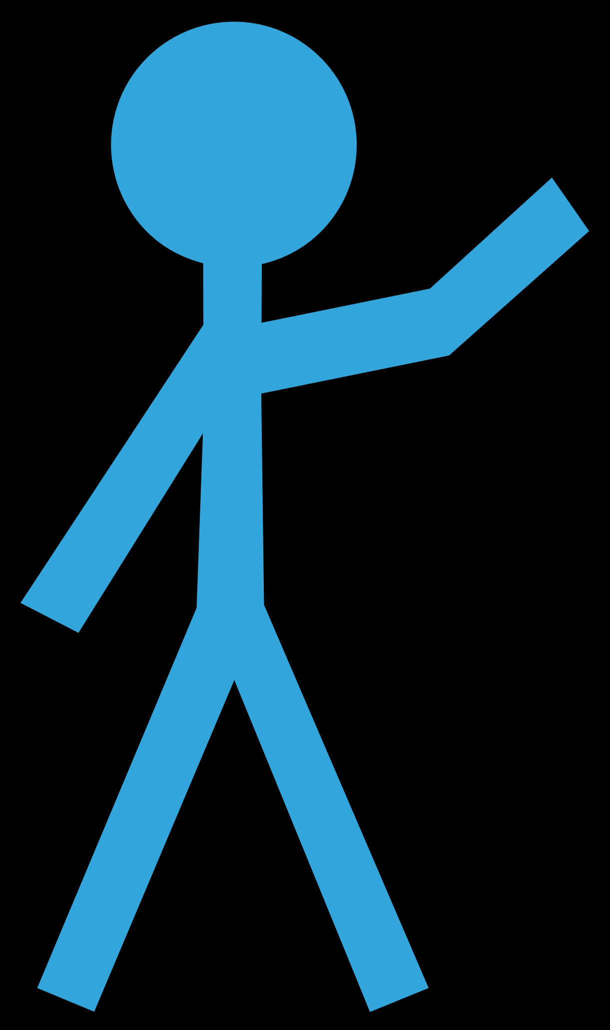
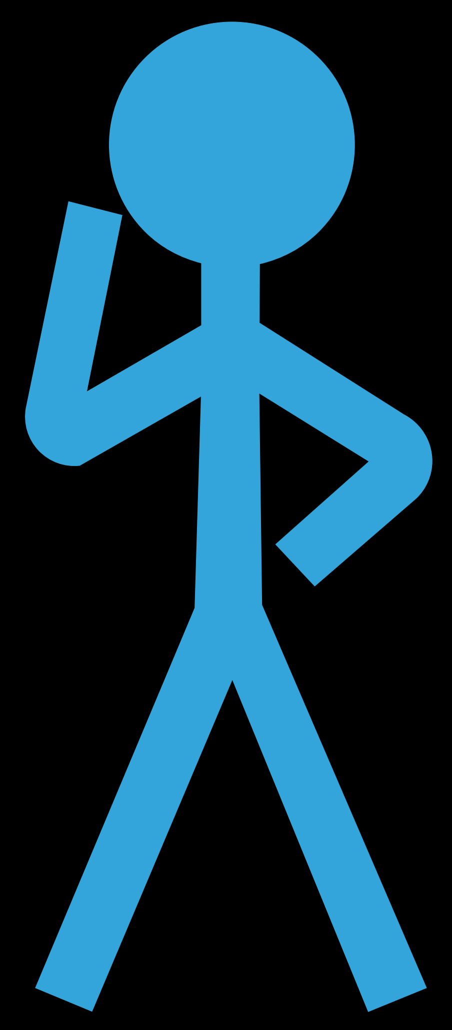
Reasoning about Aggregation of Information in Timing Attacks

