On the (Im)possibility of Obfuscating Programs

A discussion of the 2001 paper by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang.

Presented by Nicolai Moles-Benfell.

CS725

Summary of results

- Barak *et al* 2001 paper is a theoretical proof of the impossibility of a perfect obfuscator capable of obfuscating all Turing machines (programs).

- Their proof relies
  - On the existence of one way functions.
  - And their model and definition of an obfuscator.

What is Obfuscation?

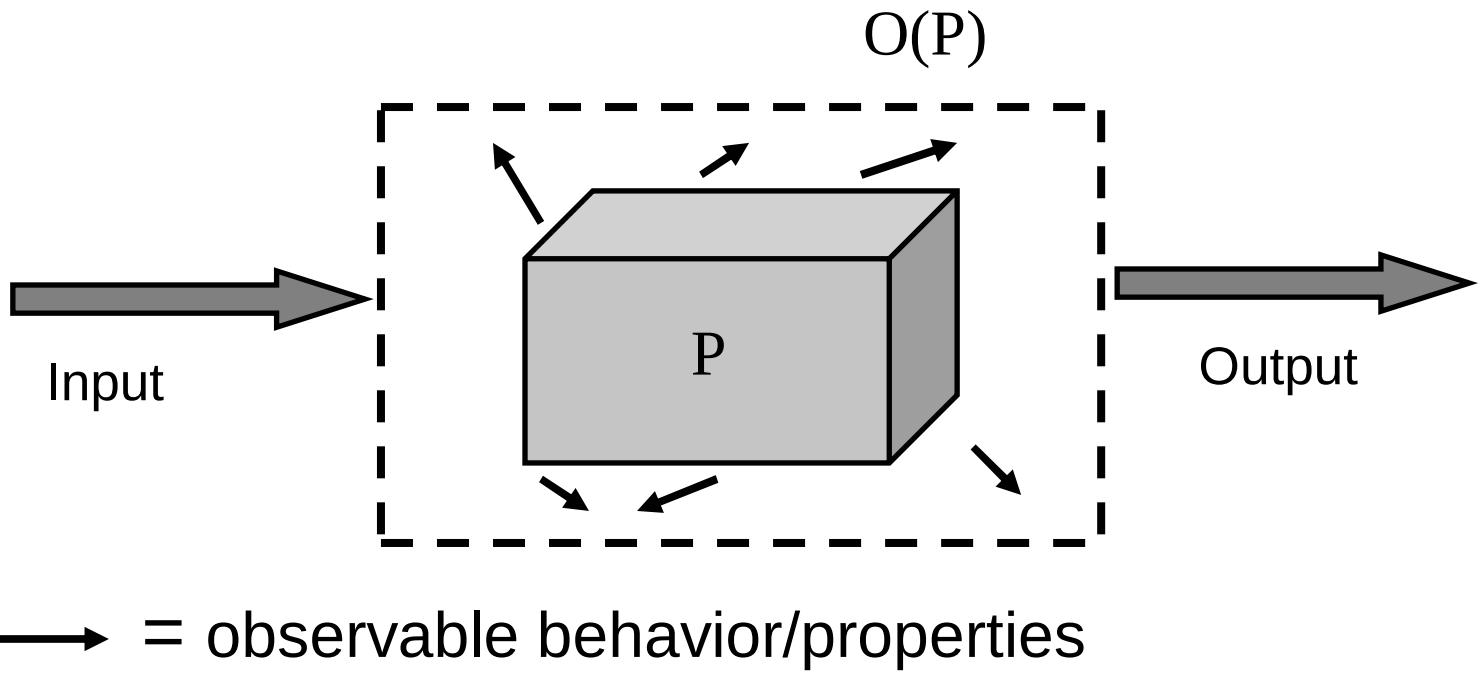
- An obfuscator is an algorithm that can be viewed as a compiler, that takes a program P as input and ouput's a different program O(P).

- Generally the goal of O(P) is to make understanding P difficult and/or time consuming.

- Barak *et al* chose a stricter definition of obfuscation:
  - O(P) should behave as a black box.
    - No information about P should be learnable from O(P), except that which is observable from studying the Input/Output of O(P).

  - O(P) should have approximately the same functionality and time complexity (running time) as P.

What is a Black Box?

