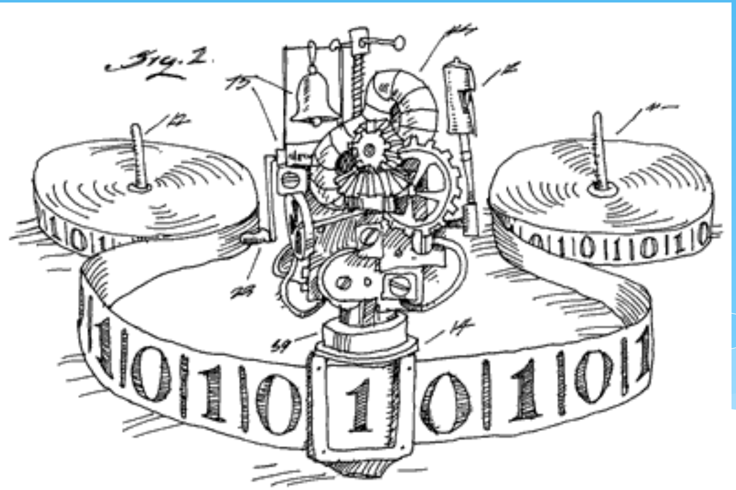
EECS 376: Foundations of Computer Science

Chris Peikert

10 April 2023



Final Feedback

* **Where:** Canvas -> Teaching Evaluations
* **When:** Now until soon
* **Why:**
  1. Part of your 2% grade for surveys
  2. This is how we learn how to teach you better
* **What to do:**
  1. Submit your evaluation.
  2. **Take a snapshot/picture of confirmation screen.**
  3. Upload the snapshot to Gradescope.

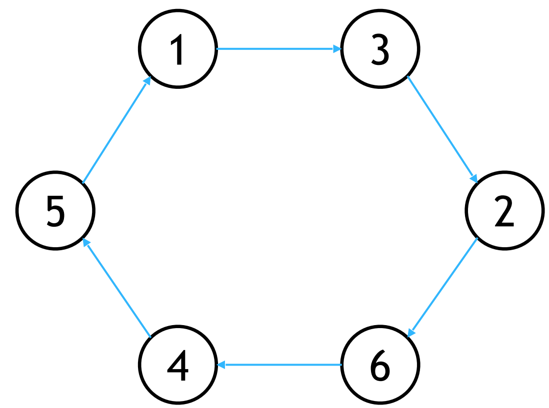
Important!!

Agenda

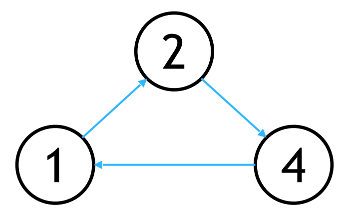
- Administrivia
- Diffie-Hellman recap
- RSA public-key encryption and signatures

A Mathematical "Lock"

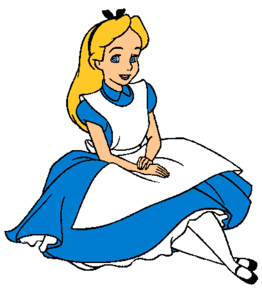
- Let p be a prime and let $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$.
- An integer g is a ***generator*** of $\mathbb{Z}_p^*$ if, for every $x \in \mathbb{Z}_p^*$, there exists $i \in \mathbb{N}$ such that $g^i \equiv x \pmod{p}$.
- **Example:** 3 is a generator of $\mathbb{Z}_7^*$, but 2 isn't.
- **Fact:** $\mathbb{Z}_p^*$ has a generator for *any* prime p.



Discrete Log Conjecture: Given (large) prime p, generator g of $\mathbb{Z}_p^*$, and $x \in \mathbb{Z}_p^*$, there is no *efficient* algorithm for finding $i \in \mathbb{N}$ such that $g^i \equiv x \pmod{p}$.
Probably an "$\mathbf{NP}$-*Intermediate*" problem.



