Zero Knowledge Proofs

Lecture 21

DNSSEC

- Recall: Name servers, when queried with a domain name, return an IP address record (signed by the zone owner), or report that no such domain name exists
- Question: How to prove that an entry is missing, without revealing anything else?
  - NSEC: Have adjacent pairs (in sorted order of domain names) signed together. Return a pair flanking the queried name.
    - Reveals the adjacent domains. Allows zone enumeration.
  - NSEC3: Use H(domain-name) in this proof.
    - Still allows offline enumeration (domain names have low-entropy)
- A recent proposal: NSEC5

DNSSEC

- A recent proposal: NSEC5
  - Using "Verifiable Random Functions" (VRF)
- VRF is a PRF, with an additional public-key (SK & PK generated honestly)
  - Remains pseudorandom even given public-key
  - SK allows one to give a proof that $F_{SK}(x) = y$, without revealing SK. Proof can be verified using a PK.
    - A **Zero-Knowledge proof**!
  - NSEC5 proposes a Random Oracle based VRF (assuming hardness of Discrete Log)

DNSSEC

- Using a VRF to protect against zone-enumeration
- Instead of H(domain name), use F_{SK}(domain name)
  - For a missing entry for a query Q, return:
    - Y, and a VRF proof that $F_{SK}(Q) = Y$
    - A pair of consecutive entries (Y_1, Y_2), signed by zone-owner, such that $Y_1 < Y < Y_2$
- Name server needs the VRF key SK (generated by the zone-owner) to compute $F_{SK}(Q)$ and the proof. But does not have access to the signing key.
- Adversary querying an honest name server learns the presence/absence of an entry (and and an upper bound on the total number of entries)
- Corrupt name server learns all entries, and can also refuse to answer queries, but it cannot give a wrong response

VRF

- How to build a VRF?
  - Original construction from [MRV'99]
    - Required PRF security even for PK generated by the adversary
  - Constructions from RSA and "bilinear pairings"
- NSEC5 uses another VRF based on the discrete log assumption, but in the random oracle model
  - R.O. used for a proof-friendly PRF and the proof system itself

A PRF from RO

- $F_{SK}(Q) = H(SK||Q)$ is a PRF if H is a random oracle (and SK long enough)
  - Why? Infeasible to guess SK correctly. Without querying H on prefix SK, F_{SK} is identical to a truly random function.
- But no PK for this F and no way to prove correct evaluation
- Instead, let (SK,PK) = $(y, Y=g^y)$ and $F_y(Q) = H'(\ C^y\ )$, where $C=H(Q)$
  - Still a PRF (remains infeasible to guess y from Y, under DLA)
  - Need a way to prove that $F_{SK}(Q) = z$
    - Plan: Reveal $D=C^y$ and <u>prove</u> that it is indeed C^y. But how?
    - A ZK proof of equality of discrete logs for (g,Y) and (C,D)
      - i.e., $\exists y$ s.t. $g^y = Y$ and $C^y = D$

ZK Proof

- Alice and Bob hold some data x. Bob wants to prove that it has some "property."
  - Properties we are typically interested in are "NP properties"
    - An NP property is specified by a poly-time computable predicate R: x has the property ≡ $\exists w$ s.t. $R(x,w)=1$
    - i.e., there's a certificate to prove the property
  - Trivial proof for NP properties: send the certificate
- Can a proof reveal nothing beyond the fact that x has the property?
- Yes!
- Will allow interactive proofs (for now)

ZK Proof

- Consider an NP property specified by a predicate R: i.e., x has the property $\equiv \exists w$ s.t. $R(x,w)=1$. A ZK proof protocol $P \longleftrightarrow V$ has the following properties
  - Completeness: if $\exists w\ R(x,w)=1$, then $\Pr[P(x,w) \longleftrightarrow V(x) = 1] = 1$
  - Soundness: if $\nexists w\ R(x,w)=1$, then $\Pr[P^*(x) \longleftrightarrow V(x) = 1] = \text{negl}$ (for any PPT P^*)
    - A stronger notion: Proof of Knowledge
  - Zero-Knowledge: if $\exists w\ R(x,w)=1$, then view of the verifier in $P(x,w) \longleftrightarrow V(x)$ can be (indistinguishably) simulated from x
    - This is called Honest Verifier ZK
    - Stronger property: For any PPT V^*, there is a simulator S s.t., $\text{View}_{V^*}(P(x,w) \longleftrightarrow V^*(x)) \approx S(x)$

