THE THETA BLOCKCHAIN

Theta is a decentralized video streaming network,
powered by a new blockchain and token.

By Theta Labs, Inc.

Last Updated: Nov 21, 2017
Version 1.0

OUTLINE

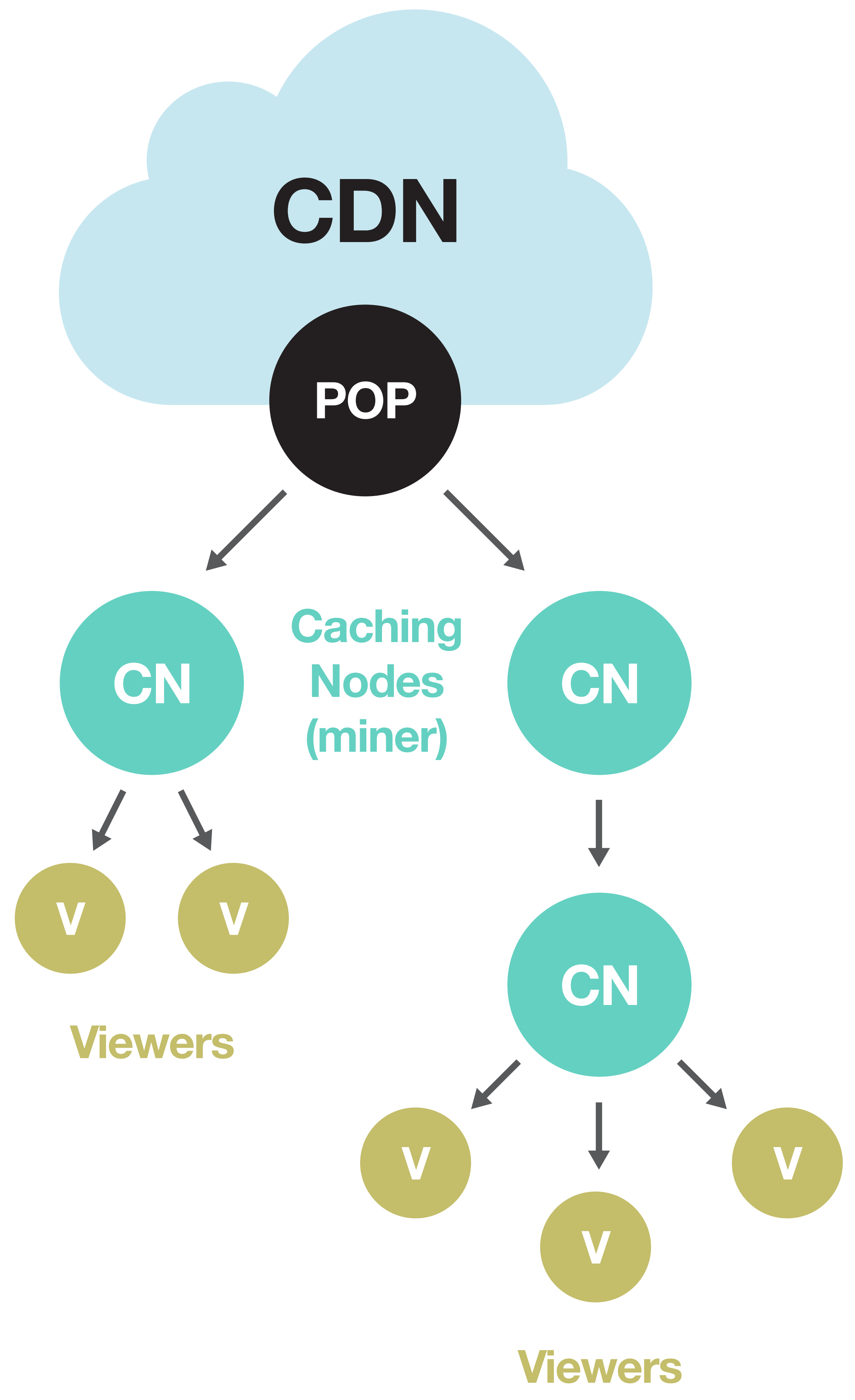
MOTIVATION

Video streaming challenges

- **"Last-mile" delivery problem:** Content delivery network (CDN) servers too far from end viewers, especially in developing countries.
- **High CDN bandwidth cost:** CDN cost for popular streaming sites could be tens of millions of dollars per year

Proposal: Encourage viewers to share their redundant bandwidth resources to relay video stream to their neighboring viewers

- **Caching nodes:** viewers that share their computers to relay video streams
- Viewers are closer to each other geographically, which help address the "last-mile" stream delivery problem
- Could reduce CDN bandwidth cost significantly if most of the end viewers pull stream from peer nodes



MOTIVATION

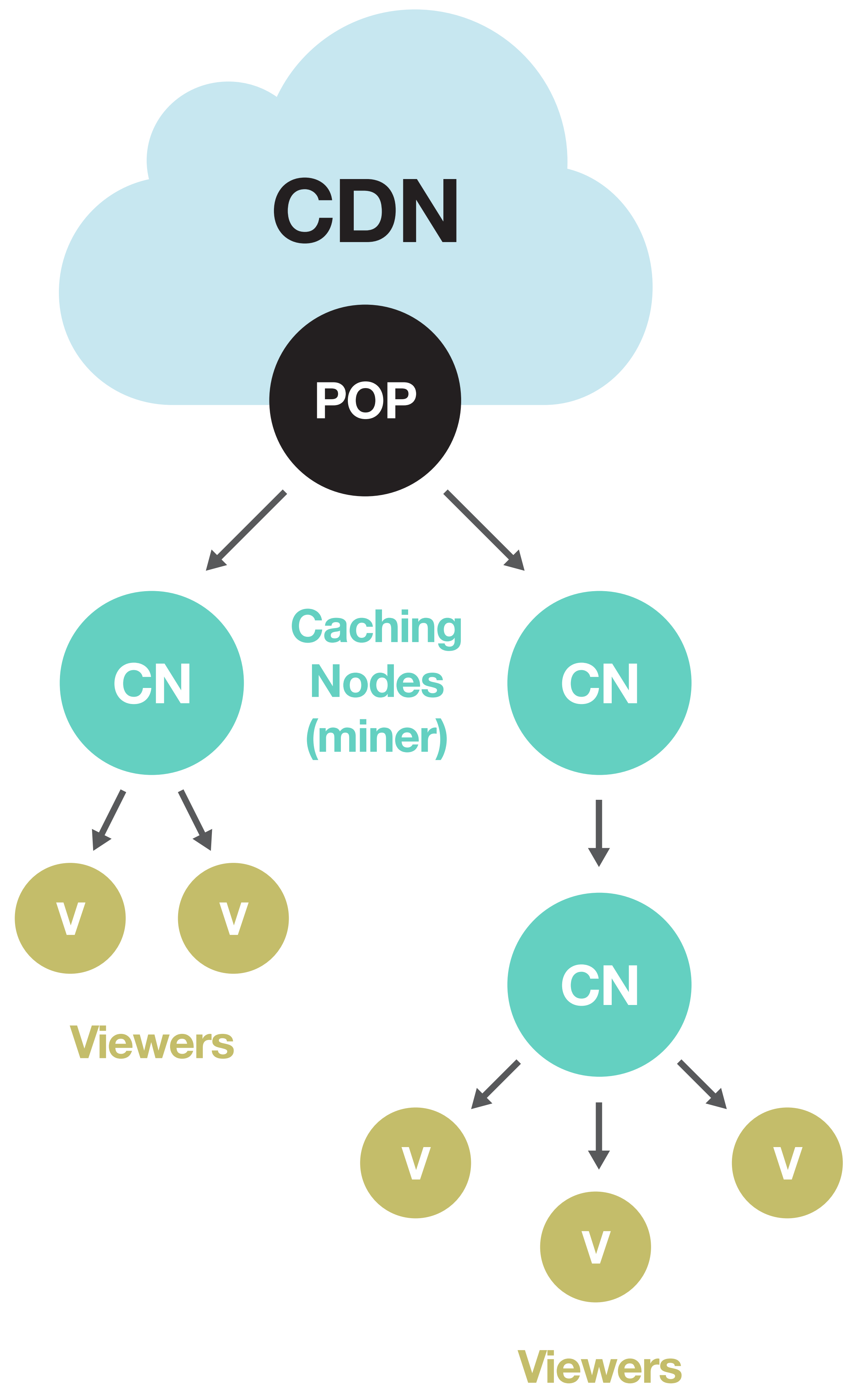
The Theta protocol: a blockchain based incentive mechanism to encourage viewers to share their bandwidth

Incentive mechanism: traditional model

- Tempting to follow traditional model where viewers send Theta tokens to caching nodes for their video relay service
- However, this model is impractical and not sustainable in the long-run since today viewers are already used to free videos on Youtube/Twitch, etc.

