Notes on Zero Knowledge

These notes on zero knowledge protocols for quadratic residuosity are based on notes from CS276, Cryptography, scribed by Milosh Drezgich, Alexandra Constantin, and Anindya De.

1 Intuition

A *zero knowledge proof* is an interactive protocol between two parties, a *prover* and a *verifier*. Both parties have in input a *statement* that may or may not be true, for example, the description of a graph G and the statement that G is 3-colorable, or integers N, r and the statement that there is an integer x such that $x^2 \bmod N = r$. The goal of the prover is to *convince* the verifier that the statement is true, and, at the same time, make sure that *no information other than the truth of the statement* is leaked through the protocol.

A related concept, from the computational viewpoint, is that of a *zero knowledge proof of knowledge*, in which the two parties share an input to an NP-type problem, and the prover wants to convince the verifier that he, the prover, *knows a valid solution for the problem on that input*, while again making sure that no information leaks. For example, the common input may be a graph G, and the prover may want to prove that he knows a valid 3-coloring of G, or the common input may be N, r and the prover may want to prove that he knows an x such that $x^2 \bmod N = r$.

If a prover "proves knowledge" of a 3-coloring of a graph G, then he also proves the statement that G is 3-coloring; in general, a proof of knowledge is also a proof of the statement that the given instance admits a witness. In some cases, however, proving that an NP statement is true, and hence proving *existence* of a witness, does not imply a proof of *knowledge* of the witness. Consider, for example, the case in which common input is an integer N, and the prover wants to prove that he knows a non-trivial factor N. (Here the corresponding "statement" would be that N is composite, but this can easily be checked by the verifier offline, without the need for an interaction.)

Identification schemes are a natural application of zero knowledge. Suppose that a user wants to log in into a server. In a typical Unix setup, the user has a password x, and the server keeps a hash $f(x)$ of the user's password. In order to log in, the user sends the password x to the server, which is insecure because an eavesdropper can learn x and later impersonate the user.

In a secure identification scheme, instead, the user generates a public-key/ secret key pair (pk, sk), the server knows only the public key pk, and the user "convinces" the server of his identity without revealing the secret key. (In SSH, for example, (pk, sk) are the public key/ secret key pair of a signature scheme, and the user signs a message containing a random session identifier in order to "convince" the server.)

If f is a one-way function, then a secure identification scheme could work as follows: the user picks a random secret key x and lets its public key be $f(x)$. To prove its identity, the user engages in a *zero knowledge proof of knowledge* with the server, in which the user plays the prover, the server plays the verifier, and the protocol establishes that the user knows an inverse of $f(x)$. Hence, the server would be convinced that only the actual person would be able to log in, and moreover from the point of view of the user he/she will not be giving away any information the server might maliciously utilize after the authentication.

This example is important to keep in mind as every feature in the definition of the protocol has something desirable in the protocol of this model.

The main application of zero knowledge proofs is in the theory of multi party protocols in which multiple parties want to compute a function that satisfies certain security and privacy property. One such example would be a protocol that allow several players to play online poker with no trusted server. By such a protocol, players exchange messages to get the local view of the game and also at the end of the game to be able to know what is the final view of the game. We would like that this protocol stays secure even in the presence of malicious players. One approach to construct such a secure protocol is to first come up with a protocol that is secure against "honest but curious" players. According to this relaxed notion of security, nobody gains extra information provided that everybody follows the protocol. Then one provides a generic transformation from security against "honest but curious" to security against malicious user. This is achieved by each user providing a ZKP at each round that in the previous round he/she followed the protocol. This would on one side convince the other players that no one is cheating and on the other side the player presenting the protocol would provide no information about his own cards. This forces apparent malicious players to act honestly, as only they can do is to analyze their own data. At at the same time this is also not a problem for the honest players.

2 The Quadratic Residuosity Problem

We review some basic facts about quadratic residuosity modulo a composite.

