0368.4162 Foundations of Cryptography June 5, 2019

Fast Verification and PCPs

Lecturer: Nir Bitansky

1 Previously

We've seen how to define non-interactive zero knowledge (NIZK) and developed tools toward their construction. Today we'll first finish the construction, and then move to speak about how fast we can verify a proof, which is interesting also beyond the context of zero knowledge.

2 Recalling the Definition of NIZK

Before that let's start by recalling their definition.

Definition 2.1 (Non-Interactive ZK (NIZK) for NP)**.** *A NIZK for an NP relation R consists of efficient algorithms (P, V, S_1, S_2) with the following syntax:*

- $\pi \leftarrow P(x, w, crs)$ *gets a statement x, witness w, and string crs and outputs a proof π.*
- $b \leftarrow V(x, \pi, crs$ *gets a statement x, proof π, and string crs and outputs $b \in \{Acc, Rej\}$.*
- $(\widetilde{crs}, td) \leftarrow S_1$ *outputs a simulated $\widetilde{crs}$ and corresponding trapdoor td.*
- $\widetilde{\pi} \leftarrow S_2(x, td)$ *gets a statement x the trapdoor td and outputs a simulated proof $\widetilde{\pi}$.*

The algorithms satisfy the following properties:

- **Completeness:** *for any $(x, w) \in R$ and crs, the verifier $V(x, \pi, crs)$ will accept π generated by $P(x, w, crs)$.*
- **Soundness:** *for any malicious prover P^* and any $x \notin L$*

$$\Pr_{crs \leftarrow \{0,1\}^n} [V(x, \pi, crs) = Acc \mid \pi \leftarrow P^*(x, crs)] \leq negl(n) \ .$$

  Again, if soundness only holds against bounded provers it's called an argument *rather than a proof.*

- **Zero Knowledge:** *for any $(x, w) \in R$*

$$crs, \pi \approx \widetilde{crs}, \widetilde{\pi} \ ,$$

  where $crs \leftarrow \{0,1\}^n, \pi \leftarrow P(x, w, crs)$ and $\widetilde{crs}, td \leftarrow S_1, \widetilde{\pi} \leftarrow S_2(x, td)$.

2.1 Construction

We first constructed a non-interactive witness-indistinguishable proof (or rather, argument) by applying Fiat-Shamir to the Hamiltonicity protocol. In such a proof, the only guarantee is that for a statement $x \in L(R)$, where R is our NP relation, with (at least) two witnesses w_0, w_1. A proof of x using w_0 is computationally indistinguishable from a proof of x using w_1.

Pseudorandom Generators. To construct NIZK from this, we necessarily have to use a common random string. For the construction, we will assume the existence of a pseudo random generator $PRG : \{0,1\}^{n/2} \to \{0,1\}^n$. Such a PRG guarantees that its output $PRG(s)$ for a random (short) seed s is indistinguishable from a truly random (long) string U. (For instance, we can use $PRG(s) = H(s||0 \dots 0)$ for a hash function, in which case we can prove security in the random oracle model.

The Proof System. We will have a CRS $U \leftarrow \{0,1\}^n$. A proof for a statement $x \in L(R)$ would be a WI (non-interactive) proof that

$$\exists w \text{ such that } (x,w) \in R \text{ or } \exists s \text{ such that } U = PRG(s) \ .$$

Claim 2.2. *Except, with probability $2^{-n/2}$ over the choice of a random CRS $U \leftarrow \{0,1\}^n$, there exist no accepting proofs for a false $x \notin L(R)$.*

Proof. The size of the image of PRG is at most $2^{n/2}$. Thus, the probability that a random $U \leftarrow \{0,1\}^n$ hits it is at most $2^{-n/2}$. If it does not, then by the soundness of the WI proof, there exists a proof only if $x \in L(R)$. □

