RSAConference2020

San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID: **CRYP-W11**

Consensus from Signatures of Work

Giorgos Panagiotakos (U. of Edinburgh)

Joint work with Juan Garay (Texas A&M) and Aggelos Kiayias (U. of Edinburgh & IOHK)

In a nutshell

Setting: synchronous, peer-to-peer, public-state setup

- Previous talk: Proofs-of-Work + Honest majority of comp. power + ROM => Consensus
- This talk: No ROM. Base security on weaker assumptions:

  Signatures-of-Work + Honest majority of comp. power + CRHF => Consensus

Random Oracle Methodology

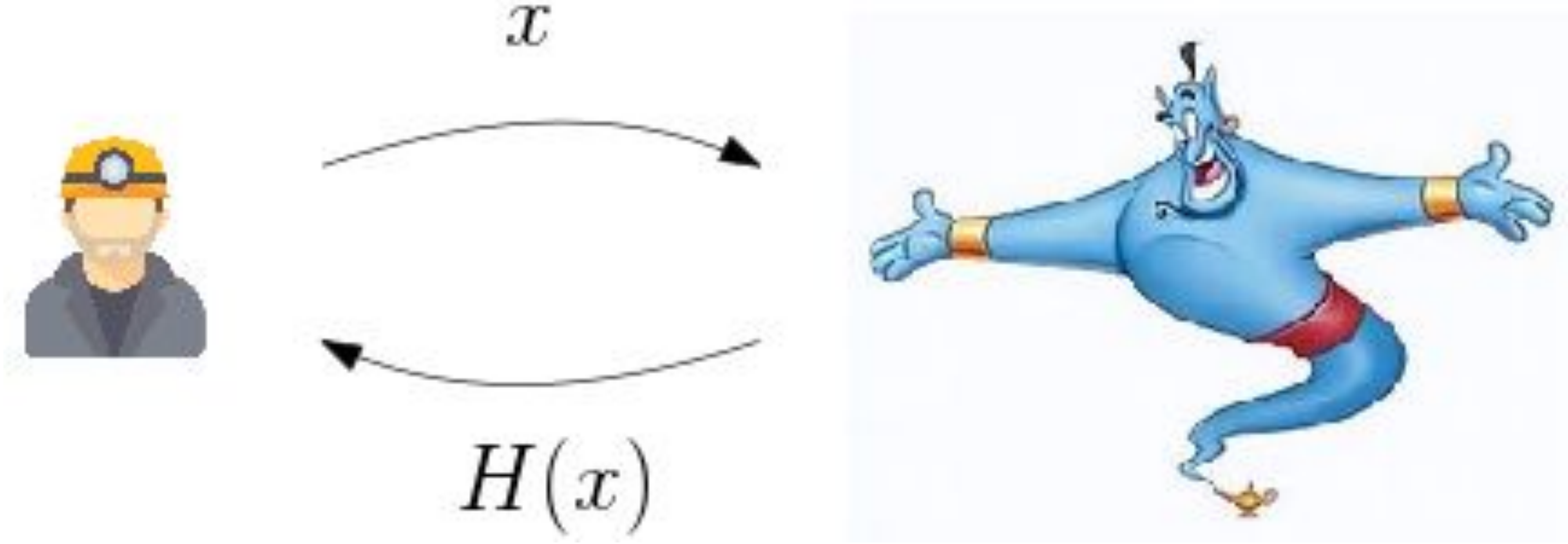
Analyze protocols that use cryptographic hashes [BR93]. E.g. Bitcoin

- PoW: Find ctr such that:

$$H(\,ctr, H(prev, trx)\,) < D$$

- $H \equiv (SHA256)^2$
- Model H as a *Random Oracle* and prove security...

Random Oracle Model



- H(x) uniform and independent, even for adversarial queries!
- Random Oracle *methodology* not sound [CGH98]

ROM in our problem

- Known results in ROM [GKL15,AD15,GKLP18]
- (Implicit) ROM-based PoW schemes too strong! e.g.,
  - Honest PoW generation algorithm optimal
  - 2-for-1 PoW [GKL15,GKLP18]

Goal: Explicitly define and base security on a *non-optimal* PoW

Our approach

1. Define suitable PoW notion
2. Non-idealized security model
3. Implement a Transaction Ledger
4. Implement Consensus

PoW

- Previous PoW definitions [DN93,Back,JJ99, SKRBN11, BGJ15,...]
  - Other applications: spam, DOS mitigation,..
  - None shown to be sufficient
- Specific PoW properties in [GKL15,AD15,GKLP18]:
  - Non-interactive
  - PoW *verifies* some data (e.g., public keys)

Similar properties found in MAC, Sig!

