Lecture 24: Zero Knowledge & Schnorr Signatures

Fall 2023 - 6.1600

MIT

C-G & Zeldovich

Plan

* Zero knowledge
* Schnorr's protocol for dlog
* Fiat-Shamir heuristic
* Schnorr signatures

This week, we are talking about privacy.

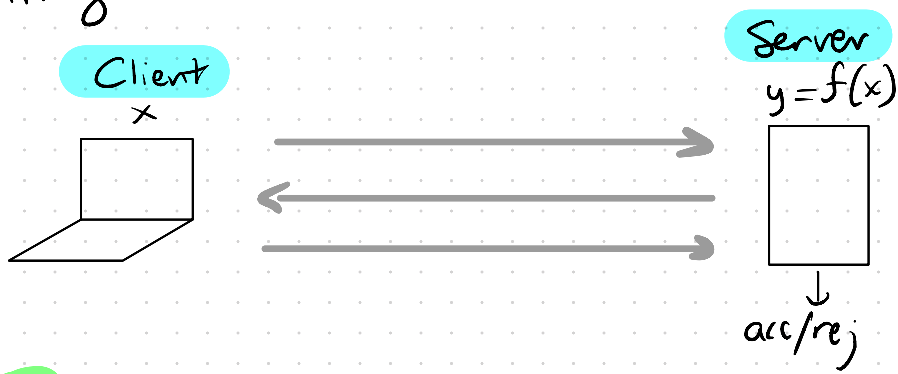
Today: How to prove something to someone (that a fact is true or that you "know" some secret thing) while leaking nothing else about what you know.
["Zero-knowledge proofs"]

* Useful in all sorts of privacy-protecting crypto protocols
* Also useful for building sig schemes.

Wednesday: When you must leak some sensitive information (e.g. US Census), how do you reason about how bad the leakage is?
["Differential privacy"]

(Remember: Always better to leak nothing!)

Setting



1. Server convinced that client "knows" x' s.t. $y = f(x')$

2. Server "learns nothing" about x from interaction.

Two questions

1. What does it mean to be convinced that a computer "knows something?"
   - Convinced that it's stored in memory?
   - Convinced that it could print it out?

2. What does it mean for a computer to "learn nothing?"
   - Never sent x?

One of the triumphs of modern crypto has been to give precise & satisfying answers to both of these questions.

Surprise: * Everything statement with a short (poly sized) proof can be proved in ZK.

GMR'85, GMW'86

↳ In theory & if OWF exist

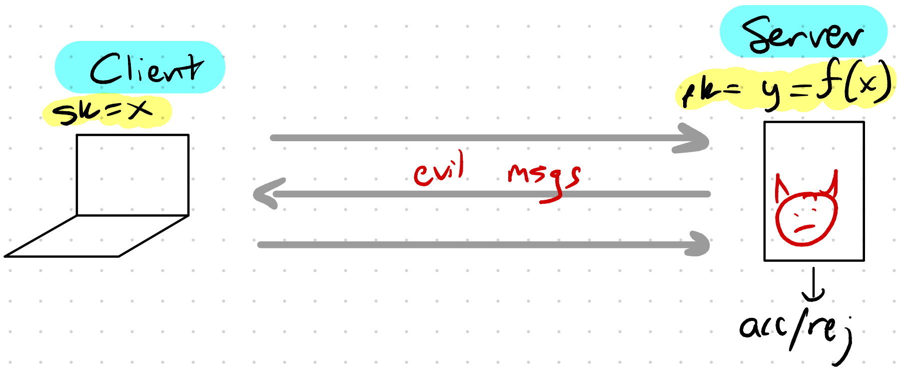
* Concretely f can be any eff fn!

↳ Actively secure auth from any OWF

Examples

* Prove know factors of int w/o leaking them
* " " 3 coloring of Graph " "
* " " SAT assignment to formula " "
* collision in hash fn.
* crashing bug in program*

ZK proofs give a simple & beautiful way to construct actively secure auth protocols from any OWF.



To auth, client proves knowledge of x in ZK.

ZK ⇒ Even if attacker compromises server and gets y and interacts w/ client, learns nothing about x ⇒ Can't impersonate client!

↳ At end, we will see how to "compile" such protocols into digital sig schemes

* Schnorr, EdDSA, ECDSA essentially all use variants of this strategy
* Basis for sigs used everywhere

* We'll see Schnorr's ZK protocol
* Concretely efficient way to prove knowledge of dlog in ZK.

