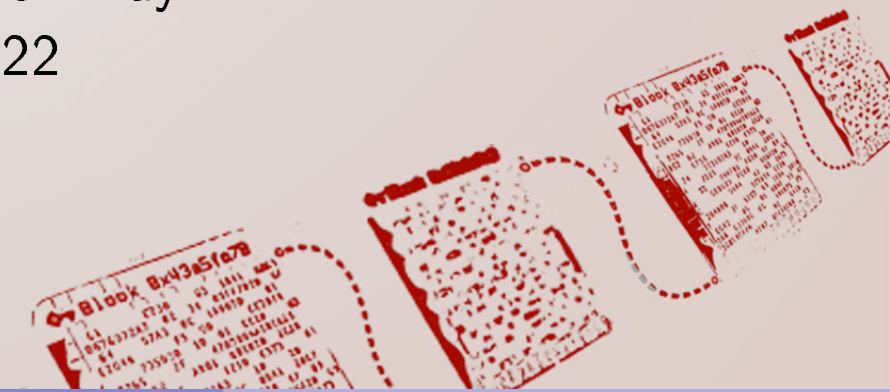
Proof-of-Useful-Work: BlockChain Mining by Solving Real-life Optimization Problems

Dušan Ramljak, Tatjana Davidović, Dragan Urošević, Tatjana Jakšić Krüger, Luka Matijević, Milan Todorović, Đorđe Jovanović

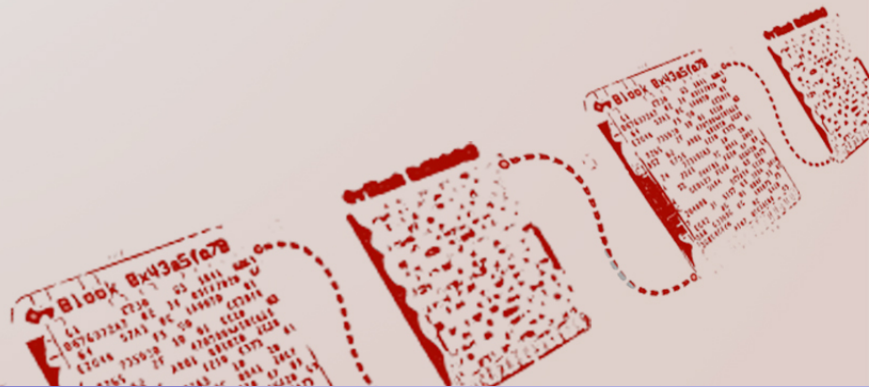
Mathematical Institute of the Serbian Academy of Sciences and Arts
Faculty of Technical Sciences

