Recent progress in MPC-in-the-Head protocols

Emmanuela Orsini

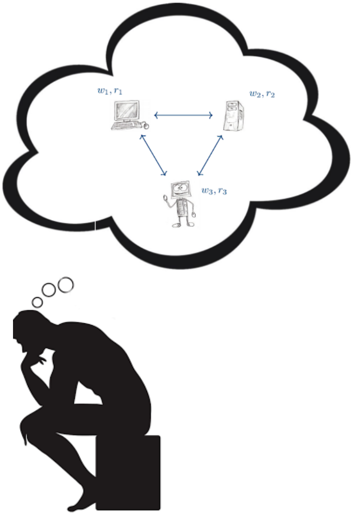
imec-COSIC, KU Leuven

MPC-in-the-Head

MPC-in-the-Head framework [Ishai,Kushilevitz,Ostrovsky,Sahai, 2007]

- Honest majority, information-theoretic secure MPC
- Completeness, soundness and ZK derived from the security of MPC

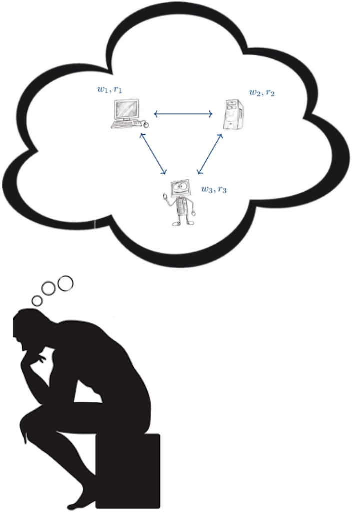
MPC in the head – Description



view_1	view_2	view_3	$\cdots$	view_i	$\cdots$	view_n

Prover

MPC in the head – Description



Prover

π

view$_1$	view$_2$	view$_3$	$\cdots$	view$_i$	$\cdots$	view$_n$

Efficient MPCitH protocols

- **ZKBoo** (Giacomelli, Madsen, Orlandi, *USENIX Security*, 2016), **ZKB++** (Chase, Derler, Goldfeder, Orlandi, Ramacher, Rechberger, Slamanig, Zaverucha, *ACM CCS 2017*)
- **KKW** (Katz, Kolesnikov, Wang, *ACM CCS* 2018)
- **BN** (Baum, Nof, *PKC 2020*)
- **Ligero, Ligero++** (Ames, Hazay, Ishai, Venkitasubramaniam, *ACM 2017, ACM 2020*), **BooLigero** (Yaron Gvili, Sarah Scheffler, Mayank Varia) *FC2021*

Efficient MPCitH protocols

- **ZKBoo** (Giacomelli, Madsen, Orlandi, *USENIX Security*, 2016), **ZKB++** (Chase, Derler, Goldfeder, Orlandi, Ramacher, Rechberger, Slamanig, Zaverucha, *ACM CCS 2017*)
- **KKW** (Katz, Kolesnikov, Wang, *ACM CCS* 2018)
- **BN** (Baum, Nof, *PKC 2020*)
- **Ligero, Ligero++** (Ames, Hazay, Ishai, Venkitasubramaniam, *ACM 2017, 2020*), **BooLigero** (Yaron Gvili, Sarah Scheffler, Mayank Varia) *FC2021*

Efficient MPCitH protocols – Common features

- Linear prover and communication*
- Post-quantum security
- Great flexibility**
- Possibility of streaming
- Stackable^{+} (Eprint 2021/422, A. Goel, M. Green, M. Hall-Andersen,G. Kaptchuk)

Limbo: MPCitH-based zk-IOP

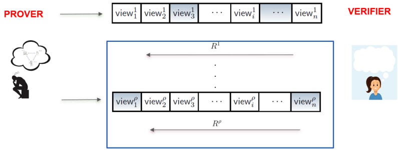
⋆ C. de Saint Guilhem, E. Orsini, T. Tanguy, *Limbo: Efficient Zero-knowledge MPCitH-based Arguments*, ACM CCS 2021.

Highlights:

- Naturally works over any fields
- More efficient in computation compared to other MPCitH protocols
- Concretely small linear proof size

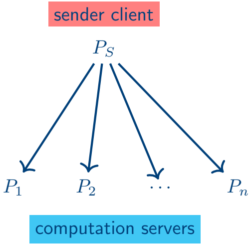
Limbo: General construction

1. Describe a general MPC model with arbitrary number of rounds
2. Compile our general MPC protocol into a zk-IOP
3. Instantiate the MPC model with an actual MPC protocol
4. Compile the resulting zk-proof system to obtain an interactive or non-interactive argument



Limbo: MPC model

Define an MPC protocol in the client-server model with ρ phases

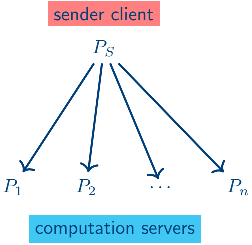


- P_S knows the entire input of the computation
- P_S sends at **most one message at the beginning of each phase**
- The servers only communicate with each other via broadcast