Reminder: **Discrete-log problem**

256-bit prime

$G = \{g, g^2, g^3, g^4, \ldots, g^q\}$ "order q"

and group operation $*: G \times G \to G$

* G can be subset of ints mod p with multiplication
* G " " group of points on elliptic curve mod p

Dlog Assumption for group $G = \{g, g^2, \ldots, g^q\}$

For all "eff" algs A,

$$\Pr\left[A(g, g^x) = x : x \xleftarrow{R} \mathbb{Z}_q\right] \leq \text{"negl"}$$

Let's look at Schnorr's protocol first and then try to figure out defn's for

1) "knowing something"
2) "Learning nothing"

$P(x \in \mathbb{Z}_q)$ — **$V(X \in \mathbb{G})$** (Should be g^x)

$r \xleftarrow{R} \mathbb{Z}$

$R = g^r \in \mathbb{G}$ →

$c \xleftarrow{R} \{0,1\}$

← $c \in \{0,1\}$ ("challenge")

if $c=0$: $z = r$

$c=1$: $z = r + x \in \mathbb{Z}_q$

$(z \leftarrow r + cx \in \mathbb{Z}_q)$

$z \in \mathbb{Z}_q$ →

Accept iff $g^z = R \cdot X^c \in \mathbb{G}$

A few points:

* Verifier's message is independent of P's first msg.
* The interaction here is crucial for security! (Inherent)

Notice that w.p 1/2 cheating P^* can cause V to accept

↳ P^* doesn't "know" dlog of X.

Cheating P^*

- Guess $\hat{c} \xleftarrow{R} \{0,1\}$
- If $\hat{c} = 0$: Follow protocol
  ↳ Reveal r. (If $c=1$, get stuck)
- If $\hat{c} = 1$: Pick $z \xleftarrow{R} \mathbb{Z}_q$
  Set $R \leftarrow g^z \cdot R^{-1} \in \mathbb{G}$
  ↳ Reveal z (If $c=0$, get stuck)

To reduce prob of accepting cheating P^*, run protocol λ times in sequence $\Rightarrow 2^{-\lambda}$ prob of cheating.

Complete: Honest P convinces honest V. ✓

Soundness/knowledge: Dishonest P^* can't convince honest V.

Idea: We say that P^* "knows" dlog of X if there's an eff alg that "extracts" dlog from P^*.

(P, V) is sound if $\exists$ eff alg Ext ("extractor") s.t. $\forall P^*$
[that convince V w.p 1, for simplicity]

$$\Pr[\mathrm{Ext}(P^*(g^x)) = x : x \xleftarrow{r} \mathbb{Z}_q] \geq 1/2.$$

] or $1 - 2^{-\lambda}$ if you want to be really sure that P^* can't cheat

For Schnorr:

$\mathrm{Ext}(P^*)$:

* Run P^* to get tx $(R, c=0, z)$
* Rewind P^*, run again with same R $(R, c=1, z')$

$g^z = R \qquad g^{z'} = R \cdot X$

$g^{z'-z} = X$

$\Rightarrow x = z' - z \in \mathbb{Z}_q$

Zero knowledge:

Formalization of idea that V shouldn't "learn anything" from interaction.

Principle: V has learned nothing from an interaction with P if V could sit at home (with no interaction with P) and write down a transcript of V's interaction with P that is indist from the true one.

Many real-world examples: "No comment", me as teen, interviews, etc.

"simulator"

(P,V) is ZK if ∃ eff alg Sim s.t. ∀ eff advs V^*

$$\{ Sim(V^*) \} \overset{c}{\approx} \{ \text{transcript of } P(x) \leftrightarrow V^*(g^x) : x \leftarrow \mathbb{Z}_q \}$$

Sim for Schnorr:

$Sim(V^*)$:

* Run strategy of cheating P^*
* If guessed c wrong, retry

Output $(R, c = V^*(R), z)$

Succeed w.p. ½ on each try.

Seems like Ext & Sim are in conflict!

Ext ⇒ V can get dlog from P

Sim ⇒ V can't learn anything from P

Resolution: * Ext has more power than V does in the "live" protocol...

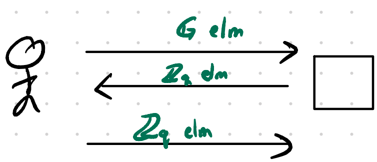
* Ext can rewind P — in reality cannot

Many subtleties: Just because a protocol is ZK doesn't mean it is when P runs many instances of it in parallel!

In practice, we use Schnorr where challenge is random in $\mathbb{Z}_q$ rather than $\{0,1\}$

+ Evil P^* can cheat w.p. $\approx \frac{1}{q}$. No need to repeat!
- Resulting scheme has a weaker form of ZK "honest-verifier" ZK.

