# Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs

Hugo Hueber, 27.04.2020

## Introduction: what is this paper about anyway?

The applications of **zero-knowledge proofs (ZPK)** are multiple. Whether it is for a private instant messagerie where you should know which contacts are using it without disclosing the contacts to a third party (Signal), or aggregating private statistics without disclosing the anonymised source (Firefox), their presence, even without knowing, is certain.

This paper is first an attempt for a unification of definitions and ideas for ZKP, then the introduction of **fully linear probabilistically checkable proof (PCP) systems**, and finally a presentation of new methods to have more efficient proof systems with sublinear proof size. Usually, the communication costs for ZKP applications would be linear in the size of the input, but the new methods presented would aim for constant or logarithmic communication costs. The paper gives as example PIR writing/private messaging with a constant communication cost, and private statistics/ad targeting - for instance Prio, as previously presented - with a logarithmic communication cost.

## Some (prior) definitions

### *Zero-knowledge proofs (ZK, ZKP)*

Two-party protocol between a prover P and a verifier V. The prover must convince the verifier of some property without disclosing any information. A typical example would be that you have colored balls, and must convince someone that the balls are of different colors (property), without showing the colors (information).

Three properties apply to a ZKP, namely **completeness**: an honest prover P should always convince an honest verifier V, **soundness**: a nasty prover P* should rarely convince an honest verifier V, and **zero-knowledge-ness**: a nasty verifier V* should learn nothing but the property (no information).

### *ZKP on distributed data*

One prover P, multiple verifiers $V_i$. Again, the prover must convince all the verifiers of some property, except that each verifier has only a piece of the data.

The properties apply in a similar fashion, except that we now have **strong zero-knowledge-ness**: a nasty verifier $V_i$* should learn nothing but the property (no information nor other verifiers' data).

### *ZKP on secret-shared data*

A prover P has a vector, and each verifier has additive secret shares (slices) of the vector according to some linear secret sharing scheme (slicing). Verifiers should not see the plain data. The prover should convince the verifiers that this vector is in a given language (**(formal) language**: words with letters taken from an alphabet and well-formed according to a specific set of rules).

### *Linear probabilistically checkable proof (PCP)*

A proof by a randomized algorithm using a bounded amount of randomness (probability of acceptance for the statement) and reading a bounded number of bits of the proof. Basically, a vector π (proof oracle) is used by a verifier to check if a vector x belongs to a given language by making a constant number of linear queries to this proof. After the given number of queries, the verifier should accept or reject with a certain probability.

The same properties as for ZKP apply.

## Fully linear PCP: PCP with a Twist™

A **fully linear probabilistically checkable proof** is essentially the same as a linear PCP, except that the verifier has no longer access to the input x, and shall pass through the proof oracle. The queries are now made to the input and the proof at the same time. Again, the same properties as for ZKP should apply.

We are going to mainly talk about the PCP, but the paper presents a **fully linear oracle proof (OP)** as well, which is basically the interactive analogue of a fully linear PCP. For a given number of rounds, a prover sends you a proof, you send a challenge back, and at the end the verifier makes a number of linear queries to the input and at the oracle, and then accepts or rejects the proof. Every step discussed can be adapted to a fully linear OP.

### *Construction of a fully linear PCP*

Many linear PCP are also already fully linear, which is good news as it shows that the thing works. However, as a downside, we have big proofs (about the size of the circuit).

To construct such a proof "from scratch", we need to do a bit of work. The example used in the paper is one of a circuit C($x_1$, $x_2$, $x_3$, w) with three input gates. The circuit takes a vector x and a witness w, and outputs $v_2$.

We want to check a certain property of this circuit, with a certain input vector, and a certain output (check that x belongs to the L language). The fully linear PCP is the coefficients of the polynomials f, g, h of degree O(|C|). These polynomials are made such that the value fi(j) encodes the value on the i-th input to the j-th gate. We define the polynomials with f(i) being the left input to one gate, g(i) the right input to one gate, and h=f.g. To check the proof, we must verify that the equivalent polynomials are consistent with inputs, and that the final output indeed corresponds to the aforementioned result.