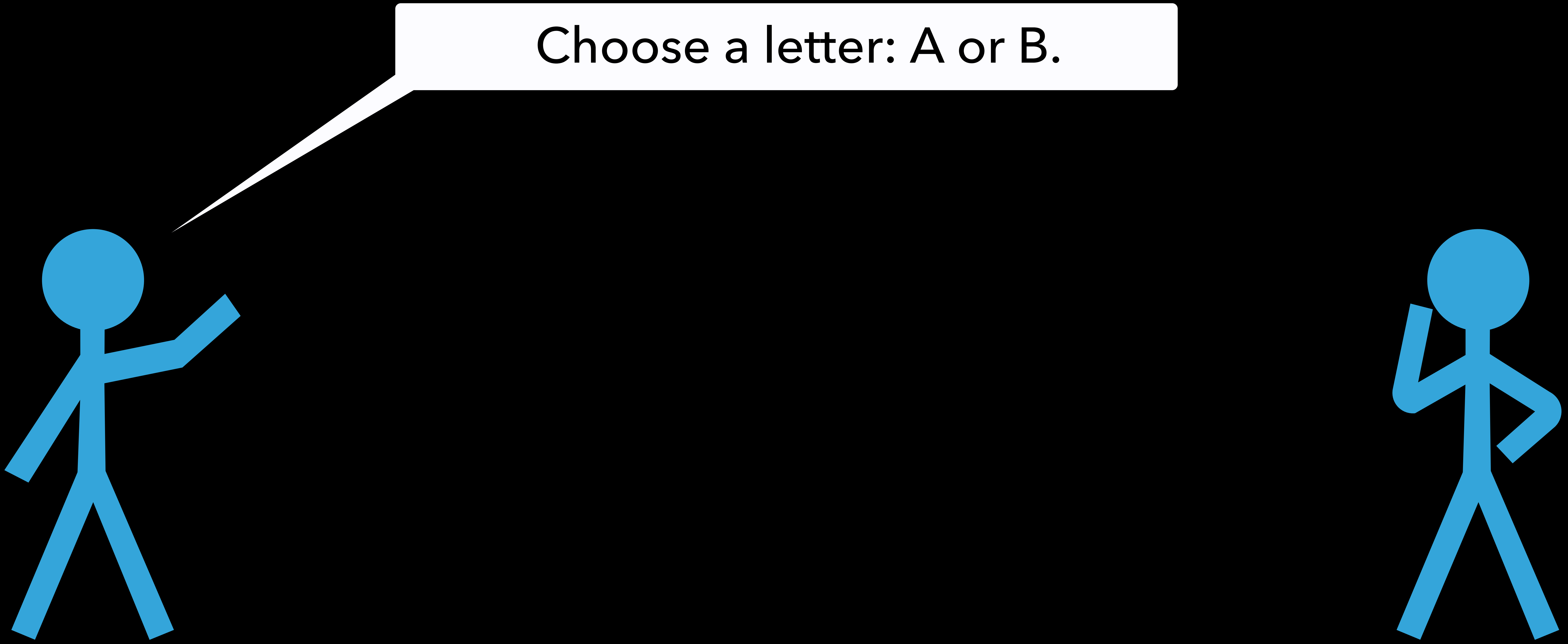
Itsaka Rakotonirina

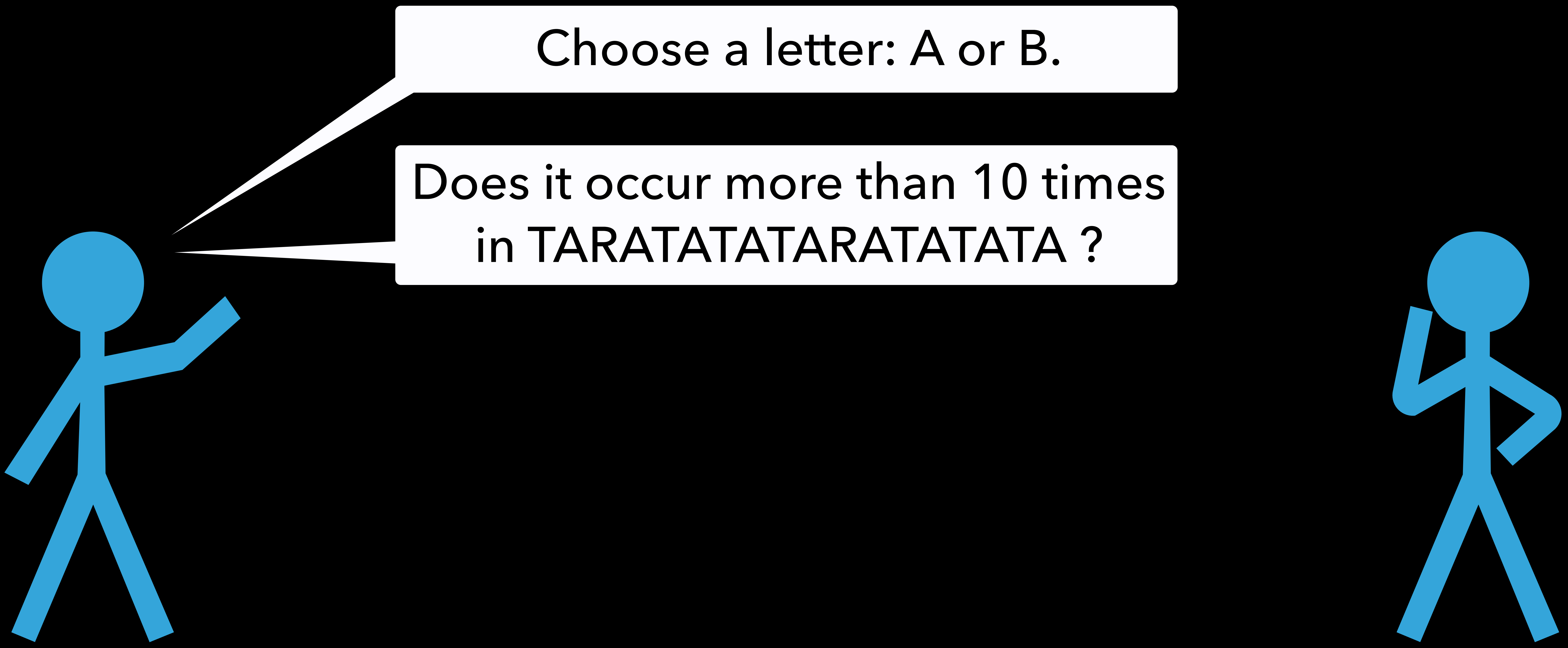
INRIA Nancy Grand-Est

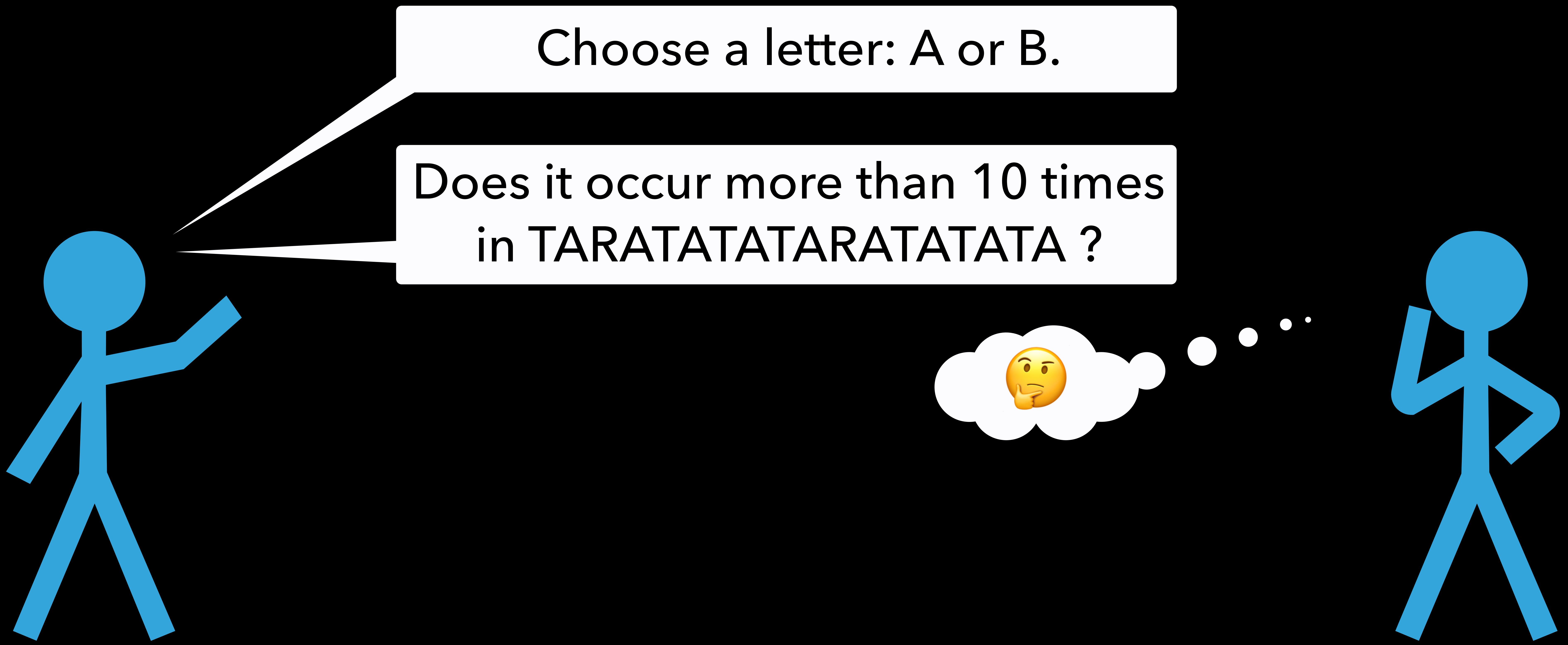
Boris Köpf

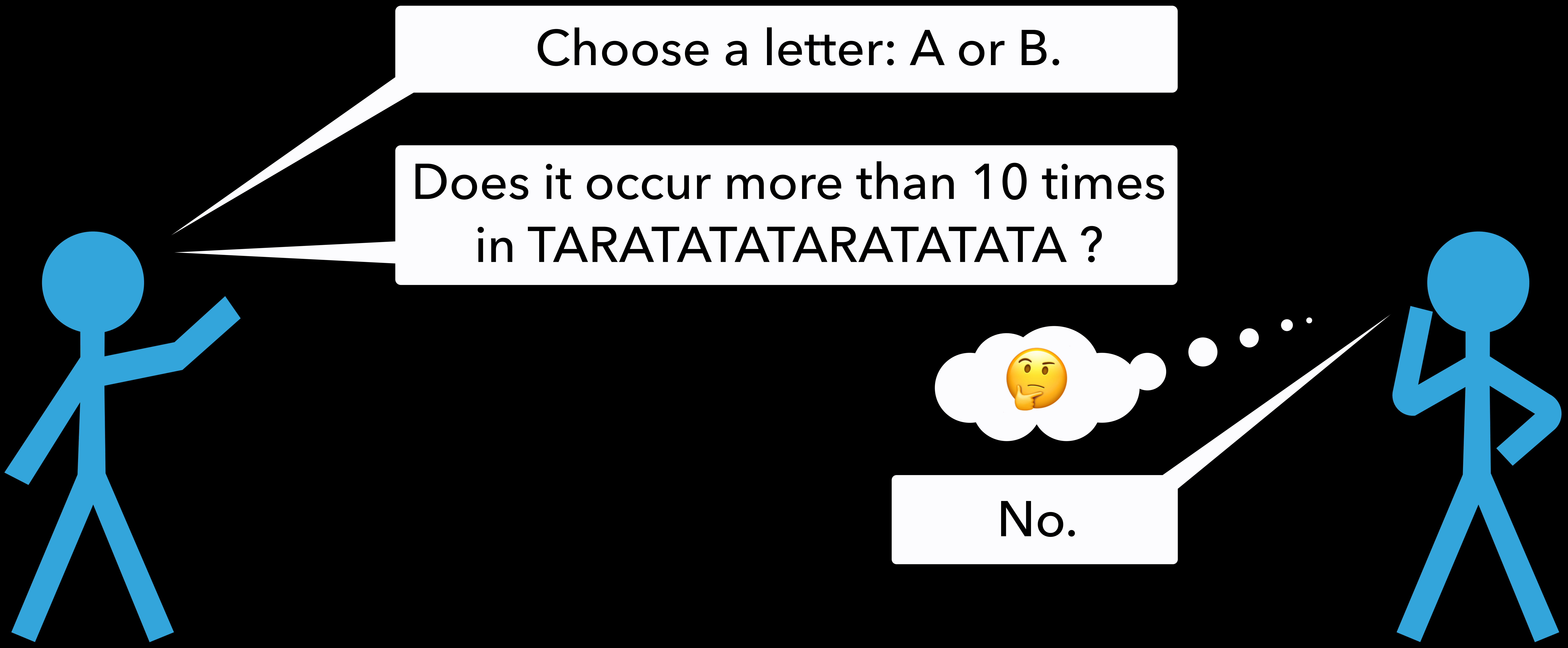
Microsoft Research

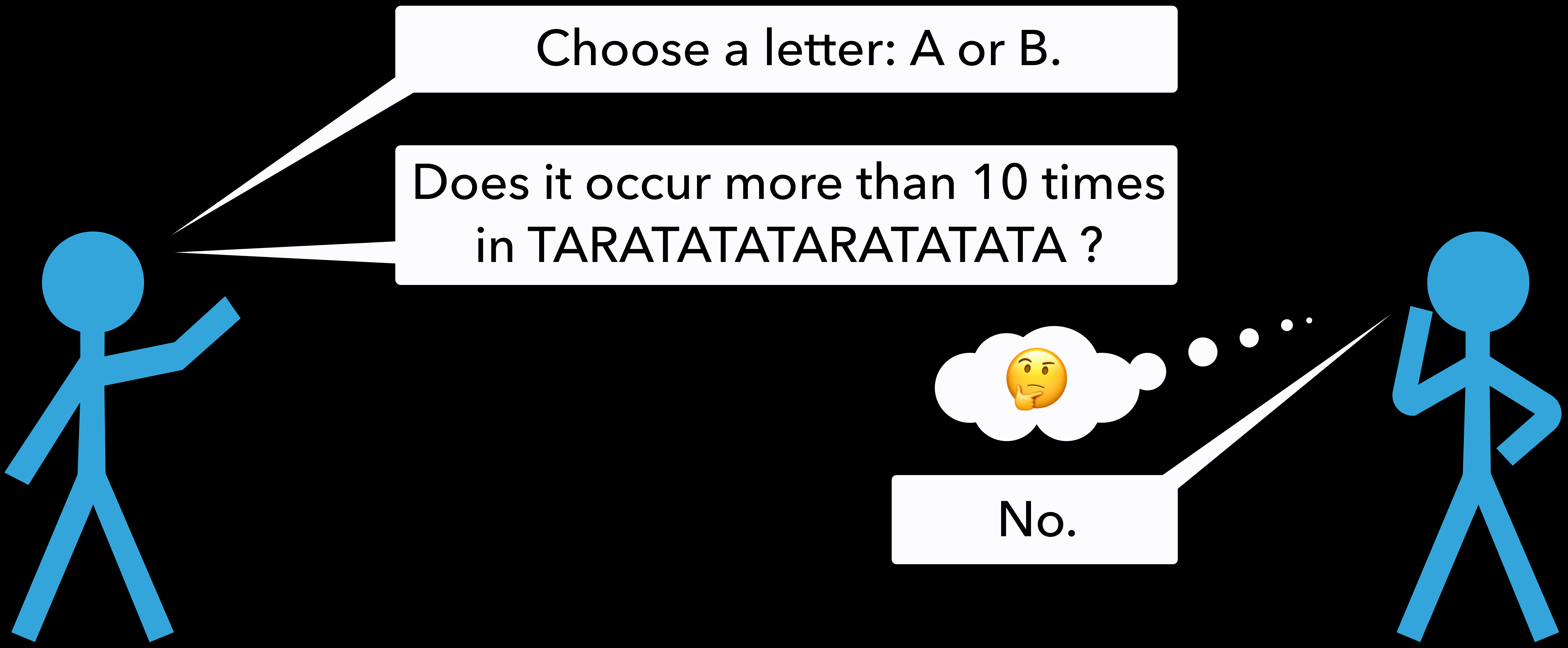




Choose a letter: A or B.


Choose a letter: A or B.
Does it occur more than 10 times in TARATATATARATATATA ?


Choose a letter: A or B.
Does it occur more than 10 times in TARATATATARATATATA ?


Choose a letter: A or B.
Does it occur more than 10 times in TARATATATARATATATA ?
No.



Q : Which letter was chosen?

