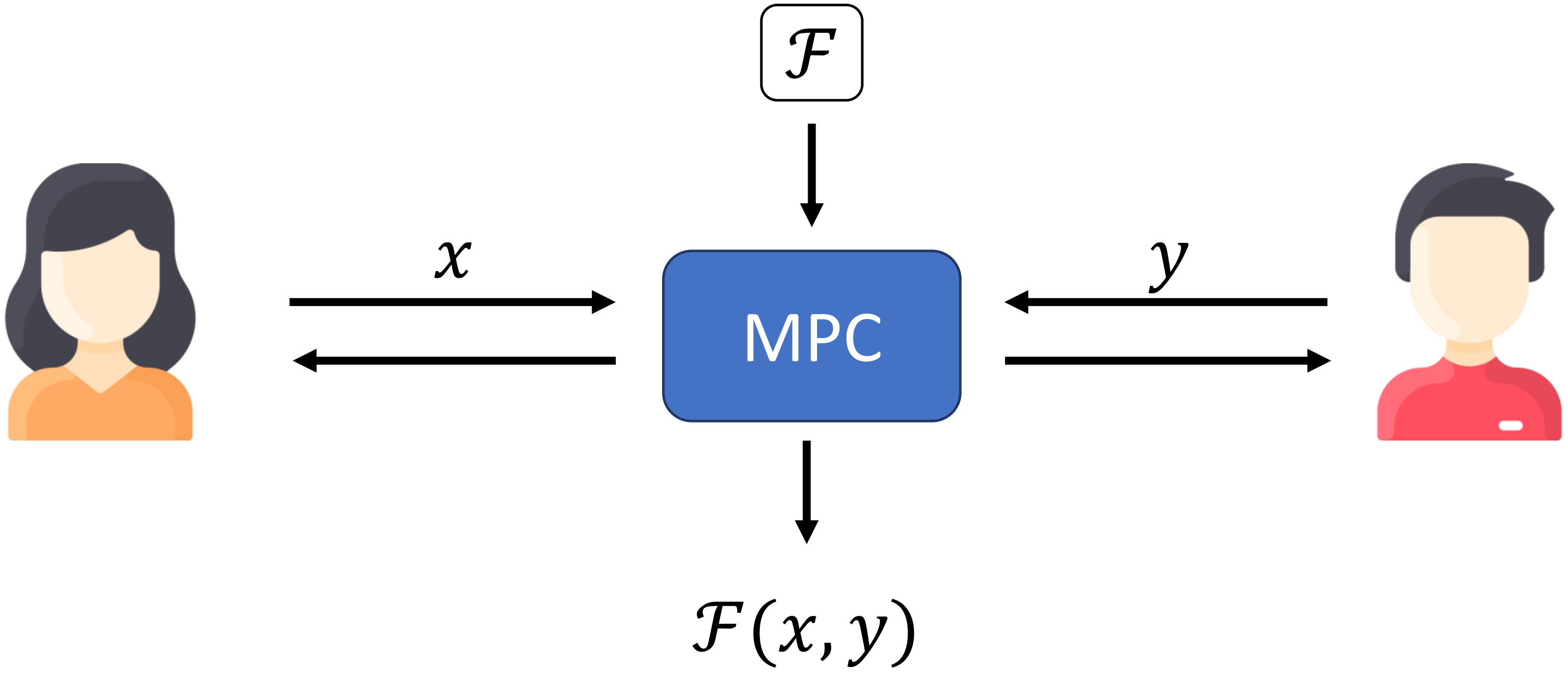
Financially Backed Covert Security

Sebastian Faust[1], Carmit Hazay[2], David Kretzler[1], and **Benjamin Schlosser**[1]

1) TECHNISCHE UNIVERSITÄT DARMSTADT

2) Bar-Ilan University
אוניברסיטת בר אילן

Multi-Party Computation



- security guarantees: correctness & privacy

Adversary Models

Semi-honest adversary:

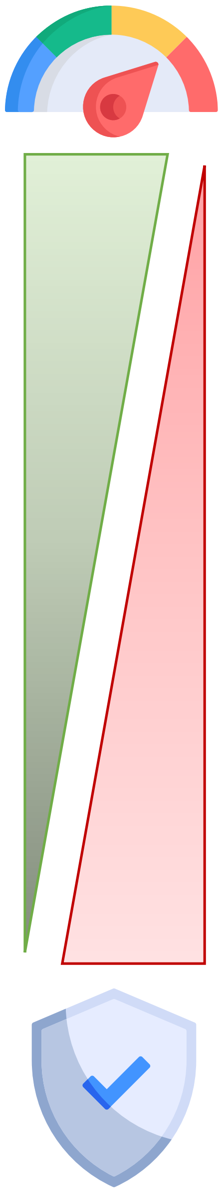
- follows protocol description

Covert adversary [AL07]:

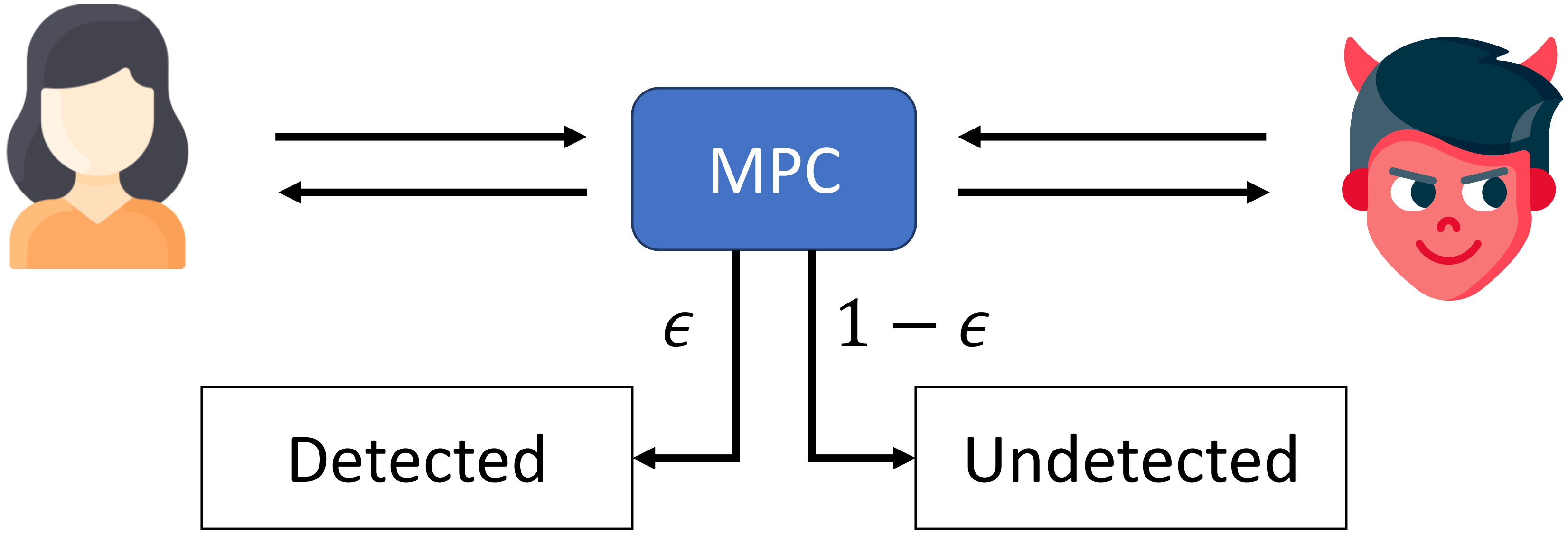
- willing to cheat only if they are not caught

