1 Zero-Knowledge (review)

1.1 Review

Alice has $y, g, p$ and claims to know $x$ such that $y = g^x \mod p$. Alice proves knowledge of $x$ to Bob w/o revealing $x$. No information leakage. How? Zero-knowledge proof.

Malicious Alice ($A'$): wants to prove $x$ even though she doesn’t know $x$. IFF Alice can guess a challenge value correctly, she can cheat.

$A'$’s method:

1. Guess that $c = c'$.
   - Needs $g^d = y^{c'}b$
   - Knows $g, y$. Only knows $c$ if guess is correct.
   - Left with $b \& d$; gets to choose values. Choose $b \& d$ such that $g^d = y^{c'}b$ holds.
   - If $A'$ chooses $b$ first, she can’t compute $d$. Why? Discrete log problem.
   - Should choose $d$, then compute $b$:
     
     \[
g^d = y^{c'}b \\
     b = g^d y^{-c}
     \]
   - If Alice guesses $c'$ correctly here, she ”wins”. Otherwise ”loses”.

March 19: Zero-Knowledge (cont.) and Signatures

March 26, 2013
1.2 Main Points

- This is really a challenge-response protocol.
- 3-way protocol, i.e. "Sigma protocol" (Σ has 3 points)
  1. Alice commits some data.
  2. Bob challenges Alice
  3. Alice opens up some data in a way that is consistent w/ the commitment and the challenge.

- This challenge must be unpredictable before Alice completes the first step.

1.3 About the Challenge

- We have \( c \in \{0,1\} \). But \( c \) doesn’t have to be \( \{0,1\} \)...
- Can be any set of numbers.
- When \( c \in \{0,1\} \), \( P(\text{Alice cheats}) = \frac{1}{2} \).
- To avoid this (high success probability), when \( c \) is \( t \) bits long note that \( P(\text{Alice cheats}) = \frac{1}{2^t} \)... So choose a cryptographically large \( t \).
- Technical note: Not important for exam. When you extend challenge to \( t \) bits, it is no longer technically zero-knowledge. This has to do with strict security definitions. It is still a sigma protocol, though.
• **MAIN POINT:** Malicious Alice locks in some value based on a challenge. She will have a hard time if she doesn’t know b.

## 2 Fiat-Shamir Heuristic

*Note: $y = g^x \mod p$ has same form as public/private key encryption.*

Note that protocol from the previous section was called the Schnorr protocol. Drawback: it is interactive. Enter Fiat-Shamir Heuristic.

- Let $\{(p, g, y), \{b, c, d\}\}$ be a ‘transcript’ for a run of Schnorr.
- Unconvincing because there’s no guarantee that $b$ was generated before $c$.
- To verify, it is sufficient to show that $b$ is chosen before $c$.
- Set $c =$ the hash of $b$. That is:

$$\{(p, g, y), \{b, H(b), d\}\}$$

*The fine print:*

1. $b$ before $c$ relies on preimage resistance
2. $c$ needs to be unpredictable. For example if you use challenge $\epsilon \{0, 1\}$, you could just try a couple values until condition is met. Assume output of hash is random, i.e. random oracle assumption holds.
3. $c$ needs to be large so that faked $< b, c, d >$ values do not by coincidence have $c = H(b)$.
4. Could actually do $c = H(b, \{p, g, y\})$

## 3 Digital Signatures

### 3.1 Overview

- Alice has a public key $y$ and a secret key $x$. Alice signs a message $m$ as:

$$s = SIG_x(m, r)$$

Sometimes we have a random $I$ (initialization vector), too.

- Bob or anyone else can easily verify $< m, s >$:

$$True/False = VERIFY_y(m, s)$$
• **Note on signature vs. MAC:** Main difference is signature convinces anyone, whereas a MAC convinces only a person you share the key with. MACs do not work for 2 parties. Signatures are slower than MACs.

### 3.2 Constructing signatures

Two primary ways:

1. Built from public-key decryption
2. Built from zero-knowledge proofs (e.g. Fiat-Shamir or non-interactive zero-knowledge)

#### 3.2.1 Constructing w/ public key encryption

- Intuition: Say you have a public-key encryption:
  
  \[ c = ENC_{pk}(m) \]
  
  \[ m = DEC_{sk}(c) \]

- Build a signature \( SIG \) where:
  
  \[ SIG_{sk}(m) \equiv DEC_{sk}(c) = s \]

  Then send \((m, s)\).

- To verify:
  
  \[ ENC_{pk}(s) = m \]

- Note this doesn’t work for Elgamal or other CPA-secure encryptions)

- Because \( ENC \) is randomized, you won’t necessarily get \( m \) back when you do \( ENC_{pk}(s) \equiv m \)

- RSA is usable as a signature scheme, since it’s not based on DLP. We’ll revisit when we see RSA.

#### 3.2.2 Constructing w/ Fiat-Shamir

- Schnorr proves that a transcript was generated by a person who knows \( x \) such that \( y = g^x \mod p \).

- \( x \) can be secret key. \( y \) can be public key.

- If there were a way to bind a message to one of these transcripts, we’d have a signature scheme. So how to bind such a message? Answer: Schnorr signature.
4 Schorr Signature

4.1 Overview

- \{p, y, g\}, \{b, c = H(b, m), d\} where y = publickey, H(b, m) = hashedmessage/challenge.
- Only person who knows x can generate a valid d here.

4.2 Sign Function

We have:

\[ s = \text{SIG}_x(m, r) = \{s_1, s_2, s_3\} \]
\[ = \{g^r, h(g^r || m), r + H(g^r || m) \cdot x \mod p \equiv \{b, c, b + cx\} = \{s_1, s_2, s_3\} \]

Verify using:

\[ \text{VERIFY}(m, s) : g^{s_3} = g^{s_2} y. \text{ Also } s_2 = H(s_1, m). \]

- Signatures should prevent forgeries. Meaning, it should be infeasible to sign a message m without knowledge of private key.

- Assumptions we’ve made for Schnorr:
  1. Secure because of the discrete log problem.
  2. Preimage resistance of involved hash functions.
  3. Random oracle assumption holds (not covered in class)