Timing attacks

1996 on RSA (Kocher)

1998 on RSA (Dhem *et al.*)

2005 on AES (Bernstein)

2007 on AES (Acıiçmez *et al.*)

2013 Lucky Thirteen (AlFardan, Paterson)

2014 Flush+Reload (Yarom, Falkner)

2016 on ECDH (Kaufmann *et al.*)

2018 Spectre (Kocher *et al.*)

2018 Meltdown (Lipp *et al.*)

2019 RIDL (van Schaik *et al.*)

2019 ZombieLoad (Schwarz *et al.*)

⋮

Timing attacks

1996 on RSA (Kocher)

1998 on RSA (Dhem *et al.*)

2005 on AES (Bernstein)

2007 on AES (Acıiçmez *et al.*)

2013 Lucky Thirteen (AlFardan, Paterson)

2014 Flush+Reload (Yarom, Falkner)

2016 on ECDH (Kaufmann *et al.*)

2018 Spectre (Kocher *et al.*)

2018 Meltdown (Lipp *et al.*)

2019 RIDL (van Schaik *et al.*)

2019 ZombieLoad (Schwarz *et al.*)

⋮

A long-term secret, and queries to an oracle
O : *public input* $\mapsto$ *execution time of a program*

Remote measurements

Timing attacks

1996 on RSA (Kocher)

1998 on RSA (Dhem *et al.*)

2005 on AES (Bernstein)

2007 on AES (Acıiçmez *et al.*)

2013 Lucky Thirteen (AlFardan, Paterson)

2014 Flush+Reload (Yarom, Falkner)

2016 on ECDH (Kaufmann *et al.*)

2018 Spectre (Kocher *et al.*)

2018 Meltdown (Lipp *et al.*)

2019 RIDL (van Schaik *et al.*)

2019 ZombieLoad (Schwarz *et al.*)

⋮

A long-term secret, and queries to an oracle
O : *public input* $\mapsto$ *execution time of a program*

Remote measurements

Exploit timing variations, and not the absolute execution time

Differential measurements

Timing attacks

1996 on RSA (Kocher)

1998 on RSA (Dhem *et al.*)

2005 on AES (Bernstein)

2007 on AES (Acıiçmez *et al.*)

2013 Lucky Thirteen (AlFardan, Paterson)

2014 Flush+Reload (Yarom, Falkner)

2016 on ECDH (Kaufmann *et al.*)

2018 Spectre (Kocher *et al.*)

2018 Meltdown (Lipp *et al.*)

2019 RIDL (van Schaik *et al.*)

2019 ZombieLoad (Schwarz *et al.*)

⋮

A long-term secret, and queries to an oracle
O : *public input* $\mapsto$ *execution time of a program*

Remote measurements

Exploit timing variations, and not the absolute execution time

Differential measurements

The secret is recovered chunk by chunk

Compositionality

Timing attacks

Attacker model

A long-term secret, and queries to an oracle
O : public input $\mapsto$ *execution time of a program*

Remote measurements

Exploit timing variations, and not the absolute execution time

Differential measurements

Under what hypotheses?

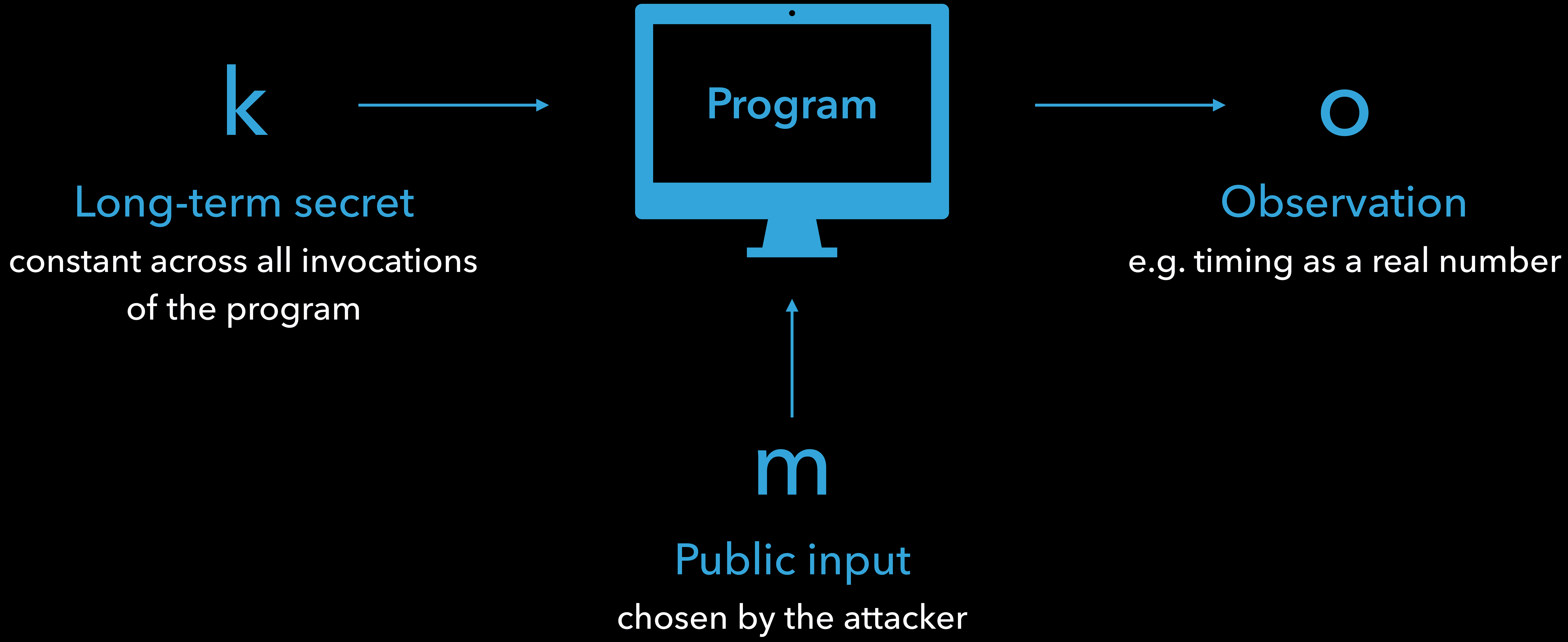
The secret is recovered chunk by chunk

Compositionality

Contributions

- A model of timing attacks
  capturing the essence of compositional attacks
- Core hypotheses giving rise to efficient attacks
  under the form of independence properties
- Generic attack descriptions + cost analyses

A model for timing leakage


k
Long-term secret
constant across all invocations of the program
Program
o
Observation
e.g. timing as a real number
m
Public input
chosen by the attacker

A simple example

```
for i = 0 to n − 1 do
    if k[i] ≠ m[i] then g()
done
```

A simple example

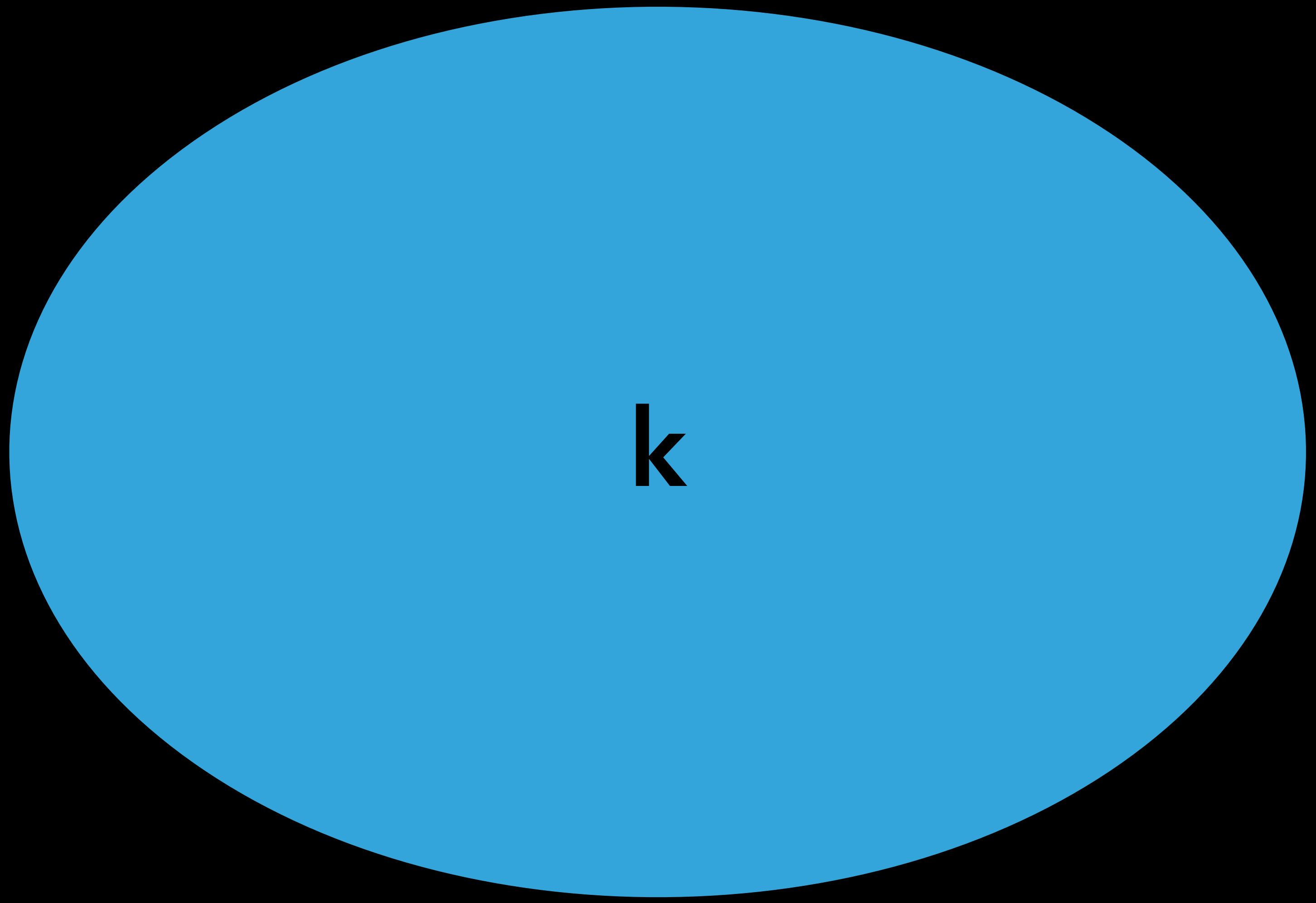
```
for i = 0 to n − 1 do
    if k[i] ≠ m[i] then g()
done
```

execution time proportional to:

$$t(k,m) = \sum_{i=1}^{n} k[i] \oplus m[i] = \text{nb of bits where k and m differ}$$

