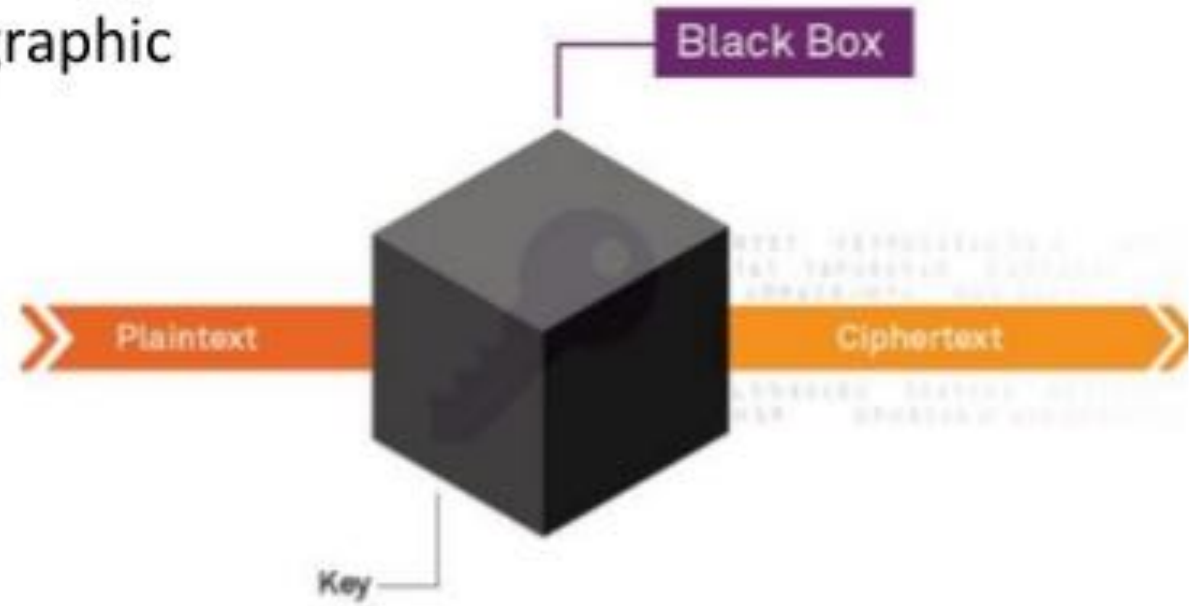
On the Impossibility of code obfuscation

Black-box security

Attacker is assumed to have:

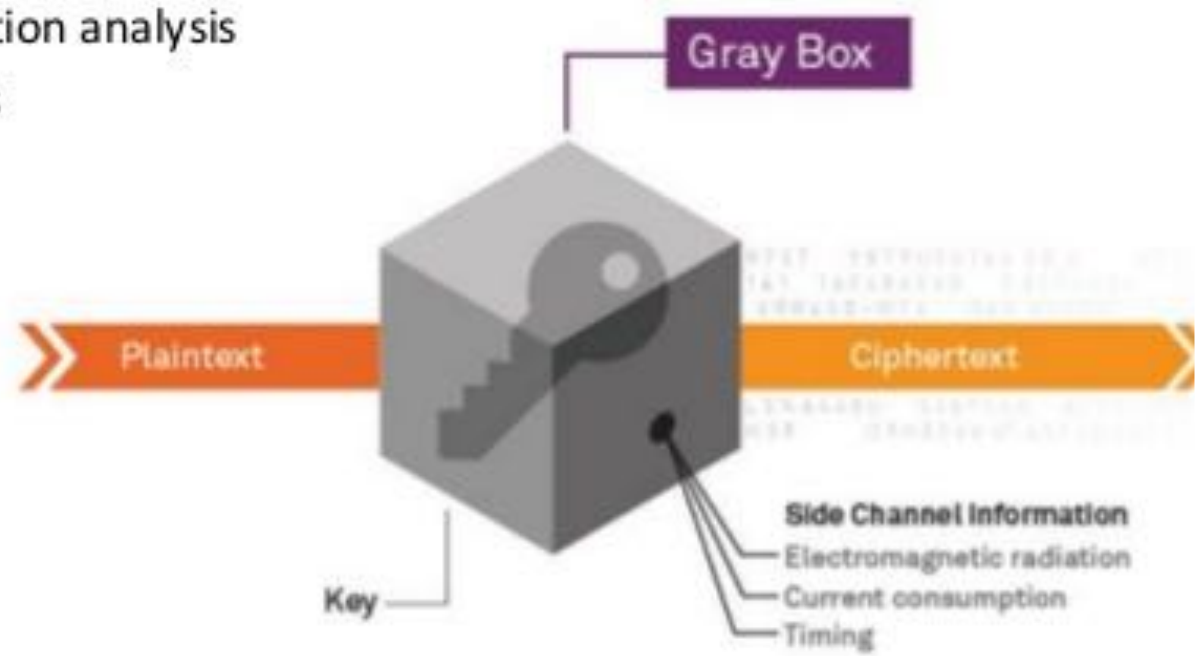
- Zero visibility on code during execution
- External information, such as plaintext or ciphertext
- Considered secure as long as the cipher has no cryptographic weaknesses



Gray-box security

Attacker is assumed to have:

- Partial physical access to the cryptographic key as a result of the cipher leaking side-channel information
  - Electromagnetic radiation analysis
  - Current/power consumption analysis
  - Operation timing analysis

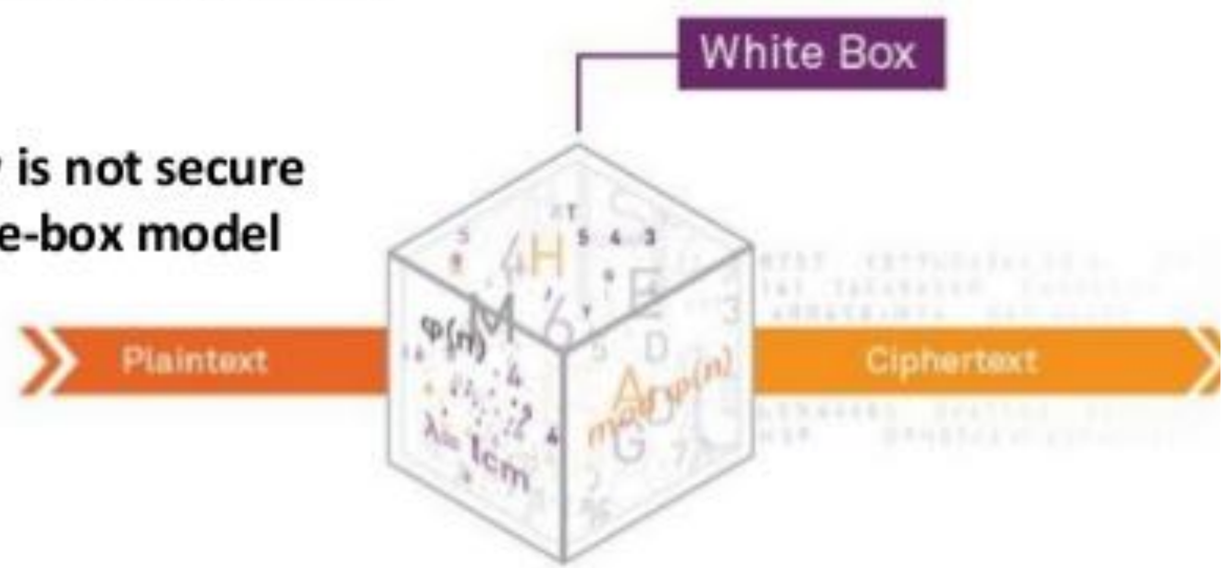


Withe-box security

Attacker is assumed to have:

- Full visibility — inputs, outputs, memory (using debuggers), and intermediate calculations
- Access to the algorithms while watching how they are carried out