Incentive mechanism: reputation dependent mining

- Caching nodes play the role of blockchain miners
- They obtain mining rewards for assembling new blocks and relaying video streams
- But unlike Bitcoin, the block reward is **not** a constant
- A caching node can gain **reputation** by relaying video streams
- With a higher reputation score, caching nodes can claim higher block rewards



REPUTATION DEPENDENT MINING

Reputation dependent block reward

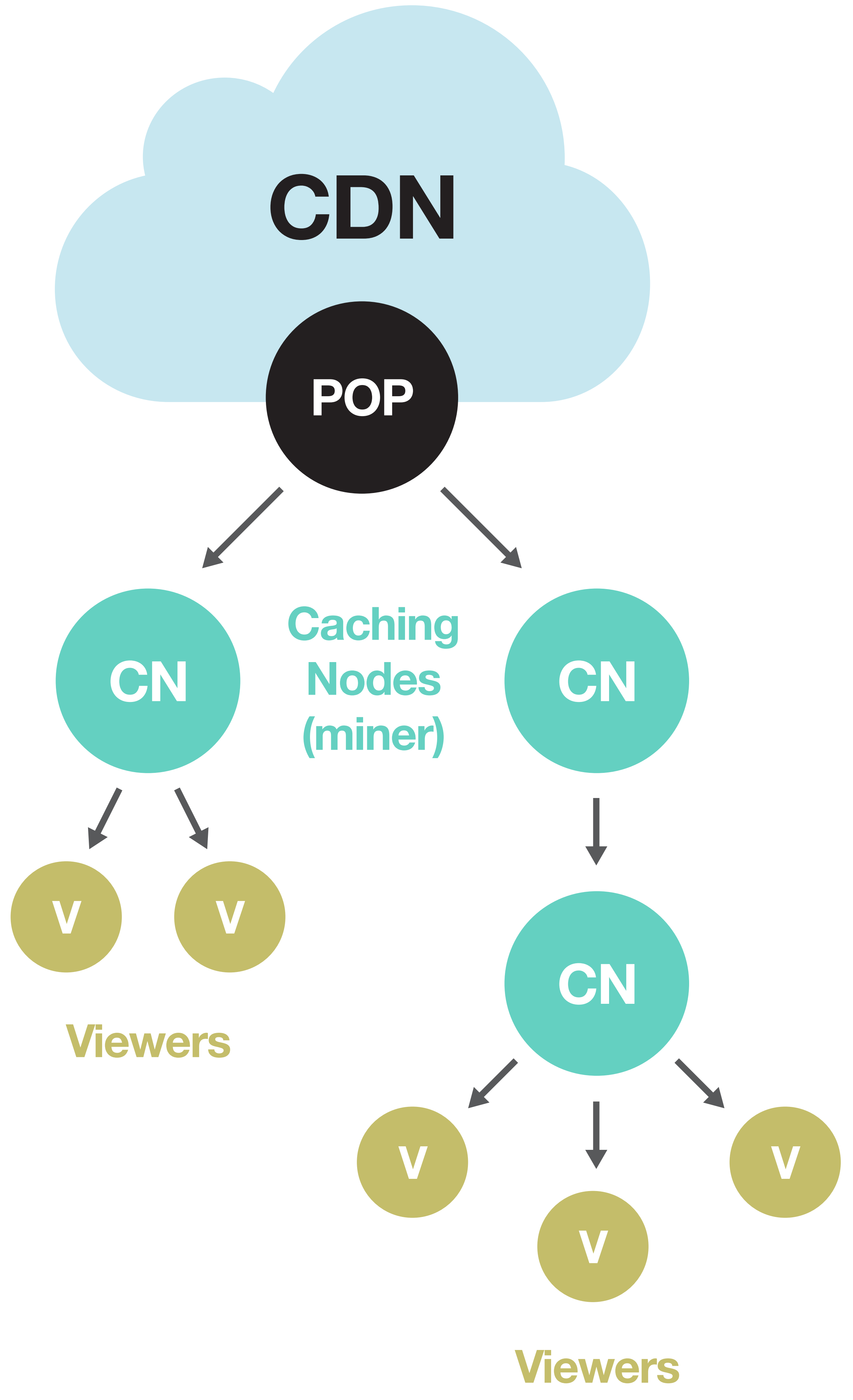
- $reward = (1 + r) \cdot reward_{base}$
- r is the reputation score of the caching node that mined the new block, $reward_{base}$ is a constant shared among all miners
- High reputation score leads to high block reward

Reputation score r

- A real number between 0 and 1 which measures the amount of video relaying work a caching node performed during a recent time window
- More video relaying work leads to a higher reputation score
- **Recent time window:** e.g. the past 24 hours, we account for recency so a caching node cannot rely on work done in the distant past to claim high reputation score

Key design elements follow

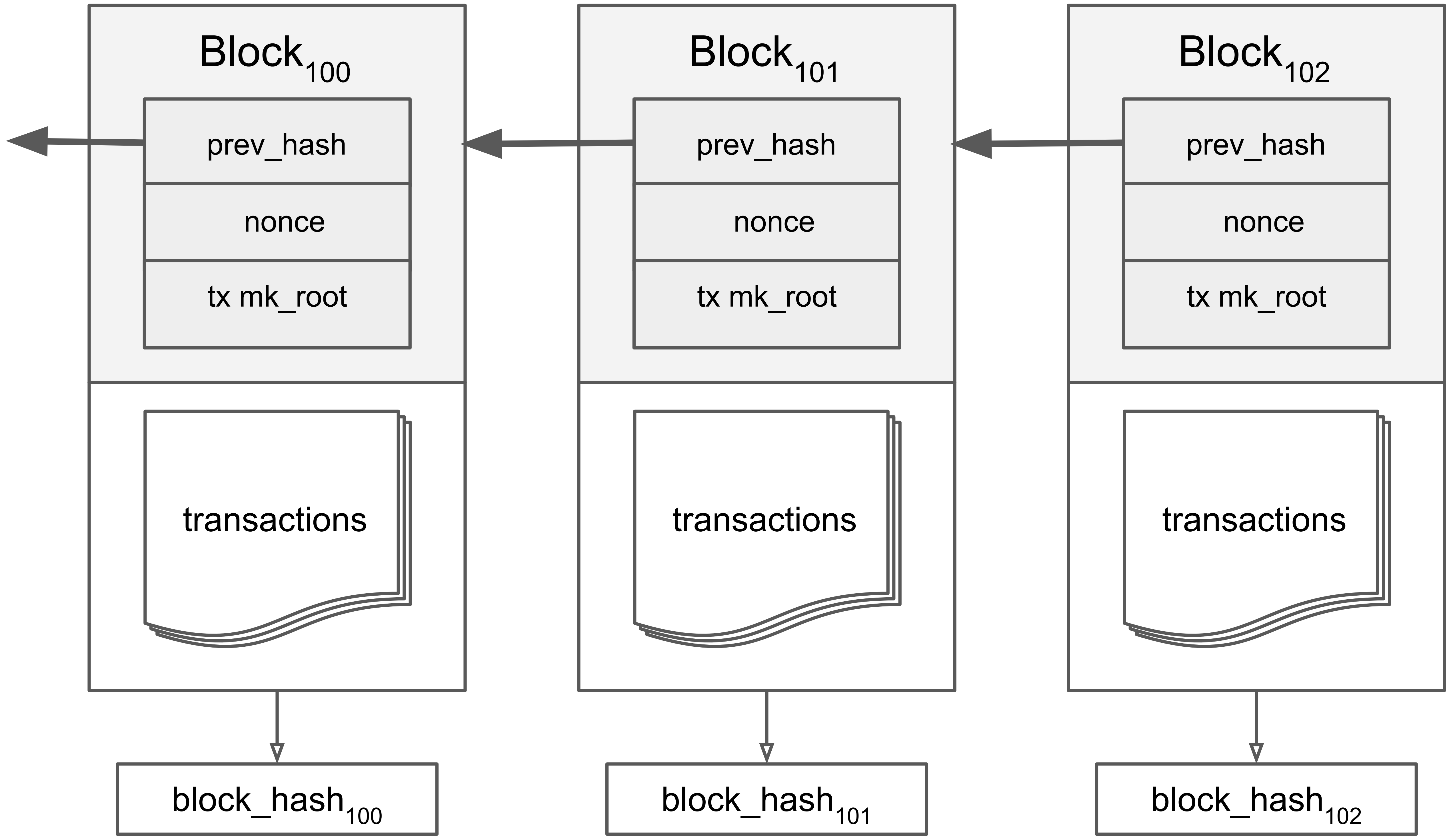
- How Theta calculates the reputation score
- When a new block is mined, how Theta proves its reputation score to other miners and form a global consensus