Diffie-Hellman Protocol



$x = (g^a \bmod p)$ $\longrightarrow$

$y = (g^b \bmod p)$ $\longleftarrow$

System parameters: a huge prime p and a generator g of $\mathbb{Z}_p^*$

Alice chooses *secret*, *random* $a \in \mathbb{Z}_p^*$, sends $x = (g^a \bmod p)$ to Bob.

Bob chooses *secret*, *random* $b \in \mathbb{Z}_p^*$, sends $y = (g^b \bmod p)$ to Alice.

Their secret shared key is $\boldsymbol{k} = \left(g^{ab} \bmod p\right)$.

Alice *locally* computes: $y^a \equiv \left(g^b\right)^a \equiv g^{ba} \pmod{p}$.

Bob *locally* computes: $x^b \equiv \left(g^a\right)^b \equiv g^{ab} \pmod{p}$.

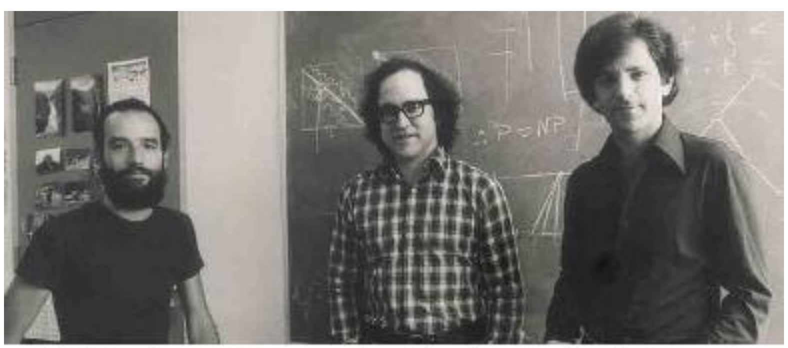
Key: These are equal!

Diffie-Hellman: Security

* Eve sees p, g, $x = g^a \bmod p$, and $y = g^b \bmod p$.
* Eve wants to compute $k = g^{ab} \bmod p$.
* **DH Assumption:** There is no *efficient* algorithm that given g, p, $(g^a \bmod p)$, and $(g^b \bmod p)$ finds $(g^{ab} \bmod p)$.
* **Best *known* attack:** solve DLog to find a (or b).
* **Upshot:** Hard problems are sometimes a *good* thing!
* Most modern cryptographic protocols have ***conditional*** security guarantees: secure if there one-way functions exist, $\mathbf{P} \neq \mathbf{NP}$, DH/RSA/lattices are hard, etc…

