Students requests

Sebastiano Tronto

2021-05-21

UNIVERSITÉ DU LUXEMBOURG

More cryptography

What we have seen:

- **RSA:** sending messages using a private key / public key pair
- **Flip-a-coin:** cryptographic "proof" that the opponent is not cheating

Cryptography

- Rely on integer factorization being hard

  Example: the best-known factorization algorithm (*General number field sieve*) has complexity

$$\sim O\left(e^{\sqrt[3]{\frac{64}{9}\log_2 n\cdot(\log_2\log_2 n)^2}}\right)$$

  Factoring a number with 300 digits:

  - Your laptop: 10^{13} billion years
  - Best supercomputer: 13 billion years (age of the universe)

UNIVERSITÉ DU LUXEMBOURG

Symmetric and asymmetric cryptography

- Our examples are *asymmetric*: different public/private keys
- Safe against eavesdroppers
- Symmetric protocols can be faster and simpler, but you need a secure way to exchange a key

UNIVERSITÉ DU LUXEMBOURG

Diffie-Hellman key exchange

- Generate a “password” without communicating it directly
- It can then be used for symmetric cryptography
- Based on a different hard problem: *discrete logarithm*

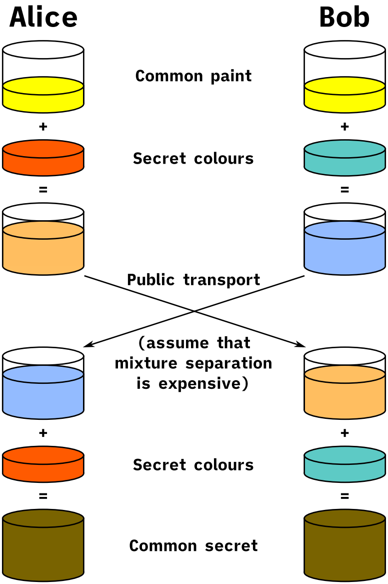
UNIVERSITÉ DU LUXEMBOURG

Diffie-Hellman key exchange

- Alice and Bob agree on a prime number p and an integer g
- Alice picks an integer a and sends (g^a mod p) to Bob
- Bob picks an integer b and sends (g^b mod p) to Alice
- Alice can compute $(g^b)^a$ mod p and Bob can compute $(g^a)^b$ mod p. This is their shared secret (key).

UNIVERSITÉ DU LUXEMBOURG

Diffie-Hellman with colors (from Wikipedia)



Diffie-Hellman key exchange

- Knowing h and a, it is hard to find g such that $g^a \bmod p = h$ (discrete logarithm problem)
- Very simple, many variants
- Any group can be used, e.g. Elliptic Curves (see Wikipedia: elliptic-curve Diffie-Hellman)

UNIVERSITÉ DU LUXEMBOURG

Numerical methods for PDEs

Solving partial differential equations

- Very, very hard
- Very important in practical applications (physics and such)
- Approximations are necessary, might as well use numerical methods

UNIVERSITÉ DU LUXEMBOURG

Numerical methods for ODEs

Problem

Given $f(x, y)$, x_0 and y_0, find an approximation for $y(x)$ such that

$$\begin{cases} y'(x) = f(x, y(x)) \\ y(x_0) = y_0 \end{cases}$$

Approximation

We can describe $y(x)$ in an interval $[x_0, x_1]$ by giving the (approximate) values $y(s_0), \ldots, y(s_n)$ for many values of $s_i \in [x_0, x_1]$.

UNIVERSITÉ DU LUXEMBOURG

Euler's method

Idea

For h small

$$y'(x) \approx \frac{y(x+h) - y(x)}{h}$$

which implies

$$y(x+h) \approx y(x) + h \cdot f(x, y(x))$$

Euler's method

Algorithm

Input: the data $f(x, y)$, x_0, y_0 and x_1 describing the problem and the desired range for the solution.