AI4TrustBC - Open Days
May 20, 2022



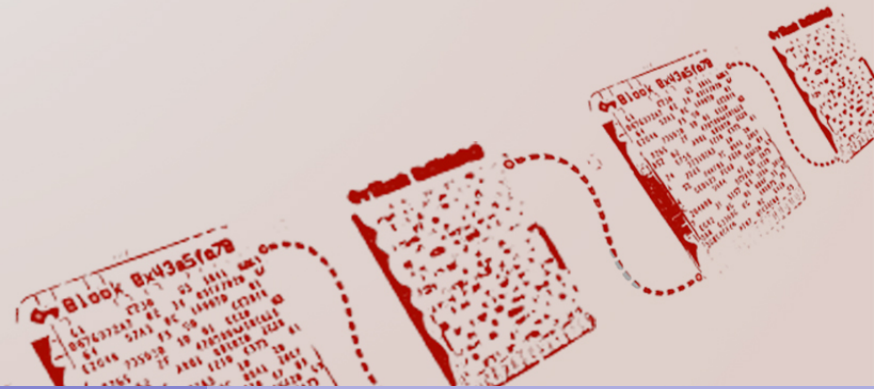
Overview

1. Blockchain background
2. Proposed PoUW consensus protocol
3. Benefits of proposed PoUW consensus protocol



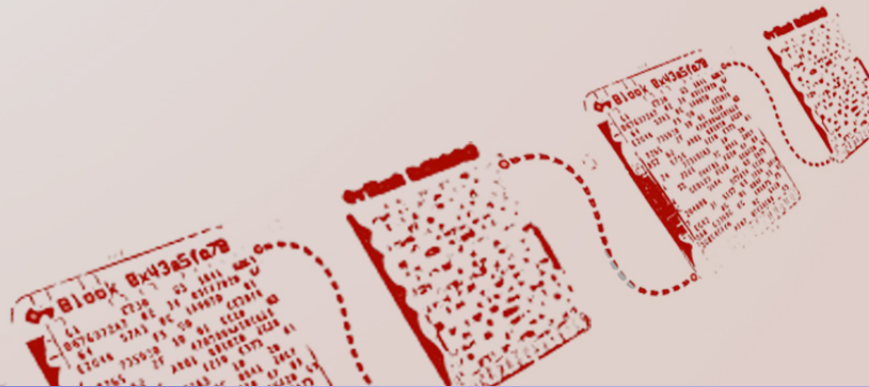
BC background

- Autonomous (unsupervised) append-only distributed data storage
- Autonomous: the removal of external authority (third party)
- Data is submitted in a form of (append-only) transactions (initially stored into Transaction pool)
- Consensus protocol controls transaction additions
- Distributed: each participant has a copy of the whole database
- Data is stored in form of blocks of transactions

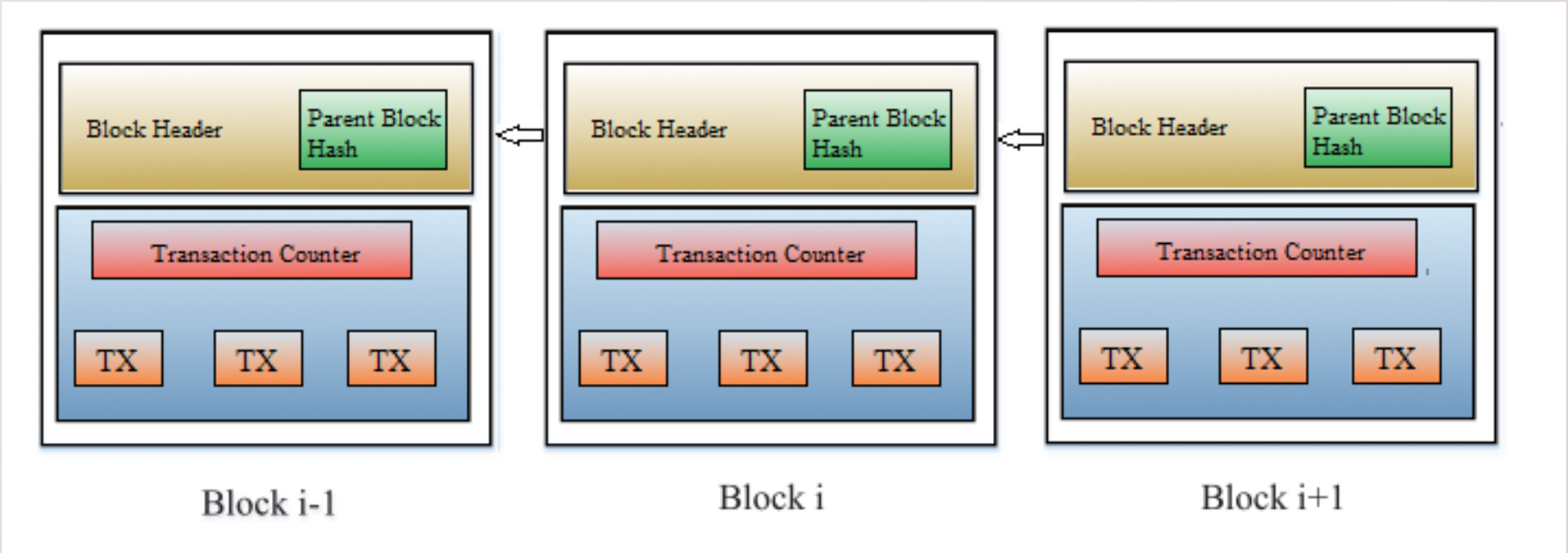


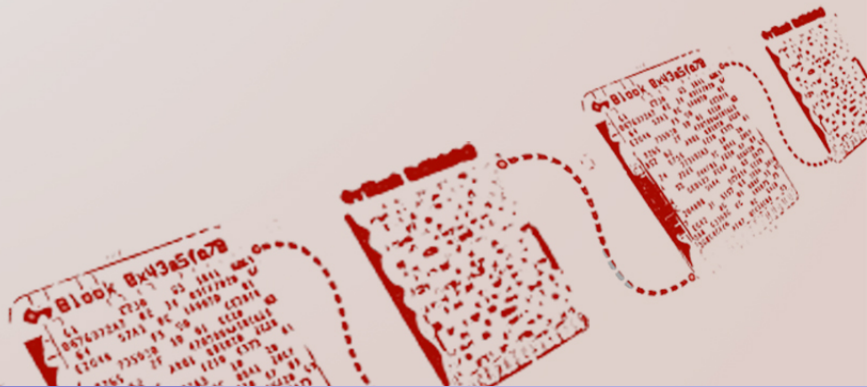
Block structure

- Data is stored in form of blocks of transactions
- Blocks consist of header and body
- Header includes:
  - Block number
  - Hash value of the previous block
  - Timestamp
  - mixHash
  - nonce (validity code), etc.
- Body contains transactions data



