600.641 Special Topics in Theoretical Cryptography 3/5/2007

Lecture 1: Honest Verifier ZK and Fiat-Shamir

Instructor: Susan Hohenberger *Scribe: Aylin Ryan*

1 Announcements

Problem Set 2 will be posted on our website tonight, and the assignment is due on Wednesday March 21st. We are encouraged to work together, although the fourth problem is entirely individual as it asks for our proposals for our final project.

1.1 Final Project Proposals

One of the goals of this seminar is to analyze a paper from a top conference and provide your interpretation and insight. Look at papers from: AsiaCrypt, Crypto, EuroCrypt, of TCC (Theory of Cryptography Conference) from 2002 or later and select three topics that you find interesting and you would like to pursue. The fourth problem in the Problem Set 2 asks you to list these topics in the order that you find them most interesting .

The final project involves a presentation to the class. When you are working on the project, look at ways you could extend or expand upon these papers. All papers and presentations will have to be done before the last day of class, April 24. Talks will take place during lecture and each person is expected to speak for about 20 minutes.

2 Honest Verifier Zero Knowledge Proofs

Let's look at real systems and tools to build efficient systems.

Zero-Knowledge says that *no* malicious verifier cannot extract additional knowledge from the prover. Recall the definition of a zero-knowledge proof.

$$\forall\ \mathsf{V}^*\ \ \exists\ \mathsf{S}\ \ \forall\ x \in L, VIEW_{\mathsf{P},\mathsf{V}^*}(x) \approx \mathsf{S}(x)$$

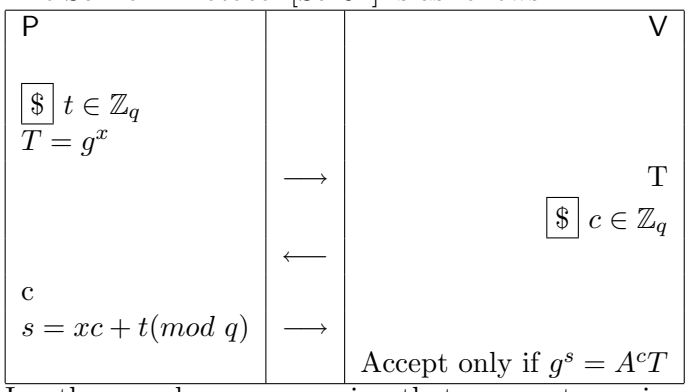
Now, suppose that instead of wanting to prove a *statement* such as $x \in L$, one wants to prove *knowledge* of some value. For example, for a given public key, I might want to prove to you that I know the corresponding secret key. Using the notation for expressing proofs of knowledge described in prior lectures, let's consider the following common case.

Schnorr [Sch91] described a protocol for proving Knowledge of Discrete Logarithm in formal symbolic notation is as follows:

$$PK\{(\alpha) : A = g^{\alpha}\}$$

where α is the information that the Prover demonstrates knowledge of and (A, g) is the public information (known by both the Prover and the Verifier). Recall that formally, Greek letters represent secret information whereas Roman letters represent public information. In the actual protocol, Greek letters are replaced with lowercase letters.

The Schnorr Protocol [Sch91] is as follows:



In other words, we are saying that we must convince some authenticating entity that you have the private key, but you need to know who is issuing the key (for instance, this could be in relation to your passport). If you need to prove your identity, you can do it, but in doing so, are you giving any secret information to this entity? Is this zero knowledge? This entire scenario could be equivalent to convincing someone that you're a Johns Hopkins student.

But, is this protocol describe above zero-knowledge? Can we describe a simulator for it? We've had a hard time providing a simulator for this. Indeed, try to think how one might work. The simulator must send some commitment, and the verifier must submit some challenge, and the simulator must make some response. For a given commitment and challenge value, there is only one valid response in the above protocol. (Notice that if the simulator tries to rewind the verifier after seeing its challenge *and then change its commitment*, it gets into trouble, because the cheating verifier might change his challenge based on the new commitment ....)

Although we might not know how to prove this protocol is (fully) zero-knowledge, note that if the verifier behaves honestly according to the specified protocol, then we could describe a simulator to simulate those conversations. This is called *honest-verifier zero knowledge*. Is this an interesting concept? Well, let's see.

2.1 Defining and Realizing HVZK and HVZK Proofs of Knowledge

There exists a simulator such that $\forall\ x \in L, VIEW_{\mathsf{P},\mathsf{V}}(x) \approx \mathsf{S}(x)$. Note that here we only consider conversations between the honest prover and the honest verifier!

Now let's describe such a simulator for Schnorr's protcol above.

1. Choose random $c, s \in \mathbb{Z}_q$.
2. Compute $T = g^s/A^c$.
3. Output the conversation in the order (T, c, s).

Notice that T is uniformly random in the group generated by g, c is independently random in $\mathbb{Z}_q$, and the response s is correct. Thus, by inspection we see that this simulator creates conversations according to the same distribution as those between the honest prover and the honest verifier.

If we could force V to follow this protocol honestly and choose its challenge randomly (without considering T), then zero knowledge would follow. This would be following the same logic as if we were to prove something against an honest guy, and then force the guy to be honest.

Why is Honest-Verifier ZK interesting? Since HVZK only protects a prover from the *honest* verifier, is it of any use? Well, here are two reasons why we think it is interesting.

