Advanced Topics in Cryptography

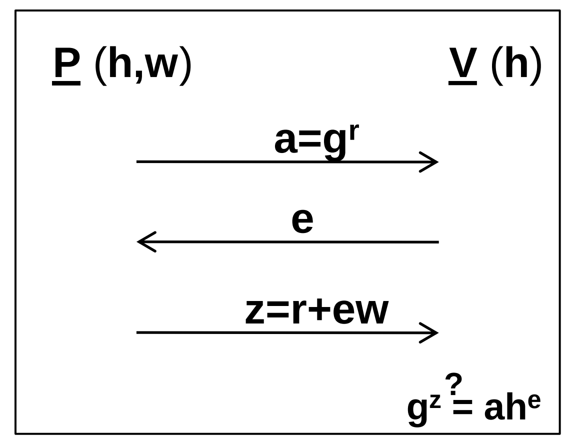
Lecture 4

Benny Pinkas

Based on slides of Yehuda Lindell

An Example – Schnorr DLOG

- Let G be a group of order q, with generator g
- P and V have input $h \in G$. P has w such that $g^w = h$
- P proves that to V that it knows $\mathrm{DLOG}_g(h)$
  - **P** chooses a random **r** and sends $\mathbf{a=g^r}$ to **V**
  - **V** sends **P** a random $\mathbf{e} \in \{\mathbf{0,1}\}^{\mathbf{t}}$
  - **P** sends **z=r+ew** mod **q** to **V**
  - **V** checks that $\mathbf{g^z = ah^e}$



ZK from Sigma Protocols

- A tool: commitment schemes
- Enables to commit to a chosen value while keeping it secret, with the ability to reveal the committed value later.
- A commitment has two properties:
  - Binding: After sending the commitment, it is impossible for the committing party to change the committed value.
  - Hiding: By observing the commitment, it is impossible to learn what is the committed value. (Therefore the commitment process must be probabilistic.)
- It is possible to have unconditional security for any one of these properties, but not for both.

ZK from Sigma Protocols

- The basic idea
  - Have **V** first commit to its challenge **e** using a perfectly-hiding commitment
- The protocol
  - **P** sends the first message α of the commit protocol
  - **V** sends a commitment c=$\mathbf{Com}_\alpha(\mathbf{e};\mathbf{r})$
  - **P** sends a message **a**
  - **V** opens the commitment by sending (**e**,**r**)
  - **P** checks that c=$\mathbf{Com}_\alpha(\mathbf{e};\mathbf{r})$ and if yes sends a reply **z**
  - **V** accepts based on (**x**,**a**,**e**,**z**)

ZK from Sigma Protocols

- Soundness:
  - The perfectly hiding commitment reveals nothing about **e** and so soundness is preserved
- Zero knowledge
  - In order to simulate:
    - **V** commits. Send **a′** generated by the simulator, for a random **e′**.
    - Receive **V**'s decommitment to **e**
    - Run the simulator again with **e**, rewind **V** and send **a**
      - Repeat until **V** decommits to **e** again
    - Conclude by sending **z**
  - Analysis…

Pedersen Commitments

- Highly efficient perfectly-hiding commitments (two exponentiations for string commit)
  - **Parameters:** generator $\mathbf{g}$, order $\mathbf{q}$
  - **Commit protocol** (commit to $\mathbf{x}$):
    - Receiver chooses random $\mathbf{k}$ and sends $\mathbf{h=g^k}$
    - Sender sends $\mathbf{c=g^r h^x}$, for random $\mathbf{r}$
  - **Hiding:**
    - For every $\mathbf{x,y}$ there exist $\mathbf{r,s}$ s.t. $\mathbf{r+kx = s+ky \bmod q}$
  - **Binding:**
    - If sender can open commitment in two ways, i.e. find $(\mathbf{x,r}),(\mathbf{y,s})$ s.t. $\mathbf{g^r h^x=g^s h^y}$, then $\mathbf{k} = (\mathbf{r-s})/(\mathbf{y-x}) \mathbf{\bmod q}$

