Verifiable Capacity-bound Functions: A New Primitive from Kolmogorov Complexity

Giuseppe Ateniese

George Mason University

Long Chen

Institute of Software
Chinese Academy of Science

Danilo Francati

Aarhus University

Dimitrios Papadopoulos

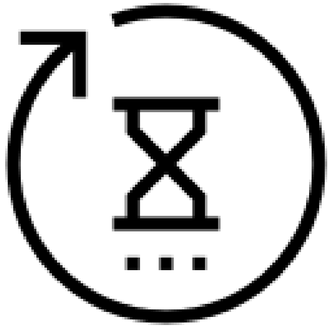
Hong Kong University of Science
and Technology

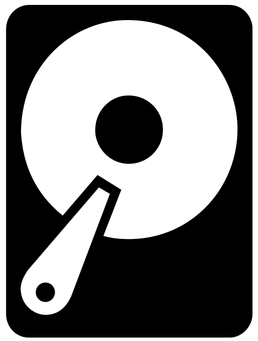
Qiang Tang

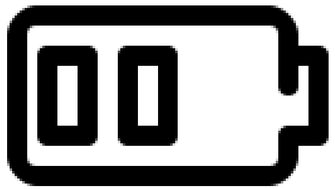
The University of Sydney

$$F(x) = y$$

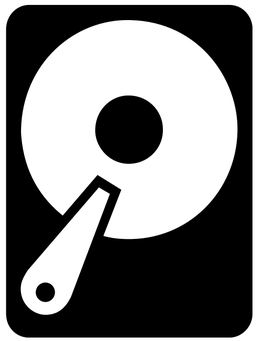
$$F(x) = y$$

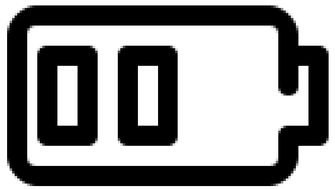






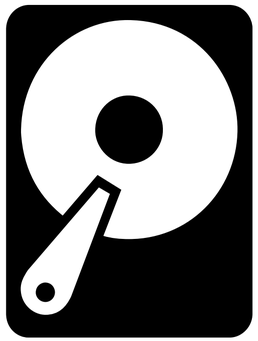
$$F(x) = y$$





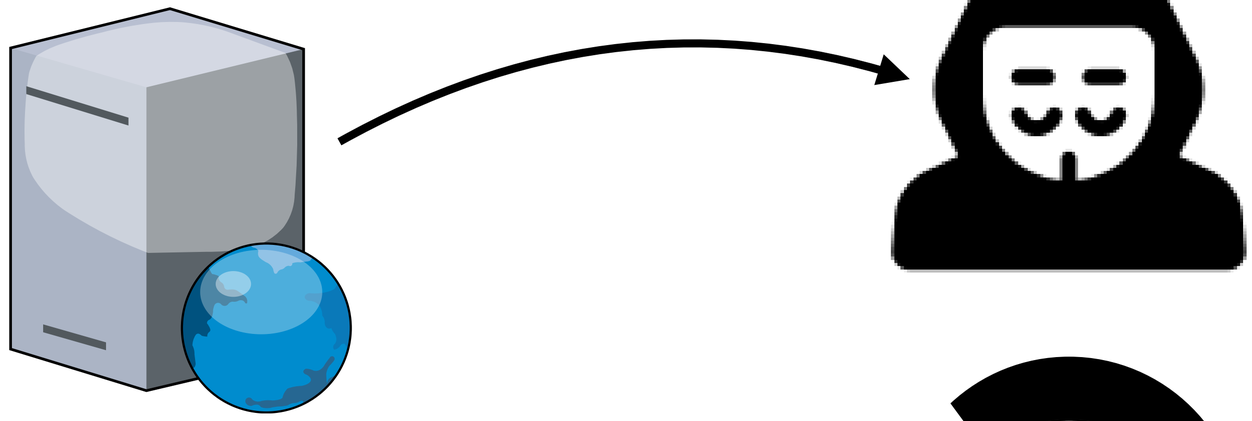
MINIMIZE

$$F(x) = y$$



~~MINIMIZE~~

MAXIMIZE



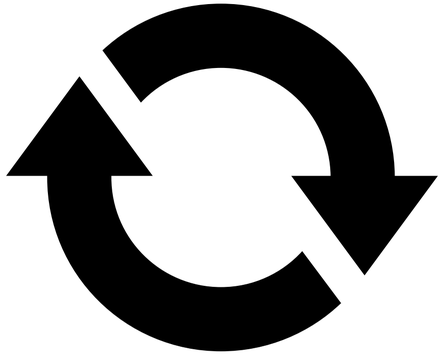
Hack

$F(pwd_1) = y_1$

$F(pwd_2) = y_2$

$\vdots$

$F(pwd_n) = y_n$



Offline Attack
Bruteforce

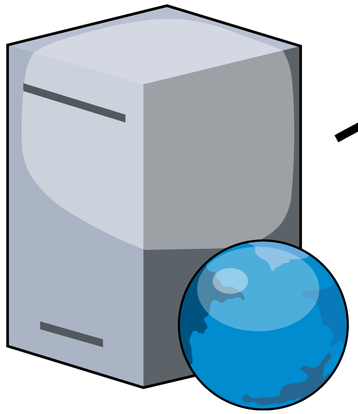
Extract

pwd_1

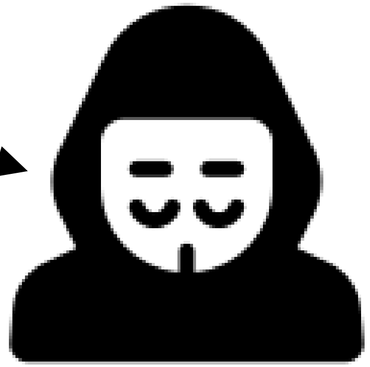
pwd_2

$\vdots$

pwd_n



Hack

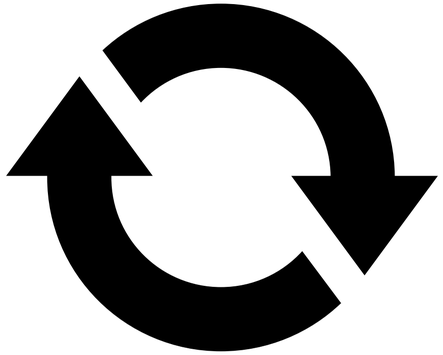


$$F(x) = y$$

(Time, Space, Energy)

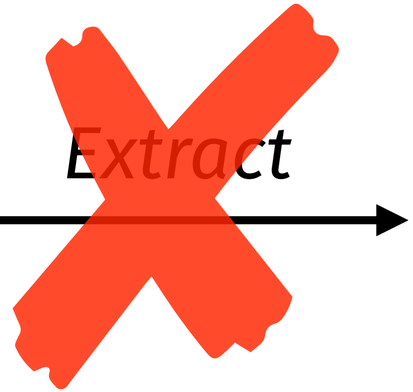
demanding Function

$$F(pwd_1) = y_1$$
$$F(pwd_2) = y_2$$
$$\vdots$$
$$F(pwd_n) = y_n$$



Offline Attack
Bruteforce

Extract



$$pwd_1$$
$$pwd_2$$
$$\vdots$$
$$pwd_n$$

Resource-demanding Functions

$$F(x) = y$$

Time

Space

Energy

Resource-demanding Functions

$$F(x) = y$$

Time — ***Time**-demanding Functions*

Time-Lock Encryption

Space

Energy

Resource-demanding Functions

$$F(x) = y$$

Time — ***Time**-demanding Functions*

Time-Lock Encryption

Space — ***Space**-demanding Functions*

Memory-Hard Functions
Code-Hard Functions

Energy

Resource-demanding Functions

$$F(x) = y$$

Time — ***Time***-*demanding Functions*

Time-Lock Encryption

Space — ***Space***-*demanding Functions*

Memory-Hard Functions

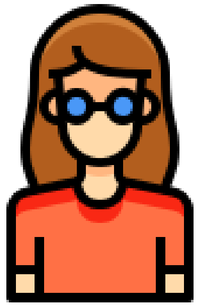
Code-Hard Functions

Energy — ***Energy***-*demanding Functions*

Bandwidth-Hard Functions

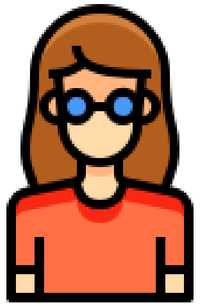
Evaluator

Verifiers



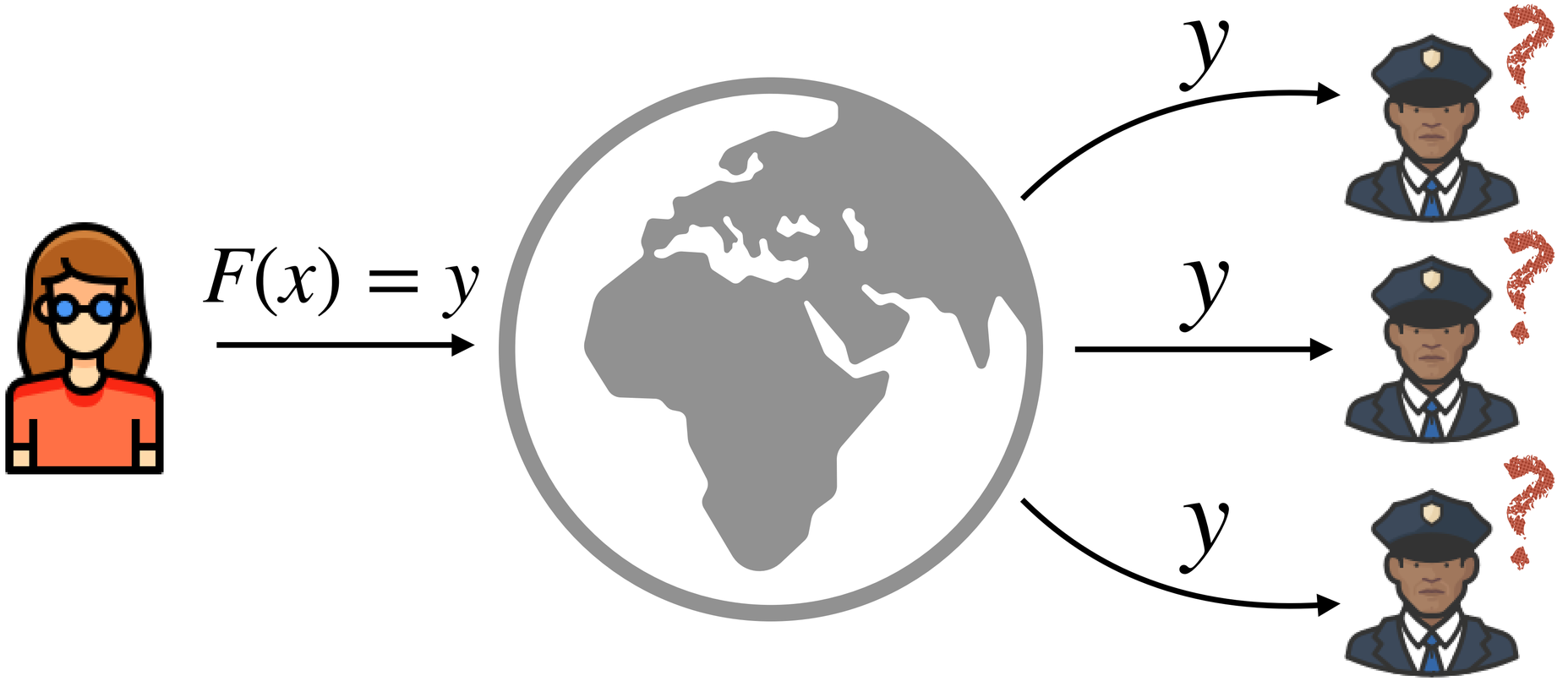
Evaluator

Verifiers



$F(x) = y$

Evaluator *Verifiers*

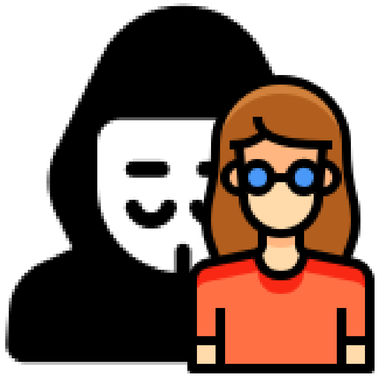


Evaluator

Malicious

Verifiers

Honest



$$F(x_1) = y_1$$
$$\vdots$$
$$F(x_n) = y_n$$

y_n

y_n

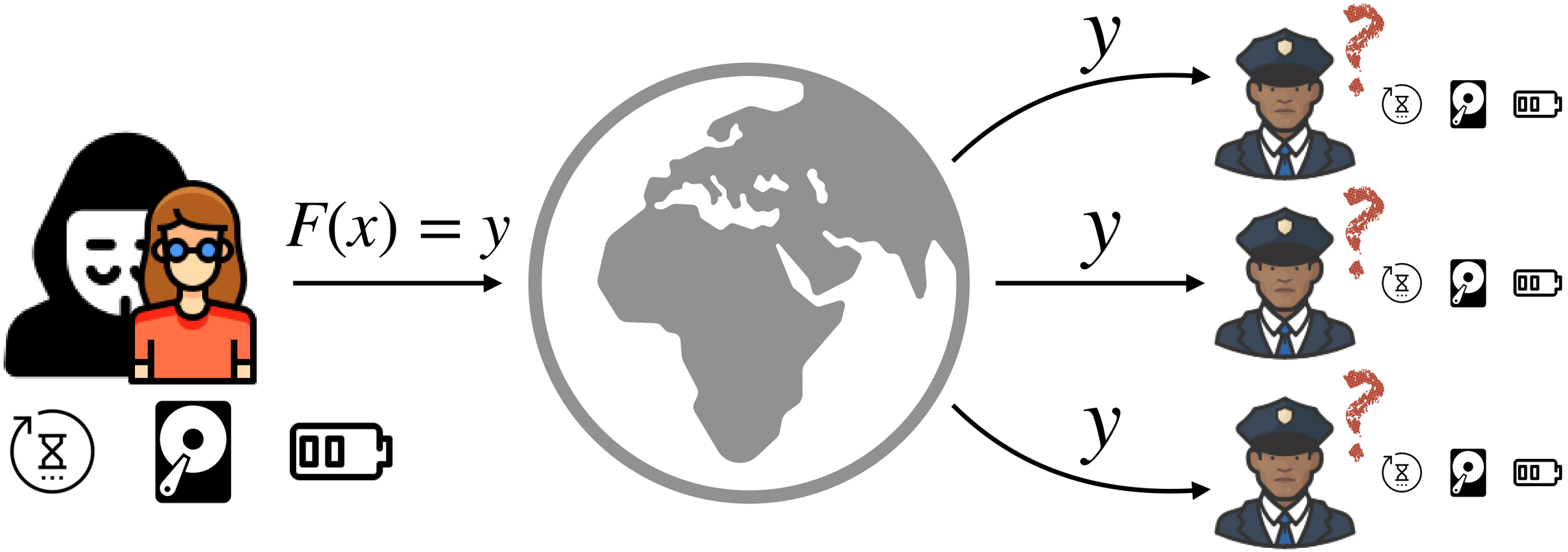
y_n

Evaluator

Malicious

Verifiers

Honest

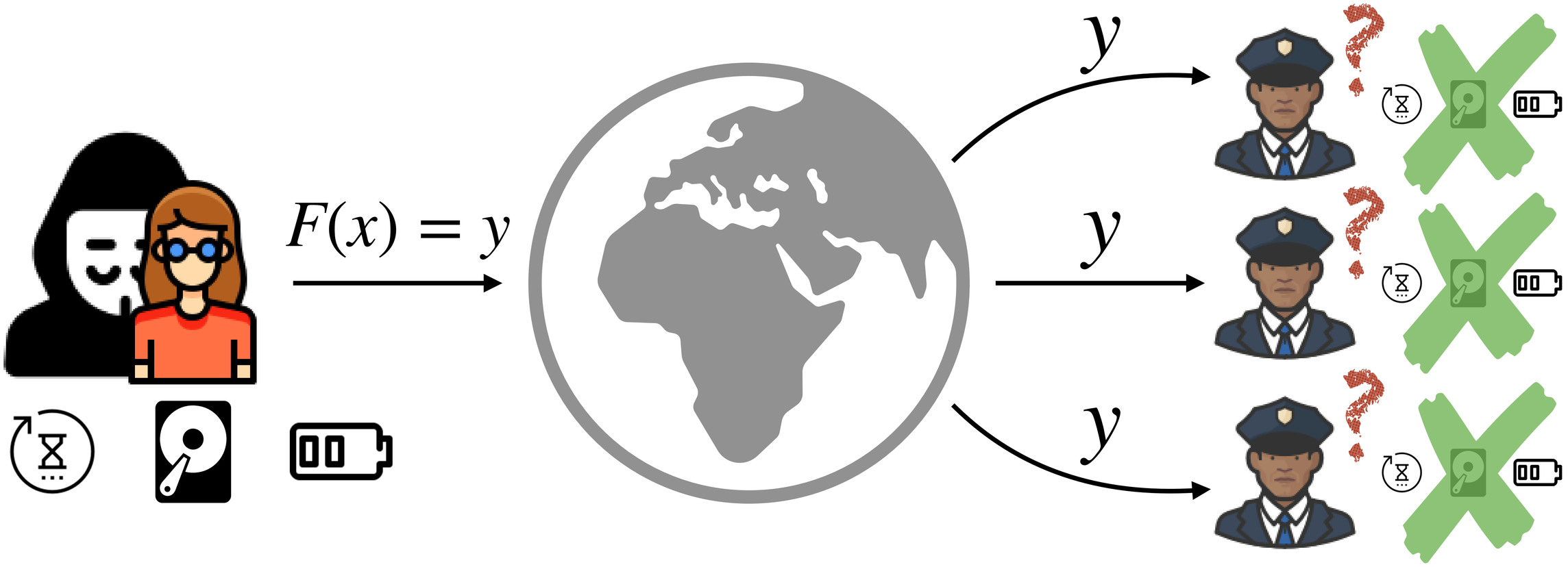


Evaluator

Malicious

Verifiers

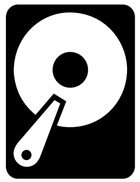
Honest



Publicly Verifiable

Verifiable
Resource-demanding Functions

Verifiable
Time*-demanding Functions*

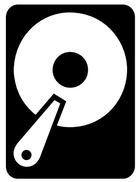


Verifiable
Space*-demanding Functions*

Verifiable
Energy*-demanding Functions*

Verifiable
Resource-demanding Functions

Verifiable
***Time**-demanding Functions*



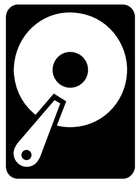
Verifiable
***Space**-demanding Functions*

