Clermont'ech API Hour #34

How to Search on Encrypted Data?

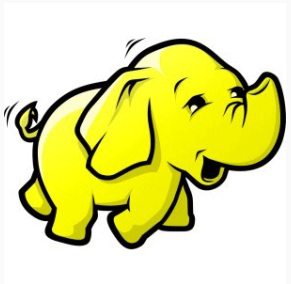
Matthieu Giraud

jeudi 15 mars 2018

LIMOS

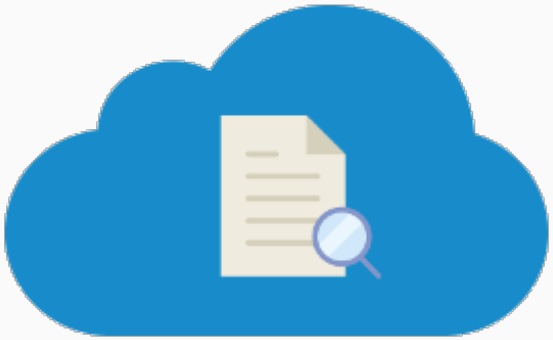
Université Clermont Auvergne

Data Outsourcing



Threat Models

1) Honest but Curious Cloud



2) Hacker

Idea: Encrypted Data

Idea: Encrypted Data

Symmetric Encryption Scheme (AES)

E( , ) = 

Idea: Encrypted Data

Symmetric Encryption Scheme (AES)

E( , ) = 

D( , ) = 

Idea: Encrypted Data

Symmetric Encryption Scheme (AES)

Asymmetric Encryption Scheme (RSA)

Idea: Encrypted Data

Symmetric Encryption Scheme (AES)

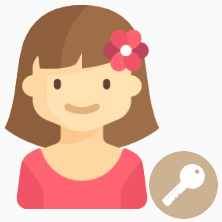
Asymmetric Encryption Scheme (RSA)

Functionnality

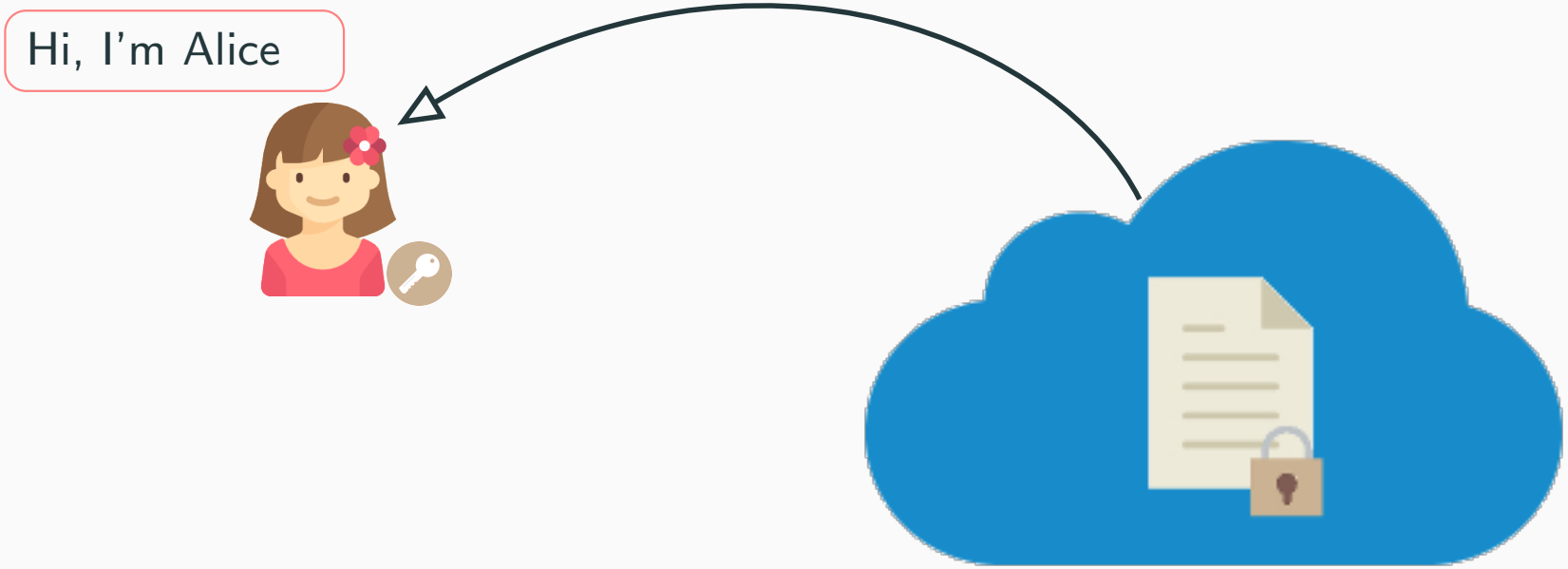
“urgent” $\in$  ?

