The Bitcoin Backbone: Analysis and Applications

Juan Garay (Yahoo Labs)
Aggelos Kiayias (U. of Athens)
Nikos Leonardos (U. Paris Diderot – Paris 7)

Decentralized Payment Systems

- Traditional *e-cash* (D. Chaum,…): centralized approach
- First decentralized "cryptocurrency" ─ Bitcoin ─ announced in 2008
- January 2009: the Bitcoin network is created. A number of other crypto currencies follow suit
- High impact; a number of other potential applications: contracts, reputation systems, name services, etc.

Bitcoin Players

Miners

- Do work to maintain the transaction ledger
- Get rewards for their work:
  i. fees
  ii. new bitcoins

Payers

- Broadcast a transaction stating they send bitcoin
- Rely on security of digital signatures to ensure money is not stolen

Payees

- Have to generate a Bitcoin address
- Have to verify their address is credited

Valid Transactions

- Transactions are organized by miners in a *transaction ledger* τ
- There is a well-defined public predicate that given a transaction ledger and a transaction decides whether the transaction "makes sense"

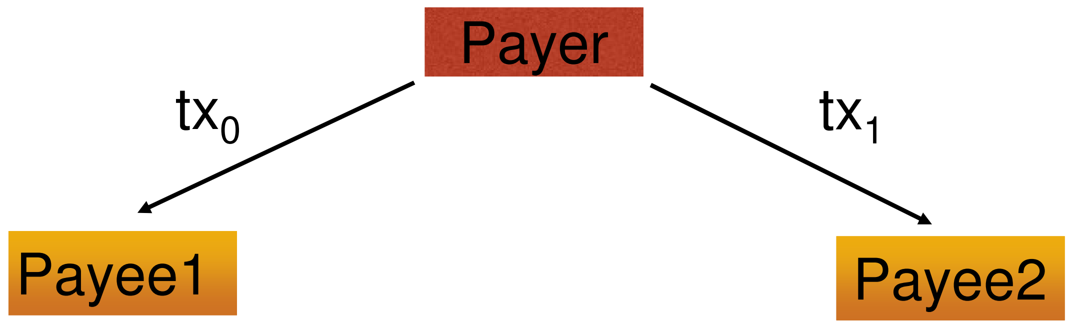
$$\text{Valid}(\tau, tx) \in \{\text{True, False}\}$$

- Each miner will accept a transaction only if it is valid given its local view of the ledger

YAHOO!

Double-spending Bitcoin

- The "litmus test" for any payment system



- Double-spending transactions are inconsistent:

$$tx_b \in T \rightarrow \mathrm{Valid}(T, tx_{1-b}) = \mathrm{False}$$

- No honest miner will accept an invalid transaction
- As long as miners agree on T no double-spending is feasible

YAHOO!

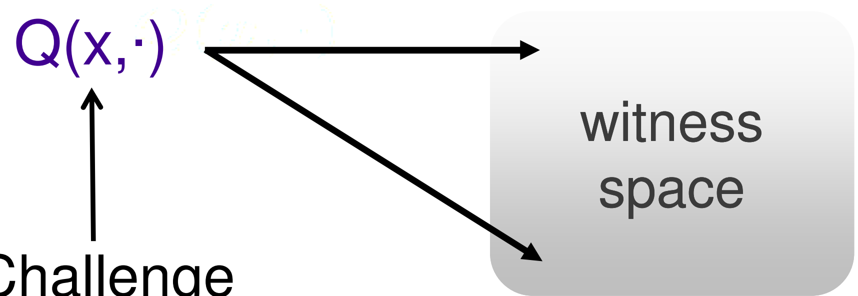
Double-spending Bitcoin (2)

- If *single* miner exists, then double-spending is infeasible — but Bitcoin would be guaranteed solely by that entity
- How to facilitate multiple miners while preventing double-spending?
- How to scale this to thousands… millions… of users at a global scale and maintain security?
  - No PKI or authenticated channels, so standard secure multi-party computation (MPC) [Yao82, GMW87] techniques cannot be used (cf. [SD14])

YAHOO!