If $N = p \cdot q$ is the product of two distinct odd primes, and $\mathbb{Z}_N^*$ is the set of all numbers in $\{1, \ldots, N-1\}$ having no common factor with N, then we have the following easy consequences of the Chinese remainder theorem:

- $\mathbb{Z}_N^*$ has $(p-1) \cdot (q-1)$ elements, and is a group with respect to multiplication;

  PROOF:

  Consider the mapping $x \to (x \bmod p, x \bmod q)$; it is a bijection because of the Chinese remainder theorem. (We will abuse notation and write $x = (x \bmod p, x \bmod q)$.) The elements of $\mathbb{Z}_N^*$ are precisely those which are mapped into pairs (a, b) such that $a \neq 0$ and $b \neq 0$, so there are precisely $(p-1) \cdot (q-1)$ elements in $\mathbb{Z}_N^*$.

  If $x = (x_p, x_q)$, $y = (y_p, y_q)$, and $z = (x_p \times y_p \bmod p, x_q \times y_q \bmod q)$, then $z = x \times y \bmod N$; note that if $x, y \in \mathbb{Z}_N^*$ then x_p, y_p, x_q, y_q are all non-zero, and so $z \bmod p$ and $z \bmod q$ are both non-zero and $z \in \mathbb{Z}_N^*$.

  If we consider any $x \in \mathbb{Z}_N^*$ and we denote $x' = (x_p^{-1} \bmod p, x_q^{-1} \bmod q)$, then $x \cdot x' \bmod N = (x_p x_p^{-1}, x_q x_q^{-1}) = (1, 1) = 1$.

  Therefore, $\mathbb{Z}_N^*$ is a group with respect to multiplication. $\square$

- If $r = x^2 \bmod N$ is a quadratic residue, and is an element of $\mathbb{Z}_N^*$, then it has exactly 4 square roots in $\mathbb{Z}_N^*$

  PROOF:

  If $r = x^2 \bmod N$ is a quadratic residue, and is an element of $\mathbb{Z}_N^*$, then:

  $r \equiv x^2 \bmod p$

  $r \equiv x^2 \bmod q$.

  Define $x_p = x \bmod p$ and $x_q = x \bmod q$ and consider the following four numbers:

  $x = x_1 = (x_p, x_q)$

$x_2 = (-x_p, x_q)$

$x_3 = (x_p, -x_q)$

$x_4 = (-x_p, -x_q)$.

$x^2 \equiv x_1^2 \equiv x_2^2 \equiv x_3^2 \equiv x_4^2 \equiv r \bmod N$.

Therefore, x_1, x_2, x_3, x_4 are distinct square roots of r, so r has 4 square roots.

$\square$

- Precisely $(p-1) \cdot (q-1)/4$ elements of $\mathbb{Z}_N^*$ are quadratic residues

  PROOF:

  According to the previous results, $\mathbb{Z}_N^*$ has $(p-1) \cdot (q-1)$ elements, and each quadratic residue in $\mathbb{Z}_N^*$ has exactly 4 square roots. Therefore, $(p-1) \cdot (q-1)/4$ elements of $\mathbb{Z}_N^*$ are quadratic residues. $\square$

- Knowing the factorization of N, there is an efficient algorithm to check if a given $y \in \mathbb{Z}_N^*$ is a quadratic residue and, if so, to find a square root.

It is, however, believed to be hard to find square roots and to check residuosity modulo N if the factorization of N is not known.

Indeed, we can show that from any algorithm that is able to find square roots efficiently mod N we can derive an algorithm that factors N efficiently.

Theorem 1 *If there exists an algorithm A of running time t that finds quadratic residues modulo $N = p \cdot q$ with probability $\geq \epsilon$, then there exists an algorithm A^* of running time $t + O(\log N)^{O(1)}$ that factors N with probability $\geq \frac{\epsilon}{2}$.*

PROOF: Suppose that, for a quadratic residue $r \in \mathbb{Z}_N^*$, we can find two square roots x, y such that $x \neq \pm y \pmod{N}$. Then $x^2 \equiv y^2 \equiv r \bmod N$, then $x^2 - y^2 \equiv 0 \bmod N$. Therefore, $(x-y)(x+y) \equiv 0 \bmod N$. So either $(x-y)$ or $(x+y)$ contains p as a factor, the other contains q as a factor.

