PRIVA LEDGE

Verifiable MPC from blockchain

Solving the World's Billionaires' problem

Toon Segers, TU Eindhoven

(joint work with Berry Schoenmakers)

PENCIL Workshop: May 18, 2019

"Capitalism is taking some lumps—and not just in the headlines. For only the second year in a decade, both **the number of billionaires and their total wealth shrank**…"

Source: Forbes.com,
Published March 5, 2019 (link)

Verifiable MPC from blockchain

Solving the World's Billionaires' problem

Outline:

- Benefits of Verifiable Multi-Party Computation
- Example using blockchain
- Next steps, opportunities, challenges

Secure Multiparty Computation

Aim: Correct computation on hidden data

Correctness and privacy depend on setting: malicious parties

- *Honest majority*: Fairness and information theoretic security (≥3 parties)
- *Dishonest majority*: Computational security

Applications:

- Voting, auctions, linear programming, linear regression, decision tree learning

Intrinsic limitations to MPC

Security model stops at protocol boundary, however…

! False inputs

- E.g. in Yao's Millionaires' problem, millionaires can lie about their riches

! All parties corrupt

- Active security up to all-but-one corrupt parties

All relevant when client outsources (i.e. does not participate)

Intrinsic limitations to MPC

Security model stops at protocol boundary, however…

! False inputs

- E.g. in Yao's Millionaires' problem, millionaires can lie about their riches

! All parties corrupt

- Active security up to all-but-one corrupt parties

All relevant when client outsources (i.e. does not participate)

Very relevant in blockchain context

Verifiable MPC by joining MPC, ZK and Blockchain

MPC

Correct computation on hidden data by multiple parties

Zero Knowledge (ZK)

Prover to convince honest verifier of given statement

Without revealing any information

Bulletin Board

Authenticated broadcast channel

Verifiable MPC

Outsider (or general public) to verify correctness of an MPC computation

A false result will not be accepted

Verifiable MPC by joining MPC, ZK and Blockchain

MPC

Correct computation on hidden data by multiple parties

Zero Knowledge (ZK)

Prover to convince honest verifier of given statement

Without revealing any information

Blockchain

Bulletin Board++

Verifiable MPC

Outsider (or general public) to verify correctness of an MPC computation

A false result will not be accepted

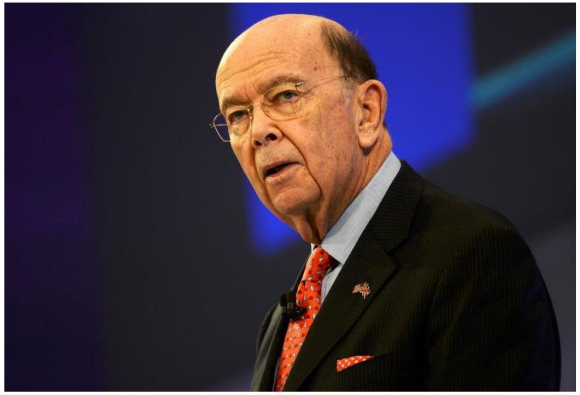
WALL STREET

Forbes says Commerce Secretary Wilbur Ross lied about being a billionaire

PUBLISHED TUE, NOV 7 2017 • 8:06 AM EST | UPDATED TUE, NOV 7 2017 • 4:23 PM EST

Fred Imbert
@FOIMBERT

SHARE



Commerce Secretary Wilbur Ross, speaks at the Confederation of British Industry's annual conference in London, Britain, November 6, 2017.

Mary Turner | Reuters

Source: CNBC.com, Fred Imbert, 'Forbes says Commerce Secretary Wilbur Ross lied about being a billionaire', Published Tue, Nov 7 2017 (link)

The *World's Billionaires* Problem

Extend Yao's *Millionaires' problem:* Privacy of inputs, verifiable inputs and outputs

Verifiable input:

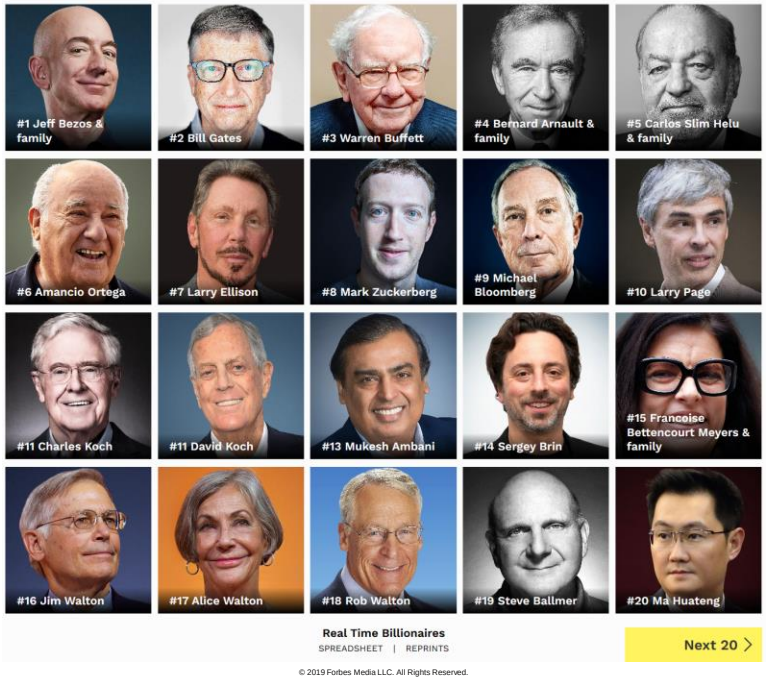
- Commitments of everybody's tax returns
- Signed by the tax authority
- Posted on a blockchain

Verifiable output:

- Top 400 billionaires world-wide

Privacy:

- Privacy for all outside top 400



Source: Forbes.com,
Published March 5, 2019 (link)

The *World's Billionaires* Problem

Extend Yao's *Millionaires' problem:* Privacy of inputs, verifiable inputs and outputs

Verifiable input:

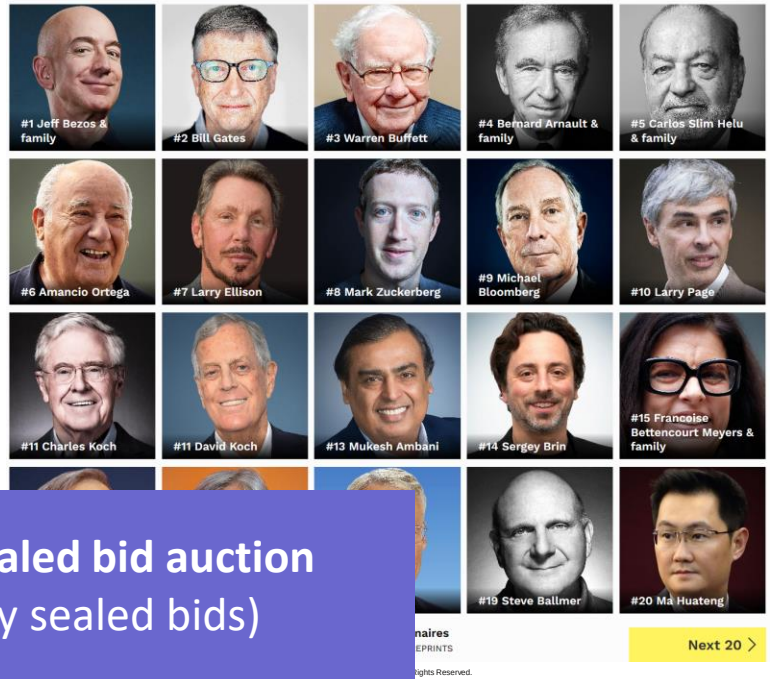
- Commitments of everybody's tax returns
- Signed by the tax authority
- Posted on a blockchain

Verifiable output:

- Top 400 billionaires world-wide

Privacy:

- Privacy for all



World's Billionaires ≈ sealed bid auction
(replace tax returns by sealed bids)

Source: Forbes.com,
Published March 5, 2019 (link)

Verifiable computation with zk-SNARKs

F represented by circuit and 'quadratic program'

