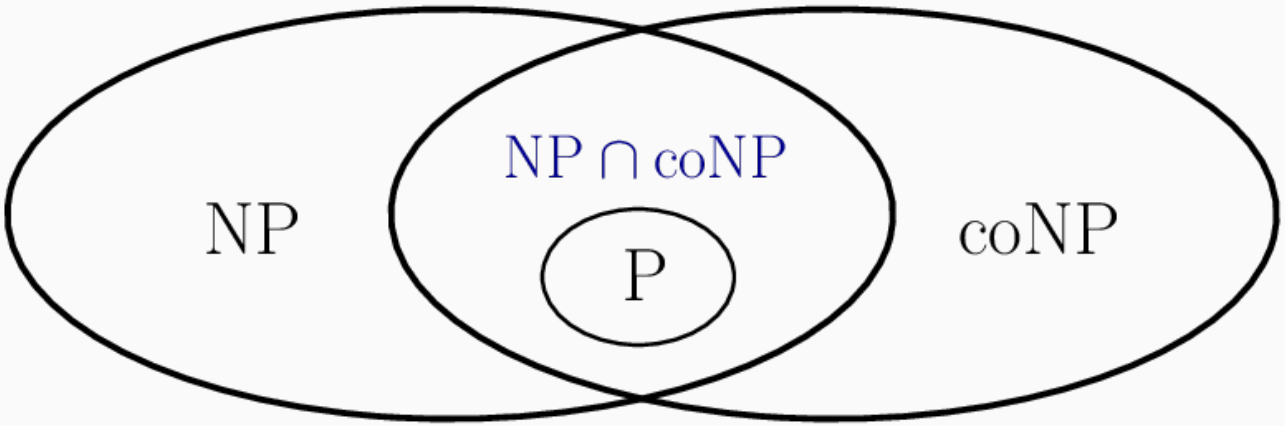
Complexity Theory

Lecture 9: Cryptography

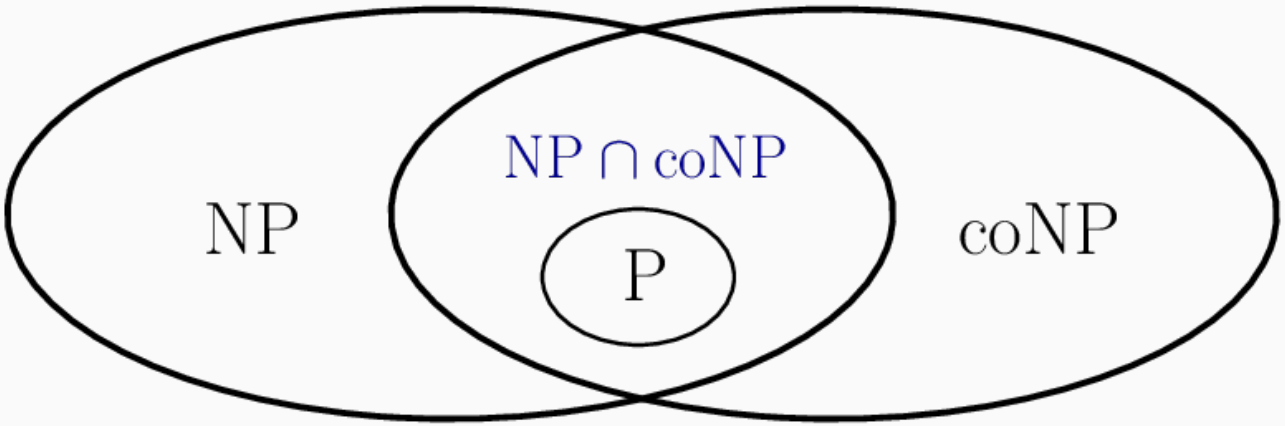
Tom Gur

http://www.cl.cam.ac.uk/teaching/2324/Complexity

Recap

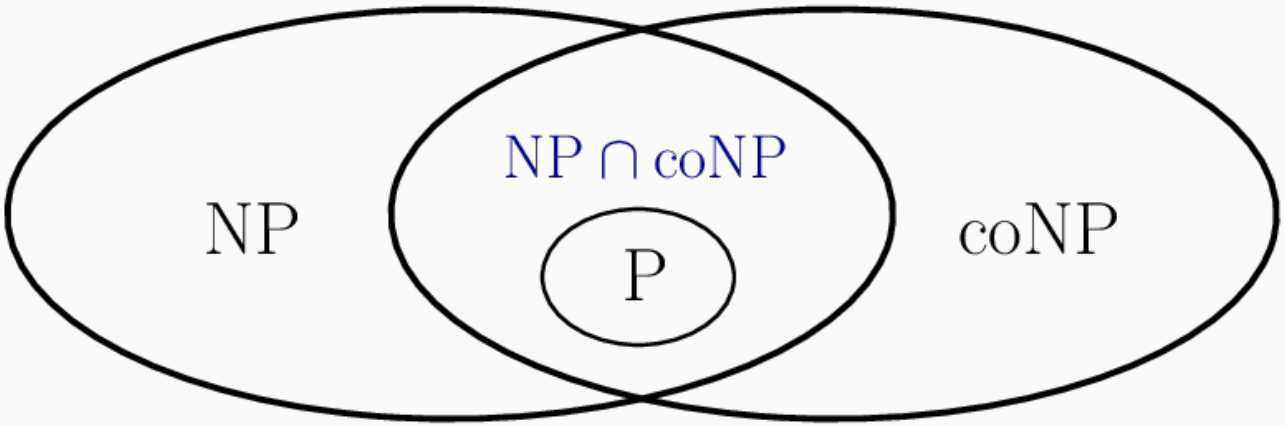


Recap



PRIME is in coNP (trivially), NP (non-trivially), and P (highly-non-trivially).

Recap



PRIME is in coNP (trivially), NP (non-trivially), and P (highly-non-trivially).

Cliffhanger: Why can't we break RSA using unary encodings?

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

What is the relation to the search version?

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

What is the relation to the search version?

In what complexity classes can we place Factor?

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

What is the relation to the search version?

In what complexity classes can we place Factor?

Factor $\in$ NP $\cap$ co-NP

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

What is the relation to the search version?

In what complexity classes can we place Factor?

Factor $\in$ NP $\cap$ co-NP

Certificate of membership—a factor of x less than k.

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

What is the relation to the search version?

In what complexity classes can we place Factor?

Factor $\in$ NP $\cap$ co-NP

Certificate of membership—a factor of x less than k.

Certificate of disqualification—the prime factorisation of x.

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

What is the relation to the search version?

In what complexity classes can we place Factor?

Factor $\in$ NP $\cap$ co-NP

Certificate of membership—a factor of x less than k.

Certificate of disqualification—the prime factorisation of x.

Factor $\in$ BQP

Graph Isomorphism

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, is there a *bijection*

$$\iota : V_1 \to V_2$$

such that for every $u, v \in V_1$,

$$(u, v) \in E_1 \quad \text{if, and only if,} \quad (\iota(u), \iota(v)) \in E_2.$$

Graph Isomorphism

Graph Isomorphism is

Graph Isomorphism

Graph Isomorphism is

- in NP

Graph Isomorphism

Graph Isomorphism is

- in NP
- not known to be in P

Graph Isomorphism

Graph Isomorphism is

- in NP
- not known to be in P
- not known to be in co-NP

Graph Isomorphism

Graph Isomorphism is

- in NP
- not known to be in P
- not known to be in co-NP
- not known (or *expected*) to be NP-complete

