ZKProof Community Event
The Edge, Deloitte, Amsterdam

October 29, 2019

Verifiable MPC

Berry Schoenmakers
Coding & Crypto group
Dept of Mathematics & Computer Science

PRIVA LEDGE

SODA
Scalable Oblivious Data Analytics

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Outline

Secure Multiparty Computation (MPC)

I. MPyC @ TUE

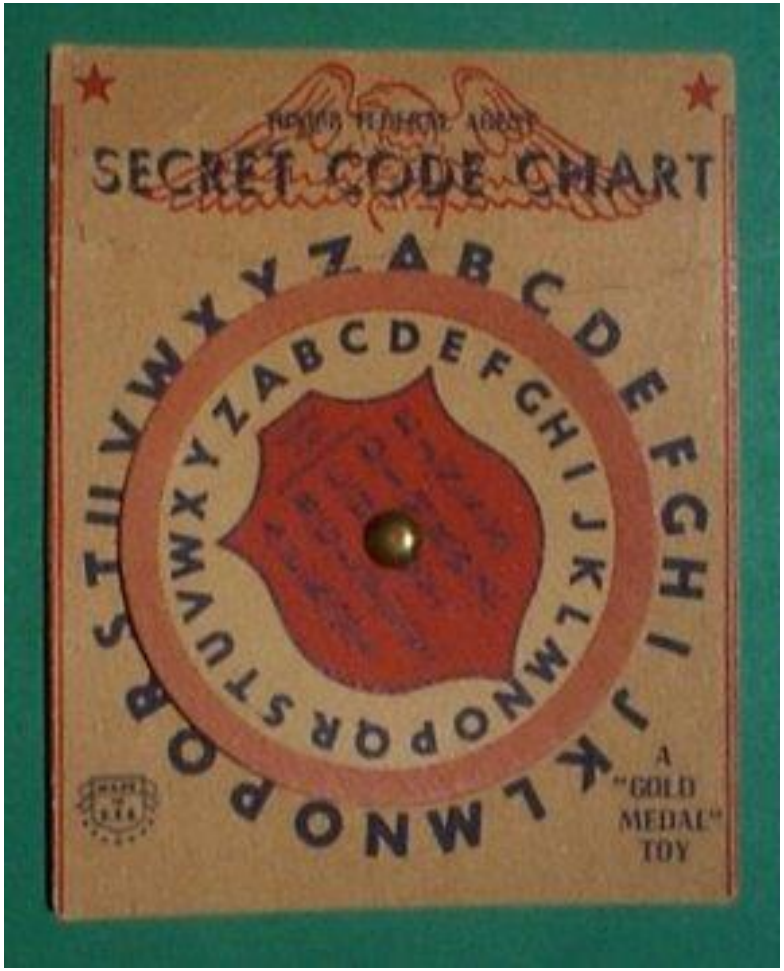
II. Verifiable MPC

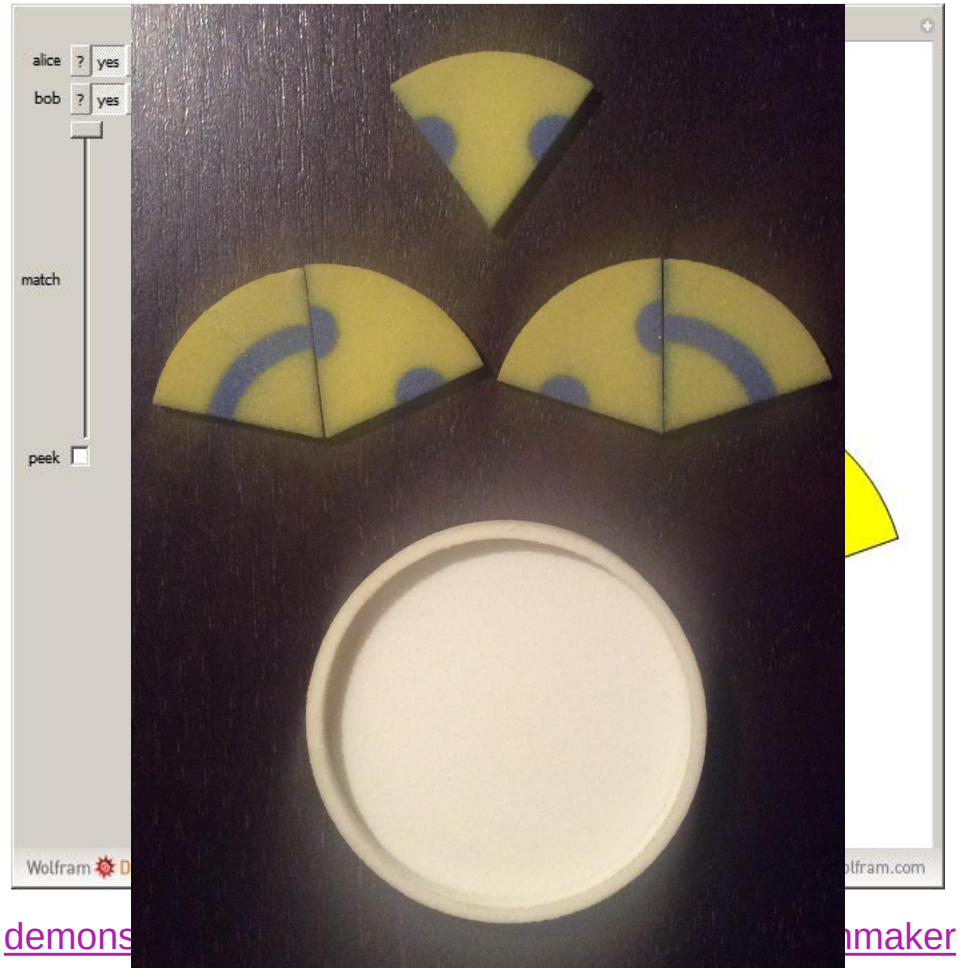
III. Verifiable MPyC

Part I

MPyC @ TUE

We've come a long way …





demons nmaker

Julius Caesar's
Crypto 1.0 gadget

Tom Verhoeff's
Crypto 2.0 gadget

Crypto 1.0

Crypto 1.0 concerns
- encryption and authentication of data
- during communication and storage/retrieval
protecting against malicious outsiders

Crypto 1.0 primitives:

- **Keyless**
  - Cryptographic hash functions
  - Hash chains, Merkle trees
- **Symmetric (secret key)**
  - Stream/block ciphers
  - Message authentication codes
- **Asymmetric (public key)**
  - Public-key encryption
  - Digital signatures
  - Key-exchange protocols

Modern Research into Crypto 1.0:

- Side channel resistant crypto
- Post Quantum crypto
- Lightweight crypto
- Quantum crypto
- …

Crypto 2.0

Crypto 2.0 additionally concerns

- **hiding identity of data owners or any link with them**
- **partial information release of data**
- **computing with encrypted data**

protecting against malicious insiders (your protocol partners)

Crypto 2.0 primitives:

- **homomorphic encryption** — Rivest/Adleman/Dertouzos ’78
- **secret sharing** — Blakley ’79, Shamir ’79
- **blind signatures** — Chaum ’82
- **oblivious transfer** — M. Rabin. ’81, EGL ’85
- **zero-knowledge proofs** — Goldwasser/Micali/Rackoff ’85, GMW’ 86
- **secure two/multiparty computation** — Yao ’82-86, GMW’87,BGW’88,CCD’88
- **secure time-stamping** — Haber/Stornetta ’90, BLLV’ 98
- **functional encryption** — Sahai/Waters ’05, Boneh/Sahai/Waters ’11, GKPVZ ’13
- **fully homomorphic encryption** — Gentry ’09
- **indistinguishability obfuscation** — Garg/Gentry/Halevi/Raykova/Sahai/Waters ’13

