Anonymity Online

MIT - 6.893
Fall 2020
Henry Corrigan-Gibbs

Plan

- * Recap: TLS
- * DC nets
- * Mix nets
- * Tor

Logistics

- * Last HW due on Friday at 5pm (no late days!)
- * Wednesday: Joe Calandrino (FTC) guest lecture.
- * Monday: Hellman Q&A ... PLEASE DO READING and bring your good questions
- * Wednesday 12/7: Precomputation attacks & wrap-up (!)
- * Will schedule an AMA / informal chat

Recap: TLS

* Seems simple ... hard to get right.

* TLS 1.3 eliminates many of the problematic features of earlier versions.

- Compress-then-encrypt
- Old cipher suites
- No forward secrecy (static RSA)
- ....

↳ Deployment underway. We will see how 1.3 fares in practice.

Today we'll talk about two beautiful ideas of David Chaum ... instrumental in development of some really neat privacy tools

[Chaum '88]

DC Nets : Anonymous broadcast

Setting: A group of n players, communicating over a broadcast channel.

Each party i holds x_i

Want to learn $\{x_1, \ldots, x_n\}$ without learning who sent what.

[Adv sees msgs that all players exchange!]
↖ Models network adversary

Applications: → Anonymous feedback form among student in a class

$x_i = \{\text{student } i\text{'s feedback.}\}$

Want all x_i without learning who said what

→ Anonymous Twitter
(FWIW, I'm less and less convinced that this is a good idea)

→ Anonymous point-to-point messaging

$$x_i = (\text{"Alice"}, E(pk_{Alice}, \text{msg for Alice}))$$

- Hides who is sending msgs to Alice.
- Can also hide recipient.

In modern terms, we'd say there are n players who want to run an MPC to compute fn that outputs $(x_1, \dots, x_n)$ in shuffled order.

↖ want security against any # of adversarial participants

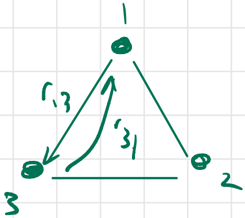
Charm's Protocol

... can think of it as a super-simple MPC

- Each input x_i is a bit $\in \{0,1\}$
- Each player i shares secrets $r_{i1}, \dots, r_{in} \in \{0,1\}$ with all other players.
- Each player publishes $\hat{x}_i := \left(\sum_{j=1}^{n} r_{ij}\right) + x_i \pmod 2$.
- Players can reconstruct

$$y = \sum_{i=1}^{n} \hat{x}_i = \sum x_i \pmod 2$$

Randomness cancels out b/c all random values included twice.



★ Generalizes to larger modulus.

...not quite what we wanted...

DC Nets

Problem: We get the sum $\sum_{i=1}^{n} x_i \pmod 2$

↳ If we work mod $p > n$ we can recover all x_is. ✓

Problem: Longer messages?

Heuristic idea: Use DC-net protocol to implement a shared anonymous broadcast channel (like Ethernet?)

↳ Use exponential backoff to handle collisions.

Nice trick to know: If working mod p and each $x_i \in \mathbb{Z}_p$, then is a simpler/cleaner approach...

- Each player i encodes x_i as $(x_i, x_i^2, x_i^3, \dots, x_i^n)$
- Given $\sum_{i=1}^{n} (x_i, x_i^2, \dots, x_i^n)$ mod p,

  ∃ an efficient alg to recover all x_i ("Newton relations")

DC Nets

Why don't we use them in practice?

- Any one party goes offline, all messages unrecoverable
- Each party sends n bits.... If $n \geq 2^{30}$ as in Twitter, each person sends gigabytes of data or worse

↳ Possible to address both of these to some degree using fancier crypto tools see: Herbivore, Dissent, Riposte, Blinder, ...

- Total work to recover all msgs is $\Omega(n^2)$...

↑ For $n \simeq 2^{20}$, this is a non-starter.

