NYU COMPUTER SCIENCE FALL 2026

Introduction to Cryptography

CSCI-GA.3210-001 Instructor: Nir Bitansky

Lecture 9: Zero-Knowledge Proofs

1 Previously on Introduction to Crypto

So far, we've focused mainly on cryptographic schemes that dealt with cryptography's oldest mission — securing communication. In the remaining weeks, we'll move on beyond secure communication. Here the first notion we will touch is that of zero-knowledge proofs.

2 Must a Proof Convey Knowledge?

Typically, when we think about the concept of *a proof* our most essential requirement is that *they're convincing* — if a statement has a proof then it must be true. Different disciplines of course have different takes on what convincing means (e.g., it could be merely evidence in criminology, or perhaps a fully detailed mathematical proof in calculus). Something that seems inherent to almost any kind of proof is that it doesn't only tell us that a given assertion x is true, but it also gives us some information on *how it is true* (e.g., a weapon with fingerprints suggests how a murder was performed). In particular, it is often the case that by witnessing a proof we gain the knowledge required to prove the assertion to someone else. In some scenarios, we may wish to avoid conveying such knowledge (for instance, we may like to prove that there's an eye witness to a murder, without revealing their identity and thereby risking them).

Zero-knowledge proofs, introduced by Goldwasser, Micali and Rackoff [GMR85], suggest that proofs could be both convincing and yet, somewhat magically, not convey any knowledge, except of course for the fact that the assertion is true.

An Example: Sudoku Puzzles [GNPR07]. A Sudoku board is a 9×9 board divided into 3×3 subsquares where the goal is to fill the board with integers in $\{1, \ldots, 9\}$ so that every integer appears exactly once in every row, column, or subsquare. A corresponding puzzle comes with some initial hints (partial assignment of integers) and has to be completed to a full solution.

Can you convince someone that you know a solution without revealing anything about it?

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Unsolved Sudoku

4	2	*6*	5	*7*	1	*3*	9	*8*
8	*5*	*7*	2	*9*	3	*1*	*4*	6
1	3	*9*	*4*	6	*8*	*2*	7	*5*
9	*7*	1	*3*	*8*	*5*	6	*2*	*4*
5	4	*3*	*7*	*2*	*6*	*8*	1	9
6	*8*	2	*1*	*4*	*9*	7	*5*	*3*
7	9	*4*	*6*	3	*2*	*5*	8	*1*
2	*6*	*5*	8	*1*	4	*9*	*3*	7
3	1	*8*	9	*5*	7	*4*	6	*2*

Solved Sudoku

Consider the following proof strategy using cards as props:

1. The prover, who knows a solution, takes 81 cards with nine sequences of the numbers $\{1, \ldots, 9\}$. They then place them on the board according to the solution, so that their backs (which all look the same) are facing the verifier. They flip over only the cards participating in the hint.
2. The verifier now chooses a random one of the three constraints (rows, columns, or subsquares) and asks the prover to prove that the constraint is satisfied.
3. The prover does this as follows. Say for instance that the verifier asked to prove consistency of the rows. The prover stacks each of the rows, and shuffles its cards at random behind their back (turning around is also fine), and hands the nine corresponding shuffled stacks to the verifier.
4. The verifier checks that each such stack contains all numbers $\{1, \ldots, 9\}$.

Notice that if the solution is valid, the prover will always manage to pass the test. However, if the prover placed an invalid solution on the board, there will be at least one type of constraint that is violated, and will catch them w.p. at least $1/3$. By repeating this proof n times independently, the probability that the prover manages to cheat can be decreased exponentially in n.

What Does a Verifier Learn? If the solution was valid, in each such proof, the verifier simply obtains 9 stacks of $\{1, \ldots, 9\}$ at a random order. This doesn't convey any information — the verifier doesn't need a solution to generate 9 shuffled stacks, they can do it themselves. In particular, the verifier doesn't gain the ability to prove to a third party that they know a solution.

The above proof seems nothing like what we're used to. It's *interactive* and it's *randomized*. Is it just a card trick? or can we actually prove meaningful statements this way? does it have a digital analog where parties simply exchange messages? As we shall discuss later on, the last proof system is already very expressive (when generalized to $n^2 \times n^2$ Sudoku boards) and has a natural digital analog. First, let's figure out how to define these proof systems.

3 Defining Zero-Knowledge Proofs

We'll start by defining the concept of interactive proofs (IPs), without addressing the zero-knowledge requirement. Indeed, while born together with zero-knowledge, this concept turned out to be fascinating on its own, and has completely revolutionized complexity theory (in particular, it can be seen as the seed for major results such as the characterization of PSPACE and the PCP theorem).

