Introduction to Computer Science

Polly Huang
NTU EE
http://homepage.ntu.edu.tw/~pollyhuang
pollyhuang@ntu.edu.tw

Chapter 12

Theory of Computation

Chapter 12: Theory of Computation

- 12.1 Functions and Their Computation
- 12.2 Turing Machines
- 12.3 Universal Programming Languages
- 12.4 A Noncomputable Function
- 12.5 Complexity of Problems
- 12.6 Public Key Cryptography

Functions

- The relationship between
  - F: x → y
- A collection of possible input values
  - x
- And a collection of possible output values
  - y
- So that each possible input is assigned a single output
  - y = F(x)

Computing

- To determine the output value given an input value
  - Given x
  - Get y

Example: Yards → Meters Conversion

Yards (input)	Meters (output)
1	0.9144
2	1.8288
3	2.7432
4	3.6576
5	4.5720
.	.
.	.
.	.

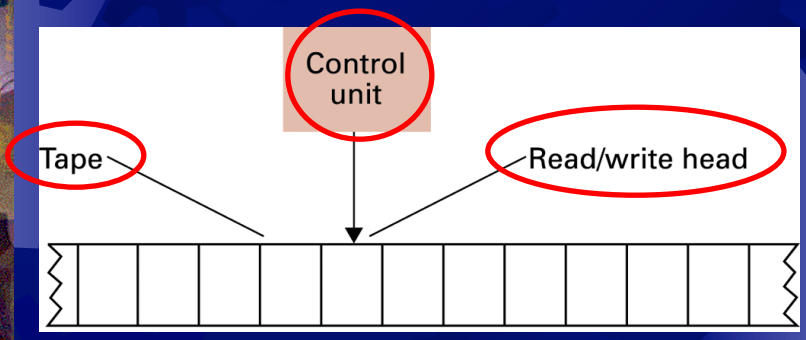
Computability

- Computable function
  - Some functions are computable
  - Output values can be determined algorithmically from the input values
- Non-computable function
  - Some functions are not computable
  - No well-defined, step-by-step process for determining output based given input values
  - Beyond the ability of algorithmic systems

Chapter 12: Theory of Computation

- 12.1 Functions and Their Computation
- 12.2 Turing Machines
- 12.3 Universal Programming Languages
- 12.4 A Noncomputable Function
- 12.5 Complexity of Problems
- 12.6 Public Key Cryptography

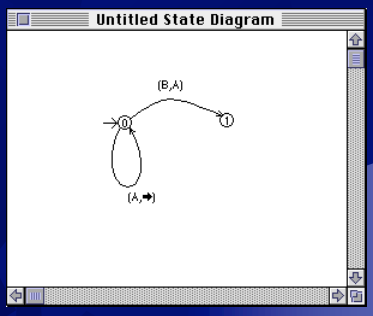
Turing Machine



That is not all!

Theory and Practice

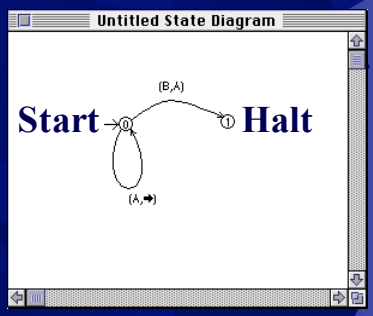
- The theoretical part
  - State machine
- The practical part
  - Head and tape



(Source: Turing's World, http://www-csli.stanford.edu/hp/Turing1.html)