An illustration of BC

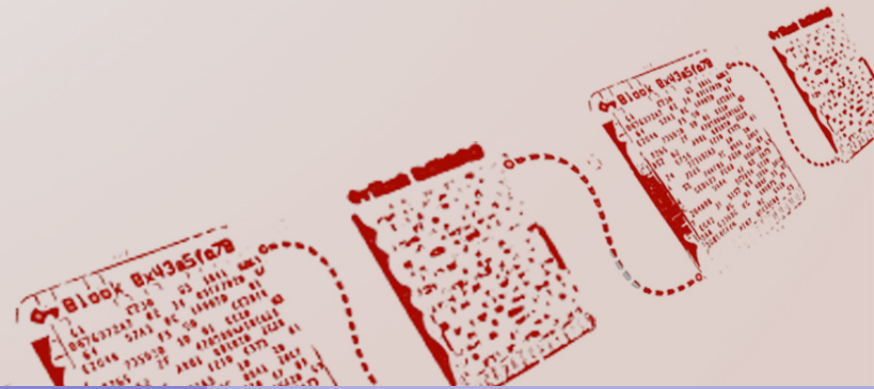




Classification of participants

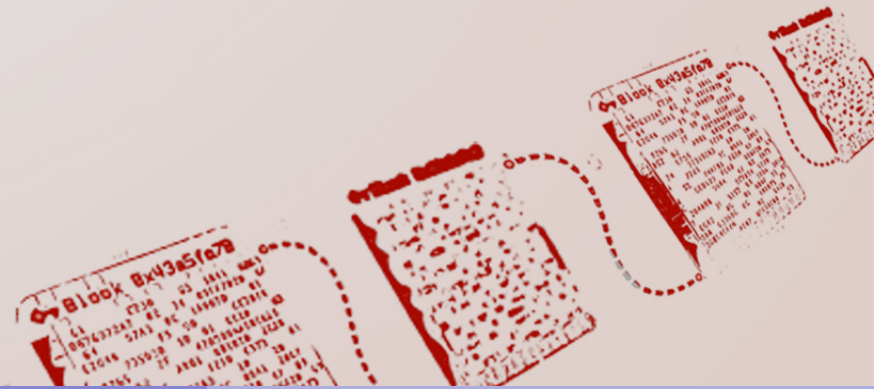
According to role in BC, participants can be divided into three groups:

- *basic users*: add new data by sending transactions to the *transaction pool*
- *verifiers*: users which perform verification of newly created blocks
- *miners*: whose task is to form new blocks of transactions that are in the transaction pool