Definition 3.1 (Interactive Proofs (IPs))**.** *An interactive proof system (P, V) for a language $L \subseteq \{0,1\}^*$ consists of a PPT verifier V and a prover P, which are both interactive. We require:*

- **Completeness:** *for any $x \in L$,*
$$\Pr\left[\langle P, V\rangle(x) = 1\right] \geq 0.9 \ ,$$
where $\langle P, V\rangle(x)$ denote the output of V after a joint interaction with P on common input x.
- **Soundness:** *for any $x \notin L$ and any malicious prover strategy P^*,*
$$\Pr\left[\langle P^*, V\rangle(x) = 1\right] \leq 0.1 \ .$$

Remarks on the Definition:

- **Soundness and Completeness Errors.** In the above we allow soundness error and completeness error $c = s = 0.1$. In general, we may consider proof systems with arbitrary $s + c < 1$. We can also amplify this by running the protocols sequentially (the number of required repetitions depends on the new desired error and the gap $1 - c - s$).

- **Honest vs Malicious.** So far in the course, a cryptographic scheme or protocol were always executed by the honest parties (Alice and Bob) and was subject to an attack by a third party (Eve). Here the protocol is supposed to behave correctly when both parties P and V are honest, and remain sound even if the prover party is malicious.

- **Efficiency of Parties.** The prover in the IP definition (honest or malicious) may be computationally unbounded. The verifier on the other hand is efficient.[1]

  A common requirement is that the honest prover is also efficient (PPT), which we will call an *efficient prover IP*. The ability to obtain an efficient prover depends on the complexity of the language and may require that the honest prover will get an extra auxiliary input.[2] For instance, to prove an NP statements efficiently we must provide the prover with a witness.

  It is also quite common to limit malicious provers to being efficient, which is sometimes called an *interactive argument*, or a *proof with computational soundness*.

Defining Zero Knowledge. We now move to formally define what it means for a protocol to be zero knowledge. The intuition that we'd like to capture is that

Whatever the verifier learns from a proof of a true statement, they could have learned on their own.

This is formally captured by requiring that there's an efficient simulation algorithm $S(x)$ that can *simulate* the *view* of the verifier.

Definition 3.2 (Zero Knowledge (ZK))**.** *A proof system (P, V) for L is zero knowledge if for any n.u. PPT V^* (with arbitrarily long output), there exists a n.u. PPT S such that*

$$\{\langle P, V^* \rangle (x)\}_{x \in L} \approx_c \{S(x)\}_{x \in L} \ ,$$

where the random variable $\langle P, V^* \rangle (x)$ denotes V^*'s output after an interaction with P on common input x.[3]

Remarks on the Definition:

- **WLOG: Simulate the View.** Unlike the honest verifier that only outputs its decision bit, the malicious V^* can produce an output that arbitrarily (but efficiently) depends on its view in the protocol's execution. We can assume w.l.o.g that the verifier outputs the view itself, including all messages and their randomness.

  Expected Polynomial Time. A common relaxation of the definition is that the simulator will only run in *expected polynomial time*.

- **Universal Simulation.** Another common formulation requires that there is one *universal simulator* PPT S that can simulate any verifier V^*, given the code of V^* as additional input. This definition turns out to be very useful when composing zero-knowledge protocols with other protocols (or with themselves). For now, we can ignore this difference.

The Simulation Paradigm More Broadly. The zero knowledge definition is a special case of a more general *simulation paradigm*, whose main idea is to capture at once *all the possible attacks*. Indeed, the verifier may try to learn different types of information. For instance, for an NP language L, it may try to learn a witness, or to just learn the first bit of the witness. Simulation guarantees that all of these potential attacks would fail, unless these are pieces of information that can anyhow be learned from the statement itself.

[1] Indeed, philosophically speaking, efficient verification is the essence of how a true statement with a proof is superior to merely a true statement.

[2] Note that an efficient prover IP where the prover doesn't get any auxiliary input implies that $L \in BPP$ (think why).

[3] Above we think of $n = |x|$ as the security parameter.

More generally, the simulation paradigm aims to capture the fact that whatever effect the adversary can have on our cryptographic scheme or protocol is restricted to what the adversary can do in *an ideal world*. (In the ZK ideal world, the only information given to the adversary is the fact that the statement is true.) We've actually already encountered this paradigm in some of the encryption definitions we've discussed, where encryptions of any message could be simulated by a sampling from fixed distribution, independent of any message.