Claim 2.3. *There is a simulator (S_1, S_2) for this system.*

Proof. The simulator samples a CRS $\tilde{U} = PRG(s)$ for a random s and proves the WI statement using s as the witness instead of w, which it does not have. The indistinguishability follows from the pseudorandomness of $\tilde{U}$ and the WI property of the proof system. Specifically, you can show this through a hybrid argument. Note that in the simulated world, the statement has two witnesses, both w and s. Thus, we can consider an experiment H where the proof uses w instead of s — this is indistinguishable by WI. Now notice that this proof is completely independent of the seed s, and thus the experiment H is indistinguishable from the real world where we replace $\tilde{U}$ with a truly random U. Here we use the fact that the prover is efficient (given w) and distinguish U from $\tilde{U}$. □

3 Fast Verification

We'll now move to discuss a different aspect of proofs on the blockchain, which is

How fast can they be verified?

Naturally, when thinking about NP statements, and certainly in the ZK systems we've seen so far, the time to verify a statement grows with both the size of the instance and the witness for the statement. Can we hope for better?

Example: Imagine that a "light node" L, which only holds the last hash value h in the blockchain would like to become convinced that some "full node" F, which holds the entire history, has a given amount U of unspent money. One thing F can do is send the all the blocks on the path P from the last block representing its unspent money to the current hash of the blockchain. This path P, however, could be quite long. Is there a way to prove something like this that does not scale with P?

If we could do this, then verifying the chain would become much faster!

3.1 Probabilistically-Checkable Proofs

Abstractly, we have an NP statement here with a short instance, given by the hash h, the user's identity F, and the unspent amount U, with a potentially long witness W including all the blocks on the path P. Could witnesses like W be generally compressed to a witness W' of length $\ll |W|$. It is not hard to see that we cannot hope to do this for NP languages that are even mildly hard to decide — for instance, languages with polylogarithmic witnesses can be decided in quasiploynomial time.

Nevertheless, theoretical CS has a quite surprising answer to this question. While we cannot construct proofs W' of length $\ll |W|$, we can construct proofs W', such that it is enough to read a constant(!!!) number of randomly chosen bits in order to get convinced that the statement is true with probability 99%. This is what is known as a probabilistically checkable proof (PCP).

Definition 3.1 ((Doubly-Efficient) PCP)**.** *A PCP system for an NP relation R is given by two algorithms (P, V) with the following syntax:*

- *$P(x, w)$ given $(x, w) \in R$, outputs a proof string π.*
- *$V^\pi(x)$, given the instance x, and oracle access to the proof π, accepts or rejects.*

We require

- **Completeness:** *for $(x, w) \in R$ and $\pi = P(x, w)$, $V^\pi(x)$ accepts.*
- **Soundness:** *for $x \notin L(R)$, $V^\pi(x)$ rejects any π w.p. $1 - \text{negl}(n)$. (Here and throughout $n = |x|$.)*
- **(Double) Efficiency:**
  - *$V^\pi(x)$ runs in time $\tilde{O}(n)$ and makes at most $\text{polylog}(n)$ queries.*
  - *$P(x, w)$ runs in time $\text{poly}(n, |w|)$, ideally $\tilde{O}(Time(R(x, w)))$.*

Remark Above we required a negligible soundness error. If we only require a constant soundness error then the number of queries can be made constant.

One of the remarkable achievements of theoretical CS is that such systems can indeed be constructed (unconditionally). They have a very interesting story. In fact, they were born in the context of ZK, in an attempt to construct ZK proof systems that are unconditionally ZK and sound. This has quickly turned out to be possible only for languages in low complexity classes, and thus a new model was invented — the model of two-prover interactive proofs. Here a verifier can interrogate two provers in separate room, without the ability to communicate, and the verifier's questions are randomized so that one cannot predict what the verifier is going to ask the other. Indeed, one can show that two-prover IPs are in fact equivalent to PCPs.

