Advanced Topics in Cryptography

Lecture 4

Benny Pinkas

Based on slides of Yehuda Lindell

Zero Knowledge

- Prover P, verifier V, language L
- P proves that $x \in L$ without revealing anything
  - **Completeness:** **V** always accepts when honest **P** and **V** interact
  - **Soundness:** **V** accepts with negligible probability when $\mathbf{x} \notin \mathbf{L}$, for any $\mathbf{P}^*$
    - Computational soundness: only holds when $\mathbf{P}^*$ is polynomial-time
- **Zero-knowledge:**
  - There exists a simulator **S** such that **S**(**x**) is indistinguishable from a real proof execution

ZK Proof of Knowledge

- Prover P, verifier V, relation R
- P proves that it knows a witness w for which $(x,w) \in R$ without revealing anything
  - The proof is zero knowledge as before
  - There exists an extractor **K** that can obtain from any $\mathbf{P}^*$, a **w** such that $(\mathbf{x},\mathbf{w}) \in \mathbf{R}$, with the same probability that $\mathbf{P}^*$ convinces **V**.
- Equivalently:
  - The protocol securely computes the functionality

    $$\mathbf{f}_{\mathbf{zk}}((\mathbf{x},\mathbf{w}),\mathbf{x}) = (\text{-},\mathbf{R}(\mathbf{x},\mathbf{w}))$$

Zero Knowledge

- An amazing concept; everything can be proven in zero knowledge
- Central to fundamental feasibility results of cryptography (e.g., GMW)

- But, can it be efficient?
  - It seemed that zero-knowledge protocols for "interesting languages" are complicated and expensive
- Zero knowledge is often avoided at significant cost

Sigma Protocols

- A way to obtain efficient zero knowledge
  - Many general tools
  - Many interesting languages can be proven with a sigma protocol

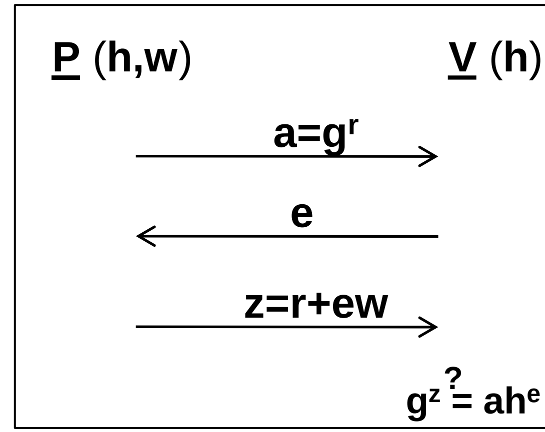
An Example – Schnorr DLOG

- 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 $DLOG_g(h)$
  - **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 $\mathbf{z = r + ew}$ mod **q** to **V**
  - **V** checks that $\mathbf{g^z = ah^e}$

- Completeness
  - $\mathbf{g^z = g^{r+ew} = g^r(g^w)^e = ah^e}$

Schnorr's Protocol

- Proof of knowledge
  - Assume **P** can answer two queries **e** and **e′** for the same **a**
  - Then, it holds that $g^z = ah^e$, $g^{z'} = ah^{e'}$
  - Thus, $g^z h^{-e} = g^{z'} h^{-e'}$ and $g^{z-z'} = h^{e-e'}$
  - Therefore $h = g^{(z-z')/(e-e')}$
  - That is: DLOG**g**(**h**) = **(z-z′)**/**(e-e′)**
- Conclusion:
  - If **P** can answer with probability greater than $1/2^t$, then it must know the dlog



Schnorr's Protocol

- What about zero knowledge? This does not seem easy.
- But ZK holds if the verifier sends a random challenge e
- This property is called "Honest-verifier zero knowledge"
  - The simulation:
  - Choose a random **z** and **e**, and compute $\mathbf{a = g^z h^{-e}}$
  - Clearly, (**a**,**e**,**z**) have the same distribution as in a real run, and $\mathbf{g^z = a h^e}$
- This is not a very strong guarantee, but we will see that it yields efficient general ZK.