Answer: Proofs of Work (aka “Time-Lock” Puzzles) [DN92, RSW96, Bac97, JB99]

$Q(\cdot,\cdot)$: Polynomial-time predicate



Challenge determines work level d

Search for witness takes time $\exp(d)$

(a complexity lower bound needs to be assumed)

Verification is easy!

YAHOO!

Using POWs

- Miners collect a set of transactions

$$\mathbf{tx} = (tx_1, tx_2, \ldots, tx_i)$$

- Then do "work"

$$i := 0; \text{ while } Hash(i; Hash(\tau, \mathbf{tx})) > D \text{ do } i{+}{+}$$

- If while loop terminates broadcast $(\tau, i, \mathbf{tx})$ (new "block")

YAHOO!

Using POWs (2)

- If a vector $(\tau', i', \mathbf{tx}')$ is received, check

$$(\tau = \tau') \wedge (\mathrm{Hash}(i'; \mathrm{Hash}(\tau, \mathbf{tx}')) \leq D)$$

- Expand the transaction ledger

$$\tau := \tau' \| \mathbf{tx}'$$

(called a "blockchain" — denoted $\mathcal{C}$)

YAHOO!

Longest Chain Wins

- Size *does matter* in Bitcoin:
  - If $(\tau \neq \tau')$ then miners compare their respective sizes in terms of number of blocks

- Miners' basic rule: If my chain is not smaller, I keep it; else I switch to the new one

YAHOO!

Analyzing the Bitcoin Protocol

- Nakamoto: Initial set of arguments of why Bitcoin prevents double-spending attacks
  - Wait for the transaction that gives credit to advance into the blockchain a number of k blocks, then prob. of attacker building another blockchain drops exp'ly with k
- Adversary vs. honest player working on a chain perform a random walk
- Assuming an honest majority the adversary cannot "catch" the honest players
- Nakamoto's analysis can be easily seen to be limited
  - The adversary can be more creative than just mining in private until he obtains a longer chain. E.g., it can broadcast conflicting chains to different sets of honest miners in order to split their mining power

Our Work

- Analysis of Bitcoin in a general adversarial model
- We extract, formally describe, and analyze the core of the Bitcoin protocol — the *Bitcoin backbone*
- Protocol parameterized by three application-specific external functions
  - V(·): *content (of chain) validation predicate*
  - I(·): *input contribution function*
  - R(·): *chain reading function*

YAHOO!