RSA

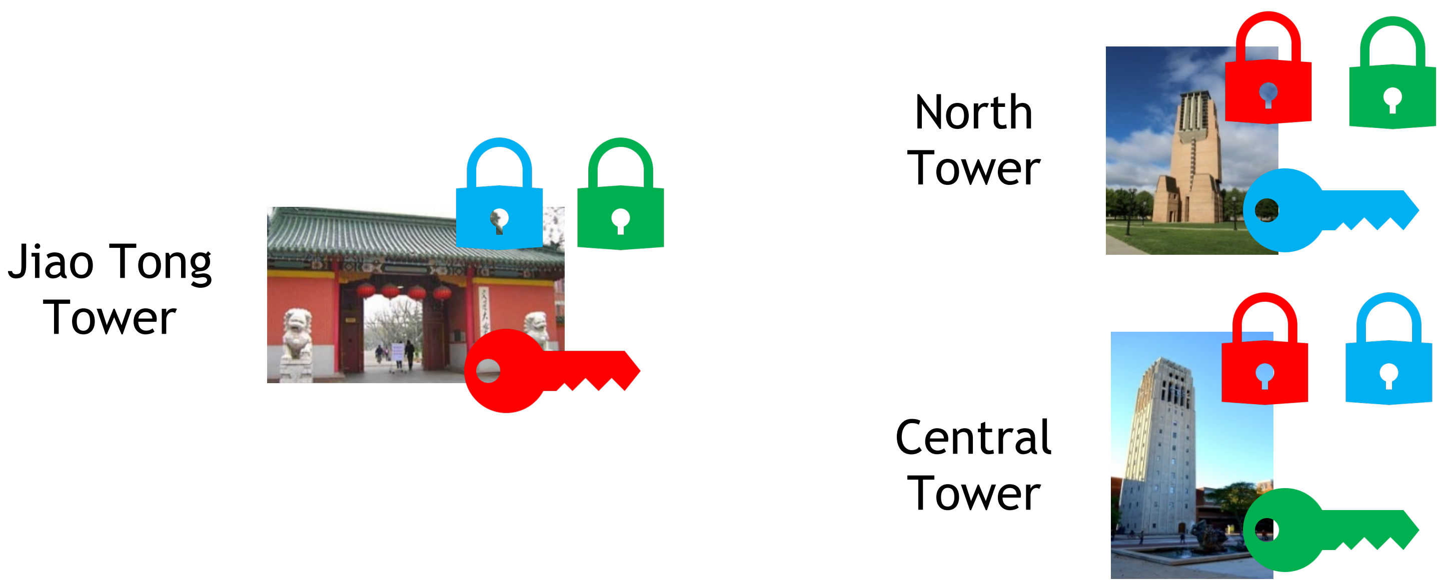
- The first public-key encryption/digital signature scheme
- Invented by Rivest, Shamir, and Adleman in 1977
- Also discovered by Clifford Cocks at British Intelligence in 1973; classified until 1997.



Adi Shamir Ron Rivest Len Adleman

Public-Key Encryption

- **Analogy:** Give your "lock" to everyone; anyone can lock a "package" meant for you using your lock; only you can unlock.
- **Public key (lock):** used by *others* to encrypt messages *to you*
- **Private key (key):** used *by you* to decrypt messages



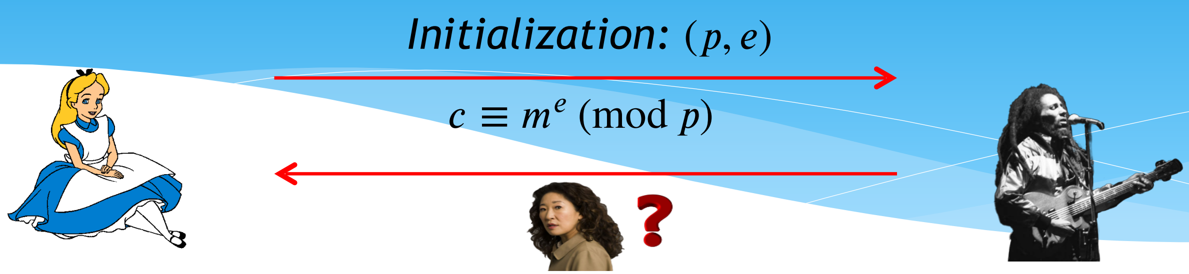
Encryption ≡ 'Trapdoor' Inversion

* A cryptosystem consists of:
  * $\mathbb{Z}_n$ is the set of possible messages (bit strings are numbers!)
  * E_{ek} is the encryption algorithm (w/ public key ek)
  * D_{dk} is the decryption algorithm (w/ secret key dk)
* **Q:** We want $D_{dk}(E_{ek}(m)) = ?$
  * $D_{dk} \circ E_{ek}$ should be the identity function!
* **Goal:** Look for function E_{ek} on $\mathbb{Z}_n$ that is *hard to invert*, but *easy to invert with a 'trapdoor'* (decryption key)

