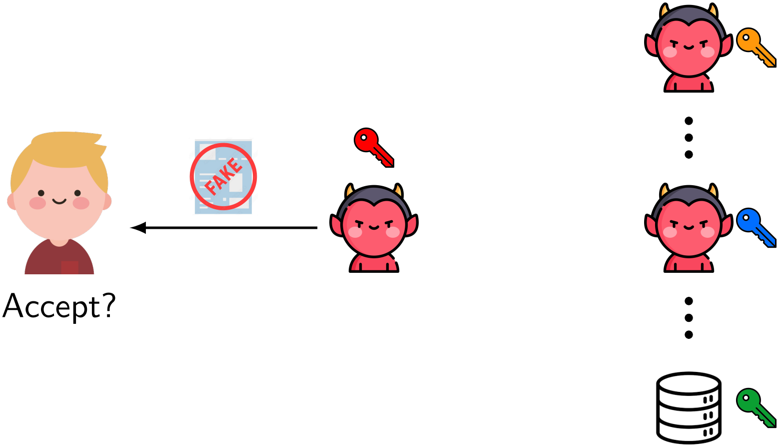


Guess the bit b ?



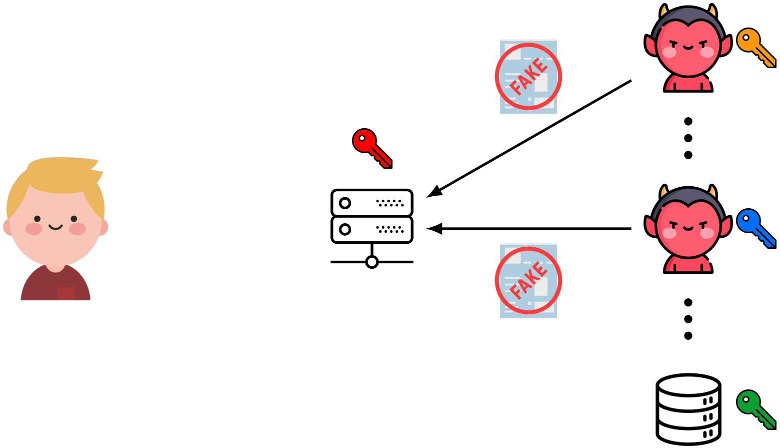
User integrity

After an honest upload of a message chosen by the adversary, send a corrupted secret accepted by the user.



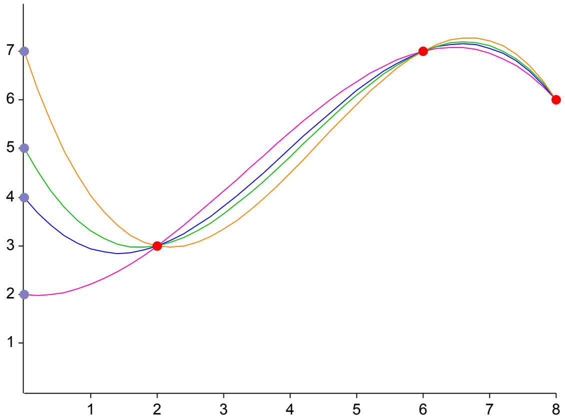
Accountability

After an upload of a message chosen by the adversary, send back corrupted shares such that either the proxy accepts them, or blame uncorrupted shares.



Outline

Shamir's secret sharing – Shamir, 1979



Split $(k, n, m \in \mathbb{Z}_p)$:

$a_1, \ldots, a_{k-1} \leftarrow\$ \mathbb{Z}_p$,

$x_1, \ldots, x_n \leftarrow\$ \mathbb{Z}_p^\times$ pairwise distinct,

$$P(x) = m + \sum_{i=1}^{k-1} a_i X^i,$$

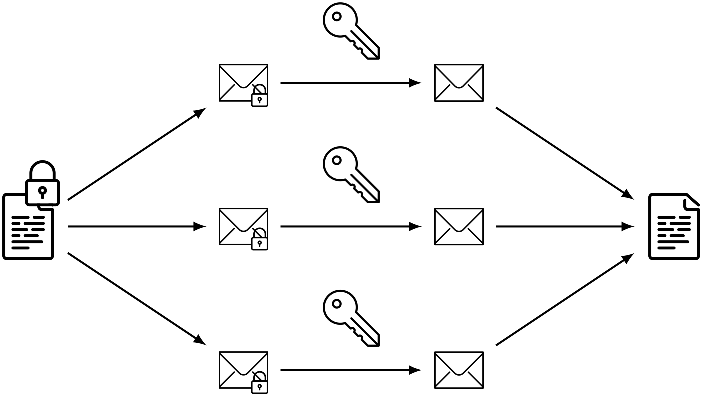
return $(x_1, P(x_1)), \ldots, (x_n, P(x_n))$