REPUTATION DEPENDENT MINING

Service Certificate

- A service certificate is a solution to the **"service certificate PoW puzzle"**, defined as a minimization problem in the right figure
- Requires hashing power to compute the solution, hence difficult to fake without spending hashing power
- Assume $nonce^*$ is the best solution viewer j found. The corresponding hash value H^* reflects the time viewer j spent to find the solution, which is equal to the time caching node i served viewer j
    - $H^* = \mathrm{HASH}(pki \,||\, pkj \,||\, block\_hash_k \,||\, nonce^*)$
    - The expected amount of time node i served viewer j $\mathrm{E}[t \mid H^*] = M / (s \cdot H^*)$, where s is the required number of hashes per unit time
    - The HASH function needs to be ASIC resistant to reduce the advantage of using special-purpose mining hardware



Service certificate PoW puzzle

$nonce^* = \mathbf{argmin}\ \mathrm{HASH}(pk_i \,||\, pk_j \,||\, block\_hash_k \,||\, nonce)$

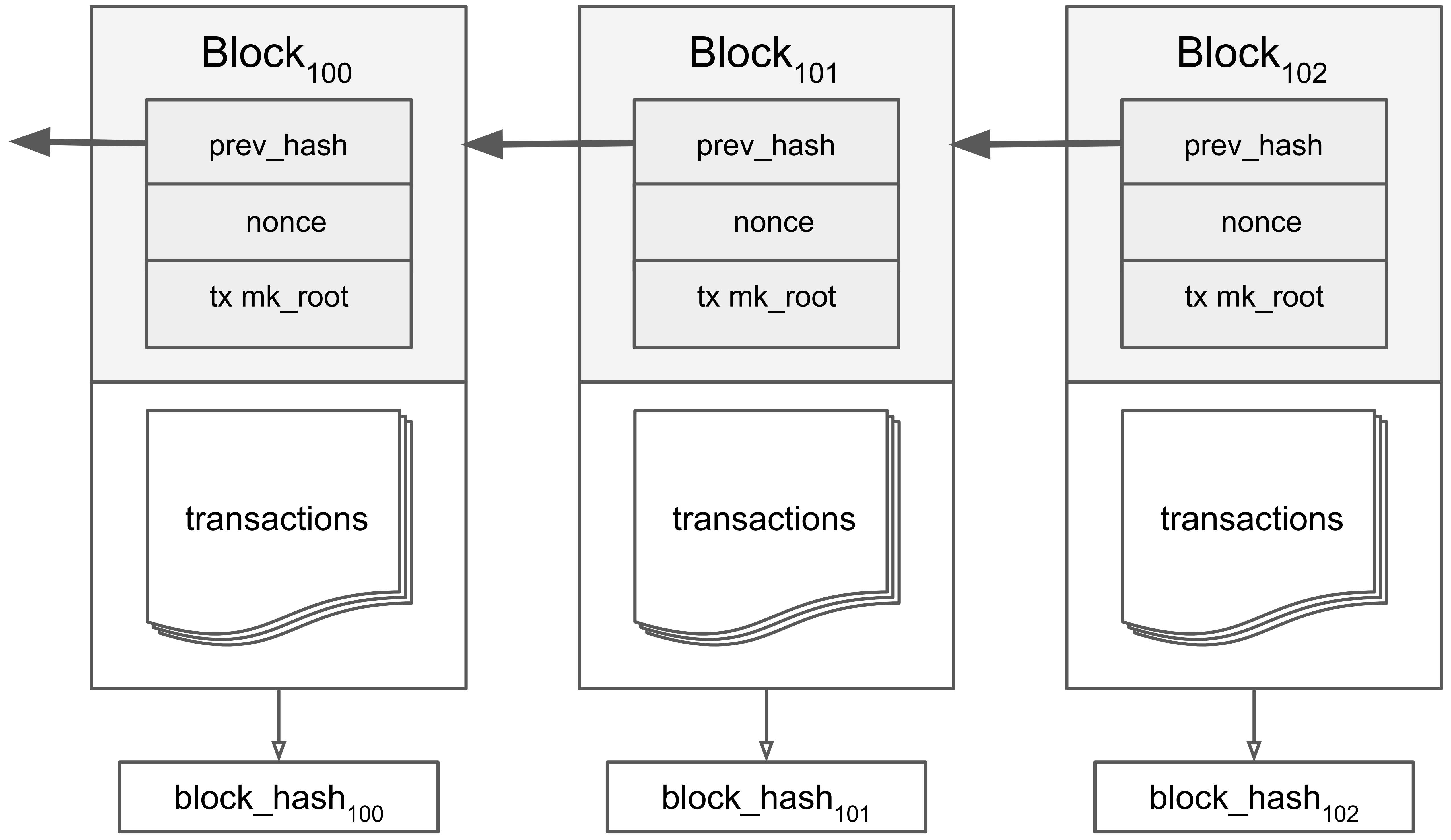
Service certificate

$(pk_i, pk_j, block\_hash_k, nonce)^*$

REPUTATION DEPENDENT MINING

Service Certificate

- The caching node sends a new service certificate PoW puzzle to its downstream viewers whenever a new block is added to the longest chain
- Each puzzle in effect points to a specific block through the $block\_hash_k$ parameter
- Thus we can use the corresponding block to infer the generation time of the service certificate



Service certificate PoW puzzle

$nonce^* = \mathbf{argmin}\ \mathrm{HASH}(pk_i \,||\, pk_j \,||\, block\_hash_k \,||\, nonce)$

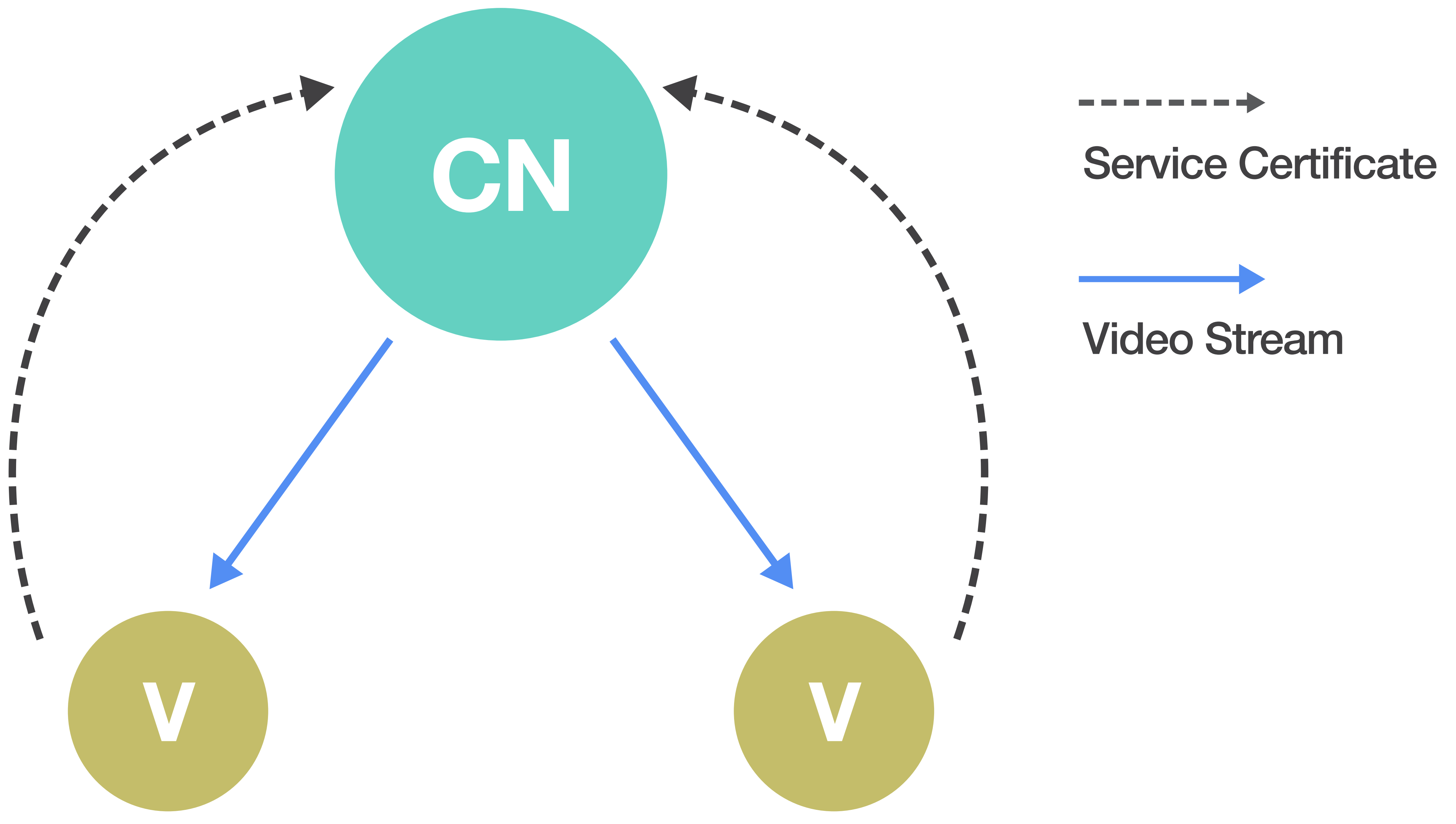
Service certificate

$(pk_i, pk_j, block\_hash_k, nonce)^*$

REPUTATION DEPENDENT MINING

The solution

- Caching miner node i relays video stream to viewer node j as a service
- In return, viewer j needs to spend its hashing power compute **"service certificates"** and sends back to caching node i regularly (e.g. every minute)
- If viewer j stops sending the service certificates, caching miner node i can terminate the video stream
- Whenever caching node i mines a new block, it gathers all the service certificates received in the defined time window (e.g. the past 24 hours)
- The caching miner node can then calculate its reputation score r using these service certificates
- The caching miner node then presents these service certificates to other miners to justify its reputation score, and thus its block reward