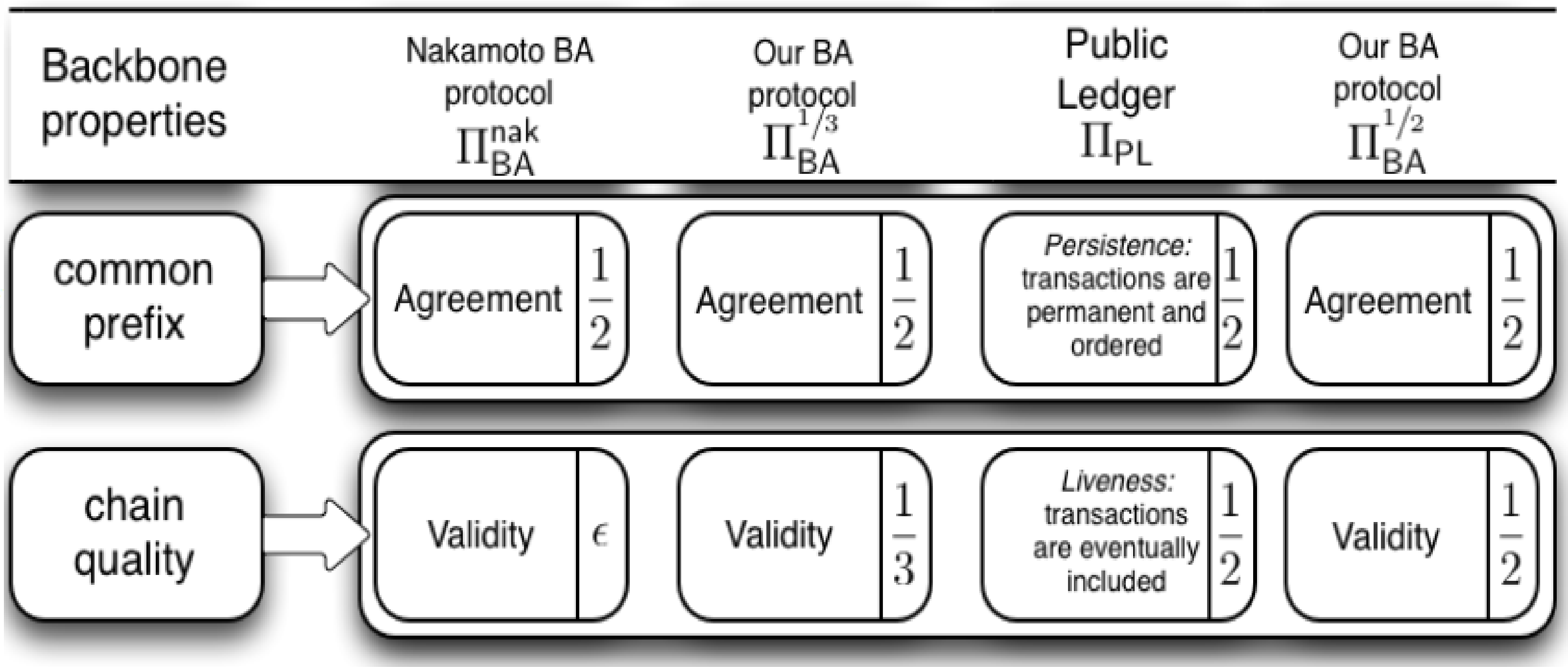
Our Work (2)

- Two fundamental properties of the Bitcoin backbone, assuming (1/2)-bounded adversary and high network synchronicity
  - *Common prefix*: After adequately "pruning" their local chains, honest parties share a common prefix
  - *Chain quality*: Guaranteed ratio of blocks contributed by the honest parties

- **Note:** Rather abstract properties of distributively maintained data structure

YAHOO!

Our Work (3)



YAHOO!

Model

- Protocol executed by fixed no. of parties n (not necessarily known to participants); (active/“rushing”/adaptive) adversary controls a subset
- Underlying graph not fully connected; messages delivered through “diffusion” mechanism (“Broadcast”)
- Parties *cannot* authenticate each other; adversary can “spoof” source of message
- Assume time is divided in rounds; within each round all messages are delivered
  - Important in terms of Bitcoin’s inherent assumption regarding the players’ ability to produce POWs

Model (2)

- "Flat model:" In a single round, all parties are allowed the same number of queries to a cryptographic hash function, modeled as a *random oracle* [BR93]
  - *"q-bounded synchronous model"*
  - $t < n$ parties controlled by adv. → $t \cdot q$ queries/round
  - $t < n/2$ corresponds to adv. controlling strictly less of the system's total "hashing power"

Model (3)

- Let

  $p = D/2^{\kappa}$: prob. of POW solution

  α: Expected POW solutions by honest parties in a round

  β: Adversary's expected POW solutions in a round

  $f = \alpha + \beta$ (Total/System's POW rate)

  $\gamma = \alpha - \alpha^2$ (Lower bound on prob. that exactly one honest party computes a POW solution in a round)

- Assume $\gamma > \lambda\beta, \ \lambda \in [1,\infty)$
  - Relation between "good" and "bad" hashing power

The Bitcoin Backbone Protocol

Algorithm 4 The Bitcoin backbone protocol, parameterized by the *input contribution function* $I(\cdot)$ and the *chain reading function* $R(\cdot)$.