Reconstruct $(k, (x_1, y_1), \ldots, (x_k, y_k))$:

return $\displaystyle\sum_{i=1}^{k} y_i \prod_{j \neq i} \frac{-x_j}{x_i - x_j}.$

Homomorphic encryption – Brakerski, Gentry, Vaikuntanathan, 2014

$$\mathrm{Dec}(\mathrm{Enc}(m, \mathrm{pk}) + \mathrm{Enc}(n, \mathrm{pk}), \mathrm{sk}) = m + n$$



Key homomorphic pseudorandom function family – Banerjee, Peikert 2014

Definition

Let D be a finite set and R, S groups. Let $\mathcal{F} = \{F_s\}_{s \in S}$ be a family of keyed functions mapping $D \to R$. The family $\mathcal{F}$ is pseudorandom if the advantage of any adversary $\mathcal{A}$ given oracle access to a function f to distinguish if $f = F_s$ for $s \leftarrow\!\$\, S$ or if f was randomly chosen from the set of functions mapping $D \to R$. $\mathcal{F}$ is key homomorphic if for all $x \in D$,

$$F_a(x) \cdot F_b(x) = F_{a+b}(x).$$

Information Dispersal Algorithm (IDA) – Rabin, 1989

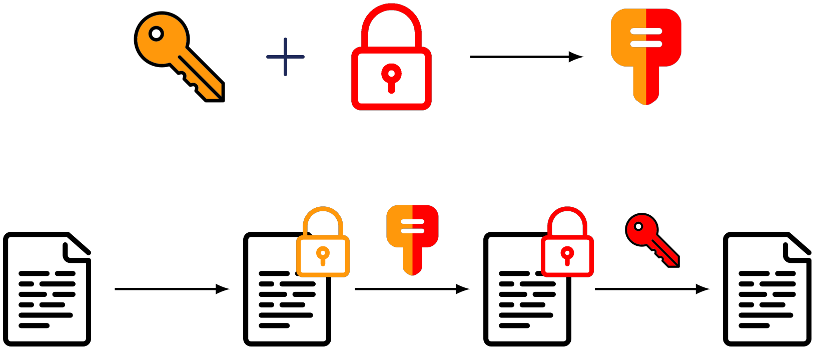
$\mathsf{Split}((m_1, \ldots, m_k) \in \mathbb{Z}_p^k, n, k)$: $A \leftarrow\!\!\$\; \mathbb{Z}_p^{k \times n}$ such that every $k \times k$ submatrix of A is invertible,

$$\textbf{return} \quad A \, m = \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} \in \mathbb{Z}_p^n.$$

$\mathsf{Rec}\big(A, r_{i_1}, \ldots, r_{i_k}\big)$: Let A' be the $k \times k$ submatrix formed by the lines $i_1, \ldots, i_k$ of A,

$$\textbf{return} \quad {A'}^{-1} \begin{pmatrix} r_{i_1} \\ \vdots \\ r_{i_k} \end{pmatrix} = m \in \mathbb{Z}_p^k.$$

Proxy Re-Encryption – KeySwitching (BGV)



Outline

User:

$\text{recK} \leftarrow \text{E.KeyGen}$

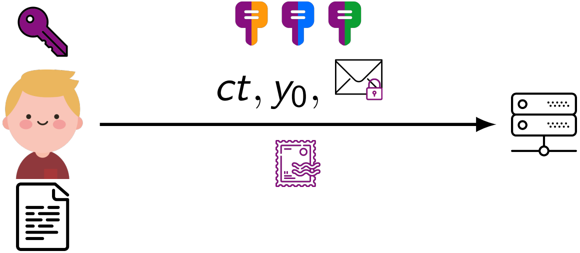
$ct \leftarrow \{$  $\}_{\text{recK}}$

$a_1, \ldots, a_{k-1} \xleftarrow{\$} \mathbb{Z}_p$

$$y_0 \leftarrow \text{recK} + \sum_{i=1}^{k-1} a_i$$

 $\leftarrow \{\text{recK}\}$  $, \{\{a_i\}$  $\}_{i=1}^{k-1}$

 $\leftarrow x, F_{\text{recK}}(x), \{F_{a_i}(x)\}_{i=1}^{k-1}$



Proxy:

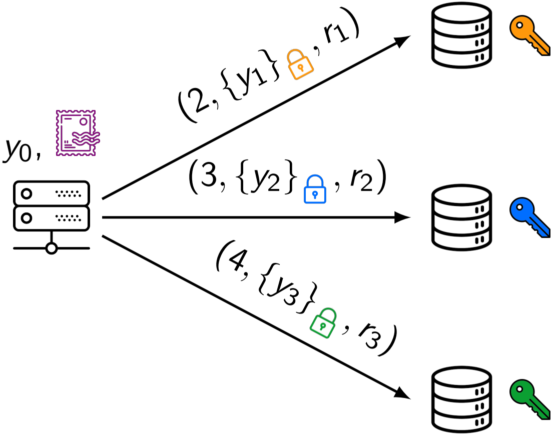
$\{r_i\}_i \leftarrow \text{IDA.Split}(ct, n+1, k)$

$\{y_i\} \leftarrow \{\text{recK}\} + \sum_{j=1}^{k-1} \{a_j\} (i+1)^j$

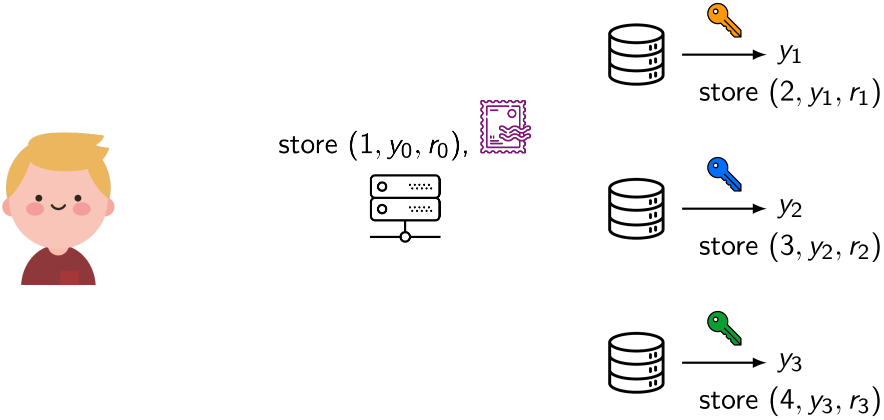
$\{y_1\} \leftarrow \text{PRE.ReEnc}(\{y_1\}, \quad)$

$\{y_2\} \leftarrow \text{PRE.ReEnc}(\{y_2\}, \quad)$

$\{y_3\} \leftarrow \text{PRE.ReEnc}(\{y_3\}, \quad)$



Upload KAPRE ($n = 3, k$) – Open



Weakness of KAPRE

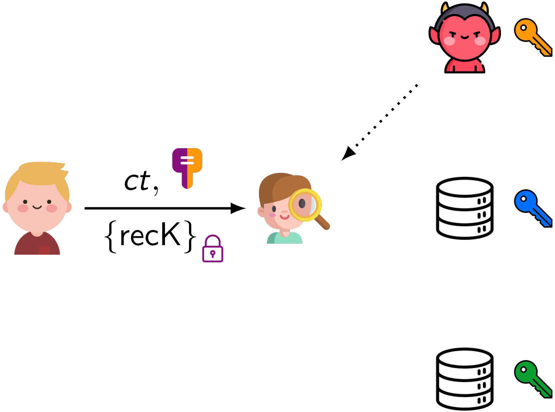
Adversary:

$\{\text{recK}\} \leftarrow \text{PRE.ReEnc}(\{\text{recK}\}, \;)$

$\text{recK} \leftarrow \text{PRE.Dec}(\{\text{recK}\}, \;)$

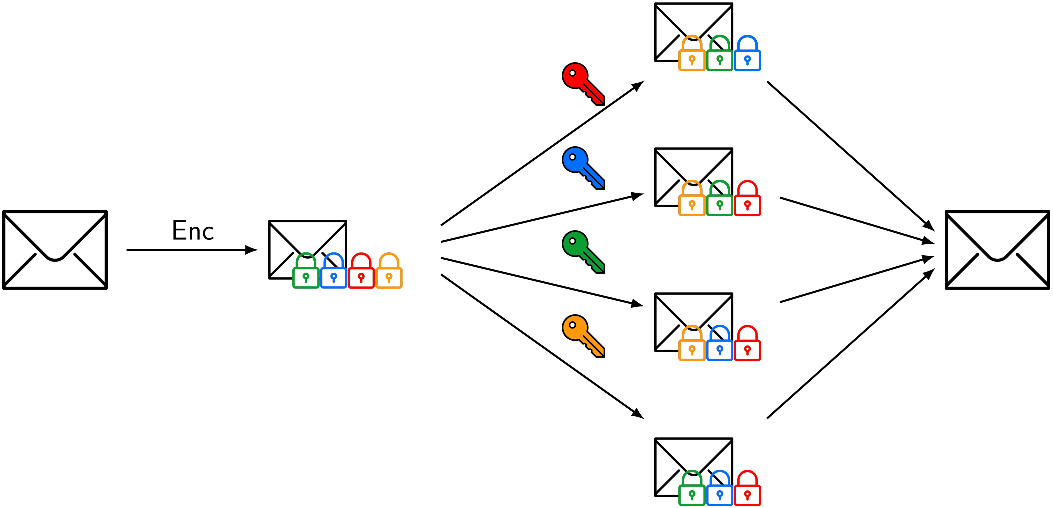
$\leftarrow \text{E.Dec}(ct, \text{recK})$

No secrecy for the user's data!



Outline

Multi-Key Encryption Scheme – Lòpez-Alt et al., 2012



Upload KAME ($n = 3, k$) – Transform

<u>User:</u>

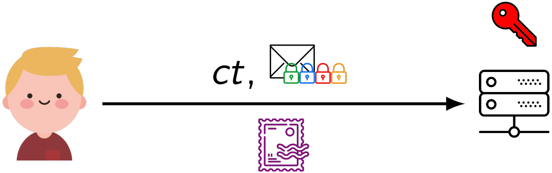
$\text{recK} \leftarrow \text{E.KeyGen}$

$ct \leftarrow \{\cdot\}_{\text{recK}}$

$a_1, \ldots, a_{k-1} \leftarrow\$ \mathbb{Z}_p$

 $\leftarrow \{\text{recK}\}, \{\{a_i\}\}_{i=1}^{k-1}$

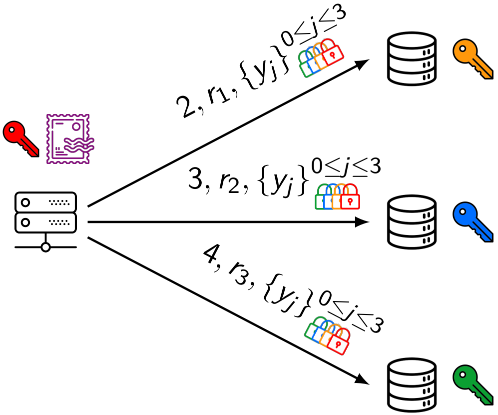
 $\leftarrow x, F_{\text{recK}}(x), \{F_{a_i}(x)\}_{i=1}^{k-1}$



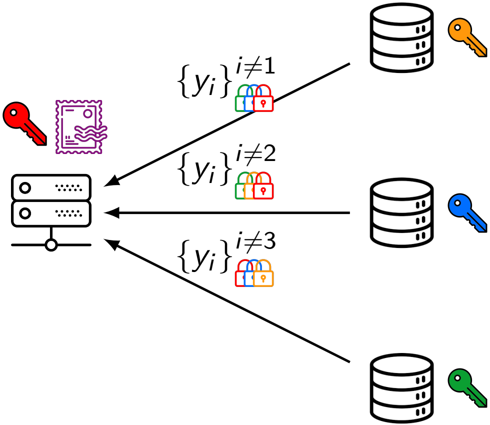
Proxy:

$$\{y_i\} \leftarrow \{\text{recK}\} + \sum_{j=1}^{k-1} \{a_j\} (i+1)^j$$

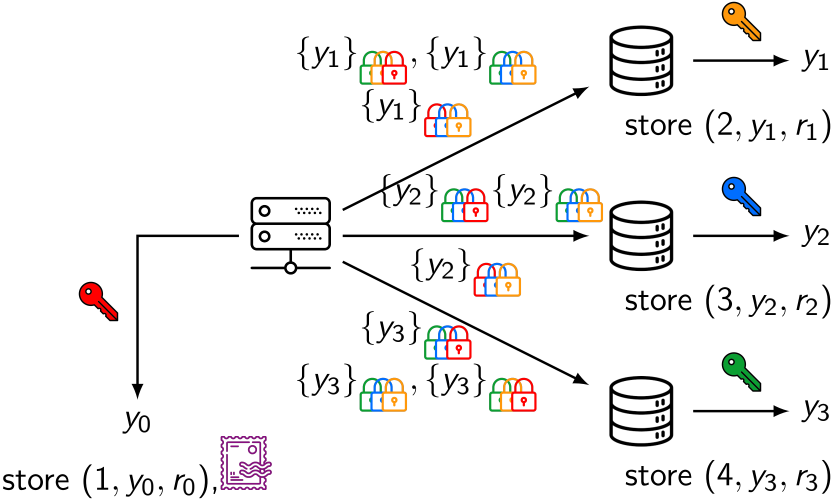
$$\{r_i\} \leftarrow \text{IDA.Split}(ct, n+1, k)$$



Upload KAME $(n = 3, k)$ – Open

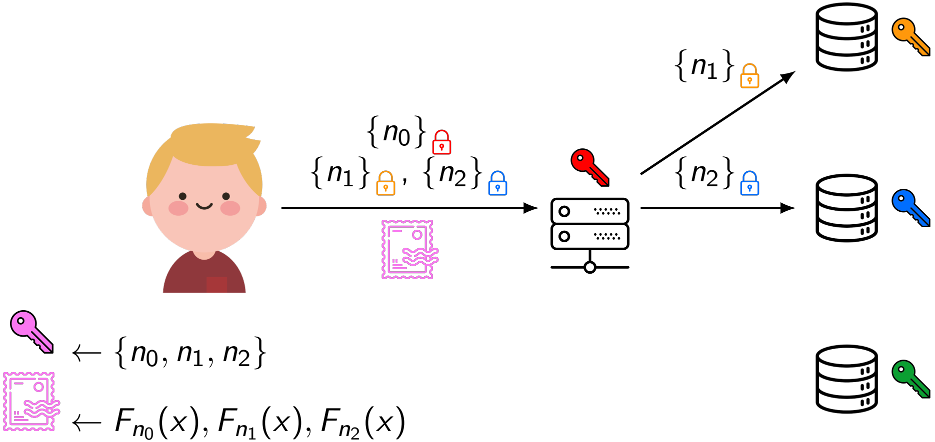


Upload KAME ($n = 3, k$) – Open



Outline

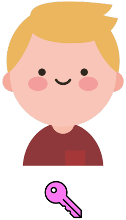
Download $(n = 3, k = 3)$ – Designate

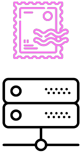


Download ($n = 3, k = 3$) – Hide

Retrieve $(2, y_1, r_1)$
$y'_1 \leftarrow y_1 + n_1$

Retrieve $(3, y_2, r_2)$
$y'_2 \leftarrow y_2 + n_2$





Retrieve $(1, y_0, r_0)$,

$y'_0 \leftarrow y_0 + n_0$

Download ($n = 3, k = 3$) – Merge

Proxy:

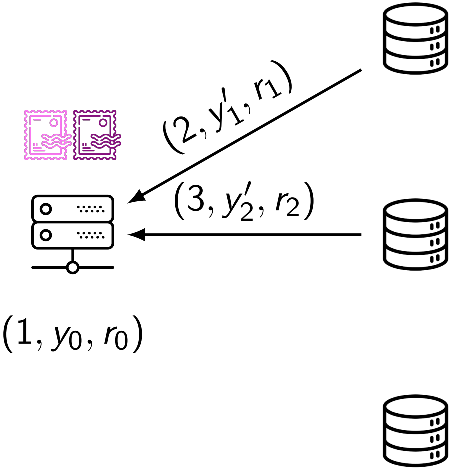
shiftK $\leftarrow \sum_{i=0}^{2} y'_i \ell_i$

if $F_{\mathsf{recK}}(x) + \sum F_{n_i}(x)\ell_i = F_{\mathsf{shiftK}}(x)$:

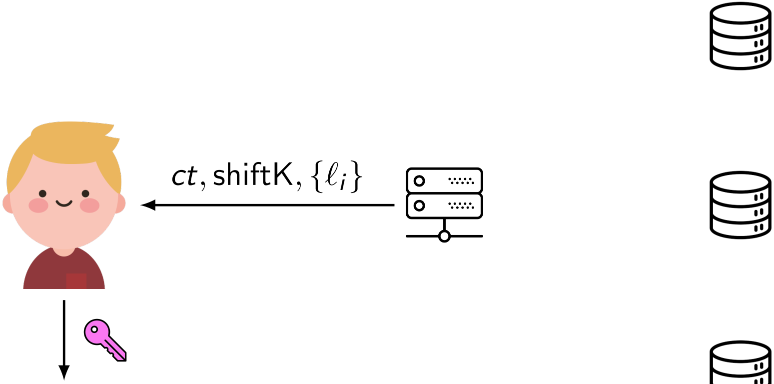
$$ct \leftarrow \mathsf{IDA.Rec}(\{r_i\}, 3)$$

else blame every party for which

$$F_{y'_i}(x) \neq F_{n_i}(x) + F_{\mathsf{recK}}(x) + \sum_{j=1}^{k-1} F_{a_j} x_i^j$$



Download – Recover



$$\mathsf{recK} \leftarrow \mathsf{shiftK} - \sum n_i \ell_i$$

 $\leftarrow \mathsf{Dec}(ct, \mathsf{recK})$

Security

Theorem

Assume that the proxy re-encryption scheme and the symmetric encryption have indistinguishability under plaintext attack and the function family $\{F_x\}_x$ is pseudorandom. Then, KAPRE *achieves 0-collusion secrecy.*

Theorem

Assume that the symmetric encryption have indistinguishability under plaintext attack. Then, KAPRE *achieves $(k-1)$ provider secrecy.*

Security

Theorem

Assume that the symmetric encryption and the multi-key encryption have indistinguishability under plaintext attack, and the function family $\{F_x\}$ *is pseudorandom. Then,* KAME *achieves* $(k-2)$ *collusion secrecy.*

Security

Theorem

Assume that the symmetric encryption and the multi-key encryption have indistinguishability under plaintext attack, and the function family $\{F_x\}$ is pseudorandom. Then, KAME *achieves* $(k-2)$ *collusion secrecy.*

Theorem

Assume that the symmetric encryption has authenticity, the function family $\{F_x\}_x$ is pseudorandom and the public key encryption has indistinguishability. Then, both schemes have user integrity.

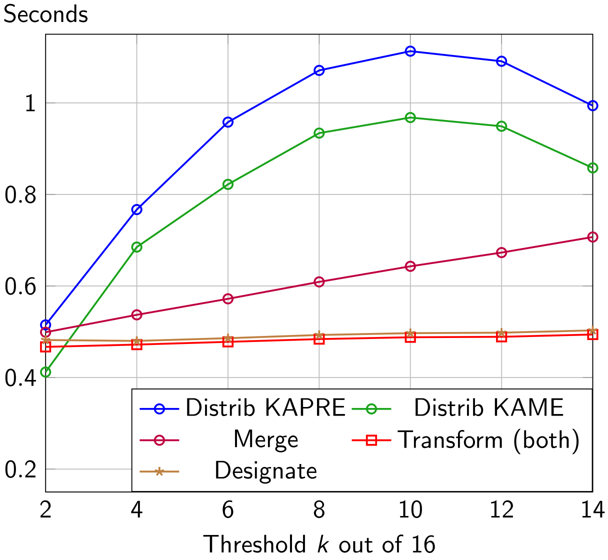
Theorem

Assume that the function family $\{F_x\}$ is pseudorandom. Then, both schemes have accountability.

Outline

Experiments – Average execution time comparison

Benchmarks:
Ubuntu 22.04.2 laptop
messages of 1MB



Complexity for a (n, k) sharing

Protocols	Security	Complexity	Communication
Upload KAPRE	Proxy, collusion of servers	$\mathcal{O}(nk - k^2)$	One round
Upload KAME	Proxy colluding with servers	$\mathcal{O}(nk - k^2)$	Interactive
Download	Collusion proxy with serveurs	$\mathcal{O}(k)$	One round

Thank you for your attention !