First Solution: Classic Encryption Scheme

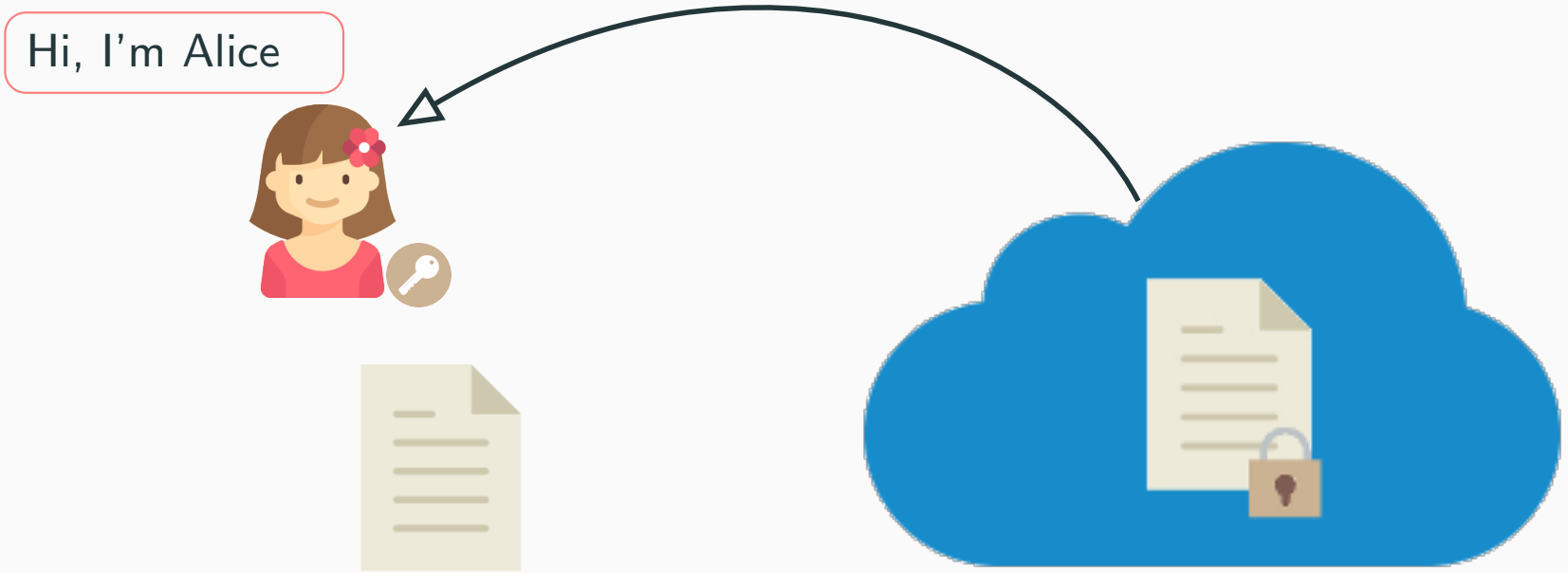
Hi, I'm Alice



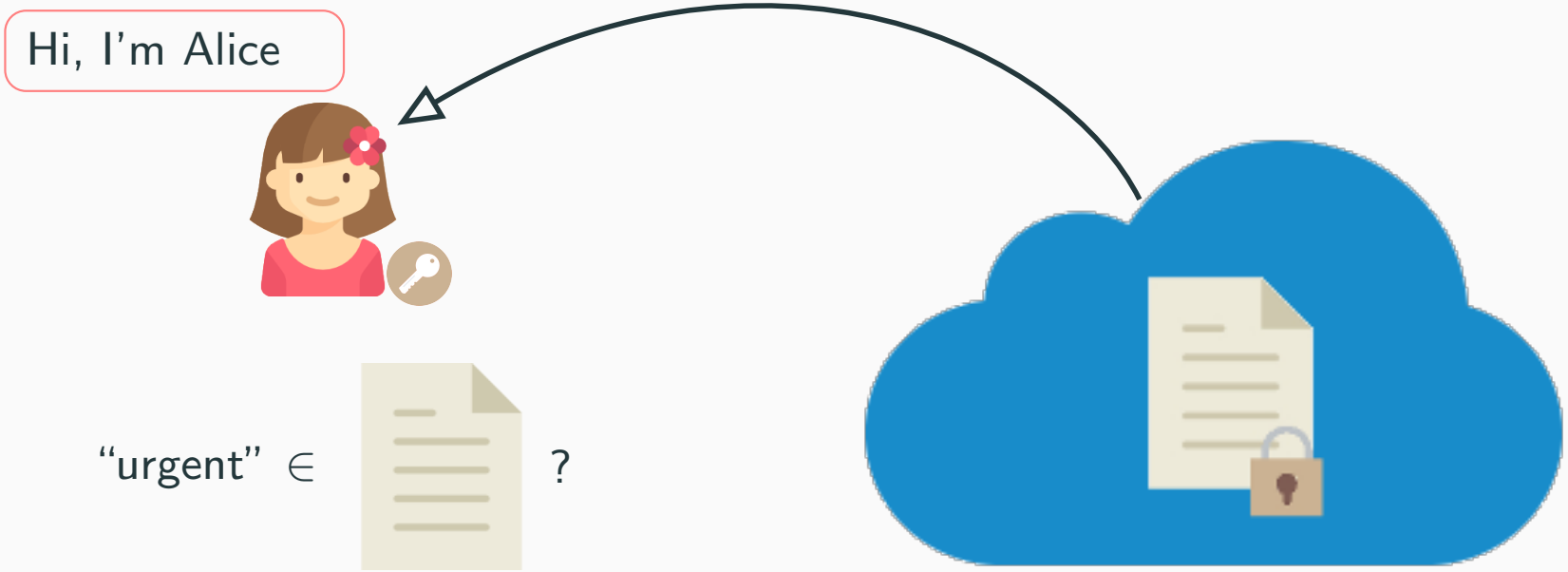
First Solution: Classic Encryption Scheme



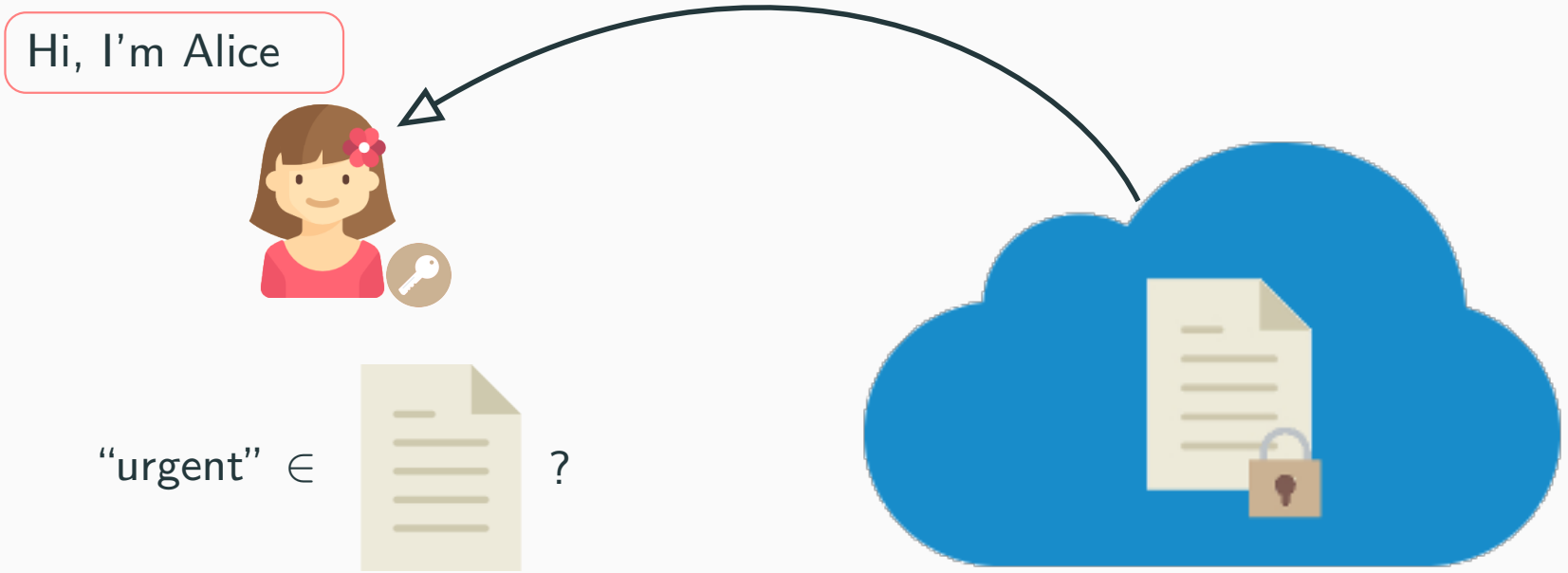
First Solution: Classic Encryption Scheme