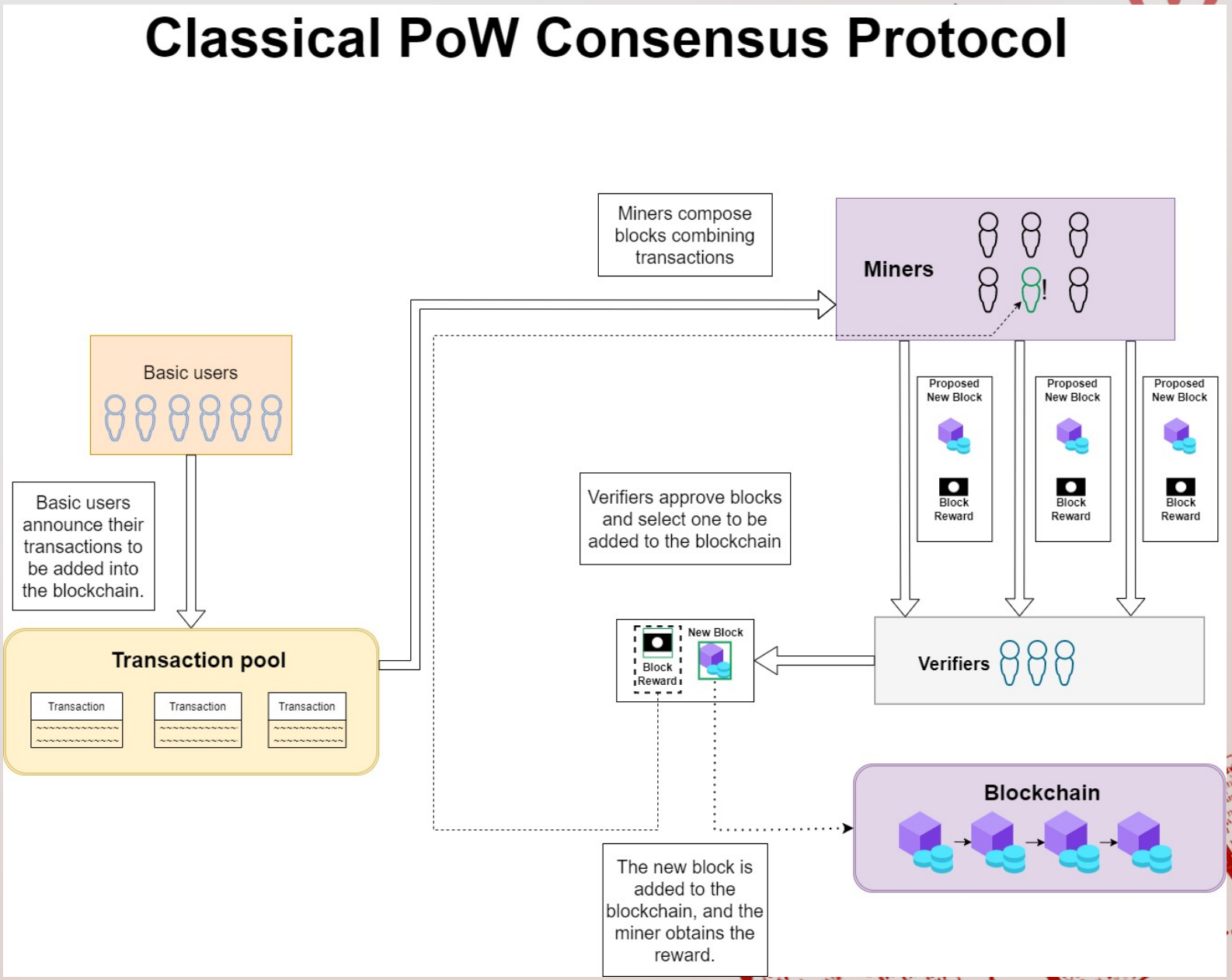


Procedure of appending a block

- Transactions are submitted by users to transaction pool
- Special users of the BC (miners):
  - Form a block by selecting transactions from the pool
  - Execute consensus protocol
  - Publish the new block
- Block validity is checked
- Valid blocks are appended to BC
- Potential forks are resolved periodically
- Successful miners receive reward