V learns nothing beyond the fact that x has the property

Honest-Verifier ZK Proofs

- ZK Proof of knowledge of discrete log of $A=g^r$
  - Aside: this can be used to prove knowledge of the message in an El Gamal encryption $(A,B) = (g^r, m\,Y^r)$
  - $P \rightarrow V$: $U := g^u$; $V \rightarrow P$: v ; $P \rightarrow V$: $w := rv + u$;
    V checks: $g^w = A^v U$
  - Proof of Knowledge:
    - Firstly, $g^w = A^v U \Rightarrow w = rv+u$, where $U = g^u$
    - If after sending U, P could respond to two different values of v: $w_1 = rv_1 + u$ and $w_2 = rv_2 + u$, then can solve for r
  - HVZK: simulation picks w, v first and sets $U = g^w/A^v$

HVZK and Special Soundness

- HVZK: Simulation for honest (passively corrupt) verifier
  - e.g. in PoK of discrete log, simulator picks (v,w) first and computes U (without knowing u). Relies on verifier to pick v independent of U.
- Special soundness: If given (U,v,w) and (U,v',w') s.t. $v \neq v'$ and both accepted by verifier, then can derive a valid witness
  - e.g. solve r from $w=rv+u$ and $w'=rv'+u$ (given v,w,v',w')
  - Implies soundness: for each U s.t. prover has significant probability of being able to convince, can extract r from the prover with comparable probability (using "rewinding", in a stand-alone setting)

Honest-Verifier ZK Proofs

- ZK PoK to prove equality of discrete logs for $((g,Y),(C,D))$, i.e., $Y = g^r$ and $D = C^r$ [Chaum-Pederson]
    - Can be used to prove equality of two El Gamal encryptions (A,B) & (A',B') w.r.t public-key (g,Y): set $(C,D) := (A/A',B/B')$
- $P \rightarrow V$: $(U,M) := (g^u,C^u)$; $V \rightarrow P$: v ; $P \rightarrow V$: $w := rv+u$;
  V checks: $g^w = Y^vU$ and $C^w = D^vM$

  Two parallel executions of the previous proof, with same v and w (and same u, r)
- Special Soundness:
    - $g^w=Y^vU$, $C^w=D^vM$ $\Rightarrow$ $w = rv+u = r'v+u'$ where $U=g^u$, $M=g^{u'}$ and $Y=g^r$, $D=C^{r'}$
    - If after sending (U,M) P could respond to two different values of v: $rv_1 + u = r'v_1 + u'$ and $rv_2 + u = r'v_2 + u'$, then $r=r'$
- HVZK: simulation picks w, v first and sets $U=g^w/A^v$, $M=C^w/D^v$

Fiat-Shamir Heuristic

- Limitation: Honest-Verifier ZK does not guarantee ZK when verifier is actively corrupt
  - Can be fixed by implementing the verifier using "secure 2-party computation"
    - If verifier is a public-coin program (as in Chaum-Pederson) — i.e., simply picks random values publicly — then 2PC needed only to generate random coins
    - Alternatively, Fiat-Shamir Heuristic: random coins from verifier defined as H(trans), where H is a random oracle and trans is the transcript of the proof so far
      - Also, removes need for interaction in the proof!

VRF

- NSEC5 VRF based on the discrete log assumption and a random oracle based non-interactive ZK proof
  - (SK,PK) = (y, $Y=g^y$) and $F_y(Q) = H'(C^y)$, where C=H(Q)
  - If H' is an R.O., then DLA ensures F is a PRF
  - Proof that $F_y(Q) = z$: D s.t. H'(D) = z and a ZK proof of equality of discrete logs for (g,Y) and (C,D)
    - i.e., ∃y s.t. $g^y = Y$ and $C^y = D$
    - Non-interactive proof using the Fiat-Shamir heuristic applied to Chaum-Pederson
  - Does adding the proof hurt PRF property?
    - Proof reveals nothing more than what (g,Y,C,D) reveals
    - Which reveals nothing more than what (g,Y) reveals: (C,D) can be simulated as (g^r,Y^r) since H random oracle

Summary

- Fairly efficient ZK proofs systems exist for all NP properties
- Even more efficient HVZK proof systems for specialised problems like equality of discrete logs
- Fiat-Shamir heuristics can convert such protocols into non-interactive proofs secure against actively corrupt verifiers too (but in the Random Oracle model)