Signatures of Work - Concept

MAC, Sig:

- “A method of verifying that a certain party/set of parties approved some data” [Gol]
- Private knowledge allows approval

In the public-state setup setting:

- No private knowledge!
- Btc idea: approve data using comp. power
- SoW: A method of verifying that work has been spend to approve some data

SoW - Syntax

Classical Signatures

- (sk,vk) <- *KeyGen*()
- σ <- *Sign*(sk, m)
- 0 or 1 <- *Verify*(vk, m, σ)

SoW

- vk <- *KeyGen*()
- σ <- *Sign*(vk, m, h)
- 0 or 1 <- *Verify*(vk, m, σ, h)

1. No private knowledge => no secret key sk
2. Moderately hard => hardness parameter h

SoW - Security Properties

Honest signer/verifier:

- t-***Verifiable***: *Ver* takes t steps
- (t,α)-***Successful***:

$$\Pr[\ \mathit{Sign}(vk,m,h) \text{ runs for} < t \text{ steps}\ ] > \alpha$$

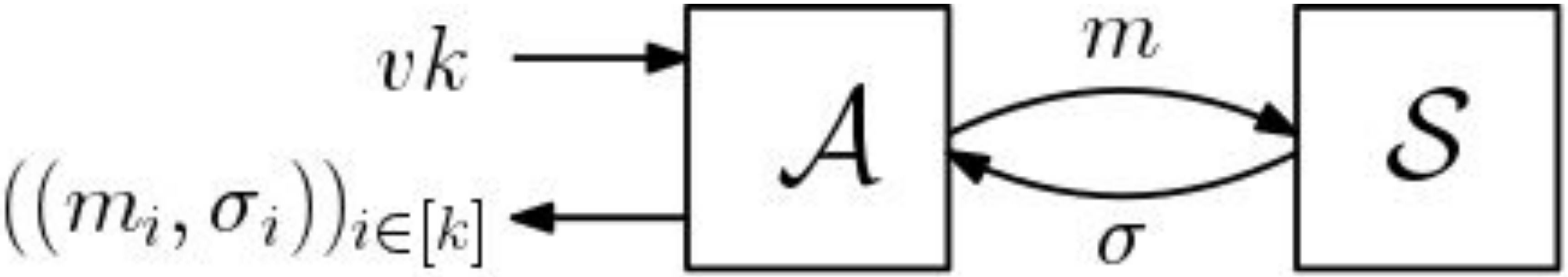
- ***Runtime independent***: runtime of *Sign*(vk,m,h) does not depend on its *inputs*

(weak randomness extractor => all of the above [GKP19])

SoW - Security Properties (II)

Malicious signer:

- (β,ε) - **Moderately Unforgeable against Tampering and Chosen Message Attacks** (MU-TCMA):



$$\forall t, \mathcal{A}_t : \Pr[\mathcal{A}_t \text{ computes } \geq \lfloor \beta(h) \cdot t \rfloor \text{ signatures}] < \epsilon(t)$$

Security Model Revisited

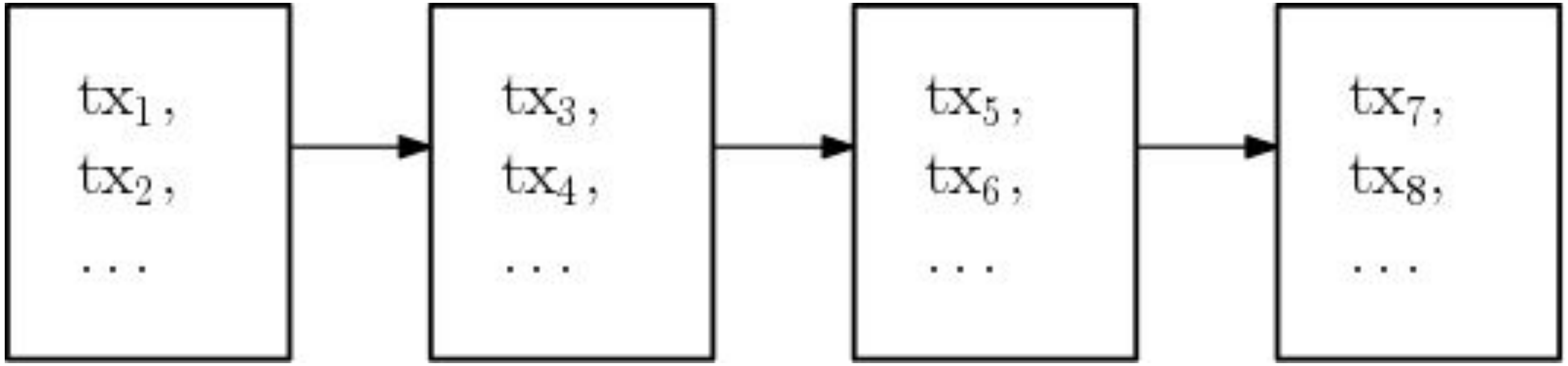
In [GKL15,AD15,GKLP18]:

- Synchronous, peer-to-peer, public state setup
- Fixed number of parties n, t corrupted
- Bounded number of RO queries per round

Instead, concrete bounds:

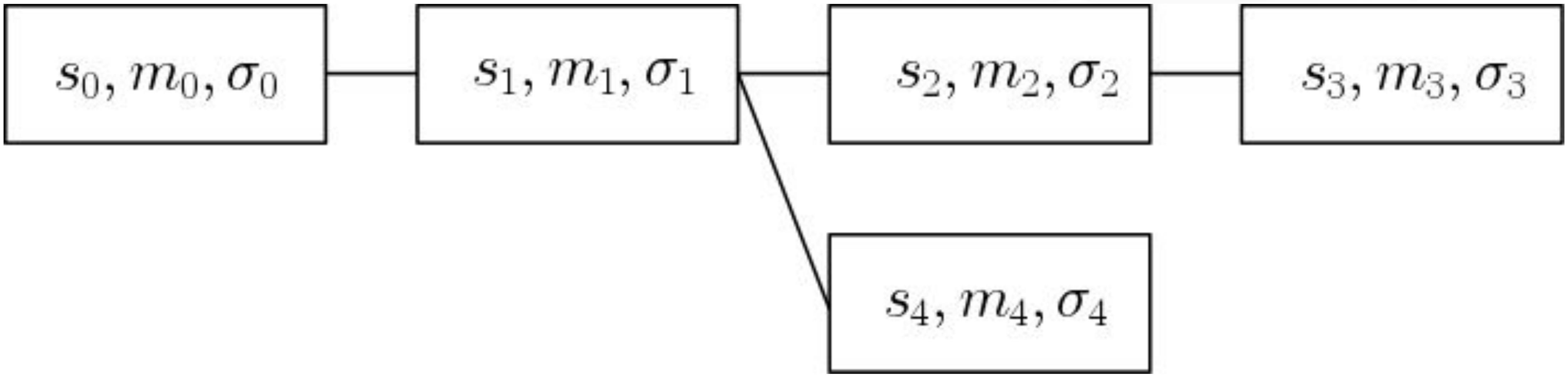
- Adversary’s steps per round < t_A
- Honest parties’ steps per round < t_H
- #messages per round < θ

Robust Public Transaction Ledger [GKL15]



- Persistence: ∃ P reports tx stable at pos i => ∀ P report tx stable and at pos i
- Liveness: non-conflicting tx provided as input long enough => ∀ P report tx stable

SoW-based Ledger



Similar to Bitcoin..

- Blocks connected using C.R. hash: $s' = G_k(s, G_k(m), \sigma)$
- Each block contains a SoW: $Ver(s, G_k(m), \sigma, h)$

Security Proof

- *MU-TCMA* => #adversarial blocks < ...
- *Runtime Independence* + *Successful*
  => #uniquely successful rounds > ...

If additionally:

- Honest majority in comp. power
- *Good* SoW scheme ($\alpha \approx \beta t_H$)
- Bounded SoW generation rate

=> Public Transaction Ledger [GKL15]

Consensus

n parties, t corrupt. Each party takes an input in {0,1}

Consensus protocol definition:

- *Agreement*: all honest parties output the same value
- *Validity*: if all honest parties take the same input b, output b
- *Termination*: all parties output some value eventually

Consensus protocol in ROM

- Consensus not immediate [selfish mining attack]
- Solution in ROM: 2-for-1 PoW [GKL15]