Malicious adversary:

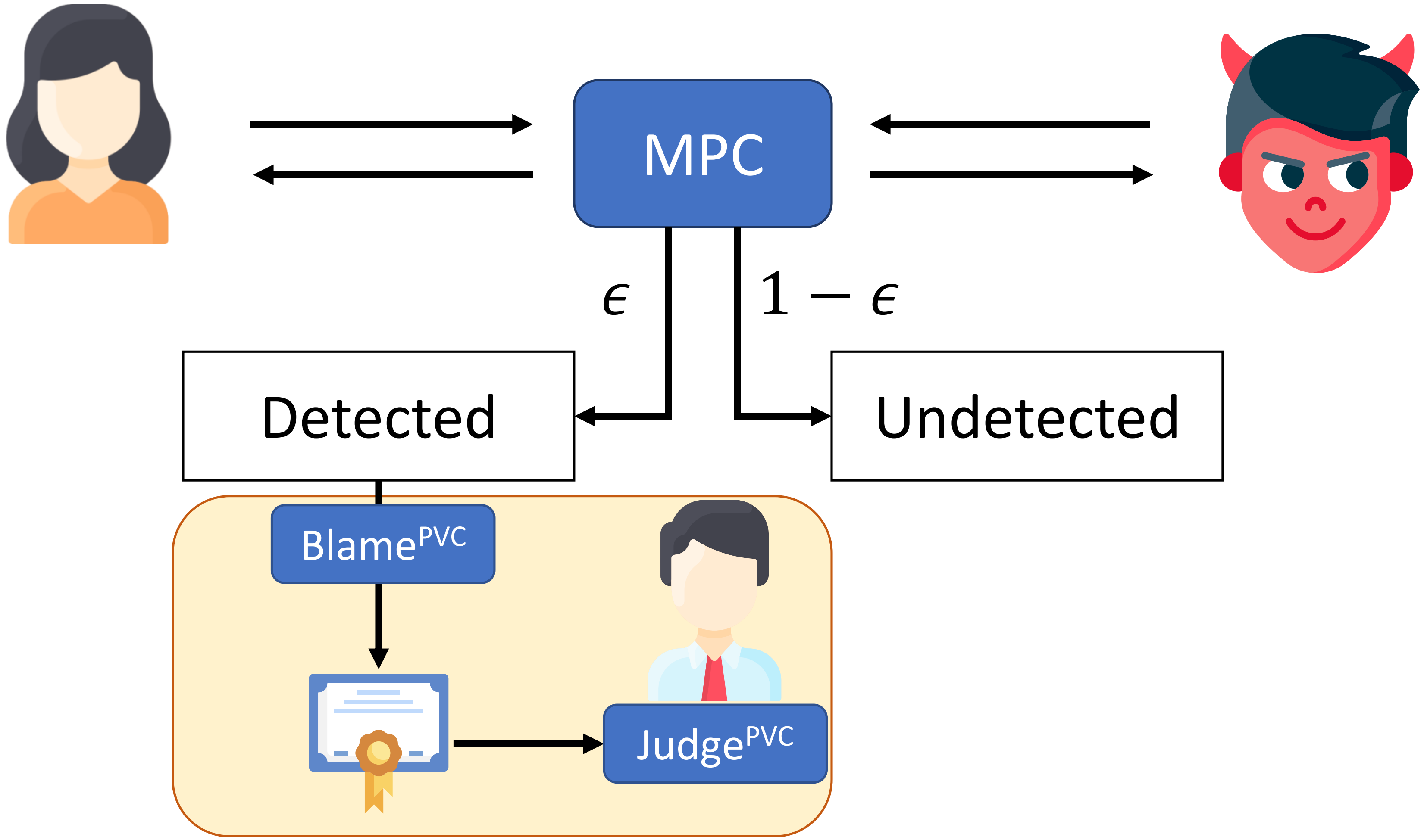
- behaves arbitrarily



Covert Security



Publicly Verifiable Covert Security (PVC) [AO12]



Shortcomings of PVC

Intention of PVC:

- increase deterrent effect if every party can verify misbehavior

Problem:

- party can hide behind digital identity in Internet-like settings (e.g. IP addresses)

Our goal:

- add financial punishment if cheating was detected

Contribution

1. Definition

- new notion financially backed covert security (FBC)