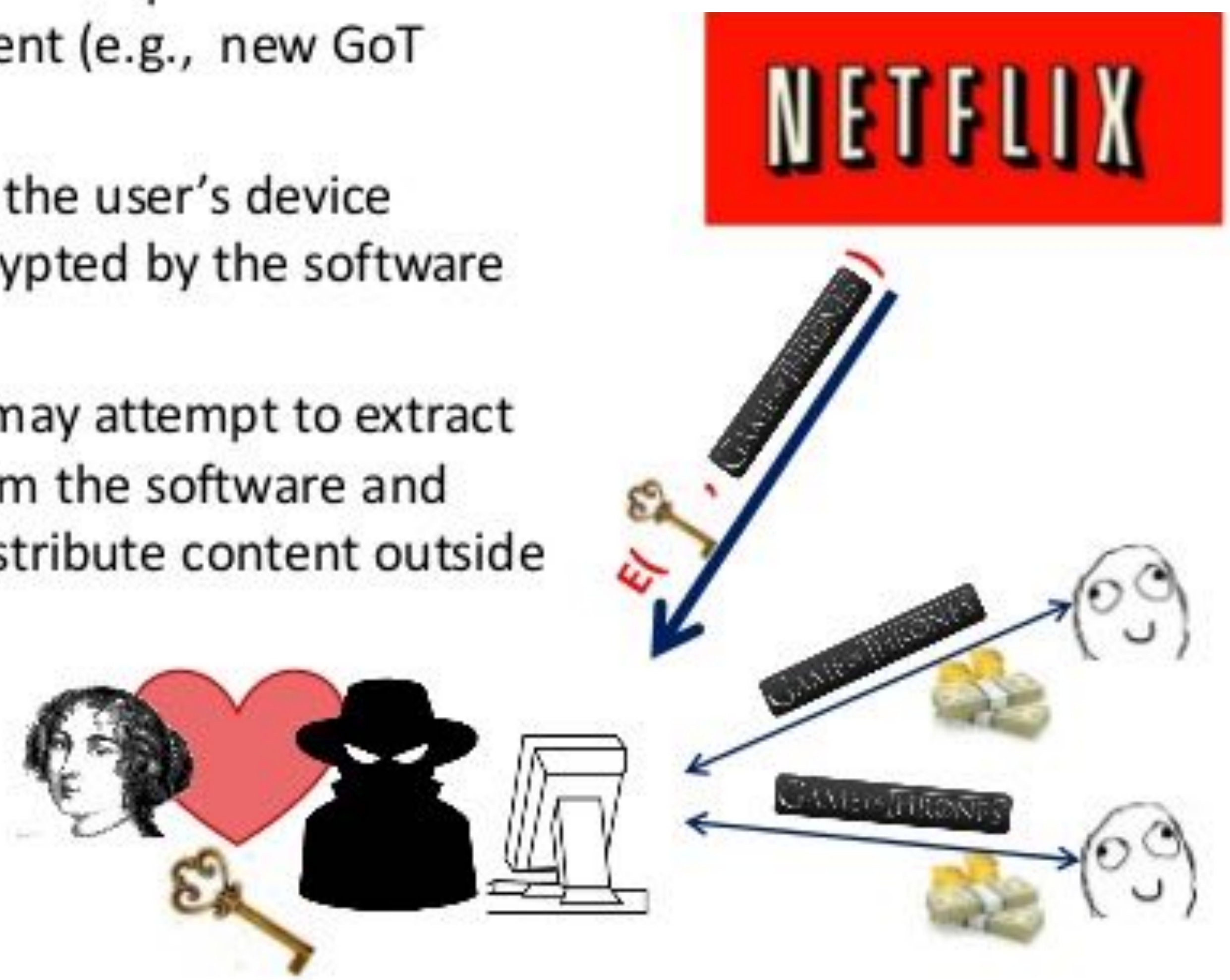
Traditional cryptography is not secure when running in a white-box model



Withe-box security

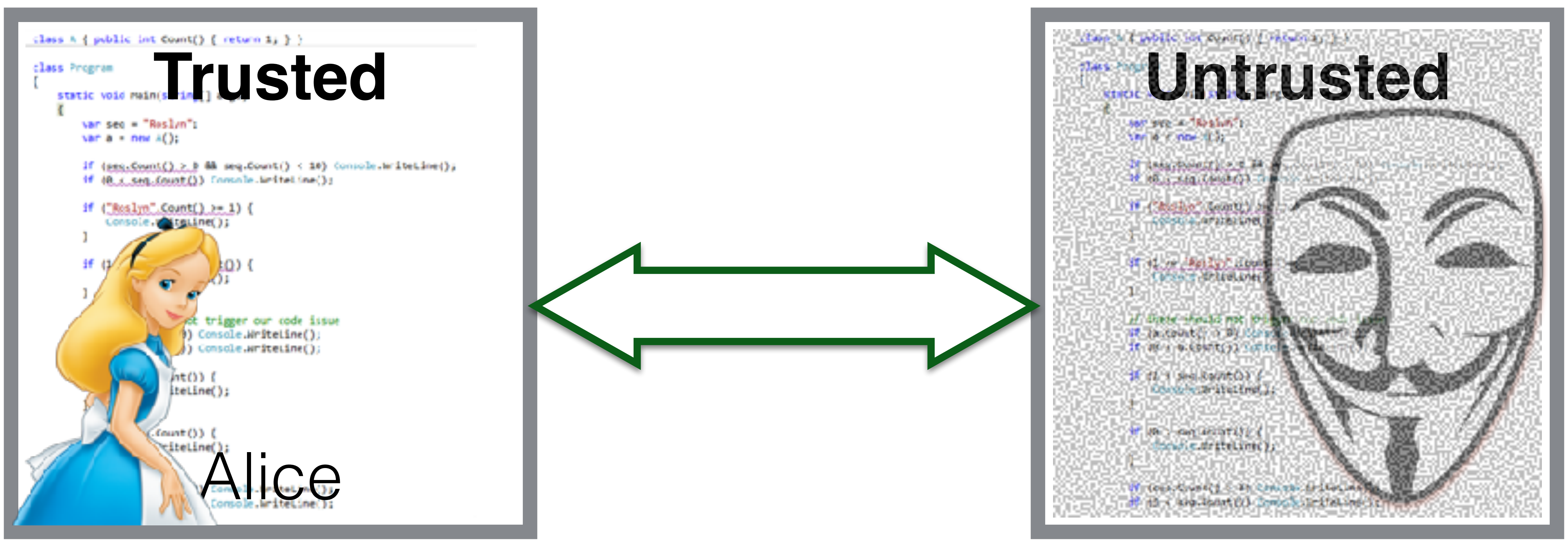
Digital Rights Management Systems

- The end-user is then able to purchase some type of premium content (e.g., new GoT season)
- The content arrives at the user's device encrypted, and is decrypted by the software as it is viewed
- A malicious end-user may attempt to extract cryptographic keys from the software and then use them to redistribute content outside the DRM system



MATE Assumption
The white-box crypto assumption

Man At The End

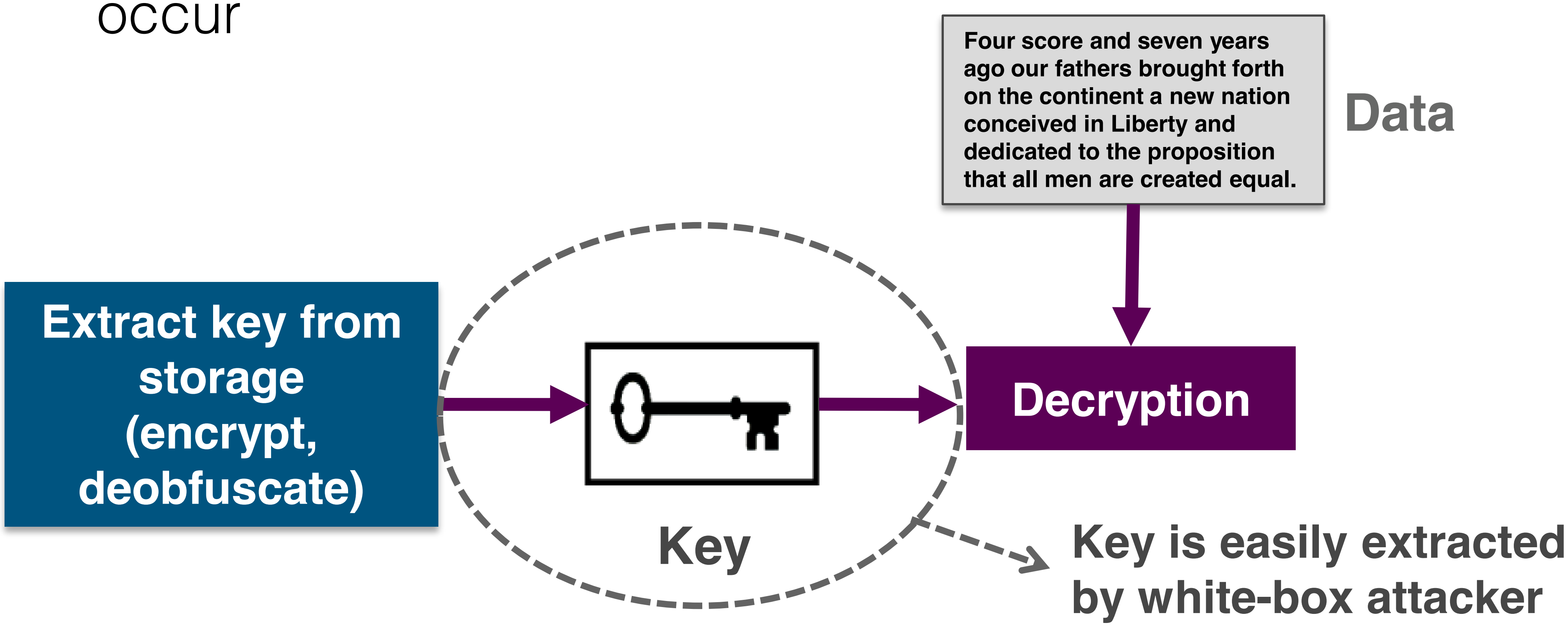


Hostile Environment

- ✓ For applications that run in an hostile environment, cryptographic keys and other valuable assets become much easier and common attack targets for a multitude of purposes than in a trusted environment
- ✓ In most business models, the recovery of some or all of these keys directly threatens the revenue from the applications, services or digital assets