Verifiable
***Energy**-demanding Functions*

Verifiable
Delay Functions
(VDF)

Verifiable
Resource-demanding Functions

Verifiable
***Time**-demanding Functions*



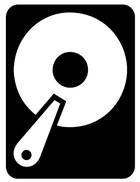
Verifiable
***Space**-demanding Functions*

Verifiable
***Energy**-demanding Functions*

Verifiable
Delay Functions
(VDF)

Verifiable
Resource-demanding Functions

Verifiable
***Time**-demanding Functions*



Verifiable
***Space**-demanding Functions*

Verifiable
***Energy**-demanding Functions*

Verifiable
Delay Functions
(VDF)

Verifiable
Capacity-bound Functions
(VCBF)

Verifiable Capacity-bound Functions

Energy-demanding evaluation

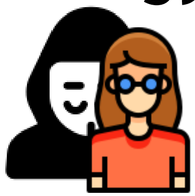
$$F(x) = y$$

Energy-“free” verification

$$F(x) \overset{?}{=} y$$

Verifiable Capacity-bound Functions

Energy-demanding evaluation

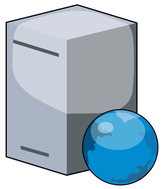


$$F(x) = y$$

Energy-"free" verification

$$F(x) \overset{?}{=} y$$

Laptops



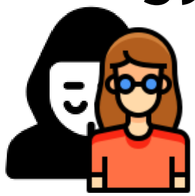
Servers

ASICs

[RD17] Ren, Ling, and Srinivas Devadas. "Bandwidth hard functions for ASIC resistance." (TCC17)

Verifiable Capacity-bound Functions

Energy-demanding evaluation

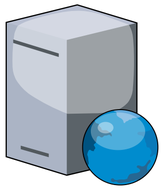


$$F(x) = y$$

Energy-"free" verification

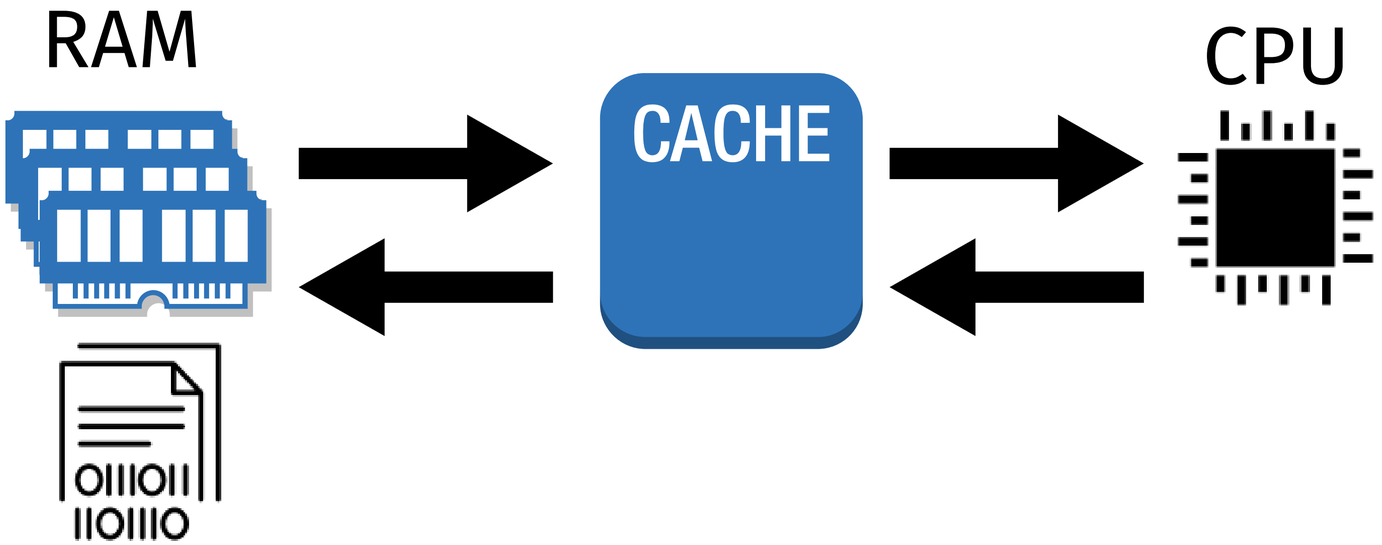
$$F(x) \overset{?}{=} y$$

Laptops



Servers

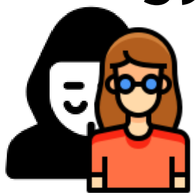
ASICs



[RD17] Ren, Ling, and Srinivas Devadas. "Bandwidth hard functions for ASIC resistance." (TCC17)

Verifiable Capacity-bound Functions

Energy-demanding evaluation

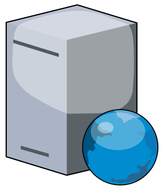


$$F(x) = y$$

Energy-"free" verification

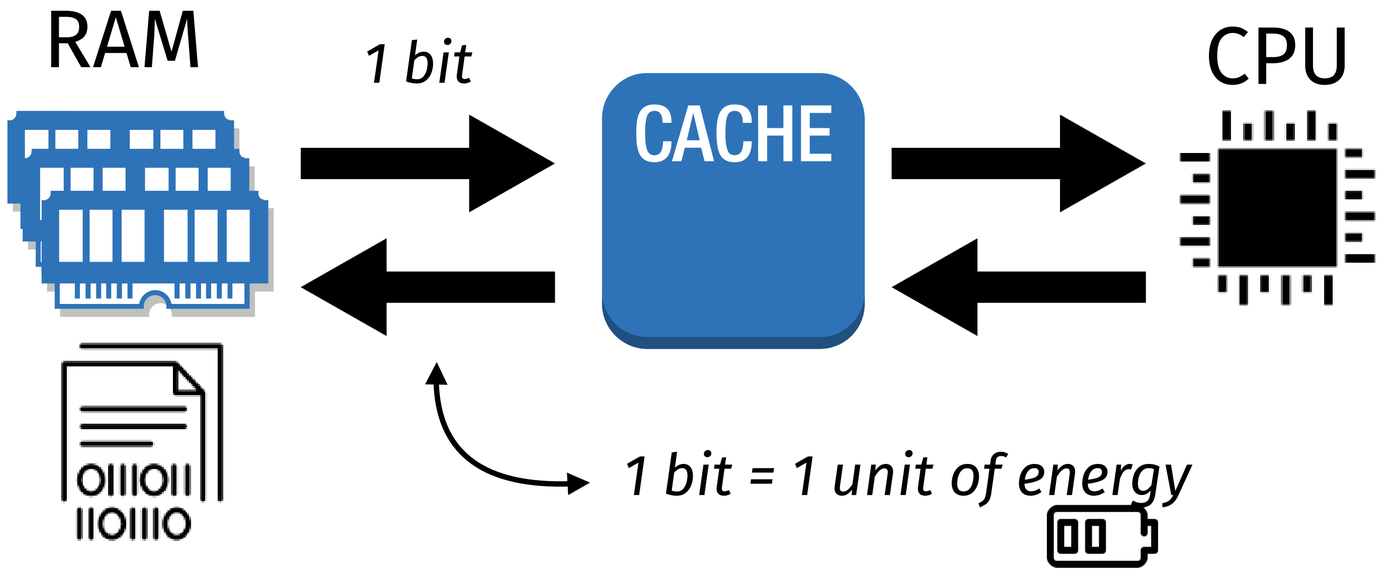
$$F(x) \stackrel{?}{=} y$$

Laptops



Servers

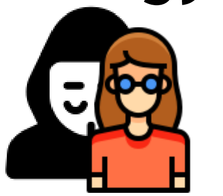
ASICs



[RD17] Ren, Ling, and Srinivas Devadas. "Bandwidth hard functions for ASIC resistance." (TCC17)

Verifiable Capacity-bound Functions

Energy-demanding evaluation



$$F(x) = y$$

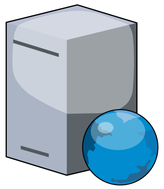
MAX

MIN

Energy-"free" verification

$$F(x) \overset{?}{=} y$$

Laptops



Servers

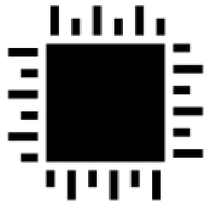
ASICs

RAM

1 bit

CACHE

CPU



1 bit = 1 unit of energy

[RD17] Ren, Ling, and Srinivas Devadas. "Bandwidth hard functions for ASIC resistance." (TCC17)

Verifiable Capacity-bound Functions

Energy-demanding evaluation

$$F(x) = y$$

MAX

MIN

Energy-"free" verification

$$F(x) \overset{?}{=} y$$

Verifiable Capacity-bound Functions

Energy-demanding evaluation

$$F(x) = y$$

MAX

MIN

Energy-"free" verification

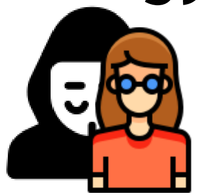
$$F(x) \overset{?}{=} y$$

Minimum Capacity

$$x \longrightarrow \longrightarrow y = F(x)$$

Verifiable Capacity-bound Functions

Energy-demanding evaluation



$$F(x) = y$$

MAX

MIN

Energy-“free” verification

$$F(x) \overset{?}{=} y$$

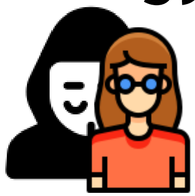
Minimum Capacity

$x \longrightarrow$  $\longrightarrow y = F(x)$

reads m *distinct bits*

Verifiable Capacity-bound Functions

Energy-demanding evaluation



$$F(x) = y$$

MAX

MIN

Energy-"free" verification

$$F(x) \stackrel{?}{=} y$$

Minimum Capacity

$$x \longrightarrow \longrightarrow y = F(x)$$

reads m *distinct bits*

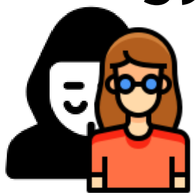
CACHE *of size B*

effective bits read

$$m - B$$

Verifiable Capacity-bound Functions

Energy-demanding evaluation



$$F(x) = y$$

MAX

MIN

Energy-"free" verification

$$F(x) \overset{?}{=} y$$

Minimum Capacity

$$x \longrightarrow \longrightarrow y = F(x)$$

reads m *distinct bits*

CACHE *of size B*

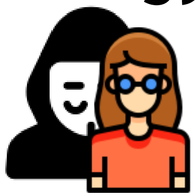
effective bits read

$$m - B$$

Units of $= m - B$

Verifiable Capacity-bound Functions

Energy-demanding evaluation



$$F(x) = y$$

MAX

MIN

Energy-"free" verification

$F(x) = y$?

Minimum Capacity

$x \longrightarrow$  $\longrightarrow y = F(x)$

reads m *distinct bits*

Verification

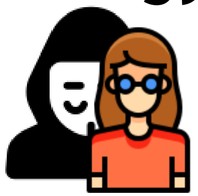
Sound:

$F(x) \neq y \longrightarrow$  $\longrightarrow$ 

$F(x) = y \longrightarrow$  $\longrightarrow$ 

Verifiable Capacity-bound Functions

Energy-demanding evaluation



$$F(x) = y$$

MAX

MIN

Energy-"free" verification

$F(x) = y$?

Minimum Capacity

$x \longrightarrow$  $\longrightarrow y = F(x)$

reads m *distinct bits*

Verification

Sound:

$F(x) \neq y \longrightarrow$  $\longrightarrow$ 

$F(x) = y \longrightarrow$  $\longrightarrow$ 

Efficient:  *reads* $t \ll m$ *distinct bits*

Outline

Kolmogorov Complexity

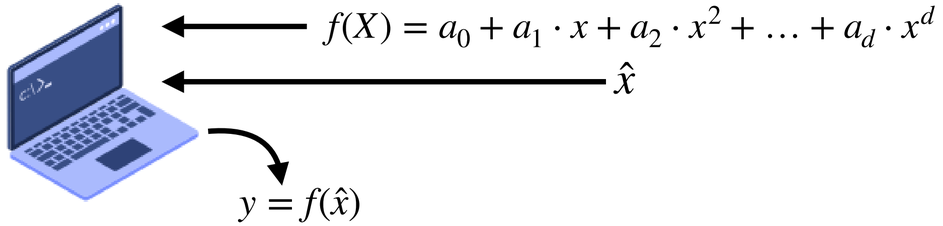
Outline

Kolmogorov Complexity

1

Space Requirements

Polynomial Evaluation



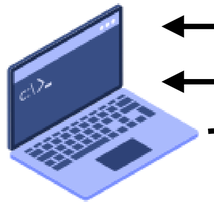
Outline

Space Requirements

Kolmogorov Complexity

1

Polynomial Evaluation



$f(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d$

$\hat{x}$

$y = f(\hat{x})$

2

Verifiable Capacity-bound Functions

$F(\cdot)$

Kolmogorov Complexity

Space Requirements

Polynomial Evaluation

$$y = f(x)???$$

Kolmogorov Complexity

Kolmogorov Complexity

Description of an object

Kolmogorov Complexity

Complex?

Description of an object 🧸

 $= x \in \{0,1\}^*$

Kolmogorov Complexity

Complex?

Description of an object

 $= x \in \{0,1\}^*$

Description

$\langle \mathsf{T}, \alpha \rangle \quad \text{if} \quad \mathsf{T}(\alpha) = x$

Kolmogorov Complexity

Complex?

Description of an object

 $= x \in \{0,1\}^*$

Description

$\langle \mathsf{T}, \alpha \rangle \quad \text{if} \quad \mathsf{T}(\alpha) = x$

Decompression Algorithm

Compression of x

Kolmogorov Complexity

Description of an object 

 $= x \in \{0,1\}^*$

Description

$\langle \mathsf{T}, \alpha \rangle \quad \text{if} \quad \mathsf{T}(\alpha) = x$

Decompression Algorithm ← T , α → *Compression of* x

Kolmogorov Complexity

$$C_{\mathsf{T}}(x) = \min_{\alpha \in \{0,1\}^*} \{ |\alpha| : \mathsf{T}(\alpha) = x \}$$

Kolmogorov Complexity *cont.*

$$C_{\mathsf{T}}(x) \xrightarrow{\text{Is it possible?}} C(x)$$

Kolmogorov Complexity *cont.*

$$C_{\mathsf{T}}(x) \xrightarrow{\text{Is it possible?}} C(x)$$

Universal Turing machine

$$\mathsf{U}(i, \alpha) = \mathsf{T}_i(\alpha)$$

Kolmogorov Complexity *cont.*

$$C_{\mathsf{T}}(x) \xrightarrow{\text{Is it possible?}} C(x)$$

Universal Turing machine

$$\mathsf{U}(i, \alpha) = \mathsf{T}_i(\alpha)$$

Kolmogorov Complexity

$$C(x) = C_{\mathsf{U}}(x) = \min_{\alpha \in \{0,1\}^*} \{\, |\alpha| : \mathsf{U}(\alpha) = x \}$$

Kolmogorov Complexity *cont.*

Kolmogorov Complexity

$$C(x) = C_{\mathsf{U}}(x) = \min_{\alpha \in \{0,1\}^*} \{ |\alpha| : \mathsf{U}(\alpha) = x \}$$

Description of an object 🧸

Description

$$\langle \mathsf{T}, \alpha \rangle \quad \text{if} \quad \mathsf{T}(\alpha) = x$$

Kolmogorov Complexity *cont.*

Kolmogorov Complexity

$$C(x) = C_{\mathsf{U}}(x) = \min_{\alpha \in \{0,1\}^*} \{ |\alpha| : \mathsf{U}(\alpha) = x \}$$

Description of an object

Description

[redacted]