Fermat's Little Theorem

* **FLT:** If p is prime, then for any $a, k \in \mathbb{Z}$, $a^{1+k(p-1)} \equiv a \pmod{p}$.
* **Example:** $a^{11} \bmod 11$ is the identity function on $\mathbb{Z}_{11}$
* **Proof:** If p is prime and $a \not\equiv 0 \pmod{p}$, then the set of numbers $\{a, 2a, 3a, \ldots, (p-1)a\} \pmod{p}$ is the same set as $\{1, \ldots, p-1\}$.
  1) For every $i \in \{1, \ldots, p-1\}$, ia is not a multiple of p since p does not divide either i or a (***Euclid's lemma***). Thus, each $ia \pmod{p} \in \{1, \ldots, p-1\}$.
  2) For every $i, j \in \{1, \ldots, p-1\}$, $i \neq j$, $(j-i)a$ is not a multiple of p. Thus, there are no "collisions": $ia \not\equiv ja \pmod{p}$.
* **Then:** Since the sets are the same, their products are too:
  * $a \cdot 2a \cdots (p-1)a \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}$
  * Hence $a^{p-1} \equiv 1 \pmod{p}$. ($\{1, \ldots, p-1\}$ all have inverses mod p, so multiply both sides by $1^{-1} \cdot 2^{-1} \cdots (p-1)^{-1} \pmod{p}$)

Cryptosystem Attempt

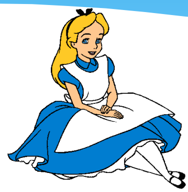


* **FLT:** If p is prime, then for any $a, k \in \mathbb{Z}$, $a^{1+k(p-1)} \equiv a \pmod{p}$.
* Alice picks a large prime p and $e \bullet d = 1 + k(p-1)$, then:
  * Alice sends (p, e) to Bob but keeps d secret.
  * **Enc:** Bob sends $c \equiv m^e \pmod{p}$ to Alice.
  * **Dec:** Alice computes $c^d \equiv (m^e)^d \equiv m^{1+k(p-1)} \equiv m \pmod{p}$.
* **Observation:** $e \bullet d = 1 + k(p-1) \iff e \bullet d \equiv 1 \pmod{p-1}$
  * Alice can choose an e that is coprime to $p-1$ and run the Extended Euclidean Algorithm (EEA) to efficiently compute its inverse $d \equiv e^{-1} \pmod{p-1}$.
* **Q:** Is this secure? Can Eve efficiently recover m from the public information $p, e, m^e \pmod{p}$? (Yes.)

RSA Identity

* **FLT:** If p is prime, then for any $a, k \in \mathbb{Z}$, $a^{1+k(p-1)} \equiv a \pmod{p}$.
* **RSA Identity:** If $n = p \bullet q$ is the product of *two distinct* primes, then for any $a, k \in \mathbb{Z}$: $a^{1+k(p-1)(q-1)} \equiv a \pmod{n}$. (Proof: holds mod each of p,q.)
* **Example:** $a^5 \bmod 10$ is an identity function on $\mathbb{Z}_{10}$
  * $n = 2 \cdot 5$ so $a^{1+4k} \equiv a \ (\text{mod } 10)$ by RSA identity
* **Example:** Compute $3^{123} \bmod 77$
  * $n = 7 \cdot 11$ so $a^{1+60k} \equiv a \ (\text{mod } 77)$
* For encryption we need $e \bullet d = 1 + k(p-1)(q-1)$ $\iff e \bullet d \equiv 1 \ (\text{mod } (p-1)(q-1))$.