State Machines

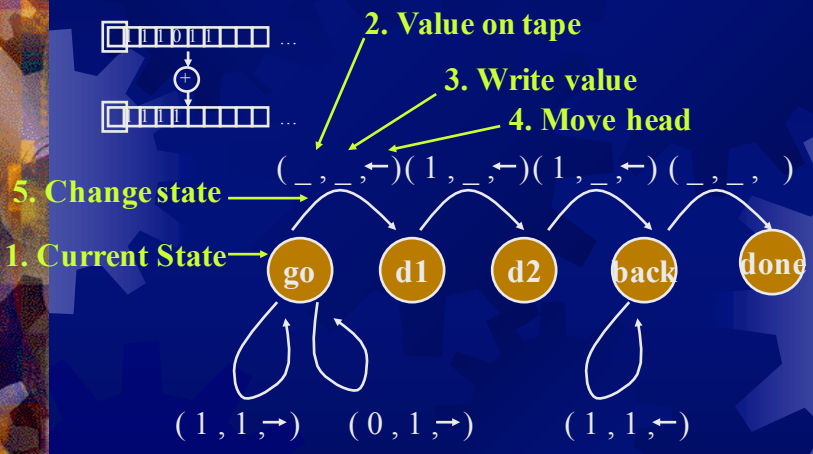
- Alphabets and symbols
- States
  - Start state
  - Halt state



Turing Machine Operation

- Inputs at each step
  - 1. Current state
  - 2. Value at current tape position
- Actions at each step
  - 3. Write a value at current tape position
  - 4. Move read/write head
  - 5. Change state

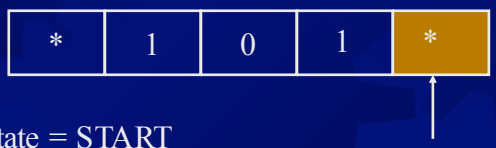
Remember This?



Ex: Incrementing a Value

Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	1	1	Right	RETURN
RETURN	*	*	No move	HALT

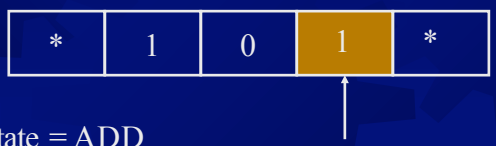
Begin



state = START

Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	1	1	Right	RETURN
RETURN	*	*	No move	HALT

Step 2

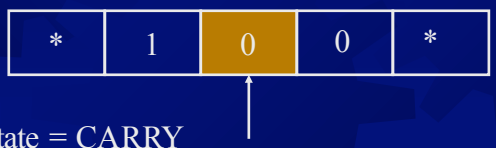


state = ADD

Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	1	1	Right	RETURN
RETURN	*	*	No move	HALT

17

Step 3



state = CARRY

Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	1	1	Right	RETURN
RETURN	*	*	No move	HALT

18

Step 4

state = RETURN

Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	1	1	Right	RETURN
RETURN	*	*	No move	HALT

19

Stop

state = HALT

Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	1	1	Right	RETURN
RETURN	*	*	No move	HALT

20

Turing Computable Functions

Functions that can be computed by a Turing machine

Church-Turing Thesis

"The computable functions and the Turing computable functions are considered one and the same."

In Other Words

- Any computable function is Turing computable
- Not proven, but generally accepted

A Turing Machine

- Essence of computational process
- As good as any algorithmic system
  - If a problem can not be solved by a Turing machine, it can not be solved by any algorithmic system
  - If a problem can be solved by any algorithmic system, it can be solved by a Turing machine
- Theoretical bound on the capabilities of actual machines

Chapter 12: Theory of Computation

- 12.1 Functions and Their Computation
- 12.2 Turing Machines
- 12.3 Universal Programming Languages
- 12.4 A Noncomputable Function
- 12.5 Complexity of Problems
- 12.6 Public Key Cryptography

Universal Programming Language

- A language that can
- Express a program to
- Compute any computable function
- Examples
  - "Bare Bones"
  - Most popular programming languages
- If a problem can be solved algorithmically, it can be expressed using this language.