Output: $x_0 = s_0 < s_1 < \cdots < s_n = x_1$ and $y_0, \ldots, y_n$ such that $y_i \approx y(s_i)$.

1. Choose a value n and let $h = \frac{x_1 - x_0}{n}$ and $s_i = x_0 + ih$
2. For $i = 0, \ldots, n-1$ compute $y_{i+1} = y_i + h \cdot f(s_i, y_i)$
3. Return $s_0, \ldots, s_n$ and $y_0, \ldots, y_n$

UNIVERSITÉ DU LUXEMBOURG

Euler's method

- Very simple and fast
- Generalization for higher-order equations: Runge-Kutta methods
- A similar idea works for some PDEs

UNIVERSITÉ DU LUXEMBOURG

The heat equation (PDE)

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \cdots + \frac{\partial^2 u}{\partial x_n^2}$$

Where

$$u(x_1, x_2, \ldots, x_n, t) : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$$

describes the quantity of heat at the point $(x_1, \ldots x_n)$ at time t.

It appears also outside thermodynamics: mathematical finance (Black-Scholes equation), quantum mechanics (Schrödinger equation), image analysis. . .

UNIVERSITÉ DU LUXEMBOURG

A simple case ($n = 1$, in $[0, 1]^2$)

Problem

Given $u_0(t)$, $u_1(t)$ and $u^0(x)$, find an approximation for $u(x, t)$ such that

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \\ u(0, t) = u_{(0)}(t) & \text{(boundary condition)} \\ u(1, t) = u_{(1)}(t) & \text{(boundary condition)} \\ u(x, 0) = u^0(x) & \text{(initial condition)} \end{cases}$$

Approximation

Values $u_i^j \approx u(s_i, r^j)$ for $(s_i, r^j) \in [0, 1] \times [0, 1]$

UNIVERSITÉ DU LUXEMBOURG

Idea

For k small:

$$\frac{\partial u(x,t)}{\partial t} \approx \frac{u(x,t+k)-u(x,t)}{k}$$

For h small (left limit + right limit):

$$\begin{aligned}
\frac{\partial^2 u(x,t)}{\partial x^2} &\approx \frac{\partial}{\partial x}\left(\frac{u(x,t)-u(x-h,t)}{h}\right)\\
&\approx \frac{1}{h}\left(\frac{\partial u(x,t)}{\partial x}-\frac{\partial u(x-h,t)}{\partial x}\right)\\
&\approx \frac{1}{h}\left(\frac{u(x+h,t)-u(x,t)}{h}-\frac{u(x,t)-u(x-h,t)}{h}\right)\\
&\approx \frac{u(x+h,t)-2u(x,t)+u(x-h,t)}{h^2}
\end{aligned}$$

UNIVERSITÉ DU LUXEMBOURG

Idea

From the equation

$$\frac{u_i^{j+1} - u_i^j}{k} = \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h^2}$$

we find the formula

$$u_i^{j+1} = \frac{k}{h^2}\left(u_{i+1}^j - 2u_i^j + u_{i-1}^j\right) + u_i^j$$

UNIVERSITÉ DU LUXEMBOURG

Finite difference method for the heat equation

Algorithm

Input: $u^j_{(0)}$, $u^j_{(1)}$ (boundary) and u^0_i (initial).

Output: values u^j_i approximating a solution.

1. Let $m = \text{len}(u_0) - 1$, $n = \text{len}(u^0) - 1$ and $k = 1/m$, $h = 1/n$
2. For $j = 0, \dots, m-1$ do the following:
   - For $i = 1, \dots, n-1$ compute

$$u_i^{j+1} = \frac{k}{h^2}\left(u^j_{i+1} - 2u^j_i + u^j_{i-1}\right) + u^j_i$$

3. Return the u^j_i

UNIVERSITÉ DU LUXEMBOURG

Other PDEs

- In general, there is no generic method
- You might need to write specific code for your equation
- Some packages exists (e.g. fem-fenics for Gnu Octave)