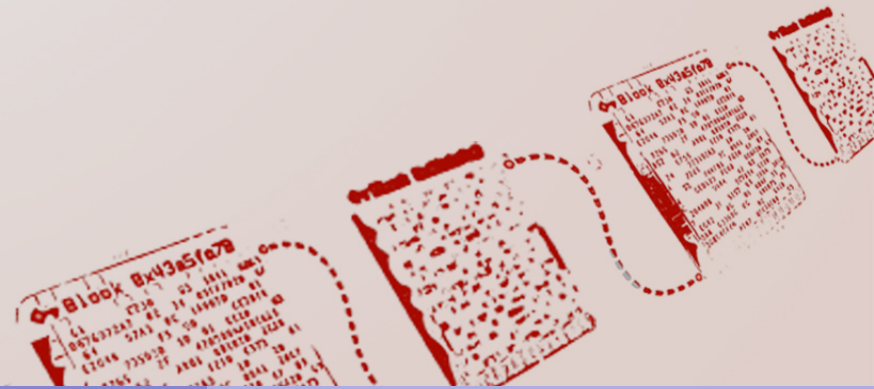
PoW illustration



BC extension

We extend classical BC by

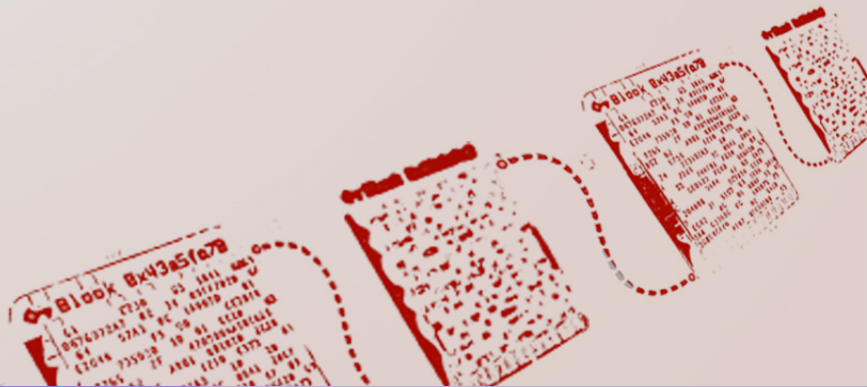
- introducing new type of participants: *customers* or *problem publishers*
- introducing *problem instance pool* for storing active (combinatorial optimization) problem instances
- introducing *Instance archive* which contains description of all solved instances as well as corresponding solution.



Problem instance details

Each problem instance has the following information:

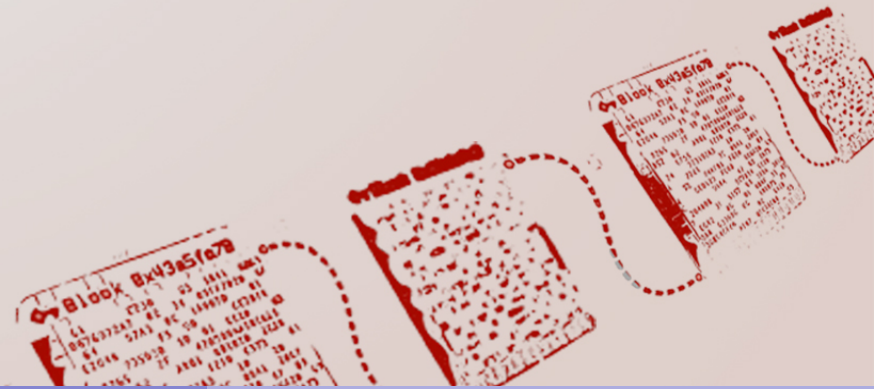
- Customer (problem publisher) identification
- Timestamp
- Valid address for problem data
- Hash value of problem data
- Solution threshold
- Deadline for finding the solution
- Reward specification



Publishing problem instances

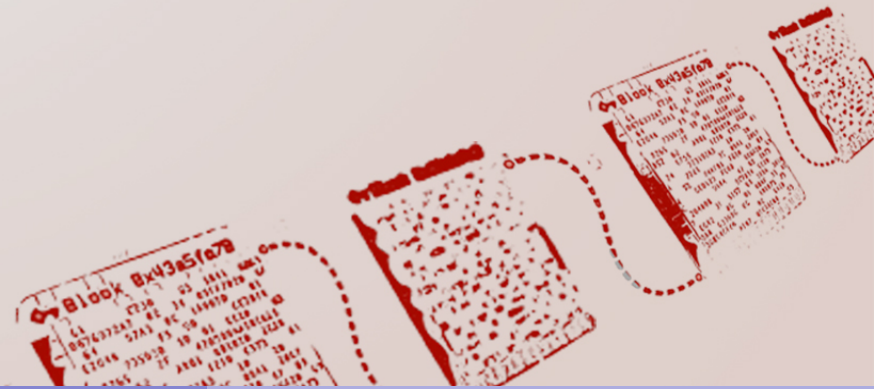
Problem instances can be published in two ways

- by publishing special kind of transactions containing all enumerated data (require introducing new type of transactions into existing BC system)
- by using smart contracts: problem instances pool contains a set of special smart contracts, at the address known to all participants; instances can be accessed by interacting with the contract.



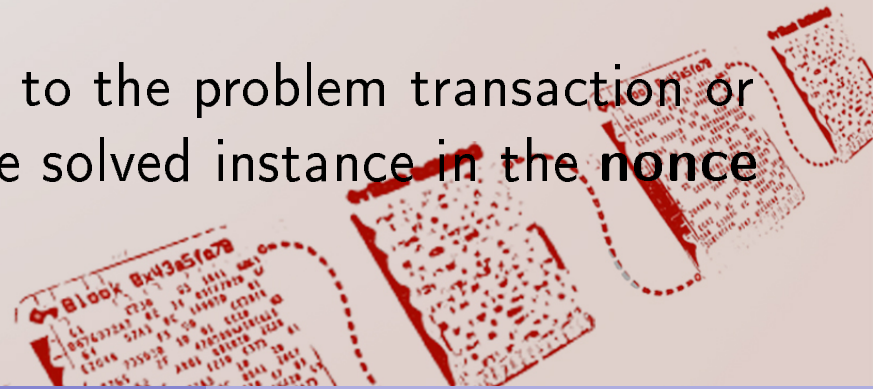
Block mining

- We assume that the active problem instances in the pool (P) are enumerated (starting with 0) according to same predefined way (for example by increasing order of their deadlines).
- When a block B to be mined is formed, the hash value, h_B, of the block's header ((without **mixHash** and **nonce** values)) and the corresponding (selected) transactions is calculated,
- The index i of the problem corresponding to block B is calculated as $i = h_B \bmod |P|$.