Description

$$\alpha \quad \text{if} \quad \mathsf{U}(\alpha) = x$$

Incompressibility

c-Incompressibility

$$C(x) \geq |x| - c$$

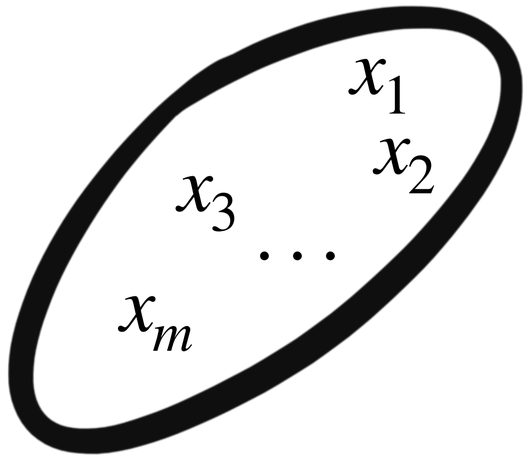
Incompressibility

c-Incompressibility

$$C(x) \geq |x| - c$$

Arbitrary Set

Cardinality
m



Number of c-incompressible strings

$$\geq m(1 - 2^{-c}) + 1$$

Polynomial Evaluation

$y = f(x)???$

Polynomial Evaluation

$y = f(x)???$

Degree-d polynomial $f(X) \in \mathbb{F}_p[X]$

$$f(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d$$

Polynomial Evaluation

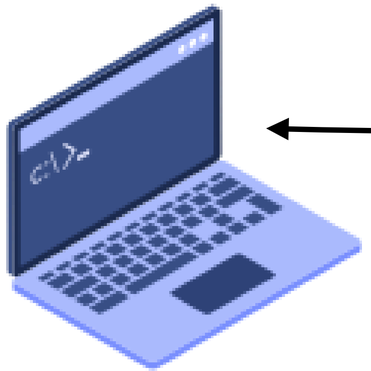
$y = f(x)???$

Degree-d polynomial $f(X) \in \mathbb{F}_p[X]$

$$f(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d$$

Multiple points

$x_1, x_2, x_3, \ldots, x_n$



Polynomial Evaluation

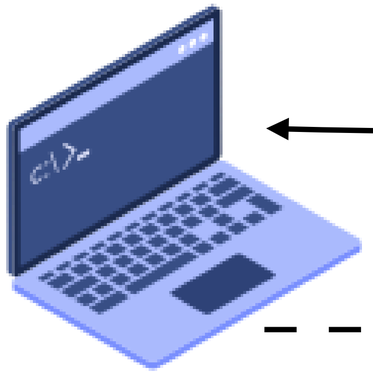
$y = f(x)???$

Degree-d polynomial $f(X) \in \mathbb{F}_p[X]$

$$f(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d$$

Multiple points

$x_1, x_2, x_3, \ldots, x_n$



Compute Evaluations

$f(x_1), f(x_2), f(x_3), \ldots, f(x_n)$

Polynomial Evaluation

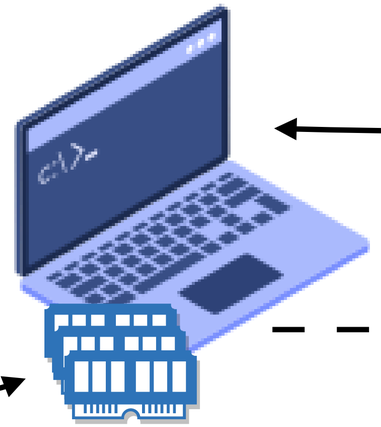
$y = f(x)???$

Degree-d polynomial $f(X) \in \mathbb{F}_p[X]$

$$f(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d$$

Multiple points

$$x_1, x_2, x_3, \ldots, x_n$$



Compute Evaluations

$$f(x_1), f(x_2), f(x_3), \ldots, f(x_n)$$

Space Requirements

$$\#bits_{read}$$

Kolmogorov Complexity

Complex?

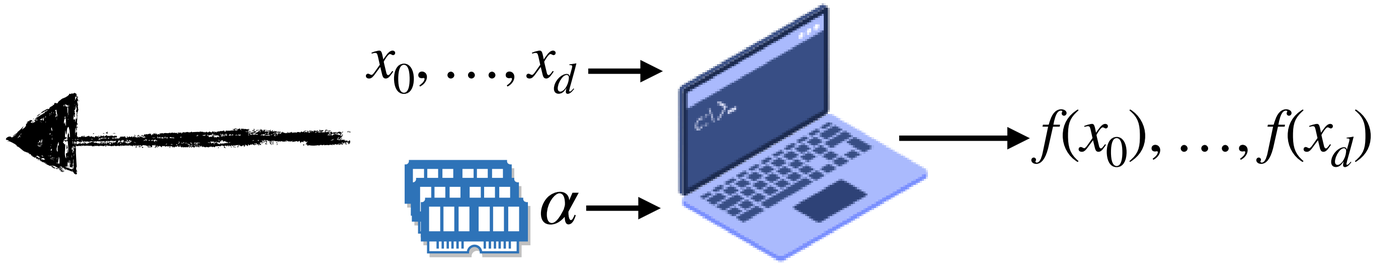
1

Polynomial Evaluation

$y = f(x)???$

Description of $a_0, \ldots, a_d$

$(\alpha, x_0, \ldots x_d)$



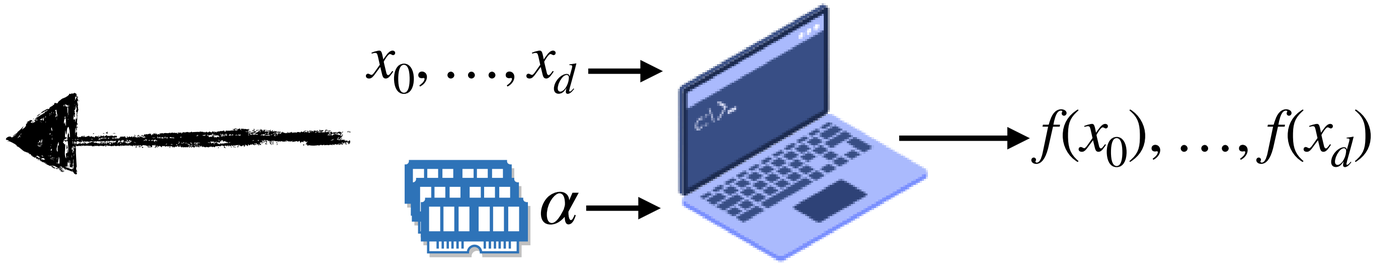
Kolmogorov Complexity ⟷ 1 ⟷ Polynomial Evaluation

Complex?

$y = f(x)???$

Description of $a_0, \ldots, a_d$

$(\alpha, x_0, \ldots x_d)$



$C(a_0, \ldots, a_d)$ ⟷ *Strong Connection* ⟷ $\#bits_{read} = |\alpha|$

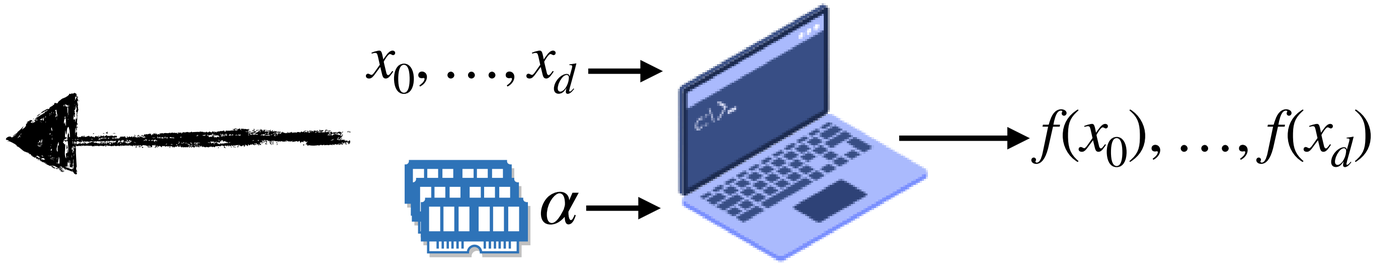
Kolmogorov Complexity ← 1 → Polynomial Evaluation

Complex?

$y = f(x)???$

Description of $a_0, \ldots, a_d$

$(\alpha, x_0, \ldots x_d)$



$C(a_0, \ldots, a_d)$ ← *Strong Connection* → $\#bits_{read} = |\alpha|$

Kolmogorov-bound for polynomial evaluation

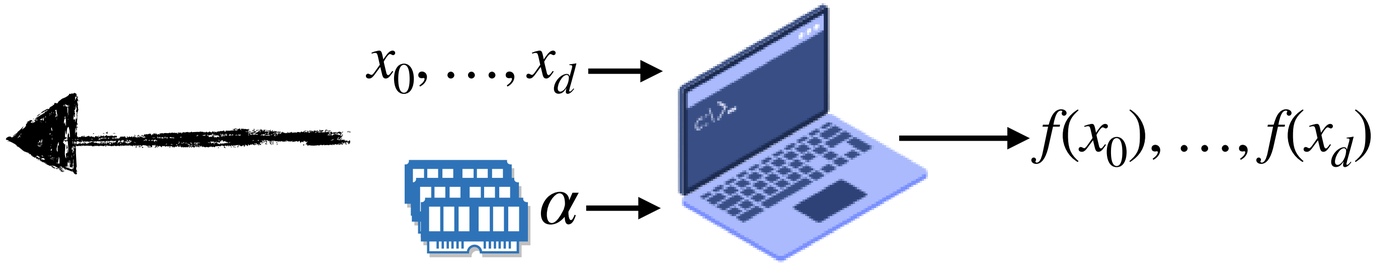
Kolmogorov Complexity ↔ 1 ↔ Polynomial Evaluation

Complex?

$y = f(x)???$

Description of $a_0, \ldots, a_d$

$(\alpha, x_0, \ldots x_d)$



$C(a_0, \ldots, a_d)$ ↔ *Strong Connection* ↔ $\#bits_{read} = |\alpha|$

Kolmogorov-bound for polynomial evaluation

If $a_0||\ldots||a_d$ *is* *c-incompressible*

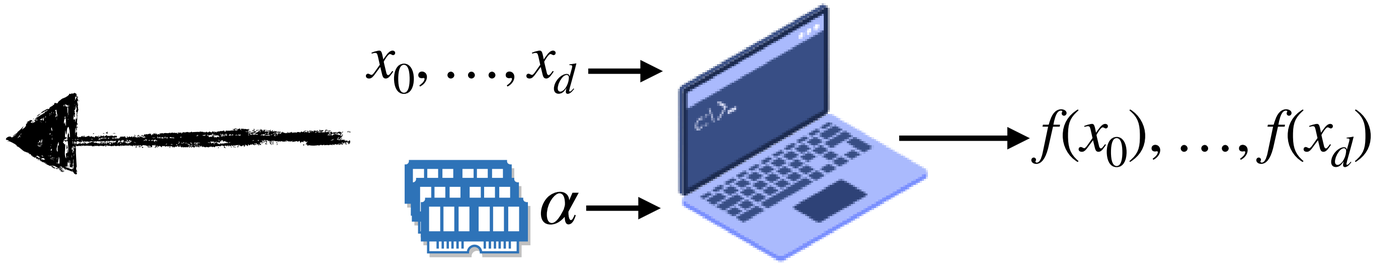
Kolmogorov Complexity ← 1 → Polynomial Evaluation

Complex?

$y = f(x)???$

Description of $a_0, \ldots, a_d$

$(\alpha, x_0, \ldots x_d)$



$$C(a_0, \ldots, a_d) \quad \text{Strong Connection} \quad \#bits_{read} = |\alpha|$$

Kolmogorov-bound for polynomial evaluation

If $a_0||\ldots||a_d$ *is* *c-incompressible* —*i.e.*→ $C(a_0||\ldots||a_d) \geq \lambda \cdot (d+1) - c$

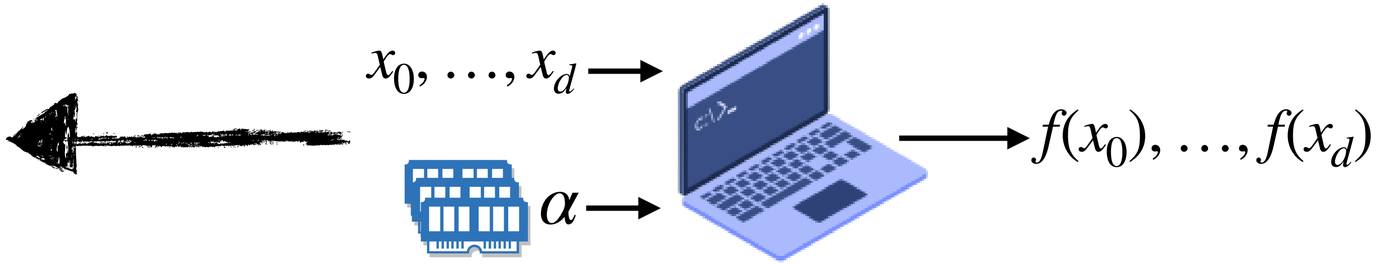
Kolmogorov Complexity ← 1 → Polynomial Evaluation

Complex?

$y = f(x)???$

Description of $a_0, \ldots, a_d$

$(\alpha, x_0, \ldots x_d)$



$$C(a_0, \ldots, a_d) \quad \text{Strong Connection} \quad \#bits_{read} = |\alpha|$$

Kolmogorov-bound for polynomial evaluation

If $a_0||\ldots||a_d$ *is* c*-incompressible* —*i.e.*→ $C(a_0||\ldots||a_d) \geq \lambda \cdot (d+1) - c$

Then $\forall (x_0, \ldots x_d) \in \mathbb{F}_p^{d+1}$ *we have* $\#bits_{read} = |\alpha| \gtrapprox \cdot (d+1)(\lambda - \ell) - c$

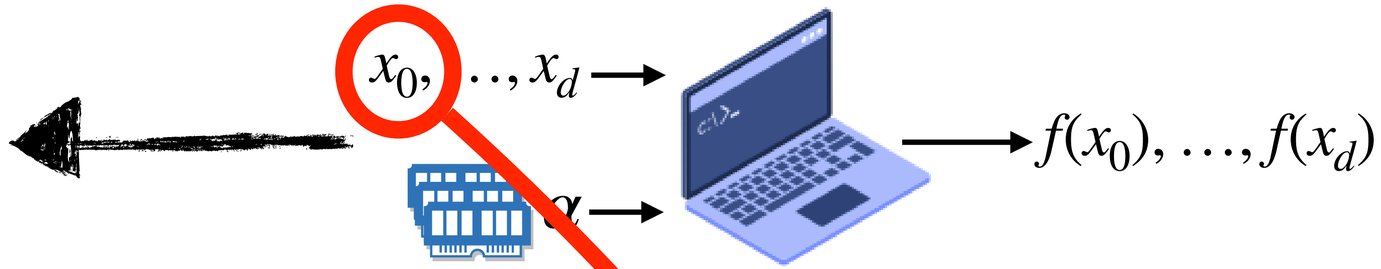
Kolmogorov Complexity ← 1 → Polynomial Evaluation

Complex?

$y = f(x)???$

Description of $a_0, \ldots, a_d$

$(\alpha, x_0, \ldots x_d)$



$C(a_0, \ldots, a_d)$ ← Strong Connection → $\#bits_{read} = |\alpha|$

Kolmogorov-bound for polynomial evaluation

If $a_0||\ldots||a_d$ *is* *c-incompressible* —*i.e.*→ $C(a_0||\ldots||a_d) \geq \lambda \cdot (d+1) - c$

Then $\forall (x_0, \ldots x_d) \in \mathbb{F}_p^{d+1}$ *we have* $\#bits_{read} = |\alpha| \gtrapprox \cdot (d+1)(\lambda - \ell) - c$

Space Requirements

Polynomial Evaluation

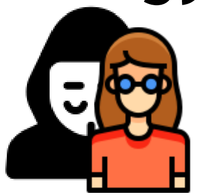
$y = f(x)???$

Verifiable Capacity-bound Functions

$F(\cdot)$

Verifiable Capacity-bound Functions

Energy-demanding evaluation



$$F(x) = y$$

MAX

MIN

Energy-"free" verification

$F(x) = y$?