Hamming distance

Aggregation of information

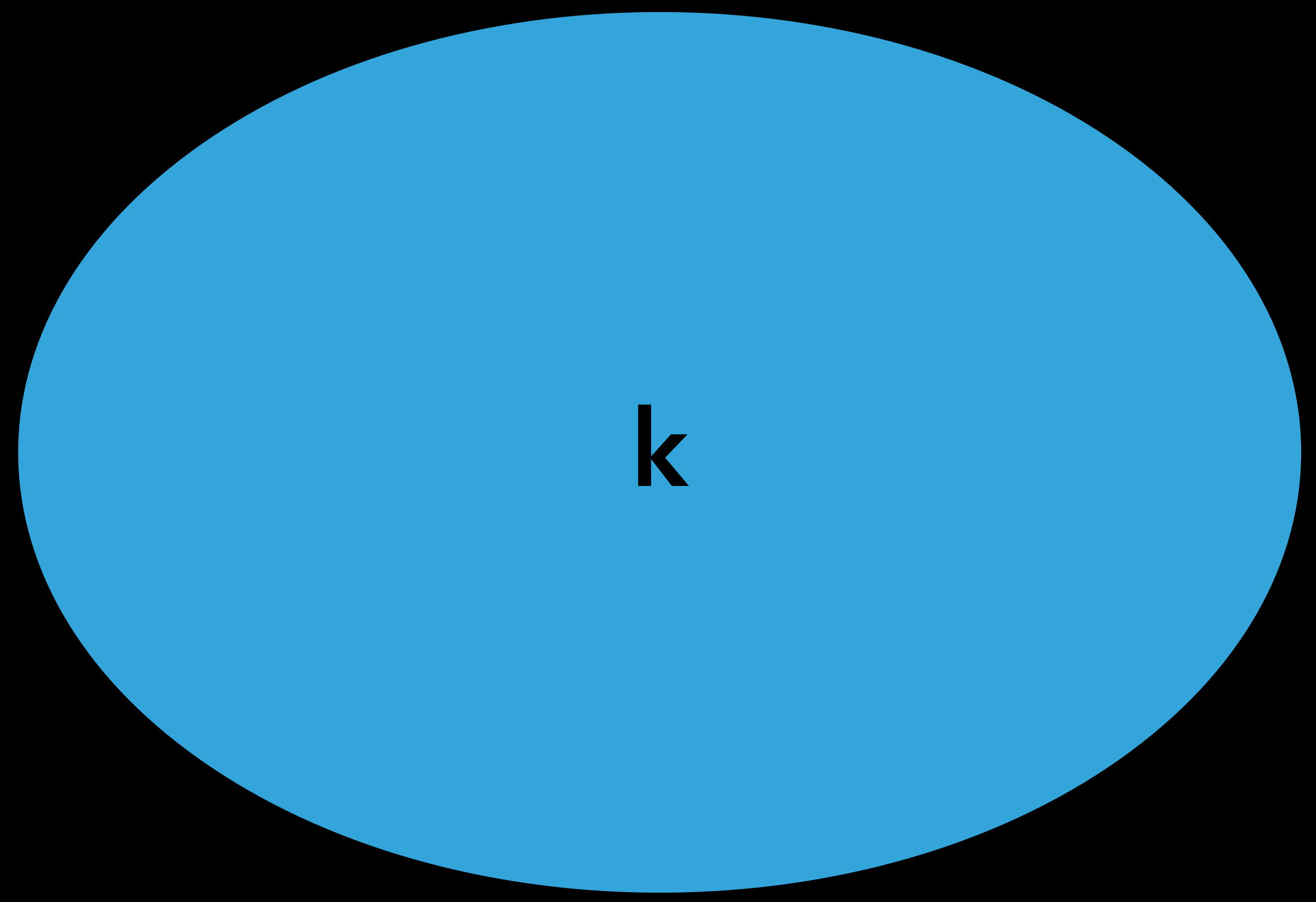


potential values of
the long-term secret

$$t(k,m) = \Sigma_{i=1}^{3}\ k[i] \oplus m[i]$$

Hamming distance

Aggregation of information



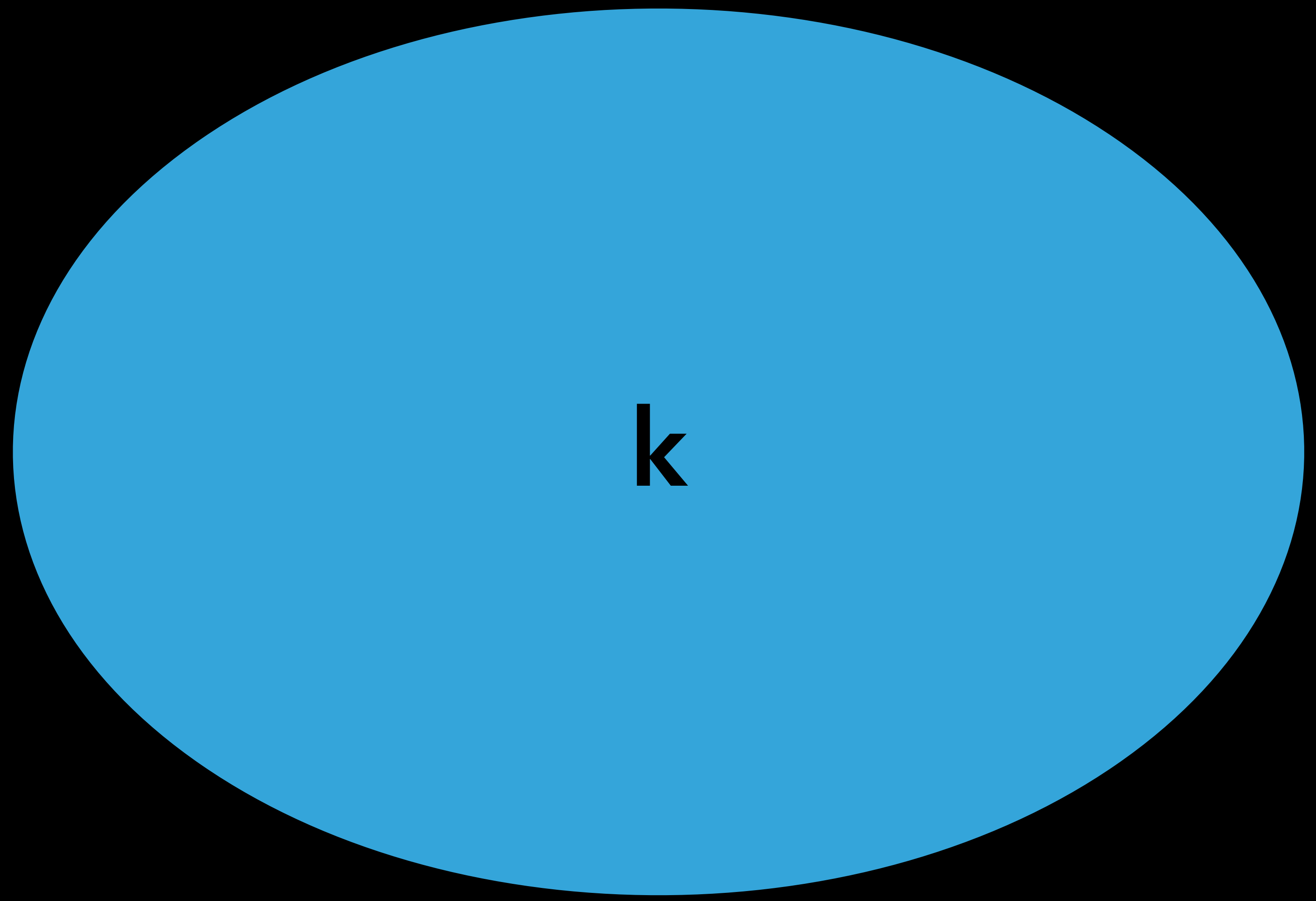
potential values of
the long-term secret

$$t(k,m) = \Sigma_{i=1}^{3}\ k[i] \oplus m[i]$$

Hamming distance

$$000 \longmapsto o = t(k,000) \in \{0,1,2,3\}$$

Aggregation of information



potential values of
the long-term secret

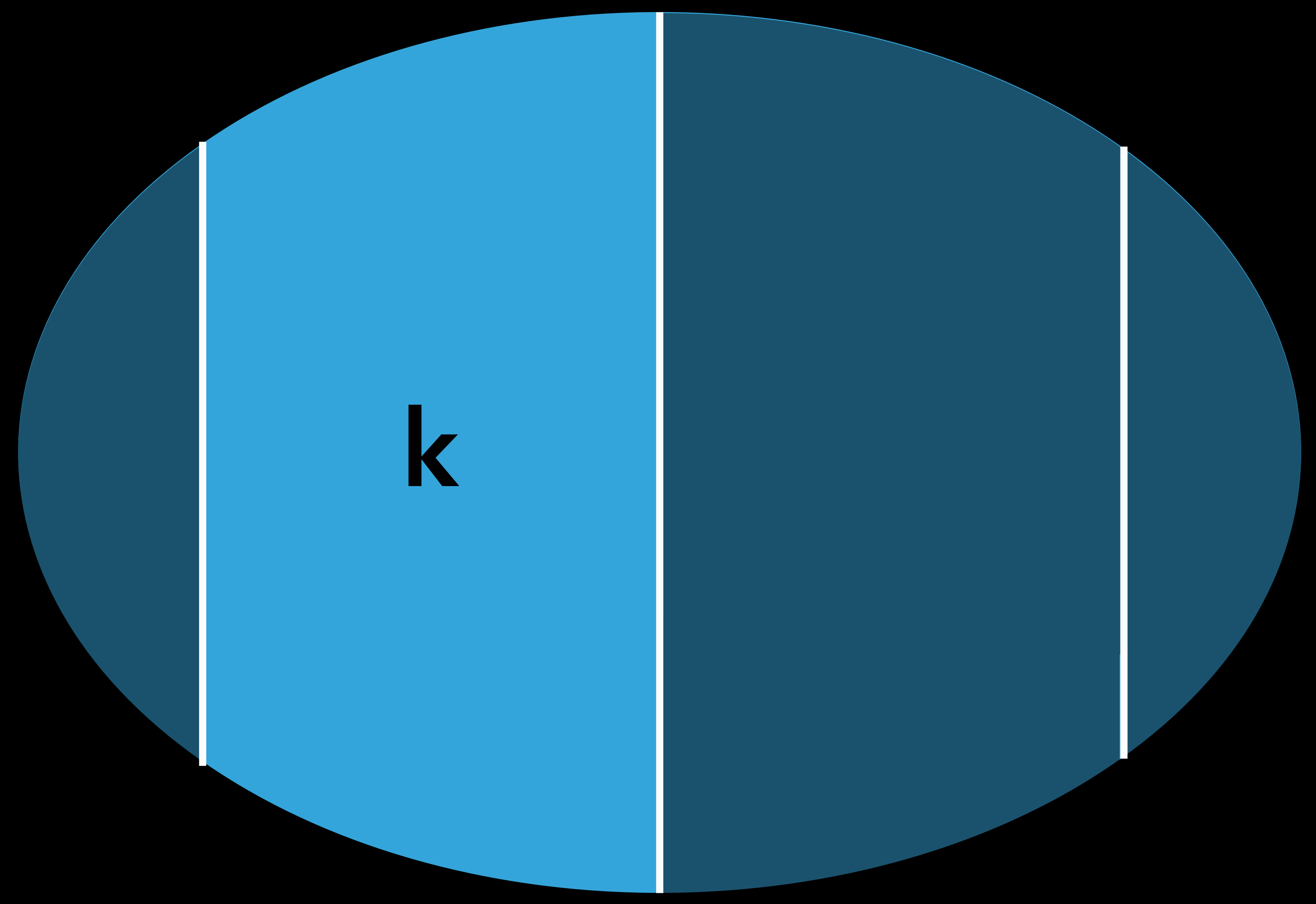
$$t(k,m) = \Sigma_{i=1}^{3}\ k[i] \oplus m[i]$$

