Logistics
- Problem set 1: graded by Thursday (drop deadline Friday)
  [average time: 15-25 hours] (355=255+100)
  [target time: 10-15 hours]
  [favorite problem: 2/3]
  [least fav: Problem 4]
  ⇒ Please submit feedback

- Solutions will be on Piazza
- HW 2 out now
  ⇒ should be somewhat easier
- Ask if unclear/under-specified/
  wrong? We try to be correct
  but still often? 😕

Plan
- Recap: Roots of Knowledge
- Non-interactive zk
- Fiat-Shamir heuristic
  ⇒ Idea
- Applications
  * Schnorr sig. and DSA/
  * (??) protocol
Recap: Proofs of Knowledge
A language \( L \) is in \( \text{NP} \) if there exists a relation \( R \) such that:
\[ x \in L \iff \exists w \in \{0,1\}^{poly(|x|)} \text{ s.t. } R(x,w) = 1 \]
x = "instance" or "statement"
w = "witness"

Example:
- \( \phi \) is SAT formula
- \( \overline{a} \) is assignment to vars
\[ R(\phi, \overline{a}) = \begin{cases} 1 & \text{if } \phi(\overline{a}) \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases} \]

Intuition behind proof vs. PoK

**Proof**

"something is true"

**PoK**

"I know why something is true"

- \( N \) is product of two primes
- \( \phi \in \text{SAT} \)
- \( x \in L(R) \)

A ZK proof for \( \text{NP} \)-complete language \( L(R) \): a PoK it has

1. Completeness
2. Knowledge
3. ZaK

Idea:
- If (possibly malicious) \( \mathbf{P}^* \) convinces honest \( \mathbf{V} \) that \( x \in L(R) \) is \( \frac{1}{2} \), then exist \( \mathbf{P}^* \) extractor \( E \) s.t.
- \[ E[ U \leftarrow \mathbf{P}^* ; R(x,w) = 1 ] \geq \frac{1}{2} - \varepsilon \]

\( \Rightarrow \) By interrogating \( \mathbf{P}^* \) can extract a witness from it
\( \Rightarrow \) To convince \( \mathbf{V} \), \( \mathbf{P}^* \) must be able to produce \( w \)

Extractor can "revert" \( \mathbf{P}^* \)

Why? \( \mathbf{P}^* \) is just a program... can run and re-run it

In real interaction, cannot re-run \( \mathbf{P}^* \) \( \Rightarrow \) Can't break ZK

Fact that time goes forward is what makes ZK possible

Q: If you had a time machine, could you break ZK?

\[ \Rightarrow \text{Aaronson and Watrous paper on CTC, VM real attacks...} \]
Non-Interactive ZK

(See recent paper from Sam & David)

So far, we have looked at interactive protocols for NP relations $R(x,w)$ $P(x,w)$ $V(x)$

$\Rightarrow$ We saw that these IPs can provide ZK — $V$ learns only that $x \in L$

In practice, it would be nice to have non-interactive ZK proofs of knowledge.

$G$ is group in which $e^{log(g)}$ $P(x \notin L \leq G)$ $V(A \in G)$ $V(A \in \hat{G})$

"Prover knows $x \in L$ such that $A = g^x$" "One-shot"

"one-shot"

Looks out, such ZK protocols only exist for "easy" (BPP) languages, in the standard model. (Why?)

When faced with impossibility, what do we do?

1) Introduce crazy assumptions
2) Change the model

CRS model

Random-Oracle Model

(Like on exponentially long CRS)
A Good Idea: Fiat-Shamir Heuristic

Convert Sigma Protocol for language $L(R)$ in R.O.M.