Hostile Environment

- ✓ Software keys can be:
  - Generated using high-quality pseudo random number generator (PRNG)
  - Securely stored
- ✓ Sooner or later the key is used and the following events occur



Goals

- ✓ The fundamental security intent of white-box cryptography is to *make the recovery of the key in the withe-box context at least as difficult, mathematically, as in the black-box context*

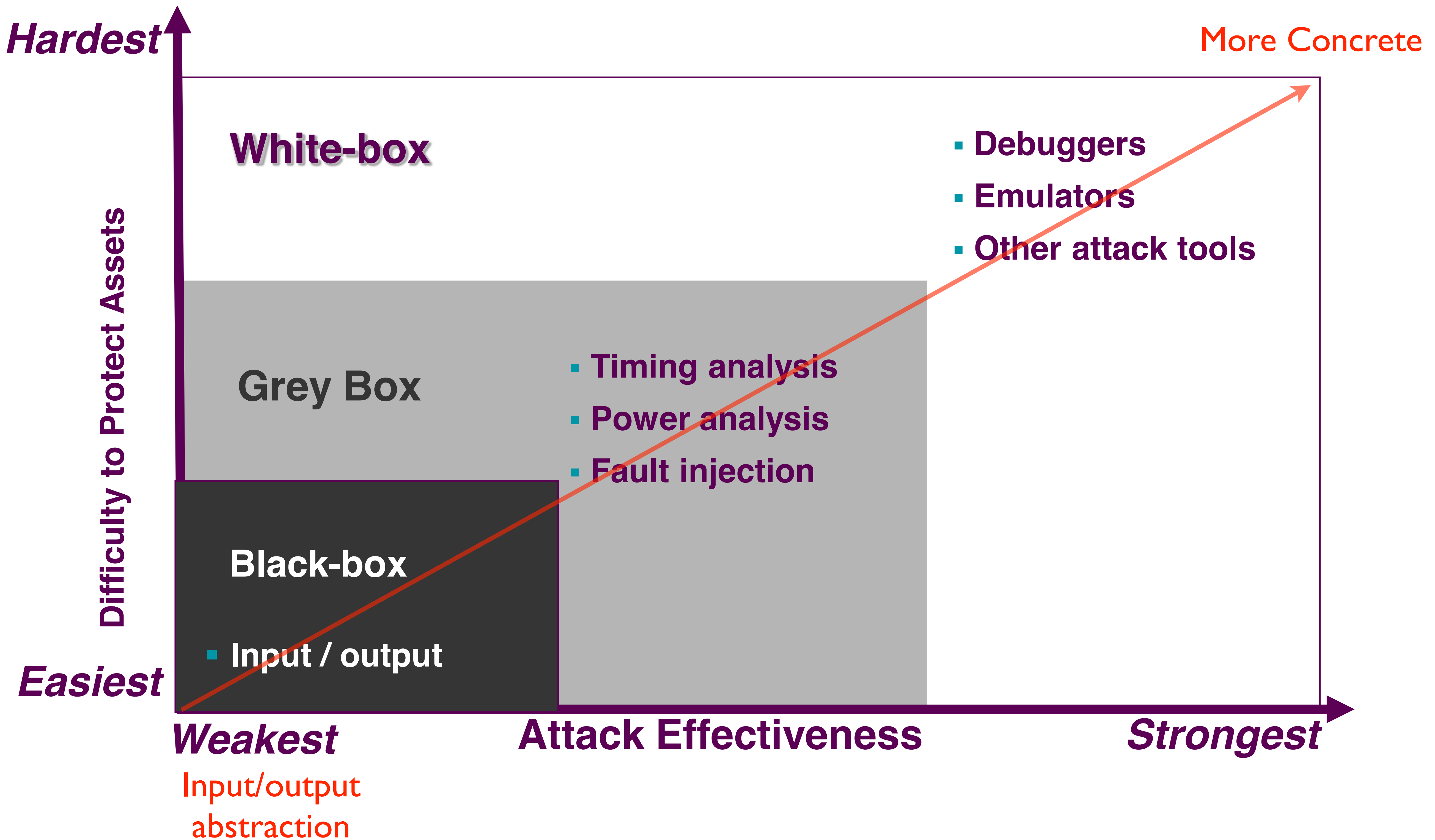
- ✓ Stated in another way, this pattern is to transform a key such that *attacking within the white box context offers no advantage to attacking in the black-box context*

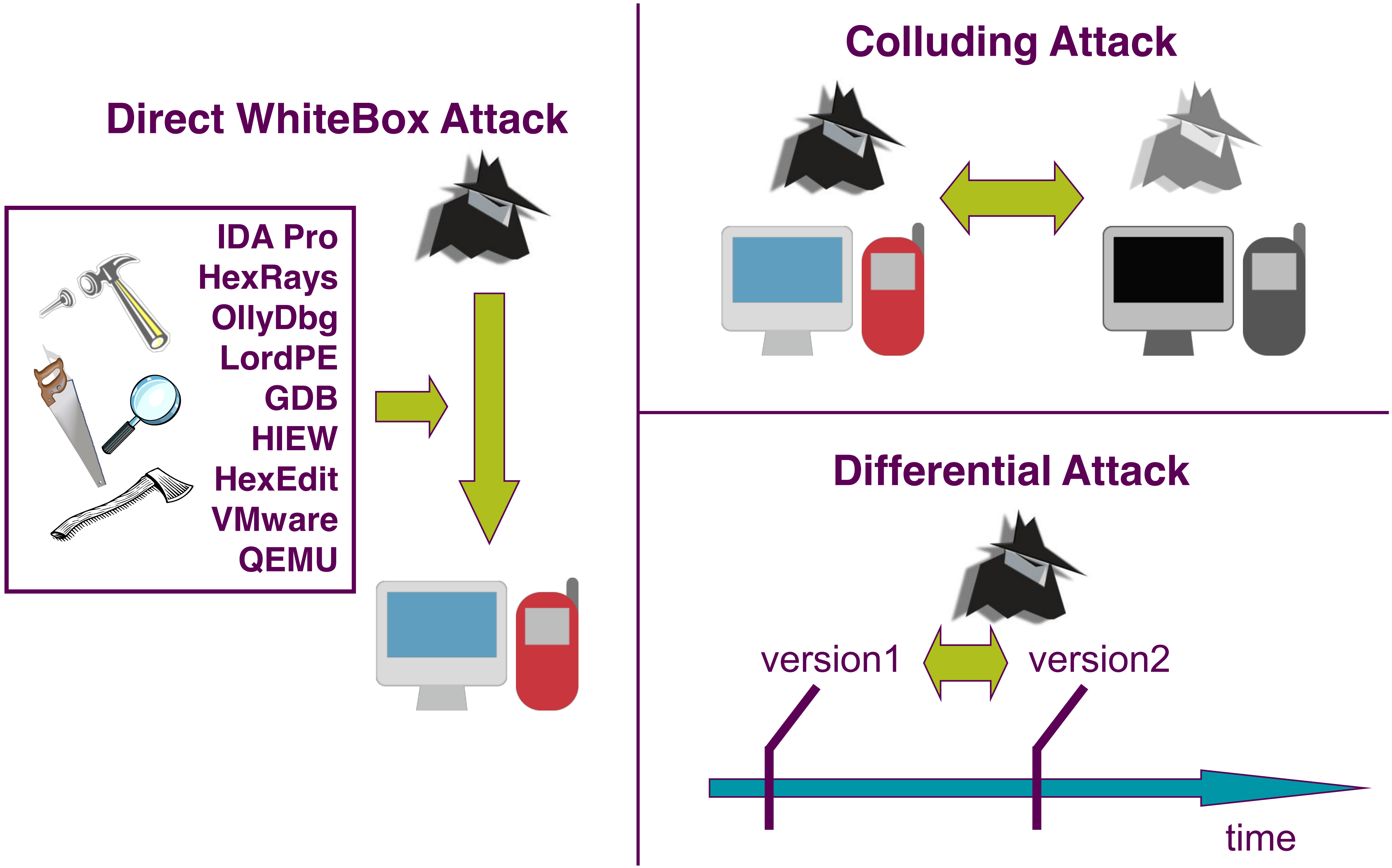
Code Obfuscation

The goal of program obfuscation is to make a program *unintelligible* while preserving its functionality. Ideally, an obfuscated program should be a *virtual black box*, in the sense that anything one can compute from it, one could also compute from the input-output behavior of the program

