Σ-Protocols Continued & Introduction to Zero Knowledge

Lecturer: Victor Shoup[1] and Scribe: Joël Alwen[1]

New York University
Courant Institute
251 Mercer St.
New York, NY 10012 shoup, jalwen@cs.nyu.edu

Abstract. A Σ-protocol for proving equality of the discrete logarithm of two pairs of numbers is presented and used in an electronic voting scheme originally proposed in [CGS97]. Further, the notion of zero knowledge is introduced and a modified (1-bit challenge) version of Schnorr's identification scheme is shown to be zero knowledge.

1 Proving Discrete Logarithm Equality

1.1 The Protocol

We present a protocol for proving equality of the discrete logarithm of two pairs of numbers as follows. Let G be a cyclic group of prime order p, g_1 and g_2 be generators of G and let u_1 and u_2 be elements of G such that $\exists\ w \in \mathbb{Z}_p$ with $\log_{g_1} u_1 = w = \log g_2 u_2$.

That is we define the language L to be such that

$$(g_1, g_2, u_1, u_2) = x \in L \ \Leftrightarrow\ \exists w \in \mathbb{Z}_p \text{ with } \log_{g_1} u_1 = w = \log_{g_2} u_2$$

L is clearly a language in NP since given the witness w the membership of a 4-tuple x in L can efficiently be verified by raising g_1 and g_2 to the w-th power mod p. In fact the following Σ-protocol is not only a proof of the existence of such a witness but actually also a proof of knowledge.

As usual, the protocol involves two players; a prover P and a verifier V, each with a private random tape. The group G and it's order p are considered public knowledge while the 4-tuple $u := (g_1, g_2, u_1, u_2)$ is given as common input while the prover has private input a witness w to $x \in L$. The (3-round) protocol is run as follows:

1) P chooses $r \leftarrow \mathbb{Z}_p$ at random and sends the two values $a_1 = g_1^r$ and $a_2 = g_2^r$ to V.
2) V selects $c \leftarrow \mathbb{Z}_p$, a random challenge, and sends it to P.
3) P computes $z = r + wc$ and sends it to V. V accepts the proof as correct iff

$$g_1^z = a_1 u_1^c \ \text{ and } \ g_2^z = a_2 u_2^c$$

1.2 Properties

Correctness: To show that this protocol is correct, (namely that an honest prover can indeed convince a verifier to accept a true statement simply by following the prescribed protocol) it suffices to verify that the equations of step 3 hold when z is computed correctly:

$$\text{For } i \in \{1, 2\} : g_i^z = g_i^{r+wc} = g_i^r * g_i^{wc} = a_i (g_i^w)^c = a_i u_i^c$$

Soundness: To show soundness (namely that even an arbitrarily malicious prover P^* can not convince V to accept a false statement with more then a negligible probability) we first demonstrate how to extract a witness w for $x \in L$ from two different conversations (runs of the protocol). Using this fact it can be concluded that for any first round message P^* may send to V, there is at most one possible challenge c which P^* could answer correctly since if there were any more P^* would effectively have to know a witness to $x \in L$ which does not even exist as, by assumption, the statement is false. Therefor we show the probability of any P^* being able to cheat is neglige (in p) in an information theoretic sense.

Assume we are given two different transcripts of the protocol (for the same statement $x \in L$) for which V accepted such that the first round (a_1, a_2) is the same but the second (and possibly third) rounds (c, z) and (c', z') are different. Then because V accepted both transcripts, it must be that

$$\begin{aligned} \text{For } i \in \{1,2\}: \qquad & g_i^z = a_i u_i^c \text{ and } g_i^{z'} = a_i u_i^{c'} \\ \Rightarrow \quad & g_i^{z-z'} = u_i^{c-c'} \\ \Rightarrow \quad & \log_{g_i} u_i = \frac{z-z'}{c-c'} \end{aligned}$$

In other, in such a case, words $w = \frac{z-z'}{c-c'}$, which can be efficiently computed from the transcript, is a witness to $x \in L$. By assumption P^* is trying to convince V of a false statement though, so no witness exists and so there is at most one challenge c which P^* could answer for any given round 1 message. However P^* must decided upon it's round 1 message *before* seeing the challenge and therefor will receive the "correct" challenge c with probability at most $\frac{1}{p}$ which is negligible.

On the side, notice that in the process of showing this protocol to be sound we have also demonstrated it to be a proof of knowledge of the witness since an honest prover, which can be rewound to answer two different challenges can be used to efficiently extract a witness. This property is also known as *special* soundness.