Recall Schnorr’s protocol for proving knowledge of $\log A = \beta \mod q$.

$$P(\pi \in \mathbb{Z}_q, A = g^\pi) \quad V(\pi \in \mathbb{Z}_q)$$

Check $g^\pi = R \cdot A^\pi \in \mathbb{G}$

$acc/rej \not\in$ Prover "knows" $\pi$

Has:
1) Completeness
2) Knowledge
3) HIZkic

Notice that $V(\pi)$ sends only random value to $P \notin$ "Public coin"

Idea of Fiat-Shamir: *Replace $V$ with Hash $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$

$$H(A, R) \in \mathbb{Z}_q$$

$P \Rightarrow V$

$$\pi = (R, H(A, R), \varepsilon) \Rightarrow acc/rej$$
Fat-Shamir & Schnorr

We need to show (informally)

1. Completeness  
2. Knowledge  
3. HVZK  

Use “rewinding” (Bench-Sharp 7.1-19.2)

Idea: * Run \( P^* \) and get \((R, c, z)\)
* Rewind \( P^* \) and run again
* When \( P^* \) queries \( R_0 \), give different answer

\( L_0 = (R, c', z') \)
* Can solve for \( \text{dlog} \alpha \) using linear algebra

Some technical subtleties

Schorr Signature Scheme \((G \text{ is group of prime order } q, \text{ dlog hard})\)

\( \text{Gen}(1) \rightarrow \begin{cases} q \alpha \leftarrow \mathbb{Z}_q^* \\ s_k = \alpha, \ pk = q \alpha \in G \end{cases} \)

\( \text{Sign}(s_k, m) \) \begin{cases} r \leftarrow \mathbb{Z}_q^* \\ R = q^r \in G \\ c \leftarrow H(pk, R, m) \\ z \leftarrow \alpha r + c \alpha \in \mathbb{Z}_q^* \end{cases} \text{ output} \((R, c, z)\)  \text{ Can compress to send only } (c, z)

\( \text{Verify}(pk, (R, c, z), m) \) \begin{cases} c = H(pk, R, m) \\ R = q^c \in G \end{cases} \)

Proof of security is not immediate…… see book

Implementation note: Knowledge error is \( Y_{(c, c', z, R)} \)
* Use 128-bit challenge in practice, while \( q \approx 256 \) bits
Schnorr

Idea: Take an interactive zk protocol for knowledge of discrete logs

1. Compile into NIZK using ROM
2. Stick msg into ROM to make it a sig scheme.

In theory, can use any NP language as basis for a sig scheme in this way

- Graph isomorphism, 3-Col, factoring, ...
- Why not NP-complete problem? e.g. 3SAT?

Schnorr sigs have beautiful theoretical basis

In real life, we all use DSA/ECDSA (why idea, but widespread)

Why? Patents! (expired 2008)

Practical Note: Implementors beware!

The prover/signer in Schnorr signatures is randomized.

⇒ Indeed, prover in all zk protocols is randomized, unless language L in question is BPP. (Why?)

Prover must use fresh randomness for each signature?

What happens if not?

\[(g^r, c = H(A, R, m), z = r + ca)\]
\[(g^{r'}, c' = H(A, R, m'), z' = r' + c'a)\]

\[z - z' = (c - c')a \in \mathbb{Z}_q\]
\[\frac{z - z'}{c - c'} = a \in \mathbb{Z}_q\]

⇒ Get signer’s secret key!

⇒ This has lead to $2^{2^n}$ practical attacks!

Playstation 3 hack, Detorn theft, ...

⇒ Many attacks even if random choice $r$ is somewhat biased
Deterministic Schnorr

To avoid attacks, can sign using PRF \( F(k, m) \rightarrow \mathbb{Z}^q \)

- secret key = \((x, k)\)
- \( r < F(k, m) \) ← Derive secret "random" value using \( F \).
  \( r \leftarrow g^r \in \mathbb{G} \)

Practical Note:
ECDSA signatures are used everywhere

\((c, z)\)

- integer in \( \mathbb{Z}^q \) (384 bits for 128-bit security)
- 128-bit challenge

RSA-FDH signature ≈ 3072 bits