Rice Theorem, the hardness of the halting problem seem to imply that the only useful thing that one can do with a program or circuit is to *run it*

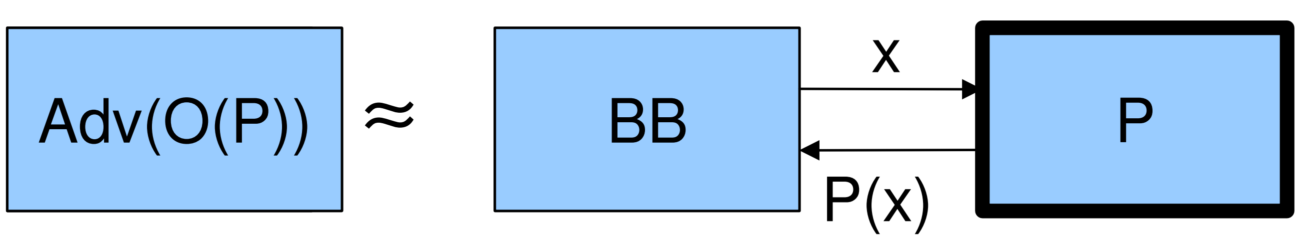
Hostile Environment




Direct WhiteBox Attack
IDA Pro
HexRays
OllyDbg
LordPE
GDB
HIEW
HexEdit
VMware
QEMU
Colluding Attack
Differential Attack
version1
version2
time

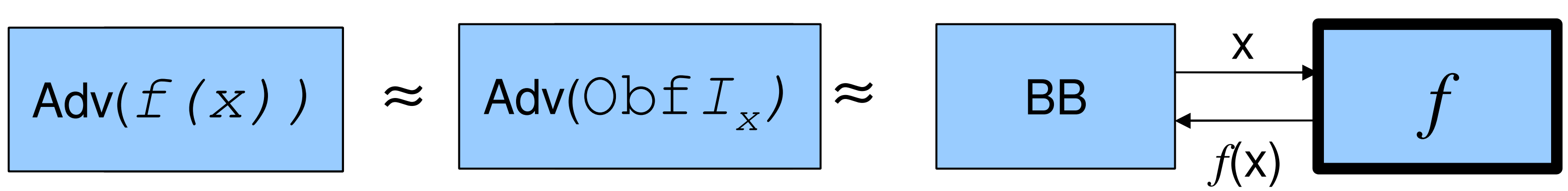
What is an obfuscation?

- ✓ An *obfuscator* is an algorithm **O** such that for any program **P**, **O(P)** is a program such that
  - **O(P)** has the *same functionality* as **P**
  - **O(P)** is *infeasible to analyse* / reverse engineer
- ✓ *Intuition*: an obfuscation should provide a *virtual black box* in the sense that giving someone **O(P)** should be equivalent to giving to the attacker a black box that computes **P**



Obfuscating Point Functions

- ✓ Point function
  - $\mathbf{I_x(w) = \{1 \text{ if } w = x, 0 \text{ otherwise}\}}$
- ✓ Obfuscate $\mathbf{I_x}$ with perfectly one-way function **f**.
- ✓ A one-way function is a function easy to compute on every input, but hard to invert given the image of a random input (fundamental in cryptography)
- ✓ Let $\mathbf{y = f(x)}$
  - $\mathbf{ObfI_x(w) = \{if\ y = f(w)\ return\ 1\ else\ return\ 0\}}$
- ✓ Intuitively, $\mathbf{y = f(x)}$ reveals no more info than black-box access to $\mathbf{I_x}$



Obfuscating Point Functions

- *Point function*
  - `I`$_x$`(w) = {1 if w = x,0 otherwise}`
- Obfuscate `I`$_x$ with perfectly *one-way function* **`f`**.
- Ler `y = f(x)`
  - `ObfI`$_x$`(w)= {if y = f(w) return 1 else return 0}`
- This is a very specific obfuscator…

Is it possible to define a general obfuscation for all programs?

Obfuscation

Functionality

```
n := n0;
i := n;
while (i <> 0 ) do
      j := 0;
      while (j <> i) do
            j := j + 1
      od;
      i := i - 1
od
```

On the impossibility of code obfuscation. Barak et al JACM 2012

Obfuscation

Functionality

```
n := n0;
i := n;
while (i <> 0 ) do
    j := 0;
    while (j <> i) do
        j := j + 1
    od;
    i := i - 1
od
```

On the impossibility of code obfuscation. Barak et al JACM 2012

Obfuscation

Polynomial Slowdown

```
n := n0;
i := n;
while (i <> 0 ) do
      j := 0;
      while (j <> i) do
            j := j + 1
      od;
      i := i - 1
od
```

On the impossibility of code obfuscation. Barak et al JACM 2012

Obfuscation

Polynomial Slowdown

```
n := n0;
i := n;
while (i <> 0 ) do
    j := 0;
    while (j <> i) do
        j := j + 1
    od;
    i := i - 1
od
```

On the impossibility of code obfuscation. Barak et al JACM 2012

Obfuscation

Virtual black-box

```
n := n0;
i := n;
while (i <> 0 ) do
    j := 0;
    while (j <> i) do
        j := j + 1
    od;
    i := i - 1
od
```

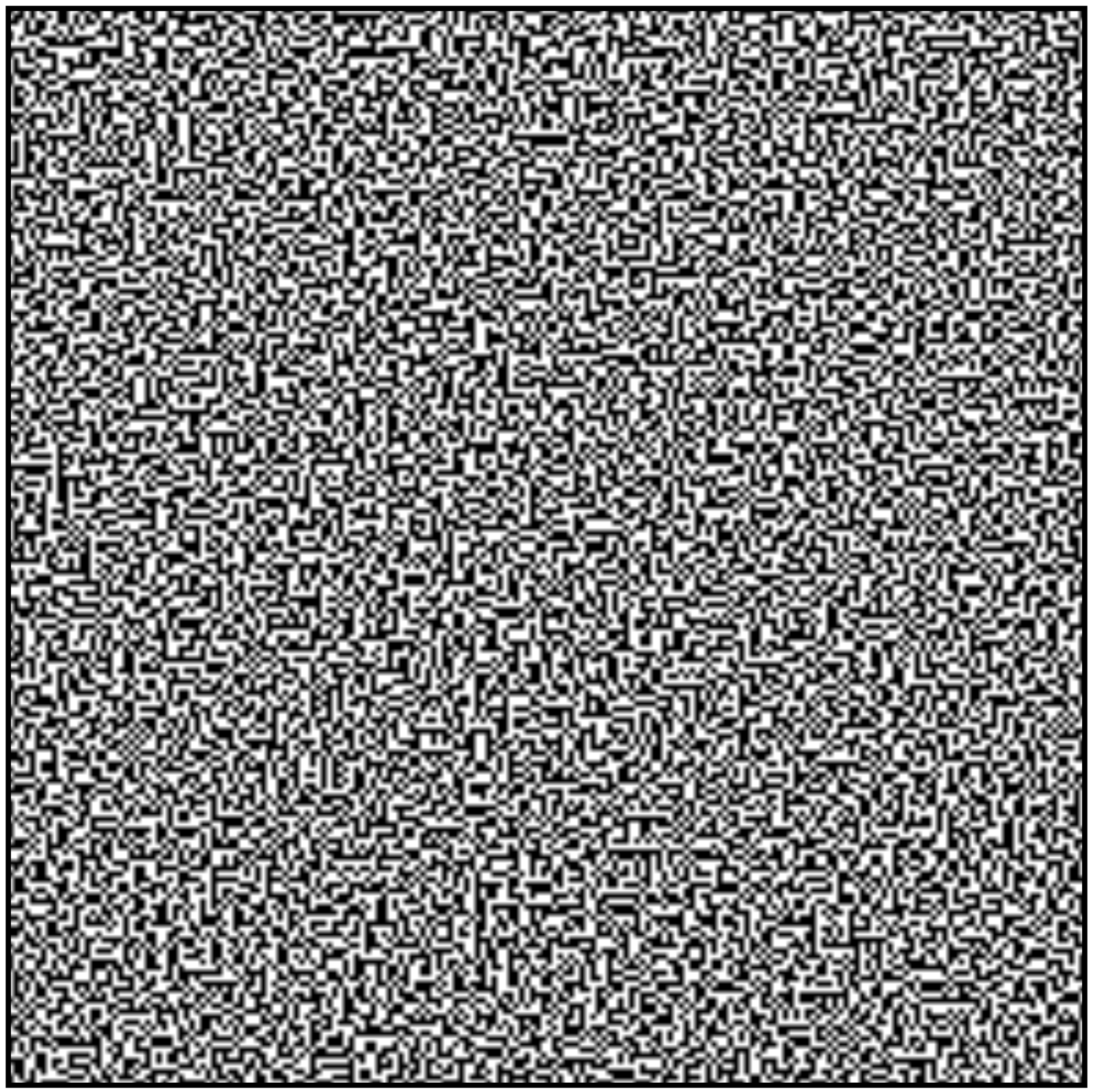
*“Anything that can be learned from the obfuscated form, could have been learned by merely observing the program’s input-output behavior (i.e., by treating the program as a **black-box**)”*

$$|Pr[A(\mathcal{O}(M)) = 1] - Pr[S^M(1^{|M|}) = 1]| \leq \varepsilon(|M|)$$

On the impossibility of code obfuscation. Barak et al JACM 2012

Obfuscation

Virtual black-box



*"Anything that can be learned from the obfuscated form, could have been learned by merely observing the program's input-output behavior (i.e., by treating the program as a **black-box**)"*

$$|Pr[A(\mathcal{O}(M)) = 1] - Pr[S^M(1^{|M|}) = 1]| \leq \varepsilon(|M|)$$

On the impossibility of code obfuscation. Barak et al JACM 2012

Obfuscation

Impossible!