With this tweak, Schnorr's ID protocol requires very little communication



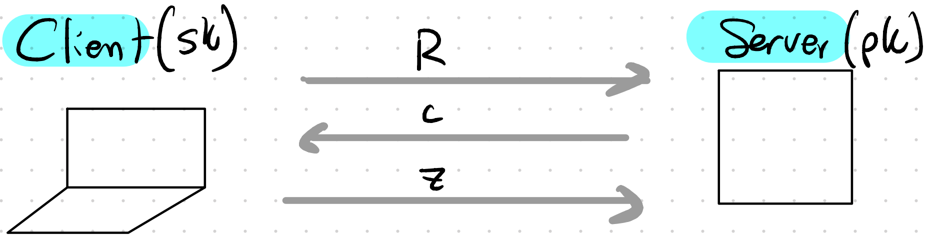
When using EC group

$|G \text{ elm}| = |\mathbb{Z}_q^* \text{ elm}| = 256$ bits

$= 96$ bytes

(vs. at least 384 for RSA sig with 128-bit sec)

Let's take stock of what we've accomplished...



+ Complete, sound (if dlog is hard), zero knowledge
  ↳ malicious-secure ID protocol
+ Small communication!

Surprise: We can make this protocol non-interactive (i.e. one msg P → V)

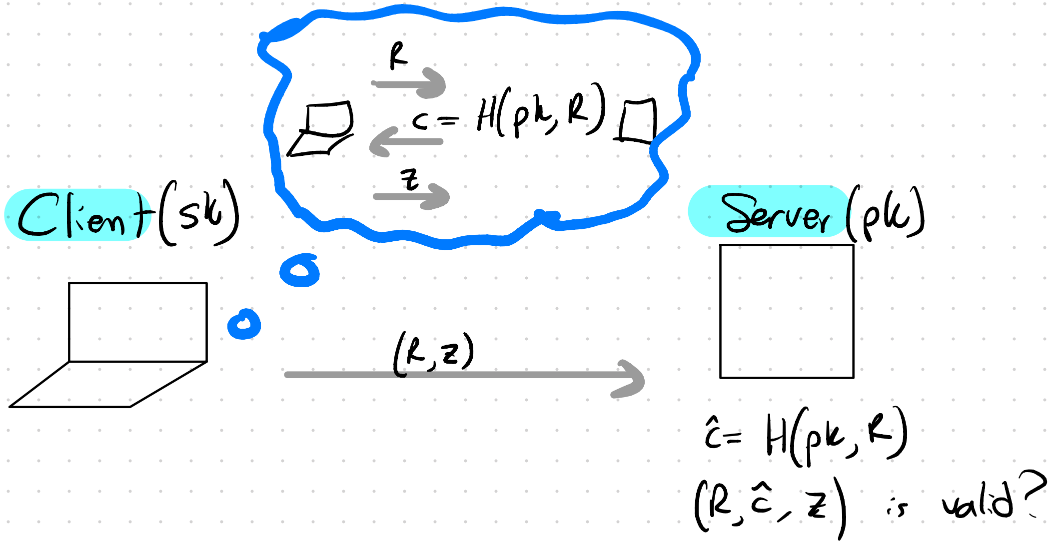
Observation: Verifier's challenge c is just a public random value.

Idea: Replace V with a hash fn (e.g. SHA2)!

Didn't we just say that interaction is crucial for security?!

Fiat - Shamir

* P imagines running ZK protocol with V replaced by hash
* Send transcript to V
* V checks transcript is valid



Not clear that this protocol is ZK any more...
BUT, is if we model hash fn H as a random oracle

What makes N.I. possible

Last step: Convert this ID protocol into a signature scheme...

Sig Scheme from F-S: Schnorr Sigs

Gen() $\longrightarrow (sk, pk) = (x, g^x)$ $\quad x \xleftarrow{R} \mathbb{Z}_q$

Sign(sk, m) $\longrightarrow \sigma$

- Run FS version of Schnorr ZK protocol with $H(m, \cdot)$ as hash function
- Output t_x as sig σ

Verify(pk, m, σ) $\longrightarrow \{0, 1\}$

- Accept is Schnorr F-S verifier accepts using $H(m, \cdot)$ as hash fn.

Can show that this is a secure sig scheme under dlog assump, provided that we model H as random oracle.

Since ZK proof has small comm & comp costs
$\Rightarrow$ Sigs are short and pretty fast!
(512 bits) (essentially one exp in $\mathbb{G}$)

What to take away:

ZK gives a way to prove knowledge/correctness while leaking the minimum possible info.

↳ See 6.875 for much more!

Next time:

How to deal with leakage of sensitive info when it's inevitable.