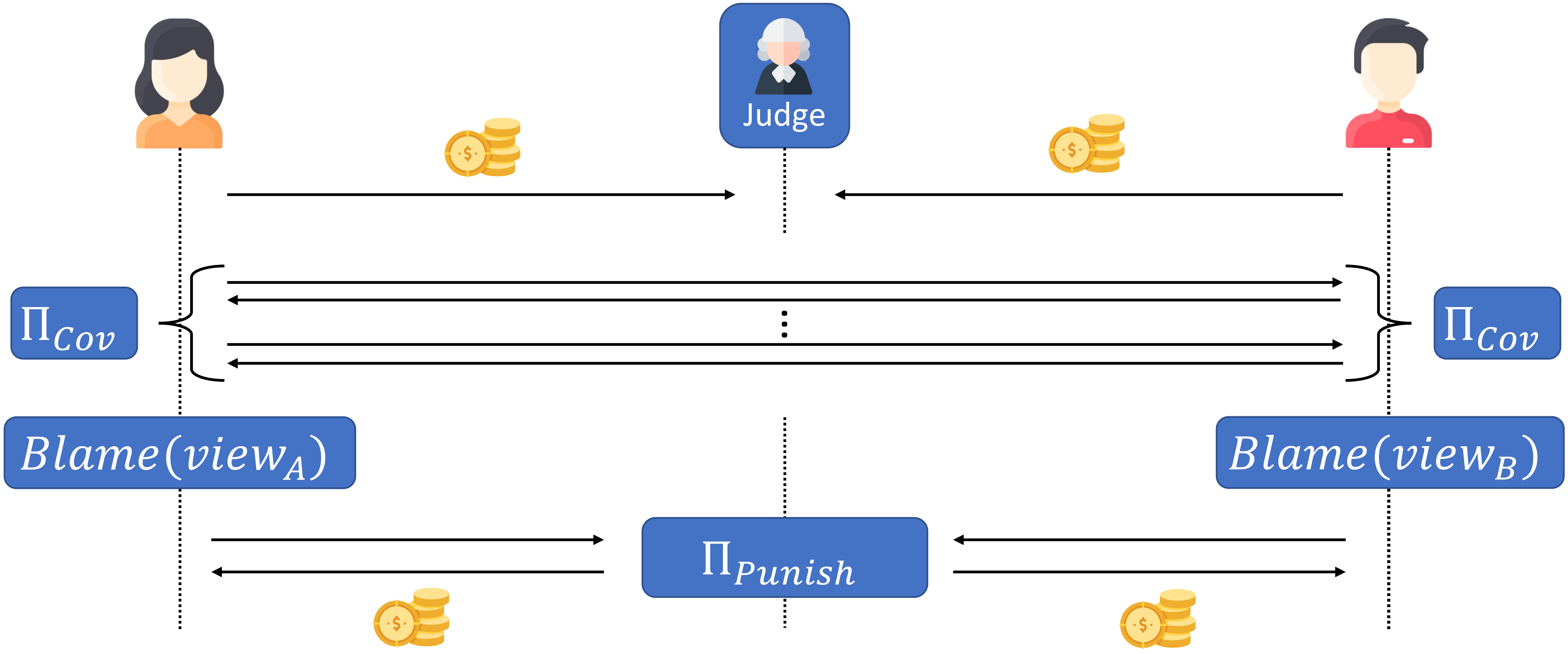
2. Constructions of FBC protocols

- FBC protocols for three classes of protocols
- efficient verification of misbehavior

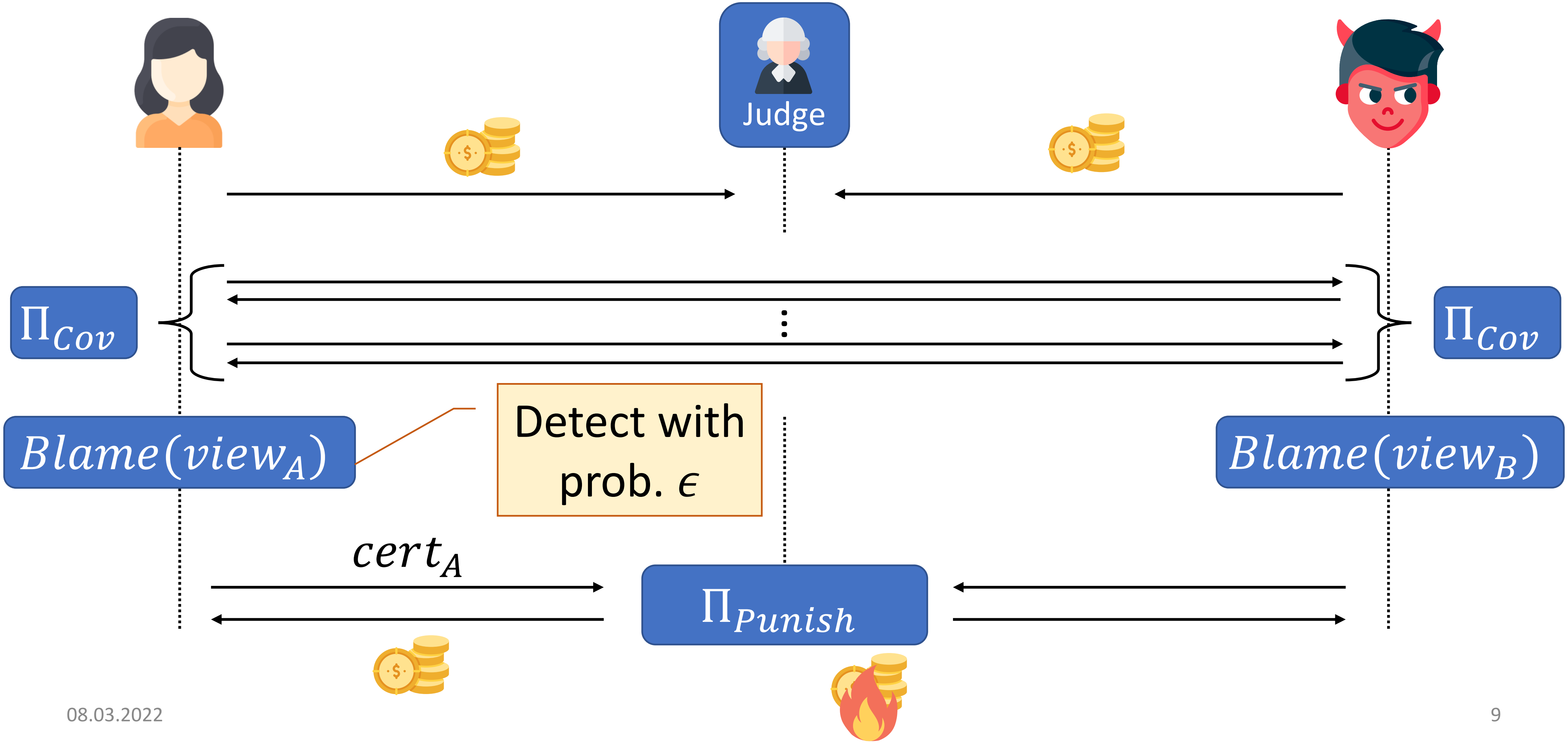
3. Evaluation

- benchmarking our constructions

Financially Backed Covert Security



Financially Backed Covert Security - Malicious



FBC Security Guarantees

Financial accountability:

- If any **honest** party detects cheating, then there exists a **corrupted** party that loses its deposits.