$\mathcal{C} \leftarrow \varepsilon$
$st \leftarrow \varepsilon$
$round \leftarrow 0$
while TRUE **do**
  $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}, \text{any chain } \mathcal{C}' \text{ found in } \textsc{Receive}())$
  $\langle st, x \rangle \leftarrow I(st, \tilde{\mathcal{C}}, round, \textsc{Input}(), \textsc{Receive}())$ ▷ Determine the x-value to insert.
  $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(x, \tilde{\mathcal{C}})$
  if $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ **then**
    $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
    $\textsc{Broadcast}(\mathcal{C})$
  end if
  $round \leftarrow round + 1$
  if $\textsc{Input}()$ contains $\textsc{Read}$ **then**
    write $R(\mathrm{x}_{\mathcal{C}})$ to $\textsc{Output}()$
  end if
end while

Our Work (2)

RECALL

- Two fundamental properties of the Bitcoin backbone, assuming (1/2)-bounded adversary and high network synchronicity
  - *Common prefix*: After adequately "pruning" their local chains, honest parties share a common prefix
  - *Chain quality*: Guaranteed ratio of blocks contributed by the honest parties

- **Note:** Rather abstract properties of distributively maintained data structure

YAHOO!

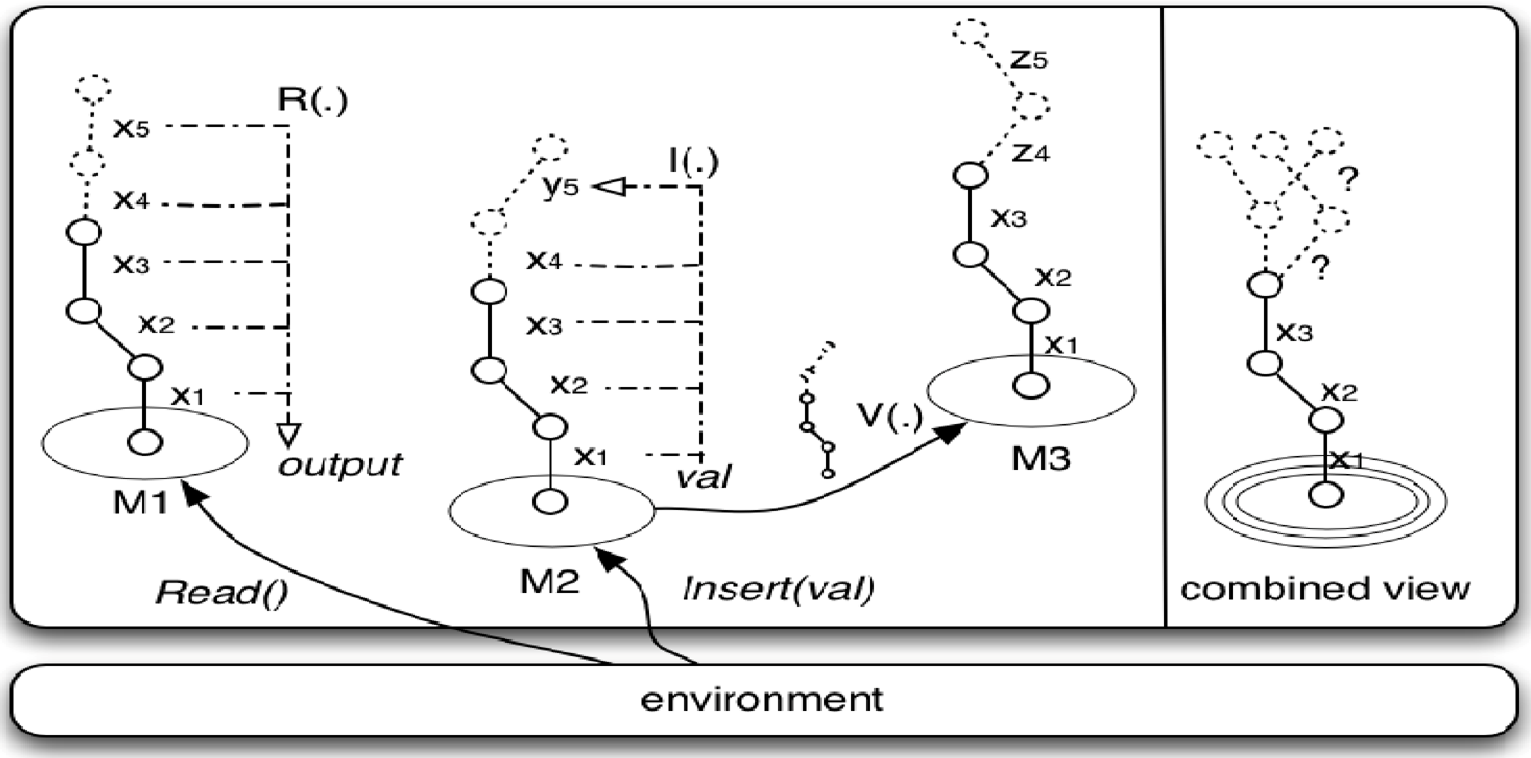
Common Prefix Property

- **Definition** (Common prefix, w/ param. k). For any pair of honest parties P_1, P_2

$$C_{1,[k} \leq C_2 \text{ and } C_{2,[k} \leq C_1$$

YAHOO!

Common Prefix Property (2)



YAHOO!

Common Prefix Property (3)

- **Definition.** (Common prefix, w/ param. k) For any pair of honest parties P_1, P_2

$$\mathcal{C}_{1,\lceil k} \preceq \mathcal{C}_2 \text{ and } \mathcal{C}_{2,\lceil k} \preceq \mathcal{C}_1$$

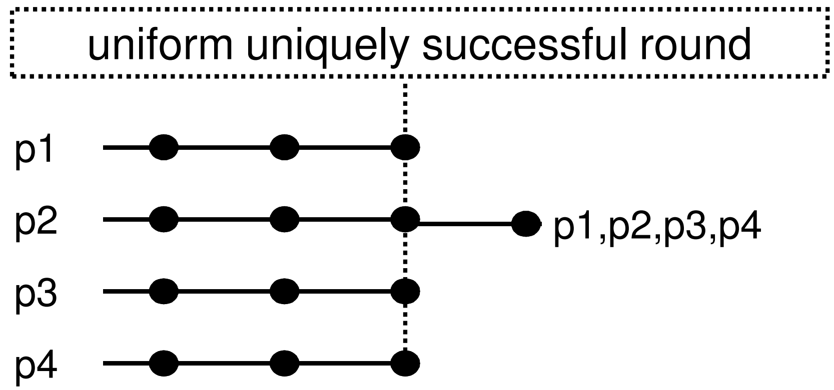
- **Theorem** (Common prefix, w/ param. k). Let $\lambda^2 - f\lambda - 1 \geq 0$. No matter the adversary's strategy, the chains of two honest parties satisfy the common-prefix property with probability

$$1 - e^{-\Omega(k)}$$

YAHOO!

Common Prefix Property (4)

- **Common-prefix theorem:** (proof idea)
  - *Uniform round:* Round where all honest parties invoke a POW with a chain of the same length
  - *Uniquely successful round:* Round when exactly one honest party is successful



YAHOO!