- ~~Functionality~~
- ~~Polynomial Slowdown~~
- ~~Virtual Black-Box~~

*"Anything that can be learned from the obfuscated form, could have been learned by merely observing the program's input-output behavior (i.e., by treating the program as a **black-box**)"*

On the impossibility of code obfuscation. Barak et al JACM 2012

Defining Obfuscators

- ✓ Anything that an adversary can compute from the obfuscation **O(P)** of **P**, it could also compute given just oracle access to **P**
- ✓ More formally we consider an adversary that is trying to decide some predetermined property of the original program
- ✓ We consider programs as represented by Turing Machines
- ✓ While adversaries are modeled as probabilistic polynomial time Turing machines PPT

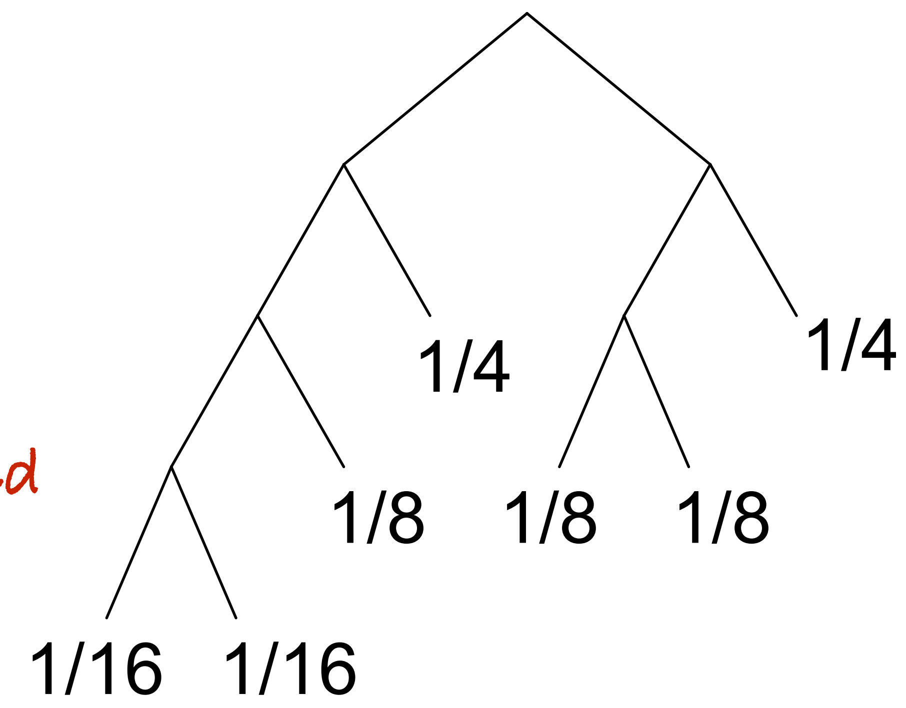
Probabilistic Polynomial Time TM

✓NTM where each nondeterministic step is a coin flip: has exactly 2 next moves and we assign to each move a probability 1/2

✓example: to each maximal branch we assign a probability

✓**PPT** TM has accept and reject states

✓We can speak about probability of acceptance and rejection on an input **w**



Computation on input **w**

Probabilistic Polynomial Time TM

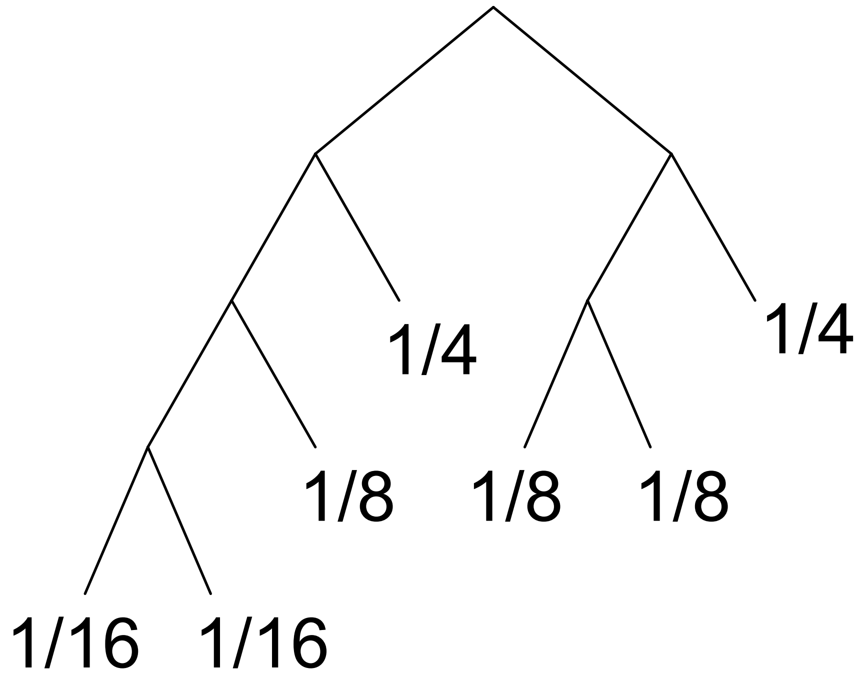
- ✓*Probability of acceptance:*

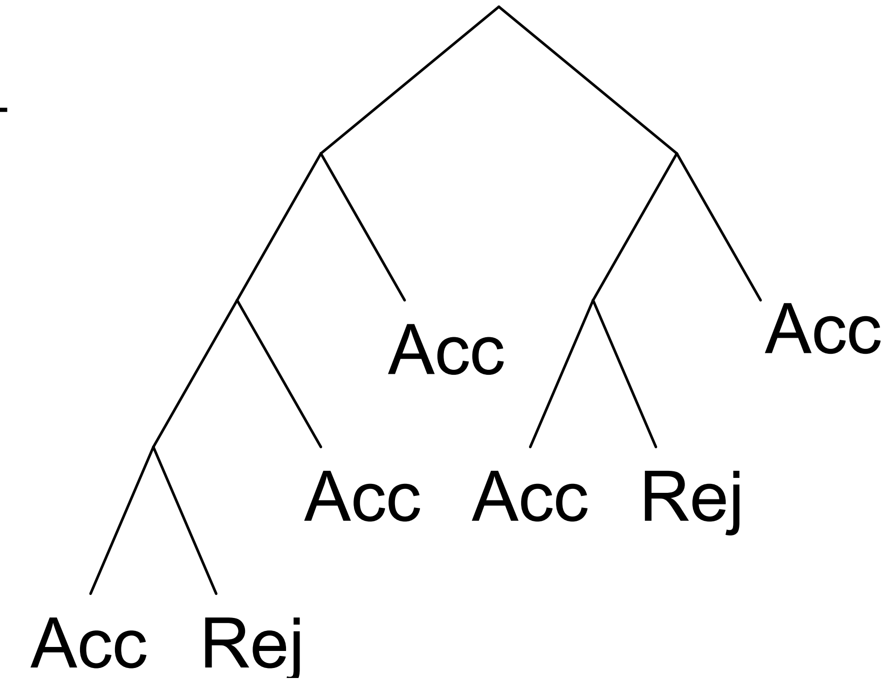
  $\Sigma_{\text{b an accepting branch}} \Pr(b)$

- ✓*Probability of rejection:*

  $\Sigma_{\text{b a rejecting branch}} \Pr(b)$

- ✓example: add accept/reject information.
  - ✓Probability of acceptance = 1/16 + 1/8 + 1/4 + 1/8 + 1/4 = 13/16
  - ✓Probability of rejection = 1/16 + 1/8 = 3 /16
- ✓We consider TMs that halt (accept/reject) on every branch, so that the total probability is 1