Financial defamation freeness:

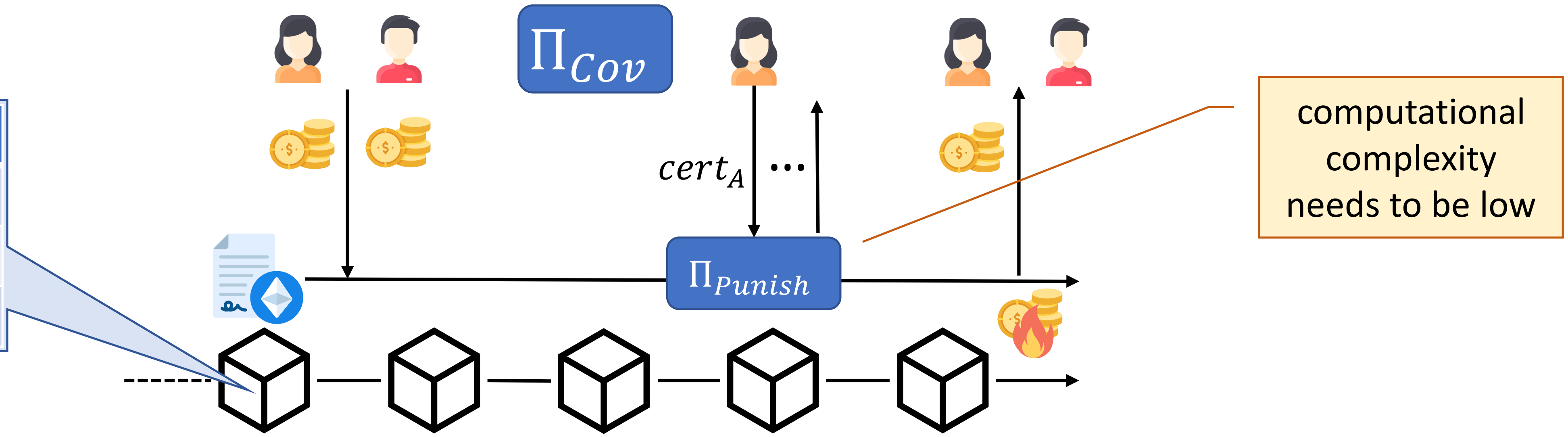
- No **corrupted** party can force any **honest** party to lose its deposits.

We present formal security games for both properties in our paper!

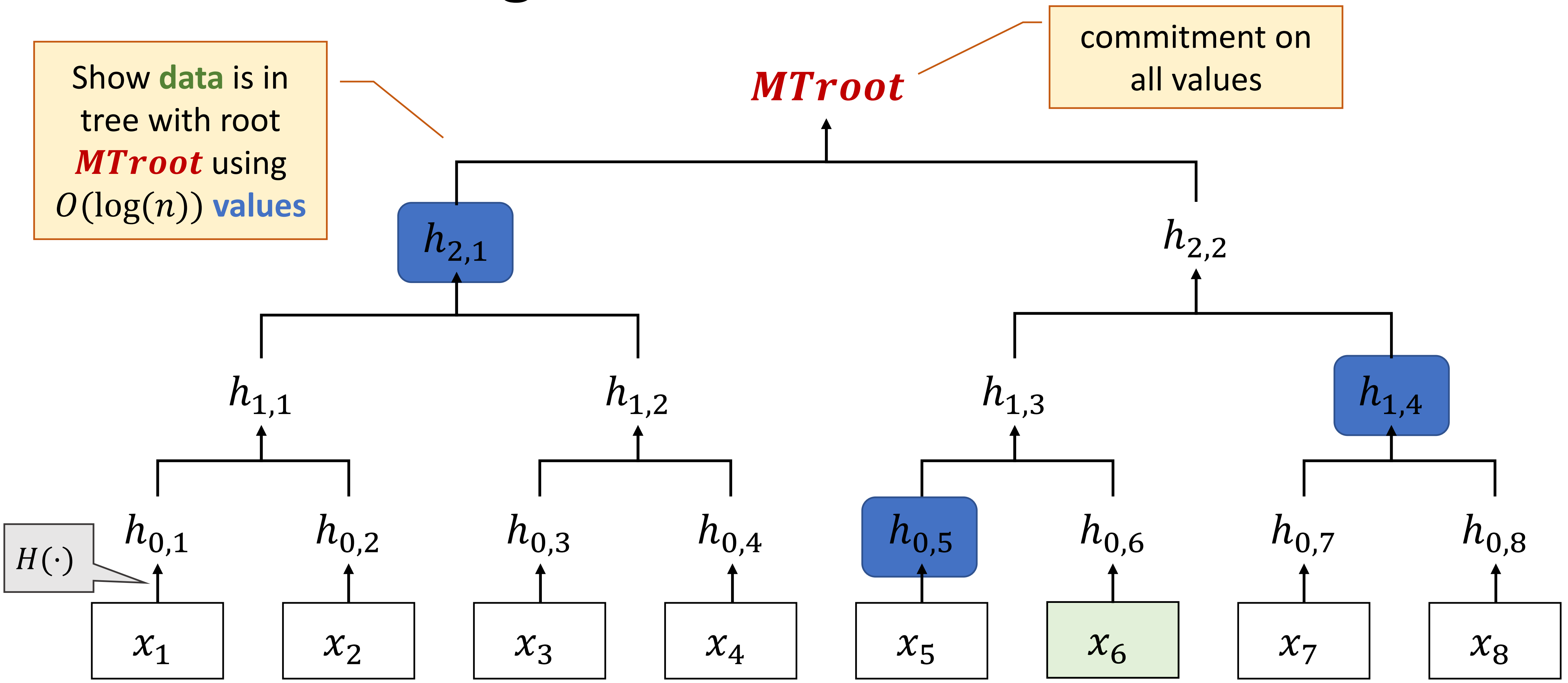
How to instantiate the Judge?

