Theory of Cryptography | **Lecture 18** **Proofs of Knowledge** | **Instructor: Chris Peikert** **Scribe: Joel Odom**

1 Proofs of Knowledge

Up to this point we have studied zero-knowledge proofs that are capable of proving whether a statement is "true" or "false," i.e., whether a statement lies in or out of a language. Sometimes, though, the notion of membership in a language does not capture the claim that the prover wants to demonstrate. For example, if the statement is true, might it be that *anyone* can give an acceptable proof? The definition of an interactive proof doesn't give us much insight into this question; it only says that *false* theorems can't be proved (even by an unbounded prover). Here we will see that it is sometimes useful to have a *proof of knowledge*, in which a prover convinces a verifier that the prover "knows why" a statement is true.

Take, for example, the application we considered at the end of the last lecture. Alice wanted to convince Bob, in zero knowledge, that she is a CIA agent — or more precisely, that the CIA has signed the message "Alice is a CIA agent" under its public key. (ZK is useful here because it would make the proof nontransferable by Bob.) How can we go about proving this in ZK? If Alice just reveals the signature, that conveys knowledge to Bob that he could not have simulated himself (and is transferable). If Alice proves merely that there *exists* a valid signature (under the CIA's public key) of the message "Alice is a CIA agent," this doesn't necessarily prove anything meaningful, because by the completeness of the signature scheme, there exists a signature for *every* message! What Alice really wants to prove is that she *knows* a valid signature of the appropriate message.

1.1 Example Protocol

What we want to capture is not only the truth of a proposition ($x \in L$), but also that the prover "possesses" a suitable witness to this fact. We demonstrate this concept by an example protocol which proves knowledge of a discrete logarithm in a cyclic group.

Protocol 1.1. Let $G = \langle g \rangle$ be a cyclic group of known order q with known generator g, and let $x \in G$ be an arbitrary group element having discrete logarithm $w = \log_g(x)$. The common input to P and V is x, and P is additionally given w.

1. $P(x, w)$ selects a random $r \leftarrow \mathbb{Z}_q$ and sends $z = g^r$ to V.
2. $V(x)$ sends a random challenge bit $b \in \{0, 1\}$ to P. (Here V is challenging the prover to reply with the discrete log of z, or of $z \cdot x$.)
3. P responds with $a = r + b \cdot w \in \mathbb{Z}_q$.
4. V accepts if $g^a = z \cdot x^b$.

Schematically,

$$\begin{array}{ccc}
P(x, w = \log_g(y)) & & V(x) \\
r \leftarrow \mathbb{Z}_q & \xrightarrow{z = g^r} & \\
 & & b \leftarrow \{0,1\} \\
 & \xleftarrow{b} & \\
 & \xrightarrow{a = r + bw} & \text{accept iff } g^a = z \cdot x^b
\end{array}$$

Is the protocol *complete*? Yes: if P and V act as described, then $g^a = g^{r+bw} = g^r \cdot (g^w)^b = z \cdot x^b$. Is the protocol *zero-knowledge*? Let V^* be a (possibly malicious) nuppt verifier. We define a simulator $\mathcal{S}^{V^*}(x)$:

1. Choose a random bit $b \leftarrow \{0,1\}$ and a random group element $a \leftarrow G$. (This can be done by choosing $t \leftarrow \mathbb{Z}_q$ and letting $a = g^t$.)
2. Send $z = g^a / x^b$ to V^* (run with fresh random coins) and get back a challenge bit b^*. (If V^* responds with a malformed message or aborts, just output the view so far.)
3. If $b^* = b$, complete the view using a as the prover's last message, otherwise rewind V^* and repeat the simulation (up to n total iterations).

It is relatively easy to show that $\mathcal{S}$ reproduces V^*'s view, up to negligible statistical distance. (Briefly, this is because the value z calculated by $\mathcal{S}$ is statistically independent of its bit b, so $b^* = b$ with probability $1/2$.)

Is the protocol *sound*? Note that *every* $x \in G$ has a discrete logarithm, so the notion of soundness is not particularly meaningful here! A separate question is whether the protocol gives evidence that the prover possesses *knowledge* of the discrete logarithm x. An informal inspection of the protocol seems to indicate that it does: if the prover can answer both challenges, then it is prepared to reply with both $\log_g(z)$ and $\log_g(z \cdot x) = \log_g(z) + \log_g(x)$, whose difference is exactly $\log_g(x)$. But we need to formalize some notions before we can come to any definite conclusion.

1.2 Definition of an NP-Relation

Definition 1.2. An NP-*relation* $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is given by a deterministic algorithm $W(\cdot,\cdot)$ that runs in time polynomial in the length of its first input. The relation is

$$R = \{(x,w) : W(x,w) \text{ accepts}\}.$$

The associated NP-language $L_R = \{x : \exists\, w \text{ such that } W(x,w) \text{ accepts}\}$. The witness set for an $x \in \{0,1\}^*$ is $R(x) = \{w : W(x,w) = 1\}$.