Minimum Capacity

$x \longrightarrow$  $\longrightarrow y = F(x)$

reads m *distinct bits*

Verification

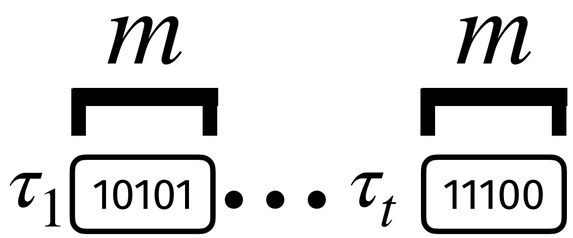
Sound:

$F(x) \neq y \longrightarrow$  $\longrightarrow$ 

$F(x) = y \longrightarrow$  $\longrightarrow$ 

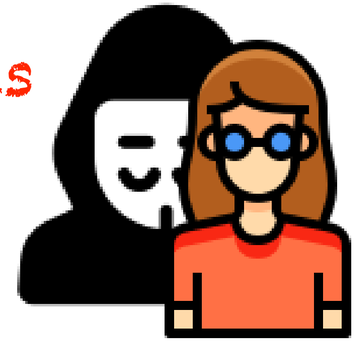
Efficient:  *reads* $t \ll m$ *distinct bits*

(ϵ, m, t)–Minimum Capacity

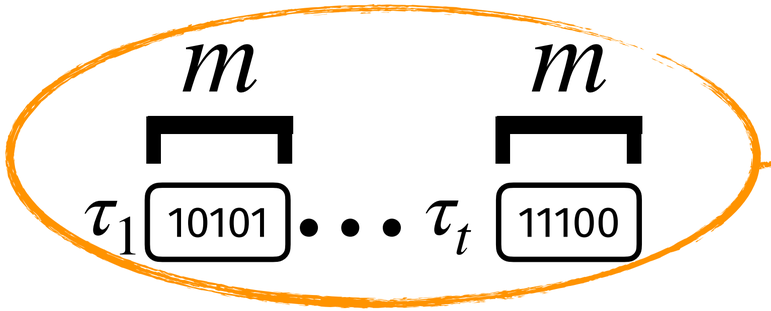


Malicious

Evaluator



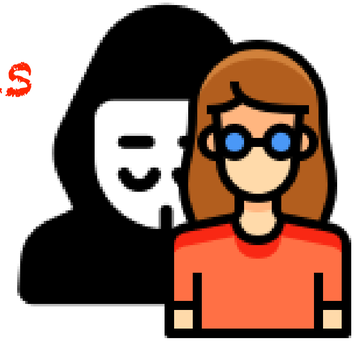
(ϵ, m, t)–Minimum Capacity



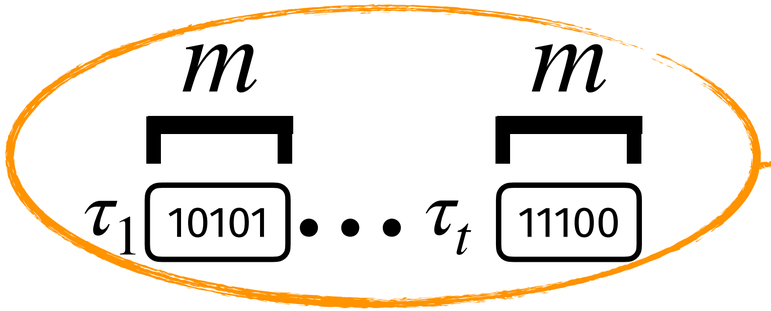
Preprocess $F(\cdot)$

Malicious

Evaluator



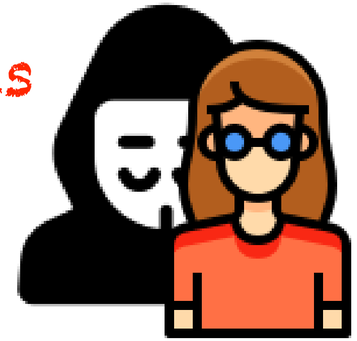
(ϵ, m, t)–Minimum Capacity



Preprocess $F(\cdot)$

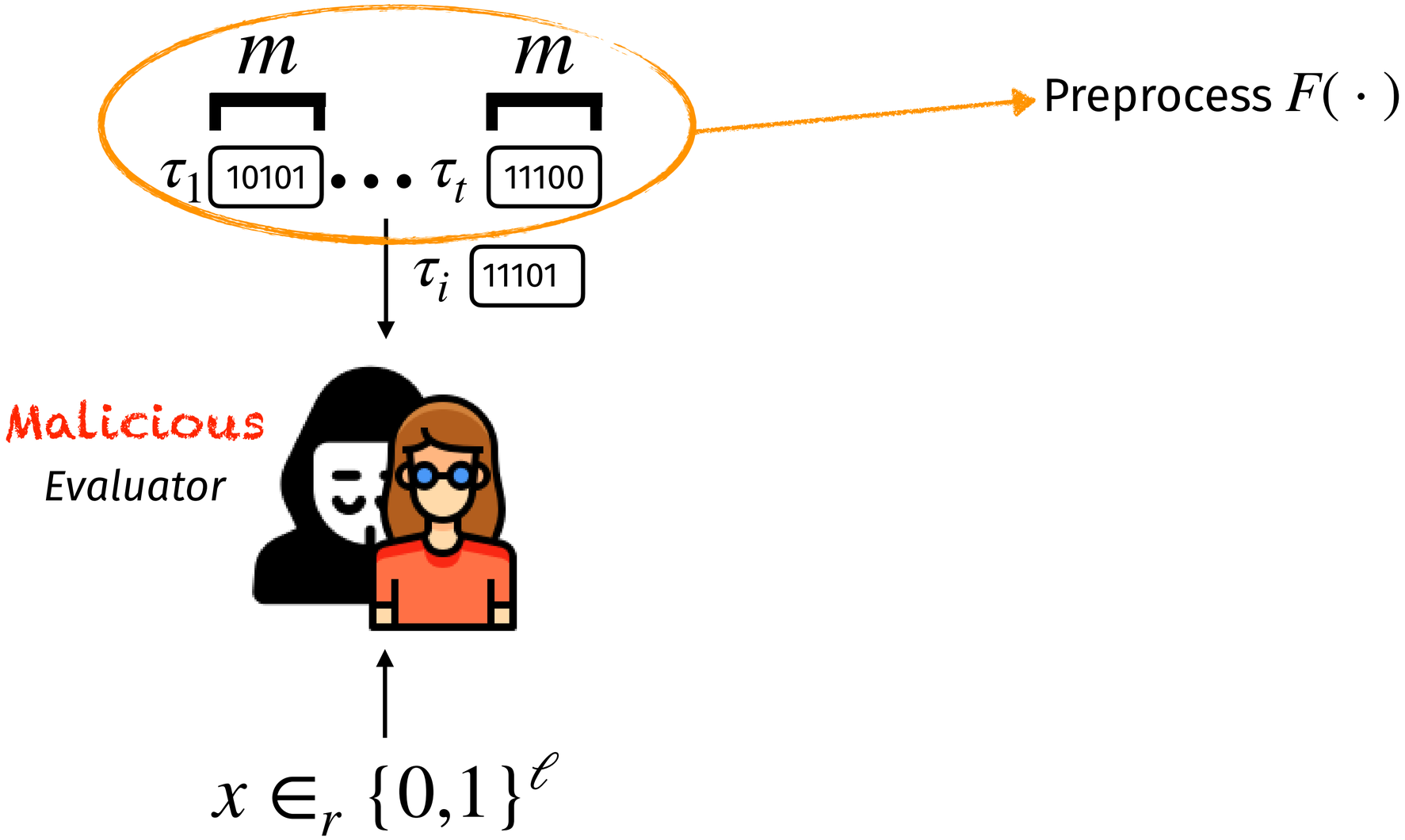
Malicious

Evaluator

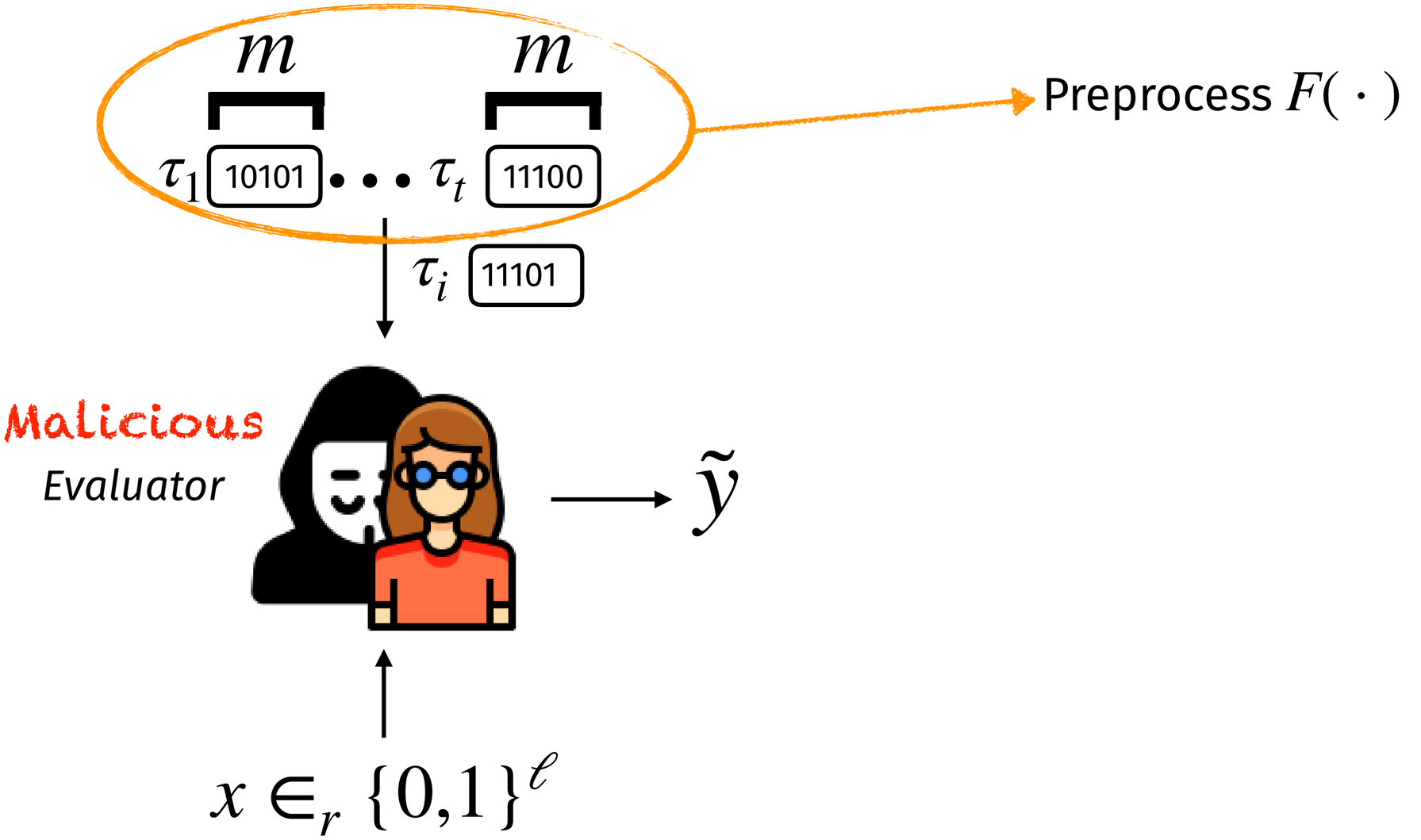


$x \in_r \{0,1\}^{\ell}$

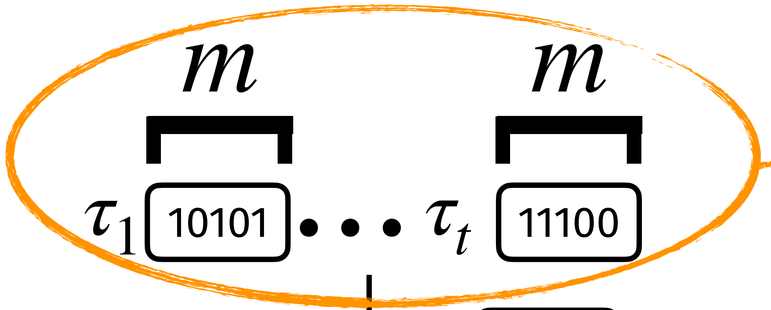
(ϵ, m, t)–Minimum Capacity



(ϵ, m, t)–Minimum Capacity

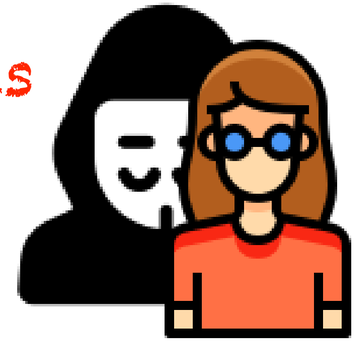


(ϵ, m, t)–Minimum Capacity



Preprocess $F(\cdot)$

Secure if



Malicious

Evaluator

$\tilde{y}$

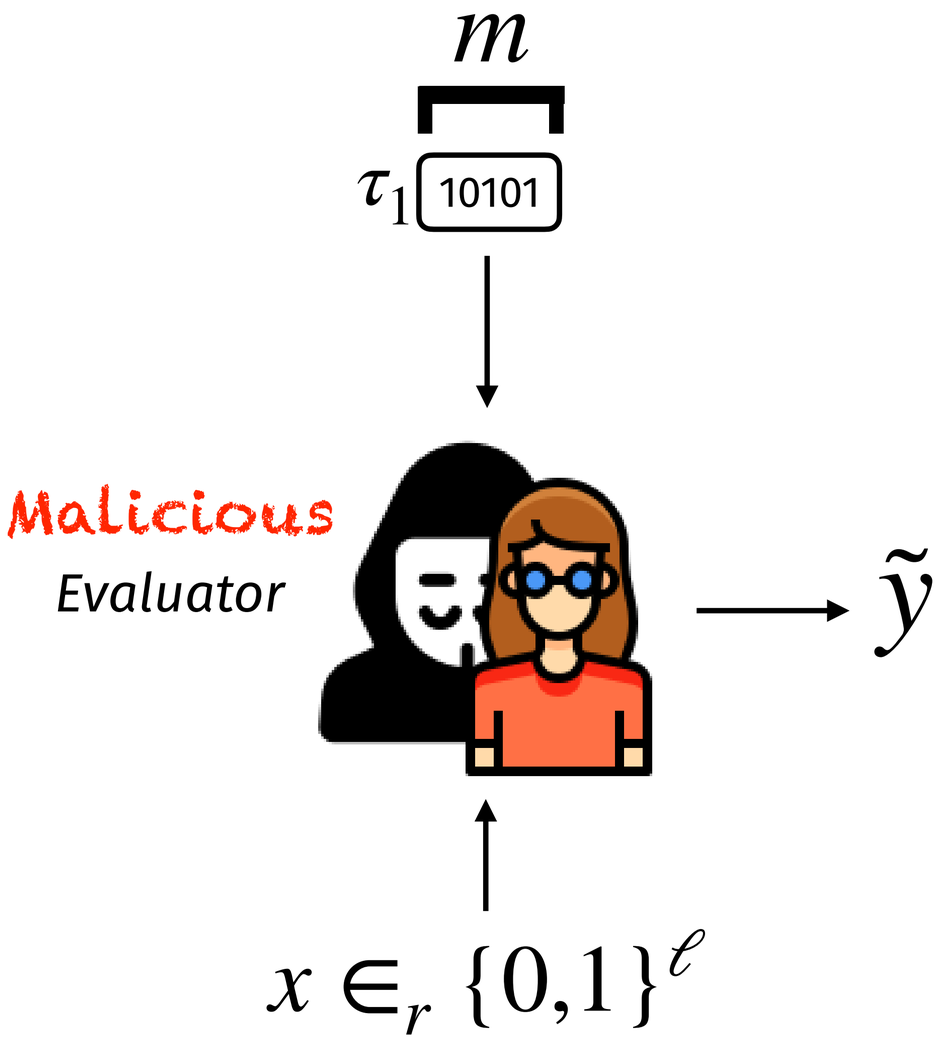
$$\Pr[\ \tilde{y} = F(x)\ ] < \epsilon$$

$x \in_r \{0,1\}^{\ell}$

$(\epsilon, m, 1)$–Minimum Capacity

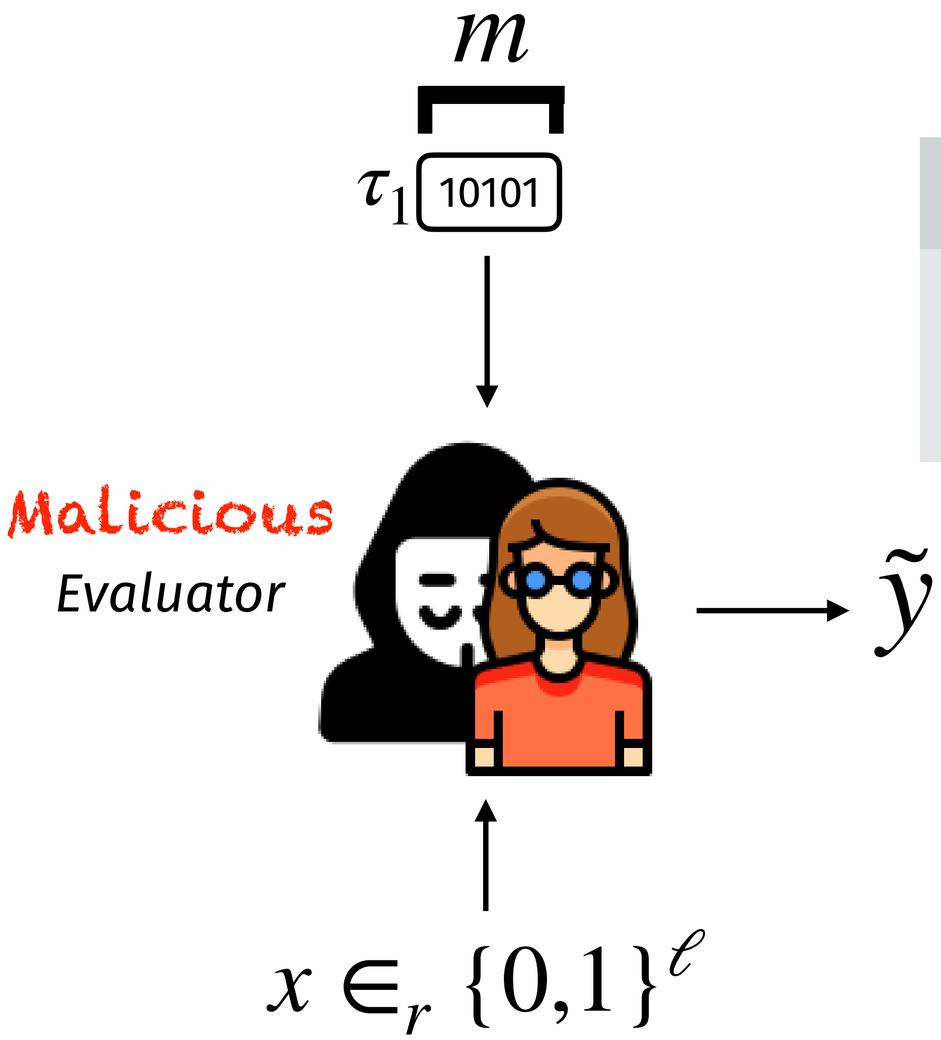
$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

