CMSC 858K — Advanced Topics in Cryptography April 27, 2004

Lecture 24

Lecturer: Jonathan Katz *Scribe(s): A. Anand, G. Taban, M. Cho*

1 Introduction and Review

In the previous classes, we have discussed proofs of knowledge (PoKs) and zero-knowledge (ZK) proofs. We briefly review these notions here:

ZK proofs. Zero-knowledge proofs involve a prover P trying to prove a statement to a verifier V without revealing any knowledge beyond the fact that the statement is true. For example, consider the problem of proving membership in an NP language L, (e.g., graph Hamiltonicity, 3-coloring, etc.). A ZK proof protects against a cheating prover, in the sense that if a prover tries to give a proof for an $x \notin L$ the verifier will reject the proof with all but negligible probability. Further, a ZK proof protects against a cheating verifier, in the sense that it ensures that the verifier (informally) does not learn anything from a proof that $x \in L$ other than the fact that $x \in L$.

A ZK proof system requires the existence of a simulator who can simulate a transcript of the protocol execution without knowing the witness to the statement. As we have seen, a simulator typically does this by rewinding the verifier to a prior state and then trying to continue the simulation until it comes up with a valid transcript.

Proofs of knowledge. Proofs of knowledge are protocols in which the prover actually proves that he "knows" a *witness*. In addition to the formal sense in which this holds (i.e., via the additional requirement that there exists a *knowledge extractor* who can extract a valid witness from any prover who succeeds in giving a correct proof with high-enough probability), there are also examples where membership is "easy" to determine but proving knowledge of a witness might be hard. The classic example is the case of a cyclic group G with generator g in which the discrete logarithm is hard. Here, for a given $h \in G$ it might be easy to determine that, in fact, h is an element of G and therefore there *exists* an x such that $g^x = h$; however, we may additionally want a prover to prove that he *knows* this x.

Thinking about it a bit, the ZK property and the PoK property seem to be at odds: a ZK proof requires a simulator who (typically) rewinds a cheating verifier to simulate a proof without knowing a witness, while a PoK requires a knowledge extractor who (typically) rewinds a cheating prover to extract a witness. And indeed, we will see in what follows that the approach to constructing a constant-round ZK proof from the previous lecture (namely, forcing the verifier to commit to its challenges in advance) seemingly destroys the PoK property. To obtain a constant-round protocol which is *both* zero-knowledge *and* admits a knowledge extractor we will consider a relaxation of proofs called *arguments* in which the soundness condition is only required to hold only with respect to *polynomial-time* cheating provers (recall that *proofs* require the soundness condition to hold even for *all-powerful* provers).[1]

[1] In fact, constant-round zero-knowledge *proofs* of *knowledge* for all of NP are possible, but we will not see an example in this course.

2 Review: A ZK PoK Protocol

In this section, we review a ZK PoK protocol for graph Hamiltonicity from the previous lecture; see Figure 1. In the figure, G is a graph and C represents a Hamilton cycle in G. In the previous lecture, we showed that this protocol is zero-knowledge and a proof of knowledge with soundness error 1/2. We now informally recall the proofs of these properties:

- To prove the zero-knowledge property, we considered the following simulator: it guesses in advance a bit $\tilde{c}$. If $\tilde{c} = 0$, it commits to a (random permutation of) the adjacency matrix for G; if $\tilde{c} = 1$, it commits to a (randomly-permuted) cycle graph. It sends the resulting commitment to the verifier who responds with a challenge c. If $\tilde{c} = c$ then the prover responds in the natural way and is done. Otherwise, the simulator rewinds and tries again. Since $\tilde{c} = c$ with probability 1/2 each time, if the simulator rewinds k times it will succeed at least once with all but negligible probability. (The rest of the proof is then devoted to showing that the simulation is computationally indistinguishable from a real execution.)
- To show the knowledge extraction property, assume we have a prover who gives a correct proof with probability better than 1/2. This implies that the prover responds correctly for both possible values of c. So we simply rewind the prover, send him both possible challenges, and use the two (correct) responses to compute a Hamiltonian cycle in G.