4 Constructing Zero Knowledge Protocols

We would now like to figure out how to construct zero knowledge protocols and in particular how to prove the existence of a simulator. We will start with a protocol for the specific problem of *quadratic residuosity*, which is in fact, the first zero knowledge protocol ever constructed [GMR85].

Consider the language

$$QR = \left\{(N, y) \;\middle|\; \exists x \in \mathbb{Z}_N^* : y = x^2 \mod N\right\} .$$

We'll design a zero knowledge proof system (P, V) for the language QR that has an efficient prover that gets as auxiliary input the square root x of the instance $y = x^2$. The system will have perfect completeness (i.e. the completeness error is $c = 0$) and soundness error $s = 1/2$. In the following protocol all arithmetics is mod N.

Protocol $\langle P(x), V\rangle\,(N, y)$**:**

1. P: chooses at random $r \leftarrow \mathbb{Z}_N^*$, and sends $s = r^2$ to V.
2. V: samples a bit $b \leftarrow \{0, 1\}$ at random and sends b to P.
3. If $b = 0$, P sends back r, and if $b = 1$, it sends back $t = rx$.
4. If $b = 0$, V checks that $s = r^2$, and if $b = 1$ it checks that $sy = t^2$.

Claim 4.1. *The above protocol is a zero knowledge proof with soundness error* $1/2$ *and expected polynomial-time simulator.*

Proof Sketch. Completeness of the protocol is straightforward. We'll focus on soundness and ZK.

Soundness. To see that the protocol has soundness error $s = 1/2$, note that if y is not a quadratic residue mod N then for any quadratic residue $s = r^2$, sy is not a quadratic residue; indeed if $sy = t^2$, then $y = s^{-1}t^2 = (r^{-1}t)^2$. This means that if $(y, N) \notin QR$, either s has no square root, or sy has no square root, and we'll catch the prover with probability $1/2$.

Zero-Knowledge Simulation. We now move to prove the that protocol is zero-knowledge in the case that $y = x^2$. First, to develop some intuition, let's start with a warmup. Imagine that the verifier always chooses $b = 0$. Then in this case, all that it sees in an interaction with P is a sample from the distribution $D_0 := \left(r^2, r \;\middle|\; r \leftarrow \mathbb{Z}_N^*\right)$, which is easy to simulate. Now imagine that the verifier always chooses $b = 1$. Can we simulate it without the witness? This verifier always sees a sample from $D_1 := \left(r^2, xr \;\middle|\; r \leftarrow \mathbb{Z}_N^*\right)$. The point is that r^2, xr is distributed exactly like $(x^{-1}r)^2, x(x^{-1}r) = (y^{-1}r^2, r)$, which is again something we can easily simulate, given y alone.

But how do we simulate a verifier that may arbitrarily choose b, possibly depending on the first message. Can the simulator predict which b is going to be used? Well it can guess b', hoping that $b = b'$, and then attempt to simulate D_b. This works because the first message in either D_0 or D_1 has the exact same distribution, independently of the simulator's guess b' and thus $b = b'$ with probability $1/2$. If the simulator fails, it can try again (how many times until it succeeds?)

Let's the describe the simulator S more formally:

1. S samples $b' \leftarrow \{0,1\}$ and $r \leftarrow \mathbb{Z}_N^*$ and starts emulating an interaction with V^*:
   - If $b' = 0$, S sends r^2 to V^*.
   - If $b' = 1$, S sends $y^{-1}r^2$.
2. S receives b from V^* and:
   - If $b = b'$, sends r to V^*, and concludes the simulation.
   - Else, returns to the beginning and performs the entire simulation again.

First, to see that the simulator runs in expected polynomial time, we use the fact that the simulator's first message is always from the same distribution — a random quadratic residue — and is independent of their choice b'. Thus the verifier's choice b is also independent of b', which implies that $b' = b$ w.p. $1/2$. So in expectation the simulator only has to perform the experiment twice.

Next, we would like to prove that the view of the verifier in a real execution with the prover is indistinguishable from the view produce by the simulator; we'll in fact show it's identically distributed. For this, it is sufficient to show that:

1. Conditioning on either verifier choice $b \in \{0,1\}$, the real interaction and simulated interaction are identically distributed.
2. The distribution of b in the real interaction and simulated interaction is identical.

The first claim follows similarly to our warmup. Indeed, in the real interaction, conditioned on b, the view is distributed as D_b, and our simulator conditioned on b samples from a distribution that is identical to D_b.