Defining Obfuscators

- ✓ <u>Definition of TM Obfuscator</u>:
- ✓ A probabilistic algorithm **O** is a TM-obfuscator if for any TM **M** the following conditions hold:
  - *functionality*: **O(M)** describes aTM that computes the same function as **M, O(M) ~ M**,
  - *polynomial slowdown*: the description length and running time of **O(M)** are at most polynomially larger than that of M: **IO(M)I<= p(IMI)** for some polynomial **p,** and if **M** halts in **t** steps on some input **x**, then **O(M)** halts within **p(t)** steps on input **x**
  - *virtual black box property:* for any PPT **A** there is a PPT **S** and a negligible function α such that for all TMs **M**:

$$\left| Pr[A(O(M)) = 1] - Pr[S^M(1^{|M|}) = 1] \right| \leq \alpha(|M|)$$

Oracle access

On the impossibility of code obfuscation. Barak et al JACM 2012

2-TM obfuscator

✓A 2-TM obfuscator is defined in the same way as a TM-obfusctor, except that the virtual black box property is changed as follows:

✓virtual black box: for any PPT **A** there is a PPT **S** and a negligible function α such that for all TMs **M** and **N**:

$$\left| Pr[A(O(M), O(N)) = 1] - Pr[S^{M,N}(1^{|M|+|N|}) = 1] \right| \leq \alpha(min(|M|, |N|))$$

✓The virtual black box property holds also when the adversary can inspect more than one TMs

✓The proof of impossibility of 2-TM obfuscator provides useful motivations to the proof of impossibility of TM-obfuscator

Impossibility of 2-TM obfuscation

- ✓ The essence of this proof is that *there is a fundamental difference between getting black-box access to a function and getting a program that computes it, no matter how obfuscated it is*
- ✓ Of course if the function is exactly learnable via oracle queries then this difference disappears
- ✓ Hence in our proofs we consider functions that cannot be exactly learned with oracle queries

Impossibility of 2-TM obfuscation

$\alpha, \beta \in \{0,1\}^k$ Secret

$$C_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ 0 & \text{otherwise} \end{cases}$$

$$D_{\alpha,\beta}(C) = \begin{cases} 1 & \text{if } C(\alpha) = \beta \\ 0 & \text{otherwise} \end{cases}$$

Distinguish if C computes $C_{\alpha,\beta}$ from $C_{\alpha',\beta'}$ for any $(\alpha, \beta) \neq (\alpha', \beta')$
NON COMPUTABLE!

$\Rightarrow$ Simply compute C(α) for poly(k) steps and check!

Impossibility of 2-TM obfuscation

$\alpha, \beta \in \{0,1\}^k$ Secret

$$C_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ 0 & \text{otherwise} \end{cases}$$

$$D_{\alpha,\beta}(C) = \begin{cases} 1 & \text{if } C(\alpha) = \beta \\ 0 & \text{otherwise} \end{cases}$$

Distinguish if C computes $C_{\alpha,\beta}$ from $C_{\alpha',\beta'}$ for any $(\alpha,\beta)\neq(\alpha',\beta')$
NON COMPUTABLE!

Simply compute C(α) for poly(k) steps and check!

Consider an adversary **A** that given two TMs runs the second on the first one

$$Pr[A(\mathcal{O}(C_{\alpha,\beta}), \mathcal{O}(D_{\alpha,\beta})) = 1] = 1$$

On the impossibility of code obfuscation. Barak et al JACM 2012

Impossibility of 2-TM obfuscation

✓Observe that PPT **S** that has oracle access to **C** and **D** has only exponentially small probability of querying either oracle at a point where its value is not zero

✓Hence consider the TM **Z** that always outputs 0

$$Z_k(x) = 0^k$$

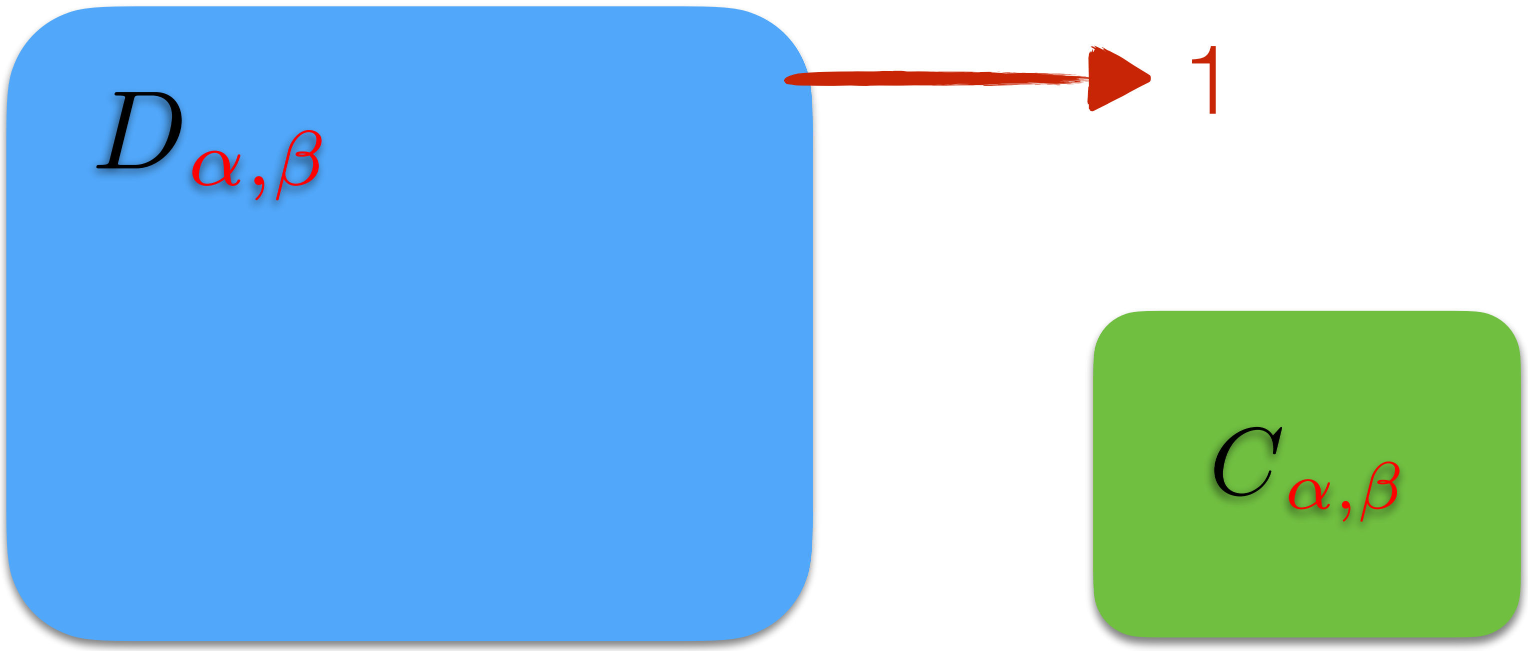
✓Then for every PPT **S** it holds that

$$\left|\Pr\left[S^{C_{\alpha,\beta},D_{\alpha,\beta}}(1^k)=1\right]-\Pr\left[S^{Z_k,D_{\alpha,\beta}}(1^k)=1\right]\right| \leq \underbrace{2^{-\Omega(k)}}_{\text{negligible}}$$

✓Moreover, by definition of **A** we have that:

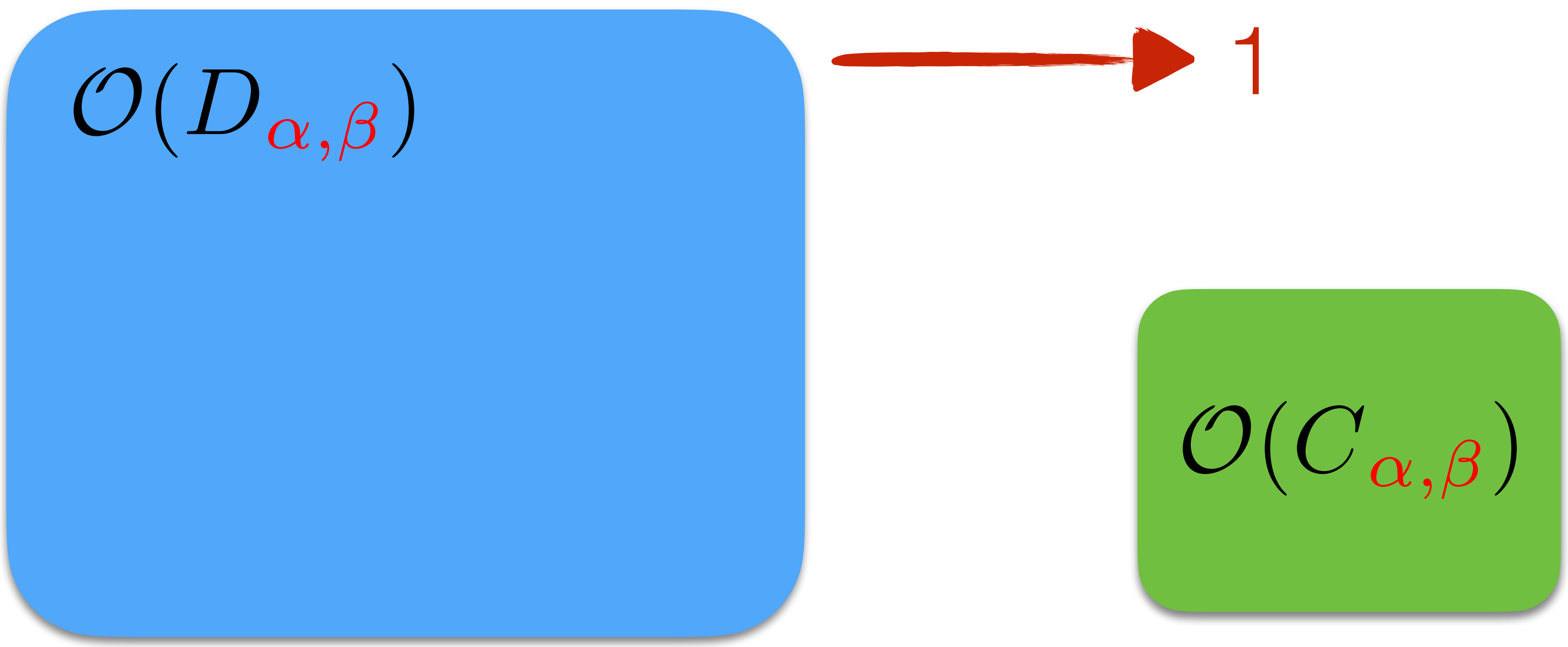
$$\Pr\left[A(\mathcal{O}(Z_k),\mathcal{O}(D_{\alpha,\beta}))=1\right] = \underbrace{2^{-k}}_{\text{negligible}}$$