Pretty soon the focus of such PCP has shifted toward *hardness of approximation* results. Perhaps their most famous form (which you may have encountered in complexity theory) us the following:

Theorem 3.2. *There is an efficient reduction that maps any 3SAT formula φ into a new SAT formula $\hat{\phi}$ such that:*

- *If φ is satisfiable then so is $\hat{\varphi}$ and there's an efficient reduction from any satisfying assignment w for φ into one for $\hat{\varphi}$.*
- *If φ is unsatisfiable then any assignment w for φ satisfies at most 88% of the clauses.*

This means that it's NP-hard to even approximate satisfying assignments well enough. In the case of 3SAT formulas and assignments are roughly the same size, however, an analogous theorem holds also for the case that the formula φ can be succinctly represented by a Turing machine M_φ, in which case it will be transformed to a new succinct TM $\hat{M}$.

There are still major open questions concerning PCPs and hardness of approximation like the state of the Unique Games Conjecture (UGC). Today, however, much of the focus in PCPs is in the context of fast verification of computations, in particular, in blockchain applications.

3.2 How to Use PCPs?

So can we use such a PCP over the internet? say to solve the blockchain verification problem discussed above? Not quite. The issue is that PCPs are only sound in a specific model, where the proof π is first fixed in full, and only then the verifier performs its queries. In fact, in any reasonable PCP, if the prover can adaptively choose the answers having already seen the queries, then it can easily cheat (e.g think of SAT, then satisfying just a few clauses is very easy). But how can we enforce the PCP model? (note that the prover cannot just send the proof).

Definition 3.3 (Doubly-Efficient Interactive Proof)**.** *A doubly-efficient IP for an NP relation R is given by two interactive algorithms (P, V) that given a common input $x \in L(R)$ and a prover input $w \in R(x)$ interact, and at the end the verifier decides whether to accept. Completeness, soundness, and efficiency are defined analogously to PCPs.*

We shall in fact allow for a relaxed definition where soundness only needs to hold against efficient provers. This is often called an *interactive argument*.

Bridging the Gap through Crypto. To bridge the above gap, we will use crypto. Specifically, we will aim to "enforce" the PCP model of soundness using tools from cryptography.

Attempt I: Send the PCP proof π. Too long...

Attempt II: Send a short hash $H(\pi)$ of the PCP proof π, and only then send the queries. But how would you check consistency given the answers w/o sending the entire proof π.

Solution: Use Merkle Trees. Instead we'll commit to the proof π using a Merkle Tree. Such commitments guarantee binding to every specific entry and the ability to locally prove consistency with the commitment (root).

Definition 3.4 (Hash w/ local opening)**.** *A collision-resistant hash function with local opening takes an input $x \in \{0,1\}^*$ and compresses it into a hash value $y \in \{0,1\}^n$. For any i, it is possible to generate a proof τ_i of size $\approx n \log |x|$ that x_i is the ith bit of x. No efficient attacker can find a local collision; namely, $y, i, (x_i, \tau_i), (x'_i, \tau'_i)$ such that $x_i \neq x'_i$ and both proofs are accepting.*

Kilian's protocol. We'll assume a non-adaptive PCP for simplicity, namely the verifier chooses all the queries $q_1, \ldots, q_k$ ahead of time before seeing any answers.

- The prover computes a PCP proof π for $(x, w) \in R$. It computes a Merkle hash h of π and sends it to the verifier.
- The verifier samples PCP queries $q_1, \ldots, q_k$ and sends them to the prover.
- The prover sends the answers $a_1, \ldots, a_k$ along with Merkle-Tree proofs of consistency $\tau_1, \ldots, \tau_k$ for each.
- The verifier verifies the PCP answers and the consistency with the root h.

Analysis. Fix $x \notin L$ and A that convinces the verifier with probability $\varepsilon = n^{-O(1)}$.

Claim 3.5. *Let ℓ be the length of a PCP proof for x. Then there exists i such that A such that for both $b \in \{0,1\}$,*

$$\Pr_{Q \leftarrow V_{pcp}}[i \in Q, A \text{ wins}, a_i = b] \geq \varepsilon / 2\ell$$

Note that if we prove this claim we are done as this would violate the binding property of the Merkle tree.