Definitions

- Sigma protocol template
  - **Common input: P** and **V** both have **x**
  - **Private input: P** has **w** such that **(x,w)**∈**R**

  - **Protocol:**
    - **P** sends a message **a**
    - **V** sends a <u>random</u> **t**-bit string **e**
    - **P** sends a reply **z**
    - **V** accepts based solely on (**x,a,e,z**)

Definitions

- Completeness: as usual
- Special soundness:
  - There exists an algorithm **A** that given any **x** and pair of transcripts (**a**,**e**,**z**),(**a**,**e′**,**z′**) with **e**≠**e′** outputs **w** s.t. (**x**,**w**)∈**R**
- Special honest-verifier ZK
  - There exists an **M** that given any **x** and **e** outputs (**a**,**e**,**z**) which is distributed exactly like a real execution where **V** sends **e**

Sigma Protocol for proving a DH Tuple

- Relation R of Diffie-Hellman tuples
  - $(\mathbf{g},\mathbf{h},\mathbf{u},\mathbf{v}) \in \mathbf{R}$ iff there exists $\mathbf{w}$ s.t. $\mathbf{u}=\mathbf{g}^{\mathbf{w}}$ and $\mathbf{v} = \mathbf{h}^{\mathbf{w}}$
  - Useful in many protocols
- This is a proof of membership, not of knowledge

- Protocol
  - **P** chooses a random $\mathbf{r}$ and sends $\mathbf{a}=\mathbf{g}^{\mathbf{r}}$, $\mathbf{b}=\mathbf{h}^{\mathbf{r}}$
  - **V** sends a random $\mathbf{e}$
  - **P** sends $\mathbf{z}=\mathbf{r}+\mathbf{ew}$ mod $\mathbf{q}$
  - **V** checks that $\mathbf{g}^{\mathbf{z}}=\mathbf{a}\mathbf{u}^{\mathbf{e}}$, $\mathbf{h}^{\mathbf{z}}=\mathbf{b}\mathbf{v}^{\mathbf{e}}$

Sigma Protocol DH Tuple

- Completeness: as in DLOG
- Special soundness:
  - Given $(a,b,e,z),(a,b,e',z')$, we have $g^z=au^e, g^{z'}=au^{e'}, h^z=bv^e, h^{z'}=bv^{e'}$ and so like in DLOG on both
    - $w = (z-z')/(e-e')$
- Special HVZK
  - Given (g,h,u,v) and e, choose random z and compute
    - $a = g^z u^{-e}$
    - $b = h^z v^{-e}$

P $((g,h,u,v),w)$ **V**

$\xrightarrow{a=g^r,\ b=h^r}$

$\xleftarrow{e}$

$\xrightarrow{z=r+ew}$

$g^z \stackrel{?}{=} au^e$

$h^z = bv^e$

Basic Properties

- Any sigma protocol is an interactive proof with soundness error 2^{-t}
- Properties of sigma protocols are invariant under parallel composition
- Any sigma protocol is a proof of knowledge with error 2^{-t}
  - The difference between the probability that $\mathbf{P}^*$ convinces $\mathbf{V}$ and the probability that $\mathbf{K}$ obtains a witness is at most $\mathbf{2^{-t}}$
  - Proof needs some work

Tools for Sigma Protocols

- Prove compound statements
  - AND, OR, subset
- ZK from sigma protocols
  - Can first make a compound sigma protocol and then compile it
- ZKPOK from sigma protocols

AND of Sigma Protocols

- To prove the AND of multiple statements
  - Run all in parallel
  - Can use the same verifier challenge **e** in all
- Sometimes it is possible to do better than this
  - Statements can be batched
  - E.g. proving that many tuples are DDH can be done in much less time than running all proofs independently
    - Batch all into one tuple and prove

OR of Sigma Protocols

- This is more complicated
  - Given two statements and two appropriate Sigma protocols, wish to prove that at least one is true, without revealing which

- The solution – an ingenious idea from [CDS]
  - Using the simulator, if **e** is known ahead of time it is possible to cheat
  - We construct a protocol where the prover can cheat in one out of the two proofs

OR of Sigma Protocols

- The template for proving x_0 or x_1:
  - **P** sends two first messages $(\mathbf{a_0},\mathbf{a_1})$
  - **V** sends a single challenge $\mathbf{e}$
  - **P** replies with
    - Two challenges $\mathbf{e_0},\mathbf{e_1}$ s.t. $\mathbf{e_0}\oplus\mathbf{e_1} = \mathbf{e}$
    - Two final messages $\mathbf{z_0},\mathbf{z_1}$
  - **V** accepts if $\mathbf{e_0}\oplus\mathbf{e_1} = \mathbf{e}$ and $(\mathbf{a_0},\mathbf{e_0},\mathbf{z_0}),(\mathbf{a_1},\mathbf{e_1},\mathbf{z_1})$ are both accepting
- How does this work?

