MPC in the Head

CS 507, Topics in Cryptography: Secure Computation

David Heath

Fall 2025

Last time, we discussed connections between maliciously-secure protocols and *zero knowledge (ZK) proofs*. In particular, we examined the GMW compiler, which is a technique for converting semi-honest-secure protocols into malicously-secure ones. Recall that the main idea was to insist that at each protocol step, the sending party proves in zero knowledge that their message was constructed as specified by the protocol. Thus, we can upgrade any semi-honest secure MPC protocol into a maliciously secure protocol.

Today we will explore quite a different connection between MPC and ZK proofs. In particular, we will see how MPC protocols can be used to construct ZK proofs for arbitrary statements. The particular paradigm we will study today is called *MPC in the Head* [IKOS07]. This paradigm has been explored in many works, and today we will study a particularly simple one called ZKBoo [GMO16]. There are a few reasons for studying this paradigm:

- It demonstrates *one way* to construct ZK proofs for arbitrary statements. Note that ZK is an *incredibly* active area of research, and what we will see today is just one piece of an enormous landscape of ideas. Still...
- The paradigm is best-in-class for constructing proofs of particular statements. For instance, the FAEST signature scheme (`https://faest.info`) is based on MPC in the Head [BBD+23]. This signature scheme is a leading candidate for constructing signatures—a central primitive for secure communication—that remain secure in the presence of an adversary with access to a powerful quantum computer.
- The paradigm is a beautiful application of the techniques we have learned so far. It applies them in a non-obvious way, and many of the artifacts that we discuss simply as a means by which to define/prove security are used in a constructive way.

The protocol we will see today is constructible from a commitment scheme.

Zero Knowledge Proofs

Recall from last time that a ZK proof is an interaction between a prover P and a verifier V. The parties agree on some statement x, and P would like to convince V that the statement x is true (formally, x belongs to some language $\mathcal{L}$). To aid in this process, P has a *witness* w, which you can think of as P's proof that x is true. We would like to achieve three properties:

- **Completeness.** If $x \in \mathcal{L}$, then V outputs 1 at the end of the protocol (with high probability).
- **Soundness.** If $x \notin \mathcal{L}$, then not even a malicious P can cause V to output 1 with high probability.
- **Zero Knowledge.** Not even a malicious V can learn anything about P's witness w.

Today, we will concern ourselves with a *general-purpose* notion of ZK proofs. In particular, suppose we have some Boolean circuit $C : \{0,1\}^n \to \{0,1\}$; C will correspond to the proof statement. Namely, the statement we will consider is the following:

There exists some input $w \in \{0,1\}^n$ such that $C(w) = 1$.

Of course, if one knows such a witness $w \in \{0,1\}^n$, it is easy to check that $C(w) = 1$ by simply running the circuit. This language (i.e., those circuits that are *satisfiable*) is *general-purpose* in the sense that any bounded-time computation outputting 0 or 1 can be converted to such a circuit, so we can essentially arrange that P can potentially prove to V that any (bounded) claim is true. Today we will build a generic ZK proof system for any such C.

Note that when we are constructing ZK proof techniques, we are typically concerned with three efficiency properties:

- **Proof size.** How many bits are transmitted between P and V.
- **Prover running time.** How long does it take P to generate the information it sends to V?
- **Verifier running time.** How long does it take V to check P's proof?

Note that in many modern proof systems there can be *huge* discrepancies between these costs. In particular, it is possible to construct proof systems where the verifier time and proof size are *extremely* low. Our focus today will be on simply achieving all three metrics scaling linearly in the circuit size $|C|$, as well as some small dependence on a security parameter.

High Level Sketch of MPC-in-the-Head

Recall, P holds a witness w and both P and V hold a circuit C. P would like to convince V that it knows w such that $C(w) = 1$, while keeping w secret.

The main idea underlying MPC-in-the-head is this: P will emulate—in its head—the execution of a semi-honest secure MPC protocol that securely computes $C(w)$, writing down the view of each emulated party. P then *commits* to the view of each emulated MPC protocol; each party view is committed separately. Next, V challenges P to open a subset of the party views, and P opens them. V checks that the opened emulated parties sent messages consistent with their views, i.e. that they indeed followed the protocol as prescribed. If (1) all opened parties behaved honestly and (2) the output of the MPC protocol was indeed a 1, then V accepts, and otherwise V rejects.

Why does this work? We can argue each ZK property in turn:

- **Completeness:** If $C(x) = 1$ and P simply behaves honestly, then the emulated MPC protocol will indeed output 1, due to correctness of the underlying MPC scheme.
- **Zero Knowledge:** V learns nothing about w due to the security of the MPC protocol. Indeed, V does not open *all* party views.
- **Soundness:** If P wishes to cheat, the insight is that she must "corrupt" at least one of the emulated MPC parties. Indeed, if all emulated parties behave honestly and $C(w) = 0$, then V will not accept, so cheating P must force some emulated party to behave dishonestly. But V opens party views randomly and checks consistency. So if we are careful in our design, V will catch the cheating party with some probability (scaling in the number emulated parties).

In a bit more detail, a cheating P must corrupt at least one *communication channel* between parties. Namely, the protocol must instruct some party p_i to send a message m to p_j, yet p_i instead sends some different message $m' \neq m$. If V opens both p_i and p_j, it will detect this discrepancy.