- Blockchain technologies provide a convenient way to handle money
- Smart Contracts are programs that enable transfer of assets based on predefined rules

Person	Balance
A	10
B	12
⋮	



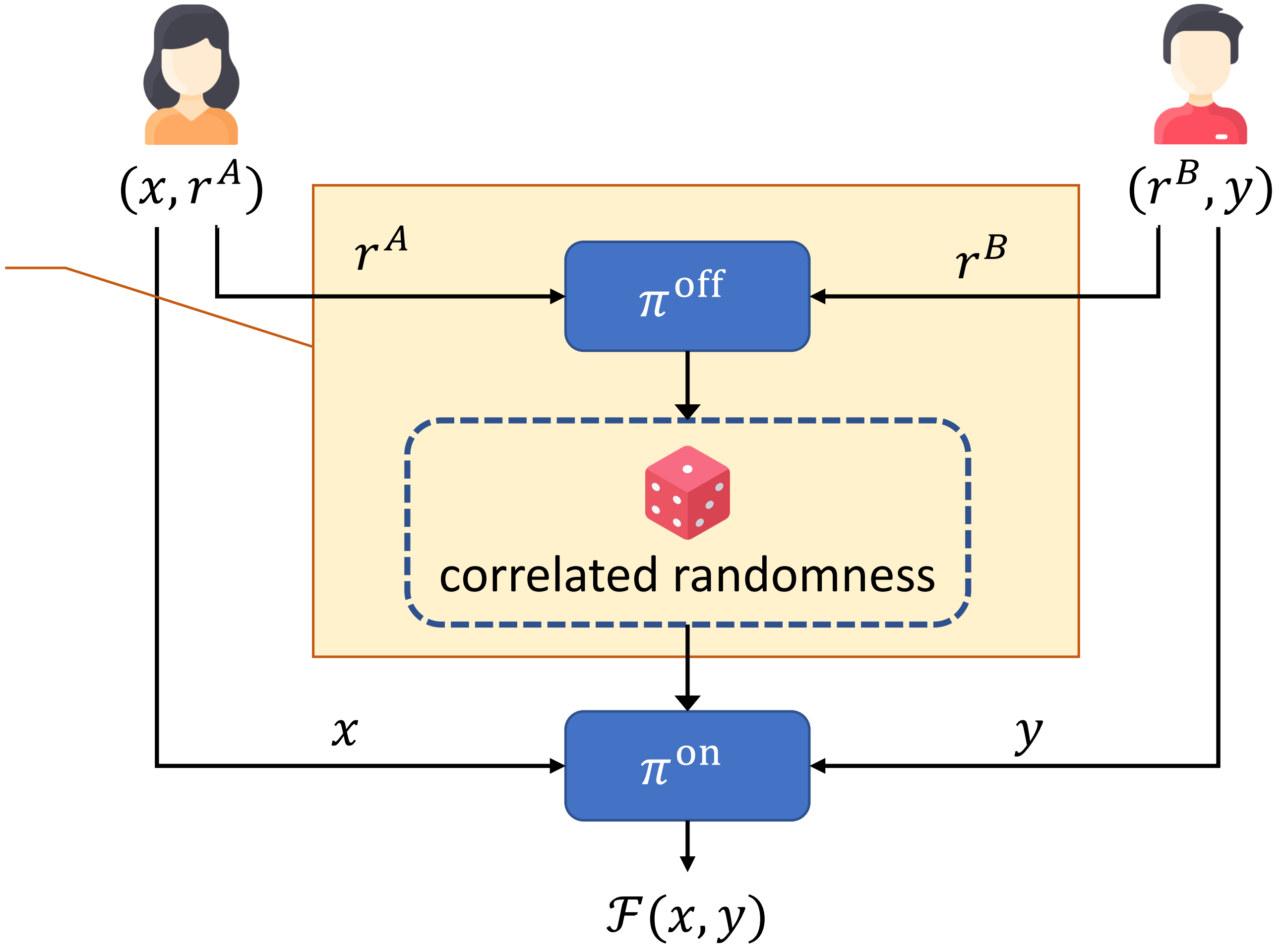
Main Building Block: Merkle Tree



Construction 1

Input-independent protocol, e.g. offline phase of SPDZ, authenticated garbling

offline phase is **independent** of secret input

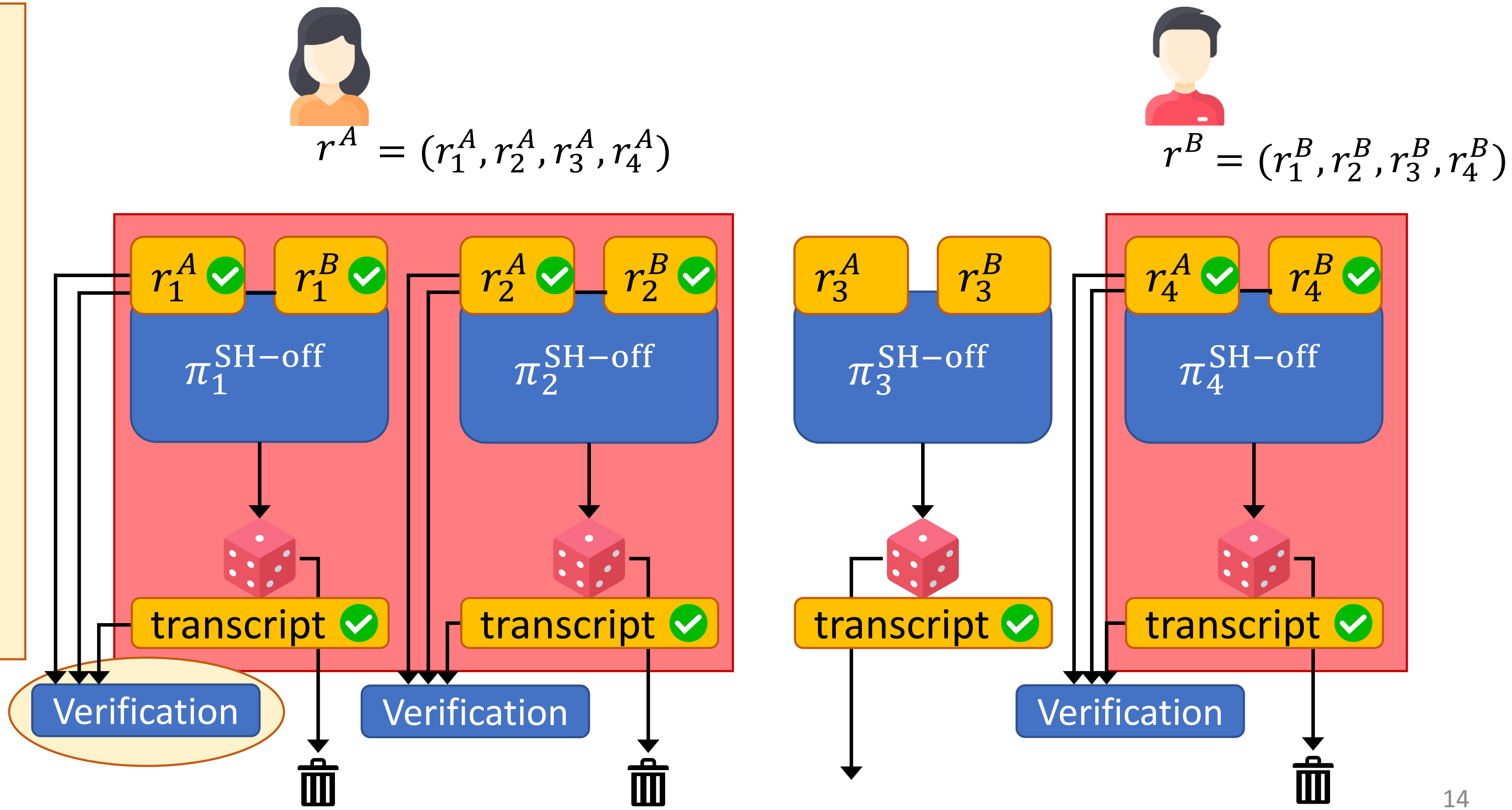


Construction 1 – Starting Point

Key features:

1. cut-and-choose
2. deterministic behavior
3. public transcript
4. publicly verifiable initial states

➢ provided by all known input-independent PVC protocols



Detour: PVC in a Nutshell – Verification

$state_0^A \coloneqq r_1^A$

$(state_1^A, msg_1^{(A,B)}) \coloneqq compRound(state_0^A, \emptyset)$

deterministic function

$msg_1^{(A,B)}$ →

← $msg_1^{(B,A)}$

$(state_2^A, msg_2^{(A,B)}) \coloneqq compRound(state_1^A, msg_1^{(B,A)})$

⋮

Verification:

- given $state_0^A$ and all messages received by A
- recompute all messages sent by A
- compare recomputed with real messages

Why not using PVC?

- most known PVC protocols require the third party to recompute the whole protocol

➢ not plausible for smart contracts

Construction 1 - Intermediate Evidence

message to receiver

hashes of messages to other parties

hash of intermediate state

$state_0^A := r_1^A$

$msg_1^{(A,B)}, \left\{H\left(msg_1^{(A,X)}\right)\right\}_{X\in\mathcal{P}}, H\left(state_1^A\right)$ →