The Bare Bones Language

- Bare Bones
  - Very simple and yet universal language
- Statements
  - `clear` *`name`*`;`
  - `incr` *`name`*`;`
  - `decr` *`name`*`;`
  - `while` *`name`* `not 0 do;` … `end;`

Tomorrow ← Today

```
clear Aux;
clear Tomorrow;
while Today not 0 do;
   incr Aux;
   decr Today;
end;
while Aux not 0 do;
   incr Today;
   incr Tomorrow;
   decr Aux;
end;
```

Aux=Today, Today=0
←Value of Aux, Today?

Today=Aux,
Tomorrow=Aux,
Aux=0
←Value of Today,
Tomorrow, Aux?

X * Y

```
clear Z;
while X not 0 do;
   clear W;
   while Y not 0 do;
      incr Z;
      incr W;
      decr Y;
   end;
   while W not 0 do;
      incr Y;
      decr W;
   end;
   decr X;
end;
```

Value of Z, W, Y?
Z+=Y, W=Y, Y=0

Value of Y and W?
Y=W, W=0

Quiz Time!

Universality of Bare Bones

- Using the 'increment' Turing machine to compute **`incr X`**
- Bare Bones program for the addition function
  - Assign, add, multiply
- Bare Bones language can be used to express algorithms for computing all (Turing-) computable functions
- Any computable function can be computed by a program written in Bare Bones

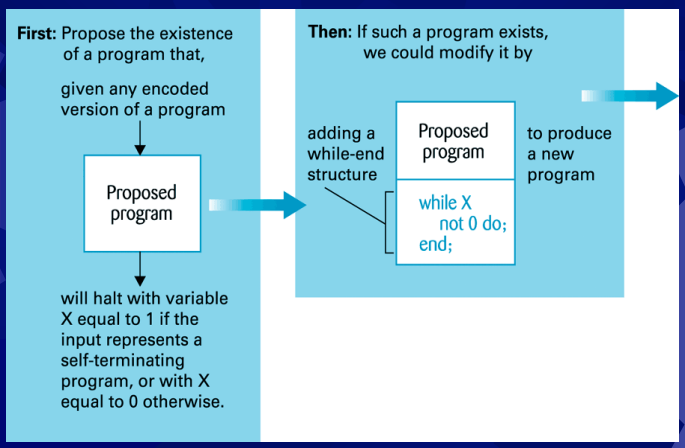
Chapter 12: Theory of Computation

- 12.1 Functions and Their Computation
- 12.2 Turing Machines
- 12.3 Universal Programming Languages
- 12.4 A Noncomputable Function
- 12.5 Complexity of Problems
- 12.6 Public Key Cryptography

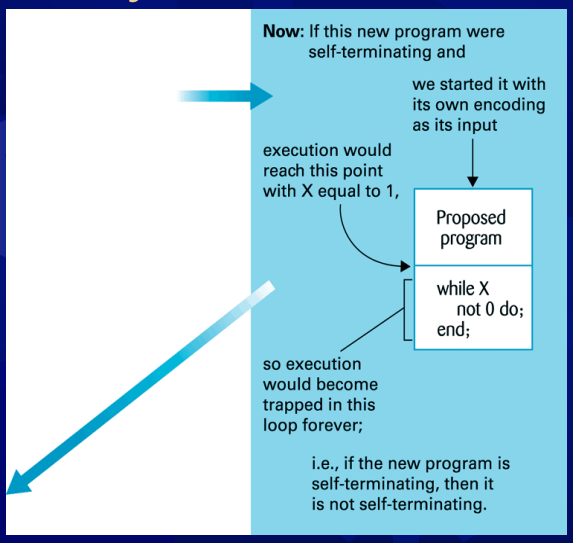
The Halting Problem

- This problem is not computable!
- Given the encoded version of any program
  - Return 1 if the program will eventually halt
  - Return 0 if the program will run forever
- Trying to predict in advance whether a program will terminate (or halt)

