FruitChains: A Fair Blockchain

Rafael Pass[1] Elaine Shi[2]

[1]Department of Computer Science
Cornell Tech

[2]Department of Computer Science
Cornell University

Speaker: 李寰

Selfish Mining

Fairness

Players controlling a ϕ fraction of computational power reap a fraction ϕ of the blocks.

Block withholding attack on Nakamoto's protocol

- An adversary A controls the delivery of messages of the network
- A delays some messages from honest player, so that it can use blocks mined by A to replace blocks mined by honest players.

Chain quality of Nakamoto's protocol

Garay et al [GKL15] (in synchronous networks) and Pass et al [PSS16] (also in networks with adversarial bounded delays) show that Nakamoto's protocol achieves chain quality close to

$$1 - \frac{\rho}{1-\rho}$$

when the mining hardness parameter is appropiately set, and thus the above-mentioned block withholding attack is optimal.

Blockchain Protocols $(\Pi, \mathrm{extract})$

Algorithm Π

- Receives messages and mines blocks
- Maintains a local chain

Algorithm $\mathrm{extract}(chain)$

- Outputs an *ordered* sequence of records, or batches, $\vec{m}$

Environment Z

- Sends a message (a record) m to each honest player in each round r
- *Corrupt* or *uncorrupt* some players at any point

Attacker A

- Delivers all messages sent by players
- Cannot modify the content of messages broadcast by honest players
- But may *delay or reorder the delivery of a message* as long as it eventually delivers all messages.

The Random Oracle in Blockchain Protocols

The ROM

A random function $H : \{0,1\}^* \to \{0,1\}^*$ with two oracles:

- $\mathrm{H}(x) = H(x)$,
- $\mathrm{H.ver}(x, y) = \begin{cases} 1, & H(x) = y \\ 0, & \text{otherwise} \end{cases}$.

Access to H

In any round r,

- an honest player can make only a *single* query to H, and
- an adversary A controlling q parties, can make q *sequential* queries to H.

Access to $\mathrm{H.ver}$

In any round r, honest players (as well as A) may make *any* number of queries to $\mathrm{H.ver}$.

Nakamoto's Protocol $(\Pi_{Nak}, \text{extract}_{Nak})$

Notation

A block (h_{-1}, η, m, h)

- h_{-1}: a pointer to the previous record
- η: a nonce
- m: a message (a transaction)
- h: a pointer to the current block

Hardness parameter p

- $D_p = p \cdot 2^{\kappa}$
- $\forall (h, b), \Pr[H(h, \eta, b) < D_p] = p$

Nakamoto's Protocol $(\Pi_{Nak}, \text{extract}_{Nak})$

Notation

Validness of block $b = (h_{-1}, \eta, m, h)$ w.r.t. block $b_{-1} = (h'_{-1}, \eta', m', h')$

- $h_{-1} = h'$
- $\text{H.ver}\big((h_{-1}, \eta, m), h\big) = 1$
- $h < D_p = p \cdot 2^{\kappa}$

Validness of blockchain $(b_0, \ldots, b_l)$

- $b_0 = \big(0, 0, \perp, H(0, 0, \perp)\big)$ is the genesis block
- $\forall 1 \leq i \leq l$, b_i is valid w.r.t. b_{i-1}

Nakamoto's Protocol $(\Pi_{Nak}, \text{extract}_{Nak})$

Execution

Π_{Nak}

- Read all incoming messages (delivered by A). If any incoming message $chain'$ is a valid sequence of blocks that is longer than its local state $chain$, replace $chain$ by $chain'$. (Note that checking the validity of $chain'$ can be done using only $\mathsf{H.ver}$ queries)
- Read local message m (from Z). Pick a random nonce $n \in \{0,1\}^\kappa$ and issue query $h = \mathsf{H}(h_{-1}, \eta, \mathsf{m})$ where h_{-1} is the 4'th element in the last block in $chain$. If $h < D_p$, then Π adds the *newly mined* block (h_{-1}, η, b, h) to $chain$ and broadcasts the updated $chain$.

extract_{Nak}

On input a valid chain $chain$, output the sequence of messages m contained in blocks in $chain$.