$\left(state_1^A, \left\{msg_1^{(A,X)}\right\}_{X\in\mathcal{P}}\right)$
$:= compRound\left(state_0^A, \emptyset\right)$

← $msg_1^{(B,A)}, \left\{H\left(msg_1^{(B,X)}\right)\right\}_{X\in\mathcal{P}}, H\left(state_1^B\right)$

⋮

after the last round:

$MTroot(msgHashes), MTroot(states)$ *Alice* →

messages and states are ordered in a unique way

Construction 1 - Blame

Bob knows:

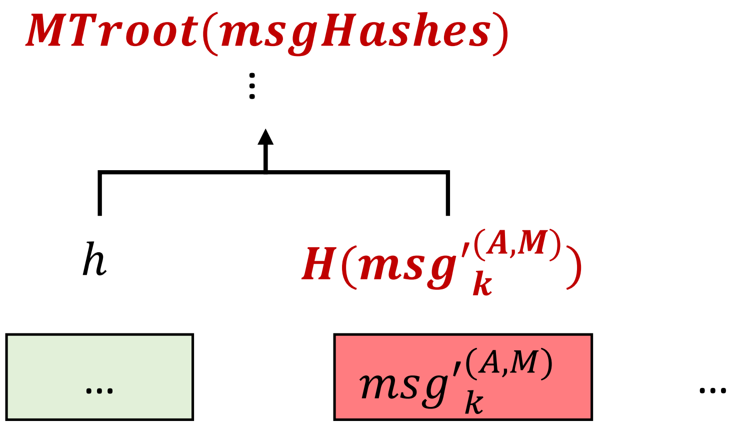
- $\forall i \in Rounds$: $msg_i^{(A,B)}, \left\{ \boldsymbol{H}\left(\boldsymbol{msg_i^{(A,X)}} \right) \right\}_{X \in \mathcal{P}}, \boldsymbol{H}(\boldsymbol{state_i^A})$
- publicly verifiable $\boldsymbol{MTroot(msgHashes), MTroot(states)}$ $\mathscr{Alice}$
- publicly verifiable $state_0^A$

Bob recomputes:

- all messages that should have been sent by Alice
- intermediate states of Alice

If malicious behavior detected:

- e.g. incorrect message sent from Alice to party M in round k



Construction 1 - Punish

non-interactive

$cert$ →

more efficient Judge-algorithm than most existing PVC

Bob provides $cert$ containing:

- $\boldsymbol{MTroot(msgHashes)}$ Alice
- $\boldsymbol{MTroot(states)}$ Alice
- state of previous round $state_{k-1}^{A}$
- message of previous rounds $\left\{msk_{k-1}^{(X,A)}\right\}_{X\in\mathcal{P}}$
- incorrect message $\boldsymbol{msg'}_{\boldsymbol{k}}^{\boldsymbol{(A,M)}}$

together with Merkel proofs

Judge executes a single step:

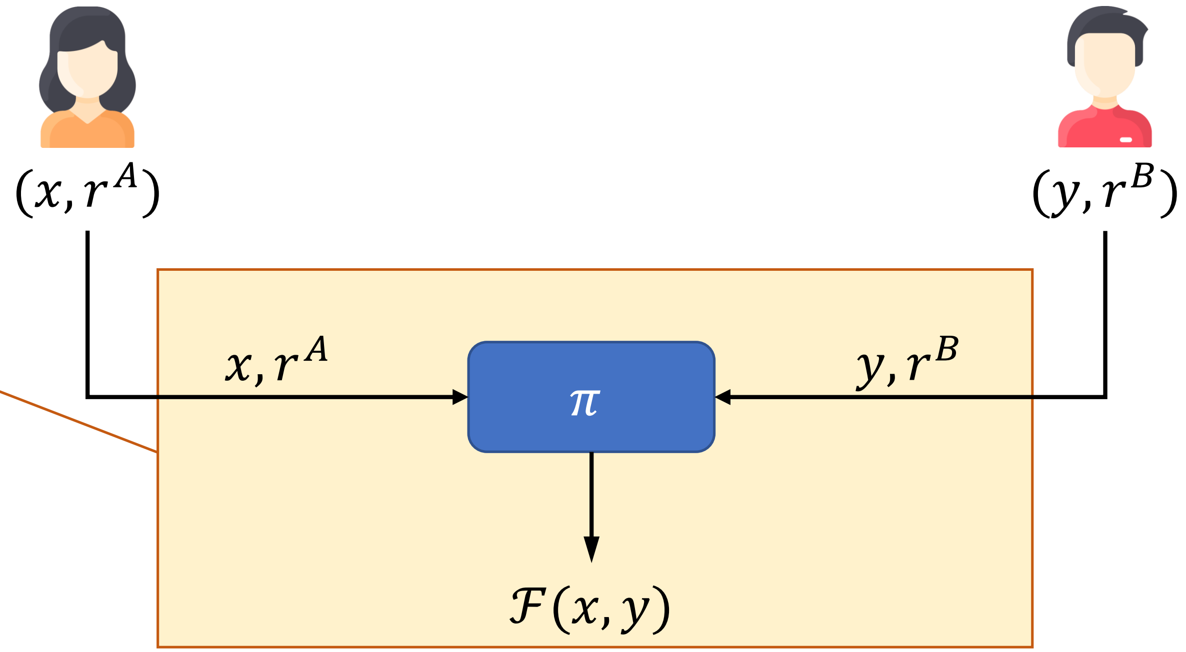
$$\left(state_k^A, \left\{\boldsymbol{msk_k^{(A,X)}}\right\}_{X\in\mathcal{P}}\right)$$
$$\coloneqq compRound\left(state_{k-1}^A, \left\{msk_{k-1}^{(X,A)}\right\}_{X\in\mathcal{P}}\right)$$

Final check:

$$\boldsymbol{msg'}_{\boldsymbol{k}}^{\boldsymbol{(A,M)}} \neq \boldsymbol{msg_k^{(A,M)}}$$

Construction 2

Input-dependent protocol



privacy of secret inputs needs to be guaranteed

see our paper for a detailed description

All PVC protocols as well as our construction 1 and 2 require consensus about the protocol transcript.

Can we relax on this requirement?
I.e., can we construct FBC without public transcript?

we exploit interactivity of π_{Punish}

this would reduce communication cost in the honest execution

Yes, for input-independent protocols.

Construction 3 – Starting Point

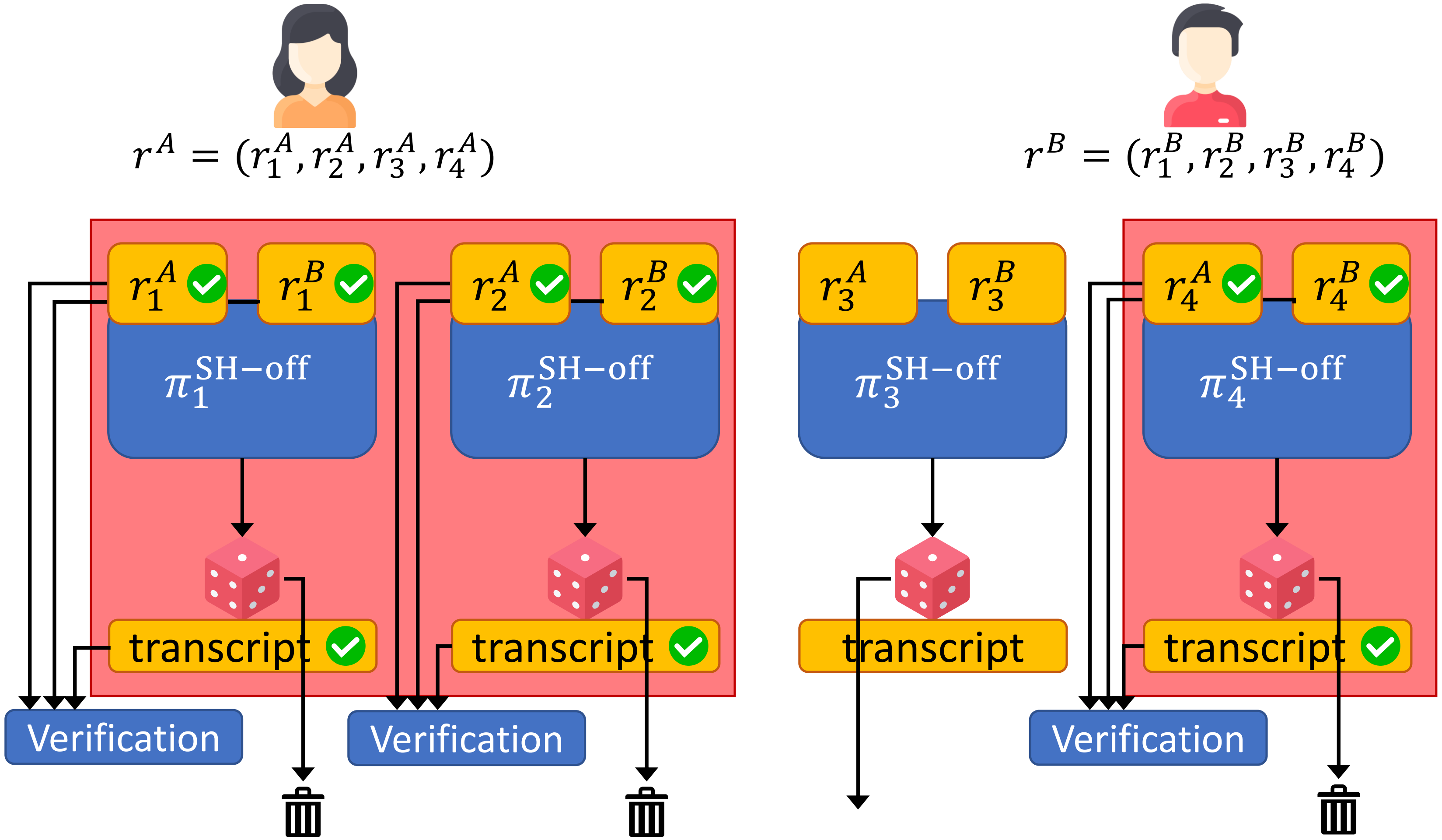
Key features:

1. cut-and-choose
2. deterministic behavior

No public transcript

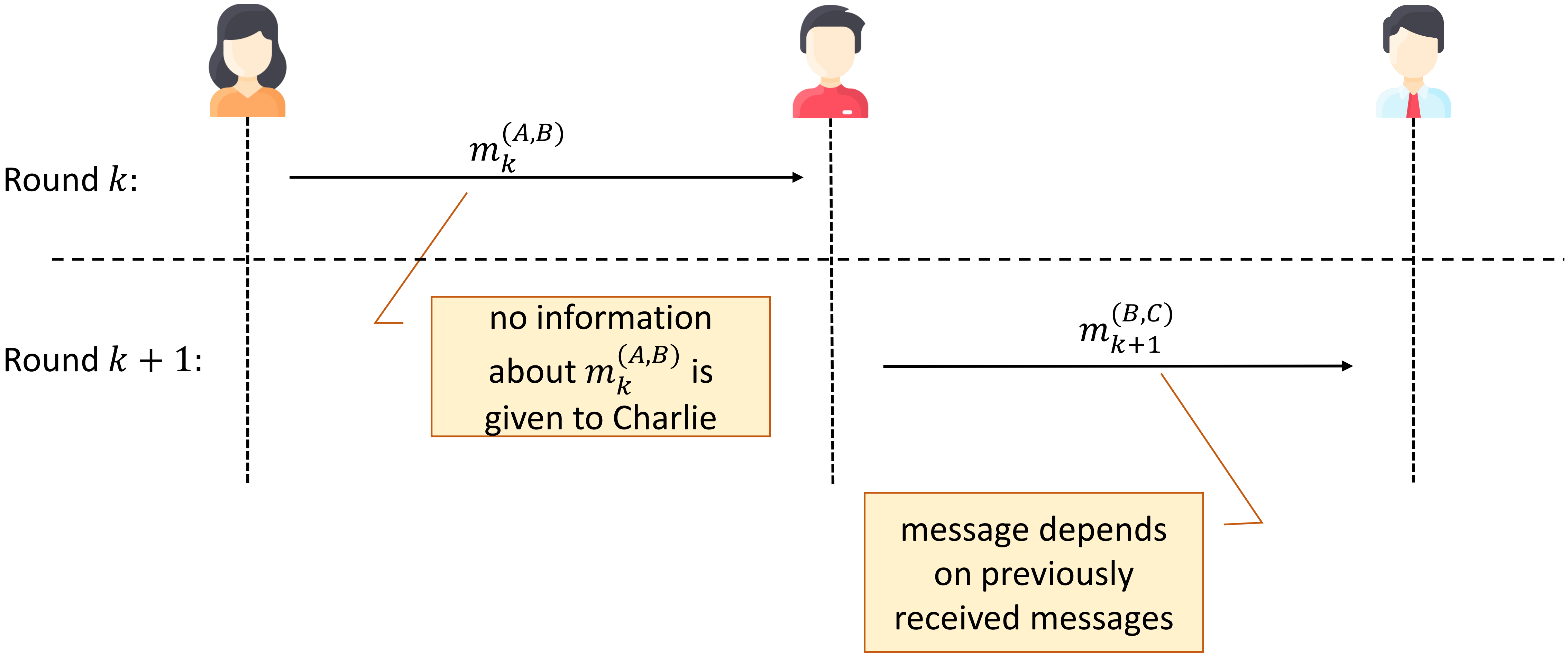
4. publicly verifiable initial states

➢ from PVC by removing public transcript



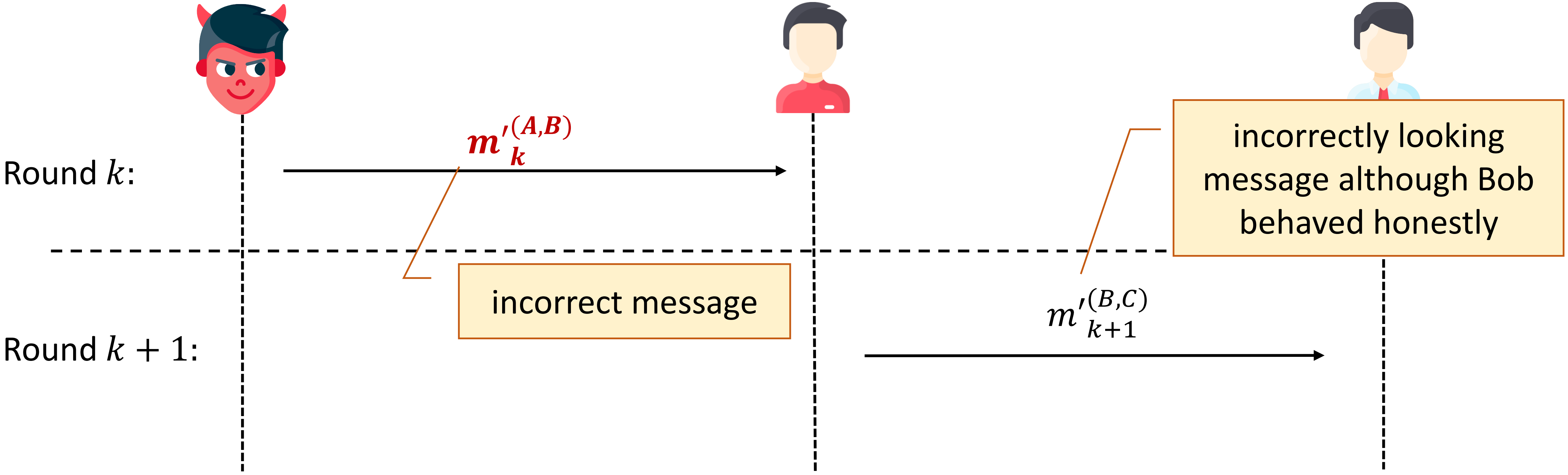
Construction 3 – Challenge

Honest execution



Construction 3 – Challenge

Malicious execution



Two-phase
Punish-protocol

Bisection if disagreement
about message history

What’s more

1. **Security proofs**

2. **Single gate verification**

- Judge needs to recompute only a single gate of an arithmetic circuit

3. **Evaluation**

- Solidity smart contract implementation
- gas cost measurements for efficiency evaluation

Conclusion

Advantages of FBC over PVC:

- **effectiveness of deterrence**: detected cheating is directly financially punished
- **computation cost of judge**: reduced from whole protocol re-execution to single step/gate validation
- **communication cost in honest execution**: relaxing on requirement of public transcript

Thank you for your attention!
Any questions?

David Kretzler: david.kretzler@tu-darmstadt.de
Benjamin Schlosser: benjamin.schlosser@tu-darmstadt.de