(IMO, this is the serious bottleneck)

[Chaum '81]

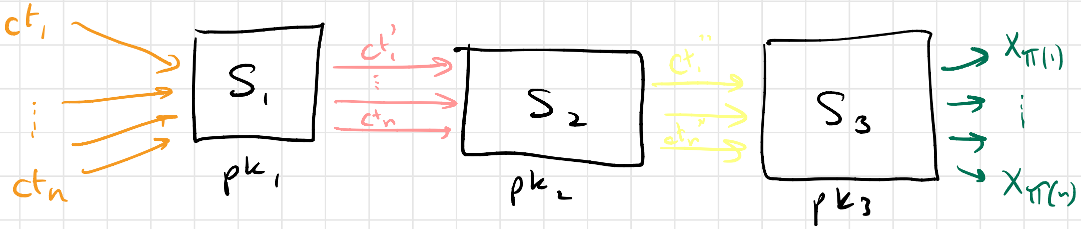
Mix-nets

Another idea of David Chaum's that has been very influential in the world of privacy-protecting systems.

... As before, each player i has message $x_i \in \{0,1\}^\lambda$. Want to learn $\{x_1, \ldots, x_n\}$ in shuffled order.

Difference: Will delegate the work to k servers. (Can also have each user be a server but this is annoying in practice.)

[Some form of security holds if ≥ 1 server honest.]



Idea: Each player i threshold encrypts her message x_i to the three servers

$$ct_i = E(pk_1, E(pk_2, E(pk_3, x_i)))$$

- Each server shuffles and decrypts and passes to the next server.
- Output messages are shuffled according to a permutation that no one server knows.

This is clever!?

↳ No crazy crypto tools... just standard PKE.

	Total Comp	Per-user com	Security
Mix-net	≈ n PK ops	1 ct	computational
Dc-net	≈ n^2 field ops	≈ n bits	inf. theoretic

But Beware...

- Security guarantees you get here are messy.
- Plain scheme is only semi-honest secure...

Ex. active attack: Server 1 drops all msgs except Alice's.

- Practically annoying: Can only mix in Batches. Doesn't achieve anything if you mix 1 msg at a time.

One way to handle active attacks is with ZK proofs... every mix server proves to others that it executed the decrypt-and-shuffle op correctly... "verifiable shuffles"

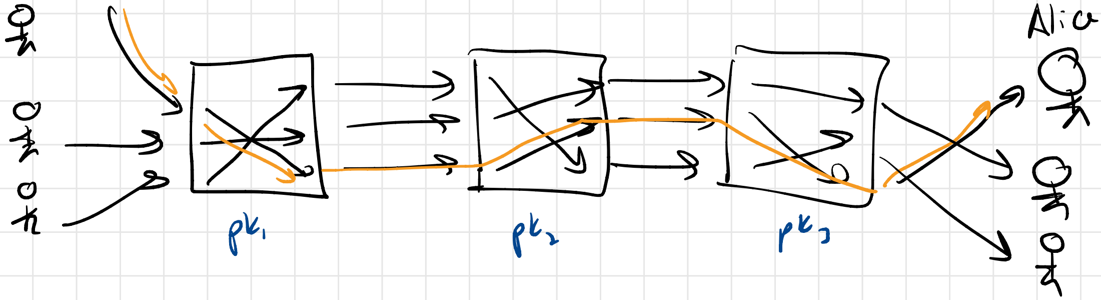
↳ Requires mixes to agree on input

↳ Doesn't change asymptotic cost, but concretely more expensive

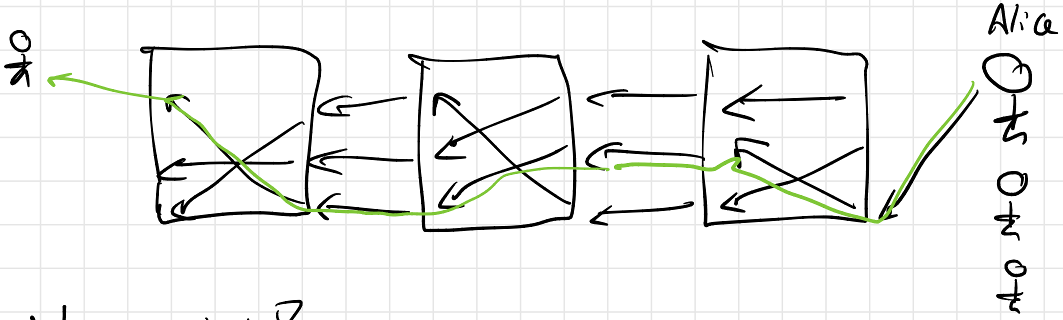
Another idea of Chaum ...