- Correct evaluation of circuit, gives wire values **c**

With c, prover can construct polynomial p and divisor h

- Cheating prover unsuccessful: Schwartz-Zippel Lemma

Zero knowledge proof uses elements in bilinear group

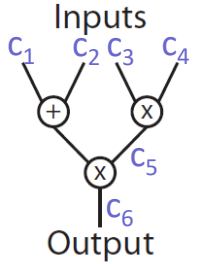
- Hides information on witness in exponent

Proof π_y uses only 9 group (elliptic curve) elements

- π_y includes polynomials v, w, y, such that p = v · w – y, evaluated in s, hidden in exponent

Exhibits

F: y = (x1+x2)*(x3*x4)



p = v · w – y,

$v = (\sum_i c_i \cdot v_i(x))$, w, y similar

π_y =

$$\{\ g^{v_{mid}(s)},\ g^{w(s)},\ g^{y(s)},\ g^{h(s)},\ g^{\alpha v_{mid}(s)},\ g^{\alpha w(s)},\ g^{\alpha y(s)},\ g^{\alpha h(s)},\ g^{\beta_v v(s)+\beta_w w(s)+\beta_y y(s)}\ \}$$

Source: Exhibits from Pinocchio paper [PHGM13]

Pinocchio steps

KeyGen(F; 1^{λ}) → (EK_F; VK_F)

- Trusted party creates *public* evaluation and verification keys for F
- F is represented by circuit of size m

Compute(EK_F; u) → (y; π_y)

- Worker evaluates circuit for F(u) to obtain y ← F(u) and wire values $\{c_k\}_{k\epsilon\{1..m\}}$
- With circuit wires, worker computes proof π_y (bilinear group elements)

Verify(VK_F, u, y, π_y) → {0, 1}

- Verifier uses bilinear map to efficiently verify proof

Pinocchio steps

KeyGen(F; 1^{λ}) → (EK_F; VK_F)

- Trusted party creates *public* evaluation and verification keys for F
- F is represented by circuit of size m

Compute(EK_F; u) → (y; π_y)

- Worker evaluates circuit for F(u) to obtain y ← F(u) and wire values $\{c_k\}_{k\epsilon\{1..m\}}$
- With circuit wires, worker computes proof π_y (bilinear group elements)

Verify(VK_F, u, y, π_y) → {0, 1}

- Verifier uses bilinear map to efficiently verify proof

Verification very efficient (ms);
Proof construction expensive: e.g. 37s for Zcash;
Recently reduced to 2.3s with 2016 result from Groth

Pinocchio with privacy: Trinocchio

Trinocchio: Privacy and input independence

- By Schoenmakers, Veeningen, De Vreede (2015)

Input parties		Post commitments of inputs, for input independence
Workers		Compute block π_i with Block Evaluation Key BEK_i via MPC
Client(s)		Use Block Verification Key VEK_i to verify π_i

Geppetri (2017): Trusted setup independent of F and reusable

- Enables efficient reuse of data committed by third-party

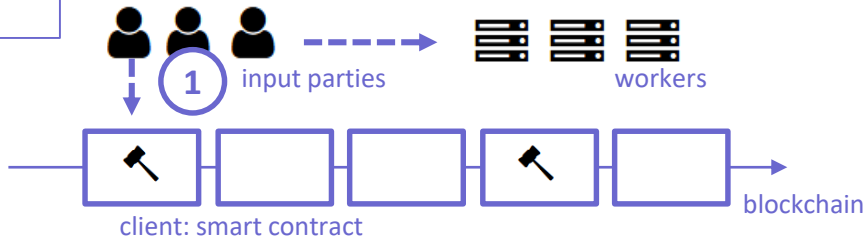
Idea: Trinocchio with blockchain

Example: Sealed-bid auction

1

Input parties

- Communicate shares of bids to workers;
- Post commitments to bids to auction contract



Idea: Trinocchio with blockchain

Example: Sealed-bid auction

1

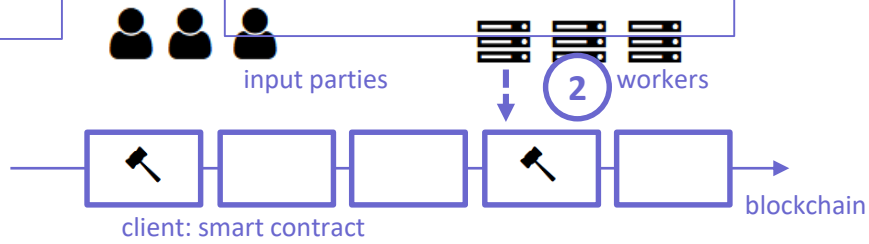
Input parties

- Communicate shares of bids to workers;
- Post commitments to bids to auction contract