MPC @ TU Eindhoven

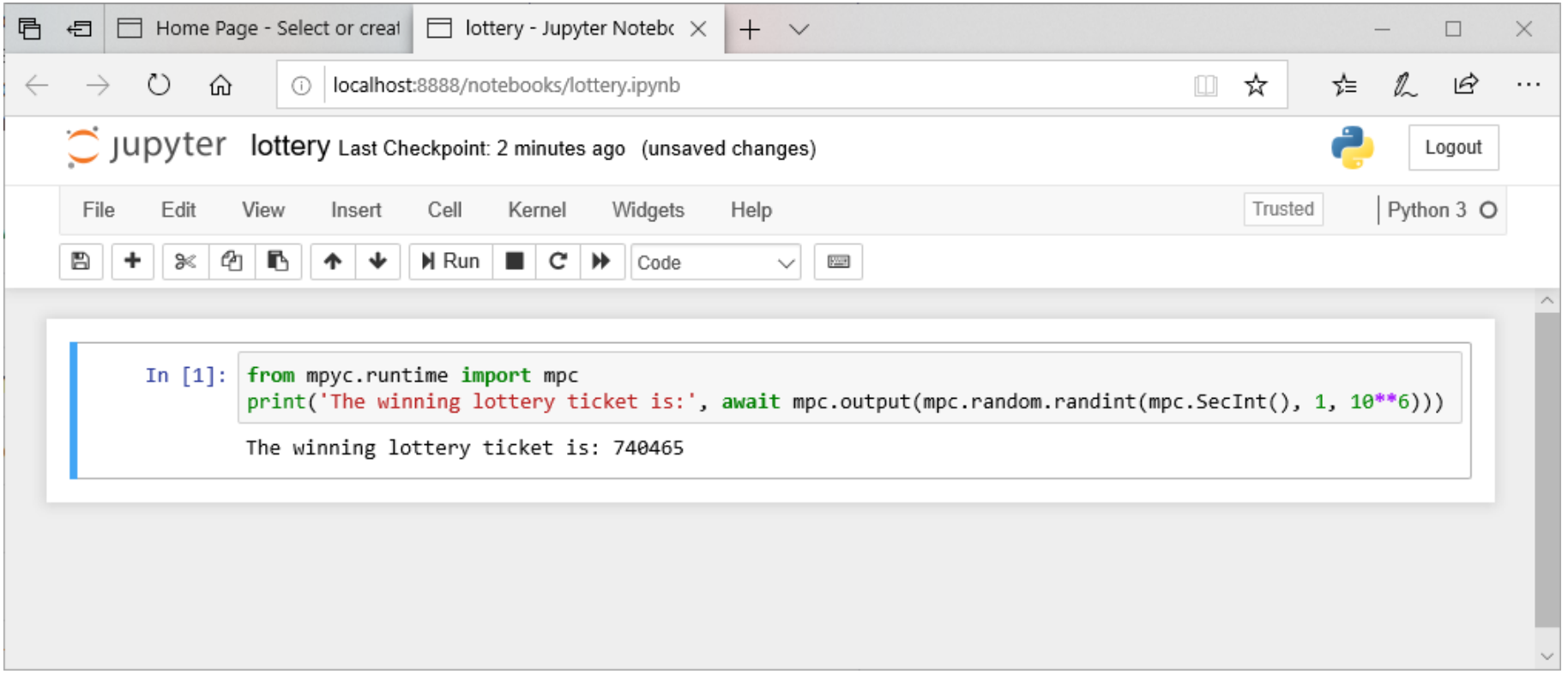
- **PhD students:**
  - **Andrey Sidorenko, Mehmet Kiraz, José Villegas, Sebastiaan de Hoogh**
  - **Current: Niels de Vreede, Frank Blom, Toon Segers**
- **Postdocs:**
  - **Tomas Toft, Mikkel Krøigård, Meilof Veeningen**
  - **Current: Niek Bouman, Stan Korzilius**
- **Research projects:**
  - **Cybervote, PASC, SecureSCM, CACE, PRACTICE, THeCS**
  - **Current: SODA, PRIVILEDGE**