Service certificate PoW puzzle

$nonce^* = \mathbf{argmin}\ \mathrm{HASH}(pk_i \,||\, pk_j \,||\, block\_hash_k \,||\, nonce)$

Service certificate

$(pk_i, pk_j, block\_hash_k, nonce)^*$

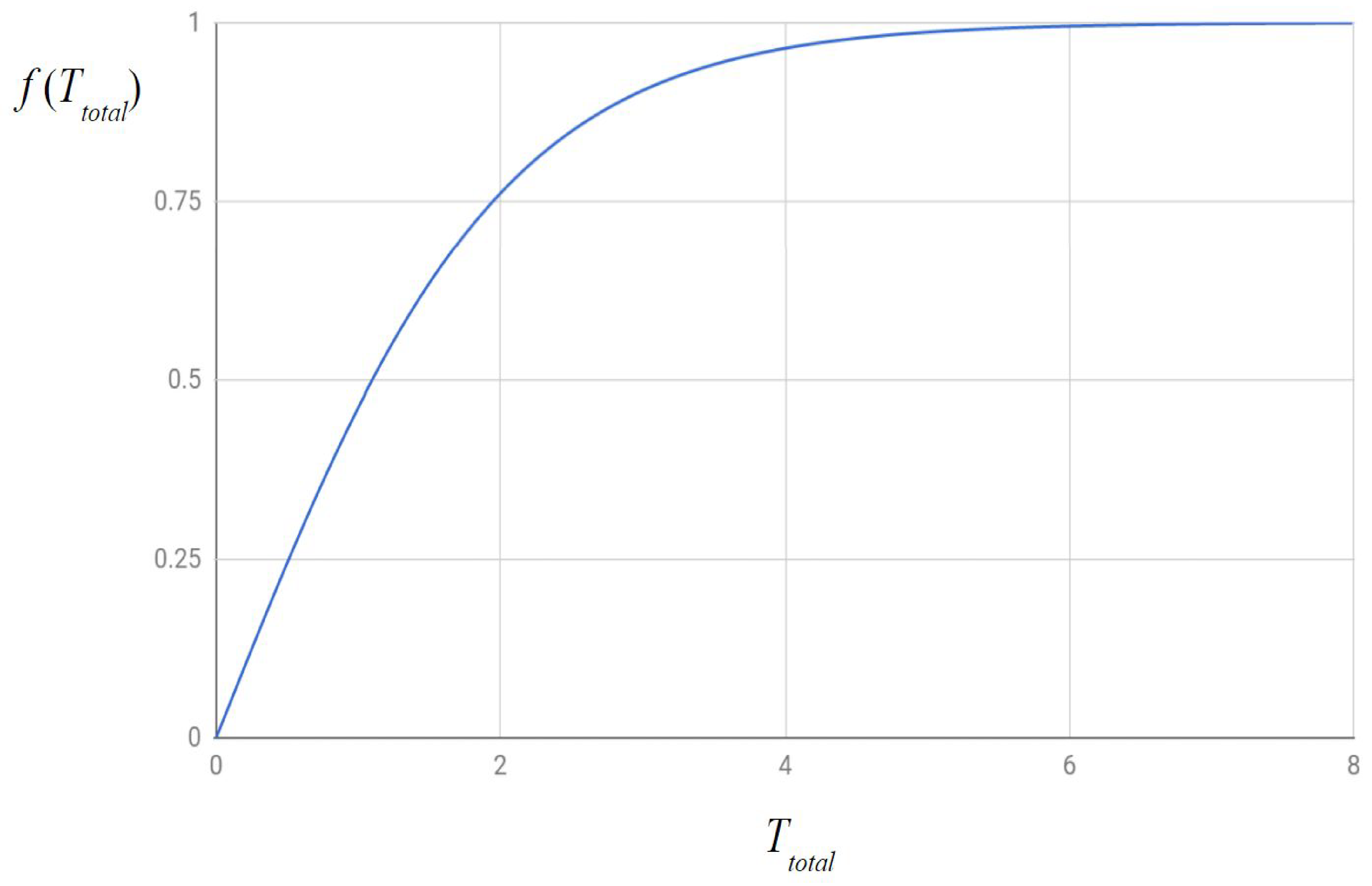
REPUTATION SCORE CALCULATION

Reputation score calculation

- Given a service certificate $sc = (pk_i, pk_j, block\_hash_k, nonce^*)$, the expected amount of time node i served node j can be calculated with
  - $T_{sc} = \mathbf{min}(\,\mathbf{E}[t \mid H^*], T_b) = \mathbf{min}(M / (s \cdot H^*), T_b)$, where T_b is the expected amount of time to mine a new block

- Thus, the expected total amount of time node i served its downstream nodes can be calculated as the sum of T_{sc} for each service receipt it collected
  - $T_{total} = \mathbf{SUM}(T_{sc})$

- The reputation score $\boldsymbol{r = f(T_{total})}$ is a monotonically increasing function of T_{total} which maps T_{total} to a real number in the range of [0, 1] as shown in the right figure

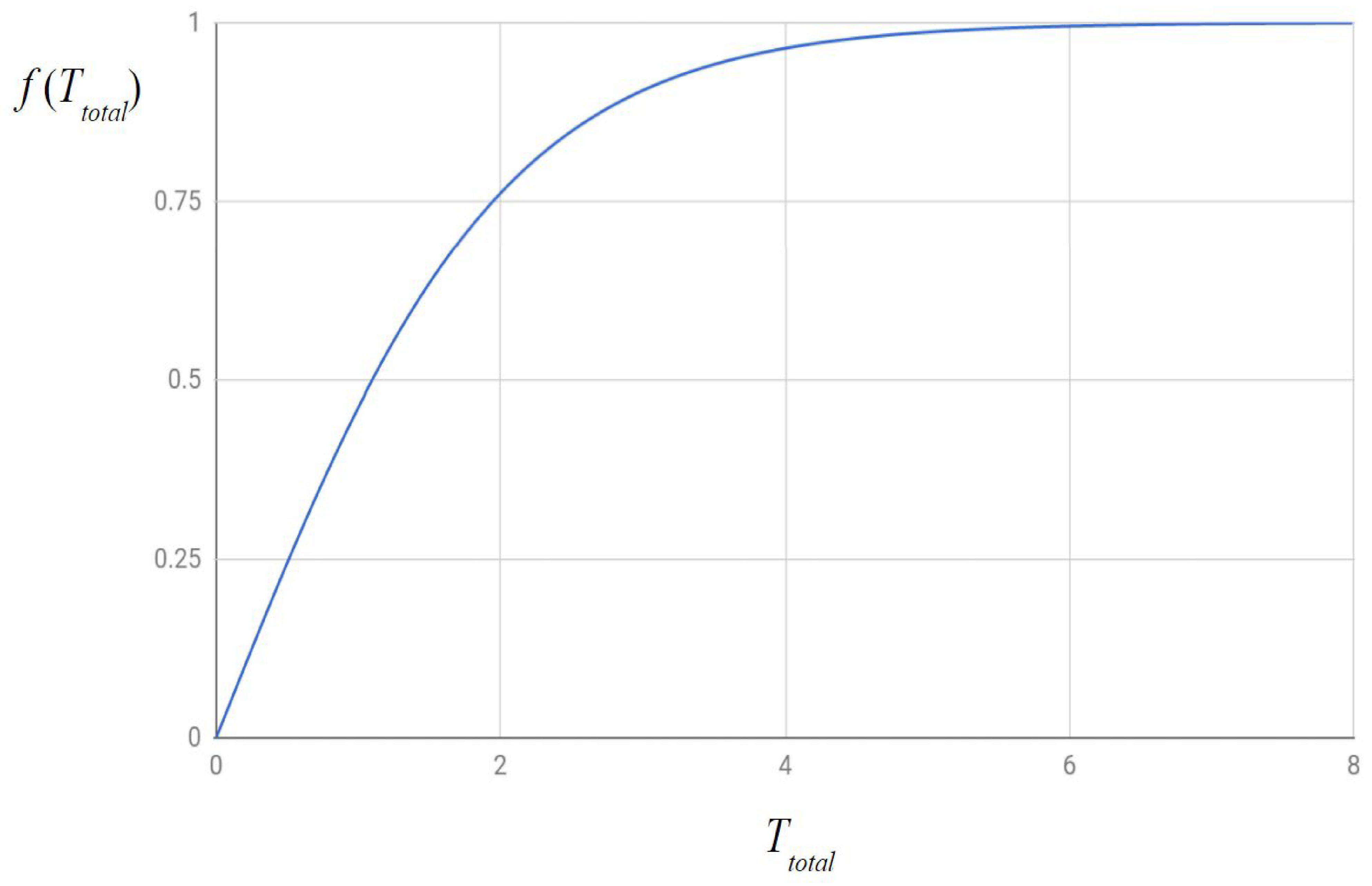


REPUTATION SCORE CALCULATION

Reputation score calculation

- Can we design $r = f(T_{total})$ such that a rational miner would not try to forge fake service certificates?