Say that you use a mix-net to send a msg to Alice ... How can she reply to you?

$$ct_1 = E(pk_1, E(pk_2, E(pk_3, E(pk_{Alice}, pk_{Anon} \| \text{msg for alice}))) \ldots)$$



When Alice wants to reply, send msgs backwards through the mix net



Very slick!

Mix-nets

- Why not used?

  Were some mix-based "remailers" active for a while ... mixmaster, mixminion

- One problem: Mix-based systems tend to have either

  High latency = Wait for most users to submit a msg before mixing

  or

  Poor Security = Don't wait for most users to submit msg before mixing

⇒ If adv can observe all net trafic, it's not clear what we can do. ☹

Possible route ahead: Consider weaker advs but that are still realistic & powerful

↑ I haven't seen a nice clean model of work that does this... let me know if you have.

Tor ("onion routing")

[Dingledine, Mathewson, Syverson 2002]

With DC-nets and mix-nets, can get precise notions of security against network adv under certain conditions.

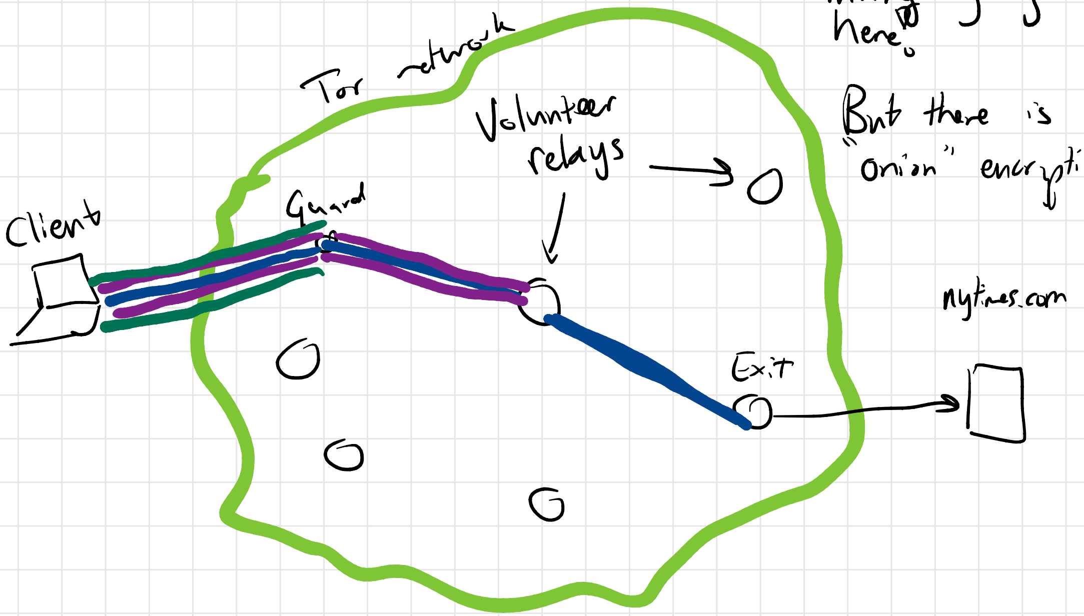
...but have severe practical limitations we've seen

Tor's approach is to sacrifice precise security guarantees (since it's not clear that real-world mix- or DC-net-based systems enjoy these anyways) in favor of practical usefulness

Very simplified...

There is no mixing going on here!

But there is "onion" encryption.



Tor

- Tor offers low-latency browsing
  - ↳ No need to wait for other users to show up... You sent traffic through network as quickly as you like
  - → Dif users can send traffic at diff rates (unlike mix net)
- Tor is used at large scale... thousands of relays. Millions (?) of users daily
  ≈ 300 Gb/s throughput in total
- Backed by 501(c)(3)

Problem: Not clear what security properties Tor provides... Somewhat unsatisfying but maybe "good enough" for important use cases.

Problem: For maximum privacy/deniability, everyone should use Tor... still a niche product

↳ Concern is that using Tor (in certain countries) could make you a target

We saw two theoretical approaches to online anonymity and one pragmatic one.

↳ I'm an optimist, but I tend to believe that there are even better solutions to these problems out there...

...will just take one of you having a good idea!