Lecture 12

Algorand
Proof-of-Stake
“Virtual Mining”
Bitcoin uses proof of work to address sybil attacks and implement consensus

- Philosophy: Chance of “winning” in a block mining round proportional to your (hash) computing power

Proof of Stake: Addresses sybil attacks by requiring that participants must have some “stake” (i.e., money) in the system

- Philosophy: Chance of winning in a round proportional to your current stake
(Potential) Advantages

- In Proof of Stake based cryptocurrency, users (who have money in the system) are the miners
- Environment friendly
- No ASIC advantage
- 51% (or higher) majority assumption potentially more realistic
51% attack prevention argument
The Bitcoin economy is smaller than the world
Wealth outside Bitcoin has to move inside
51% attack prevention argument

The Bitcoin economy is smaller than the world

Wealth *outside* Bitcoin has to move *inside*
51% attack prevention argument
The Bitcoin economy is smaller than the world Wealth *outside* Bitcoin has to move *inside*
Examples of PoS based Cryptocurrencies

- Peercoin
- Blackcoin
- Nxt
- Neucoin
- ...
PoS systems with security analysis

- Algorand  [Gilad-Hemo-Micali-Vlachos-Zeldovich’17]
- Ourboros  [Kiayias-Russel-David-Oliynykov’17]
- Snow white  [Daian-Pass-Shi’17]
- ...

Algorand: Main Highlights

- Proof of Stake based Cryptocurrency
- High throughput: ~1 min to confirm transactions vs an hour in Bitcoin
- Public ledger with low probability of forks
- Assumes 2/3-honest stake majority
- Uses a gossip communication protocol
Algorand: Main Highlights

- Adaptive adversary: May corrupt dynamically, as long as 2/3 majority assumption holds
- Achieves Consistency assuming “weak synchrony”
  - Network can be asynchronous for long bounded time period $b$, but then must have strong synchrony for short period $s < b$
- Achieves Liveness assuming “strong synchrony”
  - Most honest users (e.g., 95%) can send messages that will reach within a known time bound
Main Design Ingredients

- Users weighted by stake (to prevent sybil attacks)
- Builds on byzantine agreement (BA) protocol of Micali [ITCS’17] for consensus
- BA protocol executed between a small committee of users for scalability
- Committee chosen at random, using cryptographic techniques
Algorand Consensus: Main Highlights

- BA protocol in expectation terminates in only 4 steps (in “honest” case) or 13 steps (in “dishonest” case)
- Player replaceability: Players across different steps of BA protocol may not be the same
  - Possible because protocol does not require “private state”
- For each step, players chosen at random, non-interactively, in a “publicly verifiable” manner
Fast and Furious Byzantine Agreement

Micali [ITCS’17]
Byzantine Agreement

A protocol $P$ is an $(n, t)$ byzantine agreement protocol with soundness $s$ if:
- for every value set $V$ and adversary $A$ who corrupts $t$ out of $n$ players,
- in an execution of $P$ with $A$ in which each player starts with value $v_i$ in set $V$, each honest player halts with prob 1, outputting a value $out_i$, so as to satisfy, with prob $s$, the following properties:

  - **Agreement**: $out_i = out_j$ for all honest players $i$ and $j$
  - **Consistency**: if for some $v$, $v_i = v$ for all honest players $i$, then $out_j = v$ for all honest players $j$
Binary BA vs Arbitrary value BA

- Binary BA: Input value set $V$ is $\{0,1\}$
- [Turpin-Coan’84]: general reduction to convert binary BA into arbitrary value BA
  - assuming 2/3 honest majority
  - requires only two additional rounds of communication
- This talk: Focus on Binary BA
Why is Byzantine Agreement Hard?

- Protocol executed over point-to-point channels
- Adversarial parties may send different messages (including no message) to different honest parties
- BA over broadcast channels is trivial
Micali’s Protocol: Main Intuition

Consider idealized Protocol P(r), where b_i is the initial input of party i:

- Each player i sends b_i to all other players
- A new random and independently selected bit c(r) appears in sky
- Player i updates bit b_i as follows:
  - If \#_{i,r}(0) \geq 2t+1, set b_i = 0
  - If \#_{i,r}(1) \geq 2t+1, set b_i = 1
  - Else, set b_i = c(r)

\#_{i,r}(b): Number of players from which i received b in “iteration” r
Quick Analysis

Assuming at least $2t+1$ players are honest, $P(r)$ achieves soundness $1/2$

- If honest players start in agreement, then they remain in agreement
- If honest players do not start in agreement, then they end in agreement (on some bit) with probability $1/2$
Implementing coin in sky using Crypto

Three Ingredients:

- **Unique Digital Signatures**: For every public key $pk$ and message $m$, only one valid signature for $m$ w.r.t. $pk$
  - Can be constructed from standard cryptographic assumptions
- **Hash function**: Modeled as a random oracle
- **Common random string $R$**: fixed at the start of the protocol execution, known to each party, and not controlled by adversary
Implementing coin in sky using crypto

ConcreteCoin(r): Each player i does the following,

- Send $v_i = \text{SIG}_i(R, r)$
- Compute $m$ s.t. $H(v_m) \leq H(v_i)$ for all $i$
- Set $c_i(r) = \text{lsb}(h)$, where $h = H(v_m)$

Think: What is the probability that $c_i(r) = c_j(r)$ for all honest $i, j$?