1. P_S sends $\mathbf{m_i^1}, \forall i$ (Phase 1)
2. For each $j \in [2, \rho - 1]$
   - P_i call RandomCoin
   - P_S sends $\mathbf{m_i^j}$ (Phase j)
3. P_i call RandomCoin and, according to the received last message, output (Phase ρ)

Limbo: MPC model

Define an MPC protocol in the client-server model with ρ phases

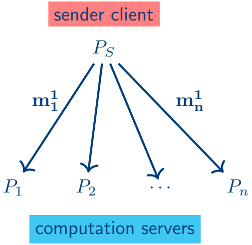


- P_S knows the entire input of the computation
- P_S sends at **most one message at the beginning of each phase**
- ~~The servers only communicate with each other via broadcast~~

1. P_S sends $\mathbf{m_i^1}, \forall i$ (Phase 1)
2. For each $j \in [2, \rho - 1]$
   - P_i call RandomCoin
   - P_S sends $\mathbf{m_i^j}$ (Phase j)
3. P_i call RandomCoin and, according to the received last message, output (Phase ρ)

Limbo: MPC model

Define an MPC protocol in the client-server model with ρ phases

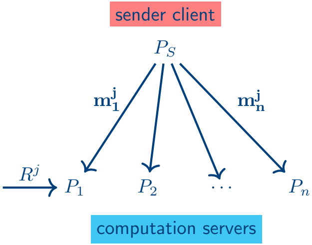


- P_S knows the entire input of the computation
- P_S sends at **most one message at the beginning of each phase**
- ~~The servers only communicate with each other via broadcast~~

1. P_S sends $\mathbf{m_i^1}, \forall i$ (Phase 1)
2. For each $j \in [2, \rho - 1]$
   - P_i call RandomCoin
   - P_S sends $\mathbf{m_i^j}$ (Phase j)
3. P_i call RandomCoin and, according to the received last message, output (Phase ρ)

Limbo: MPC model

Define an MPC protocol in the client-server model with ρ phases

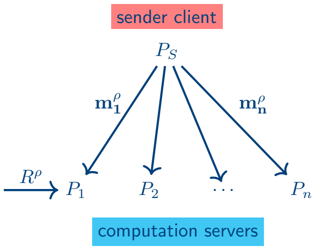


- P_S knows the entire input of the computation
- P_S sends at **most one message at the beginning of each phase**
- ~~The servers only communicate with each other via broadcast~~

1. P_S sends $\mathbf{m_i^1}, \forall i$ (Phase 1)
2. For each $j \in [2, \rho - 1]$
   - P_i call RandomCoin
   - P_S sends $\mathbf{m_i^j}$ (Phase j)
3. P_i call RandomCoin and, according to the received last message, output (Phase ρ)

Limbo: MPC model

Define an MPC protocol in the client-server model with ρ phases

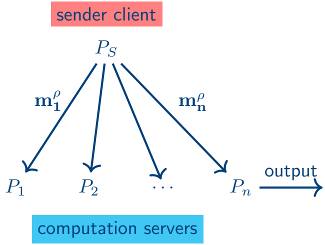


- P_S knows the entire input of the computation
- P_S sends at **most one message at the beginning of each phase**
- ~~The servers only communicate with each other via broadcast~~

1. P_S sends $\mathbf{m_i^1}, \forall i$ (Phase 1)
2. For each $j \in [2, \rho - 1]$
   - P_i call RandomCoin
   - P_S sends $\mathbf{m_i^j}$ (Phase j)
3. P_i call RandomCoin and, according to the received last message, output (Phase ρ)

Limbo: MPC model

Define an MPC protocol in the client-server model with ρ phases



- P_S knows the entire input of the computation
- P_S sends at **most one message at the beginning of each phase**
- ~~The servers only communicate with each other via broadcast~~

1. P_S sends $\mathbf{m_i^1}, \forall i$ (Phase 1)
2. For each $j \in [2, \rho - 1]$
   - P_i call RandomCoin
   - P_S sends $\mathbf{m_i^j}$ (Phase j)
3. P_i call RandomCoin and, according to the received last message, output (Phase ρ)