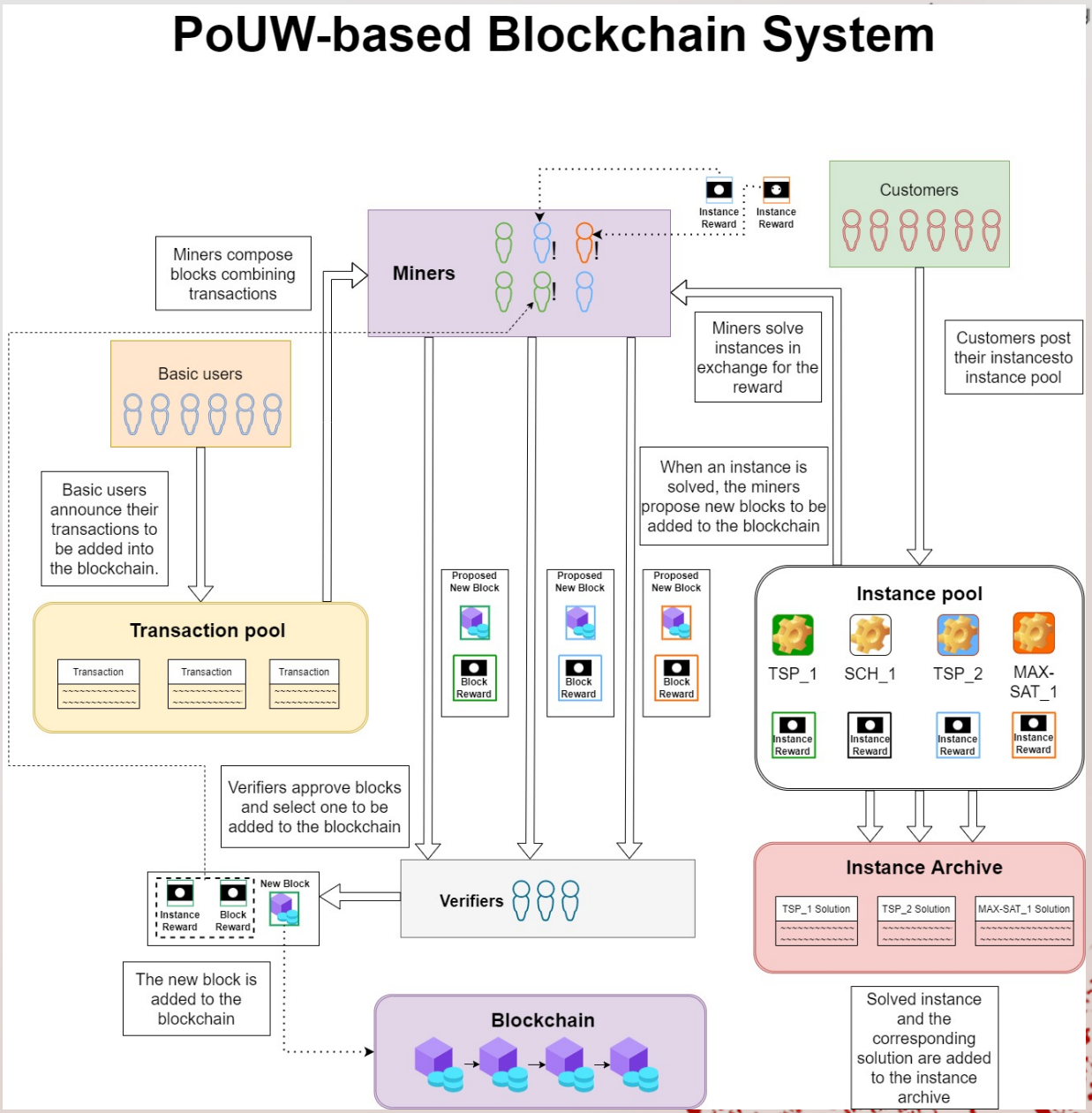


Announcing new block

- For security reasons and to save the space, solutions are stored in miner's private location, predefined and known to all participants.
- Published block must contain the hash value of the instance data and the hash value of the solution in order to prevent malicious behavior (e.g. publishing a block, and finding a solution later).
- In order to maintain compatibility with Ethereum systems, the only fields that the miner can use for storing these hash values are 64-bit **nonce** and 256-bit **mixHash** fields.
- The miner performs the **bit-wise xor** operation on problem instance hash and solution hash values and stores it in **mixHash** field in the block's header.
- In addition, the miner stores a pointer to the problem transaction or smart contract that corresponds to the solved instance in the **nonce** field.

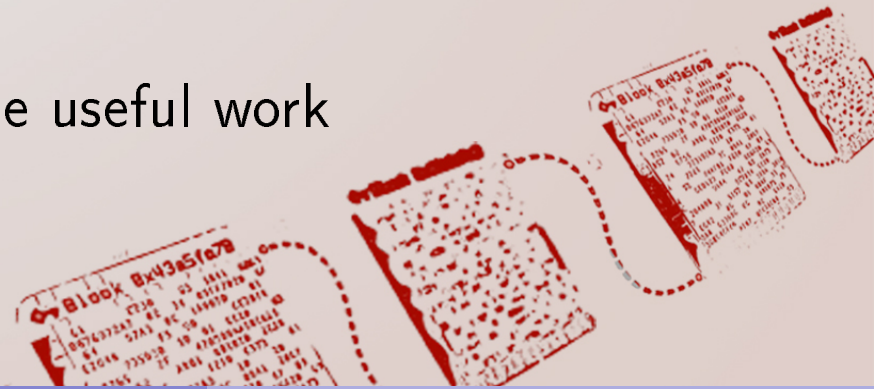


PoUW illustration



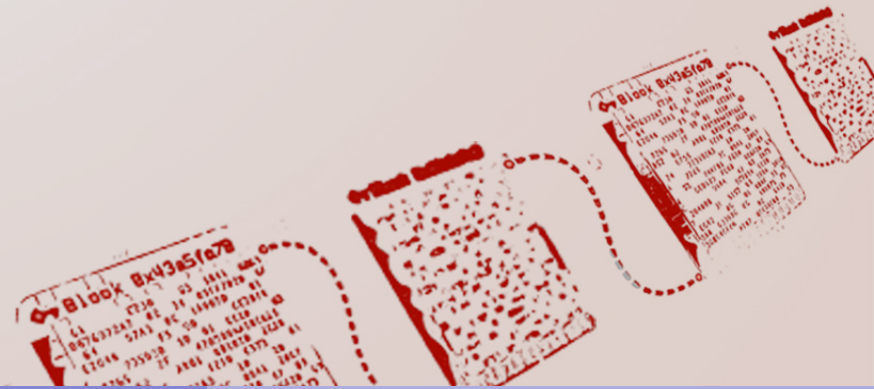