First Solution: Classic Encryption Scheme

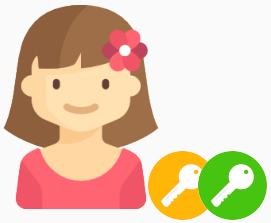


First Solution: Classic Encryption Scheme



Secure ~~Functionnality~~ ~~Efficient~~

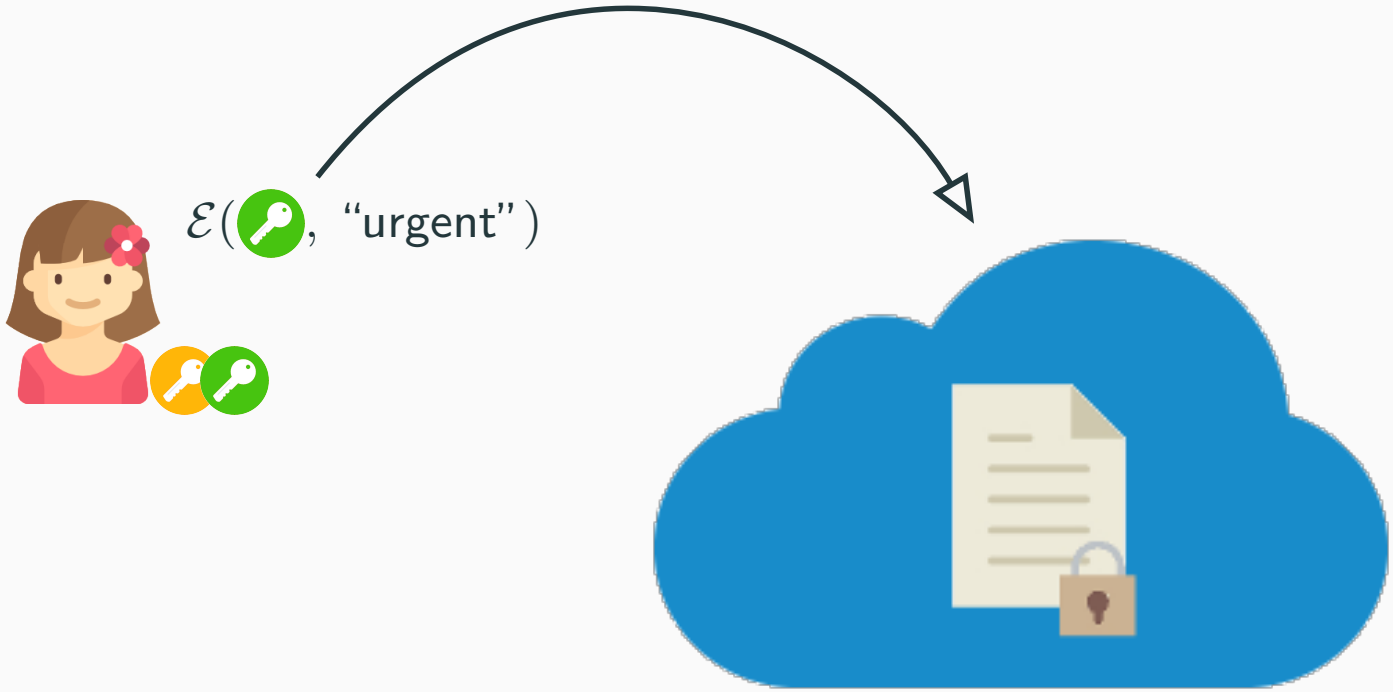
Second Solution: Homomorphic Encryption Scheme



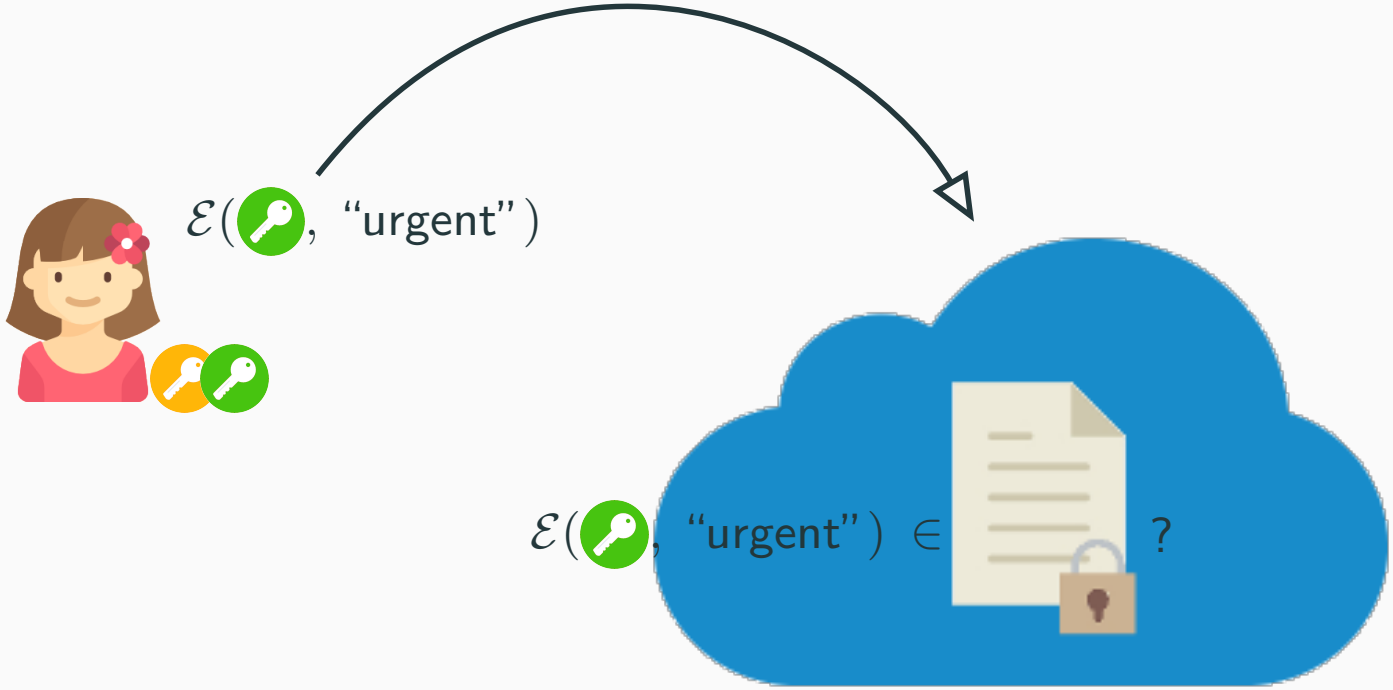
Second Solution: Homomorphic Encryption Scheme