Hamming distance

$$000 \longmapsto o \quad = t(k,000) \in \{0,1,2,3\}$$

$$\Rightarrow \quad k \in \{\, k' \mid t(k',000) = o \,\}$$

Aggregation of information



potential values of
the long-term secret

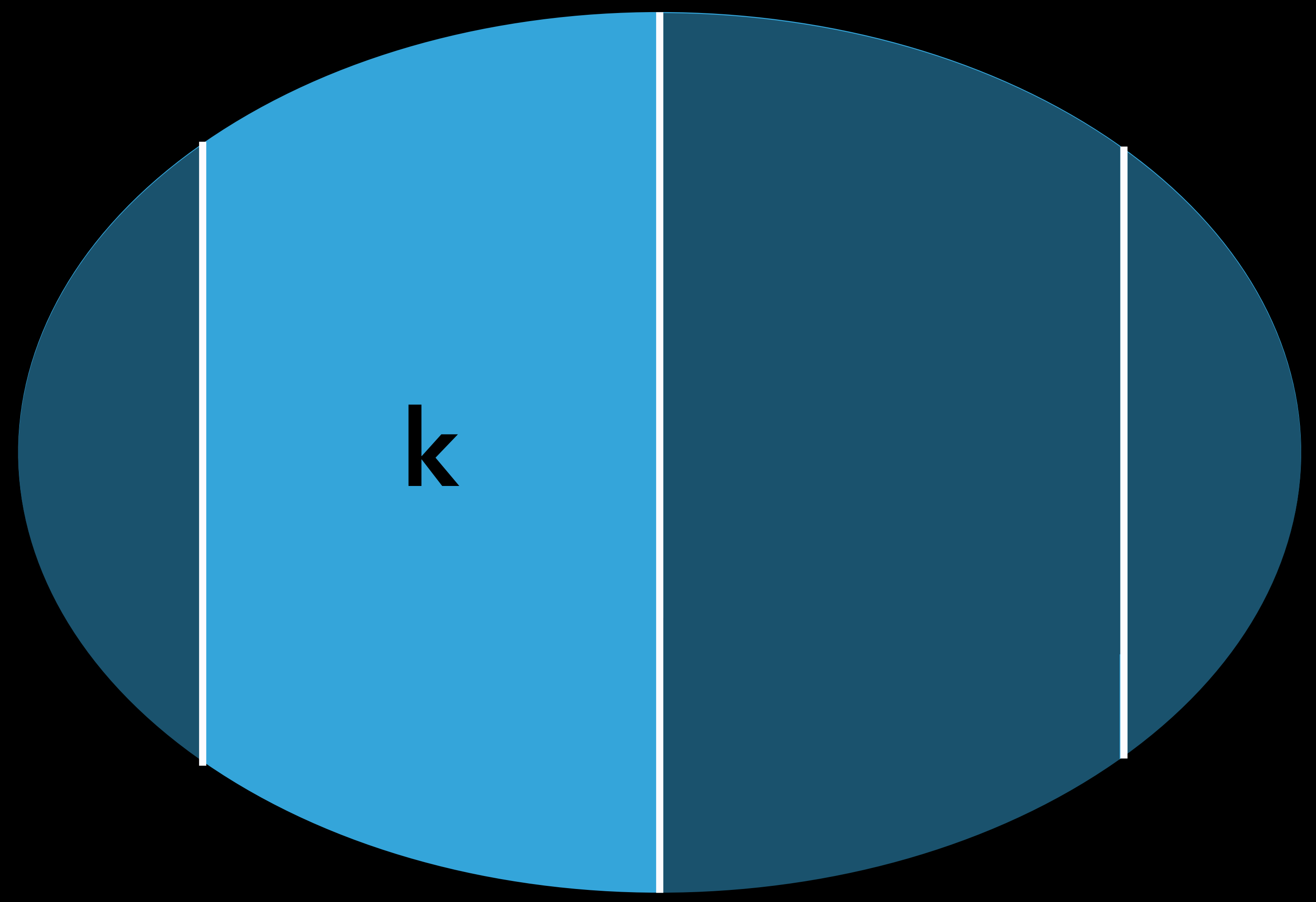
$$t(k,m) = \Sigma_{i=1}^{3}\ k[i] \oplus m[i]$$