Graph Isomorphism

Graph Isomorphism is

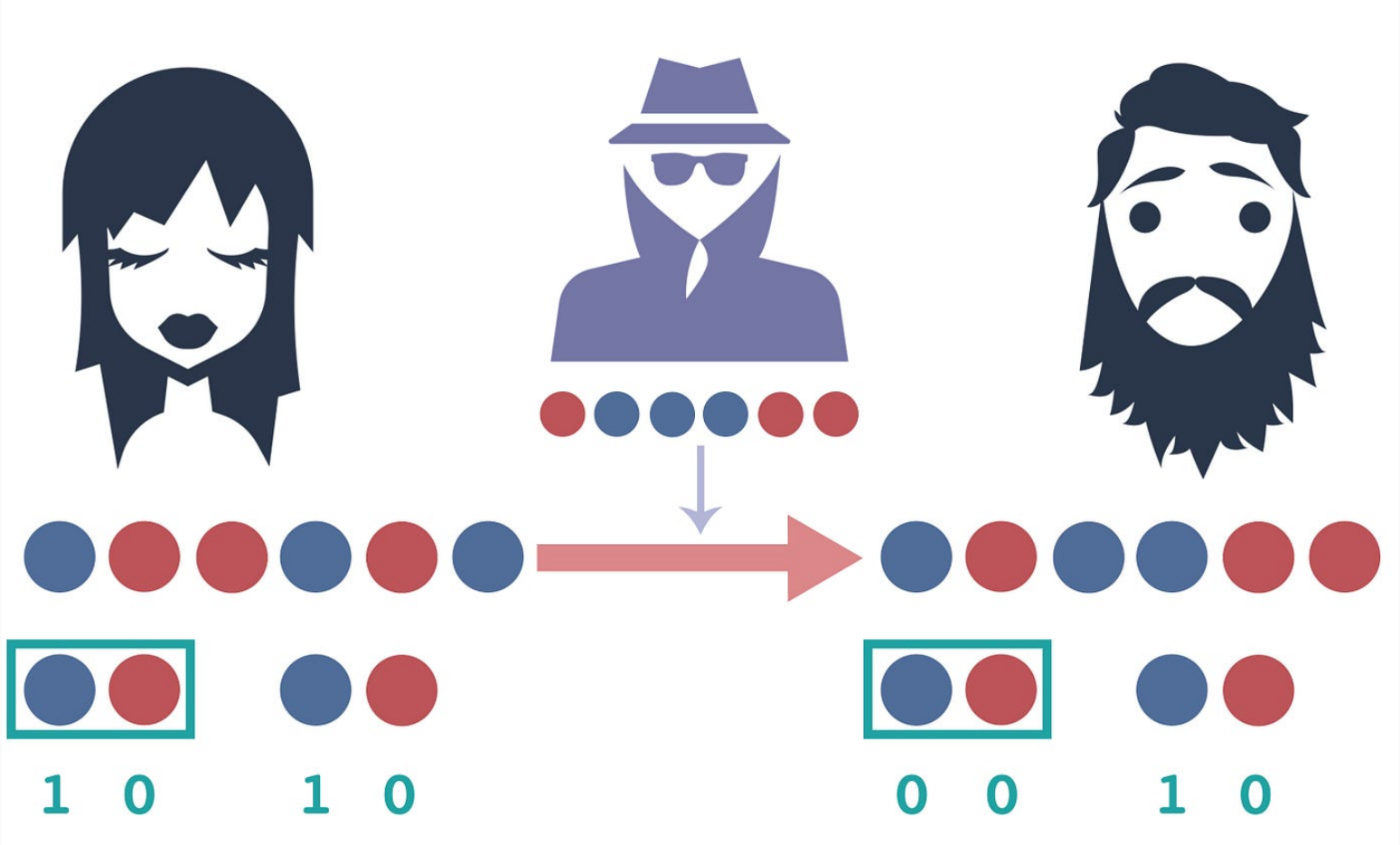
- in NP
- not known to be in P
- not known to be in co-NP
- not known (or *expected*) to be NP-complete
- shown to be in *quasi-polynomial time*, i.e. in