$\mathcal{E}($, “urgent” $)$

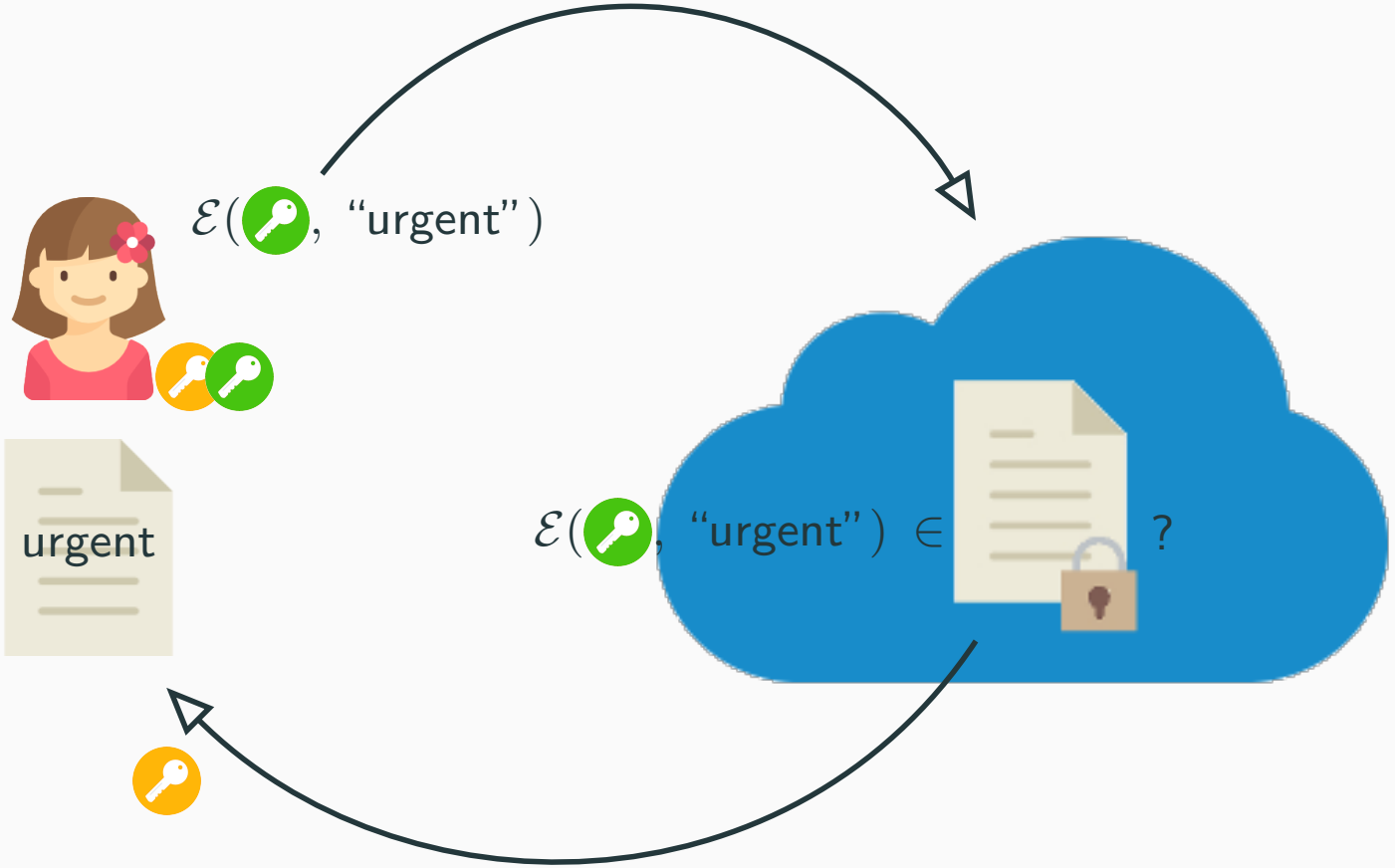
Second Solution: Homomorphic Encryption Scheme



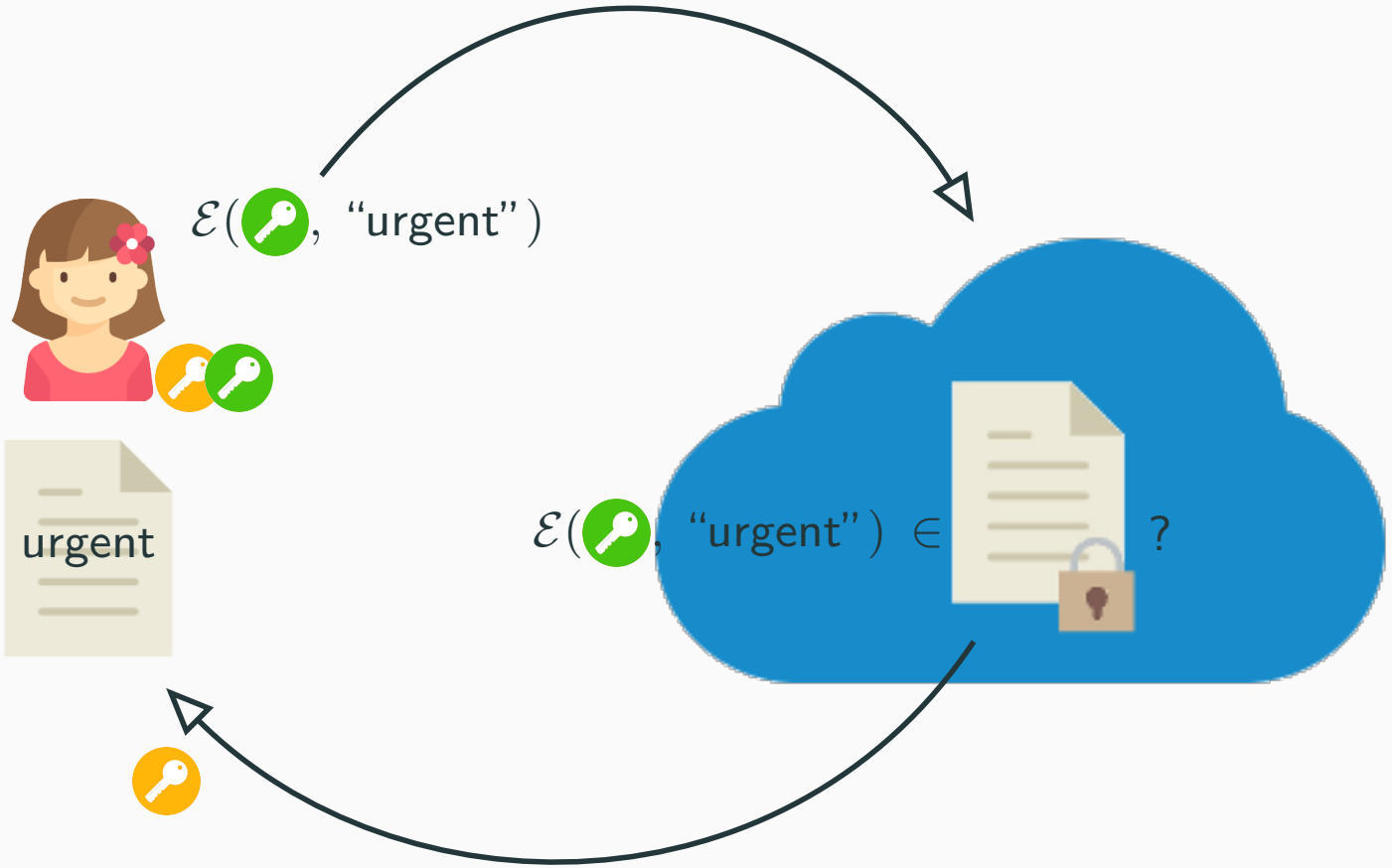
Second Solution: Homomorphic Encryption Scheme