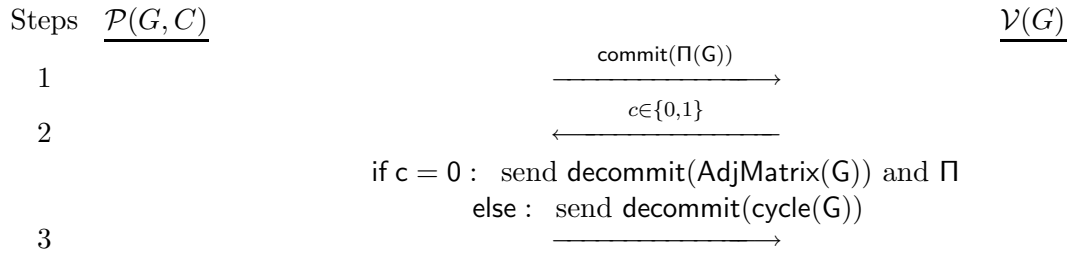

Figure 1: A ZK-PoK protocol.

We also noted that we could run the above protocol k times *sequentially* to reduce the soundness error to 2^{-k}. Doing so maintains the ZK property of the construction, and we showed that the resulting protocol was also a PoK with the claimed soundness error.

2.1 A Parallel Execution of the Protocol

What happens if we run the original protocol k times *in parallel*? The PoK property remains intact:

Claim 1 *Running the above protocol k times in parallel results in a PoK with soundness error 2^{-k}.*

Proof To prove the above claim, we consider a knowledge extractor E which works as follows.

- Obtain a first message from the prover P.
- Pick a random challenge $c_1 \in \{0,1\}^k$, and send this challenge to the prover. If the prover does not answer correctly, stop.

- Otherwise, repeatedly choose a random $c_2 \in \{0,1\}^k$, rewind the prover, and send c_2 to the prover until the prover answers correctly a second time and $c_2 \neq c_1$. In parallel, perform an exhaustive search for a Hamiltonian cycle in G and stop once one is found or when it is determined that no such cycle exists.

We now show two facts: (1) E runs in expected polynomial time (this assumes that P runs in expected polynomial time), and (2) if the probability p that P gives a successful proof is greater than 2^{-k} then E succeeds in computing a Hamiltonian cycle in G with probability p.

To prove the first statement, note that when $p = 0$ then E clearly runs in (strict) polynomial time. So consider the case that $p > 2^{-k}$. Let $n > 1$ be the number of challenges for which P answers correctly (i.e., $p = \frac{n}{2^k}$). When P does not respond correctly to the first challenge c_1 (with happens with probability $1-p$), then E runs in (strict) polynomial time. When P responds correctly to c_1, then the expected number of times E rewinds P until it finds a second (different) c_2 for which P answers correctly is $(\frac{n-1}{2^k})^{-1}$. Overall, then, the expected running time of E is given by:

$$(1-p) \cdot \mathsf{poly}(k) + \frac{n}{2^k} \cdot \frac{2^k}{n-1} \cdot \mathsf{poly}(k) = \mathsf{poly}(k)$$

(we use $\mathsf{poly}(k)$ here to refer to an arbitrary polynomial). The last case remaining is when $p = 2^{-k}$ (i.e., P responds correctly to exactly one challenge). As before, when P does not respond correctly to c_1 then E runs in strict polynomial time. When P responds correctly to c_1, then E will *never* find a $c_2 \neq c_1$ for which P answers correctly again. But, P is also running an exhaustive search for a Hamiltonian cycle in G and we assume this takes at most $2^k \cdot \mathsf{poly}(k)$ steps.[2] So, the expected running time of E is given by:

$$(1-2^{-k}) \cdot \mathsf{poly}(k) = 2^{-k} \cdot 2^k \cdot \mathsf{poly}(k) = \mathsf{poly}(k).$$

We now move on to a proof of the second statement. Note that P responds correctly to challenge c_1 with probability exactly p. We claim that as long as $p > 2^{-k}$, then E *always* computes a Hamiltonian cycle whenever P responds correctly to c_1. To see this, note first that $p > 2^{-k}$ implies that G has a Hamiltonian cycle. (We assume here that the commitments sent in the first round are perfectly binding.) When P responds correctly to c_1, then E stops its execution when either (1) it finds a $c_2 \neq c_2$ for which P also responds correctly, or (2) it completes its exhaustive search for a cycle. In either of these cases, E can then compute the desired Hamiltonian cycle. ■

