I) Zero knowledge proofs

(Goldwasser, Micali, Rackoff 1982)

Excellent technical introduction: chapter 13 in Boaz Barak's cryptography book

Some key concepts

- Proof system
- Interactive proof
- Roles: Prover (P), Verifier (V)
- Desirable properties
  - Completeness - everything works if everyone is honest & behaves
  - Soundness ($\Leftarrow$ knowledge soundness) - a bad P will fail
  - Zero knowledge - V obtains no add'l info besides correctness

Example I (Prover) want to convince you (Verifier) that I can distinguish two colors that you see as identical

Prover | **Verifier**

Here are two items A & B different colors →

(V sees A & B as identical)
flips a coin secretly,
if head, swap A & B without P looking

← challenge: here are the two items, Did I swap them?

answers the challenge →

checks P's answer

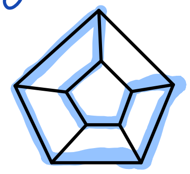
(repeat 100 times)

Hamilton cycle [Blum '87]

Common knowledge : a graph G

P wants to prove to V that he knows a Hamilton cycle of G without revealing any additional information

(Hamilton cycle : a cycle going through every vertex of the graph exactly once & returns to the start



)

P V

- Label vtx 1, .., n according to some privately randomly chosen permutation (remember this permutation)
- Btwn every pair ij of vtx, put in a locked box B_{ij} the bit indicating whether ij is an edge of G

$\xrightarrow{\text{all } \binom{n}{2} \text{ locked boxes } B_{ij}}$

$b \leftarrow_R \{0,1\}$

challenge $\begin{cases} \text{if } b=0: \text{ show me the graph} \\ \text{if } b=1: \text{ show me the Ham cycle} \end{cases}$ ←

answer $\begin{cases} \text{if } b=0: \text{ unlock all the boxes \& vtx permutation} \\ \text{if } b=1: \text{ unlock the boxes of the Ham cycle} \end{cases}$ →

check:
$\begin{cases} \text{if } b=0: \text{ same graph} \\ \text{if } b=1: \text{ Ham cycle} \end{cases}$

"sigma protocal"

P $\xrightarrow{\text{commitment}}$ V

$\xleftarrow{\text{challenge}}$

$\xrightarrow{\text{response}}$

Commitments

physical (locked box)

digital (cryptographic)

P

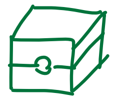
V

secret data s

lock data in a box

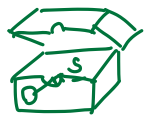
$r \leftarrow_R \{0,1\}^{256}$

$h \leftarrow \text{hash}(s,r)$



$\xrightarrow{\quad h \quad}$

$\xleftarrow{\text{open it}}$



$\xrightarrow{\quad s,\ r \quad}$

check $h = \text{hash}(s,r)$

Properties

Completeness If everyone behaves, then protocol accepts

Soundness If there is no Ham cycle, then no matter what P does, V rejects with prob $\geq \frac{1}{2}$

(There is a stronger requirement called "knowledge soundness" which says that even if the graph has a Ham but the prover doesn't "know" it, the protocol would still fail. The precise definition involves an extractor with rewinding abilities...)

Zero-knowledge If V accepts, then it learns no add'l info from the interaction, because V could have <u>simulated</u> the entire dialog by itself

Universal ZKP

Hamiltonicity is NP-complete

(every NP problem can be polynomially reduced to HAM)

P

Common: x, f

V

I know s s.t. $f(x,s) = 1$

P knows s : $f(x,s)=1$

$\Downarrow$ NP reduction

$\Uparrow$

I know a Ham cycle in graph G

$\xrightarrow{G,\ \text{ZKP of HAM}}$

$\longleftarrow$

$\longrightarrow$

Convinced that P knows Ham of G

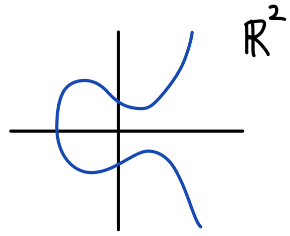
__Remark__ Can turn an interactive proof into a non interactive proof via the Fiat-Shamir heuristic

P can simulate V : whenever V picks a random value, P can simulate V's randomness by running a cryptographic hash function on the transcript (so that P can't cheat by choosing favorable "random" challenges)