Impossibility of 2-TM obfuscation



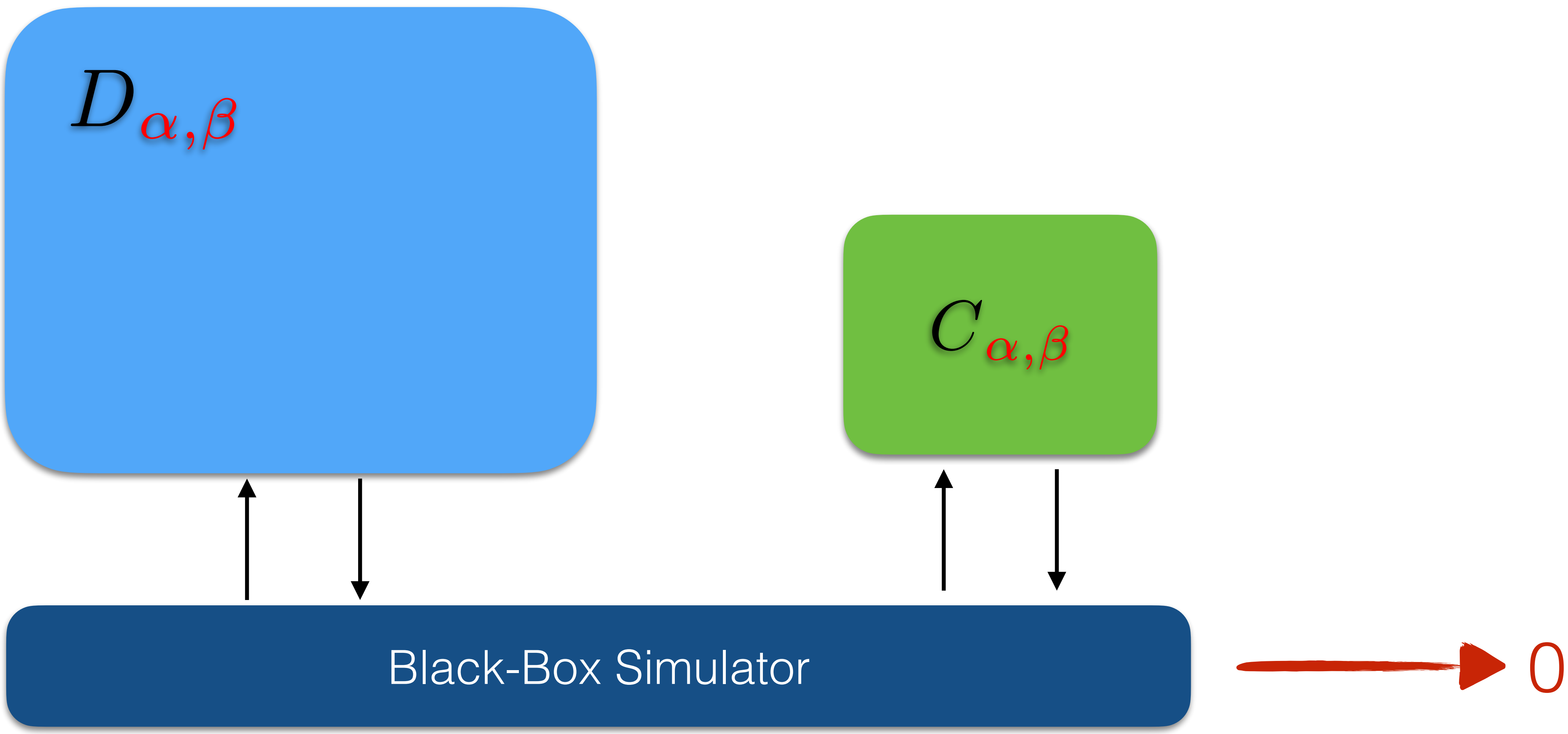
The Functionality preserves behavior

Impossibility of 2-TM obfuscation



The Functionality
preserves behavior
even if obfuscated

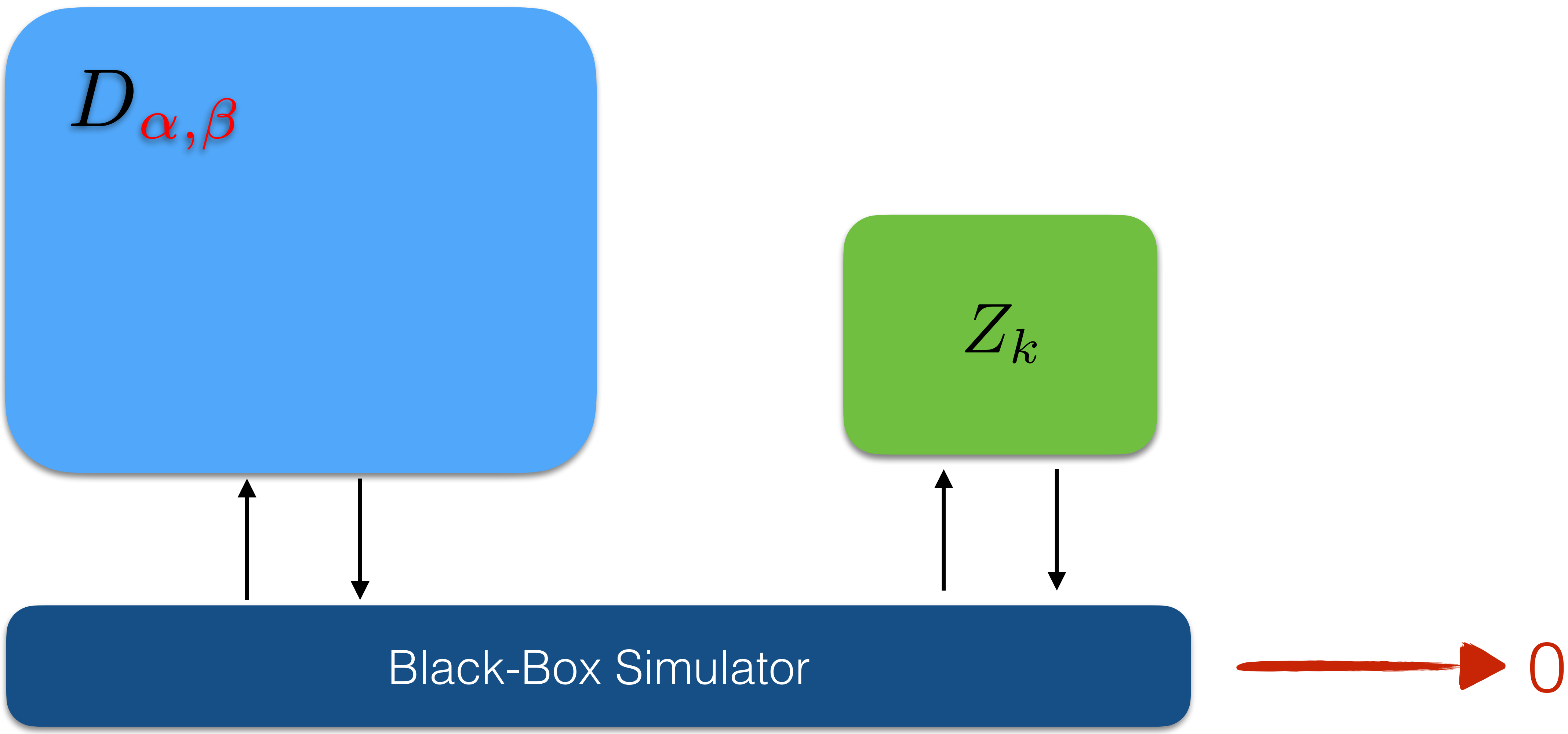
Impossibility of 2-TM obfuscation



Virtual Black-Box

On the impossibility of code obfuscation. Barak et al JACM 2012

Impossibility of 2-TM obfuscation



Virtual Black-Box

On the impossibility of code obfuscation. Barak et al JACM 2012

Proof of 2-TM obfuscation

Consider an **adversary**: $A(C_{\alpha,\beta}, D_{\alpha,\beta}) = D_{\alpha,\beta}(C_{\alpha,\beta})$

$$Pr[A(\mathcal{O}(C_{\alpha,\beta}), \mathcal{O}(D_{\alpha,\beta})) = 1] = 1 \qquad (1)$$

Exponentially small probability for random (α, β) of having non zero behaviour!!