Common Prefix Property (5)

- **Common-prefix theorem:** (proof idea, cont'd)
  - *Uniform uniquely successful rounds* allow parties to reach a "convergence block"
  - To maintain a "fork," adv. must produce a POW for each convergence block
  - The rate of u.u.s. rounds is $(1 - \beta)\lambda$

In order for the adversary to maintain a fork for a certain length

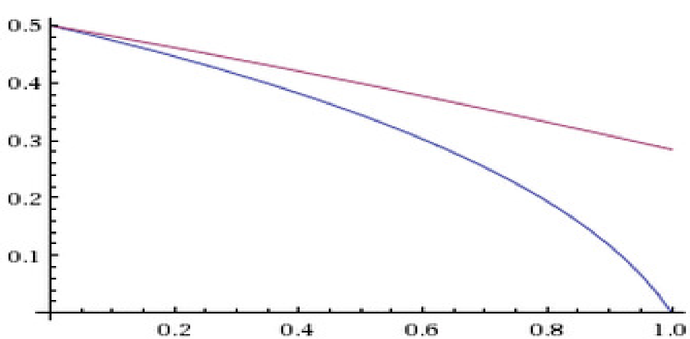
$$\beta > (1 - \beta)\lambda$$

This is equivalent to $\lambda^2 - f\lambda - 1 < 0 \rightarrow$ **...** (Chernoff bounds)

YAHOO!

Common Prefix Property (6)

- Only if f → 0 we can let λ → 1 (adversarial tolerance up to 50%) (fast information propagation)
- As f → 1 we have λ → (1 + √5)/2 (Golden Ratio)



Adversarial bound (y-axis) wrt network synchronization *f* (x-axis) so that common prefix is ensured in Bitcoin (blue) vs. Bitcoin with lexicographic tie-breaking (red)

YAHOO!

Chain Quality Property

- **Theorem** (Chain quality). Any sequence of ℓ blocks in an honest party's chain will contain $1-1/\lambda$ proportion of honest blocks with probability

$$1 - e^{-\Omega(\ell)}$$

- The theorem is tight
  - There is an adversarial strategy that restricts the honest parties to a ratio of exactly $1-1/\lambda$
  - The strategy is a type of *selfish mining* [ES14]: Malicious miners mine blocks in private attempting to "kill" honest parties' blocks when they become available

Chain Quality Property (2)

- *Ideal* chain quality: A set of parties with hashing power α may control up to αL blocks in a blockchain of length L

- Our chain quality bound is much more pessimistic (as the adv. can control almost all the blocks)

- Selfish mining implies that this is tight… Bitcoin is not incentive-compatible

YAHOO!

Applications of the Bitcoin Backbone Protocol

Applications of the Backbone: Byz. Agreement [PSL80, LSP82]

- Byzantine agreement (BA): *n* parties start with an initial value v_i
  - Agreement: All honest parties output the same value
  - Validity: If all honest parties start with the same input (say, v), then they output this value

- BA in the *anonymous synchronous setting*
  - "Anonymous model without port awareness" [Okun05]
  - Deterministic BA not possible
  - *POW-based* protocols (cf. [AJK05, KMS14])

YAHOO!

Nakamoto's BA Protocol

- The *n* parties start building a blockchain inserting their input
- If a party receives a longer blockchain switches to that one and switches its input
- When the blockchain is long enough the party outputs the value that it contains

- **Intuition:** Agreement would follow from the fact that honest parties will eventually agree on a single chain (for (1/2)-bounded adv.)
- **Issue:** If adv. finds a solution first, then honest parties will extend adv.'s solution and switch to adv.'s input