$$\mathrm{TIME}(n^{(\log n)^k})$$

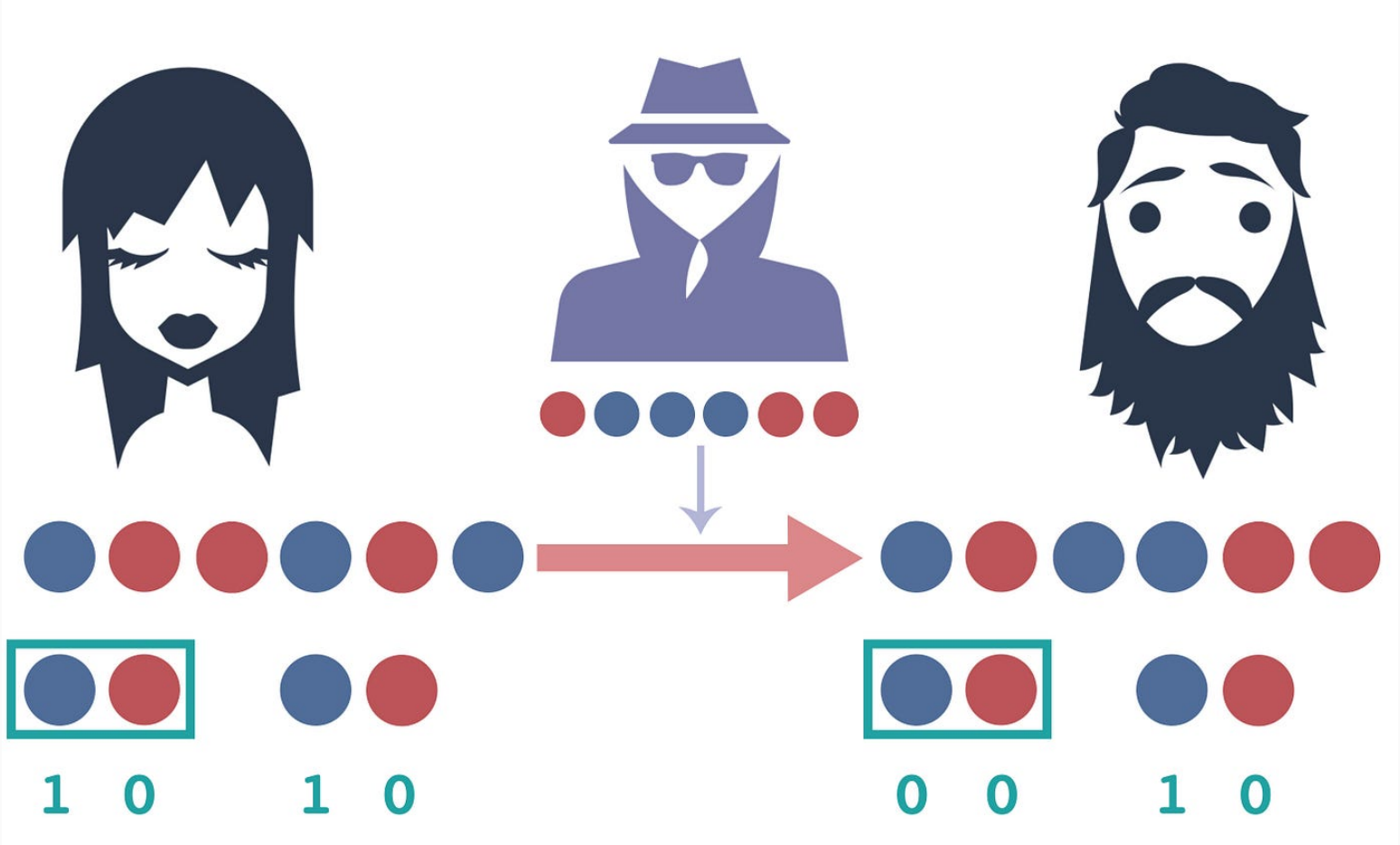
for a constant k.

Cryptography

Cryptography



Cryptography



Alice wishes to communicate with Bob without Eve eavesdropping.

Private Key

In a private key system, there are two secret keys

e – the encryption key

d – the decryption key

and two functions D and E such that:

for any x,

$$D(E(x, e), d) = x.$$

Private Key

In a private key system, there are two secret keys

e – the encryption key

d – the decryption key

and two functions D and E such that:

for any x,

$$D(E(x, e), d) = x.$$

Private Key

In a private key system, there are two secret keys

e – the encryption key

d – the decryption key

and two functions D and E such that:

for any x,

$$D(E(x, e), d) = x.$$

For instance, taking $d = e$ and both D and E as *exclusive or*, we have the *one time pad*:

$$(x \oplus e) \oplus e = x$$

One Time Pad

The one time pad is provably secure, in that the only way Eve can decode a message is by knowing the key.

One Time Pad

The one time pad is provably secure, in that the only way Eve can decode a message is by knowing the key.

If the original message x and the encrypted message y are known, then so is the key:

$$e = x \oplus y$$

One Time Pad

The one time pad is provably secure, in that the only way Eve can decode a message is by knowing the key.

If the original message x and the encrypted message y are known, then so is the key:

$$e = x \oplus y$$

Declassified files reveal how pre-WW2 Brits smashed Russian crypto

Moscow's agents used one-time pads, er, two times – ой!