Unfortunately, k-fold parallel repetition of the protocol seems to destroy the zero-knowledge property. At a minimum, the type of simulator we considered before does not work, and no simulator is known which would prove the ZK property. In particular, if we consider the simulation strategy as before then we would have a simulator who tries to guess $\tilde{c} = c$ in advance: but now $c \in \{0,1\}^k$ and so the probability of guessing correctly is negligible! (And so even repeatedly guessing polynomially-many times will not help.)

2.2 Further Modifications?

We can try to recover the ZK property (for the protocol obtained via k-fold parallel repetition of the original, 3-round protocol) by using the Goldreich-Kahan technique, in which the verifier is forced to commit (using a perfectly-hiding commitment scheme) to their challenge vector c *in*

[2] If one is unhappy with this assumption, note that exhaustive search takes at most $k! \cdot \mathsf{poly}(k)$ time and so by running the protocol $\log k! = O(k \log k)$ times in parallel the proof goes through.

advance. For future reference, let us call the round in which the verifier commits to c "round 0". This modification will indeed result in a zero-knowledge protocol... but the modified protocol no longer appears to be a proof of knowledge! Indeed, the very fact that the verifier is forced to commit in advance to c means that the knowledge extraction strategy outlined earlier will no longer work: even E cannot break the commitment scheme, and so it cannot decommit to $c_2 \neq c_1$ unless it rewinds all the way back to round 0 and sends a new set of commitments, in which case P might change its round-1 message!

Somewhat paradoxically(?), it is possible to design a constant-round ZK-PoK. Instead of showing this, however, we consider a relaxation of the notion of a "proof" and show how to achieve both knowledge extraction and zero-knowledge subject to this relaxation.

3 Zero-Knowledge *Arguments* of Knowledge

As discussed in the introduction, an *argument* requires soundness to hold only for provers running in polynomial time (whereas a *proof* requires soundness to hold even for *all-powerful* provers). (An *argument of knowledge* is defined similarly, such that a knowledge extractor is only required to extract a witness from provers running in polynomial time.) We will now show a construction of a constant-round zero-knowledge argument of knowledge due to Feige and Shamir. Our discussion will be somewhat informal and "high-level"; the reader is referred to [2, 1] for further details.

Let f be a one-way function. The basic protocol proceeds in 6 rounds (it is possible to "collapse" this to a 4-round protocol, but the proof is less intuitive in this case so we do not present this extension). Let L be an NP language; we describe the protocol assuming the honest prover is proving that $x \in L$ given some witness w.

Rounds 1–3: The verifier chooses x_1, x_2 at random, computes $y_1 = f(x_1)$ and $y_2 = f(x_2)$, and sends y_1, y_2 to the prover. The verifier then gives a 3-round witness-indistinguishable (WI) proof of knowledge (with negligible soundness error) of "$f^{-1}(y_1)$ or $f^{-1}(y_2)$". Note that the verifier can do this efficiently, since it knows witnesses x_1, x_2 (in fact, only one witness is needed). We comment briefly below on the existence of 3-round WI proofs of knowledge.

Rounds 4–6: If the proof given by the verifier fails, the prover simply aborts. Otherwise, the prover gives a 3-round witness-indistinguishable proof of knowledge of "$f^{-1}(y_1)$ or $f^{-1}(y_2)$ or $x \in L$". Note that the prover can do this efficiently since it has a witness that $x \in L$.

In the previous lecture we have already shown a 3-round WI proof of knowledge with negligible soundness error: the k-fold parallel repetition of the basic, 3-round protocol (with soundness error 1/2). We proved explicitly last time that this protocol is a proof of knowledge with negligible soundness error. The fact that it is witness indistinguishable follows from the facts that: (1) as proved last time, the basic 3-round protocol is zero-knowledge; (2) zero-knowledge implies witness indistinguishability, and hence the basic, 3-round protocol is witness indistinguishable; finally (3) we saw in an earlier lecture that witness indistinguishability is preserved under parallel repetition.