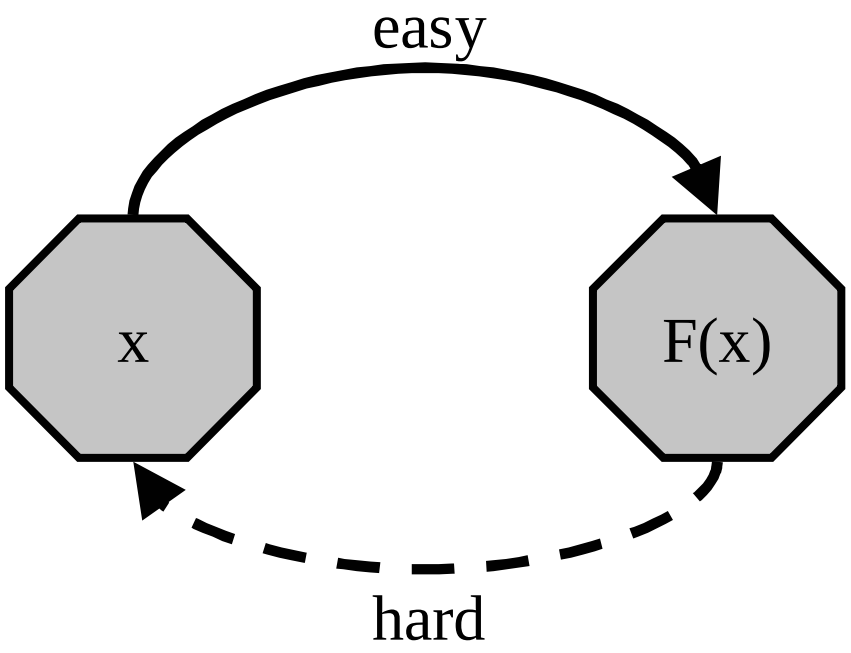


- Programs have observable properties, some are important others are not, a black box obfuscator would hides all properties and behavior.
- By adopting the black box model it is then possible to prove the absolute security of a obfuscation scheme.

One way functions

- One way functions are easy to compute, but intractable to reverse.
- Example (Integer factorization):
  - It is computationally easy to multiple any two numbers, 19 x 47 = 893
  - but if these numbers are (large) primes, then factorizing them is computationally difficult (no efficient algorithm is believed to exist)

    893 = ? x ?



- Unless the party computing the inverse of the problem has additional information specific to the problem, a ‘trap door’.
- A trap door allows the inverse of the problem to be easily computed.

Barak *et al* Main Results

- Outline of their proof:
  - They show that if one-way functions exist then there exists programs that can not be obfuscated.
- Why ?
  - Let P compute the inverse of a one-way function, with the use of a trap door.
  - Then the trap door (a hard coded value) is a distinctive property of P, that the obfuscator would need to remove from O(P).
  - But by removing it,
    - O(P) is no longer able to efficiently compute the inverse of the problem
    - therefore does not have the same functionality as P, and fails to meet Barak's definition of an obfuscator.
- Therefore universal (black box) obfuscators can not exist.

Appreciative Comment?

- Barak *et al* describe a number of interesting applications of a black box obfuscator (if it were to exist).

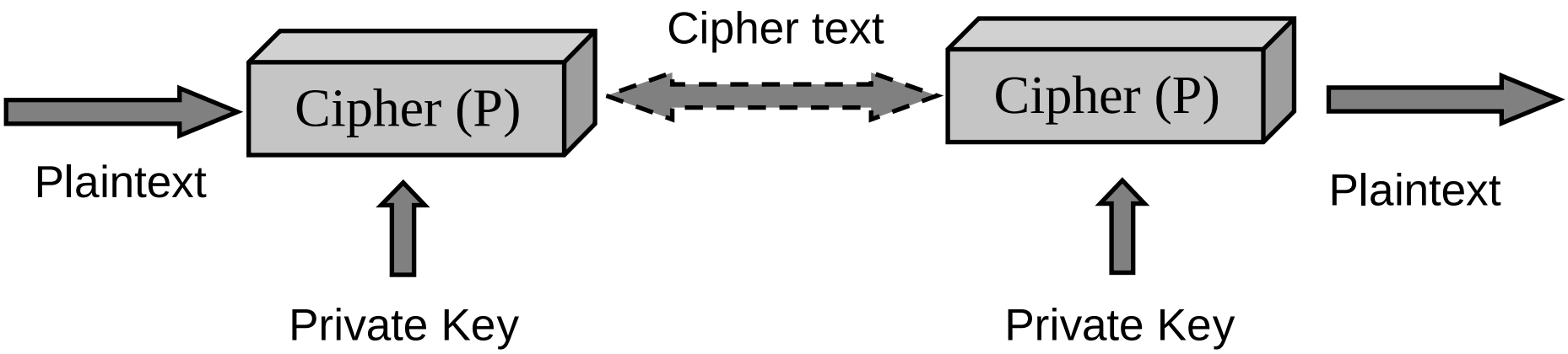
- For Example:
  - Transforming Private key encryption into Public key encryption (secure Digital Rights management)