Second Solution: Homomorphic Encryption Scheme

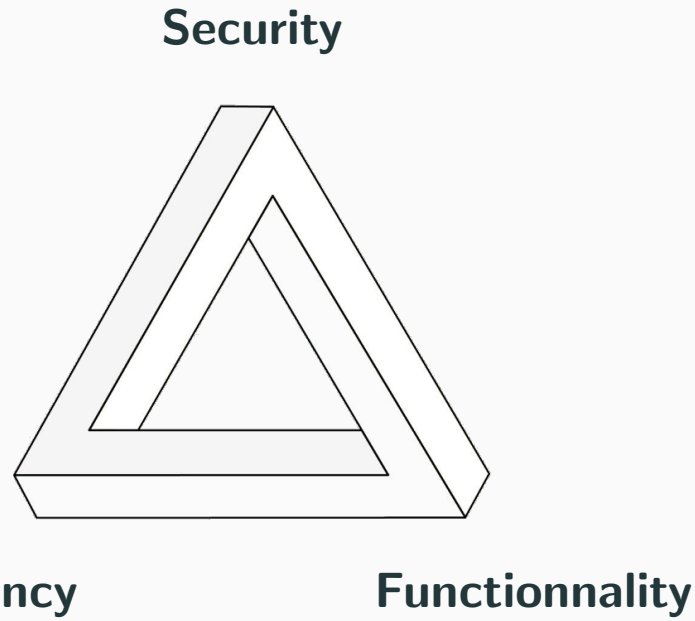


Second Solution: Homomorphic Encryption Scheme



Secure **Functionnality** ~~Efficient~~

Third Solution: Searchable Encryption Scheme



Designed by Song, Wagner and Perrig in 2000.

Index-based Searchable Encryption Scheme

Symmetric Encryption Scheme E

Index-based Searchable Encryption Scheme

Symmetric Encryption Scheme E

Pseudo-random function F

Index-based Searchable Encryption Scheme

Symmetric Encryption Scheme E

Pseudo-random function F

 $+$ “urgent” $\Rightarrow$ 45Zu:?9 $(= \text{urgent}^*)$

Index-based Searchable Encryption Scheme

Inverted Index

$d_1 =$ "hello world" $\qquad\qquad$ $d_2 =$ "hello you"

keyword	documents
hello	id_1 id_2
world	id_1
you	id_2

Index-based Searchable Encryption Scheme

Inverted Index

$d_1 =$ "hello world" $\qquad\qquad$ $d_2 =$ "hello you"

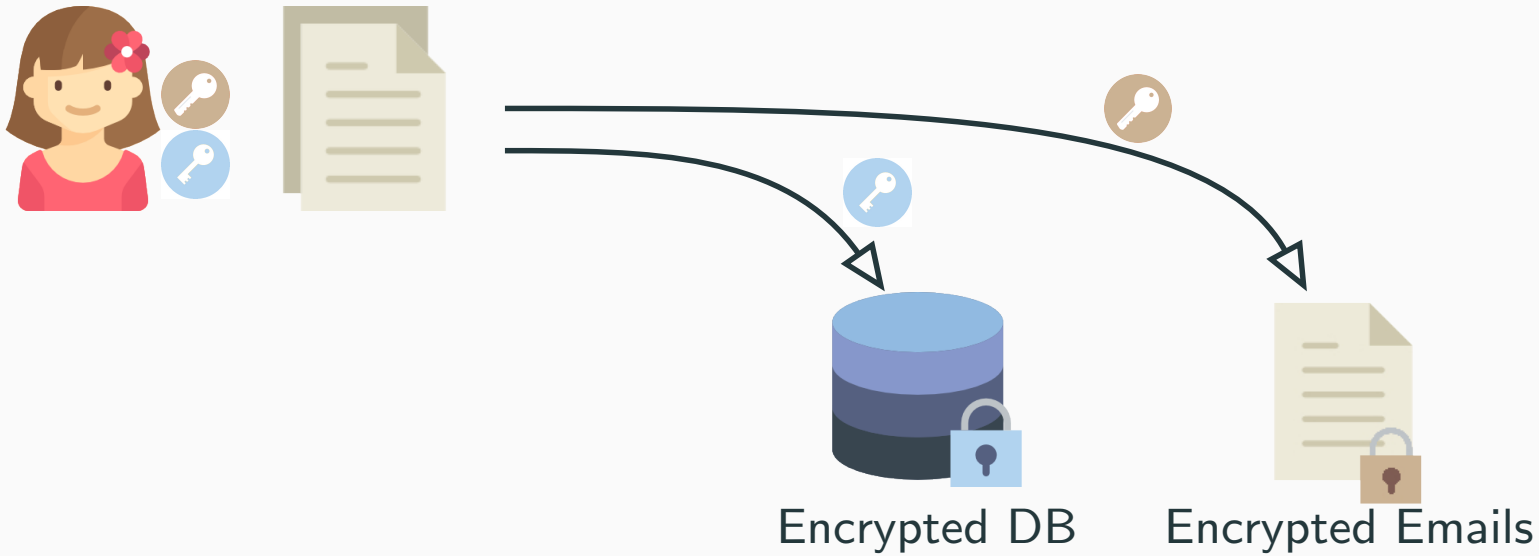
keyword	documents
hello	id_1 id_2
world	id_1
you	id_2

Encrypted Database

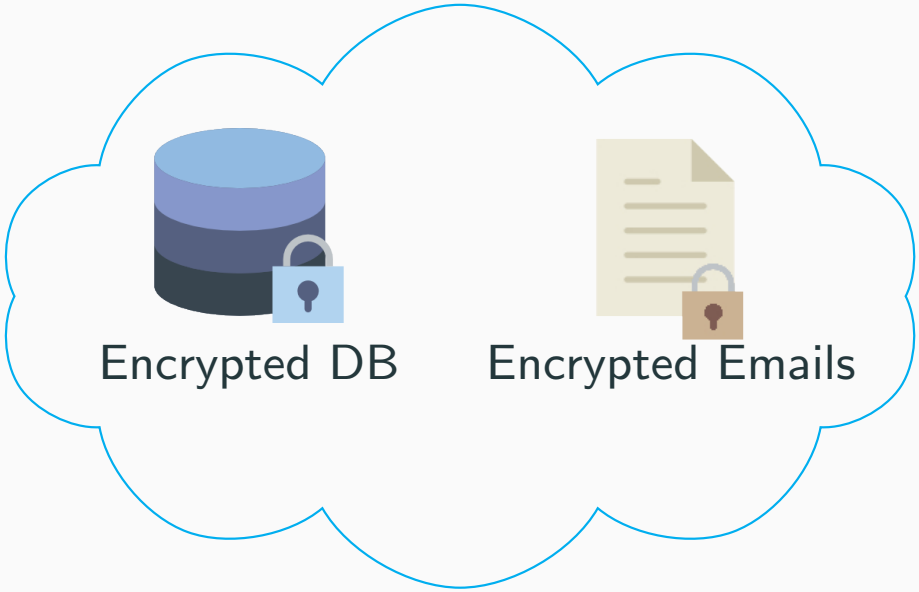
token	documents
hello*	id_1 id_2
world*	id_1
you*	id_2

What is a Searchable Encryption Scheme?