$(\epsilon, m, 1)$–Minimum Capacity



$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

$(\epsilon, m, 1)$–Minimum Capacity



$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

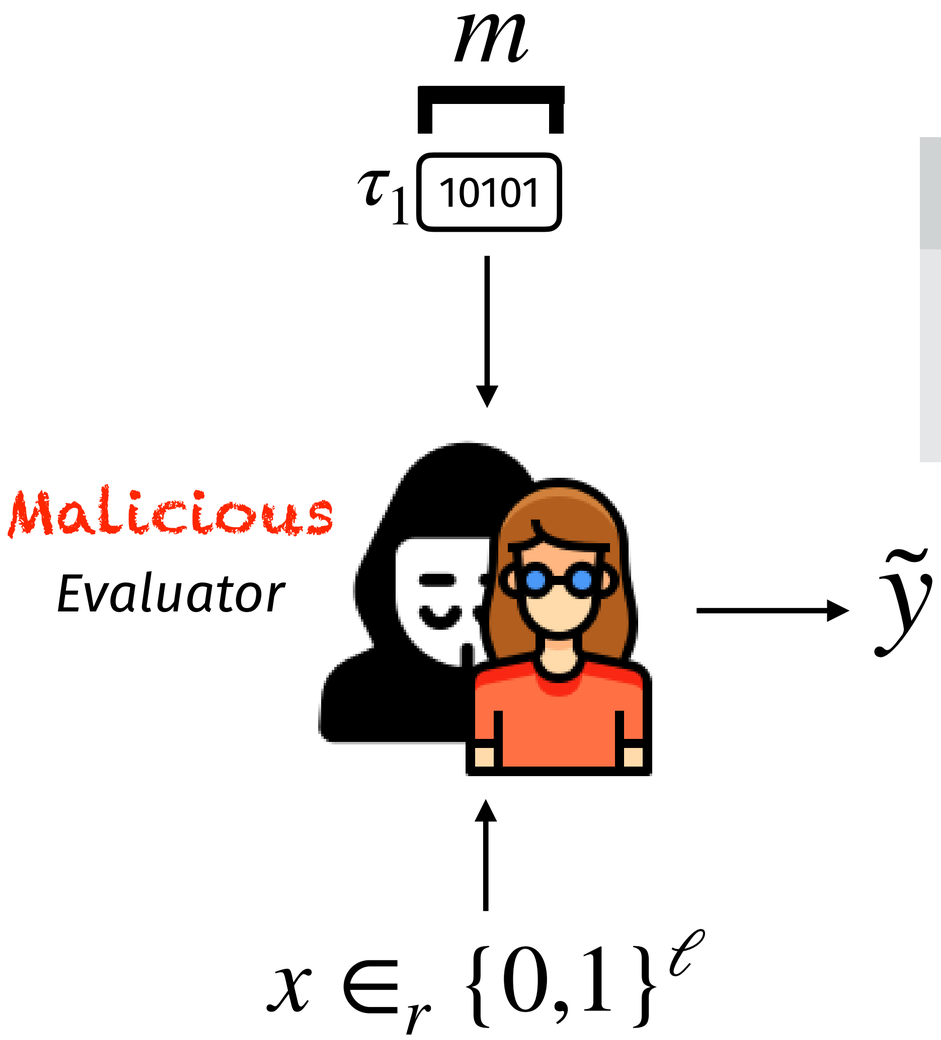
Kolmogorov-bound for polynomial evaluation

If $a_0 || \ldots || a_d$ *is* c*-incompressible*

Then $\forall (x_0, \ldots x_d) \in \mathbb{F}_p^{d+1}$ *we have* $\#bits_{read} = |\alpha| \gtrapprox \cdot (d+1)(\lambda - \ell) - c$

m

$(\epsilon, m, 1)$–Minimum Capacity



$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

Kolmogorov-bound for polynomial evaluation

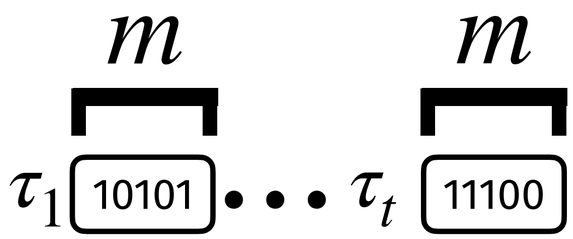
If $a_0 || \ldots || a_d$ *is* c*-incompressible*

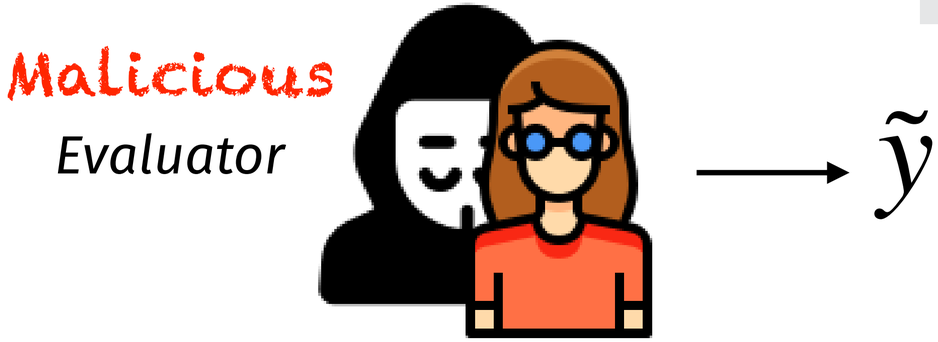
Then $\forall (x_0, \ldots x_d) \in \mathbb{F}_p^{d+1}$ *we have* $\#bits_{read} = |\alpha| \gtrapprox \cdot (d+1)(\lambda - \ell) - c$

m

$$\Pr[\tilde{y} = F(x)] \leq \epsilon = \frac{d}{2^\ell}$$

(ϵ, m, t)–Minimum Capacity





$x \in_r \{0,1\}^{\ell}$

$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

Kolmogorov-bound for polynomial evaluation

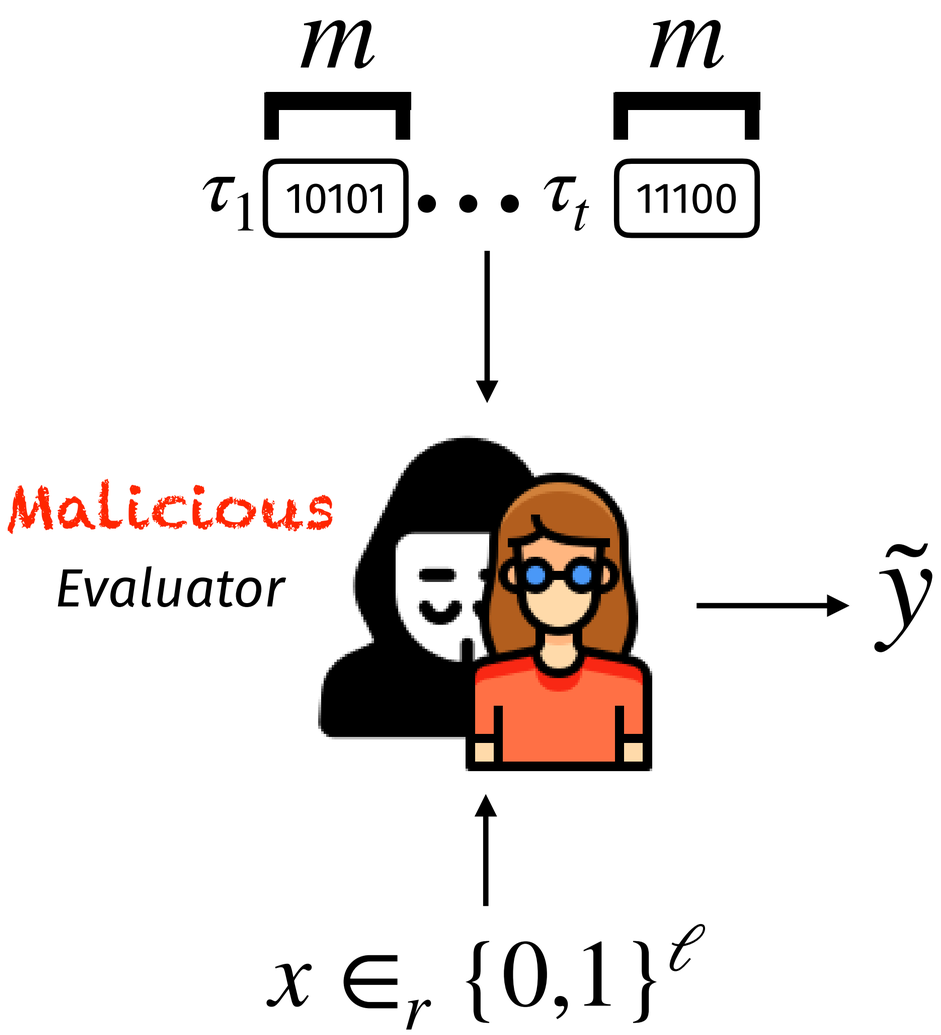
If $a_0 || \ldots || a_d$ *is* c*-incompressible*

Then $\forall (x_0, \ldots x_d) \in \mathbb{F}_p^{d+1}$ *we have* $\#bits_{read} = |\alpha| \gtrapprox \cdot (d+1)(\lambda - \ell) - c$

m

$$\Pr[\tilde{y} = F(x)] \leq \epsilon = \frac{d}{2^{\ell}} \cdot t$$

(ϵ, m, t)–Minimum Capacity

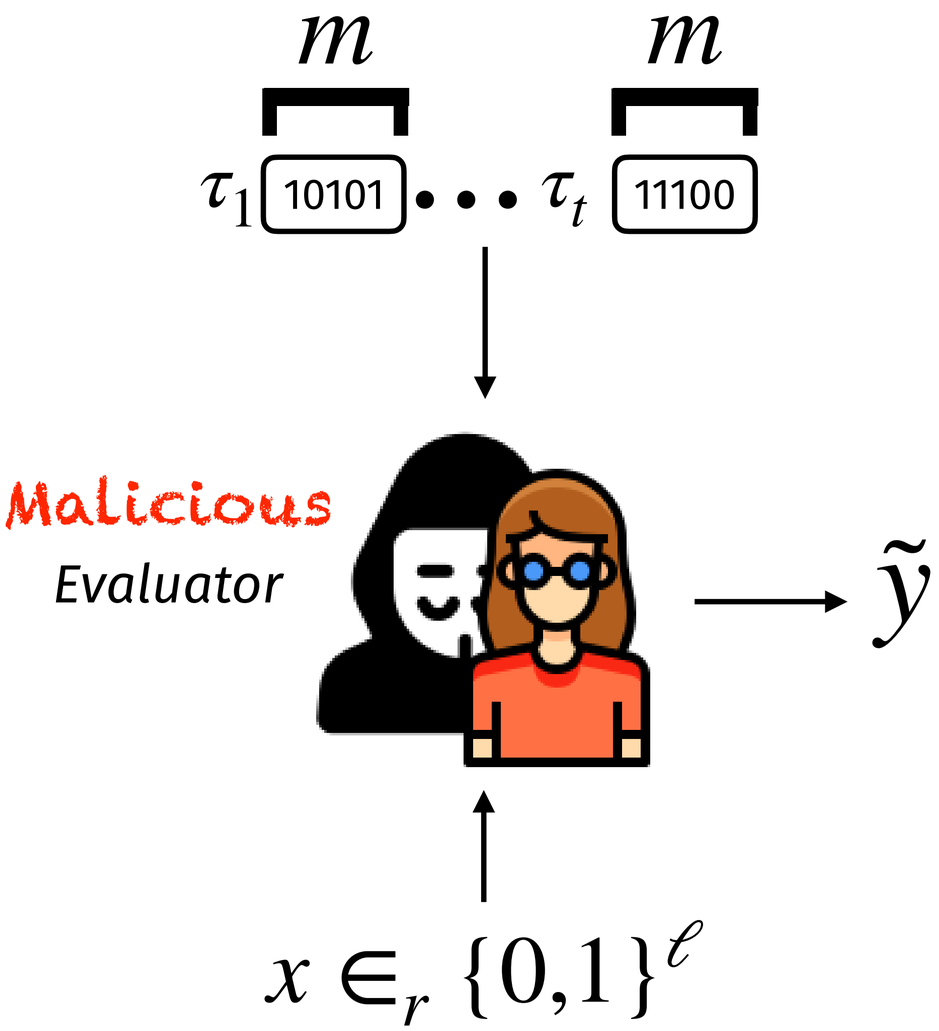


$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

Number of c-incompressible strings

$$\geq 2^{(d+1)\cdot\lambda} \cdot (1 - 2^{-c}) + 1$$

(ϵ, m, t)–Minimum Capacity



$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

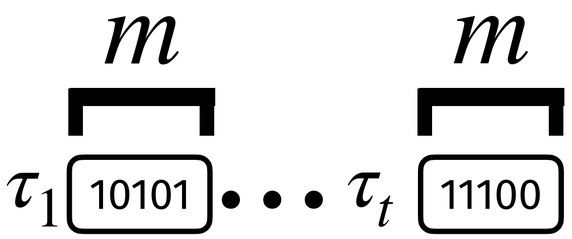
Number of c-incompressible strings

$$\geq 2^{(d+1)\cdot\lambda} \cdot (1 - 2^{-c}) + 1$$

Prob. of sampling a c-incompressible string

$$\geq 1 - 2^{-c} + \frac{1}{2^{(d+1)\cdot\lambda}} = 1 - \mathsf{negl}(\lambda)$$

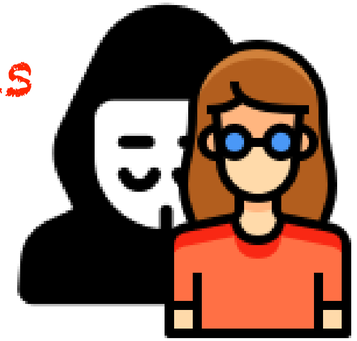
(ϵ, m, t)–Minimum Capacity



Sampled on Setup

$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

Malicious
Evaluator



$\longrightarrow \tilde{y}$

$x \in_r \{0,1\}^{\ell}$

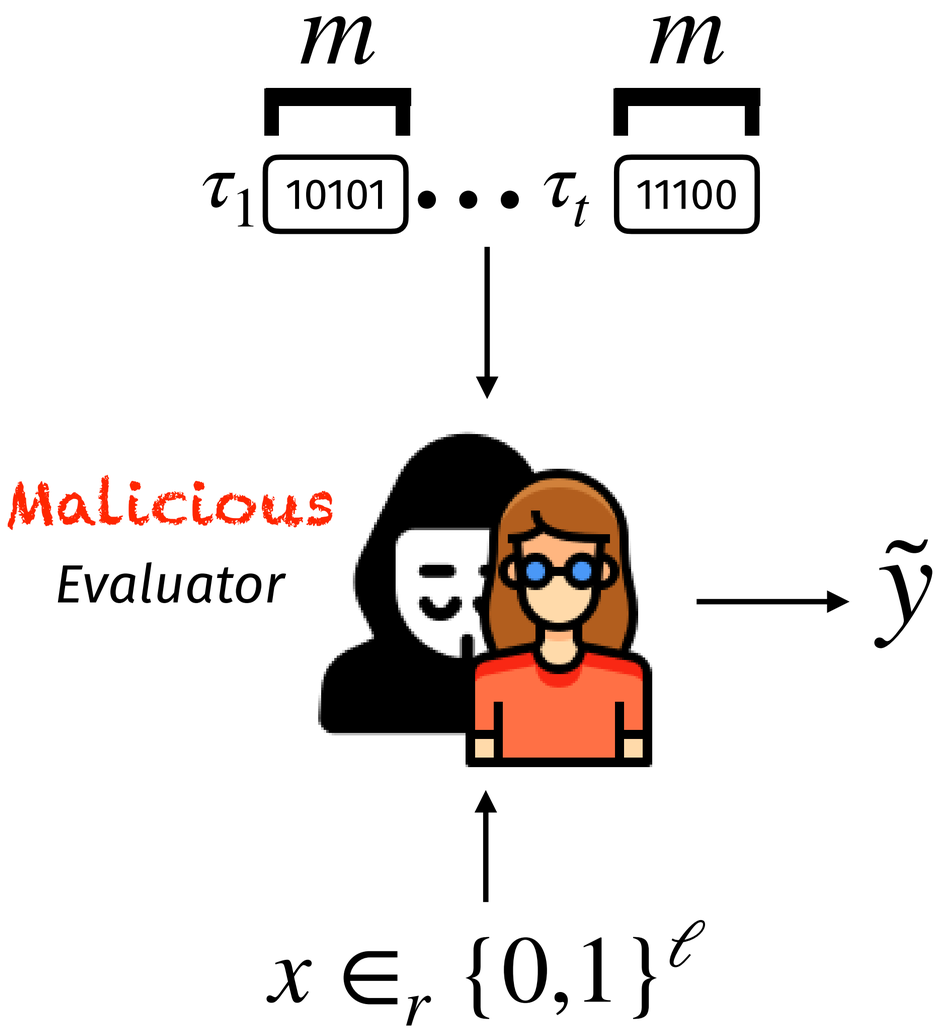
Number of c-incompressible strings

$$\geq 2^{(d+1)\cdot\lambda} \cdot (1 - 2^{-c}) + 1$$

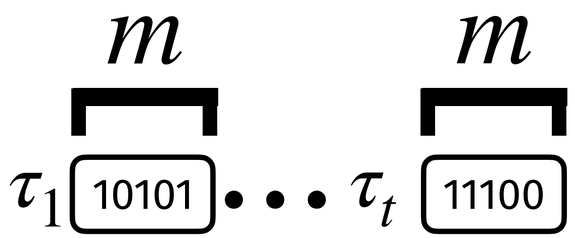
Prob. of sampling a c-incompressible string

$$\geq 1 - 2^{-c} + \frac{1}{2^{(d+1)\cdot\lambda}} = 1 - \mathsf{negl}(\lambda)$$