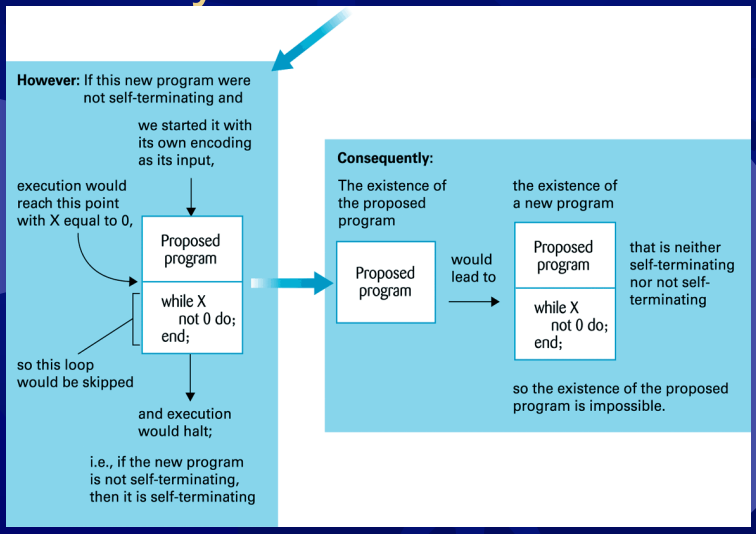
Assume a Solution Exist



Proof by Contradiction I



Proof by Contradiction II



A Story of Contradiction

- Mother and child coming across a river
- The child falls
- A crocodile grabs the child
- Mother pleads that the crocodile let go of the child

Crocodile's Challenge

- The Crocodile says to the mother
- Say something
- If it is right, I'll give you your child back
- If wrong, I'll eat your child

Smart Mom?

- The troubled mother thinks and says
- You will eat my child
- The crocodile …

Moral of the Story

There are problems in the world that is not computable!

Chapter 12: Theory of Computation

- 12.1 Functions and Their Computation
- 12.2 Turing Machines
- 12.3 Universal Programming Languages
- 12.4 A Noncomputable Function
- 12.5 Complexity of Problems
- 12.6 Public Key Cryptography

Complexity of Problems

- **Time complexity**
  - Number of instruction executions required
  - "Complexity" → "time complexity"
  - For a series of n entries, f(n)
- A problem is in class O(f(n)) if it can be solved by an algorithm in Θ(f(n)) or better.
- Θ(n^2): complexity is kn^2+o(n^2)
  - $f(n)/n^2 \rightarrow k$, $n \rightarrow \infty$

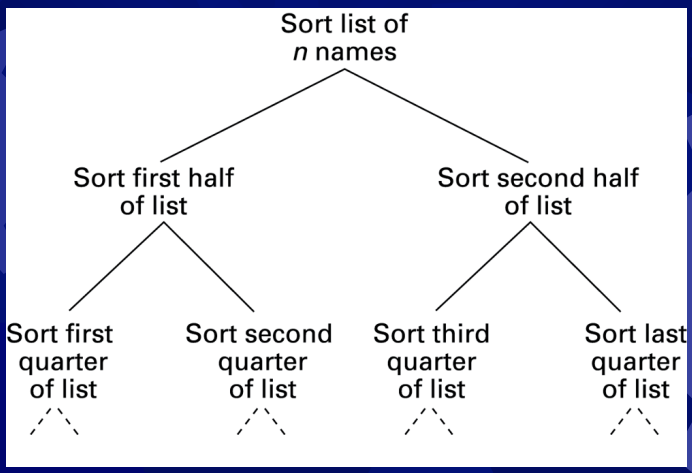
Merging Two Lists

```
procedure MergeLists (InputListA, InputListB, OutputList)

if (both input lists are empty) then (Stop, with OutputList empty)
if (InputListA is empty)
    then (Declare it to be exhausted)
    else (Declare its first entry to be its current entry)
if (InputListB is empty)
    then (Declare it to be exhausted)
    else (Declare its first entry to be its current entry)
while (neither input list is exhausted) do
        (Put the "smaller" current entry in OutputList;
          if (that current entry is the last entry in its corresponding input list)
              then (Declare that input list to be exhausted)
              else (Declare the next entry in that input list to be the list's current entry )
        )

Starting with the current entry in the input list that is not exhausted,
        copy the remaining entries to OutputList.
```