Proof. Otherwise, for any i, $\exists b_i$ such that

$$\Pr_{Q \leftarrow V_{pcp}}[i \in Q, A \text{ wins}, a_i \neq b_i] < \varepsilon/2\ell$$

The values b_i can now be used to define a PCP $\pi = b_1 \dots b_\ell$. So that

$$\Pr_{V_{pcp}}[V^\pi(x) = 1] \geq \varepsilon/2 - \ell \cdot \varepsilon/2\ell \geq \varepsilon/2 \ ,$$

which would break the soundness of the PCP. □

What Kind of Soundsness? Note that the soundness that we get in the above protocol is only guaranteed against computationally bounded attackers, which in particular, cannot find collisions in the underlying hash. Namely, it is what we call an argument. It can be shown that in fact computational soundness is inherent for succinct proofs (under reasonable complexity assumptions).

Removing Interaction. In the random oracle model, one could make this protocol non-interactive using the Fiat-Shamir heuristic. Such succinct non-interactive arguments are also commonly known as SNARKs.

ZK. Can you think of a way to make a SNARK also ZK?

3.3 Recursive Proof Composition and Incremental Computation

Using SNARKs as described above, we can obtain a blockchain where verification of a given transcation always takes fixed time regardless of how deep in the blockchain the relevant information (coins) lies. However, nodes may still need to store large amounts of information.

Can there be a blockchain where the amount of storage for each node is also fixed — a succinct blockchain.

It turns out that SNARKs do allow this. The basic idea is that instead of keeping the entire block path from its unspent money U to the current hash h, a node can just keep the proof π attesting that it has the money. Can the node then forget the entire block path? what will it do once the chain is updated? it would have to also update the proof, for which it may need the entire block path. Can you think of a way to update the proof that does not require storing the entire path?

Recursive Composition: The idea is to recursively compose proofs: given a proof $\pi(U, i, h)$ that node i has unspent coins U in the blockchain fixed by u, and a new block B that extends h and has hash $h' = H(B)$, a prover can compute a new proof $\pi'(U, i, h')$ by proving that:

1. It has *verified* a proof $\pi(U, i, h)$, and
2. h is the previous hash in the blockchain given by h' (formally $\exists$ block B that includes h and $h' = H(B)$).

Once the prover computes such a proof, it can forget B and π and only keep U, h', π'. This way regardless of how deep unspent money is, all parties only need to keep a fixed amount of data. At any point in time they can provide a proof of their unspent money.

More generally, you can think of recursive proof composition as a way to incrementally certify arbitrary computations. That is, if you have a t-step computation with intermediate states $s_0, \dots, s_t$, where $s_i = M(s_{i-1})$ for some machine M that represents the computation. Then if you prove that you verified a proof that $M^i(s_0) = s_i$ and that $M(s_i) = s_{i+1}$ then effectively you verified that $M^{i+1}(s_0) = s_{i+1}$.

Note that it is crucial that the size of the proofs (or more generally verification time) do not grow with the witness. Indeed, a proof of size even twice the witness, would blowup exponentially when proofs are composed.

Soundness vs. Knowledge. The only question is whether the above methodology is really sound. If we imagine that soundness is unconditional, namely, convincing proofs for false statements simply do not exist, then recursive composition is sound by simple induction. However, recall that we only have computational soundness. If we prove that a proof for a previous state s_{i-1} merely *exists* then it is not meaningful — it may exist, and yet the statement can be false. To be able to formally prove soundness here we have to a capture a strong form of soundness — the prover has to show that not only does such a proof exists, but *it can also be efficiently found*. Indeed, then computational soundness would kick in as false proofs are hard to find.

How do you capture something like this? this is what is known as *knowledge extraction*. We require that there is an efficient algorithmic way to extract the knowledge of the prover of such a proof — it cannot prove something without knowing what witness stands behind it.