Hamming distance

$$000 \longmapsto o = t(k,000) \in \{0,1,2,3\}$$

$$\Rightarrow\ k \in \{\, k' \mid t(k',000) = o \,\}$$

Aggregation of information



potential values of
the long-term secret

$$t(k,m) = \Sigma_{i=1}^{3}\ k[i] \oplus m[i]$$

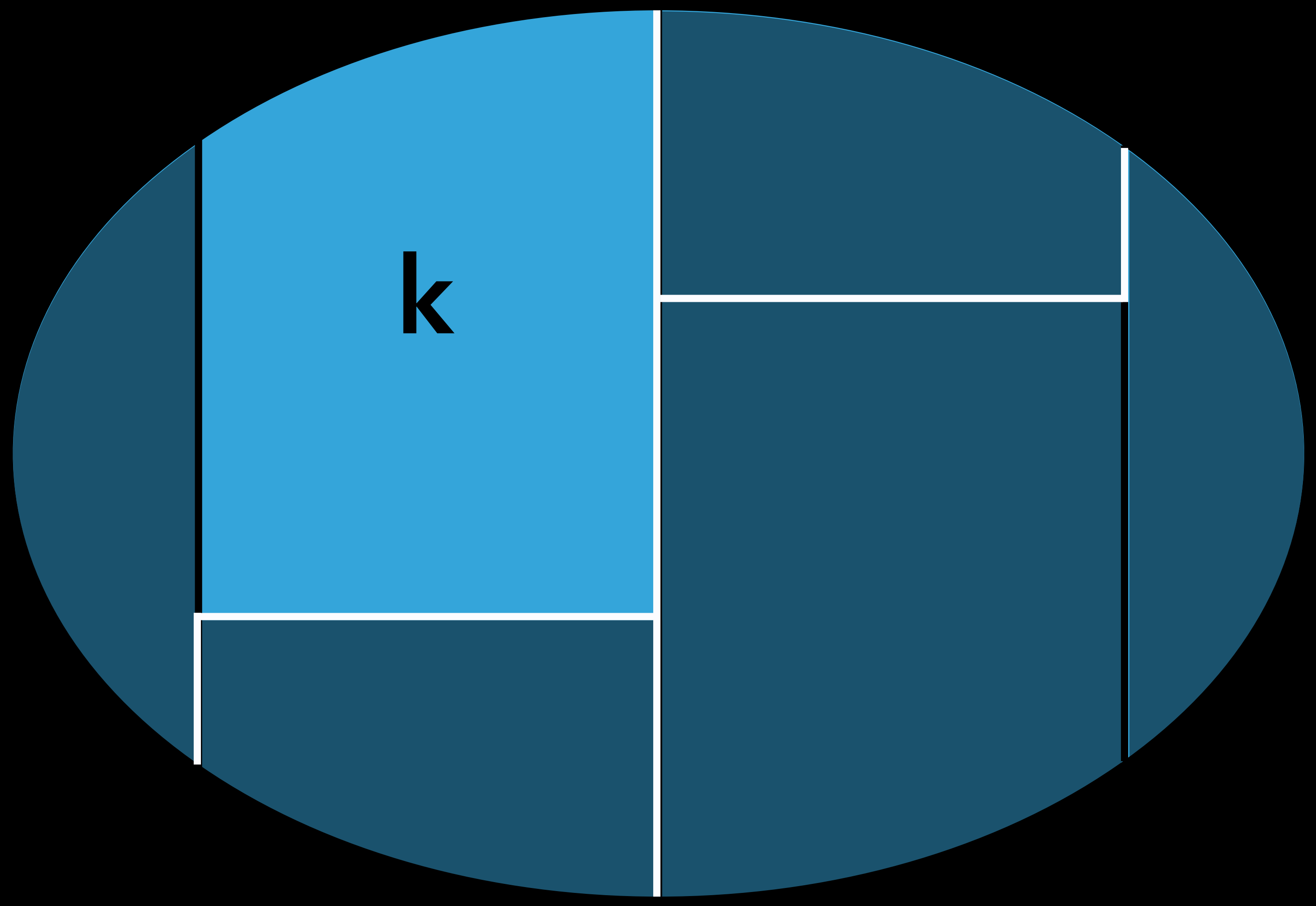
Hamming distance

$$000 \longmapsto o = t(k,000) \in \{0,1,2,3\}$$

$$\Rightarrow\ k \in \{\, k' \mid t(k',000) = o \,\}$$

$$001 \longmapsto o' = t(k,001)$$

Aggregation of information



potential values of
the long-term secret

$$t(k,m) = \Sigma_{i=1}^{3}\ k[i] \oplus m[i]$$

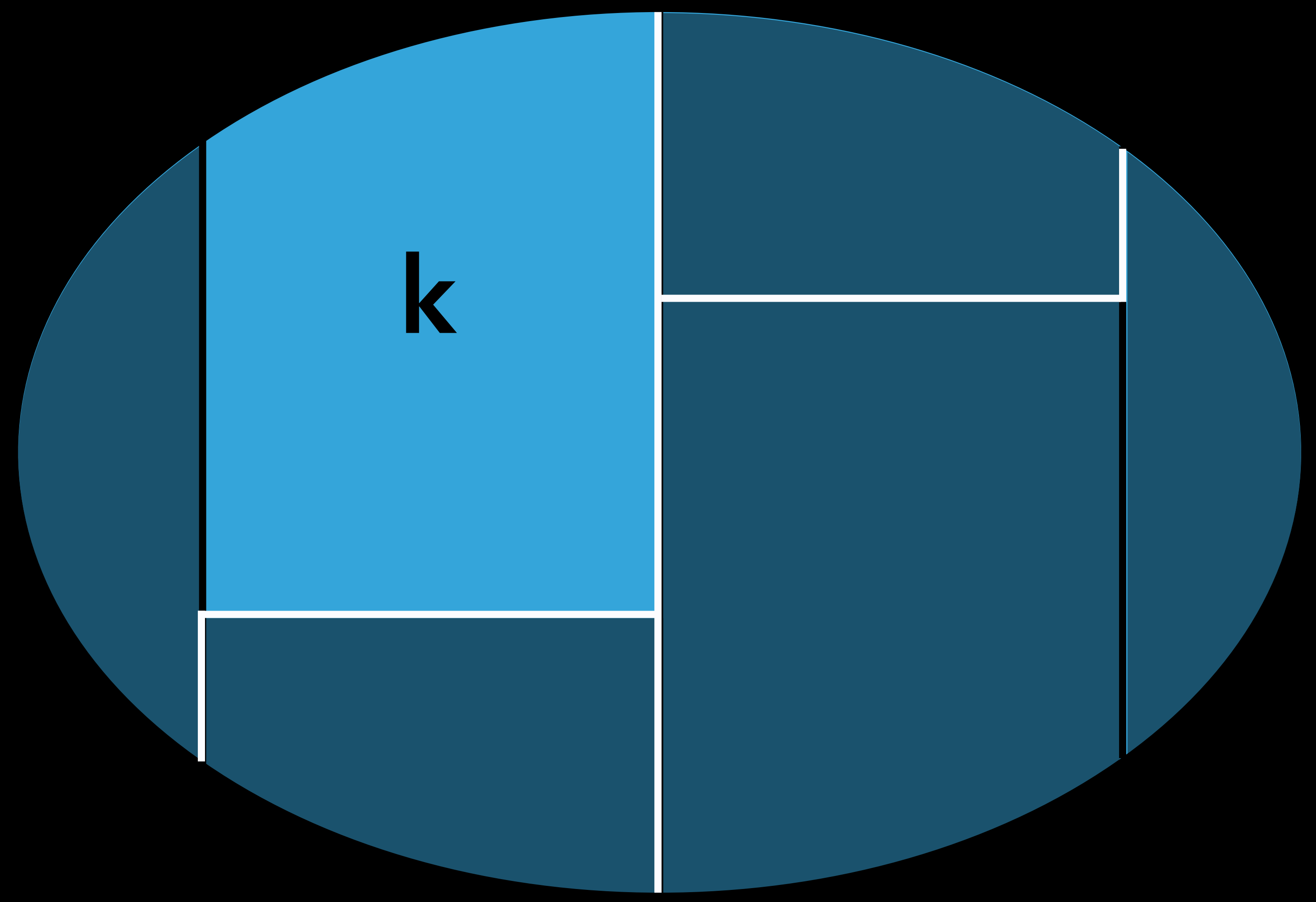
Hamming distance

$000 \longmapsto o = t(k,000) \in \{0,1,2,3\}$

$\Rightarrow\ k \in \{\, k' \mid t(k',000) = o \,\}$

$001 \longmapsto o' = t(k,001)$

Aggregation of information



potential values of
the long-term secret

$$t(k,m) = \Sigma_{i=1}^{3}\ k[i] \oplus m[i]$$

Hamming distance

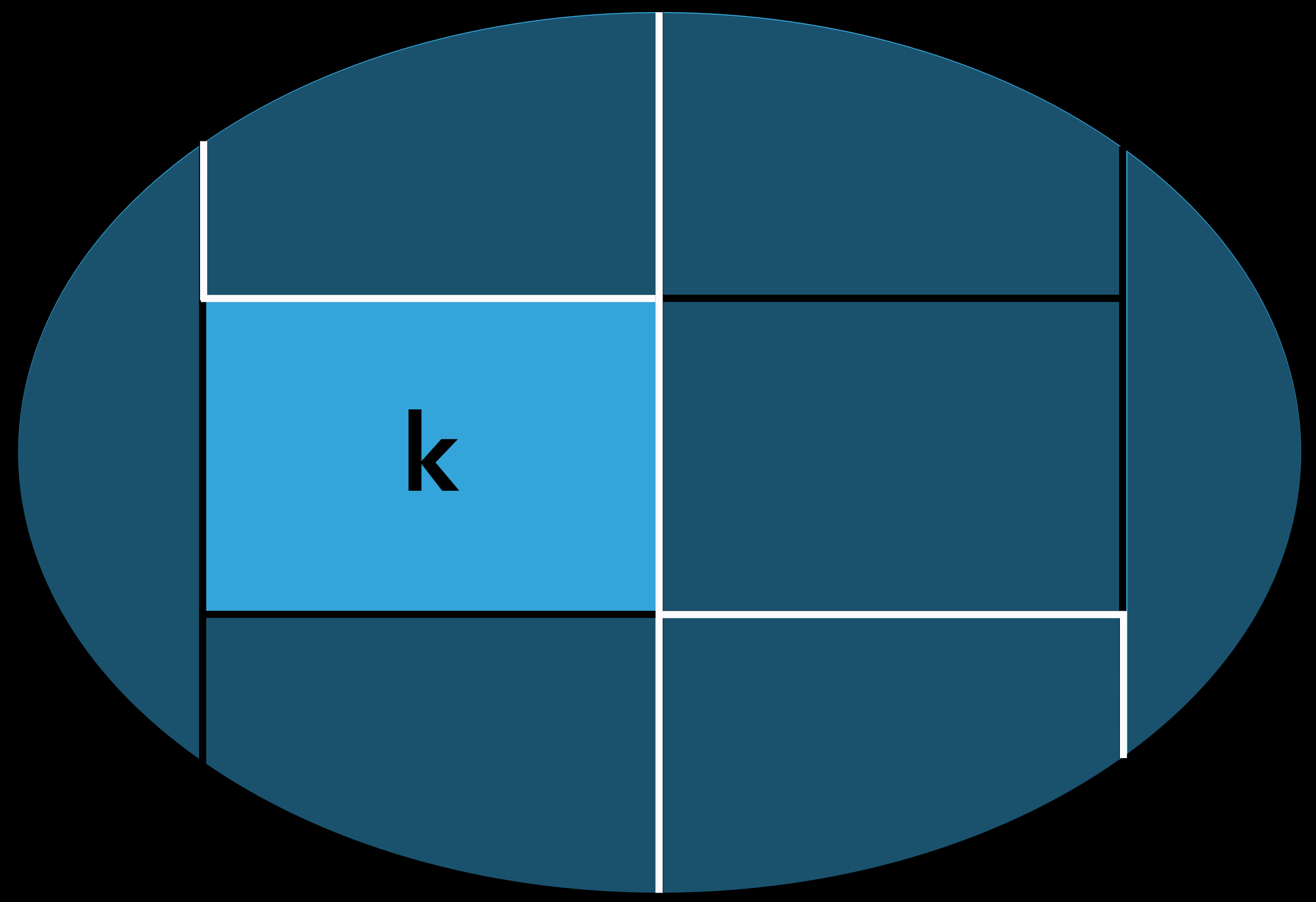
$000 \longmapsto o = t(k,000) \in \{0,1,2,3\}$

$\Rightarrow\ k \in \{\, k' \mid t(k',000) = o \,\}$

$001 \longmapsto o' = t(k,001)$

$010 \longmapsto o'' = t(k,010)$

Aggregation of information



potential values of
the long-term secret

$$t(k,m) = \Sigma_{i=1}^{3} \; k[i] \oplus m[i]$$

Hamming distance

$$000 \longmapsto o = t(k,000) \in \{0,1,2,3\}$$

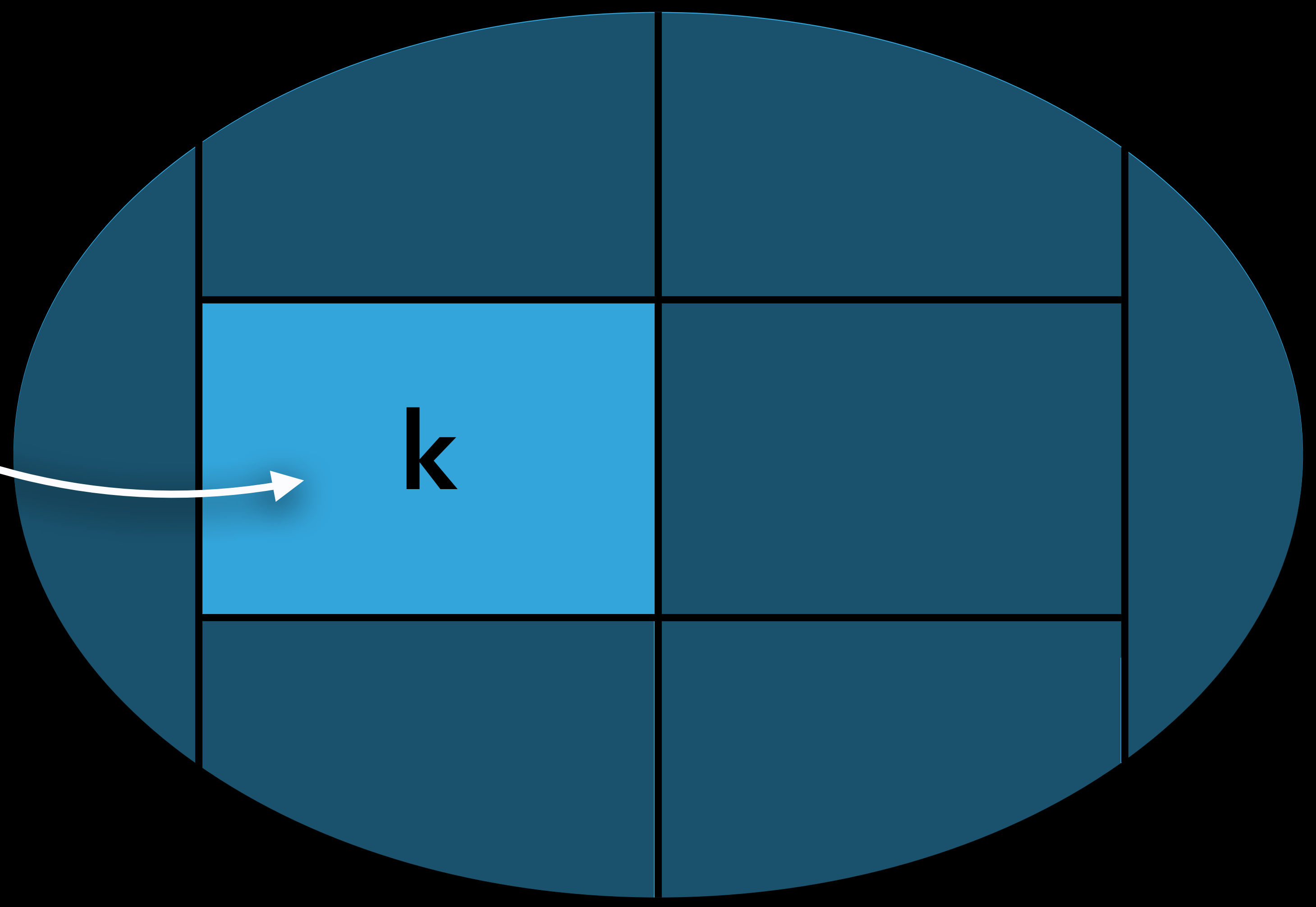
$$\Rightarrow \; k \in \{ k' \mid t(k',000) = o \}$$