Efficiency of ZK

- Using Pedersen commitments, the entire DLOG proof costs only 5 additional group exponentiations
  - In Elliptic curve groups this is very little

ZKPOK from Sigma Protocols

- Is the previous protocol a proof of knowledge?
  - It seems not to be
  - The extractor for the Sigma protocol needs to obtain two transcripts with the same **a** and different **e**
    - The prover may choose its first message **a** differently for every commitment string.
    - But in this protocol the prover sees a commitment to **e** before sending **a**.
    - So if the extractor changes **e**, the prover changes **a**

ZKPOK from Sigma Protocols

- Solution: use a trapdoor (equivocal) commitment scheme
  - Given a trapdoor, it is possible to open the commitment to any value
- Pedersen has this property – given the discrete log k of **h**, can decommit to any value
  - Commit to **x**: $\mathbf{c = g^r h^x}$
  - To decommit to **y**, find **s** such that $\mathbf{r+kx = s+ky}$
  - This is easy if **k** is known: compute $\mathbf{s = r+k(x-y) \bmod q}$

ZKPOK from Sigma Protocols

- The basic idea
  - Have **V** first commit to its challenge **e** using a perfectly-hiding trapdoor (equivocal) commitment (such as Pedersen)
- The protocol
  - **P** sends the first message α of the commit protocol (e.g., including h in the case of Pedersen commitments).
  - **V** sends a commitment c=$\mathbf{Com}_{\alpha}$(**e**;**r**)
  - **P** sends a message **a**
  - **V** sends (**e**,**r**)
  - **P** checks that c=$\mathbf{Com}_{\alpha}$(**e**;**r**) and if correct sends **z** and also the **trapdoor** for the commitment
  - **V** accepts if the **trapdoor** is correct and (**x**,**a**,**e**,**z**) is accepting

ZKPOK from Sigma Protocols

P (x,w) **V** (x)

$\mathbf{h=g^k}$, random **k** $\rightarrow$

$\leftarrow$ $\mathbf{c=g^r h^e}$

Sigma msg **a** $\rightarrow$

$\leftarrow$ **(e,r)**

Verify $\mathbf{c=g^r h^e}$

(z,k) $\rightarrow$

Verify $\mathbf{h=g^k}$
Verify (a,e,z)

ZKPOK from Sigma Protocols

- Why does this help?
  - **Zero-knowledge** remains the same
  - **Extraction:** after verifying the proof once, the extractor obtains **k** and can rewind back to the decommitment of **c** and send any (**e′**,**r′**)

- Efficiency:
  - Just 6 exponentiations (very little)

ZK and Sigma Protocols

- We typically want zero knowledge, so why bother with sigma protocols?
  - There are many useful general transformations
    - E.g., parallel composition, compound statements
    - The ZK and ZKPOK transformations can be applied on top of the above, so obtain transformed ZK
  - It is **much harder** to prove ZK than Sigma
    - ZK – distributions and simulation
    - Sigma: only HVZK and special soundness

Using Sigma Protocols and ZK

- Prove that the El Gamal encryption (u,v) under public-key (g,h) is to the value m
  - By the definition of El Gamal encryption: $\mathbf{u=g^r}$, $\mathbf{v=h^r \cdot m}$
  - Thus (**g,h,u,v/m**) is a DH tuple
  - So, given (**g,h,u,v,m**), just prove that (**g,h,u,v/m**) is a DH tuple

Another application: Efficient Coin Tossing

- P_1 chooses a random x, sends $(g,h,g^r,h^r x)$
- P_1 ZK-proves that it knows the encrypted value
  - Suffices to prove that it knows the discrete log of **h**
- P_2 chooses a random y and sends to P_1
- P_1 sends x (without decommitting)
- P_1 ZK-proves that encrypted value was x
- Both parties output x+y

- Cost: O(1) exponentiations

Prove Knowledge of Committed Value

- Relation: $((h,c),(x,r)) \in R$ iff $c=g^r h^x$
- Sigma protocol:
  - P chooses random α,β and sends $\mathbf{a=h^{\alpha}g^{\beta}}$
  - V sends a random $\mathbf{e}$
  - P sends $\mathbf{u=\alpha+ex,\ v=\beta+er}$
  - V checks that $\mathbf{h^u g^v = ac^e}$