(ϵ, m, t)–Minimum Capacity



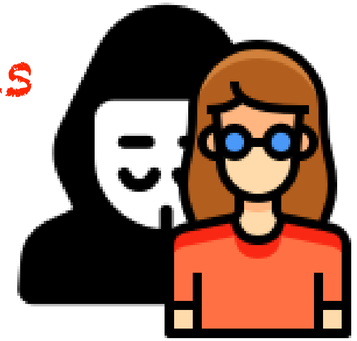
(ϵ, m, t)–Minimum Capacity



$$m \approx (d+1) \cdot \lambda$$

Malicious

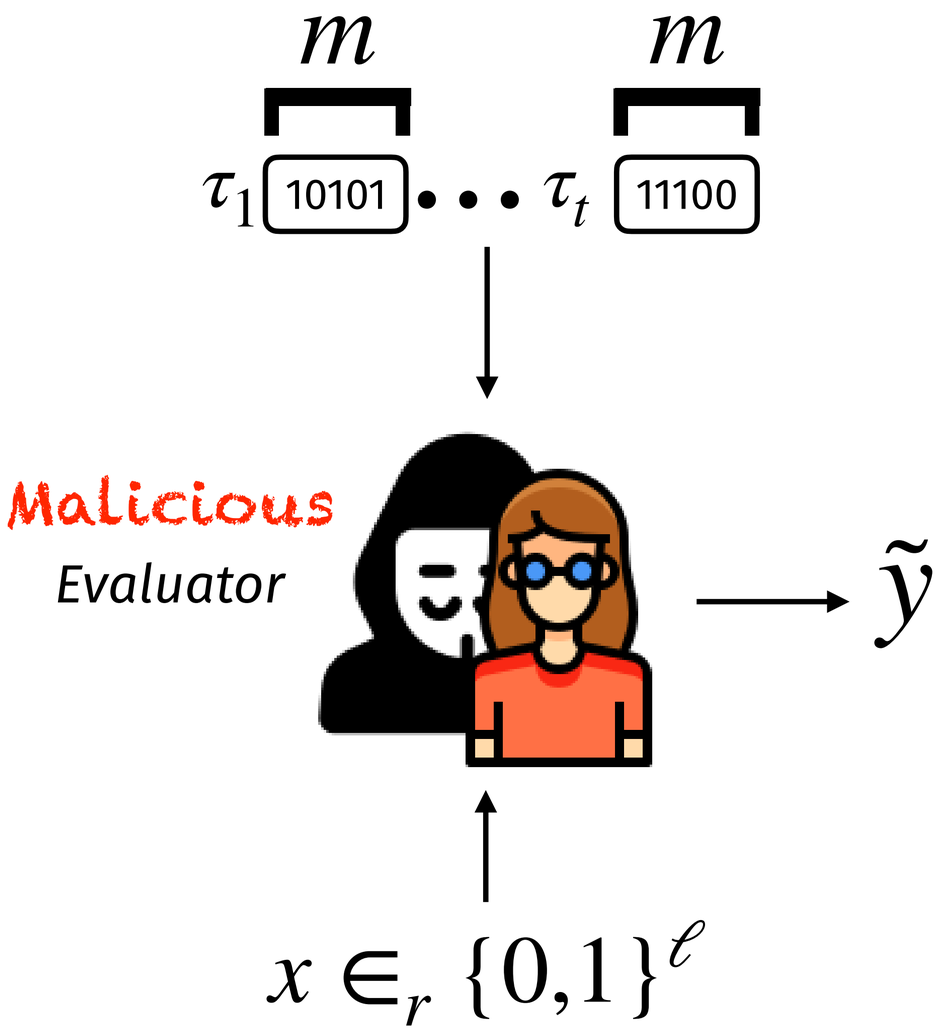
Evaluator



$\longrightarrow \tilde{y}$

$x \in_r \{0,1\}^{\ell}$

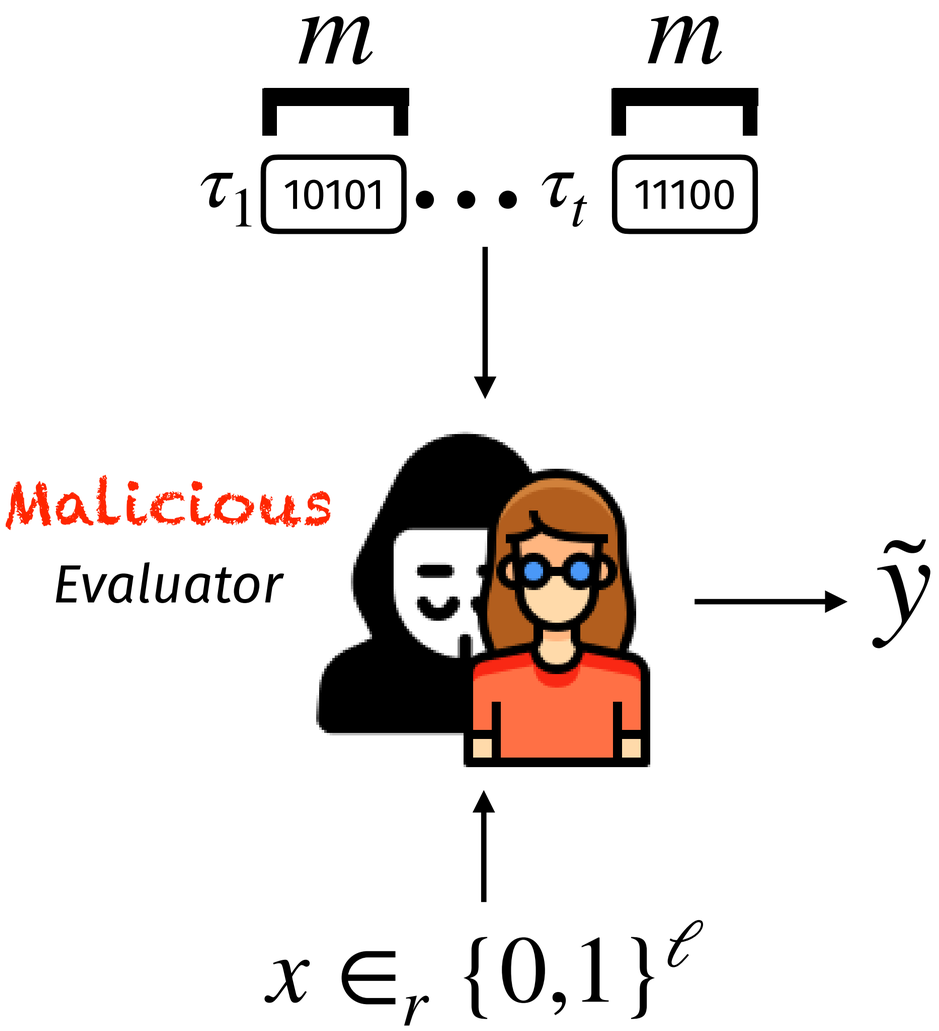
(ϵ, m, t)–Minimum Capacity



$$m \approx (d+1) \cdot \lambda$$

$$\epsilon \approx \epsilon_1 + \frac{d}{2^{\ell}} \in \mathsf{negl}(\lambda)$$

(ϵ, m, t)–Minimum Capacity

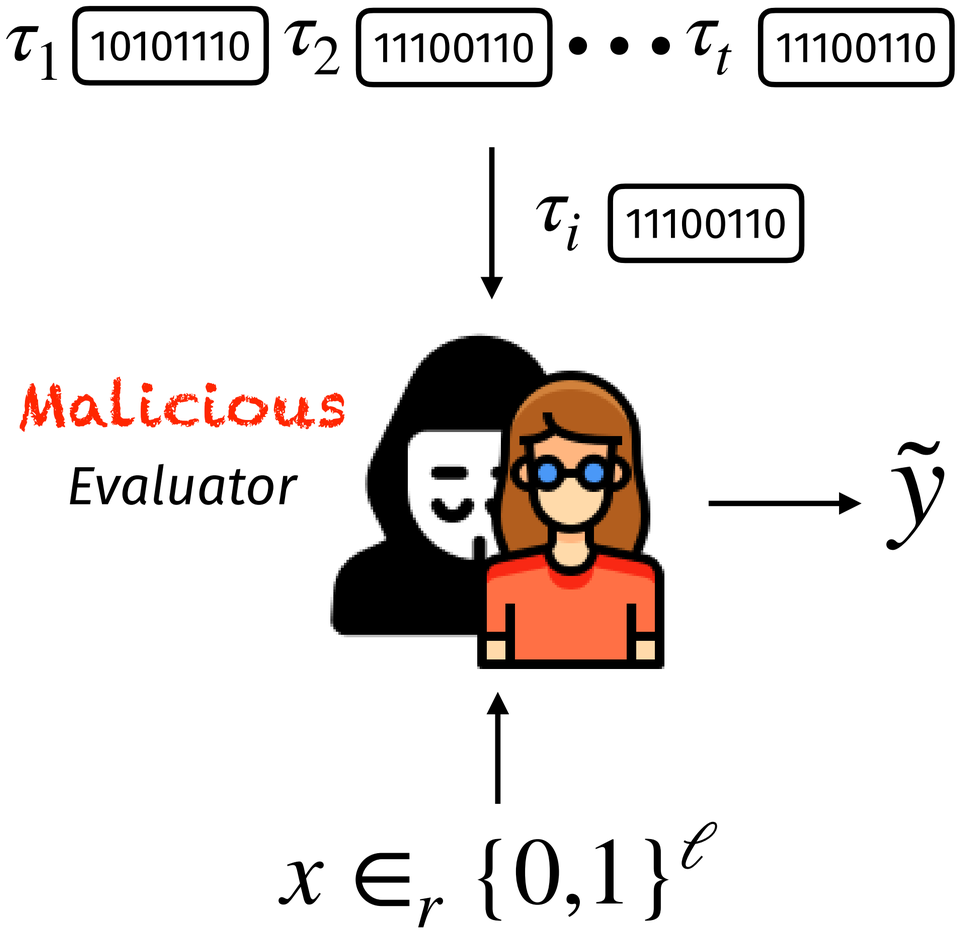


$$m \approx (d+1) \cdot \lambda$$

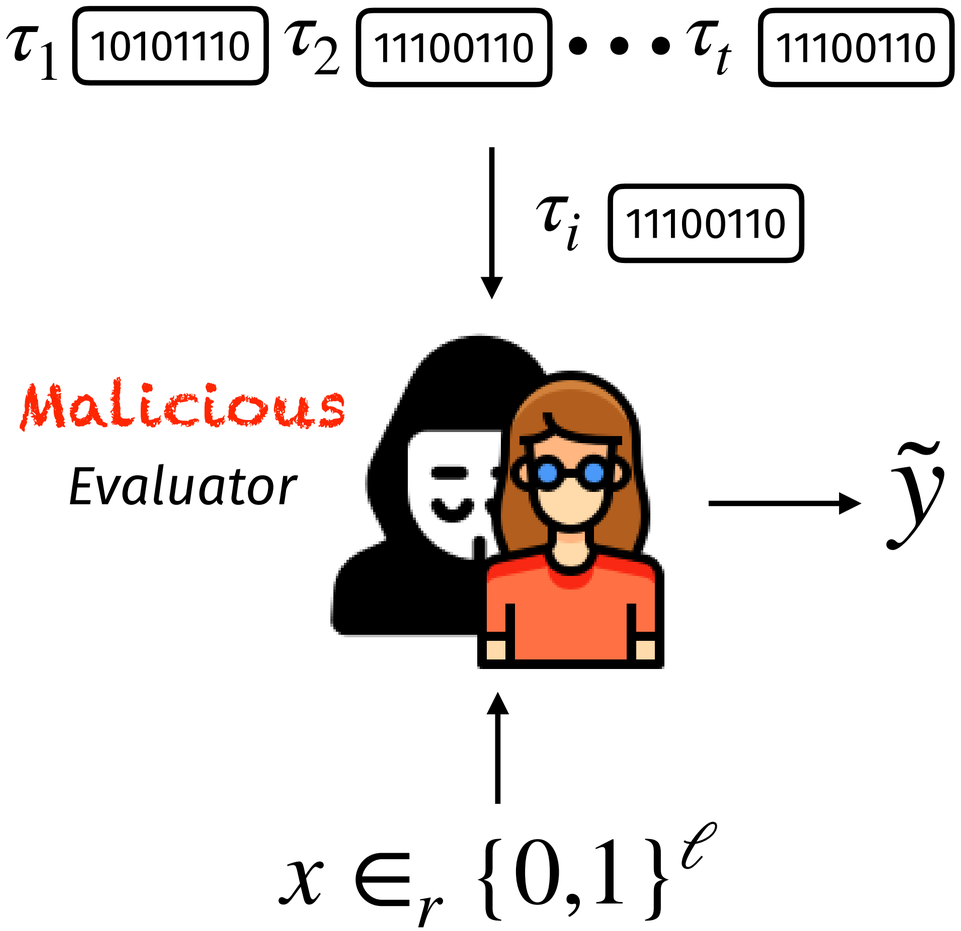
$$\epsilon \approx \epsilon_1 + \frac{d}{2^\ell} \in \mathsf{negl}(\lambda)$$