The algorithm A^* is described as follows:

Given $N = p \times q$

```
• pick $x \in \{0 \ldots N-1\}$
• if $x$ has common factors with $N$, return gcd$(N, x)$
• if $x \in \mathbb{Z}_N^*$
    – $r := x^2 \bmod N$
    – $y := A(N, r)$
    – if $y \neq \pm x \bmod N$ return $gcd(N, x+y)$
```

WIth probability ϵ over the choice of r, the algorithm finds a square root of r. Now the behavior of the algorithm is independent of how we selected r, that is which of the four square roots of r we selected as our x. Hence, there is probability 1/2 that, conditioned on the algorithm finding a square root of r, the square root y satisfies $x \neq \pm y \pmod{N}$, where x is the element we selected to generate r. $\square$

3 The Quadratic Residuosity Protocol

We consider the following protocol for proving quadratic residuosity.

- Verifier's input: an integer N (product of two unknown odd primes) and a integer $r \in \mathbb{Z}_N^*$;
- Prover's input: N, r and a square root $x \in Z_N^*$ such that $x^2 \bmod N = r$.
- The prover picks a random $y \in Z_N^*$ and sends $a := y^2 \bmod N$ to the verifier
- The verifier picks at random $b \in \{0, 1\}$ and sends b to the prover
- The prover sends back $c := y$ if $b = 0$ or $c := y \cdot x \bmod N$ if $b = 1$
- The verifier cheks that $c^2 \bmod N = a$ if $b = 0$ or that $c^2 \equiv a \cdot r \pmod N$ if $b = 1$, and accepts if so.

We show that:

- If r is a quadratic residue, the prover is given a square root x, and the parties follow the protocol, then the verifier accepts with probability 1;
- If r is not a quadratic residue, then for every cheating prover strategy P^*, the verifier rejects with probability $\geq 1/2$.

Proof:
Suppose r is not a quadratic residue. Then it is not possible that both a and $a \times r$ are quadratic residues. If $a = y^2 \bmod N$ and $a \times r = w^2 \bmod N$, then $r = w^2 (y^{-1})^2 \bmod N$, meaning that r is also a perfect square.

With probability 1/2, the verifier rejects no matter what the Prover's strategy is.

$\square$

Definition 2 *A protocol defined by two algorithms P and V is an interactive proof with efficient prover, for a decision problem if:*

- ***(Completeness)*** *for every input x for which the correct answer is YES, there is a witness w such that $P(x, w) \leftrightarrows V(x)$ interaction ends with V accepting with probability one.*
- ***(Soundness)*** *for every input x for which answer is NO, for algorithm P^* of arbitrary complexity $P^*(x, w) \leftrightarrows V(x)$ interaction ends with V rejecting with probability at least half (or at least $1 - \frac{1}{2^k}$ if protocol repeated k times)*

So the quadratic residuosity protocol described above is an *interactive proof with efficient prover* for the quadratic residuosity problem.

We now formalize what we mean by the verifier *gaining zero knowledge* by participating in the protocol. The interaction is ZK if the verifier could simulate the whole interaction by himself without talking to the prover.

Definition 3 (Honest Verifier Perfect Zero Knowledge) *A protocol (P, V) is Honest Verifier Perfect Zero Knowledge with simulation complexity s for a decision problem if there is an algorithm $S(\cdot)$ that has complexity at most s, such that $\forall x$ for which the answer is YES, $S(x)$ samples the distribution of $P(x, w) \leftrightarrows V(x)$ interactions for every valid w.*

Therefore the simulator does not know the witness but it is able to replicate the interaction between the prover and the verifier. One consequence of this is, the protocol is able to simulate all possible interactions regardless of what particular witness the prover is using. Hence the protocol does the same regardless of witness. This *witness indistinguishability* property is useful on its own.

We now show that the above protocol is also zero knowledge. More precisely, we show the following.

Theorem 4 *For every verifier algorithm V^* of complexity $\leq t$ there is a simulator algorithm of average complexity $\leq 2t + (\log N)^{O(1)}$ such that for every odd composite N, every r which is a quadratic residue $\pmod N$ and every square root of x of r, the distributions*

$$S^*(N, r) \tag{1}$$

and

$$P(N, r, x) \leftrightarrow V^*(N, r) \tag{2}$$

are identical.