OR of Sigma Protocols

- **P** sends two first messages ($\mathbf{a_0,a_1}$)
  - Suppose that **P** has a witness for $\mathbf{x_0}$ (but not for $\mathbf{x_1}$)
  - **P** chooses a random $\mathbf{e_1}$ and runs SIM to get ($\mathbf{a_1,e_1,z_1}$)
  - **P** sends ($\mathbf{a_0,a_1}$)
- **V** sends a single challenge **e**
- **P** replies with e_0,e_1 s.t. $e_0 \oplus e_1 = e$ and with z_0,z_1
  - **P** already has $\mathbf{z_1}$ and can compute $\mathbf{z_0}$ using the witness
- <u>Soundness</u>
  - If P doesn't know a witness for $\mathbf{x_1}$, he can only answer for a single $\mathbf{e_1}$
  - This means that **e** defines a single challenge $\mathbf{e_0}$, like in a regular proof

OR of Sigma Protocols

- Special soundness
  - Relative to first message $(\mathbf{a_0,a_1})$, and two different $\mathbf{e,e'}$, it holds that either $\mathbf{e_0 \neq e'_0}$ or $\mathbf{e_1 \neq e'_1}$ (because $\mathbf{e_0 \oplus e_1 = e}$ and $\mathbf{e'_0 \oplus e'_1 = e'}$).
  - Thus, we will obtain two different continuations for at least one of the statements, and from the special soundness of a single protocol it is possible to compute a witness for that statement, which is also a witness for the OR statement.
- Honest verifier ZK
  - Can choose both $\mathbf{e_0,e_1}$, so no problem
- Note: it is possible to prove an OR of different statements using different protocols

OR of Many Statements

- Prove k out of n statements $x_1,\ldots,x_n$
  - **A** = set of indices that prover knows how to prove; the other indices are denoted as **B**
  - Use secret sharing with threshold n-k
  - Field elements 1,2,…,n, polynomial **f** with free coefficient **s**
  - Share of **s** for party $\mathbf{P_i}$: **f**(i)
- Prover
  - For every $\mathbf{i}\in\mathbf{B}$, prover generates $(\mathbf{a_i},\mathbf{e_i},\mathbf{z_i})$ using SIM
  - For every $\mathbf{j}\in\mathbf{A}$, prover generates $\mathbf{a_j}$ as in protocol
  - Prover sends $(\mathbf{a_1},\ldots,\mathbf{a_n})$

OR of Many Statements

- Prover sent $(\mathbf{a_1,\ldots,a_n})$
- Verifier sends a random field element $e \in F$
- Prover finds the polynomial f of degree n-k passing through all (i,e_i) and $(0,e)$ (for $i \in B$)
  - The prover computes $\mathbf{e_j = f}(j)$ for every $\mathbf{j \in A}$
  - The prover computes $\mathbf{z_j}$ as in the protocol, based on transcript $\mathbf{a_j, e_j}$
- Soundness follows because there are |F| possible vectors and the prover can only answer one

General Compound Statements

- This can be generalized to any monotone formula (meaning that the formula contains AND/OR but no negations)
  - See Cramer, Damgård, Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols, CRYPTO'94.

ZK from Sigma Protocols

- A tool: commitment schemes
- Enables to commit to a chosen value while keeping 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** sends $(\mathbf{e},\mathbf{r})$
  - **P** checks that $c=\mathbf{Com}_{\alpha}(\mathbf{e};\mathbf{r})$ and if yes sends a reply **z**
  - **V** accepts based on $(\mathbf{x},\mathbf{a},\mathbf{e},\mathbf{z})$

ZK from Sigma Protocols

- Soundness:
  - The perfectly hiding commitment reveals nothing about **e** and so soundness is preserved

- Zero knowledge
  - In order to simulate:
    - 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…