- **Verifiable MPC = MPC + ZKP**

MPyC Secure Multiparty Computation in Python

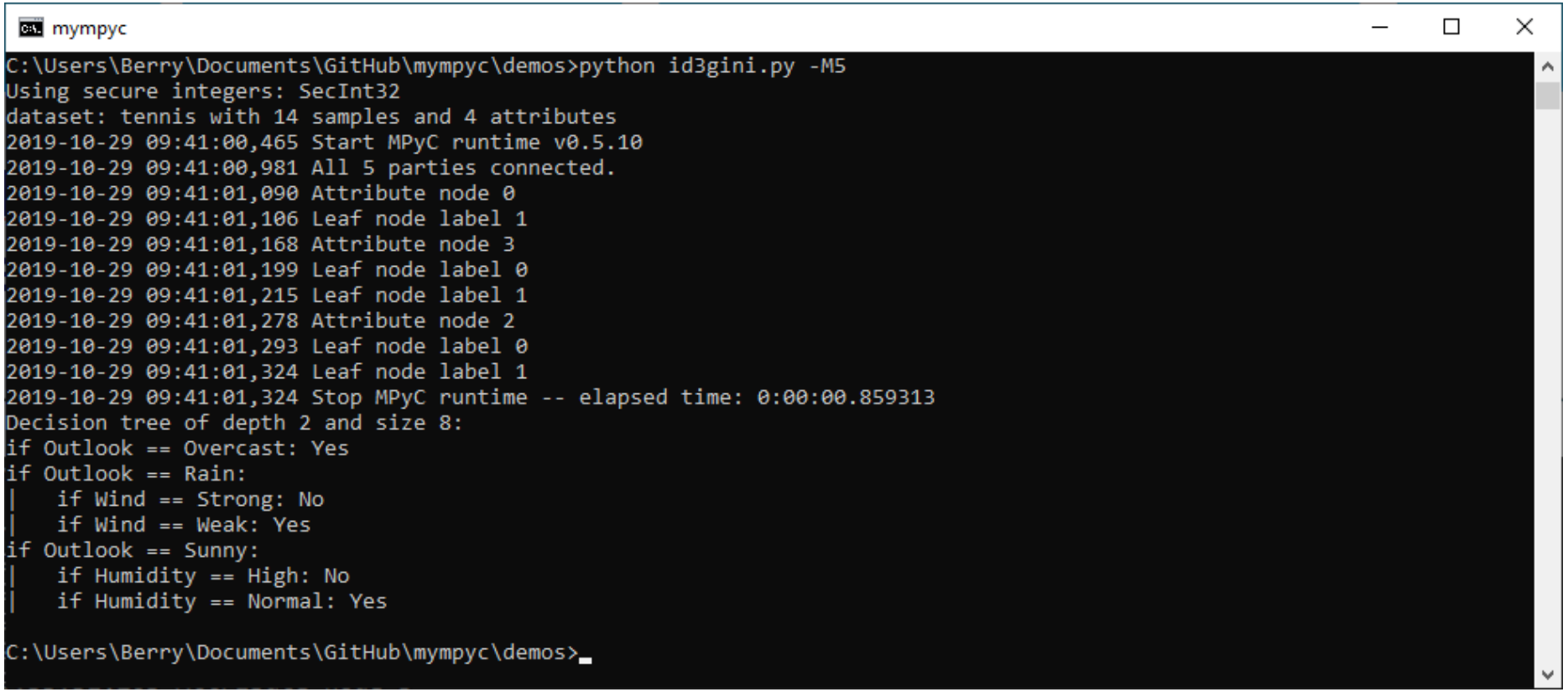
- **VIFF (2007) --> TUeVIFF (2012) --> MPyC (2018)**

- **Secure lottery in MPyC:**



Privacy-Preserving Machine Learning

- **MPyC demos: ID3 decision trees, linear/ridge regression, neural networks (CNN and binarized MLP), Kaplan-Meier survival analysis, …**



Part II

Verifiable MPC

Trust in MPC?

- **Can we trust the outcome of an MPC protocol?**
- **Yes, possibly:**
  - **e.g., if you take part in a 2-party protocol**
  - **or, if you take part in a m-party protocol tolerating m-1 corruptions (everyone else potentially corrupt)**
- **No, if you do not take part!**
  - **MPC gives no security if all parties are corrupt!!**

Similar to situation for ZKPs?

- **Suppose you observe ZKP run between P and V:**
  - **P sends announcement a to V**
  - **V sends challenge c to P**
  - **P sends response r to V**

commitment to a nonce

- **Q: How convincing is this proof?**

- **A: Depends on who needs convincing!**
  - **V should be convinced**
  - **But as an observer you shouldn't**
    - **You may as well been watching a simulated (a;c;r)**
- **Make ZKP non-interactive to convince anyone.**

Limited Scope of MPC

- MPC → secure function evaluation y = f(x) hiding input x

- What MPC does not achieve:
  - MPC does not stop parties from entering bogus inputs
    - e.g., Yao's millionaires (1982) can lie about their riches

  - MPC in outsourcing scenario:
    - Parties performing MPC could ALL be corrupt.
    - Wrong result y* ≠ f(x) cannot be detected.
      - Simulation-based security: indistinguishable views !
      - Active security does not help !