Think: Why is $c_i(r)$ random?
Using ConcreteCoin(r)

Replacing coin in sky with ConcreteCoin(r) in P(r):

- If honest players start in agreement, then they remain in agreement
- If honest players do not start in agreement, then they end in agreement (on some bit) with probability 1/3
- Overall, the resulting protocol has soundness 1/3
Remaining Problem

Can we simply repeat the protocol indefinitely until agreement is reached?

- The honest players do not know that agreement is reached
- Thus, the protocol may never terminate
Remaining Problem

Can we simply repeat the protocol indefinitely until agreement is reached?

- The honest players do not know that agreement is reached
- Thus, the protocol may never terminate

**Idea:** Simply repeat say \( k = 200 \) times to ensure that agreement is reached, except with very small probability

**Drawback:** We waste many rounds since most times, agreement will be reached earlier
Micali’s Idea:

Protocol BBA*: It consists of sequential repetitions of $P'(r)$, where each $P'(r)$ consists of three correlated executions of $P(r)$

1. Execution of $P(r)$ where $c(r) = 0$
2. Execution of $P(r)$ where $c(r) = 1$
3. Execution of $P(r)$ where $c(r)$ is implemented via ConcreteCoin$(r)$

Note 1: In the first two executions, a party will terminate if it thinks agreement is reached

Note 2: While the number of repetitions of $P'(r)$ are not fixed in advanced, the expected number of repetitions will be 3 (will follow from protocol analysis)
Notation:

1. A party $i$ may at any point send special value $b^*$ (and HALT) meaning that in all future steps, other parties should think of $i$’s message as $b$.
2. Counter $\gamma$ which indicates how many times the 3-step loop has been executed. Initially set to 0.
3. $R$ denotes the common random string.
Protocol $BBA^*$

(Communication) Step 1. [Coin-Fixed-To-0 Step] Each player $i$ propagates $b_i$.

1.1 If $\#^1_i(0) \geq 2t + 1$, then $i$ sets $b_i = 0$, sends $0^*$, outputs $out_i = 0$, and HALTS.
1.2 If $\#^1_i(1) \geq 2t + 1$, then, $i$ sets $b_i = 1$.
1.3 Else, $i$ sets $b_i = 0$.

(Communication) Step 2. [Coin-Fixed-To-1 Step] Each player $i$ propagates $b_i$.

2.1 If $\#^2_i(1) \geq 2t + 1$, then $i$ sets $b_i = 1$, sends $1^*$, outputs $out_i = 1$, and HALTS.
2.2 If $\#^2_i(0) \geq 2t + 1$, then $i$ set $b_i = 0$.
2.3 Else, $i$ sets $b_i = 1$.

(Communication) Step 3. [Coin-Genuinely-Flipped Step] Each player $i$ propagates $b_i$ and $SIG_i(R, \gamma)$.

3.1 If $\#^3_i(0) \geq 2t + 1$, then $i$ sets $b_i = 0$.
3.2 Else, if $\#^3_i(1) \geq 2t + 1$, then $i$ sets $b_i = 1$.
3.3 Else, letting $S_i = \{ j \in N : m^3_i(j) = SIG_j(R, \gamma) \}$,
   $i$ sets $b_i = c^{(\gamma)}_i \triangleq \text{lsb}(\min_{j \in S_i} H(SIG_i(R, \gamma)))$; increases $\gamma_i$ by 1; and returns to Step 1.
Analysis

Claim 1: If at start of an execution of step 3, no player has halted and agreement has not been reached, then with prob 1/3, players will be in agreement at the end of the step.
**Analysis**

**Claim A:** If at start of an execution of step 3, no player has halted and agreement has not been reached, then with prob $1/3$, players will be in agreement at the end of the step.

**Proof:** Consider 5 cases:

1. All honest $i$ update $b_i$ according to 3.1
2. All honest $i$ update $b_i$ according to 3.2
3. All honest $i$ update $b_i$ according to 3.3
4. Some honest $i$ update $b_i$ according to 3.1, others according to 3.3
5. Some honest $i$ update $b_i$ according to 3.2, others according to 3.3
Proof: Consider 5 cases:

1. All honest $i$ update $b_i$ according to 3.1
   - Agreement holds on 0
2. All honest $i$ update $b_i$ according to 3.2
   - Agreement holds on 1
3. All honest $i$ update $b_i$ according to 3.3
   - Agreement holds on c
4. Some honest $i$ update $b_i$ according to 3.1, others according to 3.3
   - Agreement on 0 reached with prob $1/2$ (assuming $c_i$’s are same)
5. Some honest $i$ update $b_i$ according to 3.2, others according to 3.3
   - Agreement on 1 reached with prob $1/2$ (assuming $c_i$’s are same)

Overall, when $m$ is honest, agreement is reached with probability at least $1/2$. $m$ is honest with prob $2/3$ (which means $c_i$’s are same), so overall agreement prob is $1/3$
Claim B: If at some step, agreement holds on bit b, then it continues to hold on bit b
Analysis (contd.)