- Completeness:
  - $\mathbf{h^u g^v = h^{\alpha+ex} g^{\beta+er} = h^{\alpha} g^{\beta} (h^x g^r)^e = ac^e}$

Pedersen Commitment Proof

- Special soundness:
  - Given (**a,e,u,v**),(**a,e′,u′,v′**), we have $h^u g^v = ac^e$, $h^{u'} g^{v'} = ac^{e'}$

    Thus, $h^u g^v c^{-e} = h^{u'} g^{v'} c^{-e'}$

    and $h^{u-u'} g^{v-v'} = c^{e-e'}$
  - Conclude: $x = (u-u')(e-e')$ and

    $r = (v-v')(e-e')$

P ((h,c),(x,r)) **V**

$$\xrightarrow{a = h^{\alpha} g^{\beta}}$$

$$\xleftarrow{e}$$

$$\xrightarrow{u=\alpha+ex,\ v=\beta+er}$$

$$h^u g^v \stackrel{?}{=} ac^e$$

- Special HVZK
  - Given (**g,h,h,c**) and **e**, choose random **u,v** and compute $a = h^u g^v c^{-e}$

Proof of Pedersen Value

- Prove that the Pedersen committed value is x
- Relation: $((h,c,x),(r)) \in R$ iff $c=g^r h^x$
  - Observe: $\mathbf{ch^{-x} = g^r}$
  - Conclusion: just prove that you know the discrete log of $\mathbf{ch^{-x}}$

- Application: statistical coin tossing

Constructions of Oblivious Transfer

1-out-of-2 Oblivious Transfer

- Two players: sender and receiver.
  - Sender has two inputs, x_0, x_1.
  - Receiver has an input $j \in \{0,1\}$.
- Output:
  - Receiver learns x_j and nothing else.
  - Sender learns nothing about j.

- Depending on the OT variant, the inputs x_0,x_1 could be strings or bits.

Security Definitions for OT

- It appeared to be quite hard to design an OT protocol that is secure against malicious adversaries in the sense of comparison to the ideal model.
    - Only recently were efficient such protocols designed.

- Therefore looser security definitions were used
  - These definitions ensure privacy but not correctness.
  - Namely, they do not ensure that the output is that of an OT functionality, or ensure independence of inputs.

Security Definitions for OT

- Defining what is means to protect the receiver's privacy is easy, since the sender receives no output in the ideal model and should therefore learn nothing about the receiver's input.

- Receiver's privacy – indistinguishability
  - For any values of the sender's inputs x_0, x_1, the sender cannot distinguish between the case that the receiver's input is 0 and the case that it is 1.

Security Definitions for OT

- Definition of sender's security:
  - This case is harder since the receiver does learn something about the sender's input

Security Definitions for OT

- Definition of sender's security:
  - For every algorithm A' that the receiver might run in the real implementation of oblivious transfer
  - there is an algorithm A" that the receiver can run in the ideal implementation
  - such that for any values of x_0,x_1 the outputs of A' and A" are indistinguishable.
  - Namely, the receiver in the real implementation does not learn anything more than the receiver in the ideal implementation.

- This definition does not handle delicate issues, such as whether the receiver "knows" j or the sender "knows" x_0,x_1

The Even-Goldreich-Lempel 1-out-of-2 OT construction (providing security only against semi-honest adversaries)

- Setting:
  - Sender has two inputs, x_0, x_1.
  - Receiver has an input $j \in \{0,1\}$.
- Protocol:
  - Receiver chooses a random public/private key pair *(E,D)*.
  - It sets $PK_j=E$, and chooses PK_{1-j} at random from the same distribution as that of public keys*. It then sends (PK_0,PK_1) to the sender.
  - The sender encrypts x_0 with PK_0, and x_1 with PK_1, and sends the results to the receiver.
  - The receiver decrypts x_j.
  - Why is this secure against semi-honest adversaries?
- (*) It is required that it is possible to sample items with the exact distribution of public keys, and do this without knowing how to decrypt the resulting ciphertexts.