Honest Verifier Zero Knowledge: Honest verifier zero knowledge (HVZK) is the property that an honest, but curious verifier can not gain even a single bit of extra information other then wether or not $x \in L$. This is usually shown by demonstrating the existence of an algorithm called the simulator S which, when given access to V as a black box, can efficiently produce a transcript for a true statement $x \in L$, which is (at least computationally) indistinguishable from the transcript of V interacting with an honest prover P. The following are a two comments concerning this definition:

- Proving the existence of the algorithm S is an attempt to capture the notion that V is not learning anything from P other then wether to accept or not because V could effectively produce the exact same output (the transcript) without ever needing to talk to P, simply by running S. (Of course when actually talking to P, V knows that the challenge was chosen and sent only after the first round and so with overwhelming probability there are at least two challenges (probably many more) to which P could answer, and so, by special soundness, a witness to $x \in L$ must exist.)
- The simulator above is defined as being given black box access to V with full control over all V's input tapes (including it's random tape). Since V is honest (and deterministic) S can actually predict exactly what V will so the definition may seem slightly redundant. However later on when defining a more general form of zero knowledge the black box access to V will make more sense as it will be replace by black box access to an *arbitrarily malicious* verifier V^* for whom S can by no means predict it's actions. Queries to these black boxes will be relevant and necessary for S to generate its simulation.

The simulator can be thought of as an algorithm which takes $u = (g_1, g_2, u_1, u_2)$ and a challenge c as input (together with a random tape) and produces a valid transcript as follows:

1) Select $z \leftarrow \mathbb{Z}_q$ at random
2) For $i \in \{1,2\}$ calculate $a_i = \frac{g_i^z}{u_i^c}$
3) Output the transcript (a_1, a_2, c, z)

It can easily be verified that this transcript will always pass the two verification equations, regardless of wether the statement $x \in L$ is true or not. In fact if the statement is true then the distribution of the output of S (taken as a random variable over the coin flips of S) is identical to the distribution of a transcript produced by P and V.

Note that if $x \notin L$ the distribution of a_1 and a_2 will not be correct since with overwhelming probability $\log_{g_1} a_1 \neq \log_{g_2} a_2$. However by the DDH assumption (g, g^a, g^b, g^{ab}) is indistinguishable from the tuple (g, g^a, g^b, g^c) where $a, b, c \leftarrow \mathbb{Z}_q$ at random. If we let $g_1 = g$, $g_2 = g^a$ this tells us that no computationally bounded verifier V (in particular the honest one) can tell an honest first round: (g_1, g_2, g_1^r, g_2^r) from $(g_1, g_2, g_1^r, g_2^{r'})$ for unknown random and independent r and r', which is the first round of S when $x \notin L$.

1.3 E-voting

Now we look at an application of the above protocol, namely the electronic voting (or "E-voting") scheme of Cramer, Genaro and Schoenmakers [CGS97]. In particular, an OR-proof combining two runs of the above protocol will be used to show that a vote is indeed a valid ballot.

The setup for the E-voting scheme consists of preparing the parameters of an ElGamal public/private key pair. In particular the voting authority will:

1) Select a cyclic group G of order q. (Often this is done by choosing a random (large) prime q such that $p = 2q + 1$ is prime. Then $G = \mathbb{Z}_q$.
2) Let $g \leftarrow G$ be a random generator of G.
3) Let $w \leftarrow \mathbb{Z}_q$ a random element and $g^w = h \in G$
4) Publish the public key (g, h). Remember private key w.

Let there be two candidates, 0 and 1. Votes for a candidate are given as ElGamal encryptions of $g^0 = 1$ for candidate 0 and $g^1 = g$ for candidate 1 under the voting authorities public key. That is, in order for player i to vote for candidate $t_i \in \{0, 1\}$ they compute the following:

1) Select $r_i \leftarrow \mathbb{Z}_q$ at random.
2) Compute ballot $c_i = (g^{r_i}, g^{t_i} h^{r_i})$

By the homomorphic properties of ElGamal, the voting authority can now form the product of all received votes

$$\prod_{i \in \mathbb{I}} c_i = \left(g^{\sum r_i}, g^{\sum t_i} g^{\sum r_i} \right) = (\alpha, \beta)$$

Using the secret key, the voting authority can now compute $g^t = \frac{\beta}{\alpha^w}$. t, the number of votes for candidate 1 can be found by repeatedly raising g to successive powers. For this step [CGS97] makes the assumption that the number of voting players is at most polynomial in the security parameter, therefor finding t can be done efficiently.

