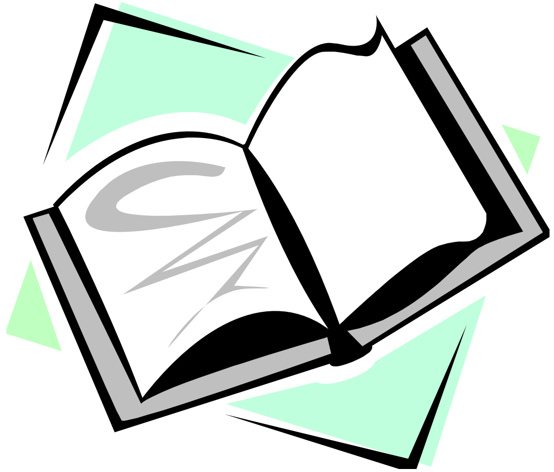
Computer Security
CS 426
Lecture 35

Commitment & Zero Knowledge Proofs

Readings for This Lecture

- Optional:
  - Haveli and Micali: "Practical and Privably-Secure Commitment Schemes from Collision-Free Hashing"
  - Jean-Jacques et al.: How to explain Zero-Knowledge Protocols to Your Children



- This lecture's topics won't be in the final exam

Commitment schemes

- An electronic way to temporarily hide a value that cannot be changed
  - Stage 1 (Commit)
    - Sender locks a message in a box and sends the locked box to another party called the Receiver
  - State 2 (Reveal)
    - the Sender proves to the Receiver that the message in the box is a certain message
- Usage scenarios: flipping fair coins, bidding for a contract

Types of commitment

- Bit commitment
- Integer commitment
- String commitment

Security properties of commitment schemes

- Hiding
  - at the end of Stage 1, no adversarial receiver learns any information about the committed value
- Binding
  - at the end of Stage 1, no adversarial sender can successfully reveal two different values in Stage 2

A broken commitment scheme

- Using encryption
  - Stage 1 (Commit)
    - the Sender generates a key *k* and sends $E_k[M]$ to the Receiver
  - State 2 (Reveal)
    - the Sender sends *k* to the Receiver, the Receiver can decrypt the message

- What is wrong using the above as a commitment scheme? Is it hiding? Is this binding?

Formalizing Security Properties of Commitment schemes

- Two kinds of adversaries
  - those with infinite computation power and those with limited computation power
- Unconditional hiding
  - the commitment phase does not leak any information about the committed message, in the information theoretical sense (similar to perfect secrecy)
- Computational hiding
  - an adversary with limited computation power cannot learn anything about the committed message (similar to semantic security)

Formalizing Security Properties of Commitment schemes

- Unconditional binding
  - after the commitment phase, an infinite powerful adversary sender cannot reveal two different values
- Computational binding
  - after the commitment phase, an adversary with limited computation power cannot reveal two different values

- No commitment scheme can be both unconditional hiding and unconditional binding

Another (also broken) commitment scheme

- Using a one-way function *H*
  - Stage 1 (Commit)
    - the Sender sends c=*H*(*M*) to the Receiver
  - State 2 (Reveal)
    - the Sender sends *M* to the Receiver, the Receiver verifies that c=*H*(*M*)

- What is wrong using this as a commitment scheme? Is it binding? Is it hiding?

Commitment Schemes Using Cryptographic Hash Functions

- A scheme likely secure enough in practice, but difficult to prove security (assuming only H is one-way and strongly collision-resistant)
  - To commit to message M, choose random, fixed-length r, send H(r || M)
  - To open commitment, send r, M
  - Receiver cannot fully recover M.
    - Is this computational or information theoretic hiding?
  - Sender cannot find another M' to open.
    - Is this computational or information theoretic binding?

Commitment must be randomized.

For Provably Secure Commitment Scheme based on Cryptogrpahic Hash

- See Haveli and Micali:
  - “Practical and Privably-Secure Commitment Schemes from Collision-Free Hashing”
  - Uses Universal Hashing (a family of hash functions with some properties)

The Pederson Commitment Scheme

- Public parameters: (p,g,h)
  - p: large prime (1024 bit)
  - g: a number in [2, p-1]
  - h: another element such that $\log_g h$ is unknown
- Protocol
  - To commit to x, committer chooses random r and sends ($g^x h^r$ mod p) to the receiver.
  - To open, the committer sends x and r to the receiver
- Benefits:
  - One can prove many things about the committed value without opening it

Pedersen Commitment Scheme (cont.)

- Unconditionally hiding
  - Given a commitment c, every value x is equally likely to be the value committed in c.
  - For example, given x,r, and any x', there exists r' such that $g^x h^r = g^{x'} h^{r'}$, in fact $r = (x-x')a^{-1} + r \bmod q$.
- Computationally binding
  - Suppose the sender open another value x' ≠ x. That is, the sender find x' and r' such that $c = g^{x'} h^{r'} \bmod p$. Now the sender knows x,r,x', and r' s.t., $g^x h^r = g^{x'} h^{r'} \pmod p$, the sender can compute $\log_g(h) = (x'-x) \cdot (r-r')^{-1}$. Assume DL is hard, the sender cannot open the commitment with another value.

Properties of Interactive Zero-Knowledge Proofs

- Zero-knowledge Proof of Knowledge
  - Proving knowing a secret, without revealing any information about the secret.
- Completeness
  - Given honest prover and honest verifier, the protocol succeeds with overwhelming probability
- Soundness
  - No one who doesn’t know the secret can convince the verifier with nonnegligible probability
- Zero knowledge
  - The proof does not leak any additional information

Intuitive Explanation of ZK

- See the paper “How to explain Zero-Knowledge Protocols to Your Children”
  – http://sparrow.ece.cmu.edu/group/630-f08/readings/ZK-IntroPaper.pdf

Schnorr Protocol (ZK Proof of Knowing Discrete Log)

- System parameter: p, q, g
  - We have $g^q = 1 \bmod p$
- Public identity: $c = g^a \bmod p$
- Private authenticator: a
- Protocol

  1. P: picks random r in [1..q], sends $d = g^r \bmod p$,
  2. V: sends random challenge e in $[1..2^t]$
  3. P: sends $y = r - ea \pmod q$
  4. V: accepts if $d = g^y c^e \bmod p$

Security of Schnorr Protocol - Soundness

- Probability of forge: $1/2^t$
  - The prover who does not know *a* can cheat by guess *e*
  - Set d= $c^e g^y$ at the first step
- We build a knowledge extractor as follows. Suppose the prover is challenged twice with on same c, first with e1, second with e2.
  - Send e1, receive y1 such that $g^{y1}c^{e1} = d$
  - Send e2, receive y2 such that $g^{y2}c^{e2} = d$
  - $g^{y1-y2}=c^{e2-e1}$, output $\log_g(c) = (y1-y2)\cdot(e2-e1)^{-1}$

Pedersen Commitment – ZK Prove know how to open

- Public commitment $c = g^x h^r \pmod p$
- Private knowledge x,r
- Protocol:

  1. P: picks random y, s in [1..q], sends $d = g^y h^s \bmod p$
  2. V: sends random challenge e in [1..q]
  3. P: sends $u=y+ex$, $v=s+er \pmod q$
  4. V: accepts if $g^u h^v = dc^e \pmod p$
- Security property – similar to Schnorr protocol

Other Things One Can Prove in ZK fashion with Pederson Commitments

- The committed value is a bit.
- The committed value is in a range.
- Two committed values equal
- Two committed values satisfy some linear relations
- And many more

Coming Attractions …

- Network Security Defenses