For any PPT *S*

0

$$\left|\Pr\left[S^{C_{\alpha,\beta},D_{\alpha,\beta}}(1^k) = 1\right] - \Pr\left[S^{Z_k,D_{\alpha,\beta}}(1^k) = 1\right]\right| \leq 2^{-\Omega(k)} \qquad (2)$$

0

||

By definition of *A*: $\Pr\left[A(\mathcal{O}(Z_k), \mathcal{O}(D_{\alpha,\beta})) = 1\right] = 2^{-k}$ (3)

Because obfuscated code is functionally equivalent to non obfuscated code!

1+2+3 contradicts Virtual black-box assumption!

Proof of 2-TM obfuscation

$$Pr[A(\mathcal{O}(C_{\alpha,\beta}),\mathcal{O}(D_{\alpha,\beta}))=1]=1 \qquad (1)$$

DIFFERENT!!!

0

$$\left|\Pr\left[S^{C_{\alpha,\beta},D_{\alpha,\beta}}(1^k)=1\right]-\Pr\left[S^{Z_k,D_{\alpha,\beta}}(1^k)=1\right]\right|\leq 2^{-\Omega(k)} \qquad (2)$$

0

||

$$\Pr\left[A(\mathcal{O}(Z_k),\mathcal{O}(D_{\alpha,\beta}))=1\right]=2^{-k} \qquad (3)$$

Impossibility of TM obfuscator

TM obfuscatory do not exist!

Consider the following TMs obtained by combining the TMs of the previous proof:

$$F_{\alpha,\beta}(b,x) \;=\; C_{\alpha,\beta}\#D_{\alpha,\beta}$$

$$G_{\alpha,\beta}(b,x) \;=\; Z_k\#D_{\alpha,\beta}$$

We consider an adversary analogous to the previous one, namely an adversary **A** that given a TM in input decomposes it into two parts and evaluates the second part on the code (encoding) of the first

Impossibility

$$A(F) = \text{decompose } F \mapsto F_0\#F_1;\ F_1(F_0)$$

Proof sketch:

- Assume $\mathcal{O}$ is a TM obfuscator, and let $C_{a,b}$, $D_{a,b}$, and $Z_{a,b}$ be the TMs defined before.
- Define TMs $F_{a,b} = C_{a,b}\#D_{a,b}$ and $G_{a,b} = Z_k\#C_{a,b}$.
- On input a TM F, adversary A first decomposes F into $F_0\#F_1$ and then outputs $F_1(F_0)$.

Impossibility

As in the previous proof, we have:

- $\Pr[A(\mathcal{O}(F_{a,b})) = 1] = 1$
- $\Pr[A(\mathcal{O}(G_{a,b})) = 1] = 0$
- $|\Pr[S^{F_{a,b}}(1^k) = 1] - \Pr[S^{G_{a,b}}(1^k) = 1]| \leq 2^{-\Omega(k)}$

where the probability is taken over a,b, and the the coin tosses of A,S, and $\mathcal{O}$, which contradicts the assumption.

On the impossibility result

✓The paper showed that the virtual black box paradigm for obfuscation is inherently flowed

✓There may still be methods to make program intelligible in a meaningful and precise sense

✓The proof relies on the construction of unnatural function families

Does a more intuitive proof exists?

On the impossibility result

✓ Many researchers have tried to understand the implications and limits if the impossibility result of Barak et al.

✓ Since the impossibility result of Barak et al. many techniques for obfuscating particular properties of particular programs have been introduced

✓ In practice software protection is interested in a weaker notion of obfuscation (make it harder for a certain time, side effect, slowdown,…)

On the impossibility of code obfuscation. Barak et al JACM 2012

Indistinguishability obfuscator

- ✓Weaker notion of obfuscator introduced in Barak et al. 2001
- ✓An *indistinguishability obfuscator* is defined in the same way as an obfuscation, except that the virtual black box property is replaced with the following:
  - ✓For any PPT **A** there is a negligible function α such that, for any two circuits **C1** and **C2** that compute the same function and are of the same size **k**, it holds that

$$|\Pr[\ A(O(C1))\ ] - \Pr[\ A(O(C2))]\ | \le \alpha\ (\ k\ )$$

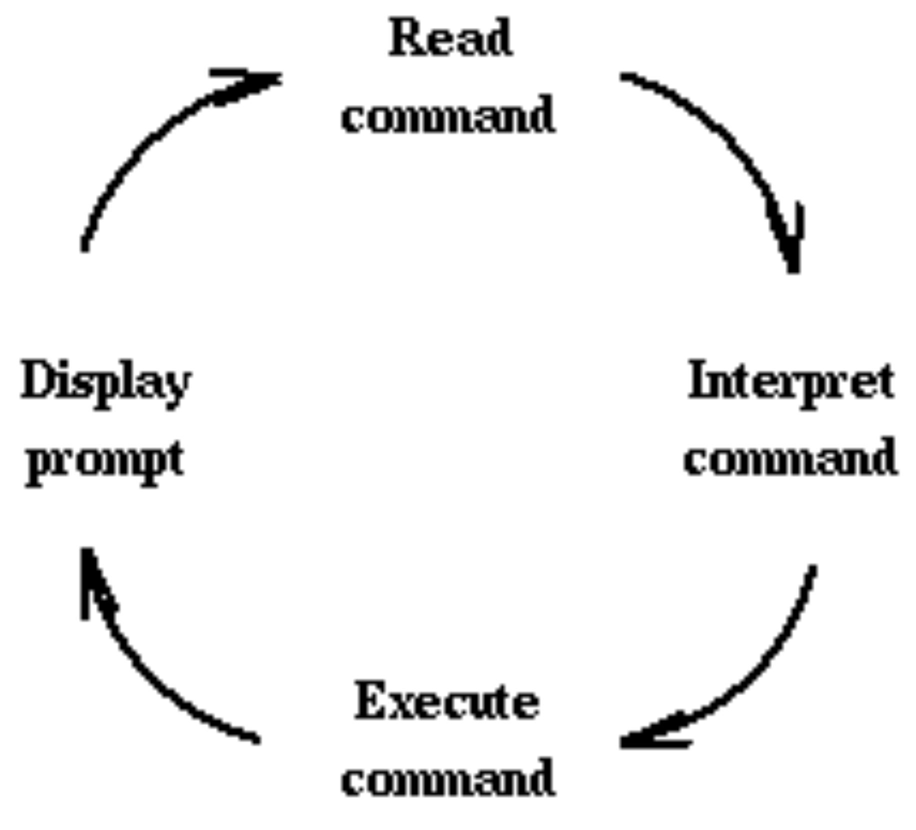
Indistinguishability obfuscator

- ✓We note that if the circuit class **C** has efficiently computable canonical forms, then the computation of that canonical form would already be an indistinguishability obfuscator
- ✓Do there exists indistinguishability obfuscatory for all polynomial-size circuits?
- ✓Breakthrough: Amit Sahai FOCS 2013, Candidate indistinguishability obfuscation and functional encryption for all circuits
- ✓Still impractical!!

On the impossibility result

Whenever we disclose the code we always disclose more that its input/output relation!!

The notion of <u>interpretation</u> is fundamental here!



On the impossibility result

Whenever we disclose the code we always disclose more that its input/output relation!!

CLASSES OF RECURSIVELY ENUMERABLE SETS AND THEIR DECISION PROBLEMS(1)

BY

H. G. RICE

1952

On the (Im)possibility of Obfuscating Programs*

Boaz Barak† Oded Goldreich† Russell Impagliazzo‡ Steven Rudich§ Amit Sahai¶ Salil Vadhan|| Ke Yang§

2001

How to verify programs?

1970

Formal Methods
Abstract Interpretation
Program Analysis
Verification

How to protect programs?

2001

?