The *World's Billionaires* Problem

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

Verifiably correct input:

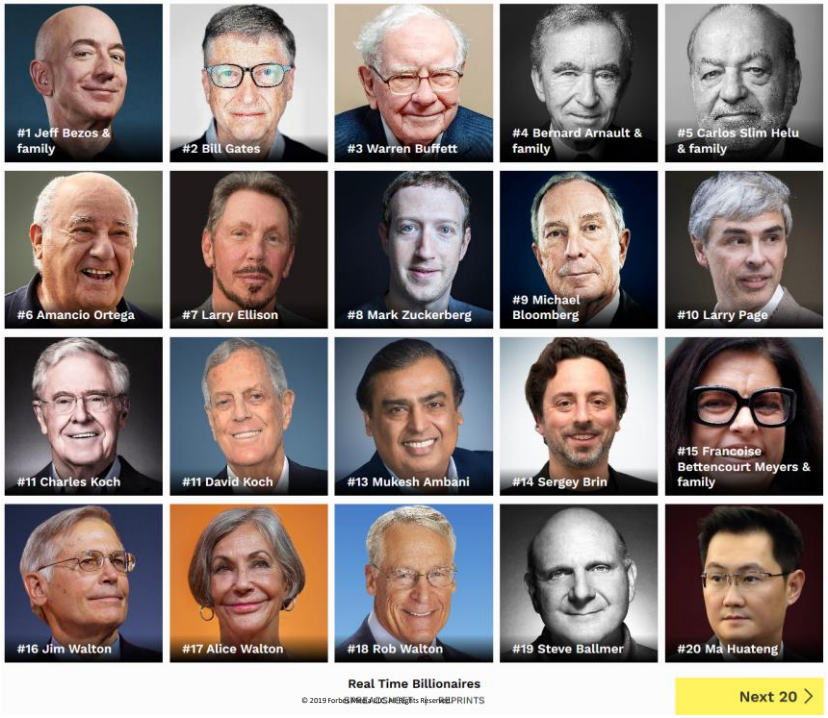
- **Committed/encrypted tax returns**
- **Signed by the tax authority**
- **Posted on a blockchain**

Verifiably correct output:

- **Top 400 billionaires world-wide**

Privacy:

- **Privacy for all outside top 400**



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

Wannabe billionaires …

WALL STREET

Forbes says Commerce Secretary Wilbur Ross lied about being a billionaire

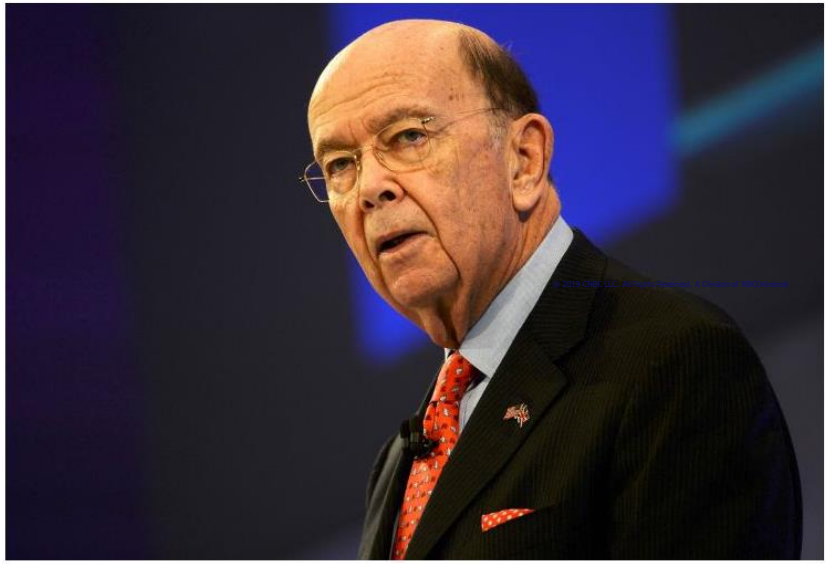
KEY POINTS

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

Fred Imbert
@FOIMBERT

SHARE

- "It seems clear that Ross lied to us," Forbes' report says.



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

Mary Turner | Reuters

The *World's Billionaires* Problem

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

Verifiably correct input:

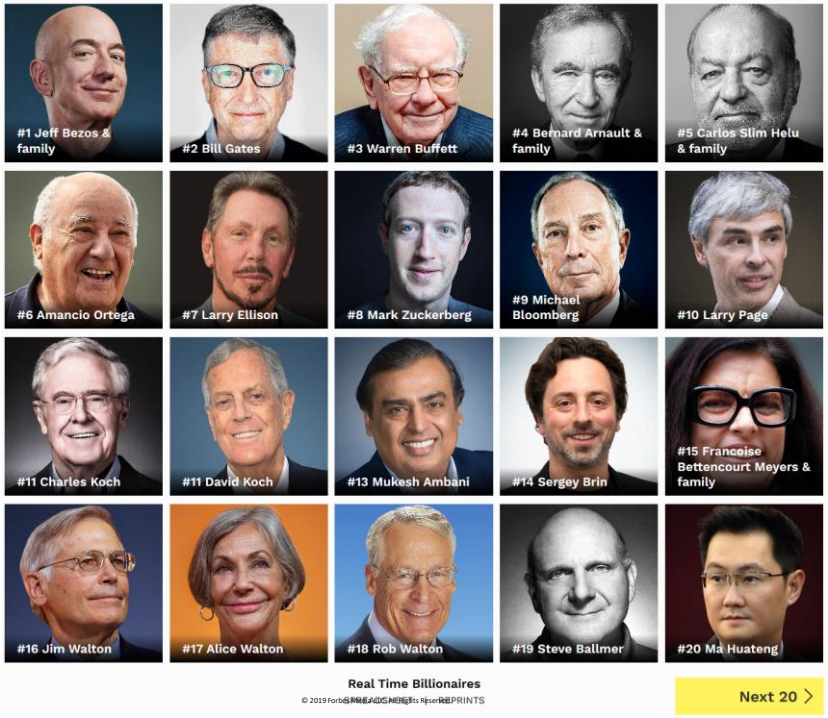
- **Committed/encrypted tax returns**
- **Signed by the tax authority**
- **Posted on a blockchain**

Verifiably correct output:

- **Top 400 billionaires world-wide**

Privacy:

- **Privacy for all outside top 400**



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

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

Part III

Verifiable MPyC

Key ingredients for verifiable MPC

- **Verifiable input x:**
  - **Committed or encrypted input values**
    - **Public input values also possible**
  - **Digitally signed**
  - **Optionally, posted on blockchain (timestamp, uniqueness)**

ElGamal encrypted inputs E(x_i)

- **Verifiable output y = f(x):**
  - **Committed or encrypted output values**
    - **Public output values also possible**
  - **Threshold signed**
  - **Optionally, posted on blockchain**
  - **Noninteractive ZKP that y= f(x) holds.**

ElGamal encrypted output E(y) with y = $\prod_i x_i$ + ZKP that this holds

Computation vs proofs (verification)

- For y = f(x):
  - Compute y from x
  - Compute proof for y = f(x)

- Verification can be much easier than computation!
  - NP-complete problems:
    - computation -> exponential time?
    - verification -> polynomial time
  - y = 1/x harder to compute than verifying x * y = 1
  - y = $\sqrt{x}$ harder to compute than verifying y * y = x
  - …

Extend MPyC

- **Details of secure computation protocols transparent in MPyC:**
  - **sophisticated operator overloading combined with asynchronous evaluation of associated protocols**
  - **we like to retain this for verifiable MPyC**

- **Secure m-party computation tolerating dishonest minority of t passively corrupt parties, $1 \le 2t+1 \le m$**
  - **Case *m* = 1 included: verifiable MPC with 1 party corresponds to ordinary ZK proofs for statements of the form y = f(x)**

Candidate ZK Proofs for MPC

- **Pinocchio-based: multiparty computation of proof**
  - **Prototype for simple arithmetic**
    - **Building on work by Meilof Veeningen (on GitHub)**
      - **Trinocchio/Geppetri protocols**
      - **pysnark**
- **Sigma-proofs can be used for simple cases:**
  - **E.g., threshold Schnorr signatures are obtained for function f(sk; m) = (c; r) where c = H(g^r/pk^c; m)**
- **Bullet-proofs: nice middle ground**

Extend MPyC

- MPyC protocols based on threshold secret sharing:
  - Shamir threshold scheme
  - PRSS (pseudorandom secret sharing)

- Need conversion between encrypted inputs/outputs and secret-shared representation
  - involves threshold (multiparty) decryption
  - prototype for ElGamal encryption

Conclusion

- **MPyC: pure Python (runs on Cpython and PyPy), small footprint (5000 lines), code on GitHub**

- **If sufficient number of parties can be trusted:**
  - **ordinary MPC for privacy and correctness,**

- **If potentially all parties are corrupt:**
  - **verifiable MPC ensures no false results are accepted**
    - **Case m=1 party corresponds to ZKP**
    - **Much harder to do then ordinary MPC**
    - **But verification is easier than computation**

H2020 EU-projects

priviledge-project.eu

soda-project.eu

This work is part of projects that have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780477 (PRIVILEDGE) and No 731583 (SODA)

MPyC: core modules