In 1998, Goldreich, Sahai and Vadhan [GSV98] showed that honest verifier statistical zero-knowledge is equal to general statistical zero-knowledge. (Recall that statistical ZK states that the simulator can generate something that is statistically close to the actual distribution). In order to demonstrate this, they built a compiler to transform any HVSZK protocol into a SZK protocol, which had the following features:

1. doubles the rounds (between the prover and the verifier)
2. actual running time is still polynomial, but isnt something you want to actually implement.

This gives us some useful intuition when designing protocols. Perhaps first consider honest parties and then generalize to the malicious case. However, we also have another (perhaps more practical) reason why HVZK is interesting.

3 Fiat-Shamir Paradigm

In 1986 [FS86], Fiat and Shamir introduced a method to transforms 3-round public-coin identification schemes (such as our running example of the Schnorr protocol) into digital signature schemes. The significance of this method was that it introduced an efficient design for digital signatures with the aim to maintain a security against chosen message attacks. The Fiat-Shamir paradigm transforms a 3-round standard honest-verifier proof of knowledge with public coinsinto a non-interactive (1-round) general proof of knowledge with the use of a hash function modeled as a *random oracle*. (We'll say more about random oracles in a moment. Let's first see how the hash function is used.) We have already seen that there are zero-knowledge proofs of knowledge for all NP relations.

Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function. The Fiat-Shamir paradigm can be applied to the Schnorr protocol as follows. The prover first commits to $T = g^t$ as normal. Then the prover computes $c = H(T)$. Here the hash function replaces the response of the verifier. Finally the prover computes $s = xc + t$ as before, and sends (T, s) to the verifier. The verifier accepts iff $g^s = A^c T$.

Note that if we had wanted to design a signature scheme, instead of just a non-interactive proof of knowledge, then we could have added the message m to be signed to the hash function as $c = H(T, m)$. This is sometimes called a "signature of knowledge." This term has been along for a long time, but was recently studied formally by Chase and Lysyanskaya [CL06].

Assume that H's output is distributed uniformly at random in Z_q and that it is unpredictable.

3.1 Properties of Fiat-Shamir

3.1.1 Zero-Knowledgeness

Is this new protocol zero-knowledge? Since we are working in the random oracle model, we will give the simulator control over the outputs of the hash function. That is, we pretend that in the "real world" the prover and verifier evaluate the hash H by submitting an input x to an "oracle" and obtaining the response $H(x)$. In the "ideal world", we let the simulator take control of this oracle, so that when the verifier inputs x to the oracle, the simulator can decide (on the spot!) what value to return. The only requirements here is that the outputs are consistent (i.e., the same input always results in the same output) and the outputs are uniformly random.

So, our simulator now works as follows. Recall the statement $PK\{(\alpha) : A = g^\alpha\}$. (Remember that here the simulator does not know x such that $A = g^x$!) Instead, the simulator chooses random values $c, s \in \mathbb{Z}_q$ as before and computes $T = g^s/A^c$. The simulator then "sets" $H(T) = c$.

3.1.2 Proof of Knowledge (PoK)

Is this new protocol a proof of knowledge? Again, since we are working in the random oracle model, we will give the extractor control over the outputs of the hash function. That is, suppose that in the "ideal world" the prover queries the random oracle on T and the extractor responds with some value c. Now, if the extractor rewinds the prover back to before the prover queried on T, then we let the extractor respond with a (possibly) new value c'.

Observe that from two accepting transcripts of the form (T, c, s) and (T, c', s') where $c \neq c'$, the extractor can compute $\frac{s-s'}{/} \frac{c-c'}{=} x$. This is just some intuition for now and we'll discuss how this can be considered more formally later.

4 Random Oracle Model

The random oracle model is both hated and loved. On the negative side, this model is *known* to have problems. That is, there are several results showing that a scheme proven secure in the random oracle model might not be realizable in the standard model by *any* real hash function. Thus, analyzing the security of a protocol in this model is more of a security heuristic than a security proof.

On the positive side, it is often the case that cryptographers first design a scheme in the random oracle model (which is easier to think about) and then later build on this intuition to see how to design a scheme in the standard model. Another positive point is that, for some reason which no one can explain, no one has yet broken a major system because of this issue. Indeed, a proof in the random oracle model guarantees that to break the system one must be using the hash function in a non-trivial way – and hash functions are typically so complicated that they aren't so easy to thus manipulate.

5 Anonymous Credentials

Tomorrow we will discuss anonymous credentials. Assume that some authority has a public key in the sky (U.S. Govt., who issues passports). Say that Lori wants a passport. What she's going to get is a signature under U.S. government signing key on several things: SigSK (Lori, birth-date, address, passport number).

This is where crypto comes in : What if I take my passport to a local bar and I want to prove that Im over 21, but I would prefer that the person does not know how old I am. Wouldnt this be cool?

IBM, Microsoft, and other companies are currently investing in this technology. We'll learn more about it tomorrow.

References

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *CRYPTO '06*, volume 4117 of LNCS, pages 78–96, 2006.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86*, volume 263 of LNCS, pages 186–194, 1986.

[GSV98] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *STOC '98*, pages 399–408, 1998.

[Sch91] Claus P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.