Claim B: If at some step, agreement holds on bit $b$, then it continues to hold on bit $b$

Proof: If agreement held at the start of step, then all honest parties send bit $b$, which means $\#_i(b) \geq 2t+1$
Analysis (contd.)

**Claim B**: If at some step, agreement holds on bit b, then it continues to hold on bit b

**Proof**: If agreement held at the start of step, then all honest parties send bit b, which means $#(b) \geq 2t+1$

**Claim C**: If at some step, a player i halts, then agreement will hold at the end of the step
Analysis (contd.)

Claim B: If at some step, agreement holds on bit b, then it continues to hold on bit b

Proof: If agreement held at the start of step, then all honest parties send bit b, which means $\#_i(b) \geq 2t+1$

Claim C: If at some step, a player i halts, then agreement will hold at the end of the step

Proof: WLOG, suppose i halts in Coin-Fixed-To-0 step. Since $\#_i(0) \geq 2t+1$, at least $t+1$ honest players sent 0. Thus, $\#_j(0) \geq t+1$ for every other honest j. If $\#_j(0) \geq 2t+1$, then j sets $b_j=0$ in step 1.1, else it sets $b_j=0$ in step 1.3. (Main point: step 1.2 cannot be executed)
Analysis (contd.)

**Property 1**: Consistency (if initial bit $b$ for all honest players, then $\text{out}_i = b$)

**Proof**: Easily follows from Substep 1.1 or 2.2 (depending upon whether starting input was 0 or 1)

**Property 2**: Agreement ($\text{out}_i = \text{out}_j$ for all honest $i, j$)

**Proof**: Follows from Claims A, B and C
Algorand using Byzantine Agreement
Main idea

- Users weighted by stake (to prevent sybil attacks)
- For every block generation round, a small committee of users is chosen at random, using crypto, based on user weights
- One of the committee members who has the highest “priority” proposes a block
- The committee then runs a BA protocol to reach consensus on the proposed block
Verifiable Random Functions

- On any input $x$, $VRF_{sk}(x)$ outputs $(\text{hash}, \text{proof})$
- hash is uniquely determined given $sk$ and $x$ but indistinguishable from random to anyone who does not know $sk$
- Given $pk$ and proof, anyone can check that hash corresponds to $x$
Notation

- $W$: total amount of currency units
- $t$: threshold, denoting expected number of users selected
- $p$: $t/W$
- $w_i$: stake/money of user $i$
- $B(k;w,p)$: Prob of getting $k$ successes in $w$ trials, where prob of success in each trial is $p$ (Binomial distribution)
  \[ \sum_{k=0}^{w} B(k; w, p) = 1 \]
- Division of interval $[0,1)$ into multiple consecutive internals
  \[ I_j = \left[ \sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p) \right) \]
Cryptographic Sortition

Sortition(sk,seed,p,w):

- $\operatorname{VRFs}_k(\text{seed}) \rightarrow (\text{hash}, \text{proof})$
- $j \rightarrow 0$
- While $\frac{\text{hash}}{2^{\text{hashlen}}} \notin \left[ \sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p) \right]$, $j++$
- Return (hash,proof,j)
Cryptographic Sortition

- Any user can check on its own whether it was selected by using its sk, and then send a proof to others for the same
- User’s whose hash is highest is the “block proposer”
- Users then run BA to reach consensus on proposed block
Consensus

- For each step of the consensus protocol, a different set of users is chosen (using cryptographic sortition algorithm)
- All users can passively participate in the protocol (by listening to the gossip network), and whenever selected for a step, they send a message based on what they heard so far on the network
- BA protocol has player replaceability; therefore using different users in each step is possible
Security Challenges

- For BA to have any security, a high majority of players must be honest.
- Why can’t adversary simply corrupt all the committee members?
- Main Idea: Committee members for any step are disclosed only when they send their respective messages. If adversary corrupts now, it's too late. The messages are already sent.
Security Challenges (contd.)

● How to select the threshold $t$?

● Use a threshold such that:
  ● $\#\text{good} > \text{threshold}$: for agreement
  ● $\frac{1}{2} \#\text{good} + \#\text{bad} \leq \text{threshold}$: to avoid forks
Other Points:

- The seed (used in sortition) has to be chosen carefully. Initially, it is set to be a common random string; later, for each round $r$, seed is determined from seed for round $r-1$ by using VRFs$_k$ of the block proposer in round $r-1$.

- What are the chances of forks? - Forks can happen with some probability (if network has weak synchrony), but a recovery process can be used to eliminate fork assuming there is a strong synchrony period, using same BA procedure.
Research Challenges

- Can we reduce the 2/3-honest majority assumption to 1/2?

- Consistency in Algorand requires strong synchrony periods interspersed between weak synchrony periods. Can this be relaxed?

- Liveness in Algorand requires strong synchrony. Can this be relaxed?