The FruitChain Protocol $(\Pi_{fruit}, \mathrm{extract}_{fruit})$

Overview

Based on Nakamoto's blockchain protocol

- Records are stored in fruits instead of blocks
- (Recency of fruits) A Fruit is required to hang from a block not too far from the block recording it

Fruit mining

- Fruits themselves requires solving some proof of work, with a *different hardness parameter* p_f
- In each round, honest players simultaneously mine for a fruit and a block by making one invokation of the hash function
- 2-for-1 trick: the prefix and suffix of H's output for block and fruit mining.

In execution

- Whenever a player mines a fruit, it broadcasts it to all other players
- Fruits that have not yet been recorded in the blockchain (and that are still recent) are stored in a buffer and all honest players next attempt to add them to the blockchain

The FruitChain Protocol $(\Pi_{fruit}, \text{extract}_{fruit})$

Notation

Random oracle H and hash function d

- We assume that the random oracle H outputs strings of length at least 2κ. Let d be a collision-resistant hash-function (technically, it is a family of functions, and the instance from the family is selected as a public-parameter; in the sequel we ignore this selection and simply treat it as a single function (for instance, selected using randomness $H(0)$.)

Hardness parameters p, p_f and recency parameter R

- Our protocol is parametrized by two "hardness" parameters $p = p(\kappa), p_f = p_f(\kappa)$, and a recency parameter R. (p is the mining hardness parameter for Π^p_{Naka} and p_f is the "fruit mining" hardness parameter, as mentionned above, the recency parameter will specify how far back a fruit is allowed to "hang"); the quantity $q = q(\kappa) = \frac{p_f(\kappa)}{p(\kappa)}$ will be useful in our analysis.

The FruitChain Protocol $(\Pi_{fruit}, \text{extract}_{fruit})$

Notation

Validness of fruits, blocks and chains

- We say that a *fruit*, $f = (h_{-1}; h'; \eta, \mathsf{digest}; m; h)$, *is valid* iff $H(h_{-1}; h'; \eta; \mathsf{digest}; \mathsf{m}) = h$ and $[h]_{-\kappa:} < D_{p_f}$ where $[h]_{-\kappa:}$ denotes the last κ bits of h; we call h' the pointer of f. F *is a valid fruit-set* if either $F = \emptyset$ or F is a set of valid fruits.
- We say that a *block*, $\mathsf{b} = ((h_{-1}; h'; \eta; \mathsf{digest}; m; h), F)$, *is valid* iff $\mathsf{digest} = \mathsf{d}(F)$, F is a valid fruit-set, $H(h_{-1}; h'; \eta, \mathsf{d}(F); \mathsf{m}) = h$ and $[h]_{:\kappa} < D_{p_1}$ where $[h]_{:\kappa}$ denotes the first κ bits of h; we call h the reference of b.
- We say that a *sequence of blocks*, $chain = (\mathsf{b}_0, \ldots, \mathsf{b}_\ell)$, *is valid* where $\mathsf{b}_i = ((h^i_{-1}; h'^i; \eta^i, \mathsf{digest}^i; \mathsf{m}^i; h^i), F^i)$ iff
  - $\mathsf{b}_0 = genesis$ where $genesis := ((0; 0; 0; 0; \perp; \mathsf{H}(0; 0; 0; 0, \perp)), \emptyset)$ is the "genesis" block;
  - for all $i \in [\ell]$, $h^i_{-1} = h^{i-1}$,
  - for all $i \in [\ell]$, all $f \in F^i$, there exists some $j \geq i - R\kappa$ such that the pointer of f is h^j.

Recency of fruits w.r.t. *chain*

- finally, we say that the *fruit* f *is* recent *w.r.t. chain* if the pointer of f is the reference of a block in $chain[-R\kappa :]$ (i.e., one of the last $R\kappa$ blocks in *chain*).

The FruitChain Protocol $(\Pi_{fruit}, \text{extract}_{fruit})$

Execution

Π_{fruit}

Initialize: $chain := genesis$, $F = \emptyset$

Upon receiving a valid $fruit$,

- let $F := F \cup \{fruit\}$

Upon receiving a valid $chain'$, if $|chain'| > |chain|$:

- let $chain := chain'$

Every time step, upon receiving input m from the environment:

- let F' be all fruits $f \in F$ that are recent w.r.t. $chain$;
- let h' be the reference of $chain[pos]$ where $pos = max(1, |chain| - \kappa)$;
- let h_{-1} be the reference of $chain[-1]$;
- Pick random $\eta \in \{0,1\}^\kappa$ and let $h := \mathsf{H}(h_{-1}; h'; \eta; \mathsf{d}(F'); \mathsf{m})$
- If $[h]_{-\kappa:} < D_{p_f}$ (i.e., we "mined a fuit")
  - let $fruit := (h_{-1}; h'; \eta; \mathsf{d}(F'); \mathsf{m}, h)$, $F := F \cup \{fruit\}$, and broadcast $fruit$
- If $[h]_{:\kappa} < D_p$ (i.e., we "mined a block")
  - let $chain := chain || ((h_{-1}; h'; \eta, \mathsf{d}(F'); \mathsf{m}, h), F)$, and broadcast $chain$

extract_{fruit}

On input a valid chain *chain*, output the sequence of messages m contained in the fruit contained in blocks of *chain*, ordered by:

- the block (which contained the fruit, which contains the message)—that is, messages inside fruits inside earlier blocks, come earlier;
- in the case of ties (i.e., if some block contains multiple fruits), break ties by the pointer of the fruit (giving preference to fruit pointing to earlier blocks), and finally if have the same pointer, just lexicographically.[a]

[a] The method for how we break times among fruit in the *same* block is inconsequential for our results.

Security of Blockchain Protocols

(With Overwhelming Probability)

T_0-consistency

At any point, the chains of two honest players can differ only in the last T_0 blocks.

Chain growth rate T_0, g_0, g_1

At any point, the chain of honest players grows by

- at least T_0 messages in the next $\frac{T_0}{g_0}$ rounds, and
- at most T_0 messages in the next $\frac{T_0}{g_1}$ rounds.

Chain quality T_0, μ

For any T_0 consecutive messages in any chain held by some honest player, the fraction of messages that were contributed by honest players is at least μ.

Fairness T_0, δ w.r.t. ρ attackers $\left(\Rightarrow \text{chain quality } T_0, (1-\delta)(1-\rho)\right)$

Any $\phi \leq 1-\rho$ fraction coalition of honest users is guaranteed to get at least a $(1-\delta)\phi$ fraction of the blocks in every T_0-long window.
This condition implies chain quality $T_0, (1-\delta)(1-\rho)$ by considering $\phi = 1-\rho$.

Security of Nakamoto and FruitChain

Theorem 2.7 (Security of Nakamoto [PSS16]). *For any $\delta > 0$, any $\lambda > 1$, any $p(\cdot)$, any superlogarithmic function $T_0(\cdot)$, $(\Pi^p_{Nak}, \mathsf{extract}^p_{nak})$ satisfies:*

- *T_0-consistency;*
- *chain growth rate $(T_0, g_0^{p,\delta}, g_1^{p,\delta})$ where*

$$g_0^{p,\delta}(\kappa, n, \rho, \Delta) = (1-\delta)\gamma$$

$$g_1^{p,\delta}(\kappa, n, \rho, \Delta) = (1+\delta)np$$

- *chain quality $T_0, \mu^p_\delta(\kappa, n, \rho, \Delta)$ where*

$$\mu^p_\delta(\kappa, n, \rho, \Delta) = 1 - (1+\delta)\frac{\beta}{\gamma};$$

in Γ^p_λ environments.

Theorem 4.1. *For any $0 < \delta < 1$, any $\lambda > 1$, and any $p(\cdot), p_f(\cdot)$, let $R = 17$, $\kappa_f(\kappa) = 2q(\kappa)R\kappa$, and $T_0(\kappa) = 5\frac{\kappa_f}{\delta}$. Then $(\Pi^{p,p_f,R}_{fruit}, \mathsf{extract}^{p,p_f,R}_{fruit})$ satisfies:*

- *κ_f-consistency;*
- *chain growth rate $(T_0, g_0^{p,\delta}, g_1^{p,\delta})$ where*

$$g_0^{p,\delta}(\kappa, n, \rho, \Delta) = (1-\delta)(1-\rho)np_f,$$

$$g_1^{p,\delta}(\kappa, n, \rho, \Delta) = (1+\delta)np_f$$

- *fairness (T_0, δ).*

in Γ^p_λ environments.