$$t = \frac{\epsilon_1 \cdot 2^\ell}{d} + 1 \in \exp(\lambda)$$

The GAP: Encoding Abstraction



The GAP: Encoding Abstraction



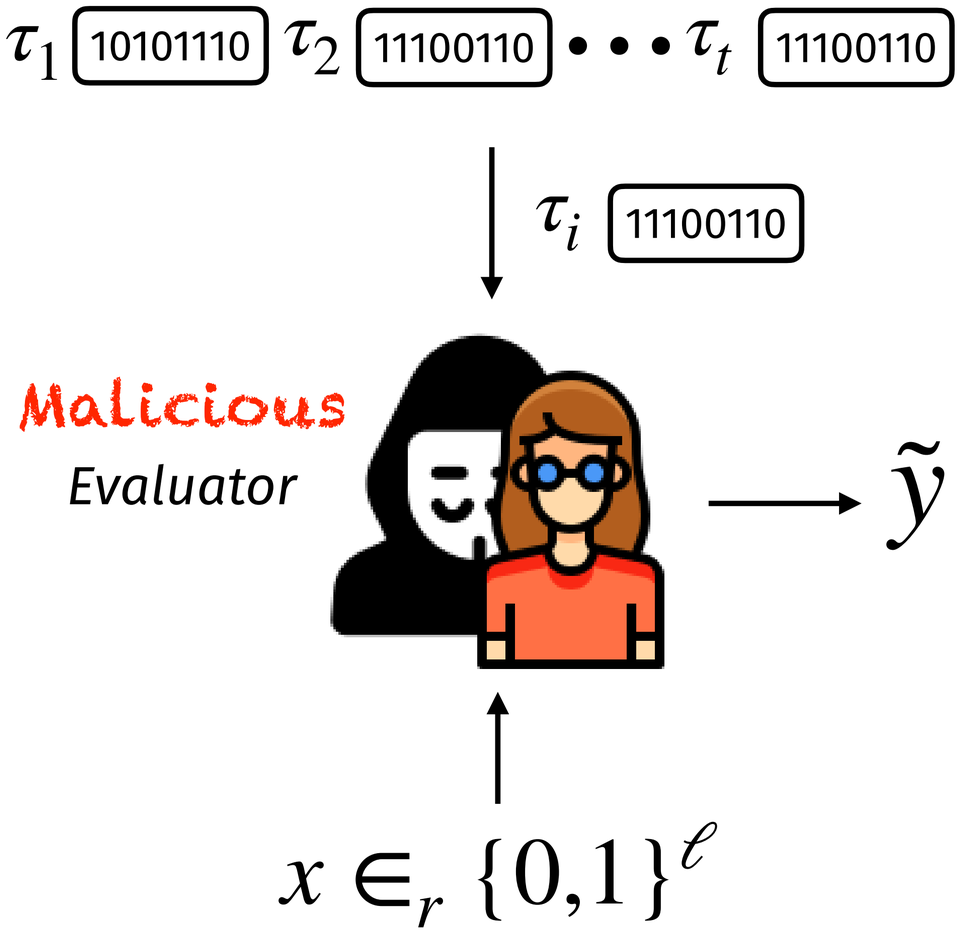
Real world

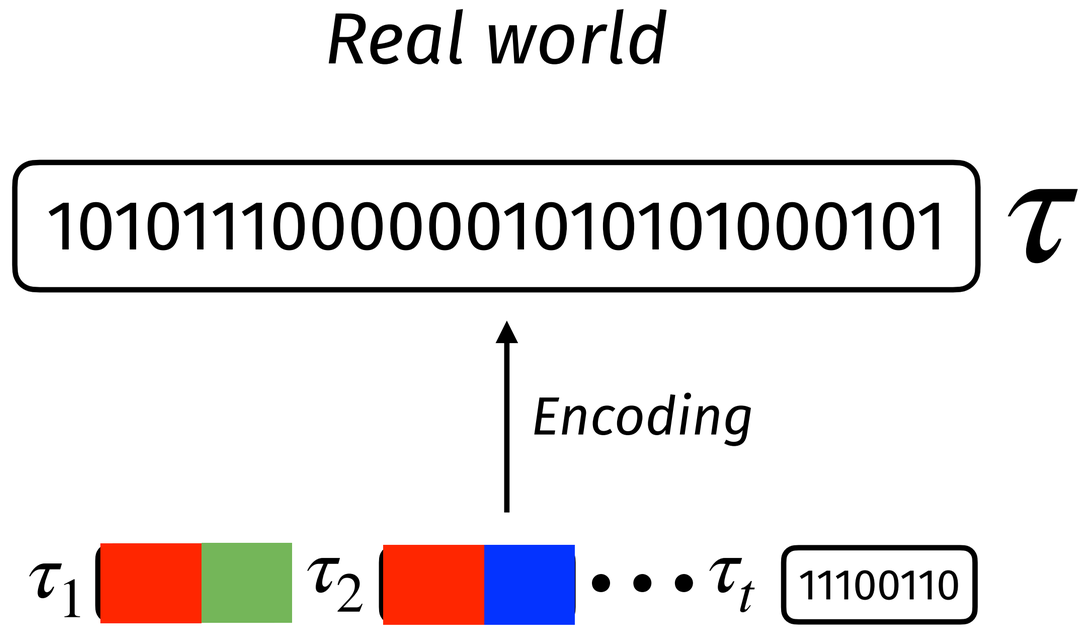
10101110000001010101000101 τ

Encoding

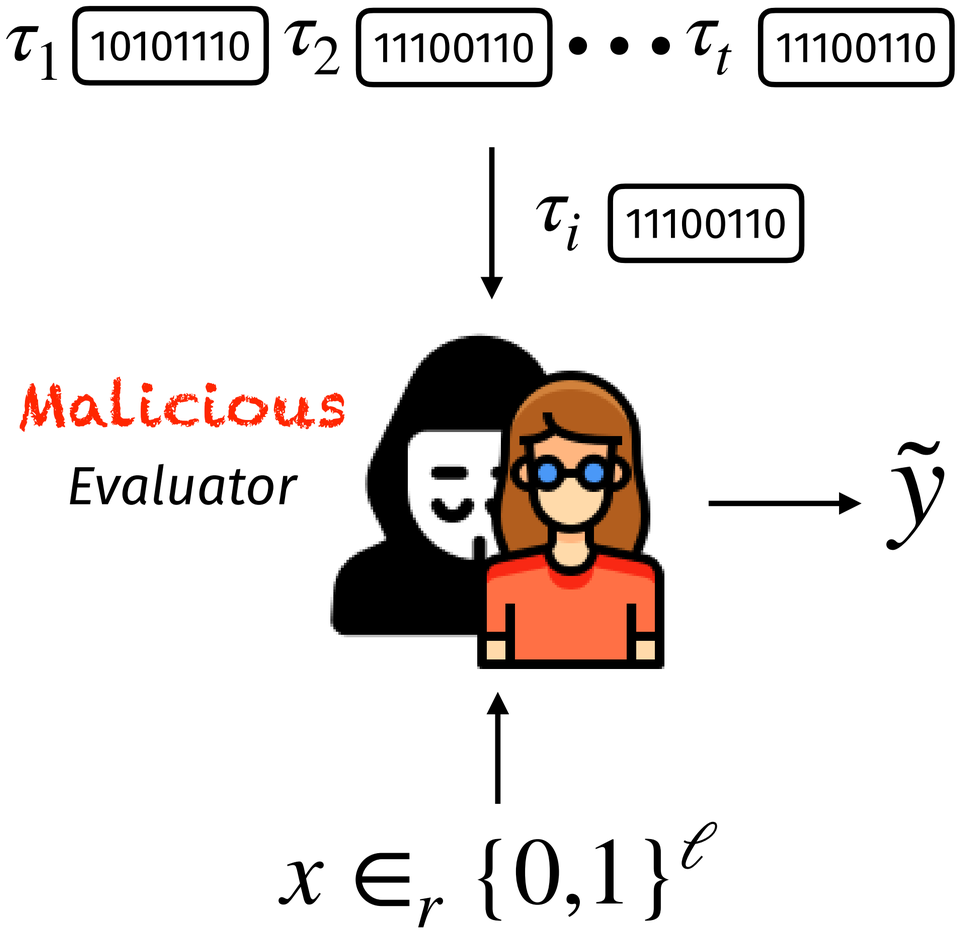
τ_1 10101110 τ_2 11100110 • • • τ_t 11100110