What is a Searchable Encryption Scheme?

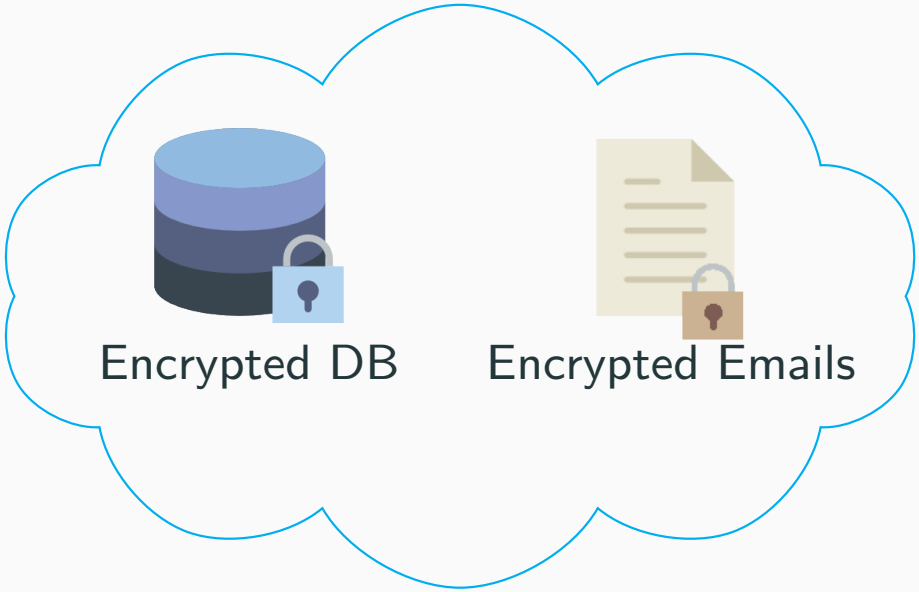


What is a Searchable Encryption Scheme?

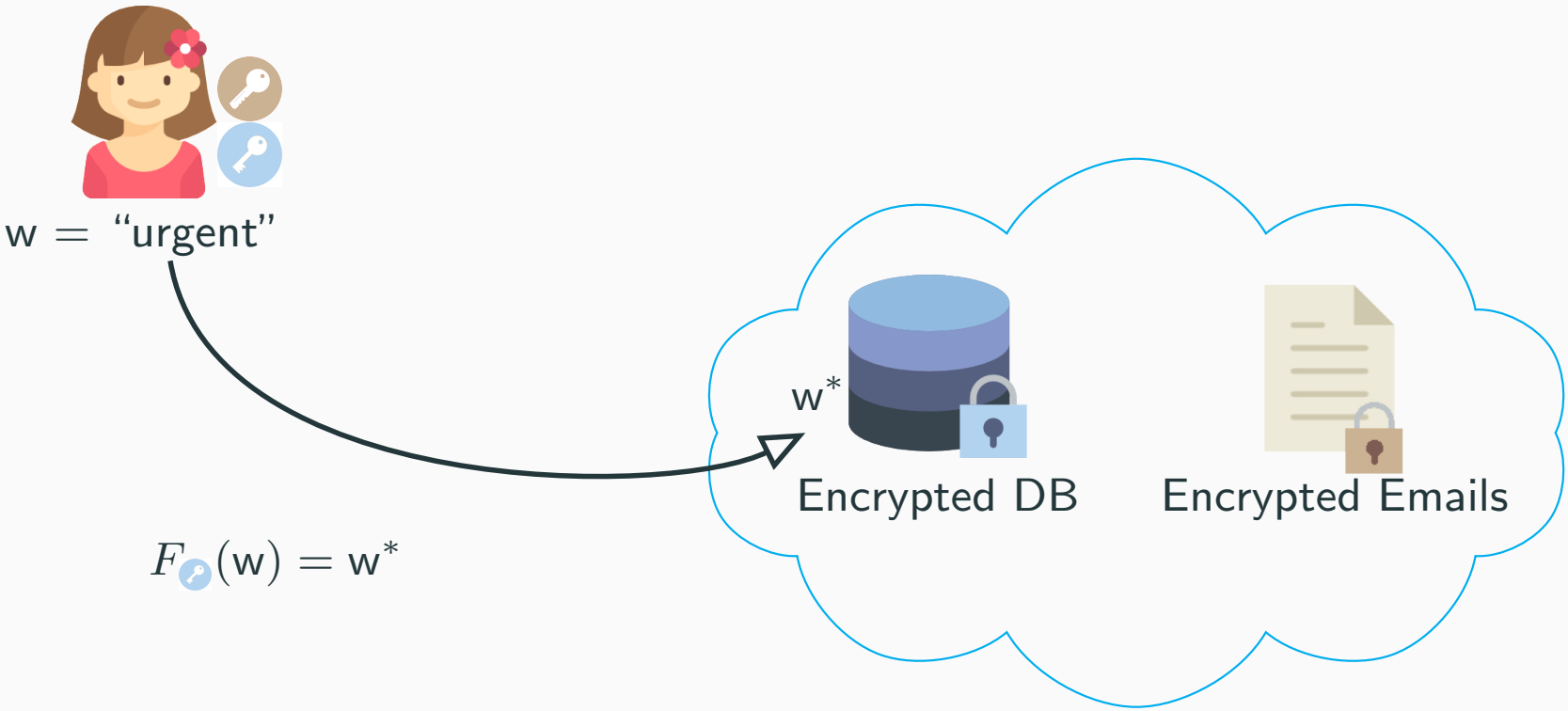


What is a Searchable Encryption Scheme?