The second claim follows directly from the fact that both in a real and simulated interaction the distribution of the first message is identical (a random quadratic residue). □

5 More Expressive Zero Knowledge

Quadratic residuosity is a specific problem, and we have strongly relied on its specific algebraic structure to construct the proof systems.

Do the protocols we saw generalize to broader classes? What can be proved in zero knowledge?

Remarkably it turns out that we can prove much more in zero knowledge!

Theorem 5.1 ([GMW86])**.** *Assuming OWFs, any NP language has a zero-knowledge proof.*

We will prove this theorem, but first we need to define a tool which is very commonly used in protocols.

Commitments. To be precise, GMW did not assume OWFs, but a more expressive primitive called *a commitment scheme*, which were later constructed from OWFs [Nao89]. We will now define such commitments and use them to prove the GMW theorem. Intuitively speaking, such commitment schemes are digital analogs of "locked boxes". The sender puts a message in a box, locks it, and sends it to the receiver. The receiver cannot see what's in the box, until the opening phase, when the sender unlocks the box. The sender on the other hand cannot change the message, after they had sent the box.

Today, we'll define and use a simple case of commitments schemes that are non-interactive. The more general definition addresses protocols.

Definition 5.2 (Non-Interactive Commitment Scheme)**.** *A (bit) commitment scheme is given by a PPT commitment algorithm Com satisfying:*

- **Computational Hiding:**

$$\{Com(0;r) \mid r \leftarrow \{0,1\}^n\}_{n\in\mathbb{N}} \approx_c \{Com(1;r) \mid r \leftarrow \{0,1\}^n\}_{n\in\mathbb{N}} \ .$$

- **Perfect Binding:** *For any bits b_0, b_1 and randomness r_0, r_1 if $Com(b_0; r_0) = Com(b_1; r_1)$, then $b_0 = b_1$. (This means that any commitment can be "opened", by exhibiting a randomness, to at most one message.)*

We can naturally define commitments for longer messages, and it is easy to see that given bit commitments we can construct commitments for arbitrary polynomially long messages.

A Simple Construction from Injective OWFs [BM82]. We now describe a simple construction of non-interactive commitments based on injective OWFs. Let f be an injective OWF, we define commitments for one bit messages $b \in \{0,1\}$ as follows

$$Com(b; x, r) = f(x), r, \langle x, r \rangle \oplus b \ ,$$

where $x, r \leftarrow \{0,1\}^n$.

Claim 5.3. *The above construction satisfied hiding and binding.*

Proof Sketch. To prove binding, assume that $Com(m; x, r) = Com(m'; x', r')$. Then by definition $r = r'$and since f is injective, $x = x'$ as well. Thus, $\langle x, r \rangle \oplus m = \langle x, r \rangle \oplus m'$, which means that $m = m'$.

To prove hiding, note that by the GL theorem, $\langle x, r \rangle$ is a hardcore bit of the function $(x, r) \mapsto (f(x), r)$ and so

$$f(U_n), U'_n, \langle U_n, U'_n \rangle \approx_c f(U_n), U'_n, U_1 \ ,$$

where U_n, U'_n are independent. Thus for any message m:

$$Com(m; x, r) = f(U_n), U'_n, \langle U_n, U'_n \rangle \oplus m \approx_c f(U_n), U'_n, U_1 \oplus m \equiv f(U_n), U'_n, U_1$$

Since the r.h.s. does not depend on m, it follows that $Com(m_0) \approx_c Com(m_1)$ for any two messages $m_0, m_1 \in \{0,1\}$. □

Given a commitment for one-bit messages, we can easily construct commitments for messages of arbitrary length, by committing each bit separately (make sure you understand why this construction is secure).

The GMW Protocol. To construct a protocol for any NP language it suffices to construct a protocol for one NP-complete language The GMW protocol is for the language of 3-colorable graphs

$$3COL = \{(U, E) \mid \exists c : U \to [3] : \forall (u, v) \in E, c(u) \neq c(v)\} \ .$$

We'll design a zero knowledge proof system (P, V) for the language $3COL$ that has an efficient prover that gets as auxiliary input a legal coloring c of G. The system will have perfect completeness (i.e. the completeness error is $c = 0$) and soundness error $s = 1 - \frac{1}{|E|}$.