Above construction of universal ZKP is not succinct (also not practical). Later in the course: zk-SNARK

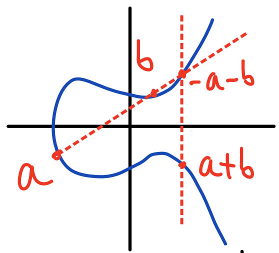
II) Elliptic curves

$y^2 = x^3 + Ax + B$



In cryptography, work over $\mathbb{F}_p$ instead of $\mathbb{R}$

Group law: addition of points on elliptic curve



• O = pt at infinity

$\rightsquigarrow$ abelian (commutative) group G whose elements are points on the curve

We write group operation additively $(G, +)$

Fix a generator $g \in G$ of prime order $q \approx 2^{256}$

In practice, (G, g) chosen from some standard (e.g. NIST)

Discrete log: secret $x \leftarrow_R \mathbb{Z}_q$, public $xg \in G$

recovering x from xg is hard "Discrete Log Assumption (DLA)"

Diffie-Hellman Key exchange

Alice		Bob
$\alpha \leftarrow_R \mathbb{Z}_q$	αg $\quad$ βg	$\beta \leftarrow_R \mathbb{Z}_q$

shared secret: $\alpha\beta g$ (hard to compute given αg, βg)

"Computional Diffie-Hellman (CDH) assumption"

Also "Decisional Diffie Hellman (DDH) assumption": hard to distinguish

$(\alpha g, \beta g, \alpha\beta g)$ $\alpha, \beta \leftarrow_R \mathbb{Z}_q$ vs. $(\alpha g, \beta g, \gamma g)$ $\alpha, \beta, \gamma \leftarrow_R \mathbb{Z}_q$

Schnorr protocol

P wants to convince V that
P knows a secret $s \in \mathbb{Z}_q$ s.t. $\underbrace{x = sg}_{\text{public}} \in \mathbb{G}$

$$
\begin{array}{lcl}
\textbf{P} & & \textbf{V} \\
s \in \mathbb{Z}_q,\ x = sg \in \mathbb{G} & & x \in \mathbb{G} \\
r \xleftarrow{R} \mathbb{Z}_q & \xrightarrow{\quad u = rg \quad} & \\
& \xleftarrow{\quad c \quad} & c \leftarrow \mathbb{Z}_q \\
& \xrightarrow{\quad z = r + cs \quad} & \text{check } zg = u + cx
\end{array}
$$

<u>Claim</u> The above is a ZKP of discrete log.
(proof omitted)

Remarks

(I) Can be turned into a NIZK (noninteractive) via Fiat-Shamir, by setting $c = \text{Hash}(x, u)$

(II) Can be furthermore turned into a digital signature scheme
To sign message m, set $c = \text{Hash}(x, u, m)$ and publishing (c, z) as sig for m
(secret key $= s$, public key $= sg = x$)

Pairing based cryptography

Given cyclic groups $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ all same prime order q,

a pairing is a nondegenerate bilinear map

$$e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$$

(A) bilinear : $e(x+x', y) = e(x,y) + e(x', y)$
& $e(x, y+y') = e(x,y) + e(x, y')$ $\forall x, x', y, y'$
($\Rightarrow e(ax, y) = a\,e(x,y) = e(x, ay)$ $\forall a \in \mathbb{Z}_q$)

(B) nondegenerate : with generators $g_0 \in \mathbb{G}_0$ & $g_1 \in \mathbb{G}_1$
$g_T := e(g_0, g_1) \in \mathbb{G}_T$ is a generator

Certain elliptic curves (Weil, Tate, ...) have useful pairings
- efficiently computable
- cryptographic hardness assumptions

An application : BLS signature scheme (Boneh-Lynn-Shacham)

Keygen : $sk := s \leftarrow_R \mathbb{Z}_q$ $\quad pk := s g_0 \in \mathbb{G}_0$

Sign $(sk, m) \to \sigma := s\, H(m) \in \mathbb{G}_1$ $\quad H(m) \in \mathbb{G}_1$ hash of message

Verify $(pk, m, \sigma) \to$ check $e(pk, H(m)) \stackrel{?}{=} e(g_0, \sigma)$

- Can be extended to allow signature aggregation