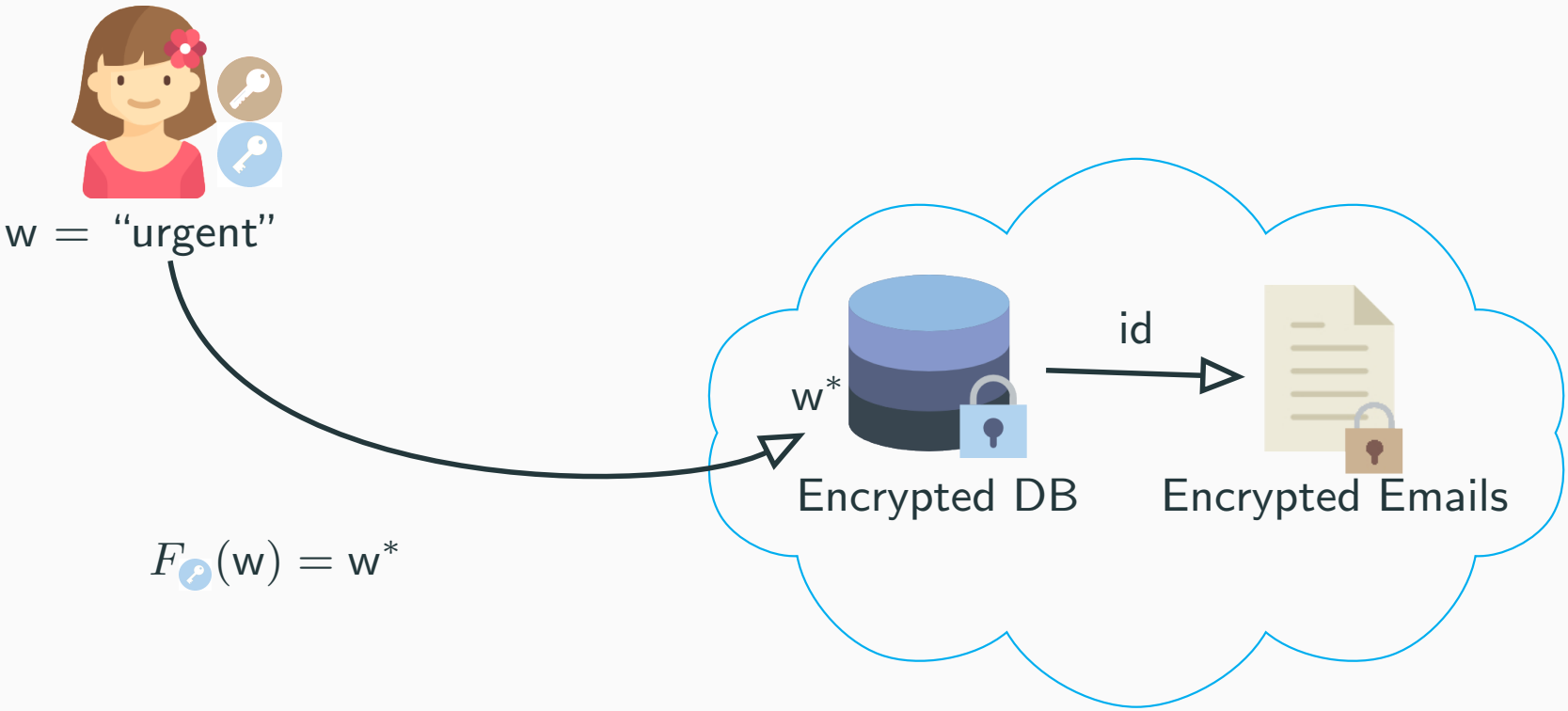
w = “urgent”



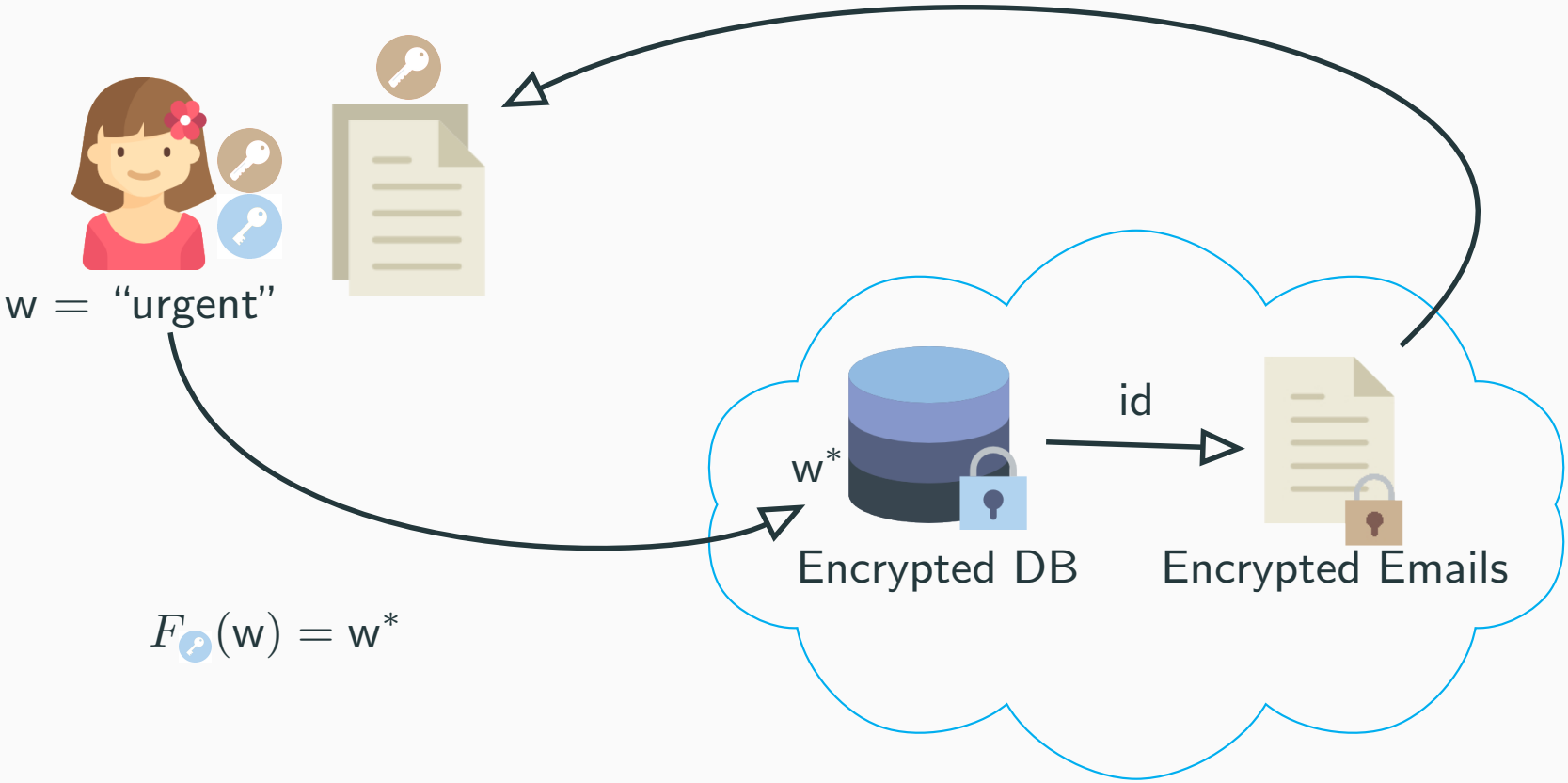
What is a Searchable Encryption Scheme?