Block PoW: $H(\ ctr, H(\ prev, block, trx)\ ) < D$

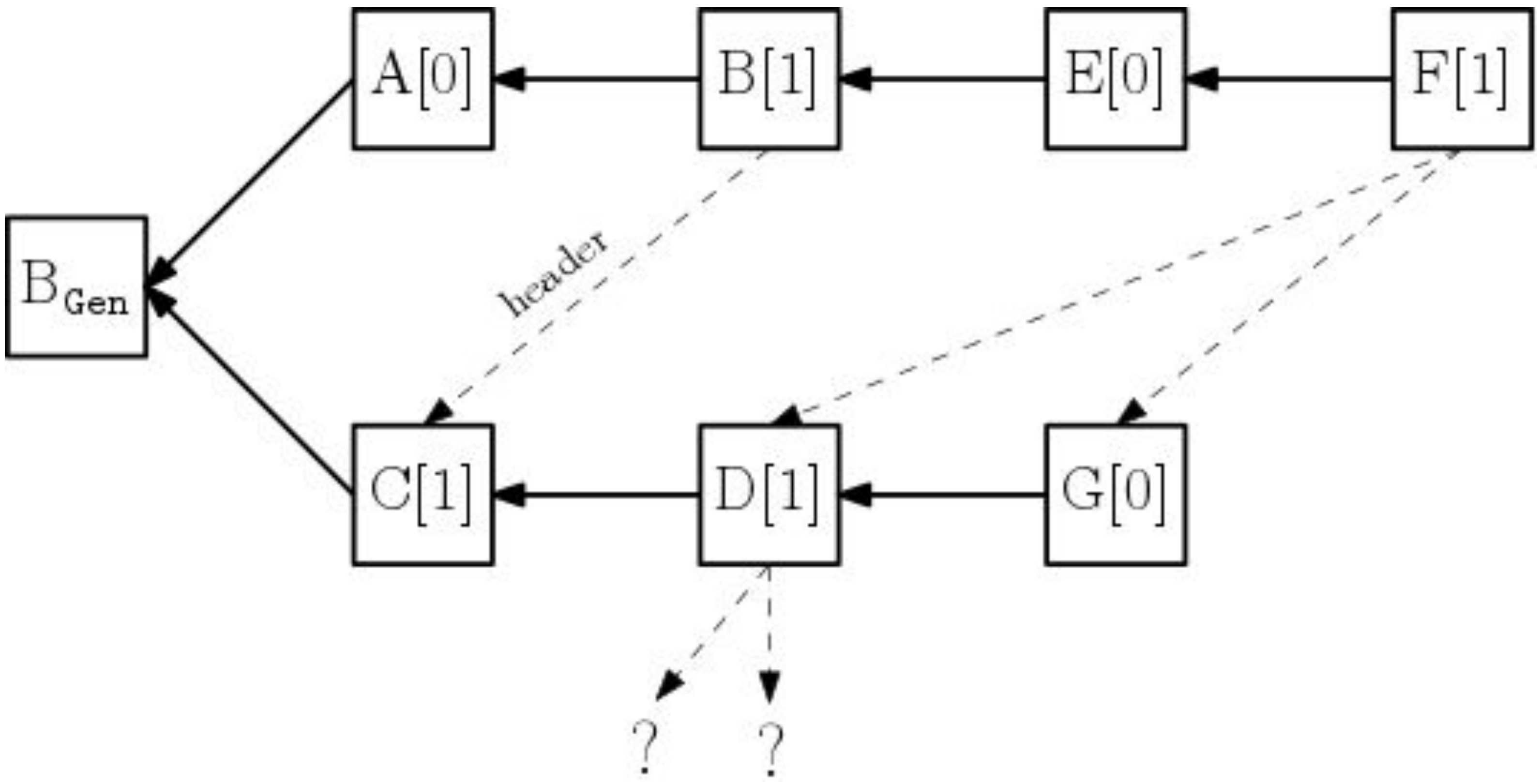
Trx PoW: $[\ H(\ ctr, H(\ prev, block, trx)\ )\ ]^R < D$

- $D < 2^{\kappa/2}$ => *independent* events!

Can we avoid the extra property?

Consensus protocol

Idea: chain agreement => block tree agreement [Inclusive,...]



Consensus protocol (II)

Protocol:

- Blockchain extension/selection as in Bitcoin
- Blocks contain off-chain *headers* and vote
- SoW can be verified from the block header

$$\mathit{Ver}(\, s_i,\, G(m_i) \,||\, \text{vote},\, \sigma_i,\, h)$$

Decision: *majority* among block header votes in chain prefix

Consensus protocol (III)

- Chain agreement => consensus agreement
- Tree agreement => consensus validity
- Simultaneous termination

Theorem

SoW + Honest majority in comp. power + CRHF => Consensus

Conclusion

We do not really need all the ROM power, only SoW + CRHF.

Open questions:

- How to implement SoW?
- Weaker security notions?

Some progress in [Iterated Search Problems and Blockchain Security under Falsifiable Assumptions, GKP19]