$$001 \longmapsto o' = t(k,001)$$

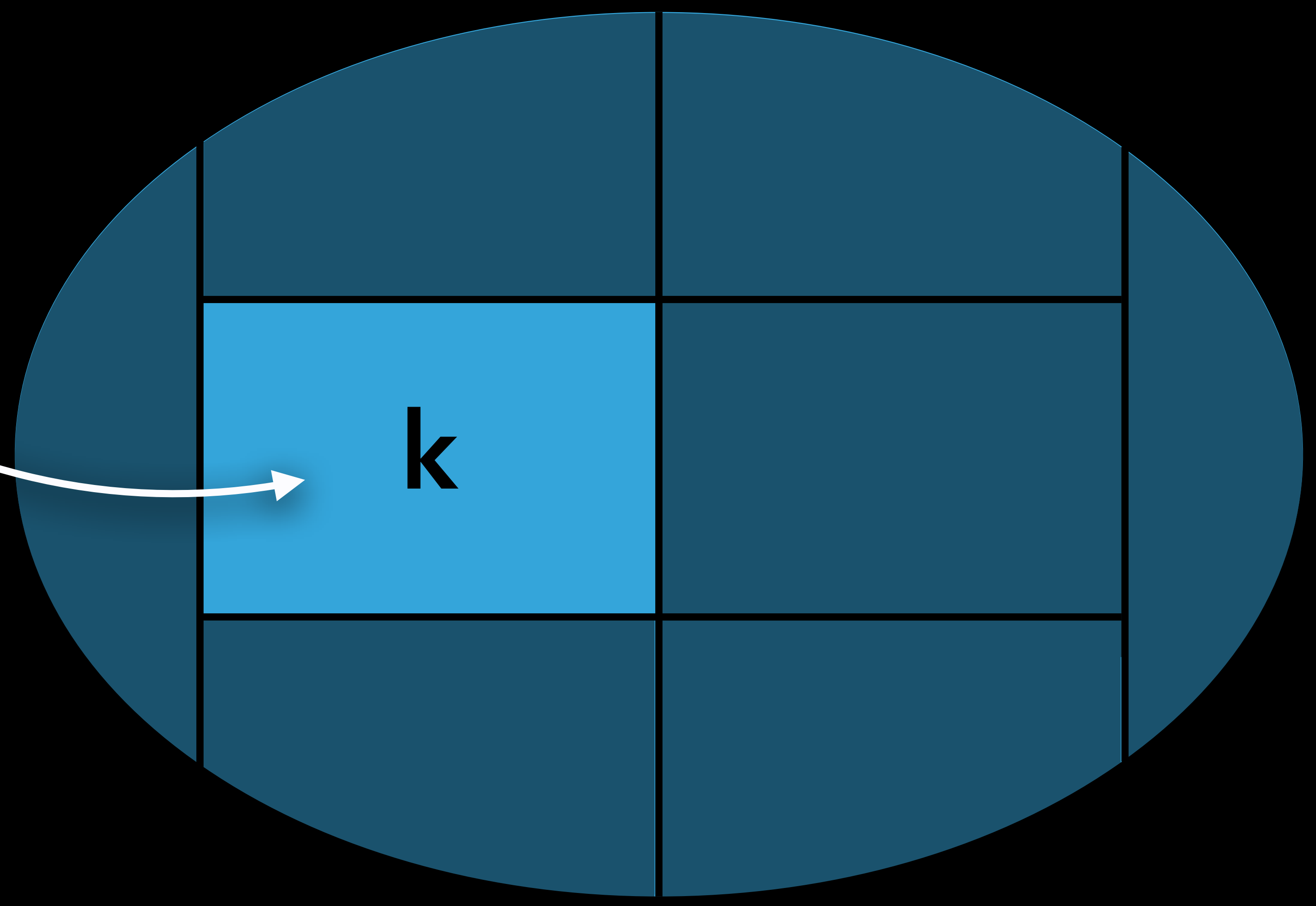
$$010 \longmapsto o'' = t(k,010)$$

Aggregation of information

potential values of
the long-term secret



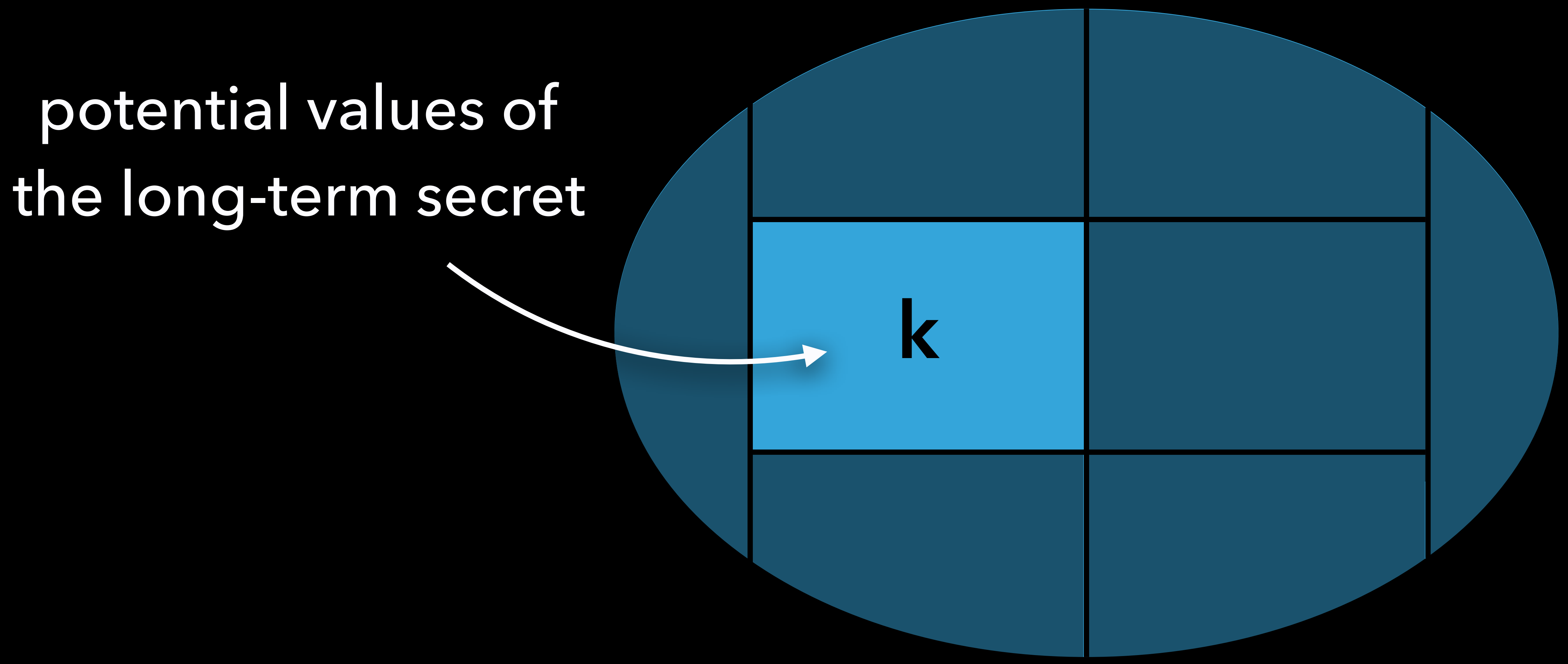
Aggregation of information



Compute this equivalence relation over the set of secrets

static approach
(security bounds)

Aggregation of information



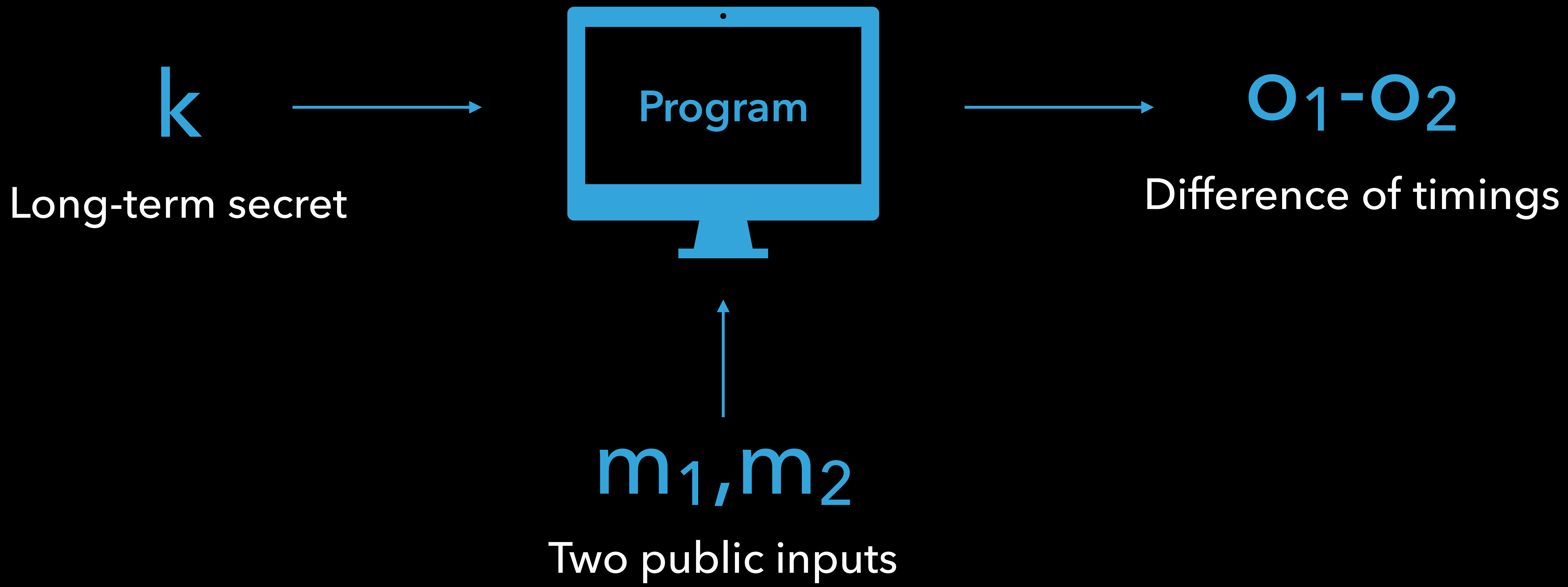
Compute this equivalence relation over the set of secrets

static approach
(security bounds)