Advantages (Gains) of PoUW

- Reducing the amount of energy unnecessary spent for block mining
- Participants who have fewer computing resources obtain the new motivation to participate in the mining process.
- Proposed PoUW consensus protocol introduces new initiatives for participants to join the network as problem publishers who benefit from having their problems solved.
- Miners who add a new block are doubly rewarded: by problem publisher for solving the problem and, as before, for adding a block.
- Miners who solve the problem, but don't add blocks can still be rewarded by problem publishers.
- Moreover, resources are spent for some useful work



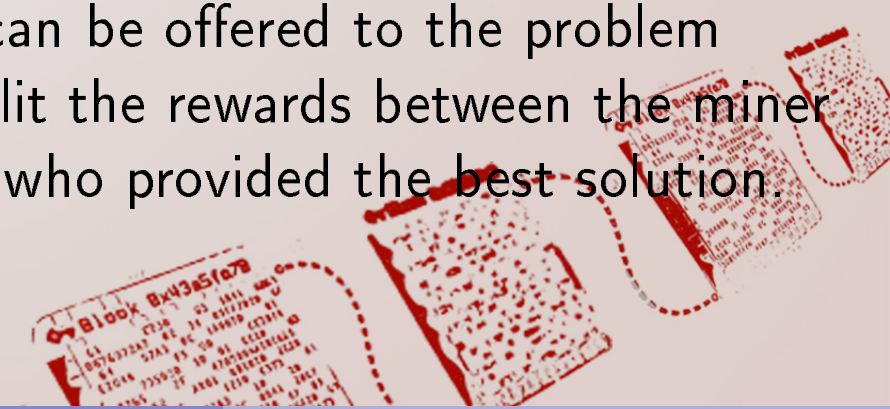
Examples of optimization problems

- Scheduling problems
  - Scheduling independent tasks to identical processors (P||Cmax)
  - Weighted scheduling problem with deadlines and release times
- Asymmetric vehicle routing problem
- Maximum satisfiability problem
- Community detection problem