Proving a Vote to be Valid: The main problem with this scheme so far is that a player i does is not constrained to vote for $t_i \in \{0, 1\}$. A vote for g^6 or g^{-10} for example is also possible. So before submitting their vote (which we will refer as $c = (u, v)$ for simplicity) each player is required to run an OR-protocol (with the authority) proving that they have either voted for candidate 0 or 1. Specifically the above Σ-protocol for the equality of discrete log is used to show that one of the two following statements is true:

1. $\log_g u = \log_h v$ which implies a vote for candidate 0
2. $\log_g u = \log_h \frac{v}{g}$ which implies a vote for candidate 1

The OR-protocol follows the usual paradigm: the prover (in this case the player submitting the vote) uses the honest verifier zero knowledge algorithm to compute an accepting transcript (a, c, z) for the false statement and uses the honest provers algorithm (and the witness r_i) to prove the true statement.

By the correctness property of the OR-protocol the voter is guaranteed to be submitting a valid ballot. To show the protocol suffices to maintain privacy we must show it to be witness indistinguishable. In general this seems to be an open problem! However for the case of an honest verifier things are more straightforward. That is if we assume that the voting authority is merely honest but curious (rather then arbitrarily malicious) we can assume that it's challenge c in round 2 is chosen independently of the first round message. So, upon learning c, the voter can rewind back to the beginning of the OR-protocol, select a random pair of challenges $c_1 \oplus c_2 = c$ and run the HVZK algorithm on both challenges (effectively faking both proofs). In such a run, by the semantic security of ElGamal it is computationally infeasible to tell wether the first or the second statement of the OR-protocol was true. (If this were not the case then an efficient distinguisher for ElGamal could be constructed out of HVZK and V.) This observation allows for the honest verifier hybrid argument:

- Let $\mathcal{T}$ stand for a true statement, and $\mathcal{F}$ for a false one
- Let subscript *sim* mean a statement is proven with the HVZK algorithm and *do* mean that the statement is proven with the witness and honest provers algorithm

- Let $(\mathcal{T} \vee \mathcal{F})$ be the random variable denoting the verifiers view when running the OR-protocol on input the first statement a true one and the second a false one
- Let $\approx_s$ denote statistical indistinguishability
- Let $\approx_c$ denote computational indistinguishability

$$(\mathcal{T}_{do} \vee \mathcal{F}_{sim}) \approx_s (\mathcal{T}_{sim} \vee \mathcal{F}_{sim}) \approx_c (\mathcal{F}_{sim} \vee \mathcal{T}_{sim}) \approx_s (\mathcal{F}_{sim} \vee \mathcal{T}_{do})$$

The first and third equivalences hold by the correctness of the HVZK algorithm for true statements and the middle equivalence holds by semantic security of ElGamal.

2 Zero Knowledge

Zero knowledge is a relatively broad field in cryptography with many special instances of zero knowledge defined. The notion we will use here is essentially identical to the honest verifier zero knowledge discussed above except for the modification that the simulator S now needs to work against an arbitrary verifier V^*. As before, it is the simulator's job to produce a view of V^* which is indistinguishable from V^*'s view when interacting with the honest prover. By V^*'s view we mean the (true) statement being proved, V^*'s common and private input (including it's random tape) and a complete transcript of the interaction with the prover.

2.1 Black-box ZK

There are a few virtually equivalent ways of defining black-box access to V^*. For the purpose of these lecture notes it suffices to think of V^* as a deterministic oracle which takes a verifiers view as input and returns V^* next action when faced with the given view of an interaction with a prover. Along with the random tape, common input, etc. the view contains a prefix of a conversation which could possibly be empty or contain some messages which have already been played between the prover and verifier.

2.2 Converting Schnorr to ZK

In this section we show how to turn Schnorr's identification protocol into a ZK proof that the public key h is a power of the common input g. That elements of the underlying NP language are (g, h) and the witness is an x such that $g^x = h$. Recall that we demonstrated Schnorr's scheme to be only honest verifier zero knowledge. The problem (as with Σ-protocols in general) was that a sneaky verifier could make their challenge c somehow depend on the first round of the protocol. Therefor after learning c, rewinding and playing a new first round to fit c, the verifier might now send c'.