In more detail...

Let's assume that we have a p-party MPC protocol Π that is secure against any $p-1$ semi-honest corruptions (e.g. the GMW protocol). For such a Π, the MPC-in-the-head paradigm proceeds as follows:

- P secret shares its witness w into p shares $w_0, ..., w_{p-1}$ and gives one share to each of the p emulated parties.
- P emulates execution of an MPC protocol Π that securely computes $C(w_0 + ... + w_{p-1})$ and records the view of each party. P commits to each view $c_i = \texttt{Commit}(\texttt{view}_\texttt{i}; r_i)$ (r_i is commitment randomness) and sends each c_i to V.
- V randomly samples a size $p-1$ subset of parties and sends their ids to P.
- P opens all such commited views to V.
- V checks that communication between the opened parties is consistent with the protocol description. If not V rejects. V also checks that each emulated party outputs 1. If so, V accepts.

In a p-party protocol, there are $p(p-1)$ total communication channels, and V opens $(p-1)(p-2)$ of those channels. Again, to cheat it is necessary for cheating prover to corrupt at least *one* of these channels, so the cheating prover will be caught with *at least* the following probability:

$$\texttt{Pr}_{\texttt{catch}} \geq \frac{(p-1)(p-2)}{p(p-1)} = \frac{p-2}{p}$$

Setting $p = 3$, we see that V catches a cheating P with probability at least $1/3$.

Soundness amplification. Of course, we might want *much* better soundness than this. For instance, we might want to achieve a soundness error that is negligible in the security parameter. This can be achieved by simply repeating the protocol many times. This is a standard trick in interactive proof systems.

Improving Efficiency

So far, the protocol we have described is incredibly inefficient. Indeed, if we recall how the GMW protocol works, it involves pairwise utilization of Oblivious Transfer (OT). Therefore, if we plug GMW into the above schematic as is, we implicitly make use non-black-box OT: the prover P must write down all messages sent/received as part of the OT protocol. Note that this implies non-black-box usage of some cryptographic primitive, e.g. DDH-friendly groups.

One crucial and elegant observation is that due to the structure of the above proof, we can replace invocations of the OT protocol by its ideal functionality. Indeed, V already checks communication channels between parties, so it can just as easily check that the ideal OT functionality is properly executed between that pair of parties.

With this crucial optimization and setting $p = 3$, the total communication cost per repetition is only $O(|C| + \lambda)$ bits. Essentially, the whole proof involves (1) transmitting commitments, and (2) transmitting views of size $O(|C|)$. Of course, achieving negligible soundness error requires repeating the protocol $O(\lambda)$ times.

One major improvement of subsequent works in the area of MPC-in-the-head was to (1) decrease the number of needed repetitions and (2) achieve *sublinear* scaling in the circuit size. See e.g. [BBD+23]. A key technique for achieving this improvements is to change the underlying MPC protocol to something more suited to the setting.

Non-interactive ZK Proofs (NIZKs)

So far, the scheme we have described is a so-called *designated verifier* proof. Namely, the prover and verifier interact, and that interaction convinces only that verifier of the truth of the statement. If P wishes to convince some other verifier, she will need to run the proof again.

In many cases, it might be desirable for P to prove the desired statement "once and for all". Namely, P would like to one time write down a proof π, then distribute π to verifiers, or even post π online. The

document π achieves all the desired guarantees of an interactive proof: each verifier learns the proved statement must be true, while learning nothing about V's witness. Such an object is called a *non-interactive* zero-knowledge proof.

The MPC-in-the-head proof system sketched above can be used to achieve NIZKs in the random oracle model, using the *Fiat Shamir* heuristic [FS87]. The key observation about our ZK protocol is that it is a so-called *public coin* protocol. This means that the verifier V has no private randomness. Indeed, V does have randomness (the choice of which parties it opens), but these choices are not "secret". It is as if, once P commits to party views, V can flip its coins in plain view of the prover P.

Public-coin protocols can be made non-interactive in the random oracle model. The idea is simply this: rather than having V flip random coins, we will have the random oracle flip the coins. This is the Fiat Shamir heuristic.

In particular, our ZK protocol involves three rounds:

- **Commit:** P sends commitments of party views to V.
- **Challenge:** V sends back a public-coin random challenge.
- **Response:** P opens party views according to the challenge.

(Note, this style of three-round protocol is called a *sigma protocol*.) In the non-interactive version of the protocol, P does not send its commitments to V (indeed, there is no V), but rather inputs the commitments to the random oracle. The output of the RO is treated as the random challenge. P's proof π is the concatenation of the first-round commitments with the third-round openings.

To check the proof π, any verifier can check that the third-round response is consistent with randomness sampled from the RO. If so, the verifier performs the rest of the verification normally.

Thus, the MPC-in-the-head paradigm can yield simple, reasonably efficient proof systems, and they can even achieve non-interactivity.

Next Time

We have now seen our first techniques for achieving maliciously-secure protocols, and we have introduced the idea of emulating MPC players. From here, we will investigate more concretely-efficient techniques for achieving malicious security.

References

[BBD+23] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 581–615. Springer, Cham, August 2023.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.