2

Workers

- Compute shares of output and proof
- Post both to contract



Idea: Trinocchio with blockchain

Example: Sealed-bid auction

1

Input parties

- Communicate shares of bids to workers;
- Post commitments to bids to auction contract

2

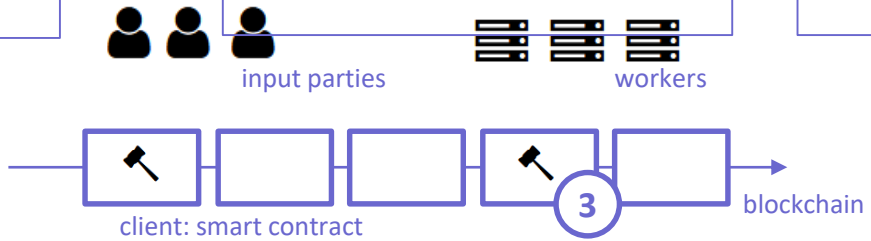
Workers

- Compute shares of output and proof
- Post both to contract

3

Contract (“client”)

- Stores output
- Verifies proof by recombining shares



Next steps: Extend MPyC

lschoe/mpyc

Python package for MPC

- Successor of VIFF (see viff.dk)
- Based on Shamir-secret sharing and pseudo-random secret sharing

Focus on usability

- Expressive, small footprint, high-level, open and free
- Convenient abstraction with operator overloading and async evaluation of underlying protocols

Our next steps:

- Expand MPyC with ‘Verifiable MPC’ and interaction with blockchain
- Develop Verifiable MPC with other ZK protocols, likely Bulletproofs

Conclusion

Value of Verifiable MPC

- Valuable for use cases with sensitive input
- Efficient re-use of data particularly interesting (ideally 3rd party attested)
- Blockchain to instantiate bulletin board

Improvement opportunities

- Efficiency: waiting time for client (round complexity matters)
- Cheater detection: identify and deter cheating workers
- Fairness: ensure that if a party receives the result, then all do
- Trapdoor: avoid or secure trusted setup

Questions

- Further improvements to Verifiable MPC? (see above)
- Could functionalities of blockchain enable those?

Thank you,

Toon Segers, TU Eindhoven

APPENDIX

Informal Pinocchio proof approach

- **Represent F by circuit, circuit by QAP**
  - N in/outputs, d mult. gates
- **Definition: QAP over prime field**
  - Poly's $\{v_k\}$, $\{w_k\}$, $\{y_k\}$, and target poly t (all public)
  - $\mathbf{c} \in \mathbb{F}^m$ exists s.t. t divides $p = V \cdot W - Y$; here $V = (\sum_i c_i \cdot v_i(x))$, W and Y similar
- **Prove knowledge of c satisfying QAP_F**
  - If worker knows witness c, can construct p(x) and h(x) = p(x)/t(x)
  - Probability p(s)=h(s) · t(s) for random s in large prime field $\mathbb{F}$ very small (Schwartz-Zippel)
- **Protocol (informal):**
  - Protocol hides witness in exponent of bilinear group (two different elliptic curves)
  - Setup: Public evaluation and verification keys for F, random s
    - Keys contain $\{v_k\}$, $\{w_k\}$, $\{y_k\}$ evaluated in s hidden 'in exponent' (e.g. $g^{v\_k(s)}$)
  - Worker: Computes $y \leftarrow F(u)$ and π_y by evaluating circuit
    - π_y contains hidings of V(s), W(s), Y(s) and t(s) evaluated in s
  - Client: Checks V(s) · W(s)) – Y(s) = t(s) · h(s) in exponent (a.o.)