To get around this problem we will modify the protocol to have as small a challenge space as possible, namely $\{0, 1\}$. This way the simulator has a $\frac{1}{2}$ probability of guessing c in advance and prepare accordingly. Whenever it guesses wrong it throws away it's work and tries again. When it guesses correctly it has produced a valid run! Eventually it will have gathered enough valid runs to output a convincing transcript. Of course this begs the question why can't any malicious prover do the same. Well by running the protocol say t times, such a prover is forced to guess c correctly t times in a row, which is highly unlikely. However S can simply throw away the output to oracle calls where it guessed wrong and continue with the view from where it guessed correctly last.

2.3 The Modified Protocol

The setup is identical to that of the original scheme. G a cyclic group of prime order q and g, a random element are common input. The prover has a secret key $x \leftarrow \mathbb{Z}_q$ chosen at random and a public key $h = g^x$ which is also public input. That statement being proved is that $\exists\ x \in \mathbb{Z}_q\ :\ h = g^x$. The protocol runs as follows:

1) P chooses $r \leftarrow \mathbb{Z}_q$ at random and sends $a = g^r$ to V
2) V chooses $c \leftarrow \{0, 1\}$ at random and sends it to P
3) P computers $z = r + xc$ and sends it to V. V accepts iff $g^z = ah^c$

These three steps are repeated t times. V the statement to be true iff it accepts at each repetition.

Soundness To show this protocol is sound we proceed by contradiction. Assume there exists some P^* which can make V accept a false statement. With overwhelming probability any algorithm which attempts to guess all challenges in advance will fail. Therefor there must be some repetition where P^* doesn't fail even when guessing wrong. In other words, for at least this repetition P^* can answer both possible challenges correctly. However as in the proof of soundness of Schnorr's identification protocol, this implies P^* can be used to (efficiently) extract a witness! This implies the statement is true which is a contradiction. So no such P^* exists.

2.4 The Simulator

To show this protocol is also zero knowledge we give a specific implementation of S and show that it's output has the correct distribution. S takes input a statement and makes use of the HVZK algorithm of Schnorr's identification scheme. Calls to V^* are denoted by the function $V^*(.)$ which takes a view as an argument. Denote by $View$ a variable which holds a view of V^*. It is initialized with a fixed random tape, statement, public and private input for V^*. Initially the conversation prefix is empty. The simulator's algorithm is the following:

```
i := 0
initialize(View)
loop
    loop
        TempView := View
        c̃_i ← {0,1}                                  // Guess challenge
        (a_i, c̃_i, z̃_i) := HVZK(c̃_i)                 // Simulate a repetition
        TempView.prefix := TempView.prefix + a_i      // Append first round message
        c_i = V*(TempView)                            // Get real challenge from oracle
    until c_i = c̃_i
    View.prefix := View.prefix + (a_i, c̃_i, z̃_i)     // Correct guess so store transcript
    i:=i+1
until i = t
output(View)
```

The proof that the simulator's output has the correct distribution relies heavily on the proof of the HVZK simulator. Since the HVZK algorithm outputs triples (a, c, z) with the correct distribution so does this simulator since all it's output consists of such triples for an identical language. (Recall we only modified the challenge space, nothing else.) It only remains to be shown that simulator is efficient. But this is easy, since at each run of the internal loop the probability of guessing the challenge correctly is $\frac{1}{2}$, and so the expected number of iterations of the external loop is $2t$. Therefor S is an expected poly-time algorithm.

2.5 Problems and Variations on ZK

The above ZK protocol has several problems. For starters it is not efficient as it requires $3t = O(t)$ rounds of communication. Nor is it clear wether the protocol is still zero knowledge when run concurrently with other copies of it's self (though soundness still holds.)

Other variations of zero knowledge are statistical [SV00] (or perfect) ZK in which S produces output which is *statistically close* (or *identically* distributed) to the honest prover. Concurrent zero knowledge [DNS98] and efficient (constant round) zero knowledge [GK96] have also received some attention in recent years as have non-black box zero knowledge [Bar01] in which the simulator uses some knowledge of the internal workings of the oracle such as it's Kolmogorov complexity. For further a rather extensive overview of the literature see [Lip].

References

[Bar01] Boaz Barak. How to Go Beyond the Black-Box Simulation Barrier. In *IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *Lecture Notes in Computer Science*, 1233:103+, 1997.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent Zero-knowledge. pages 409–418, 1998.

[GK96] Oded Goldreich and Ariel Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 9(3):167–189, Summer 1996.

[Lip] Helger Lipmaa. Cryptology Pointers, http://www.adastral.ucl.ac.uk/ helger/crypto/link/zeroknowledge/.

[SV00] Amit Sahai and Salil P. Vadhan. A Complete Problem for Statistical Zero Knowledge. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(84), 2000.