That is, the NP-relation R is just the set of theorem-witness pairs (x,w) that are accepted by the verifier W. It is easily seen that the language L_R associated with the relation is an NP language.

Example 1.3. For the discrete logarithm problem in G, the natural relation is

$$R = \{(x \in G, w \in \mathbb{Z}_q) : g^w = x\}.$$

1.3 Definition of a Proof of Knowledge

We can now define a proof of knowledge for a relation R. The definition captures the intuition that from *any* (possibly cheating) prover P^* that is able to convince the verifier with good enough probability on a statement $x \in L_R$, there is a way to extract a valid witness for x from P^* with a related (not too small) probability.

Definition 1.4. An interactive proof system (P,V) is a *proof of knowledge* with *knowledge error* $\kappa \in [0,1]$ for an NP-relation R if there exists an nuppt oracle machine $K^{\cdot}$ (the *knowledge extractor*) such that for any $x \in L_R$, and for any (possibly unbounded) P^* for which $p_x^* = \Pr[\text{out}_V[P^* \leftrightarrow V(x)] = 1] > \kappa$, we have

$$\Pr[K^{P^*}(x) \in R(x)] \geq \text{poly}(p_x^* - \kappa).$$

In words, the probability that the knowledge extractor K finds a valid witness for x using its access to prover P^* is at least polynomially related to the probability p_x^* that P^* convinces the honest verifier on x, less some knowledge error.

Remark 1.5. A number of remarks on this definition are appropriate.

1. We don't make any restriction on the kind of prover P^* under consideration: it could be unbounded and/or malicious.

2. The definition is *per instance* x: if P^* convinces V that a *particular* x is true, then it should know a witness for *that* x.

3. Notice that in the discrete logarithm protocol above, there is a trivial cheating prover that convinces the verifier with probability $\frac{1}{2}$. (This is the prover that prepares an answer for a known b and succeeds only if the verifier issues b as its challenge bit.) There is no hope of extracting any witness from this prover, without just solving the discrete log problem directly. Just as with soundness error, the knowledge error κ captures the "threshold" probability with which a cheating prover P^* must convince the verifier before it can be said to know a valid witness.

4. Because the relation R is efficiently checkable, we can run K^{P^*} many times and output any valid witness for x that is found in any execution. The expected running time of this is $1/\operatorname{poly}(p_x^* - \kappa)$. If $p_x^* - \kappa$ is at least inverse-polynomial in the security parameter, then a witness is extracted in expected polynomial time.

5. The definition is not formally related to soundness (i.e., neither property necessarily implies the other). However, notice that the definition only applies to $x \in L_R$, because otherwise the set $R(x)$ is empty and therefore K could never output a valid witness.

1.4 Knowledge Extractor for Discrete Log Protocol

Theorem 1.6. *Protocol 1.1 is a proof of knowledge for the discrete logarithm relation, with knowledge error* $\kappa = 1/2$.

Proof. We construct a knowledge extractor $K^{P^*}(x)$ that works as follows:

1. Run P^* and receive a message $z \in G$ from P^*.

2. Send challenge bit $b = 0$ to P^* and receive an answer a_0.

3. Rewind P^* to its state before Step 2. Send challenge bit $b = 1$ to P^*, and receive an answer a_1.

4. Output $a_1 - a_0$.

Note that *if* a_0 and a_1 are both correct answers to the respective challenges on z, then $a_1 - a_0 = \log_g(z \cdot x) - \log_g(z) = \log_g(x)$, so the knowledge extractor's output is a correct witness for x.

Suppose that $\Pr[\text{out}_V[P^* \leftrightarrow V(x)] = 1] = 1/2 + \alpha$ for some positive α (otherwise, we have nothing to prove). We claim that

$$\Pr[a_0 \text{ and } a_1 \text{ correct}] \geq \alpha^3/2 = \operatorname{poly}(\alpha),$$

as required by Definition 1.4.

To prove the claim, define

$$p_z = \Pr[P^* \text{ convinces } V(x) \mid P^* \text{ outputs } z \text{ in the first message}].$$

By an averaging argument (Markov inequality), we have $\Pr_{z \leftarrow P^*}[p_z \geq \frac{1}{2} + \frac{\alpha}{2}] \geq \frac{\alpha}{2}$.

Now consider any such "good" z, where $p_z \geq \frac{1}{2} + \frac{\alpha}{2}$. Define

$$p_{z,b} = \Pr[P^* \text{ convinces } V(x) \mid z, \text{challenge } b].$$

Because $p_z = (p_{z,0} + p_{z,1})/2$ and both $p_{z,0}, p_{z,1} \leq 1$, we have both $p_{z,0}, p_{z,1} \geq \alpha$. The events that P^* returns a valid a_0 (respectively, a_1) *conditioned* on z and challenge $b = 0$ (resp., $b = 1$) are *independent*, so both occur with probability at least α^2. All together then, the probability that K outputs $\log_g(x)$ is at least $\alpha^3/2$, as claimed.

□