John Leyden Thu 19 Jul 2018 | 18:35 UTC

Efforts by British boffins to thwart Russian cryptographic cyphers in the 1920s and 1930s have been declassified, providing fascinating insights into an obscure part of the history of code breaking.

America's National Security Agency this week released papers from John Tiltman, one of Britain's top cryptanalysts during the Second World War, describing his work in breaking Russian codes [PDF], in response to a Freedom of Information Act request.

The Russians started using one-time pads in 1928 – however, they made the grave cryptographic error of allowing these pads to be used twice, the release of Tiltman's papers has revealed for the first time.

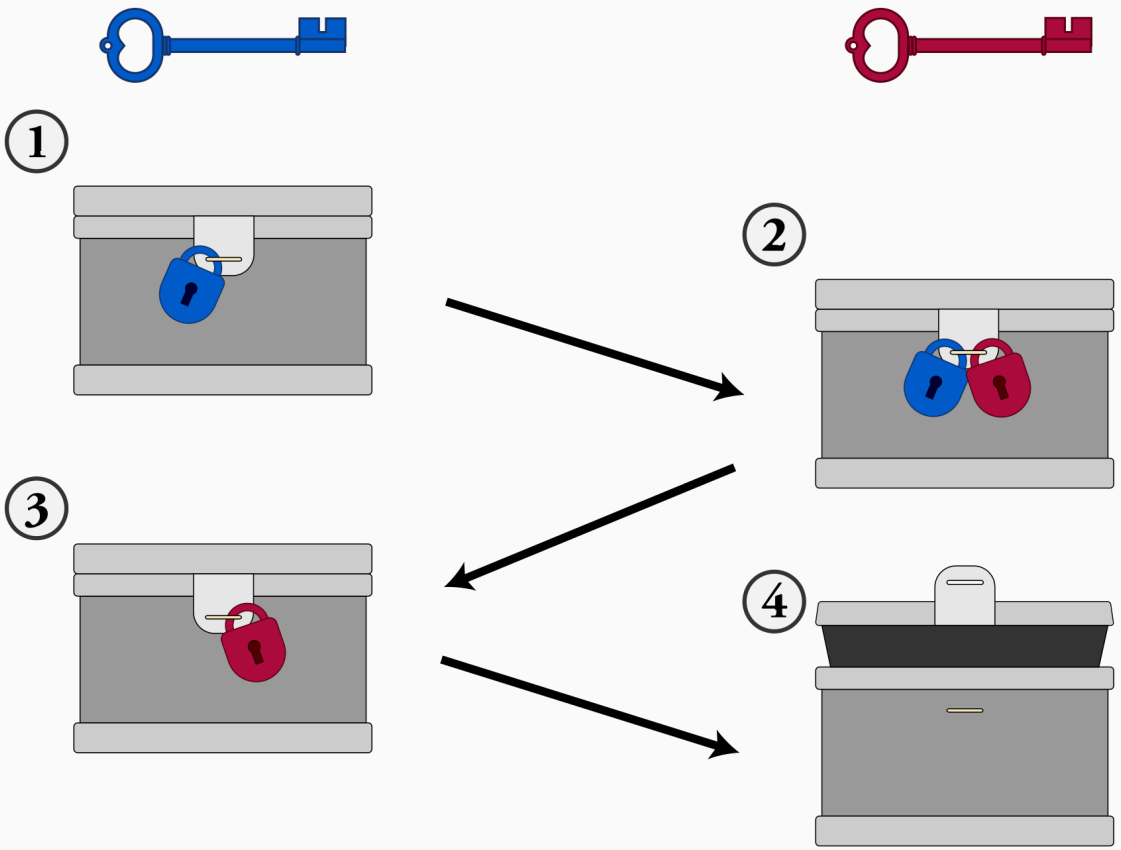
By reusing one-time pads, Russian agents accidentally leaked enough information for eavesdroppers in Blighty to figure out the encrypted missives' plaintext. Two separate messages encrypted reusing the same key from a pad could be compared to ascertain the differences between their unencrypted forms, and from there eggheads could, using stats and knowledge of the language, work out the original words.

Public Key

Is it possible to exchange a message without a secret key?

Public Key

Is it possible to exchange a message without a secret key?



Public Key

In public key cryptography, the encryption key e is public, and the decryption key d is private.

Public Key

In public key cryptography, the encryption key e is public, and the decryption key d is private.