- **The answer is yes.** Intuitively, a miner needs to spend hashingpower to make fake service certificates. Thus if it tries to make fake receipts, its chance of mining the next block reduces. It is possible to design $r = f(T_{total})$ such that the expected mining reward per unit time **strictly** reduces if the miner splits its hashing power to make fake service certificates. See the Appendix in the whitepaper for further analysis.



GLOBAL REPUTATION CONSENSUS

Extend the blockchain data structure to facilitate global reputation consensus

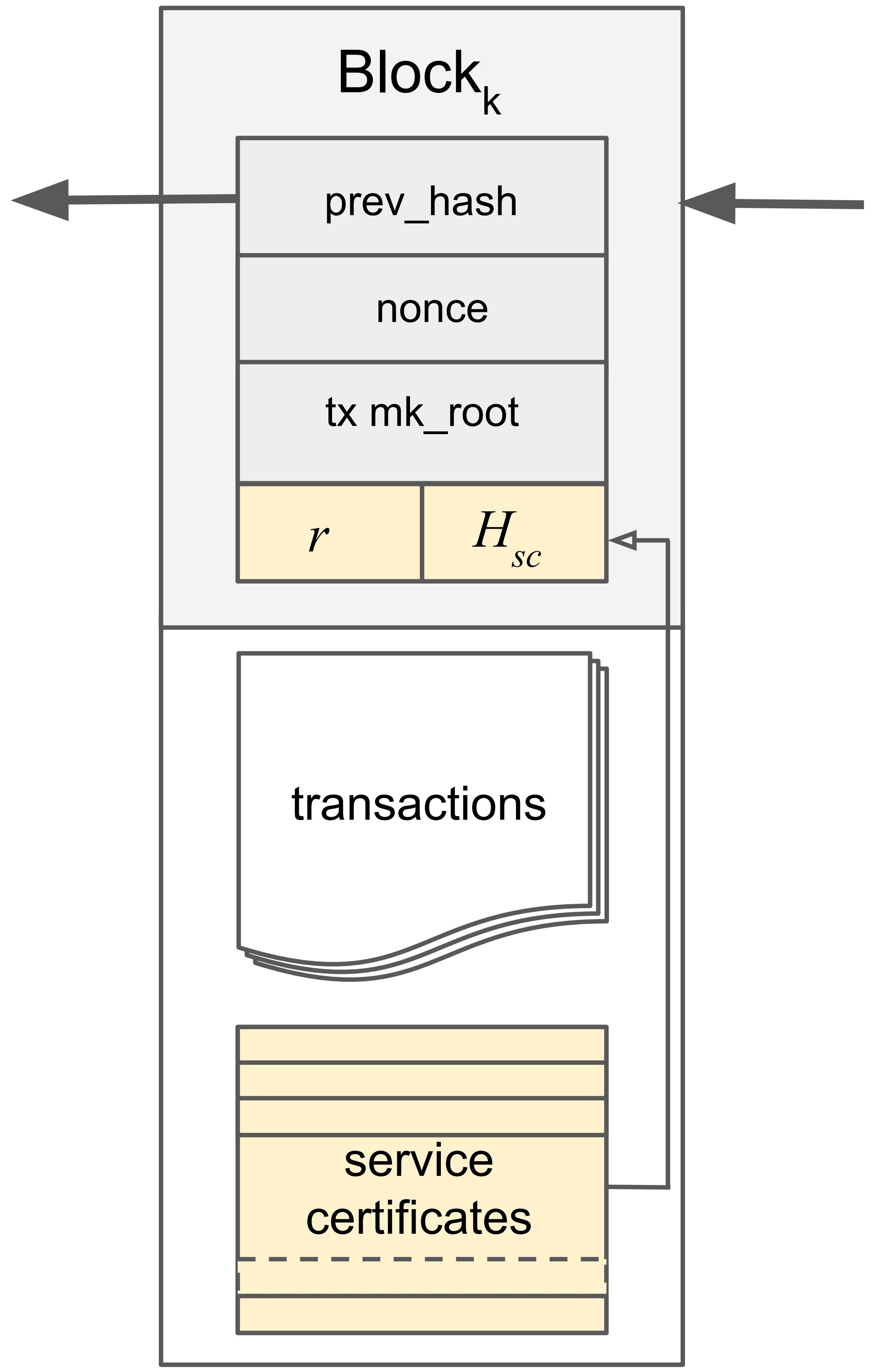
- So that other miners can validate the reputation score of the miner that mined the new block

Block header additional fields

- Reputation score r of the miner that mined the block
- H_{sc}, the hash of the concatenation of all "service certificates"
- Note that in the following mining PoW puzzle, the *block_header* needs to contain r and H_{sc} to prevent potential tampering
  - $\mathrm{HASH}(block\_header \,||\, nonce) < \mathrm{R}$

Block body

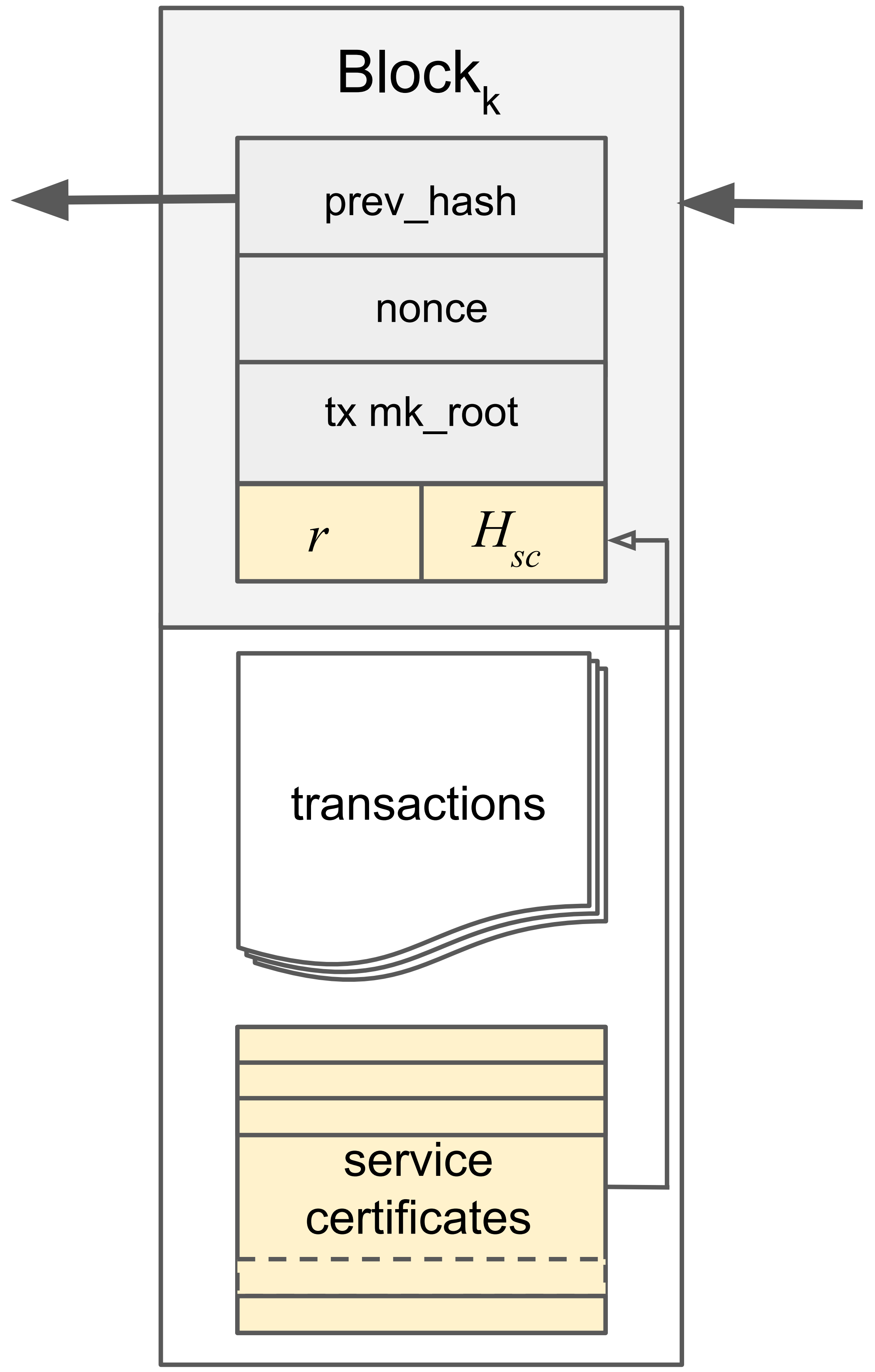
- Adding the service certificates to the block body. Typical size is 100K+ bytes per block for individual caching miners