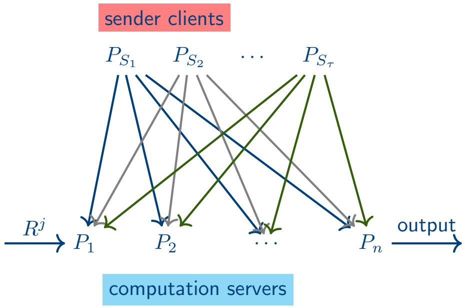
Improving soundness

Let Π_f a ρ-phase MPC protocol in the client/server model with $(0, n-1)$-privacy and $(P_S, 0)$-robustness with robustness error δ, the protocol $\Pi_{\rho-\mathsf{zkIOP}}$ is a ZKIOP with soundness error

$$1/n + \delta \cdot (1 - 1/n)$$

Amplify soundness: RandomCoin shared across multiple execution

Improving soundness



- **Note**. Oracle queries are not calls to RandomCoin
- The new soundness is

$$(1/n)^\tau + \delta \cdot (1 - (1/n)^\tau)$$

Instantiating the MPC protocol

- $\mathcal{R}$ an NP relation and C over a finite field: $C(w) = 1 \iff (x, w) \in \mathcal{R}$
- MPC protocol divided in 2 phases:
  1. **Input/evaluation phase**: P_S generates and distributes to P_i an additive sharings of the input and output of each multiplication gate. P_i evaluates the circuit locally.
  2. **Multiplication check**: servers check that the multiplication gates were correctly computed
- Main ideas of the test : 1. check inner-products instead of multiplications 2. use a protocol that allows to reduce the size of the inner product.

Checking multiplication [BBCGI19,BGIN19,GS20]

1. Given m multiplication triples over $\mathbb{F}$, $\langle z_i \rangle, \langle y_i \rangle, \langle z_i \rangle$

$$\langle \mathbf{x} \rangle = \boxed{\langle x_1 \rangle \mid R \cdot \langle x_2 \rangle \mid R^2 \cdot \langle x_3 \rangle \mid \cdots \mid \cdots \mid \cdots \mid \cdots \mid R^{m-1} \cdot \langle x_m \rangle}$$

$$\langle \mathbf{y} \rangle = \boxed{\langle y_1 \rangle \mid \langle y_2 \rangle \mid \langle y_3 \rangle \mid \cdots \mid \cdots \mid \cdots \mid \langle y_{m-1} \rangle \mid \langle y_m \rangle}$$

$$\langle z \rangle = \sum_i R^{i-1} \cdot \langle z_i \rangle$$

2. Reduce the dimension of the inner product of a factor k

3. Repeat the previous step until a single triple is obtained $\langle \mathbf{x} \rangle * \langle \mathbf{y} \rangle = \langle z \rangle$

If the final triple is correct, the initial m triples are correct except with negligible probability δ_k.

The resulting schemes

- Given an NP relation $\mathcal{R}$, and a circuit C such that $C(w) = 1 \iff (x, w) \in \mathcal{R}$, and using the MPC protocol described in the previous slide we obtain an interactive ZK protocol with soundness error

  $$\frac{1}{n^t} + \delta_k \cdot (1 - \frac{1}{n^t})$$

  and $\lfloor \log_k(m) \rfloor + 2$.

- We use standard crypto compilers to get the final schemes

Performance

(Interactive) Limbo over $\mathbb{F}_2$, sec=40, $\mathbb{G} = \mathbb{F}_{2^{64}}$

$\lvert C\rvert = 2^{20}$	n	$t_{\mathcal{P}}$ (s)	$t_{\mathcal{V}}$ (s)	size (KB)	per mult
Single threaded	8	1.7	1.5	1878	$1.6\mu s$
	64	7.4	6.8	932	
4 threads	8	0.9	0.8		$< 1\mu s$

(Non-interactive) Limbo over $\mathbb{F}_2$, sec=128, $\mathbb{G} = \mathbb{F}_{2^{64}}$

$\lvert C\rvert = 2^{20}$	n	$t_{\mathcal{P}}$ (s)	$t_{\mathcal{V}}$ (s)	size (KB)	per mult
Single threaded	8	7	5.9	6444	$6.6\mu s$
	64	31	29	4162	
4 threads	8	4.3	2.9		$3.8\mu s$

Optimizations

Beyond the gate-by-gate approach

Proving matrix multiplication $M \times M$ over $\mathbb{F}_2$

$M = 400$	$t_{\mathcal{P}}$ (s)	$t_{\mathcal{V}}$ (s)	size (KB)	per mult
Single threaded	62	57	834	$0.969\mu s$
4 threads	38	32		$0.593\mu s$

Optmizations

- AES: Computation over $\mathbb{F}_{2^8}$
  - Interactive: 170ns (prover computation 1.09ms)
  - Non-interactive: 515ns (prover computation 3.3 ms, $n = 16$) or 375ns ($n = 8$)
- Comparison with other schemes (security L1)

Scheme	$t_{\mathcal{S}}$ (ms)	$t_{\mathcal{V}}$ (ms)	size (bytes)
Picnic	5.33	4.03	12466
Banquet	6.34	4.84	19776
SPHINCS + fast	14.42	1.74	16976
SPHINCS + −small	239.34	0.73	8080
Limbo-Sign 8	2.4	1.9	21369
Limbo-Sign 16	3.3	3.2	18626
Limbo-Sign 57	8.5	8.5	15728

Next steps

Research topic:

- Different MPC protocols
- Tighter soundness analysis
- Generalization to rings
- Different verification checks and gadgets
- Multi-instance case

Implementation:

- Larger fields
- Streaming