ZK Proofs IV

Sunday, September 17, 2017 12:04 PM

Non-interactive ZK Proofs

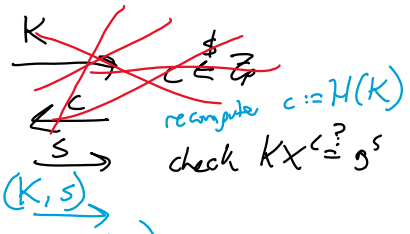
$ZkPoK\{(x): g^x\}$

$k \xleftarrow{\$} Z_p$

$K := g^k$

$c := H(K)$

$s := k - cx$



Hash function (e.g. SHA2)

Modeled hash function as a random oracle.

Random Oracle

randomOracle(1^λ):
  initialize table := {} (table : $\{0,1\}^* \to \{0,1\}^\lambda$)
  on query($m \in \{0,1\}^*$):
    if $m \notin$ table:
      $h \xleftarrow{\$} \{0,1\}^\lambda$
      table[m] := h
    return table[m]

- Truly random mapping from strings to λ-bit strings.
- There are an infinite # of collisions.
  (x, x'), $x \neq x'$, $H(x) = H(x')$
- In practice we prove a protocol secure in RO model, but replace w/ concrete function like SHA.
- Paradox; we know of a protocol that is secure in RO model, but insecure when instantiated w/ any concrete function.

Proof that Schnorr w/ RO is secure:

$ZkPoK\{(x): g^x = X\}$

- Simulatability: $x \xleftarrow{\$} Z_p$

$$View_V\left[P^{(\cdot)}(x) \leftrightarrow V^{H(\cdot)}(g^x)\right] \approx Sim(1^\lambda, X)$$

Note: oracle access

$M^{H}(1^\lambda, \ldots)$ means M has oracle access to H
- at most poly(λ) oracle queries

- Correctness RO

$$\forall x, \Pr\left[P(x) \leftrightarrow V(g^x)\right] = 1$$

$H \leftarrow RO()$

$P^H(x) \longleftrightarrow V^H(g^x)$

- Simulatability: $x \xleftarrow{\$} \mathbb{Z}_p$

$$View_V\left[P^H \longleftrightarrow V^H(g^x)\right] \approx Sim(1^\lambda, X)$$

Main idea:

Simulator "programs" the random oracle

$View_V \rightarrow (K, s, \{(K_1, h_1), (K_2, h_2), \ldots\})$

($K_1, K_2, \ldots$: queries made to RO; $h_1, h_2, \ldots$: responses from RO.)

Simulator:

$Sim(1^\lambda, X)$:

$s \xleftarrow{\$} \mathbb{Z}_p$

$c \xleftarrow{\$} \mathbb{Z}_p$

$K := g^s / X^c$

return $(K, s, \{(K, c)\})$

(K: guess; c: challenge)

Distribution:

given X, all s and c equally likely;

given X, s, c, one choice of K

- Extractability

New proof technique, "forking lemma"

$\exists \mathcal{E}_A$.

$$\frac{Pr\left[Out_V\left[A^{RO}(1^\lambda, X) \longleftrightarrow V^{RO}(X)\right] = 1\right] = \rho}{\rho > negl(\lambda)}$$

$$\Rightarrow Pr\left[x' \leftarrow \mathcal{E}_A(1^\lambda, X) : g^x = g^{x'}\right] > 1 - negl(\lambda)$$

Main idea:

Run $\mathcal{A}$ multiple times, giving it different RO responses.

Defining $\mathcal{E}_A$:

Let $q(\lambda)$ be a polynomial bound on #oracle queries from $\mathcal{A}$.

Sample $\{h_i \xleftarrow{\$} \{0,1\}^\lambda\}$ $\forall i \in [0, q(\lambda)]$

Let ρ be a stream of random bits.

$(K_1, s_1) \leftarrow \mathcal{A}^{H_1}(1^\lambda, X; \rho)$ (run with ρ as the random bits made by $\mathcal{A}$.)

Where:

H_1 returns h_i for the i'th query,

and stores $K_{1,1}, K_{1,2}, \ldots$ as queries.

Look for index I such that $K_{1,I} = K_{1,I}$ and $\left(K_{1,I} \overset{h_I}{=} g^{s_1}\right)$

(Break Point-1 pass w/p) otherwise return $\perp$

with probability approx ρ, we pass BP_1.

Now sample $\{ h'_I \xleftarrow{\$} \{0,1\}^\lambda \}$ $\forall I \le i \le q$
$h'_{I+1} \xleftarrow{\$} \{0,1\}^\lambda$
...

$(K_2, s_2) \leftarrow A^{H_2}(1^\lambda, X; \rho)$ — same random coins

where H_2 returns $\{h_1, h_2, \ldots h_{I-1}, h'_I, h'_{I+1}, \ldots h'_q\}$
and stores $\{K_{2,1}, K_{2,2}, \ldots\}$ as queries.

$h_1, h_2, \ldots, h_{I-1}, h_I, h_{I+1}, \ldots h_q$ ← run 1
$h_1, h_2, \ldots, h_{I-1}, h'_I, h_{I+1}, \ldots h'_q$ ← run 2

$K_{2,i} = K_{1,i}$ for $i \le I$

BP2 If $K_1 \neq K_2$ then return $\perp$

BP3 otherwise if $g^{s_2} \neq X^{h'_I} K_1$ then return $\perp$

otherwise: $g^{s_1} / X^{h_I} = K_1 = K_2 = g^{s_2} / X^{h'_I}$

assume w.l.o.g. $h_I \neq h'_I$

Solve $x = (s_1 - s_2)/(h'_I - h_I)$ s.t. $g^x = X$ then output x

Let $frk = \Pr[x \leftarrow C_A(1^\lambda; X) : X = g^x]$

We want $frk = \rho \cdot \left(\frac{1}{q}\rho\right) - \frac{1}{2^\pi} \geq negl(\lambda)$

- ρ: probability of passing BP_1
- $\frac{1}{q}\rho$: prob. of choosing same index I for output (BP3)
- $\frac{1}{q}\rho$: prob of passing BP_2
- $\frac{1}{2^\pi}$: naive prob of $h_I = h'_I$

This is the Forking Lemma:
(Lemma 3.1 of Bellare, Neven '06)
Uploaded on website

Fiat-Shamir Heuristic:
Replace Verifier in HVZK 3-round protocol with RO.

Schnorr Signature
Idea: include message to sign in input RO

Sign (x, m):
$k \xleftarrow{\$} \mathbb{Z}_p$
$K := g^k$
$c \leftarrow H(K, g^x, m)$ — public key, msg to sign

Note: "How not Prove Yourself"

$s \leftarrow \{0,1\}^n$

return $\sigma := (K, s)$

Verify $(X, m, \sigma = (K, s))$:

$c \leftarrow H(K, X, m)$

check $g^s \stackrel{?}{=} X^c K$

— unforgeable

informally, cannot create new signatures on m^* even after seeing signatures on $m_1, m_2, \ldots \neq m^*$

Wrapping up ZK:

— ZK for all NP languages.

3COL ← NP complete

$ZK\{(x) : P(x)\}$ ← any Turing machine

— Component of larger protocols

consistency checks

— Later: zkSNARKs

Efficient proof (doesn't depend on size of predicate or witness)