GLOBAL REPUTATION CONSENSUS

The blockchain protocol

- Once a miner solves the PoW mining puzzle, it assembles a new block with the additional fields described in the previous slide including its reputation score calculated with the service certificates it collected
- The miner sends the new block through the network
- Other miners validate the PoW mining puzzle solution, the Theta token transactions, and verify the reputation score using the service certificates embedded in the block



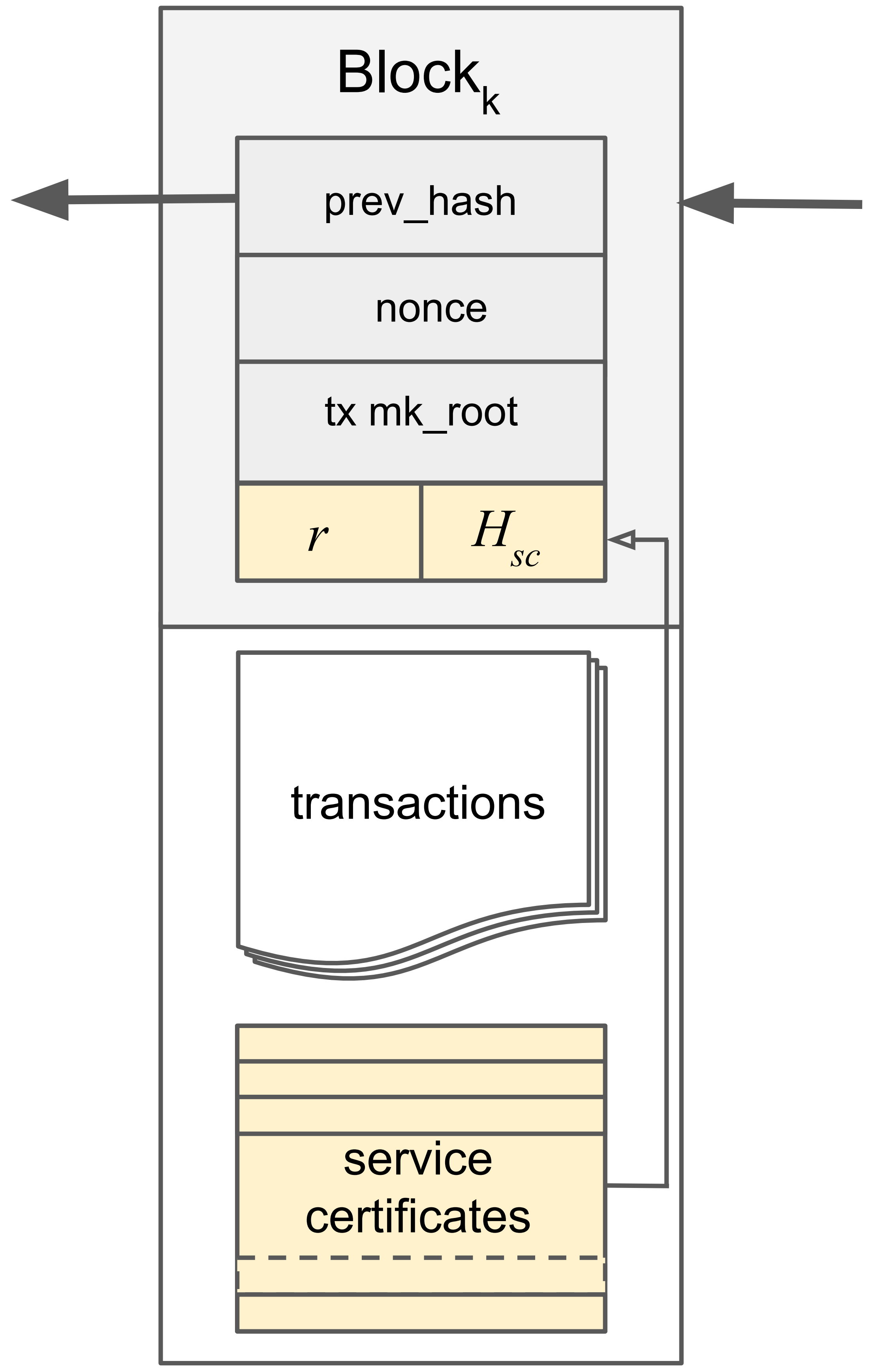
SUPPORT FOR POOLED MINING

Pooled Mining

- Pooled mining is a popular approach to reduce mining reward variance
- On the Theta blockchain, miners can also form pools. Similar to Bitcoin mining pools, miners solve the PoW mining puzzle on behalf of the master. In addition, their downstream viewers generate the service certificates for the master, by using the public key of the master as pki in the service certificate PoW puzzle
- **New pooled mining protocols**: reward split for a caching node is not just based on the number of "near-miss solutions" found for the mining puzzles, but also on the number of service certificates it collected

Challenges with Pooled Mining

- The size of service certificates for each block could potentially be large, since a pool can serve many viewers. This increases both bandwidth and storage overhead



SUPPORT FOR POOLED MINING

- In the next couple slides, we present several techniques to address the two challenges mentioned earlier

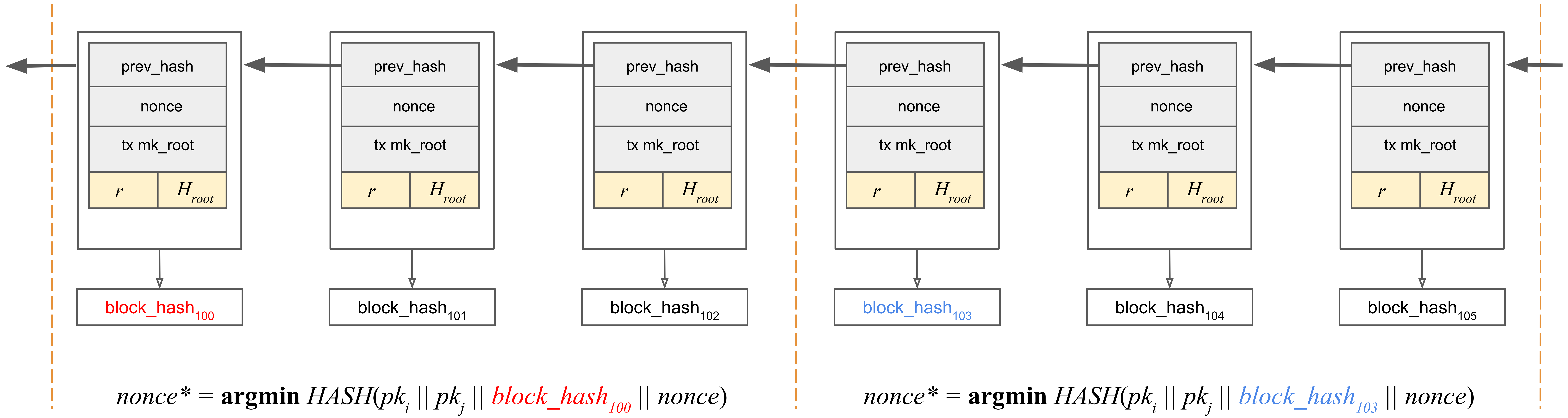
Technique #1: Limit the total number of service certificates per block

- Miners need to select the **best subset** of solutions to the service certificate PoW puzzles
- This puts a limit of the size of the services certificates for each block

SUPPORT FOR POOLED MINING

Technique #2: Less frequent service certificate PoW puzzle change

- No need to change the block_hash in the "service certificate PoW puzzle" each time a new block is added
  - We can use the same puzzle for d consecutive blocks. In the example below, $d = 3$
  - This reduces the total number of certificates given the same amount of service time