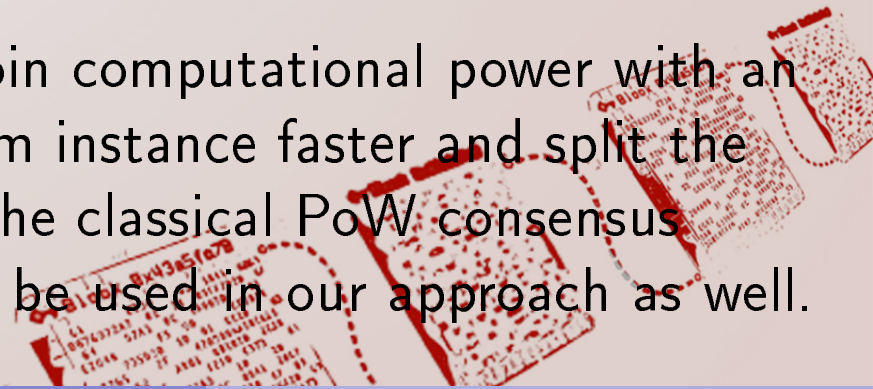
Some of possible scenarios

- **Scenario 1.** Each miner creates their own block, calculates the hash value of the block and selects the appropriate problem instance from the pool. The fastest miner who solves the corresponding problem instance publishes the created block.
- **Scenario 2.** It can happen that two or more miners compose the same block and solve the corresponding problem instance almost simultaneously (approximately at the same time). Each of them is publishing the created block, however, only one can be accepted for inclusion into the BC network. On the other hand, several potentially different solutions are available for the corresponding problem instances and the best among them can be offered to the problem publisher. In that case, it is fair to split the rewards between the miner whose block is accepted and the one who provided the best solution.



Some of possible scenarios

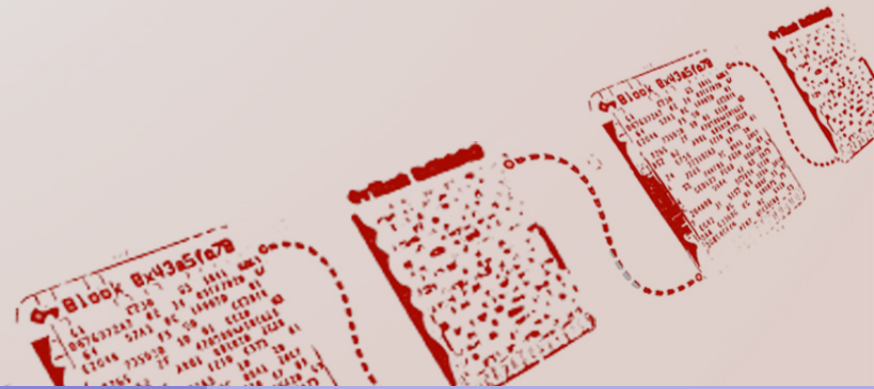
- **Scenario 3.** In the case when two or more miners create several different blocks, various problem instances may correspond to these blocks. Consequently, it may happen that for some of them the solution is obtained approximately at the same time. Again, only one block can be included into BC, but solutions for all solved instances may be offered to the corresponding problem publishers. In this case the miner who added a block is given the appropriate reward, as well as the reward for solving the corresponding problem instance. In addition, each miner who provides a solution for one of the instances from the pool is also rewarded. So, the computer power usually spent only for adding a single block is now engaged to obtain solutions for one or more problem instances.
- **Scenario 4.** A group of miners can join computational power with an aim to solve the corresponding problem instance faster and split the reword. This is common situation in the classical PoW consensus protocols and we believe that it could be used in our approach as well.



Instead of conclusion

The new concept require also solving the following details

- Formal description of problem instance
- Procedure for determining problem instance corresponding to block created by miner
- Resources (hardware and software) for solving corresponding problem instance
- Possibility of grouping of miners in order to solve corresponding problem faster (more efficiently)



Фонд за науку
Републике Србије

Thank you on your attention.