Proof Overview

Preventing block/fruit withholding attack

- Messages are stored in fruits instead of blocks
- Once a fruit f is mined, it's broadcast to all other players; other players will try to mine a block and include f in it
- By the chain growth and chain quality (of Nakamoto), such an block arrives soon enough, so that f is still recent
- As long as f is valid, other player will be accepted by other honest players; it can't be replaced (erased)

Preventing attackers from releasing lots of fruits at a time

- Which creates a very high fraction of adversarial fruit in some segment of the chain
- By requiring the fruits to be recent, and by chain growth (of Nakamoto), the attacker can't wait too long to release fruits, otherwise they'll expire

An Application

Reasons for mining pools

- In Bitcoin, the mining hardness is set so that the world finds a new block every 10 minutes, to ensure consistency
- It takes a very long time for an individual miner to mine a block
- The payments received by miners has a very high variance

How mining pools work

- Miners come together to pool their work and share the reward
- To prevent free-riding, miners submit partial proofs of work
- Rewards are distributed among the contributor of the partial proofs-of-work
- Centralization

Preventing mining pools in FruitChains

- Set p appropriately to ensure consistency
- Set p_f larger, and as large as the probability to find a partial proof-of-work
- Reduce the variance in a *fully decentralized* way

α, β, γ and Γ_λ^p

Parameters α, β, γ that parameterize security of blockchain

- let $\alpha(\kappa, n, \rho, \Delta) = 1 - (1 - p(\kappa))^{(1-\rho)n}$. That is, α is the probability that *some* honest player succeeds in mining a block in a round;
- let $\beta(\kappa, n, \rho, \Delta) = \rho n p(\kappa)$. That is β is the expected number blocks that an attacker can mine in a round.
- let $\gamma(\kappa, n, \rho, \Delta) = \frac{\alpha}{1+\Delta\alpha}$. γ is a "discounted" version of α which takes into account the fact that messages sent by honest parties can be delayed by Δ rounds and this may lead to honest players "redoing work"; γ corresponds to their "effective" mining power.

Predicate Γ_λ^p

Let $\Gamma_\lambda^p(n(\cdot), \rho, \Delta(\cdot)) = 1$ iff $n(\cdot), \Delta(\cdot)$ are polynomially-bounded functions $\mathbb{N} \to \mathbb{N}^+$, $0 \leq \rho \leq 1$ and for all $\kappa, n = n(\kappa), \Delta = \Delta(k)$,

$$\alpha(1 - 2(\Delta + 1)\alpha) \geq \lambda\beta$$

Consistency

consistent^T(view)

Let $\mathsf{consistent}^T(\mathsf{view}) = 1$ iff for all rounds $r \leq r'$, and all players i, j (potentially the same) such that i is honest at view^r and j is honest at $\mathsf{view}^{r'}$, we have that the prefixes of $\mathsf{extract}_i^r(\mathsf{view})$ and $\mathsf{extract}_j^{r'}(\mathsf{view})$ consisting of the first $\ell = |\mathsf{extract}_i^r(\mathsf{view})| - T$ records are identical.[14]

$T_0(\cdot)$-consistency in Γ environments

Definition 2.4. *A blockchain protocol* $(\Pi, \mathsf{extract})$ *satisfies* $T_0(\cdot)$-consistency in Γ environments, *if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$, *there exists some negligible function* ϵ *such that for every* $\kappa \in \mathbb{N}$ *and every* $T \geq T_0(\kappa)$ *the following holds:*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi,\mathit{extract})}(A, Z, \kappa) : \mathit{consistent}^T(\mathit{view}) = 1\right] \geq 1 - \epsilon(\kappa)$$

Consequence of T-consistency

Note that a direct consequence of consistency is that the chain *length* of any two honest players can differ by at most T (except with negligible probability).

Chain Growth

$$\mathsf{min\text{-}chain\text{-}increase}_{r,t}(\mathsf{view}) = \min_{i,j} |\mathsf{extract}_j^{r+t}(\mathsf{view})| - |\mathsf{extract}_i^r(\mathsf{view})|$$

$$\mathsf{max\text{-}chain\text{-}increase}_{r,t}(\mathsf{view}) = \max_{i,j} |\mathsf{extract}_j^{r+t}(\mathsf{view})| - |\mathsf{extract}_i^r(\mathsf{view})|$$

Let $\mathsf{growth}^{t_0,t_1}(\mathsf{view}, \Delta, T) = 1$ iff the following two properties hold:

- **(consistent length)** for all rounds $r \leq |\mathsf{view}| - \Delta$, $r + \Delta \geq r' \leq |\mathsf{view}|$, for every two players i, j such that in view i is honest at r and j is honest at r', we have that $|\mathsf{extract}_j^{r'}(\mathsf{view})| \geq |\mathsf{extract}_i^r(\mathsf{view})|$
- **(chain growth lower bound)** for every round $r \leq |\mathsf{view}| - t$, we have

$$\mathsf{min\text{-}chain\text{-}increase}_{r,t_0}(\mathsf{view}) \geq T.$$

- **(chain growth upper bound)** for every round $r \leq |\mathsf{view}| - t$, we have

$$\mathsf{max\text{-}chain\text{-}increase}_{r,t_1}(\mathsf{view}) \leq T.$$

Chain growth rate $T_0(\cdot)$, $g0(\cdot,\cdot,\cdot,\cdot)$, $g_1(\cdot,\cdot,\cdot,\cdot)$ in Γ-environments

Definition 2.2. *A blockchain protocol* $(\Pi, \mathit{extract})$ *has* chain growth rate $T_0(\cdot), g_0(\cdot,\cdot,\cdot,\cdot), g_1(\cdot,\cdot,\cdot,\cdot)$ in Γ-environments *if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, there exists some negligible function* ϵ *such that for every* $\kappa \in \mathbb{N}$, $T \geq T_0(\kappa)$, $t_0 \geq \frac{T}{g_0(\kappa,n(\kappa),\rho,\Delta(\kappa))}$ *and* $t_1 = \frac{T}{g_1(\kappa,n(\kappa),\rho,\Delta(\kappa))}$ *the following holds:*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi,\mathit{extract})}(A, Z, \kappa) : \mathit{growth}^{t_0,t_1}(\mathit{view}, \Delta(\kappa), \kappa) = 1\right] \geq 1 - \epsilon(\kappa)$$

Chain Quality

Non-adversarial record m w.r.t. view and prefix $\vec{m}$

We say that a *record* m *is non-adversarial (or honest) w.r.t. view and prefix* $\vec{\mathsf{m}}$ if there exists a player j and some round r' such that in $\mathsf{view}^{r'}$, j is honest, the environment provided m as input to j, and $\vec{\mathsf{m}}$ is a prefix of $\mathsf{extract}_i(\mathsf{view}^{r'})$. (That is, there exists some honest player that received m as an input when their chain contained $\vec{\mathsf{m}}$).

$\mathrm{quality}^T(\mathrm{view}, \mu)$

Let $\mathsf{quality}^T(\mathsf{view}, \mu) = 1$ iff for every round r and every player i such that i is honest in view^r, among any consecutive sequence of T records M in $\mathsf{extract}_i^r(\mathsf{view})$, the fraction of records m that are honest w.r.t. view^r and $\vec{\mathsf{m}}$, where $\vec{\mathsf{m}}$ is the prefix of $\mathsf{extract}_i^r(\mathsf{view})$ preceeding M, is at least μ.

Chain quality $T_0(\cdot)$, $\mu(\cdot,\cdot,\cdot,\cdot)$ in Γ environments

Definition 2.3. *A blockchain protocol* $(\Pi, \mathsf{extract})$ *has* chain quality $T_0(\cdot), \mu(\cdot,\cdot,\cdot,\cdot)$ in Γ environments, *if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$, *there exists some negligible function* ϵ *such that for every* $\kappa \in \mathbb{N}$ *and every* $T \geq T_0(\kappa)$ *the following holds:*

$$\Pr\left[\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa) : \mathsf{quality}^T(\mathsf{view}, \mu(\kappa, n(\kappa), \rho, \Delta(\kappa))) = 1\right] \geq 1 - \epsilon(\kappa)$$

Fairness

- Let a *player subset selection*, S, be a function that given a view view outputs a subset of the players that are honest at view.
- We say that S is a *ϕ-fraction player subset selection* if $S(\mathsf{view})$ always outputs a set of size ϕn (rounded upwards) where n is the number of players in view.
- Given an player subset selection S, we say that a *record m is S-compatible w.r.t. view and prefix $\vec{\mathsf{m}}$* if there exists a player j and some round r' such that j is in $S(\mathsf{view}^{r'})$, the environment provided m as an input to j at $\mathsf{view}^{r'}$, and $\vec{m}$ is a prefix of $\mathsf{extract}_i^{r'}(\mathsf{view})$.
- Let $\mathsf{quality}^{T,S}(\mathsf{view}, \mu) = 1$ iff for every round r and every player i such that i is honest in view^r, we have that among any consecutive sequence of T records M in $\mathsf{extract}_i^r(\mathsf{view})$, the fraction of records m that are S-compatible w.r.t. view^r and $\vec{m}$, where $\vec{\mathsf{m}}$ is the prefix of $\mathsf{extract}_i^r(\mathsf{view})$ preceeding M, is at least μ.