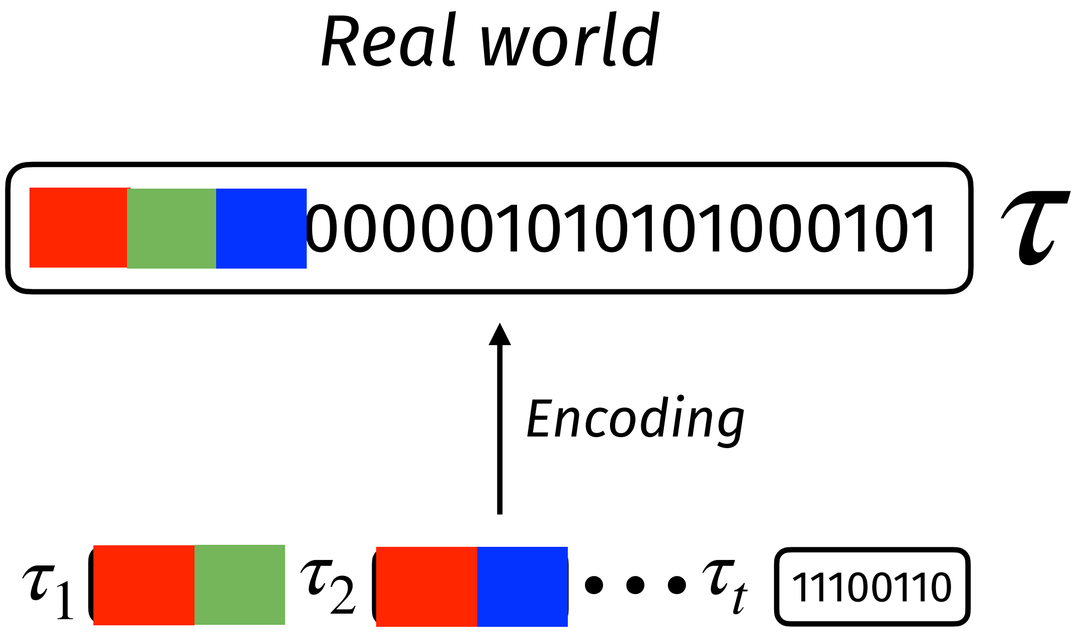
The GAP: Encoding Abstraction



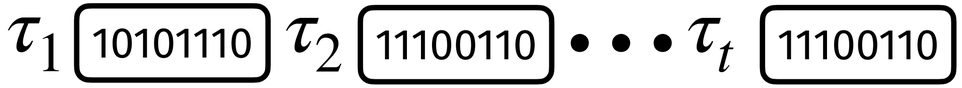


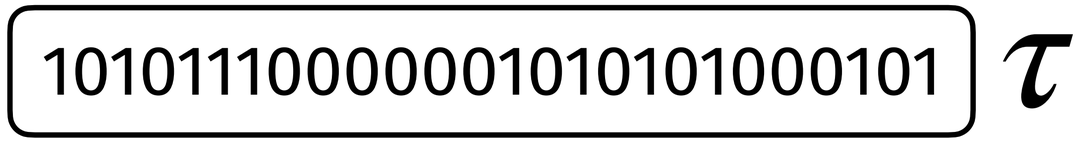
The GAP: Encoding Abstraction



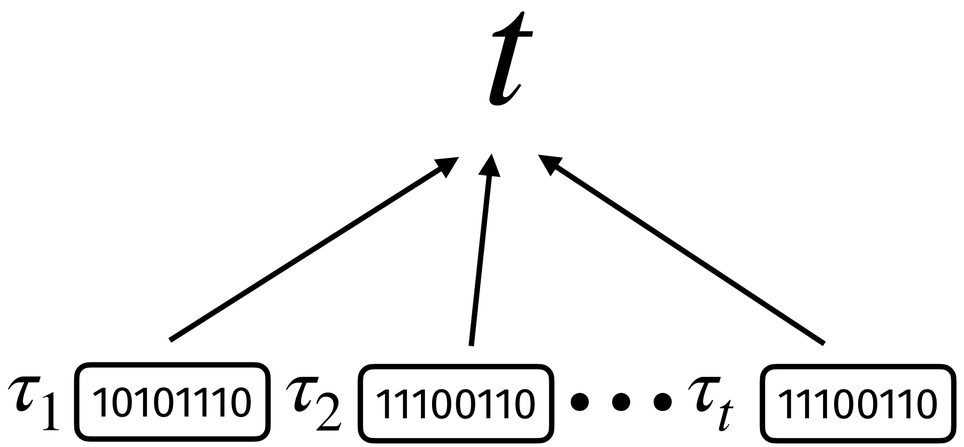


Relation between the two definitions



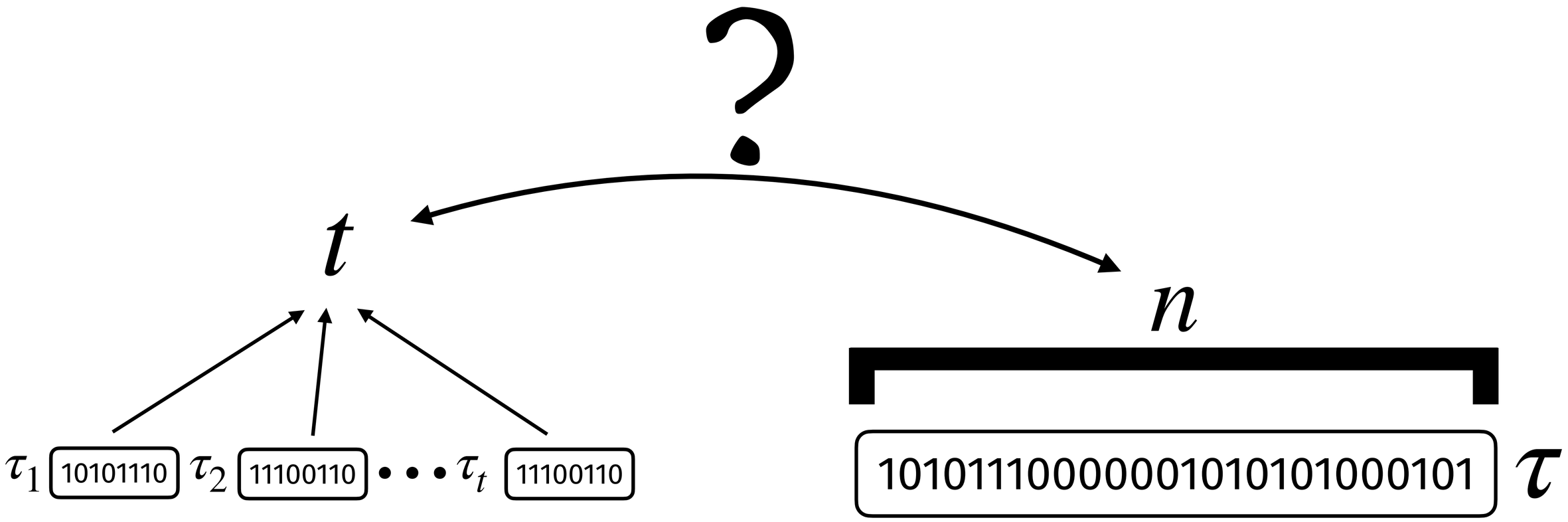


Relation between the two definitions

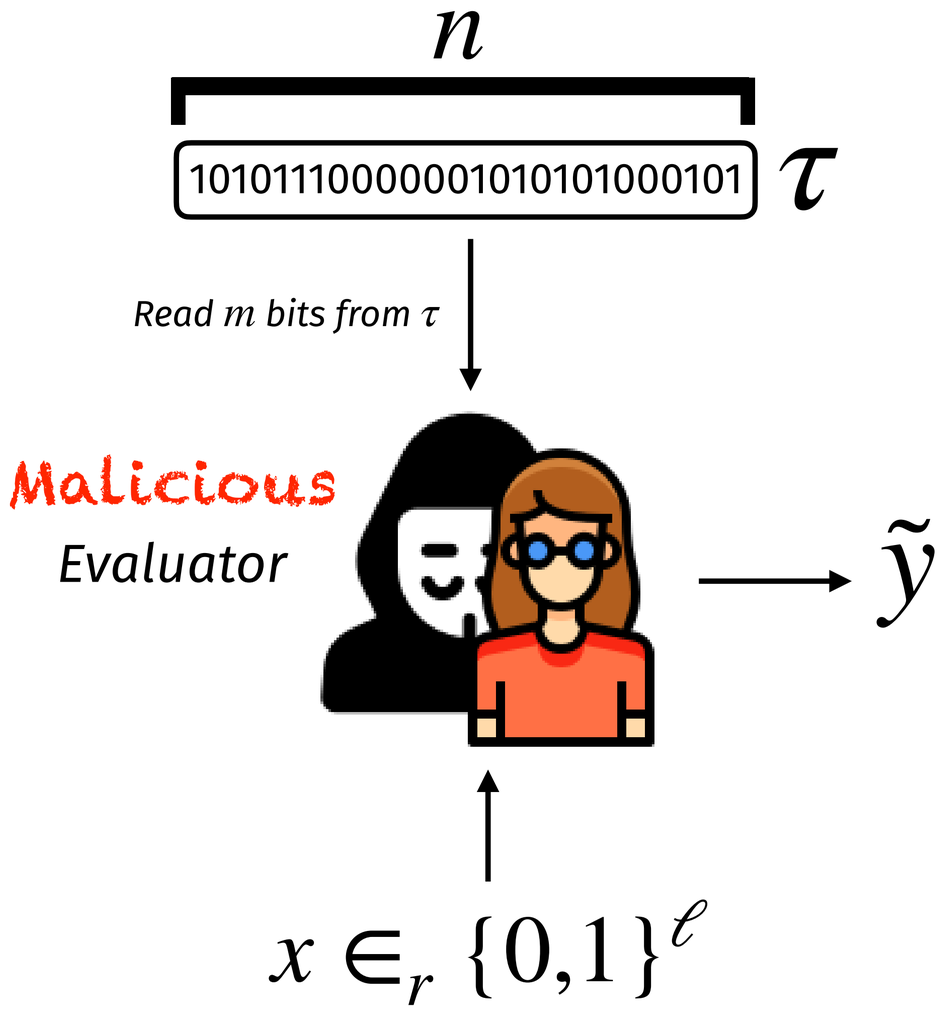


10101110000001010101000101 τ

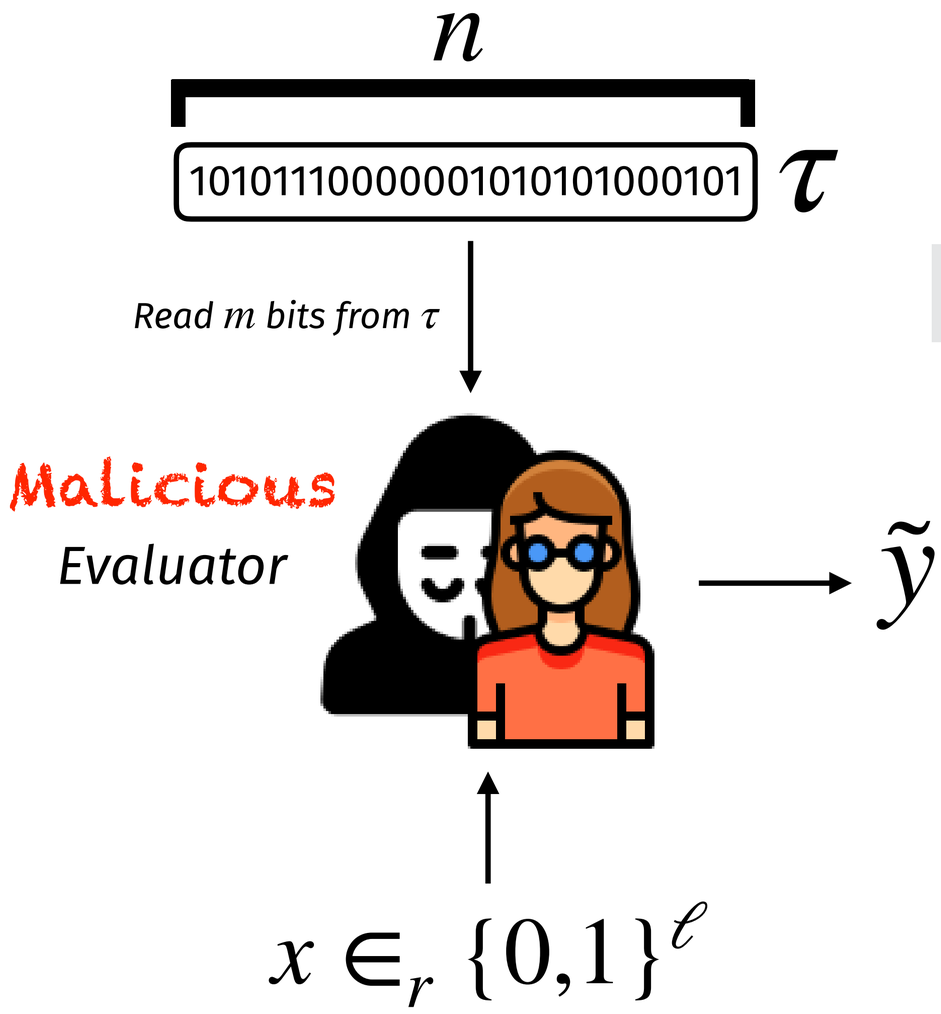
Relation between the two definitions



$(\epsilon, m, n)-$Minimum Capacity

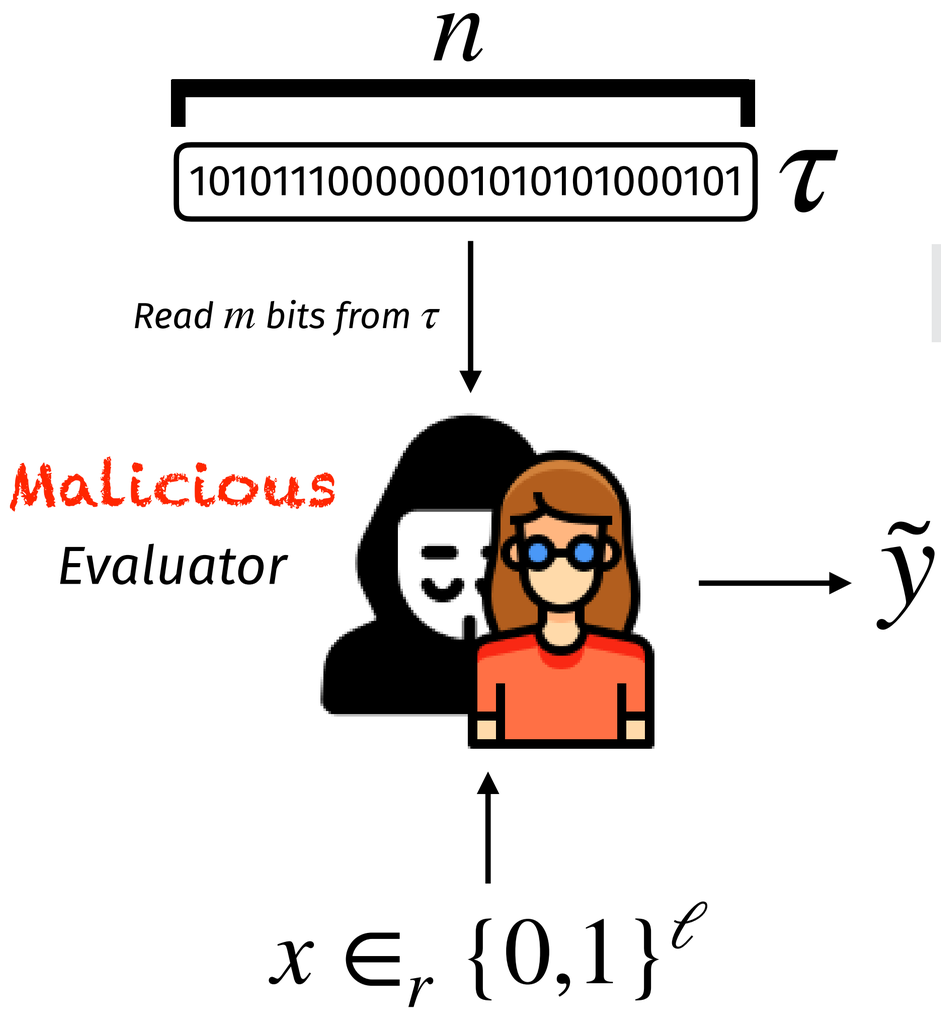


(ϵ, m, n)−Minimum Capacity



$$m \approx (d+1) \cdot \lambda$$

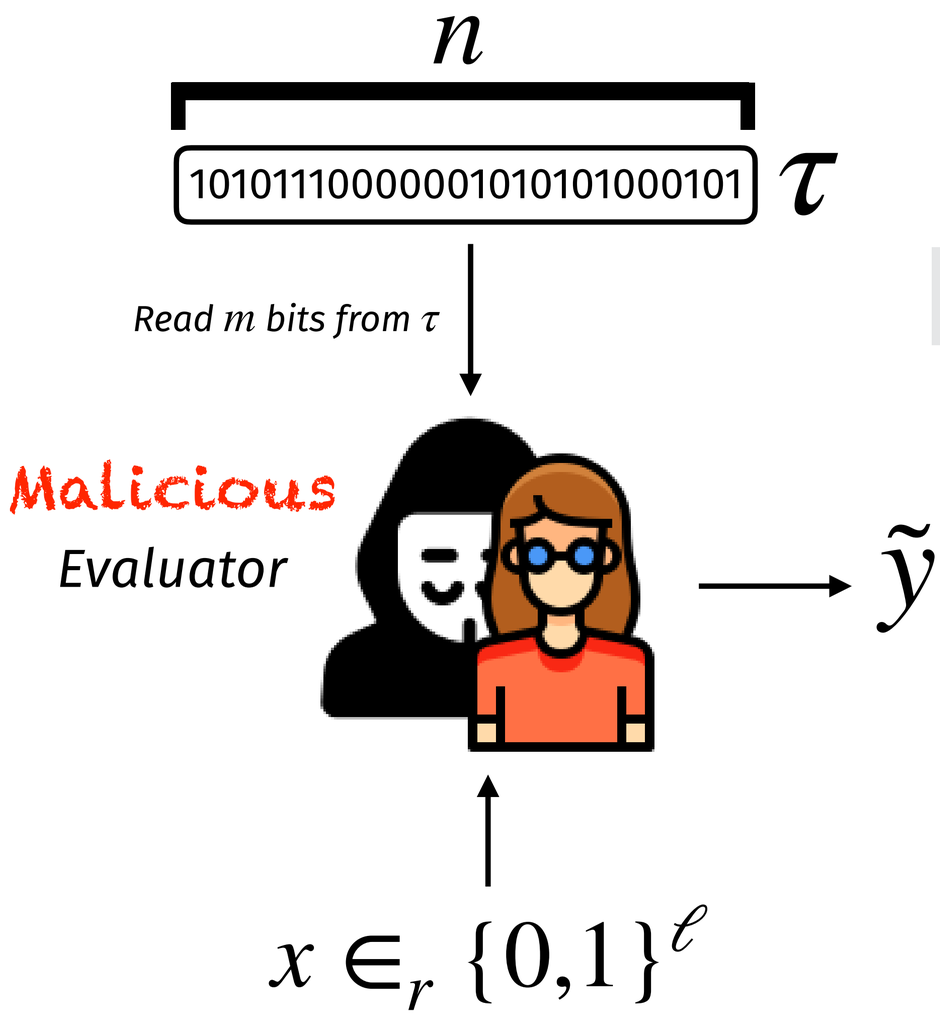
(ϵ, m, n)−Minimum Capacity



$$m \approx (d+1) \cdot \lambda$$

$$\epsilon \approx \epsilon_1 + \frac{d}{2^{\ell}} \in \mathsf{negl}(\lambda)$$

$(\epsilon, m, n)-$Minimum Capacity

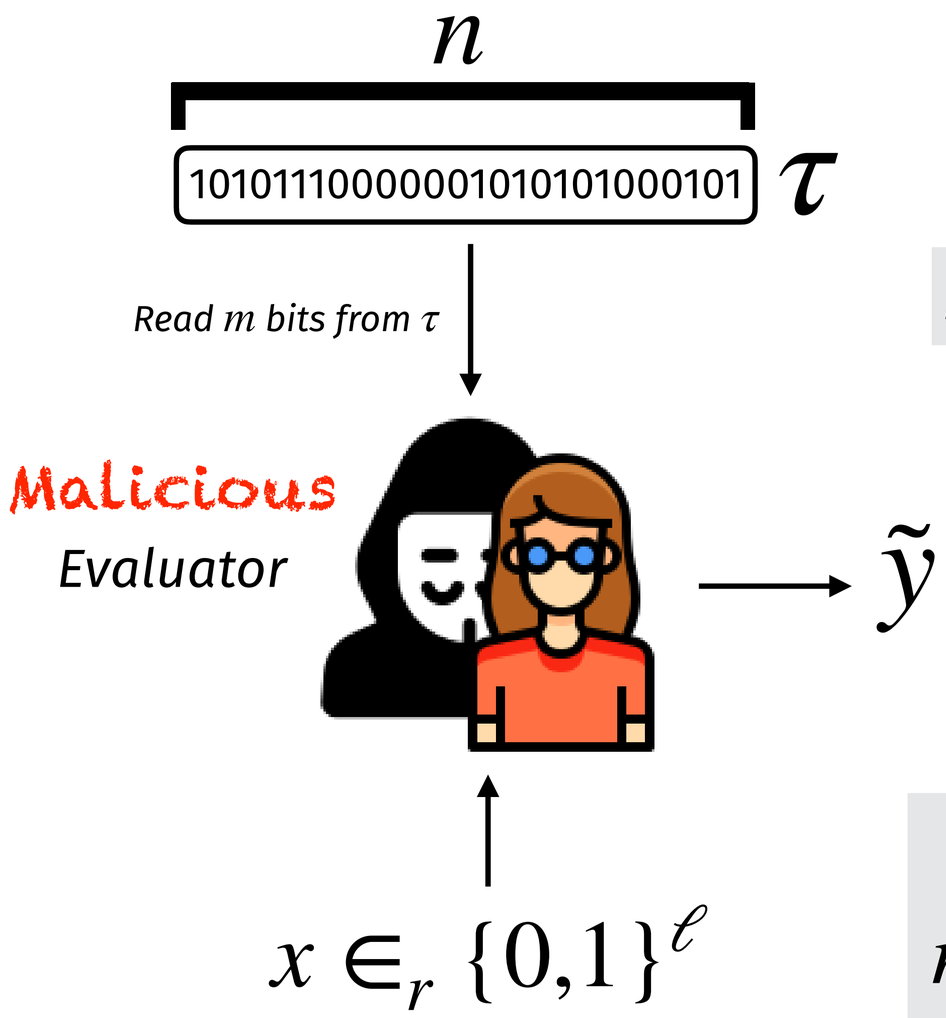


$$m \approx (d+1) \cdot \lambda$$

$$\epsilon \approx \epsilon_1 + \frac{d}{2^{\ell}} \in \mathsf{negl}(\lambda)$$

$$n \in \exp(\lambda) \qquad O(1)-\text{rnd-access}$$

$(\epsilon, m, n)-$Minimum Capacity



$$m \approx (d+1) \cdot \lambda$$

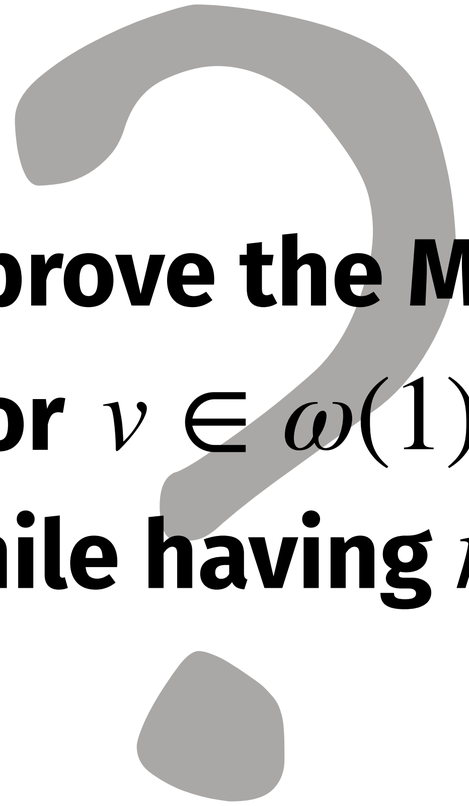
$$\epsilon \approx \epsilon_1 + \frac{d}{2^\ell} \in \mathsf{negl}(\lambda)$$

$$n \in \exp(\lambda)$$

$O(1)-$rnd-access

$$n \approx \sqrt[v]{\frac{\epsilon_1 \cdot 2^\ell}{d} \cdot g(v,m)}$$

$v-$rnd-access
where $v \in \omega(1)$



Improve the Memory Bound n

for $v \in \omega(1)$ rnd-accesses

while having $m \approx (d+1)\lambda$?

Data Structure for Polynomial Evaluation

Kedlaya, Kiran S., and Christopher Umans. "Fast polynomial factorization and modular composition." *SIAM Journal on Computing* 40, no. 6 (2011): 1767-1802.

Data Structure for Polynomial Evaluation

Precomputation of $F(\cdot)$

Precompute $F(x) = a_0 + a_1 \cdot x_1 + \ldots a_d \cdot x^d$ and obtain a data structure D of size

$$|D| = d^{1+\delta} \log^{1+o(1)}(2^\lambda)$$

where $\delta > 0$ is an arbitrary constant

Kedlaya, Kiran S., and Christopher Umans. "Fast polynomial factorization and modular composition." *SIAM Journal on Computing* 40, no. 6 (2011): 1767-1802.

Data Structure for Polynomial Evaluation

Precomputation of $F(\cdot)$

Precompute $F(x) = a_0 + a_1 \cdot x_1 + \ldots a_d \cdot x^d$ and obtain a data structure D of size

$$|D| = d^{1+\delta} \log^{1+o(1)}(2^\lambda)$$

where $\delta > 0$ is an arbitrary constant

$$n \approx (d^{1+\delta})\lambda$$

Kedlaya, Kiran S., and Christopher Umans. "Fast polynomial factorization and modular composition." *SIAM Journal on Computing* 40, no. 6 (2011): 1767-1802.

Data Structure for Polynomial Evaluation

Precomputation of $F(\cdot)$

Precompute $F(x) = a_0 + a_1 \cdot x_1 + \ldots a_d \cdot x^d$ and obtain a data structure D of size

$$|D| = d^{1+\delta} \log^{1+o(1)}(2^\lambda)$$

where $\delta > 0$ is an arbitrary constant

$$n \approx (d^{1+\delta})\lambda$$

Evaluation of $F(x) = y$

Given any x, there exists an algorithm with random access to D that computes $F(x)$ in time $\text{polylog}(d) \cdot \log^{1+o(1)}(2^\lambda)$

Kedlaya, Kiran S., and Christopher Umans. "Fast polynomial factorization and modular composition." *SIAM Journal on Computing* 40, no. 6 (2011): 1767-1802.

Data Structure for Polynomial Evaluation

Precomputation of $F(\cdot)$

Precompute $F(x) = a_0 + a_1 \cdot x_1 + \ldots a_d \cdot x^d$ and obtain a data structure D of size

$$|D| = d^{1+\delta} \log^{1+o(1)}(2^\lambda)$$

where $\delta > 0$ is an arbitrary constant

$$n \approx (d^{1+\delta})\lambda$$

Evaluation of $F(x) = y$

Given any x, there exists an algorithm with random access to D that computes $F(x)$ in time $\text{polylog}(d) \cdot \log^{1+o(1)}(2^\lambda)$

$$m \in \text{polylog}(d)\lambda$$

Kedlaya, Kiran S., and Christopher Umans. "Fast polynomial factorization and modular composition." *SIAM Journal on Computing* 40, no. 6 (2011): 1767-1802.

Data Structure for Polynomial Evaluation

Precomputation of $F(\cdot)$

Precompute $F(x) = a_0 + a_1 \cdot x_1 + \ldots a_d \cdot x^d$ and obtain a data structure D of size

$$|D| = d^{1+\delta} \log^{1+o(1)}(2^\lambda)$$

where $\delta > 0$ is an arbitrary constant

$$n \approx (d^{1+\delta})\lambda$$

Evaluation of $F(x) = y$

Given any x, there exists an algorithm with random access to D that computes $F(x)$ in time $\text{polylog}(d) \cdot \log^{1+o(1)}(2^\lambda)$

$$m \in \text{polylog}(d)\lambda \ll (d+1)\lambda$$

Kedlaya, Kiran S., and Christopher Umans. "Fast polynomial factorization and modular composition." *SIAM Journal on Computing* 40, no. 6 (2011): 1767-1802.

Soundness

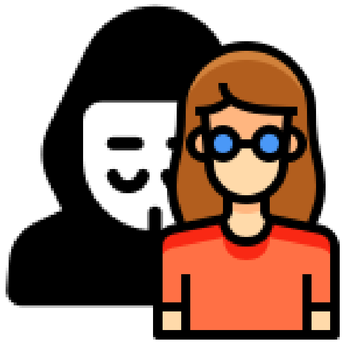
Publicly verifiable + Publicly delegatable

Verifiable Computation for Polynomial Evaluation

$$F(X) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d \in \mathbb{F}_p[X]$$

Malicious *Evaluator*

Honest *Verifier*



x

$\tilde{y}$

π

Proof

$F(x) = \tilde{y}$

Elkhiyaoui, Kaoutar, Melek Önen, Monir Azraoui, and Refik Molva. "Efficient techniques for publicly verifiable delegation of computation." (AsiaCCS 16)

Efficiency

MAX

MIN

Efficiency

MAX

MIN

$\approx m \approx (d+1) \cdot \lambda$

If $O(1)$−rnd-access

Efficiency

MAX

MIN

$\approx m \approx (d+1) \cdot \lambda$

If $O(1)$−rnd-access

$\in O(\lambda)$

Verifiable Computation

Elkhiyaoui, Kaoutar, Melek Önen, Monir Azraoui, and Refik Molva. "Efficient techniques for publicly verifiable delegation of computation." (AsiaCCS 16)

Thank You!