What the paper talks about as well, as stated before, is to develop simpler proofs, mainly by reusing circuits with identical structures. In this way, a proof size proportional to the circuit is achieved.

The paper shows that constructing efficient ZKP on distributed data for a given language is easy. If an efficient fully linear PCP for this language exists, it implies that the efficient ZKP exists.

### *Usability for ZK proofs*

Now that we have defined our fully linear PCP, we can use it to construct an efficient ZK proof on distributed data. First, we have the prover with a given input which belongs to a language L. The prover generates a fully linear PCP and splits it using secret sharing among the verifiers, which have each a share of the input. Each verifier now has a piece of the proof and a piece of the input, and can sample query vectors using common randomness, basically simulating the fully linear PCP on their own, as defined above. The verifiers will then publish shares of the query answers, and we can now reconstruct the whole thing. After a constant number of queries, we can run the whole fully linear PCP verifier and check if we shall accept the proof or not, namely if x is in the language L or not.

The most important result would be that if a language L has a short fully linear PCP, L has a low-communication ZK proof on distributed data. The communication cost of the whole thing is about 2 times the size of the proof, plus a constant number of operations (queries).

## Examples and applications

This paper proposes numerous examples and applications of the techniques presented. One of the most telling examples, because studied in a parallel course, will be that of **secure multi-party computation (MPC)**.

One of the theorem that this paper presents is that for any arithmetic circuit C over a field F, there is a secure three-party protocol for computing C that:

- Tolerates **one malicious party** (a dishonest one).
- Is **computationally secure with abort**.
- Has **amortized communication 1 element of F** per party per gate.

The fully linear version of MPC are up to 7 times less costly than the state-of-the-art, but can one construct such a marvelous thing? Basically, we take a **semi-honest** (corrupted parties cooperate to gather information out of the protocol, but do not deviate from the protocol) MPC protocol, and add two properties to it:

- Protocol **reveals nothing until the very last message**. Usually, proofs are applied on a "per action" basis, but on this case, everything remains under the radar. Instead of several little proofs, we do one big proof right before the last message.
- Each of the parties' messages should be a **degree-two function of the party's input and the messages that the party has received so far**.

The protocol would be such that first we run the semi-honest MPC protocol, and parties halt before publishing the very last message (first property!). At this point, each party will prove to the others that so far the messages complied with the protocol (and this is ZKP!) with a logarithmic complexity in the size of the circuit. Finally, each party will reveal their last message, to recover the output. The total complexity is constant in the size of the circuit.

## Conclusion

Zero-Knowledge Proofs on Secret-Shared Data have one prover, multiple verifiers, different inputs, and while everyone can communicate, they should not exchange inputs. With Fully Linear PCPs, we can reduce proofs (size) and make them quicker (complexity). This tool is as powerful as it gets, as ZKP are used in a lot of fashion (e.g. MPC), but work needs to be done about optimality (are we there yet?) and generalization (could we apply equivalent techniques to all languages/circuits/situations?).

## Sources

- Boneh D., Boyle E., Corrigan-Gibbs H., Gilboa N., Ishai Y. (2019) Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs. In: Boldyreva A., Micciancio D. (eds) Advances in Cryptology – CRYPTO 2019. CRYPTO 2019. Lecture Notes in Computer Science, vol 11694. Springer, Cham
- What does public-coin mean in interactive proof and zero-knowledge?
- Formal language
- Interactive proof system: Public coin protocol versus private coin protocol
- Secure multi-party computation
- Game-icons.net: 3817 free SVG and PNG icons for your games or apps
- Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs
- Fully Linear PCPs and their Cryptographic Applications
- Lectures notes from Stanford

Thanks to Camille Montemagni for the review.