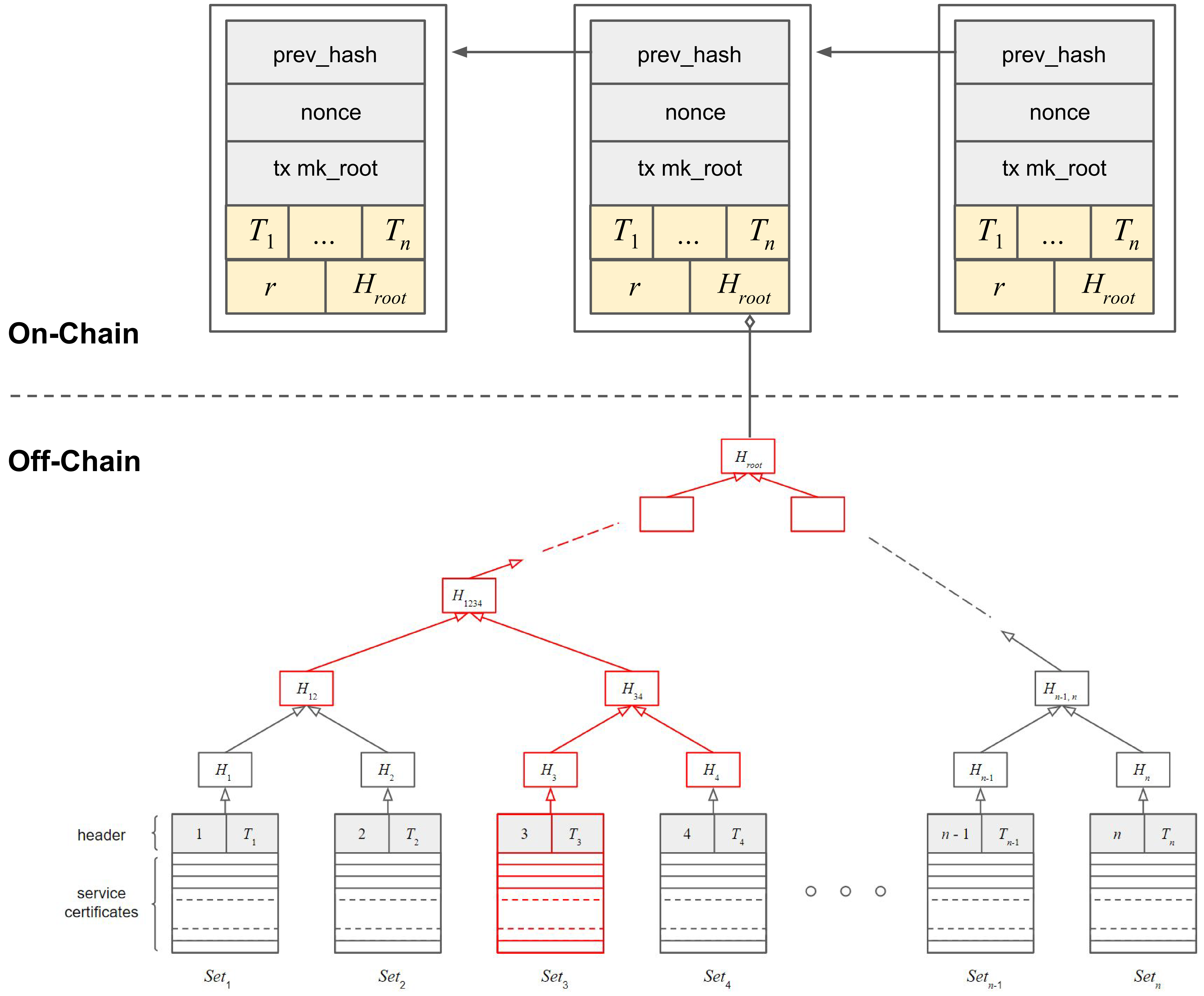
$$nonce^* = \textbf{argmin}\ HASH(pk_i \,||\, pk_j \,||\, block\_hash_{100} \,||\, nonce)$$

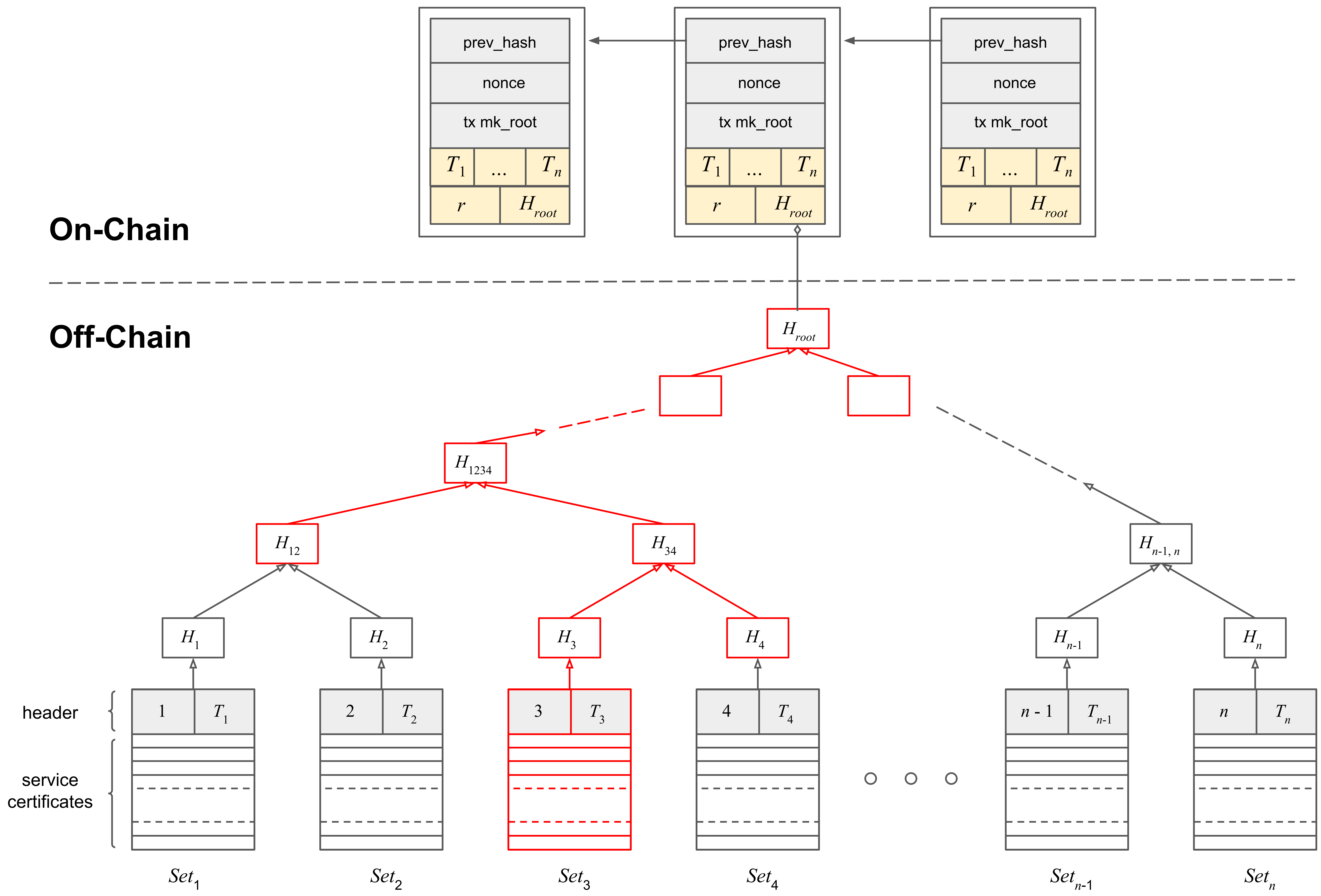
$$nonce^* = \textbf{argmin}\ HASH(pk_i \,||\, pk_j \,||\, block\_hash_{103} \,||\, nonce)$$

SUPPORT FOR POOLED MINING

Technique #3: Randomized reputation score validation protocol

- The service certificates are stored off-chain (either on the miner or decentralized storage systems)
- The service certificates are split into small sets (each 100K+ bytes), and organized into a Merkle tree. The header of each set contains the set index
- The block header contains the $T_1, T_2, \ldots, T_n$, the expected total viewing time for each set of service certificates