Trinocchio steps

Setup
- Trusted party creates trusted commitment keys (“mixed commitments”)
- Trusted party creates public Evaluation and Verification Keys
- Trusted party throws away trapdoor information

Input
- Input parties post commitment to its input blocks first (needed for input independence)
- Input parties open commitments for client(s) to verify; then provide secret-shared inputs to workers
  - MPC basis: Shamir secret sharing, multiplication protocol from Gennaro et al. (1998)
- Workers check if shares correspond to the broadcast blocks

Compute
- Workers compute function F, produce Pinocchio proof of correct computation
  - Calculation of polynomial h mostly local with FFT
  - Computation over bilinear group elements all performed locally
- Workers communicate shares of function output to the client(s)
- Workers then post the shares of the proof elements to the bulletin board (randomized for ZK)

Result
- Client(s) obtain their results and verify them w.r.t. information on the bulletin board

Trinocchio: Additional background on changed setup vs Pinocchio

- Multiple input parties, workers and clients
- Privacy of I/O by introducing proof *blocks* π_i for all N I/O parties
  - π_i: Includes proof π terms restricted to a subset of wires (inputs of party i)
- Pinocchio's **KeyGen** adapted to **MultiKeyGen**:
  - EK becomes $\{BEK_i\}$ and VK becomes $\{BVK_i\}$, with separate random for each
  - BEK_i, BVK_i: Only include EK and VK terms for relevant wires
- Setup expanded to include commitments of inputs - for input independence

Back-up

Trinocchio with blockchain (detailed 1/2)
Scenario with multiple inputs, public output

	Input phase	Compute phase	Result phase
Off-chain	Trusted party: KeyGen(F; 1^{λ})→ $(BEK_i; BVK_i)_{i\in\{1..N\}}$ Input party i: CreateShare(x_i)→ $[[x_i]]$ ShareToWorkers($[[x_i]]$)	Workers: Compute(BEK_i; x_i) → ($[[y]]$; $[[\pi_y]]$)	
On-chain	Input party i: Post(Commitment($[[x_i]]$))	Workers: Post($[[\pi_i]]$, $[[y]]$)	Client (contract): Recombine($[[y]]$) → y Recombine($[[\pi_i]]$) → π_y Verify(BVK_i, y, π_i)

Note: Input-dependent CRS generation is omitted for simplicity

Back-up

Trinocchio with blockchain (detailed 2/2)
Multiple private outputs

	Input phase	Compute phase	Result phase
Off-chain	Trusted party: KeyGen(F; 1^λ)→ (BEK_i; BVK_i) $_{i\epsilon\{1..N\}}$ Input party i: CreateShare(x_i)→ [[x_i]] ShareToWorkers([[x_i]])	Workers: Compute(BEK_i; x_i) → ([[y_i]]; [[π_i]]) ShareToClients([[y_i]])	Clients: Recombine([[y_i]]) → y_i Recombine([[π_i]]) → π_i Verify(BVK_i, y_i, π_i)
On-chain	Input party i: Post(Commitment([[x_i]])	Workers: Post([[π_i]])	No on-chain verification in this case; verification of π_i requires private output y_i

Note: Input-dependent CRS generation is omitted for simplicity

Prior work relevant to blockchain (examples)

Private outsourcing

- **Make smart contract computation and validation private**
  - Zether [BAZB19] uses Σ-Bullets (based on Bulletproofs from [BBB+18]) to build a new smart contract that keeps account balances encrypted
    - Avoids commitments and uses ElGamal encryptions with messages in the exponent based on [CGS97]
    - Could implement sealed bid auctions on Ethereum (via a smart contract)
  - E.g. ARPA [ZSXC18], Enigma [ZNP15], HAWK[1] [KMS+16] using MPC (or TEEs[2])
- **Conduct multiple transaction rounds off-chain, aggregate on-chain**
  - E.g. Bolt [GM17] implements anonymous payment channels using NIZKs to proof knowledge of a committed value and that a committed value is in a range

Universal Verifiability

- **Electronic voting: Use blockchain as a ballot box, use smart contracts to verify correctness of result**
  - [YLS+18] describe a platform-independent voting system based on a smart contract blockchain, Paillier encryption, message membership ZKP and ring signatures
- **Compress chain history to succinct proof**
  - Avoid validators having to download full transaction history by using (recursive) SNARKs/STARKs; E.g. Tezos and CODA explore this
- **Avoid trusted setup on increase security**
  - Avoid trusted setup: Zcoin to remove trusted setup by using Σ-protocol from [GK14]
  - Apply MPC to trusted setup: Zcash 'powers of tau' ceremony based on [BMG17]

1. HAWK's "strictly generalizes ZeroCash since Zerocash implements only private money transfers [..] without programmability". Zerocash: [BCG+14]
2. Trusted Execution Environments; Typically Intel SGX

References (I)

[BAZB19] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards Privacy in a Smart Contract World

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, Greg Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. 2018 IEEE Symposium on Security and Privacy (SP)

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity

[BCG+14] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In S&P, 2014.