Proof: The simulator S^* is defined as follows. It first picks $b_1 \in \{0, 1\}$ uniformly at random. It also picks $y \in Z_n^*$ uniformly at random. If $b_1 = 0$, set $a = y^2$ and if $b_1 = 1$, set $a = y^2 r^{-1}$. Note that irrespective of the value of b_1, a is a uniformly random element of Z_n^*. With this S^* simulates the interaction as follows. First, it simulates the prover by sending a. If the second round reply from V^* (call it b) is not the same as b_1, then it aborts the simulation and starts again. If not, then clearly $c = y$ is the reply the prover will send for both $b = 0$ and $b = 1$. Hence whenever the simulation is completed, the distribution of the simulated interaction is same as the actual interaction. Also observe that b_1 is a random bit statistically independent of a while b is totally dependent on a (and probably some other random coin tosses). Hence in expectation, in two trials of the simulation, one will be able to simulate one round of the actual interaction.Hence the expected time required for simulation is the time to simulate V^* twice and the time to do couple of multiplications in Z_n^*. So, in total it is at most $2t + (\log N)^{O(1)}$. $\square$

4 Proofs of Knowledge

Suppose that L is a language in NP; then there is an NP relation $R_L(\cdot, \cdot)$ computable in polynomial time and polynomial $p(\cdot)$ such that $x \in L$ if and only if there exists a witness w such that $|w| \leq p(|x|)$ (where we use $|z|$ to denote the length of a bit-string z) and $R(x, w) = 1$.

Recall the definition of *soundness* of a proof system (P, V) for L: we say that the proof system has soundness error at most ϵ if for every $x \notin L$ and for every cheating prover strategy P^* the probability that $P^*(x) \leftrightarrow V(x)$ accepts is at most ϵ. Equivalently, if there is a prover strategy P^* such that the probability that $P^*(x) \leftrightarrow V(x)$ accepts is bigger than ϵ, then it must be the case that $x \in L$. This captures the fact that if the verifier accepts then it has high confidence that indeed $x \in L$.

In a proof-of-knowledge, the prover is trying to do more than convince the verifier that a witness exists proving $x \in L$; he wants to convince the verifier that he (the prover) *knows* a witness w such that $R(x, w) = 1$. How can we capture the notion that an algorithm "knows" something?

Definition 5 (Proof of Knowledge) *A proof system (P, V) for an NP relation R_L is a* proof of knowledge *with* knowledge error *at most ϵ and* extractor slowdown *es if there is an algorithm K*

(called a knowledge extractor*) such that, for every prover strategy P^* of complexity $\leq t$ and every input x, if*

$$\Pr[P^*(x) \leftrightarrow V(x) \text{ accepts}] \geq \epsilon + \delta$$

then $K(P^*, x)$ outputs a w such that $R(x, w) = 1$ in average time at most

$$es \cdot (n^{O(1)} + t) \cdot \delta^{-1}$$

In the definition, giving P^* as an input to K means to give the code of P^* to K. A stronger definition, which is satisfied by all the proof systems we shall see, is to let K be an oracle algorithm of complexity $\delta^{-1} \cdot es \cdot poly(n)$, and allow K to have oracle access to P^*. In such a case, "oracle access to a prover strategy" means that K is allowed to select the randomness used by P^*, to fix an initial part of the interaction, and then obtain as an answer what the next response from P^* would be given the randomness and the initial interaction.

Theorem 6 *The protocol for quadratic residuosity of the previous section is a proof of knowledge with knowledge error* $1/2$ *and extractor slowdown 2.*

PROOF: Consider an a such that the prover returns the correct answer both when $b = 0$ and $b = 1$. More precisely, when $b = 0$, prover returns a in the third round of the interaction and if $b = 1$, prover returns $a.r$ in the third round of interaction. If we can find such an a, then upon dividing the answers (for the cases when $b = 1$ and $b = 0$) returned by the prover strategy in third round, we can get the value of r. Note that if verifier V accepts with probability $\frac{1}{2} + \delta$, then by a Markov argument, we get that with probability δ, a randomly chosen $a \in Z_n^*$ is such that for both $b = 0$ and $b = 1$, the prover returns the correct answer. Clearly, the knowledge error of the protocol is $\frac{1}{2}$ and for one particular a, the prover strategy is executed twice. So, the extractor slowdown is 2. Note that in expectation, we will be sampling about $\frac{1}{\delta}$ times before we get an a with the aforementioned property. Hence, the total expected time for running K is $2 \cdot ((\log N)^{O(1)} + t) \cdot \delta^{-1}$ $\square$