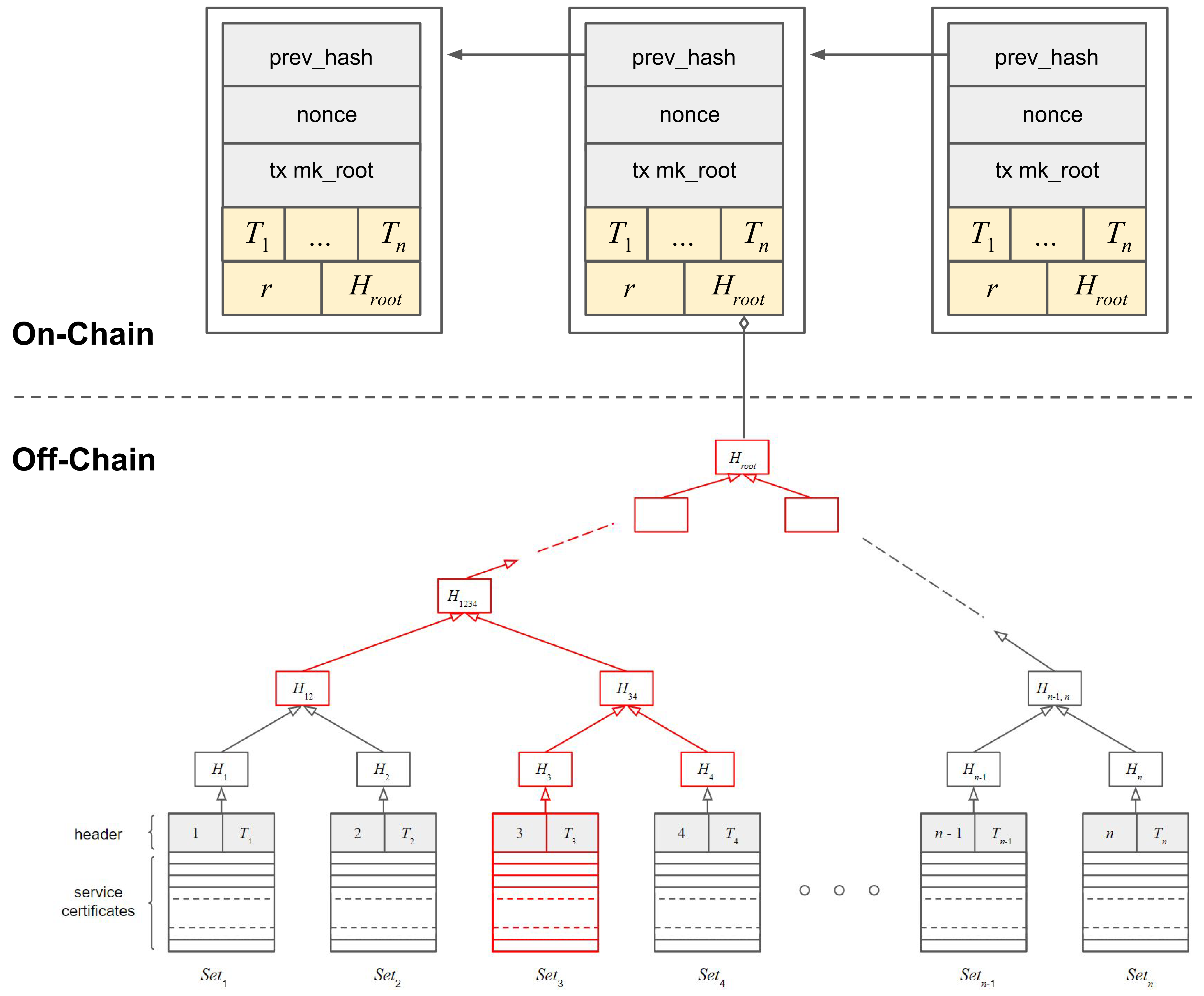



prev_hash
nonce
tx mk_root
T_1
…
T_n
r
H_{root}
On-Chain
Off-Chain
H_{root}
H_{1234}
H_{12}
H_{34}
$H_{n-1, n}$
H_1
H_2
H_3
H_4
H_{n-1}
H_n
header
service certificates
1
T_1
2
T_2
3
T_3
4
T_4
n - 1
T_{n-1}
n
T_n
Set_1
Set_2
Set_3
Set_4
Set_{n-1}
Set_n

SUPPORT FOR POOLED MINING

Technique #3: Randomized reputation score validation protocol

- Once a validator miner receives a new block, it **randomly** selects and downloads k sets of service certificates and their Merkle branches, and verifies the viewing time of each of these set matches with the corresponding watching time in the header
- If a validator miner finds a mismatch, it propagates the set of certificates throughout the network. Other miners can then validate this set and decide whether to reject the block
- The probability that the majority of miners accept an invalid block approaches zero as more miners join the network



FUTURE WORK

Compress the service certificates further to reduce bandwidth and storage overhead

- May be possible to generate a succinct non-interactive proof of the service certificates for the mining pools

Incorporate Proof-of-Stake to reduce energy waste

- Service certificate computation requires hashing power, and could lead to energy waste.

- We can design the block reward to be depending on the stake the caching node owns. Thus, instead of service certificate, a viewer can send **token guarantees** to the upstream caching node during the streaming session. The caching node returns the same amount of tokens to the viewer after the streaming session (enforced by a smart contract). Since the block reward depends on the caching node's stake, the viewer still gets the video stream for free, but the expected mining reward of the upstream caching node increases. This only requires a slight modification of the block reward formula

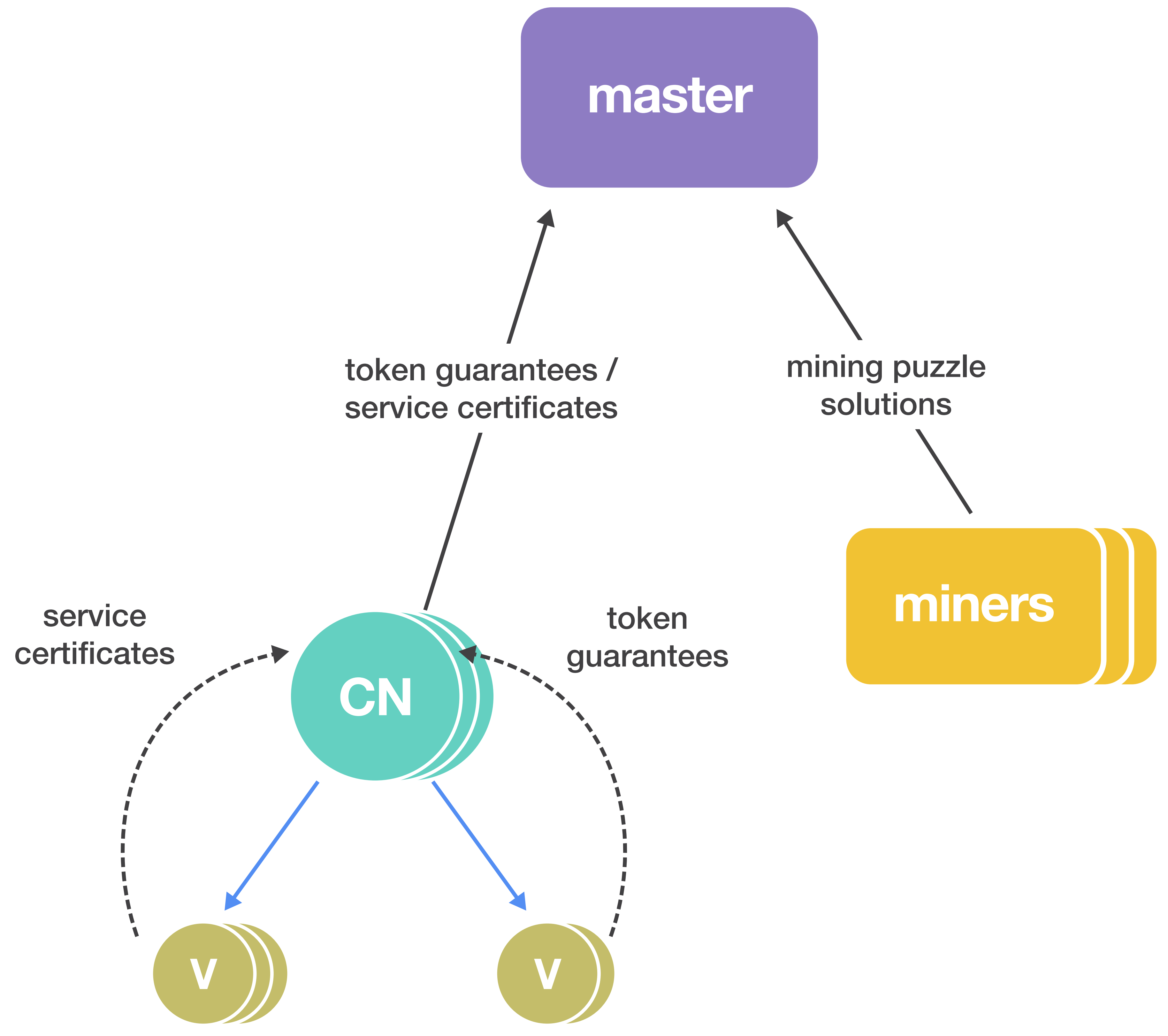
  - $reward = (1 + r + g(stake)) \cdot reward_{base}$
  - $g(stake)$ is a monotonically increasing function of the $stake$ (i.e. the total amount of tokens) the caching node owns

FUTURE WORK

Heterogeneous mining pool

Architecture

- **Master** manages the reward split. The master can be just a smart contract (similar to smartpool.io).
- **Miners** solve mining puzzles and assemble blocks. Their primary job is to secure the system. They do not relay video streams.
- **Caching nodes** relay video stream and collect service certificates and token guarantees. They don't need to assemble the blocks.
- **Viewers** can submit either service certificates or token guarantees in exchange for video streams.



Mining reward split

- A miner gets the mining reward split based on how many "near-miss solutions" it found.
- A caching node gets the split based on how many service certificates and token guarantees it collected

Benefits

- The caching nodes can get mining reward split without computing hashes to solve the mining puzzle. They are rewarded purely for the relay service.
- If a viewer chooses to send token guarantees instead of service certificates, he doesn't need to perform extra computation. This allows low-end devices (e.g. mobile phones) to benefit from the relay service.