Advanced Topics in Cryptography

Lecture 2

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 $x \notin L$, for any $P^*$
    - Computational soundness: only holds when $P^*$ 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 $P^*$, a $w$ such that $(x,w) \in R$, with the same probability that $P^*$ convinces $V$.

- Equivalently:
  - The protocol securely computes the functionality $f_{zk}((x,w),x) = (-, R(x,w))$
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 $\text{DLOG}_g(h)$
  - $P$ chooses a random $r$ and sends $a = g^r$ to $V$
  - $V$ sends $P$ a random $e \in \{0, 1\}$
  - $P$ sends $z = r + ew \mod q$ to $V$
  - $V$ checks that $g^z = ah^e$

Completeness

- $g^z = g^{r+ew} = g^r(g^w)^e = ah^e$
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, $g^zh^{-e} = g^{z'}h^{-e'}$ and $g^{z-z'} = h^{e-e'}$

Therefore $h = g^{(z-z')/(e-e')}$

That is: $\text{DLOG}g(h) = (z-z')/(e-e')$

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 $a = g^zh^{-e}$.
  - Clearly, $(a, e, z)$ have the same distribution as in a real run, and $g^z = ah^e$.

- 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 $(x, w) \in \mathbb{R}$

- **Protocol:**
  - $P$ sends a message $a$
  - $V$ sends a random $t$-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 $e \neq e'$ outputs $w$ s.t. $(x,w) \in R$

- **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
  \[ (g,h,u,v) \in R \text{ iff there exists } w \text{ s.t. } u=g^w \text{ and } v = h^w \]
  - Useful in many protocols
- This is a proof of membership, not of knowledge

- Protocol
  - P chooses a random \( r \) and sends \( a=g^r, \ b=h^r \)
  - V sends a random \( e \)
  - P sends \( z=r+ew \mod q \)
  - V checks that \( g^z=a^e v^e, \ h^z=b^e v^e \)
**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
    - \(a = g^zu^{-e}\)
    - \(b = h^zv^{-e}\)
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 $P^*$ convinces $V$ and the probability that an extractor $K$ obtains a witness is at most $2^{-t}$
  - 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
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
The template for proving $x_0$ or $x_1$:

- $P$ sends two first messages $(a_0, a_1)$
- $V$ sends a single challenge $e$

$P$ replies with

- Two challenges $e_0, e_1$ s.t. $e_0 \oplus e_1 = e$
- Two final messages $z_0, z_1$

$V$ accepts if $e_0 \oplus e_1 = e$ and $(a_0, e_0, z_0), (a_1, e_1, z_1)$ are both accepting

How does this work?
OR of Sigma Protocols

- **P** sends two first messages \((a_0, a_1)\)
  - Suppose that **P** has a witness for \(x_0\) (but not for \(x_1\))
  - **P** chooses a random \(e_1\) and runs SIM to get \((a_1, e_1, z_1)\)
  - **P** sends \((a_0, a_1)\)
- **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 \(z_1\) and can compute \(z_0\) using the witness
- **Soundness**
  - If **P** doesn’t know a witness for \(x_1\), he can only answer for a single \(e_1\)
  - This means that \(e\) defines a single challenge \(e_0\), like in a regular proof
OR of Sigma Protocols

- Special soundness
  - Relative to first message \((a_0, a_1)\), and two different \(e, e'\), it holds that either \(e_0 \neq e'_0\) or \(e_1 \neq e'_1\) (because \(e_0 \oplus e_1 = e\) and \(e'_0 \oplus e'_1 = e'\)).
  - 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 \(e_0, e_1\), 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 $P_i$: $f(i)$

- **Prover**
  - For every $i \in B$, prover generates $(a_i, e_i, z_i)$ using SIM
  - For every $j \in A$, prover generates $a_j$ as in protocol
  - Prover sends $(a_1, \ldots, a_n)$
OR of Many Statements

- Prover sent \((a_1, \ldots, a_n)\)
- 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 \(e_j = f(j)\) for every \(j \in A\)
  - The prover computes \(z_j\) as in the protocol, based on transcript \(a_j, e_j\)
- 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.