Given an oracle to **t(k, .)**, retrieve the class enclosing k

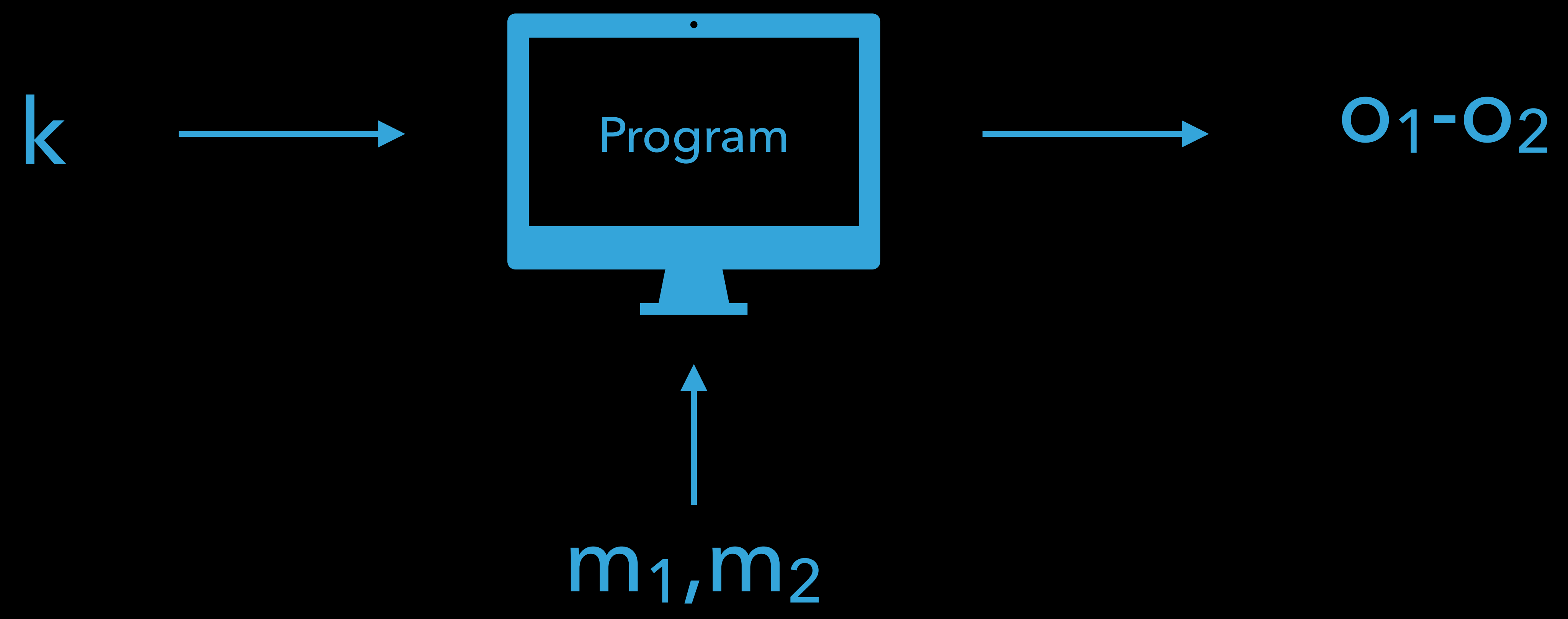
dynamic approach
(attacks)

A *more practical* model for timing leakage



k
Long-term secret
Program
$o_1 - o_2$
Difference of timings
m_1, m_2
Two public inputs

Differential measurements



Less powerful attacker, but…

Closer to the models used in actual attack research

Compositionality

Compositionality for differential measurements

Compositional attacks

```
for i = 0 to n − 1 do
    if k[i] ≠ m[i] then g()
done
```

recovering k?

with oracle to execution time ***m*** $\mapsto$ ***t(k,m)***

Compositional attacks

```
for i = 0 to n – 1 do
  | if k[i] ≠ m[i] then g()
done
```

recovering k?

with oracle to execution time $\boldsymbol{m} \mapsto \boldsymbol{t(k,m)}$

$t(k,0) - t(k,2^i)$

Compositional attacks

```
for i = 0 to n – 1 do
    if k[i] ≠ m[i] then g()
done
```

recovering k?

with oracle to execution time $\boldsymbol{m \mapsto t(k,m)}$

if $t(k,0) < t(k,2^i)$
then $K := K \cap \{\, k \mid k[i] = 1 \,\}$
else $K := K \cap \{\, k \mid k[i] = 0 \,\}$

Exploiting the i^{th} iteration

Sequential composition

```
x = m
for i = 0 to n – 1 do
    x = f_i(k,x)
    if Test_i(k,x) = 1 then g()
done
```

Sequential composition

```
x = m
for i = 0 to n – 1 do
    x = f_i(k,x)
    if Test_i(k,x) = 1 then g()
done
```

Goal: writing this code under the form

$$p = p_0;\ p_2\ ;\ \dots\ ;\ p_{n-1}$$

Sequential composition

```
x = m
for i = 0 to n – 1 do
    x = f_i(k,x)
    if Test_i(k,x) = 1 then g()
done
```

Goal: writing this code under the form

$$p = p_0;\ p_2\ ;\ \dots\ ;\ p_{n-1}$$

p_i computes $f_i : K \times M \rightarrow M$ with execution time $Test_i : K \times M \rightarrow \{0,1\}$

Sequential composition

$$p_{comp} = p_1 \,;\, p_2$$

Sequential composition

$$p_{comp} = p_1 ; p_2$$

$\boldsymbol{p_\ell}$ computes $\boldsymbol{f_\ell : K \times M \to M}$ with execution time $\boldsymbol{t_\ell : K \times M \to O}$

Sequential composition

$$p_{comp} = p_1 \,;\, p_2$$

$\boldsymbol{p_\ell}$ computes $\boldsymbol{f_\ell : K \times M \to M}$ with execution time $\boldsymbol{t_\ell : K \times M \to O}$

$$f_{comp} = f_2 \circ f_1$$

States are composed

Sequential composition

$p_{comp} = p_1 \,;\, p_2$

p_ℓ computes $f_\ell : K \times M \rightarrow M$ with execution time $t_\ell : K \times M \rightarrow O$

composition of public values,
i.e. $(f \circ g)(k,m) = f(k, g(k,m))$

$f_{comp} = f_2 \circ f_1$

States are composed

Sequential composition

$p_{comp} = p_1 \,;\, p_2$

p_ℓ computes $f_\ell : K \times M \to M$ with execution time $t_\ell : K \times M \to O$

composition of public values, i.e. $(f \circ g)(k,m) = f(k, g(k,m))$

$f_{comp} = f_2 \circ f_1$

States are composed

$t_{comp} = t_1 + (t_2 \circ f_1)$

Timings are summed

Key hypothesis: independence

Key hypothesis: independence

Hypotheses

- t,t' timing functions

Theorem

$\mathrm{Leak}(t+t') = \mathrm{Leak}(t) \cap \mathrm{Leak}(t')$

Key hypothesis: independence

Hypotheses

- t,t′ timing functions

Theorem

Leak(t+t′) = Leak(t) ∩ Leak(t′)

Leak(t) = the equivalence relation on secrets characterising timing leakage

Key hypothesis: independence

Hypotheses

- t,t′ timing functions
- X distribution of public inputs
- for all secrets k,k′, the distributions t(k, X) and t′(k′, X) are **independent**

Theorem

Leak(t+t′) = Leak(t) ∩ Leak(t′)

Leak(t) = the equivalence relation on secrets characterising timing leakage

Randomised compositional attack

Randomised compositional attack

Inputs

independent blocks $p_1 = (f_1,t_1),\ldots,p_n = (f_n,t_n)$

oracle to $t(k^*, .)$ execution time of $(p_1;\ldots;p_n)$

for some k^*

Randomised compositional attack

Inputs

independent blocks $p_1 = (f_1,t_1),\ldots,p_n = (f_n,t_n)$

oracle to $t(k^*, .)$ execution time of $(p_1;\ldots;p_n)$

*for some k**

Output

equivalence class of k^* in **Leak(t)**

Randomised compositional attack

Inputs

independent blocks $p_1 = (f_1,t_1),\ldots,p_n = (f_n,t_n)$

oracle to $t(k^*, .)$ execution time of $(p_1;\ldots;p_n)$

*for some k**

Output

equivalence class of k* in **Leak(t)**

Algorithm

K := set of all secrets

M := sample of r random messages

for i=1 to n do

    $K := K \cap \text{Attack}(\bar{t}_{i\,|\,K \times M})$

done

return K

Randomised compositional attack

Inputs

independent blocks $p_1 = (f_1,t_1),\ldots,p_n = (f_n,t_n)$

oracle to $t(k^*, .)$ execution time of $(p_1;\ldots;p_n)$

*for some k**

Output

equivalence class of k^* in **Leak(t)**

Algorithm

K := set of all secrets

M := sample of r random messages

for i=1 to n do

  $K := K \cap \text{Attack}(\bar{t}_{i\,|\,K \times M})$

done

return K

timing attack on
$\bar{\mathbf{t}}_{\mathbf{i}} = \mathbf{t}_{\mathbf{i}} \circ \mathbf{f}_{\mathbf{i-1}} \circ \ldots \circ \mathbf{f}_{\mathbf{1}}$
with oracle to $\mathbf{t(k^*, .)}$

Applications

Cost analysis

for simple bit-serial operations, n bits	
Bruteforce	$O(2^n)$ measurements
Random. attack	$O(n \log(n/\varepsilon))$ random measurements (to guarantee proba of success $1 - \varepsilon$)

Cost analysis

for simple bit-serial operations, n bits	
Bruteforce	$O(2^n)$ measurements
Random. attack	$O(n \log(n/\varepsilon))$ random measurements (to guarantee proba of success $1 - \varepsilon$)

complexity gain by exploiting the program structure

Explaining documented attacks

as instances of the randomised attack VS independent blocks

Explaining documented attacks

as instances of the randomised attack VS independent blocks

1998 on RSA (Dhem *et al.*)

Targets: implem. of modular exponentiation with Montgomery multiplications

Exploits: timing variations of squaring operations

Extracts: all bits of the secret exponent but one

Explaining documented attacks

as instances of the randomised attack VS independent blocks

1998 on RSA (Dhem *et al.*)

Targets: implem. of modular exponentiation with Montgomery multiplications

Exploits: timing variations of squaring operations

Extracts: all bits of the secret exponent but one

Decomposition:

1 block = 1 multiplication

Explaining documented attacks

as instances of the randomised attack VS independent blocks

2007 on AES (Acıiçmez *et al.*)

Targets: implem. of AES with precomputed tables

Exploits: timing variations due to cache

Extracts: all bits of the encryption key

Decomposition:

1 block = 1 table lookup

Conclusion

Conclusion

A formal model for reasoning about timing attacks

Conclusion

A formal model for reasoning about timing attacks

- Compositionality results

Conclusion

A formal model for reasoning about timing attacks

- Compositionality results
- Generic description of attacks / cost analysis

Conclusion

A formal model for reasoning about timing attacks

- Compositionality results
- Generic description of attacks / cost analysis
- Captures several documented attacks

Conclusion

A formal model for reasoning about timing attacks

- Compositionality results
- Generic description of attacks / cost analysis
- Captures several documented attacks
- Future: use as a basis for automating attack synthesis