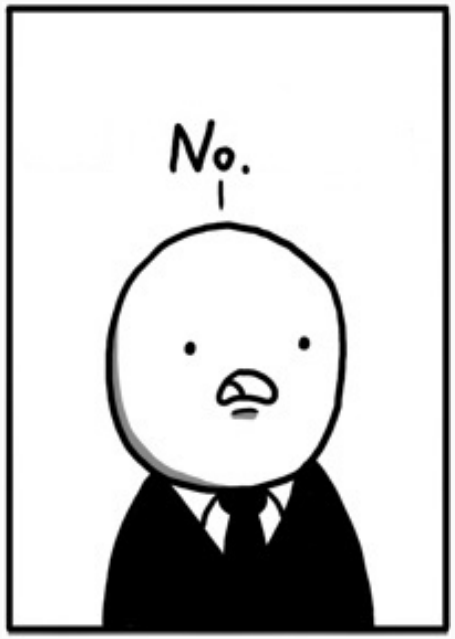
What is a Searchable Encryption Scheme?



What is a Searchable Encryption Scheme?



Are the SSE Really Secure?



Are the SSE Really Secure? (No)

Example

- $d_1 =$ "Where is Clermontech ?"
- $d_2 =$ "Hello Clermontech"
- $d_3 =$ "Clermontech is in Clermont"
- $d_4 =$ "Hello Clermontech and Clermont"

Are the SSE Really Secure? (No)

Example

- $d_1 =$ "Where is Clermontech ?"
- $d_2 =$ "Hello Clermontech"
- $d_3 =$ "Clermontech is in Clermont"
- $d_4 =$ "Hello Clermontech and Clermont"

token	docs
where*	id_1
is*	id_1 id_3
clermontech*	id_1 id_2 id_3 id_4
hello*	id_2 id_4
in*	id_3
clermont*	id_3 id_4
and*	id_4

Are the SSE Really Secure? (No)

Example

- $d_1 =$ "Where is Clermontech ?"
- $d_2 =$ "Hello Clermontech"
- $d_3 =$ "Clermontech is in Clermont"
- $d_4 =$ "Hello Clermontech and Clermont"

token	docs
where*	id_1
is*	id_1 id_3
clermontech*	id_1 id_2 id_3 id_4
hello*	id_2 id_4
in*	id_3
clermont*	id_3 id_4
and*	id_4

docs	tokens
id_1	clermontech* is* where*
id_2	clermontech* hello*
id_3	clermont* clermontech* in* is*
id_4	and* clermont* clermontech* hello*

Are the SSE Really Secure? (No)

Example

1. Assume the Cloud knows:
    - $d_1 =$ "Where is Clermontech ?"
    - $d_2 =$ "Hello Clermontech"
    - $d_3 =$ "Clermontech is in Clermont"

Are the SSE Really Secure? (No)

Example

1. Assume the Cloud knows:
   - $d_1 =$ "Where is Clermontech ?"
   - $d_2 =$ "Hello Clermontech"
   - $d_3 =$ "Clermontech is in Clermont"
2. Cloud computes: $(\#d_1 \cap d_2, \#d_1 \cap d_3, \#d_2 \cap d_3)$

Are the SSE Really Secure? (No)

Example

1. Assume the Cloud knows:
   - $d_1 =$ "Where is Clermontech ?"
   - $d_2 =$ "Hello Clermontech"
   - $d_3 =$ "Clermontech is in Clermont"
2. Cloud computes: $(\#d_1 \cap d_2, \#d_1 \cap d_3, \#d_2 \cap d_3) = (1, 2, 1)$

Are the SSE Really Secure? (No)

Example

1. Assume the Cloud knows:
   - $d_1 =$ "Where is Clermontech ?"
   - $d_2 =$ "Hello Clermontech"
   - $d_3 =$ "Clermontech is in Clermont"
2. Cloud computes: $(\#d_1 \cap d_2, \#d_1 \cap d_3, \#d_2 \cap d_3) = (1, 2, 1)$
3. Using

docs	tokens
id_1	$\mathsf{clermontech}^*$ is^* where^*
id_2	$\mathsf{clermontech}^*$ hello^*
id_3	$\mathsf{clermont}^*$ $\mathsf{clermontech}^*$ in^* is^*
id_4	and^* $\mathsf{clermont}^*$ $\mathsf{clermontech}^*$ hello^*

   The Cloud deduces:
   - $d_1 \leftrightarrow \mathsf{id}_1$
   - $d_2 \leftrightarrow \mathsf{id}_2$
   - $d_3 \leftrightarrow \mathsf{id}_3$

Are the SSE Really Secure? (No)

Example

The Cloud knows:

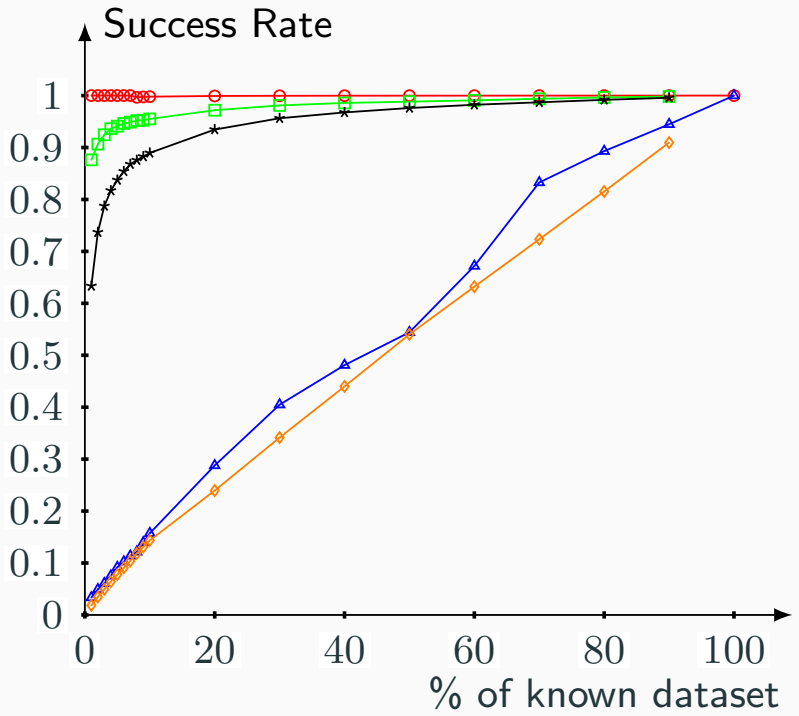
- $d_1 =$ "Where is Clermontech ?"
- $d_2 =$ "Hello Clermontech"
- $d_3 =$ "Clermontech is in Clermont"

4. "Clermontech" is the *only* keyword shared between these **3** documents
5. Using

docs	tokens
id_1	clermontech* is* where*
id_2	clermontech* hello*
id_3	clermont* clermontech* in* is*
id_4	and* clermont* clermontech* hello*

The Cloud deduces: clermontech$^* \leftrightarrow$ clermontech

In Practice (Gutenberg)



Keyword/token associations over $\mathcal{I}$

Keyword/token associations over the dataset

Document recovered >80%

Document recovered >90%

Documents completely recovered

Conclusion

Thank you for your attention.

matthieu.giraud@uca.fr
http://sancy.univ-bpclermont.fr/~giraud/