[BGM17] Sean Bowe and Ariel Gabizon and Ian Miers. Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model, 2017

[BKM17] Iddo Bentov, Ranjit Kumaresan, and Andrew Miller. Instantaneous decentralized poker. In Tsuyoshi Takagi and Thomas Peyrin, editors, Advances in Cryptology - ASIACRYPT 2017

[CDN01] R. Cramer, I. Damgard, and J.B. Nielsen. Multiparty computation from threshold homomorphic encryption. In EUROCRYPT 2001, volume 2045 of LNCS, pages 280{300. Springer-Verlag, 2001.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally ecient multi-authority election scheme. In Walter Fumy, editor, EUROCRYPT'97, volume 1233 of LNCS, pages 103{118. Springer, May 1997.

[dH12] Sebastiaan de Hoogh. Design of large scale applications of secure multiparty computation : secure linear programming. PhD thesis 2012.

[G10] Martin Geisler. Cryptographic Protocols: Theory and Implementation. PhD Dissertation

[GK14] Jens Groth and Markulf Kohlweiss. One-out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin

[GM17] Matthew Green, Ian Miers. Bolt: Anonymous Payment Channels for Decentralized Currencies

[GMW87] O. Goldreich, S. Micali, A. Wigderson. How to play ANY mental game, STOC '87 Proceedings of the nineteenth annual ACM symposium on Theory of computing

References (II)

[KAS18+] Eleftherios Kokoris-Kogias, Enis Ceyhun Alp, Sandra Deepthy Siby, Nicolas Gailly, Philipp Jovanovic, Linus Gasser, and Bryan Ford. Hidden in plain sight: Storing and managing secrets on a public ledger. IACR Cryptology ePrint Archive, 2018:209, 2018.

[KMS+16] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In IEEE Symposium on Security and Privacy, SP 2016

[KZZ16] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In Fischlin and Coron (2016), pages 705–734.

[L16] H. Lipmaa. Prover-Efficient Commit-and-Prove Zero-Knowledge SNARKs. In Proceedings AFRICACRYPT, 2016.

[MBB+18] Patrick McCorry and Surya Bakshi and Iddo Bentov and Andrew Miller and Sarah Meiklejohn. Pisa: Arbitration Outsourcing for State Channels

[PHGM13] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly Practical Verifiable Computation. In Proceedings of S&P, 2013.

[SSV19] Alessandra Scafuro and Luisa Siniscalchi and Ivan Visconti. Publicly Verifiable Proofs from Blockchains

[SVV15] Berry Schoenmakers, Meilof Veeningen, and Niels de Vreede. Trinocchio: Privacy-Preserving Outsourcing by Distributed Verifiable Computation. ACNS 2016: Applied Cryptography and Network Security pp 346-366

[V17] Meilof Veeningen. Pinocchio-Based Adaptive zk-SNARKs and Secure/Correct Adaptive Function Evaluation. 2017

[WW12] Tzer jen Wei and Lih-Chung Wang. A fast mental poker protocol. J. Mathematical Cryptology, 6(1):39–68, 2012

[YLS+18] Bin Yu and Joseph Liu and Amin Sakzad and Surya Nepal and Paul Rimba and Ron Steinfeld and Man Ho Au. Platform-independent Secure Blockchain-Based Voting System. 21st Information Security Conference

[ZNP15] Guy Zyskind, Oz Nathan, Alex ‘Sandy’ Pentland. Enigma: Decentralized Computation Platform with Guaranteed Privacy. PhD thesis. arXiv 2015

[ZSXC18] Derek Zhang and Alex Su and Felix Xu and Jiang Chen, ‘ARPA Whitepaper’