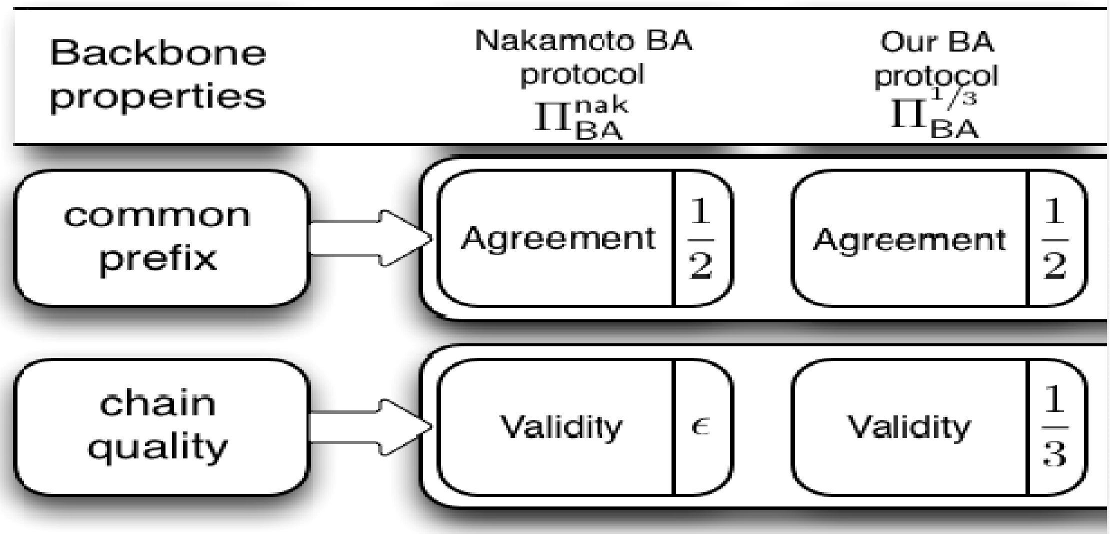
Our First BA Protocol

- The *n* parties start building a blockchain inserting their inputs
- If a party receives a longer blockchain switches to that one but *keeps the same input*
- Once the blockchain is long enough the parties prune the last k blocks and output the *majority value* in the prefix

- Protocol tolerates (1/3)-bounded adversaries

YAHOO!

Summary of Results (1)



YAHOO!

Applications of the Backbone (2)

- **Robust transaction ledgers:** n unauthenticated parties accept transactions and build a *ledger* so that the following properties are satisfied:

  (i) Persistence: If a transaction is "deep" enough in the ledger for one honest party, then it will be reported by all honest parties at the same location

  (ii) Liveness: All honestly generated transactions eventually get deep enough in the ledger of an honest party

- We show how to instantiate the public transaction ledger for Bitcoin, by defining the sets of transactions and valid ledgers (see paper)

YAHOO!

Applications of the Backbone (3)

- **Our second BA protocol**
  - The *n* parties build a ledger but now *generate transactions based on POWs that contain their inputs* – input itself must satisfy POW pred.
  - Once the blockchain is long enough the parties prune the last k blocks and output the majority of the values drawn from the unique transactions
  - Protocol tolerates (1/2)-bounded adversaries
  - POWs are *now used for two different tasks*

How do we prevent the adversary from shifting its hashing power from one to the other?

YAHOO!