RSA: Protocol



Initialization: (n, e)

$c \equiv m^e \pmod{n}$

$c' \equiv (m')^e \pmod{n}$

- To initialize the protocol, **Alice:**
  - picks two *large*, *secret* primes p, q and sets $n = pq$
  - generates ***matching public/private*** exponents (e, d)
    - $e \bullet d \equiv 1 \pmod{(p-1)(q-1)}$ (use EEA)
  - sends Bob (n, e) (public modulus and exponent)

Alice initializes the “lock” and “key” and gives the lock to Bob; only need to do this *once*

- To send m to Alice, Bob sends the ciphertext:

$$c \equiv m^e \pmod{n}$$

- After receiving c, Alice computes:

$$c^d \equiv m^{e \cdot d} \equiv m^{1+k(p-1)(q-1)} \equiv m \pmod{n}$$

RSA Identity

RSA: Toy Example

* Set $n = p \cdot q = 3 \bullet 17 = 51$ (the primes are *secret*, shhh…)
* Generate matching public/private key pair $(e, d) = (3,11)$
  * $e \bullet d \equiv 1 \pmod{32}$
  * E.g., pick e coprime to 32 and compute inverse d using EEA
* Alice sends $(n, e) = (51, 3)$ to Bob
* To send $m = 4$, Bob sends the ciphertext:

$$m^e \equiv 4^3 \equiv 13 \pmod{51}$$

* After receiving $c = 13$, Alice computes:

$$c^d \equiv 13^{11} \equiv 4 \pmod{51}$$

RSA: Security

- Eve knows public n, exponent e, ciphertext $m^e \pmod{n}$.
- Eve wants to compute $m \pmod{n}$.
- **RSA Assumption:** There is no efficient algorithm to find m, given the above info.
  - *Seems* to require knowledge of (p, q), or d, or $(p-1)(q-1)$
- **Factorization Hardness Assumption:** There is no efficient algorithm for integer factorization.
- **Exercise:** Show that, given n and $(p-1)(q-1)$, we can determine p and q.

RSA Factoring Challenge

640 bits, 193 digits

- In 2005, J. Franke et al. **won $20,000 for showing:**

n=3107418240490043721350750035888567930037346022842727545720161488232064405808150455634682967172328678243791627283803341547107310850191954852900733772482278352574238645401469173660247765234 6609

is the product of

p=16347336458092538484431338838650908598417836700330923121811108523893331001045081512121181675115 79

and

q=19008712816648221131268515739354139754718967899685154936666385390880271038021044989571912614655 71

RSA Factoring Challenge

829 bits, 250 digits

* In 2020, F. Boudot et al. showed that:

n=2140324650240744961264423072839333563008614715144755017797754920881418023447140136643345519095804679610992851872470914587687396261921557363047454770520805119056493106687691590019759405693457452230589325976697471681738069364894699 9871578494975937497937

is the product of

p=64135289477071580278790190170577389084825014742943447208116859632024532344630238623598752668347708737661925585694639798853367

and

q=33372027594978156556226010605355114227940760344767554666784520987023841729210037080257448673296881877565718986258036932062711

Factoring is Hard (?)

1024 bits, 309 digits

- **RSA $100,000 challenge (defunct):** factor the following modulus n into two large primes:
- n=135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563

RSA Signatures

* **Motivation:** Ensure that Alice sent m, w/o tampering.
* **Idea:** Run RSA "backwards": sign w/secret, verify w/public
* Setup: public key (n, e) and matching secret key d.
* Sign a message (hash) m: $s = m^d \bmod n$.
* Verify a signature s for m: check that $s^e \equiv m \pmod{n}$.
* Correctness follows from the RSA identity.
* Security from RSA assumption: seems hard to compute "eth root" of a random message hash m.

Quantum Computers, Cryptography and **NP**-Completeness

* Quantum Computers can factor integers, compute DLOG efficiently.
  * So they can break RSA and Diffie-Hellman!
* (Un)fortunately, Quantum Computers don't (yet) scale up enough to break real crypto… but in 15 years? 25? 50?
  * "Post-quantum" crypto: usable today, secure(?) vs. quantum
* If $\mathbf{P} = \mathbf{NP}$, then there is "no cryptography".
* The problems underlying cryptographic protocols (RSA, DH, DLOG, integer factorization, …) are believed to be hard, but ***not*** to be **NP**-Hard.
  * Probably in **NP-Intermediate**: problems in **NP** that are neither in **P** nor **NP**-Complete.