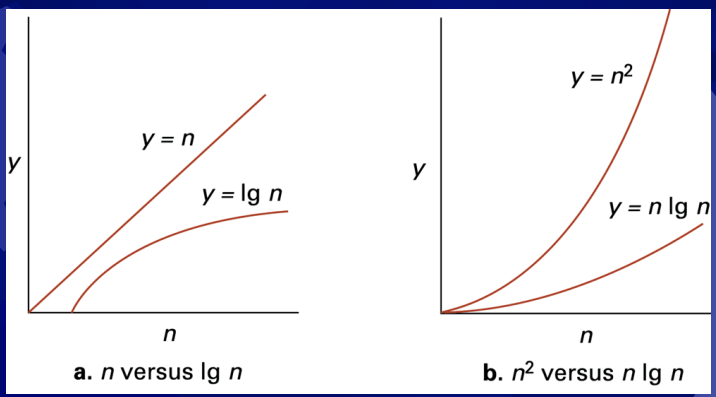

Merge Sort

```
procedure MergeSort (List)

if (List has more than one entry)
   then (Apply the procedure MergeSort to sort the first half of List;
              Apply the procedure MergeSort to sort the second half of List;
              Apply the procedure MergeLists to merge the first and second
                   halves of List to produce a sorted version of List
             )
```

The Hierarchy



n, lgn, nlgn, n^2



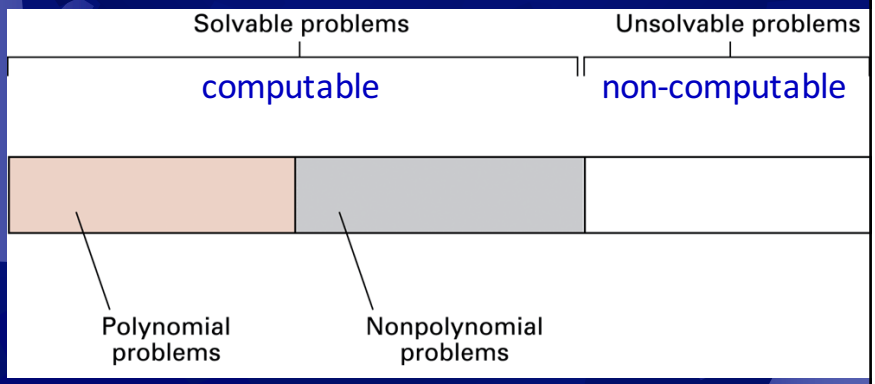
Class P

- Class P
  - All problems in any class $\Theta(f(n))$, where f(n) is a polynomial
- Intractable
  - All problems too complex to be solved practically
  - Most computer scientists consider all problems not in class P to be intractable.

Intractable Problems

- Intractable - All problems
  - too complex to be solved practically
- (1) Most computer scientists consider all problems not in class P to be intractable.
- (2) Non-computable problems, of course intractable
  - E.g., halting problem, the self-contradiced crocodile problem

Problem Classification



The Challenge is

- Given a problem
- Can you tell for sure if it's P or non-P or non-computable?
- For some problems, yes
  - Search, sort, halting
- But for some problems, not quite

P or non-P?

How do you know if there exists a polynomial time algorithm for a problem?