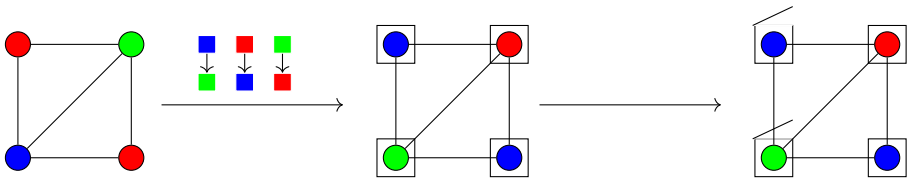


Figure: Illustration of the GMW protocol.
The sealed black boxes represent the commitments. The open boxes are the opened commitments.

Protocol $\langle P(c), V\rangle\,(U, E)$**:**

1. P: chooses a random permutation $\varphi : [3] \to [3]$, and sends V commitments to all colors after applying the permutation:
$$(Com(\varphi(c(v))) \mid v \in U) \quad .$$
2. V: samples $e \leftarrow E$ at random and sends $e = (u, v)$ to P.
3. P opens the commitments corresponding to u and v.
4. V verifies that the two opened colors are distinct.

Claim 5.4. *The above protocol is a zero knowledge proof with soundness error* $1-\frac{1}{|E|}$ *and expected polynomial-time simulator.*

Proof Sketch. Once again completeness of the protocol is straightforward and we'll focus on soundness and ZK.

Soundness. To see that the protocol has soundness error $s = 1 - \frac{1}{|E|}$, note that if (U, E) is not three colorable, then in any coloring, there must exist an edge e that is not properly colored. Since the commitment is perfectly binding, it fixes a coloring of the corresponding vertices (some of the commitments may be invalid altogether, so this coloring may be partial). The probability that the verifier asks to open an improper edge (where improper may mean either that the coloring is improper or that the commitments are invalid) is thus at least $1/|E|$.

Zero-Knowledge Simulation. We now move to prove the that protocol is zero-knowledge. To get some high-level intuition, let us indeed think of the commitments as ideal locked boxes. Then, all that the verifier sees is a bunch of opaque boxes, they then open two boxes of their choice and simply see two distinct random colors, which they could simulate on their own.

The actual simulator will act as follows:

1. S samples $e' \leftarrow E$ and commitments
$$(Com(0) \mid v \in U \setminus e') \quad .^4$$
For the vertices v', u' of e' it samples $c(v')$ and $c(u')$ to be two distinct random colors in $[3]$.
S then emulates V^*, sending the simulated commitments.
2. S receives e from V^* and:
   - If $e = e'$, opens the corresponding commitments and concludes the simulation.
   - Else, returns to the beginning and performs the entire simulation again.

Intuitively, we'd like to apply a similar argument as we did in the quadratic residuosity protocol. That is claim that the simulator's first message distributed similarly to the first prover message and is independent of their choice e'. Then, we'd argue that conditioned on any particular e the views in a real interaction and an ideal interaction are distributed identically. Similarly, we would argue that the simulator runs in expected polynomial time (that scales this time with $1/|E|$).

Of course that all of the above is simply not true, the prover first message and simulator's are distributed very differently, and the simulator's first message not only depends on e', but uniquely determines it. However, we do know that the prover's first message is computationally indistinguishable from that for the simulator, and thus there should be a computational analog to the above argument.

[4]Here 0 is an abitrary choice and we assume 0 and each $i \in [3]$ are represented by the same number of bits.

This is indeed the case. To show this, we consider an imaginary simulator S' that also gets a legal coloring c (this is only a mental experiment). It proceeds similarly to S, except that instead of computing commitments to zeros for all $v \notin e'$, it uses c to commit to their colors, consistently with the permutation it chose (note that choosing two distinct random colors for e' indeed fixes a random permutation $\varphi : [3] \to [3]$). It then proceeds just like S. We can show that

- S and S' have roughly the same expected running times.
- Output computationally indistinguishable views.

Otherwise, we can use them to break the computational hiding of the commitment scheme.

Now, it is left to show that S' run in expected poly time and produces a view that is indistinguishable (in fact, identical) to the an interaction with the real prover. This can already be done following the argument we started with (which didn't work for S). □

References

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 112–117, 1982.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 171–185, 1986.

[GNPR07] Ronen Gradwohl, Moni Naor, Benny Pinkas, and Guy N. Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. In *Fun with Algorithms, 4th International Conference, FUN 2007, Castiglioncello, Italy, June 3-5, 2007, Proceedings*, pages 166–182, 2007.

[Nao89] Moni Naor. Bit commitment using pseudo-randomness. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 128–136, 1989.