We now discuss, informally, why this protocol is both zero-knowledge and an argument of knowledge. (Note that it is certainly not a proof, since an all-powerful prover can invert f and then give a successful proof even when $x \notin L$.)

Zero-knowledge. We show a simulator demonstrating that the protocol is zero knowledge (although no formal proof will be given). The simulator, on input $x \in L$ but *without* a witness, proceeds as follows:

Rounds 1–3: Interact with the (possibly cheating verifier) V^* to obtain values y_1, y_2 and a transcript T of the first three rounds. If V^* does not successfully complete its proof of knowledge, then the simulator can abort (just like the real prover would) and the simulation is done by simply outputting T. Otherwise, if V^* does give a successful proof of knowledge, the simulator runs the knowledge extractor for this 3-round proof to obtain a witness for "$f^{-1}(y_1)$ or $f^{-1}(y_2)$". (If this extraction fails, then the entire simulation is aborted.) Note that, assuming a witness is extracted, this gives an x such that either $f(x) = y_1$ or $f(x) = y_2$.

Rounds 4–6: Continuing in an execution with V^* with initial transcript T, the simulator now simply gives a WI proof of knowledge of "$f^{-1}(y_1)$ or $f^{-1}(y_2)$ or $x \in L$". The key point is that the simulator can do this without any further rewinding since it does indeed know a witness for this statement.

A proof that this results in a simulation which is computationally-indistinguishable from a real execution is relatively straightforward given all the machinery at our disposal. The initial portion of the transcript (i.e., the transcript T of the first 3 rounds) is identically distributed to the first 3 rounds in a real execution of the protocol. If V^* gives a successful proof in rounds 1–3, knowledge extraction will succeed with all but negligible probability. Assuming this to be the case, then the last 3 rounds in the simulation consist of a WI proof using the witness x extracted in the previous phase; in a real execution, these last 3 rounds would be a WI proof using a witness for $x \in L$ (the same statement is being proved in either case). But witness indistinguishability of the proof system used in rounds 4–6 implies that these two resulting transcripts are indeed computationally indistinguishable.

Argument of knowledge. We next argue that the protocol is an argument of knowledge, which will imply soundness (for poly-time provers). The knowledge extractor E is the obvious one: simply run the knowledge extractor for the WI proof of knowledge given by the prover in rounds 4–6. The analysis of E is the tricky part. A proof that E extracts a witness for $x \in L$ follows from two claims along the following lines:

Claim 2 *(Informal) E extracts a witness for the statement "$f^{-1}(y_1)$ or "$f^{-1}(y_2)$ or $x \in L$" with sufficiently-high probability ("sufficiently-high probability" here simply refers to the probability required by the definition of an argument/proof of knowledge).*

This claim follows immediately from the fact that the proof given in rounds 4–6 is a proof of knowledge. Next:

Claim 3 *E extracts a witness for "$f^{-1}(y_1)$ of $f^{-1}(y_2)$" with only negligible probability.*

Thus, whenever E extracts a witness for "$f^{-1}(y_1)$ or "$f^{-1}(y_2)$ or $x \in L$" (which is does sufficiently-often, by the previous claim) it in fact extracts a witness, as desired, for $x \in L$ except with negligible probability.

The proof of the above claim is more difficult, and proceeds in the following steps:

1. Say the probability of extracting a witness for "$f^{-1}(y_1)$ of $f^{-1}(y_2)$" is p. Let p_b denote the probability of extracting a witness for $f^{-1}(y_b)$. Clearly, either $p_1 \geq p/2$ or $p_2 \geq p/2$; assume the former without loss of generality.

2. The above refers to the probability of extraction when, in rounds 1–3, the verifier (i.e., the knowledge extractor playing the role of the verifier) gives a WI proof using witnesses for both

$f^{-1}(y_1)$ and $f^{-1}(y_2)$. In fact, it is enough to use only a witness for $f^{-1}(y_2)$ when giving this proof. Witness indistinguishability of the proof system in rounds 1–3 can be used to show that the probability of extracting a witness for $f^{-1}(x_1)$ is not affected by more than a negligible amount, and so the probability of extracting a witness for $f^{-1}(x_1)$ in this case is negligibly close to $p_1 \geq p/2$.