Recall Quiz #17

- The old bank ATM system requires a password of 4 digits (from 0-9)
- Hack the password

Many Solutions

- Exhaustively try passwords
  - 0000, 0001, 0002, …, 0010, 0011, …
- Try the popular passwords
- Identify the birthday of the potential victim
- …

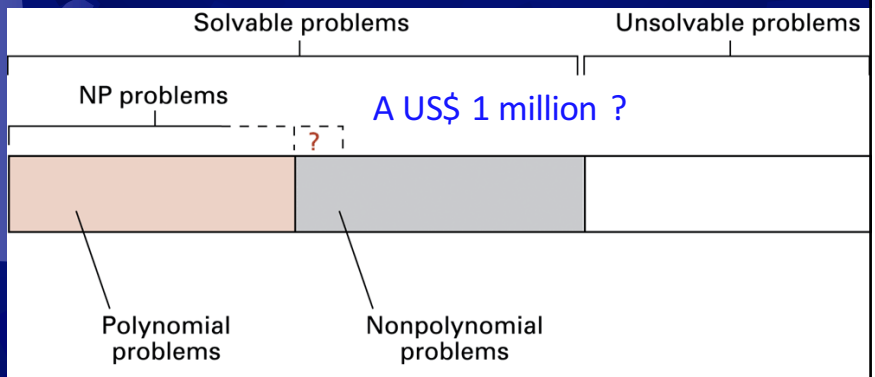
Common of these Solutions

- One can't compute for the sure password
- But one can compute to verify if the guess/attempt is successful

Class NP

- Class NP - All problems
  - (1) whose answers can be verified in polynomial time
  - (2) that may be solved by a non-deterministic algorithm
- Non-deterministic algorithm - An "algorithm"
  - whose steps may not be uniquely and completely determined by the process state
    - May require "creativity", "guessing", "randomness"
- Whether the class NP is bigger than class P is currently unknown.

Problem Classification



NP-Complete Problems

- Problems having the property that
  - A polynomial time solution for any of them
  - Would provide a polynomial time solution for all the other problems in NP as well
- ~Hardest problems in NP

The Quest is simiplified to

Find a polynomial solution for any of the NP-Complete problems

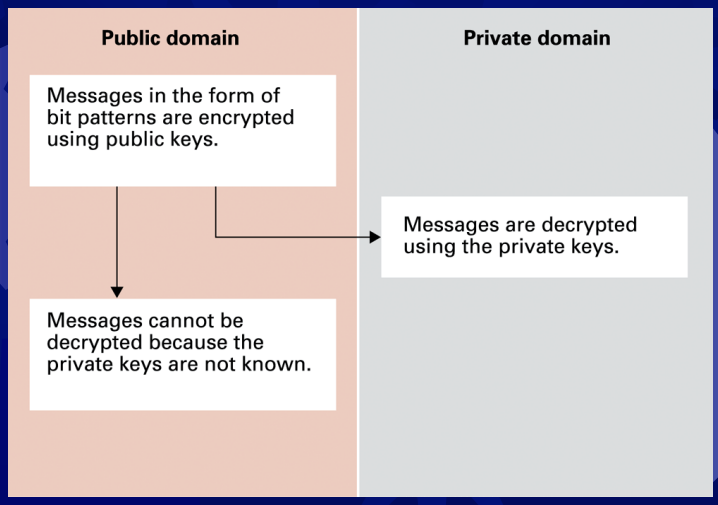
Chapter 12: Theory of Computation

- 12.1 Functions and Their Computation
- 12.2 Turing Machines
- 12.3 Universal Programming Languages
- 12.4 A Noncomputable Function
- 12.5 Complexity of Problems
- 12.6 Public Key Cryptography

Public Key Cryptography

- Key
  - Specially generated set of values used for encryption
  - Public key: used to encrypt messages
  - Private key: used to decrypt messages
- RSA
  - A popular public key cryptographic algorithm
  - Relies on the (presumed) intractability of the problem of factoring large numbers