2-for-1 POWs

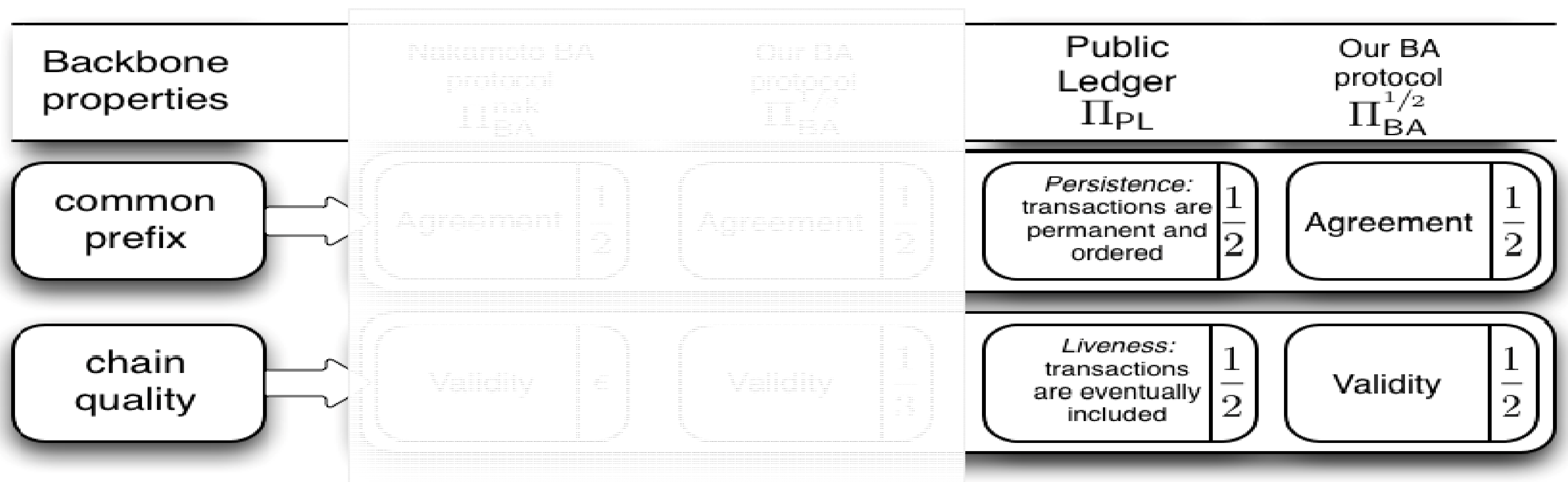
Algorithm 6 POW-based protocol fragment of Π_b, $b \in \{0,1\}$ parameterized by q, D and hash functions $H_b(\cdot), G(\cdot)$, $b \in \{0,1\}$. The value w_b is determined from the protocol's context.

```
1:  ...                          ▷ Value w_b is determined
2:  ctr ← 1
3:  B ← ε
4:  h_b ← G(w_b)
5:  while (ctr ≤ q) do
6:      if (H(ctr, h_b) < D) then
7:          B_b ← ⟨ctr, w_b⟩
8:          break
9:      end if
10:     ctr ← ctr + 1
11: end while
12: ...                          ▷ The POW B is exploited here
```

Algorithm 7 The *double proof of work* function, parameterized by q, D and hash functions $H(\cdot), G(\cdot)$ that substitutes steps 2-11 of two POW-based protocols.

```
1:  function double-pow(w_0, w_1)
2:      B_0, B_1 ← ε
3:      ctr ← 1
4:      while (ctr ≤ q) do
5:          h ← H(ctr, G(w_0), G(w_1))
6:          if (h < D) then
7:              B_0 ← ⟨ctr, w_0, G(w_1)⟩
8:              break
9:          end if
10:         if ([h]^R < D) then
11:             B_1 ← ⟨ctr, w_1, G(w_0)⟩
12:             break
13:         end if
14:         ctr ← ctr + 1
15:     end while
16:     return ⟨B_0, B_1⟩
17: end function
```

Summary of Results (2)



YAHOO!

Conclusions

- Formal treatment of core of Bitcoin's transaction ledger ─ the Bitcoin "backbone"
  - "Common prefix" and "chain quality" as foundations for BA and robust transaction ledger protocols

- Deviations of concern
  - Network synchronization *vis-à-vis* POW rate: fast information propagation is essential
  - Adv.'s contributions to blockchain can be strictly larger than β: transaction liveness becomes fragile as $\beta \rightarrow 1/2$

- Fixed no. of participants
  - Difficulty D ("target T") may be calibrated according to the no. of active players

YAHOO!

References

- J. Garay, A. Kiayias and N. Leonardos, “The Bitcoin Backbone Protocol: Analysis and Applications.” Cryptology ePrint Archive: Report 2014/765

  http://eprint.iacr.org/2014/765

- T. Holenstein, “Is There a Theory Behind Bitcoin?” ITS Science Colloquium 06.11.2014

  http://www.eth-its.ethz.ch/activities/its-science-colloquium/Holenstein.html

YAHOO!

YAHOO! LABS

Science-Driven Innovation