Private Key Cryptography

- Private Key cryptography is computationally inexpensive

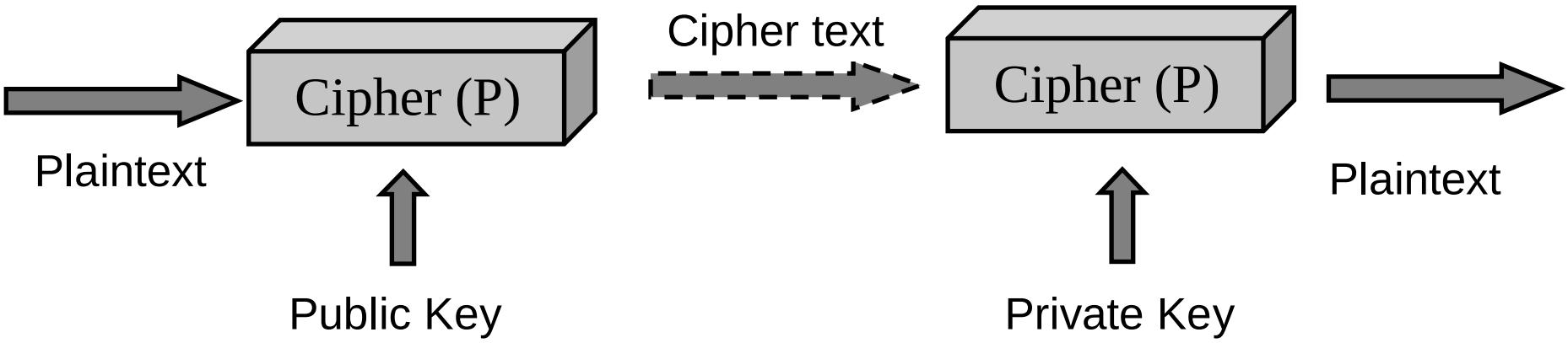
-Security (should be) dependent on the secrecy of the key.

- **But** need a method to securely share the key with other parties.



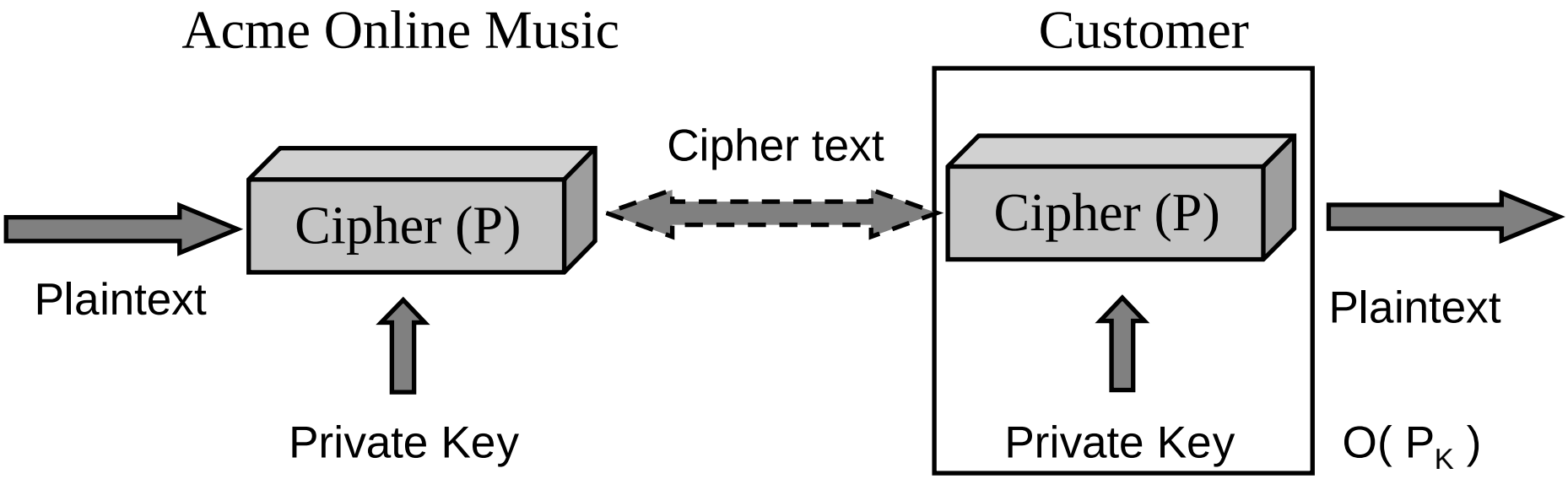
Public Key Cryptography

- Public Key cryptography is computational expensive, and employees one-way functions.
- And uses two keys, one public and the other private.
- The public key is published so anyone can encrypt data, but only the trapdoor (the secret key) can be used to decrypt.



Digital Rights management

- If a Black Box Obfuscator existed, then hardwire the secret key into decryption algorithm of a *private key cipher*, and obfuscate the two, to create $O(P_K)$
- Why?
  - Because By encrypting the digital media with a customers private key, only the obfuscated program $O(P_K)$ can be used to view the media.



Critical Thoughts

- On face value these results appear disappointing for software obfuscation.
- But they arise from defining and modeling obfuscation as a black box.
- The impossibility of a universal black box obfuscator limits some theoretical applications.
- But in practice several more important questions araise
  - What proportion (or class) of programs do the results apply to?
  - Is there a more practical model of obfuscation then the Black box paradigm? For example that might allow non sensitive information to be leaked from the obfuscated code?

The other Appreciative Comment

- Barak *et al* point out that their approach to obfuscation is theoretical, there might exist heuristically strong methods for commercial use!

Questions