RSA



Encrypt and Decrypt

- Public key: e
- Private key: d
- Generator: n

- $(data)^e \bmod n = secret$
- $(secret)^d \bmod n = data$

- $(secret)^e \bmod n <> data$

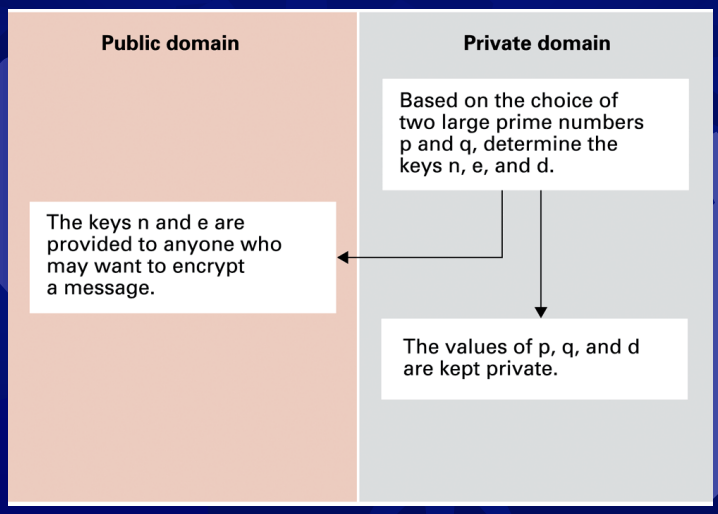
Encrypting 10111

- Encrypting keys: e = 5 and n = 91
- $10111_{two} = 23_{ten}$
- $23^{e} = 23^{5} = 6{,}436{,}343$
- 6,436,343 ÷ 91 has a remainder of 4
- $4_{ten} = 100_{two}$
- Therefore, encrypted version of 10111 is 100.

Decrypting 100

- Decrypting keys: d = 29, n = 91
- $100_{two} = 4_{ten}$
- $4^{d} = 4^{29} = 288{,}230{,}376{,}151{,}711{,}744$
- 288,230,376,151,711,744 ÷ 91 has a remainder of 23
- $23_{ten} = 10111_{two}$
- Therefore, decrypted version of 100 is 10111.

RSA Key Generation



Given 2 Prime #: p and q

- Compute n = pq
- Compute φ = (p-1)(q-1)
- Find d, e such that de mod φ = 1

- Keep d in private
- Give e to the public

Quiz Time!

One algorithm

- 1. Factorize 91
- 2. Compute φ = (p-1)(q-1)
- 3. Populate possible de, say DE
- 4. Find DE/e such that {(DE mod e) = 0}

Remember Q25?

- Listen to Gru's plan on stealing the moon
- Sounds easy?
- Suffices as an algorithm?

Which step is the hardest?

- 1. Factorize 91
- 2. Compute φ = (p-1)(q-1)
- 3. Populate possible de, say DE
- 4. Find DE/e such that {(DE mod e) = 1}

Quiz Time!

295927 is an RSA-6

6 decimal digits (19 bits)

RSA-100

100 decimal digits (330 bits)

RSA-100 =
1522605027922533360535618378132637429718068114961380688657908494580122963258952897654000350692006139

RSA-100 =
3797522793694367392280887275544562785456553663819 9 ×
4009469095092088103068373529276146838921489972406 1

RSA-2048

617 decimal digits (2048 bits)
Cash prize: US$200,000

RSA-2048 =
2519590847565789349402718324004839857142928212620403202777713783604366202070759555626401852588078440691829064124951508218929855914917618450280848912007284499268739280728777673597141834727026189637501497182469116507761337985909570009733045974880842840179742910064245869181719511874612151517265463228221686998754918242243363725908514186546204357679842338718477444792073993423658482382428119816381501067481045166037730605620161967625613384414360383390441495263443219011465754445417842402092461651572335077870774981712577246796292638635637328991215483143816789988504044536402352738195137863656439121201039712282212072035 7

About RSA

Rivest, Shamir, Adleman

Questions?