Day 1: Cryptography & Cryptocurrencies

Welcome!

Today:

$\boxed{1}$ Course overview

$\boxed{2}$ Logistics

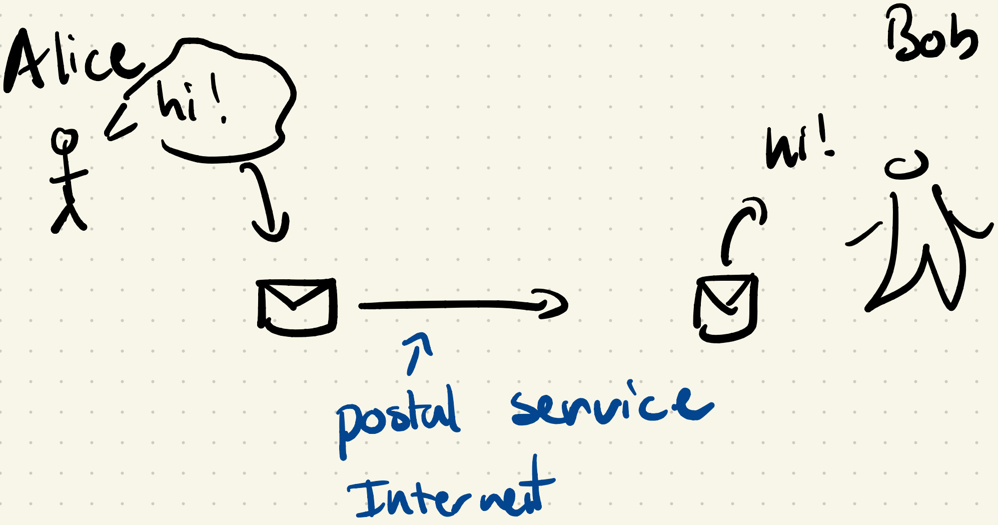
$\boxed{3}$ Lecture

- character representation
- rotation ciphers
- using Replit

$\boxed{4}$ Problem work

$\boxed{5}$ Solution Presentations

$\boxed{1}$ $\underbrace{\text{crypto}}_{\text{hidden}}\underbrace{\text{graphy}}_{\text{writting}}$



might

- not deliver msg → oh well
- read msg → crypto!
- tamper with msg

"confidentiality"

"privacy"

"integrity"

"authenticity"

classic crypto:

- communicating w/o trusting messenger

this course

modern crypto:

- finance w/o trusting a bank
- voting w/o trusting the polls
- outsourcing w/o trusting the cloud

common idea:

"co-operation without trust"

Schedule:

Part I: Cryptography

- symmetric encryption
- hash functions
- groups & modular arithmetic
- key exchange
- digital signatures

Part II: crypto currency

- UTXO model
- Proof-of-work
- a block-chain
- bonus lecture:
  - elliptic curves OR
  - private crypto currencies OR
  - multi-party computation.

2 Logistics

Ethos: balance of lecture and problem-based learning

Every class:

- short lecture, some problems required
- problem work - some are **bonus**
  - Replit
  - groups! < shared code / separate code.

- solution presentations
  - "harkness style"

After class: finish required problems (and bonus problems)

Before class: office hours.

→ have to attend with a friend.

→ schedule on Canvas

$\boxed{\text{3a}}$ Data representation

good for doing math (and crypto)

$$\text{characters} \xrightarrow{\substack{\text{ASCII} \\ \text{(or UTF-8)}}} \text{numbers} \longrightarrow \text{bits}$$

↑ meaningful to humans

↑ a "string" is a sequence of characters

↑ can be stored in a computer

characters :

can be very complex!

- accents Özdemir
- emoji : ☺
- layout control : "change to R→L layout"

in this class we'll limit ourselves to **ASCII** characters

→ those on a US keyboard (more or less)

→ ASCII also defines a 'code' (number) for each character between $\underbrace{0 \text{ and } 255}_{\text{1 byte - 8bits.}}$

→ see asciitable.com

contiguous

$a \mapsto 97$ $A \mapsto 65$

$b \mapsto 98$

$\vdots$

$z \mapsto 122$ $Z \mapsto 90$

$\{ \mapsto 123$

Writing numbers as bits (binary)

Normally we write #s in "base 10"

5 6 1 5×10^2 6×10^1 1×10^0 ✓

5 hundreds

6 tens

1 one

We can also write #s in "base 2"

binary → #

$$\underbrace{0\quad 0}_{\text{leading zeros}}\quad 1\quad 0\quad 1\quad 1$$

$$1\times2^3 + \cancel{0\times2^2} + 1\times2^1 + 1\times2^0 = 11$$

"eights"   "fours"   "twos"   "ones"

→ binary

~~15~~ =

0	0	1	1	1	1
	16	8	4	2	1

~~7~~ ~~3~~ ~~1~~ 1

1111 or 00001111 or 01111

in Python   int('001111',2)==15 ✓

f"{15:06b}"=='001111' ✓

3b Rotation cipher

idea: encrypt letters by "rotating" them:
rot 3

msg:	a	b	c	d	e	...	x	y	z
ct:	d	e	f	g	h	...	a	b	c

↰ "ciphertext"

msg: b a d

↓

ct: e d g

key: amount to rotate by: from 0 to 25.

Welcome to Replit.