3. We show that if p_1 is non-negligible then we can use the cheating prover to invert the one-way function f as follows: Given a point y_1, choose a random x_2, compute $y_2 = f(x_2)$, and then run the above experiment (giving the appropriate WI proof in rounds 1–3 and then extracting in rounds 4–6). By what we have said above, the probability that we extract a witness for $f^{-1}(y_1)$ is negligibly-close to p_1. But extracting a witness for $f^{-1}(y_1)$ exactly means that we have computed $f^{-1}(y_1)$! Since this cannot occur with more than negligible probability, we conclude that p_1 (and hence p) is negligible.

We remark that the use of *two* values y_1, y_2 in the argument system is essential to the above proof. If we had used only a single value y_1, then in order to reach the extraction phase (i.e., rounds 4–5) we would need to successfully complete the proof in rounds 1–3. . . but since this is not a ZK proof we actually need a witness to do so. But if we have the witness $f^{-1}(y_1)$ in rounds 1–3, then extracting this witness in rounds 4–6 is not a contradiction! The nice feature of the Feige-Shamir system is that it allows the extractor to "switch" witnesses in rounds 1–3, and hence derive the desired contradiction.

Very detailed and formal proofs of the above properties (as opposed to the hand-waving proof sketches above) are given in [1].

4 Proof of Knowledge for Number Theoretic Arguments

The preceding ends our discussion (for now, anyway) of generic proofs/arguments for languages in NP. We now focus in *efficient* proofs of knowledge for *specific* number-theoretic relations. In particular, we show a proof of knowledge of discrete logarithms. Let G be a finite, cyclic group of prime order q. Let g be a generator for g, and let $h \in G$ be arbitrary. Consider the following protocol in which a prover P tries to convince a verifier that P indeed known the discrete logarithm $\log_g h$ (i.e., an x such that $g^x = h$):

Steps	$P(g, x, h = g^x)$		$V(g, h)$
1	$r \leftarrow \mathbb{Z}_q$		
2		$\xrightarrow{A = g^r}$	
3		$\xleftarrow{c}$	$c \leftarrow \mathbb{Z}_q$
4	$\gamma = cx + r \bmod q$	$\xrightarrow{\gamma}$	verify $g^\gamma = h^c \cdot A$

Figure 2: A PoK for discrete logarithms.

It is not hard to see that the above protocol is complete: if the prover is honest then

$$g^\gamma = g^{cx+r} = g^{cx} \cdot g^r = h^c \cdot A\,.$$

We now sketch why this protocol is a proof of knowledge. The knowledge extractor E will be similar to the one defined earlier: if P responds correctly to an initial challenge c then E will

repeatedly rewind P until it finds a different challenge c' for which P also responds correctly.[3] If E can find two such challenges c, c', we claim that it can then compute the desired discrete logarithm. Indeed, this means that E has values $A, c, c', \gamma, \gamma'$ such that $g^\gamma = h^c A$ and $g^{\gamma'} = h^{c'} A$. We claim that $\log_g h = (\gamma - \gamma')(c - c')^{-1} \bmod q$ (which can be computed easily; note that $c - c' \neq 0$ by construction). Indeed:

$$\begin{aligned} g^{\frac{\gamma-\gamma'}{c-c'}} &= \left(g^{\gamma-\gamma'}\right)^{1/(c-c')} \\ &= \left(g^\gamma g^{-\gamma'}\right)^{1/(c-c')} \\ &= \left(\frac{h^c A}{h^{c'} A}\right)^{1/(c-c')} \\ &= \left(h^{c-c'}\right)^{1/(c-c')} = h\,, \end{aligned}$$

as claimed.

References

[1] U. Feige. Alternative Models for Zero Knowledge Interactive Proofs. PhD thesis, Weizmann Institute of Science, 1990. Available at http://www.wisdom.weizmann.ac.il/~feige.

[2] U. Feige and A. Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. Crypto '89.

[3] O. Goldreich. *Foundations of Cryptography, Vol 1: Basic Tools*. Cambridge University Press, 2001.

[4] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 1996.

[3] As previously, E will also have to perform an exhaustive search for $\log_g h$ in parallel so that it doesn't run "too long". Details omitted.