ZK from Sigma Protocols

- Question
  - If computational soundness suffices, can we use a computationally-hiding commitment scheme?
- No:
  - Try to prove that cheating in the proof involves distinguishing commitments
  - Receive a random commitment, and see if P^* can cheat
    - The reduction fails because we only know if P^* cheated after we opened the commitment

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 $\mathbf{k = (r\text{-}s)/(y\text{-}x) \bmod q}$

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**

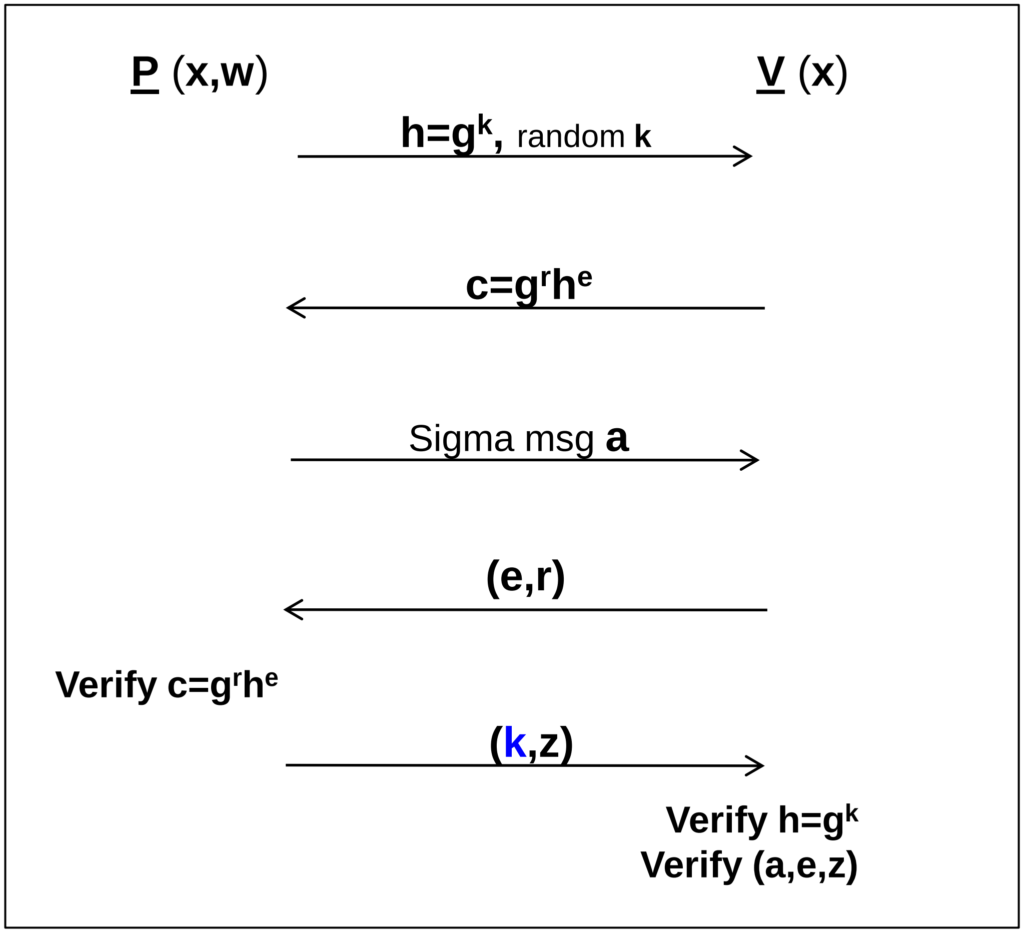
ZKPOK from Sigma Protocols

- 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
- 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=$\mathbf{Com}_{\alpha}(\mathbf{e};\mathbf{r})$
  - **P** sends a message **a**
  - **V** sends (**e**,**r**)
  - **P** checks that c=$\mathbf{Com}_{\alpha}(\mathbf{e};\mathbf{r})$ and if yes sends the **trapdoor** for the commitment and **z**
  - **V** accepts if the **trapdoor** is correct and (**x**,**a**,**e**,**z**) is accepting

ZKPOK from Sigma Protocols



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 encryption definition $\mathbf{u=g^r}$, $\mathbf{v=h^r \cdot m}$
  - 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

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