We still have,

for any x,

$$D(E(x, e), d) = x$$

Public Key

In public key cryptography, the encryption key e is public, and the decryption key d is private.

We still have,

for any x,

$$D(E(x, e), d) = x$$

Public Key

In public key cryptography, the encryption key e is public, and the decryption key d is private.

We still have,

for any x,

$$D(E(x, e), d) = x$$

D and E are polynomial time computable, hence decoding is in NP (why?)

Public Key

In public key cryptography, the encryption key e is public, and the decryption key d is private.

We still have,

for any x,

$$D(E(x, e), d) = x$$

D and E are polynomial time computable, hence decoding is in NP (why?)

The key e is public. Given y, a certificate is x such that $y = E(x, e)$.

Public Key

In public key cryptography, the encryption key e is public, and the decryption key d is private.

We still have,

for any x,

$$D(E(x, e), d) = x$$

D and E are polynomial time computable, hence decoding is in NP (why?)

The key e is public. Given y, a certificate is x such that $y = E(x, e)$. (Is that precise?)

Public Key

In public key cryptography, the encryption key e is public, and the decryption key d is private.

We still have,

for any x,

$$D(E(x, e), d) = x$$

D and E are polynomial time computable, hence decoding is in NP (why?)

The key e is public. Given y, a certificate is x such that $y = E(x, e)$. (Is that precise?)

Thus, public key cryptography is not *provably secure* in the way that the one time pad is. It relies on the assumption that $\mathrm{P} \neq \mathrm{NP}$.

Abstracting chests and locks with computational hardness

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

1. f is one-to-one.

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

1. f is one-to-one.
2. for each x, $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for some k.

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

1. f is one-to-one.
2. for each x, $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for some k.
3. f is computable in polynomial time.

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

1. f is one-to-one.
2. for each x, $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for some k.
3. f is computable in polynomial time.
4. f^{-1} is *not* computable in polynomial time.

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

1. f is one-to-one.
2. for each x, $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for some k.
3. f is computable in polynomial time.
4. f^{-1} is *not* computable in polynomial time.

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

1. f is one-to-one.
2. for each x, $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for some k.
3. f is computable in polynomial time.
4. f^{-1} is *not* computable in polynomial time.

We cannot hope to prove the existence of one-way functions without at the same time proving $\mathrm{P} \neq \mathrm{NP}$.

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

1. f is one-to-one.
2. for each x, $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for some k.
3. f is computable in polynomial time.
4. f^{-1} is *not* computable in polynomial time.

We cannot hope to prove the existence of one-way functions without at the same time proving $\mathsf{P} \neq \mathsf{NP}$.

It is strongly believed that the RSA function:

$$f(x, e, p, q) = (x^e \bmod pq, pq, e)$$

is a one-way function.

UP

Though one cannot hope to prove that the RSA function is one-way without separating P and NP, we might hope to make it as secure as a proof of NP-completeness.

UP

Though one cannot hope to prove that the RSA function is one-way without separating P and NP, we might hope to make it as secure as a proof of NP-completeness.

Definition

A nondeterministic machine is *unambiguous* if, for any input x, there is at most one accepting computation of the machine.

UP

Though one cannot hope to prove that the RSA function is one-way without separating P and NP, we might hope to make it as secure as a proof of NP-completeness.

Definition

A nondeterministic machine is *unambiguous* if, for any input x, there is at most one accepting computation of the machine.

UP is the class of languages accepted by unambiguous machines in polynomial time.

UP

Equivalently, UP is the class of languages of the form

$$\{x \mid \exists y R(x, y)\}$$

Where R is polynomial time computable, polynomially balanced, *and* for each x, there is *at most one* y such that $R(x, y)$.

UP One-way Functions

We have

$$\mathsf{P} \subseteq \mathsf{UP} \subseteq \mathsf{NP}$$

UP One-way Functions

We have

$$\mathrm{P} \subseteq \mathrm{UP} \subseteq \mathrm{NP}$$

It seems unlikely that there are any NP-complete problems in UP.

UP One-way Functions

We have

$$\mathsf{P} \subseteq \mathsf{UP} \subseteq \mathsf{NP}$$

It seems unlikely that there are any NP-complete problems in UP.

One-way functions exist *if, and only if,* $\mathsf{P} \neq \mathsf{UP}$.

Bonus: randomisation and BPP