Fairness $T_0(\cdot), \delta$ in Γ environments

Definition 3.1. *A blockchain protocol $(\Pi, \mathsf{extract})$ has (approximate)* fairness $T_0(\cdot), \delta$ in Γ environments, *if for all Γ-admissible $(n(\cdot), \rho, \Delta(\cdot), A, Z)$, every $\phi \leq 1 - \rho$, every ϕ-fraction subset selection S, there exists some negligible function ϵ such that for every $\kappa \in \mathbb{N}$ and every $T \geq T_0(\kappa)$ the following holds:*

$$\Pr\left[\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa) : \mathsf{quality}^{T,S}(\mathsf{view}, (1-\delta)\phi)) = 1\right] \geq 1 - \epsilon(\kappa)$$

Fact 3.2. *If a blockchain protocol $(\Pi, \mathsf{extract})$ satisfies $(T_0(\cdot), \delta)$-fairness in Γ environments, then it satisfies (T_0, μ)-chain quality, where $\mu(\kappa, n, \rho, \Delta) = (1-\delta)(1-\rho) \geq 1 - (1+\delta)\rho$ when $\rho \leq \frac{1}{2}$.*

Blockchain Protocols

Blockchain protocols

A blockchain protocol is a pair of algorithms $(\Pi, \mathsf{extract})$ where Π is a stateful algorithm that receives a security parameter κ as inputs and maintains a local state *chain*. The algorithm $\mathsf{extract}(\kappa, chain)$ outputs an *ordered* sequence of "records", or "batches", $\vec{\mathsf{m}}$ (e.g., in the bitcoin protocol, each such record is an ordered sequence of transactions). We call $\mathsf{extract}(\kappa, chain)$ the "record chain" of a player with security parameter κ and local variable *chain*; to simplify notation, whenever κ is clear from context we often write $\mathsf{extract}(chain)$ to denote $\mathsf{extract}(\kappa, chain)$.

A blockchain execution

A Blockchain Execution We consider the execution of a blockchain protocol $(\Pi, \mathsf{extract})$ that is directed by an environment $Z(1^\kappa)$ (where κ is a security parameter), which activates a number of parties $1, 2, \ldots, n$ as either "honest" or corrupted parties. Honest parties execute Π on input 1^κ with an empy local state *chain*; corrupt parties are controlled by an attacker A which reads all their inputs/message and sets their outputs/messages to be sent.

A Blockchain execution

- The execution proceeds in *rounds* that model time steps. In round r, each honest player i receives a message (a "record") m from Z (that it attempts to "add" to its chain) and potentially receives incoming network messages (delivered by A). It may then perform any computation, *broadcast* a message to all other players (which will be delivered by the adversary; see below) and update its local state $chain_i$.
- A is responsible for delivering all messages sent by parties (honest or corrupted) to *all* other parties. A cannot modify the content of messages broadcast by honest players, *but it may delay or reorder the delivery of a message* as long as it eventually delivers all messages. (Later, we shall consider restrictions on the delivery time.) The identity of the sender is not known to the recipient.[10]
- At any point, Z can communicate with adversary A or access $\mathsf{extract}(chain_i)$ where $chain_i$ is the local state of player i.
- At any point, Z can *corrupt* an honest party j which means that A gets access to its local state and subsequently, A controls party j. (In particular, this means we consider a model with "erasures"; random coin tosses that are no longer stored in the local state of j are not visible to A.)[11]
- At any point, Z can *uncorrupt* a corrupted player j, which means that A no longer controls j and instead player j starts executing $\Pi(1^\kappa)$ with a fresh state $chain_j$. (This is also how we model Z spawning a "new" honest player.) A gets informed of all such uncorrupt messages and is required to deliver all messages previously sent by (currently alive) honest players.[12]