Detecting MEV Vulnerabilities

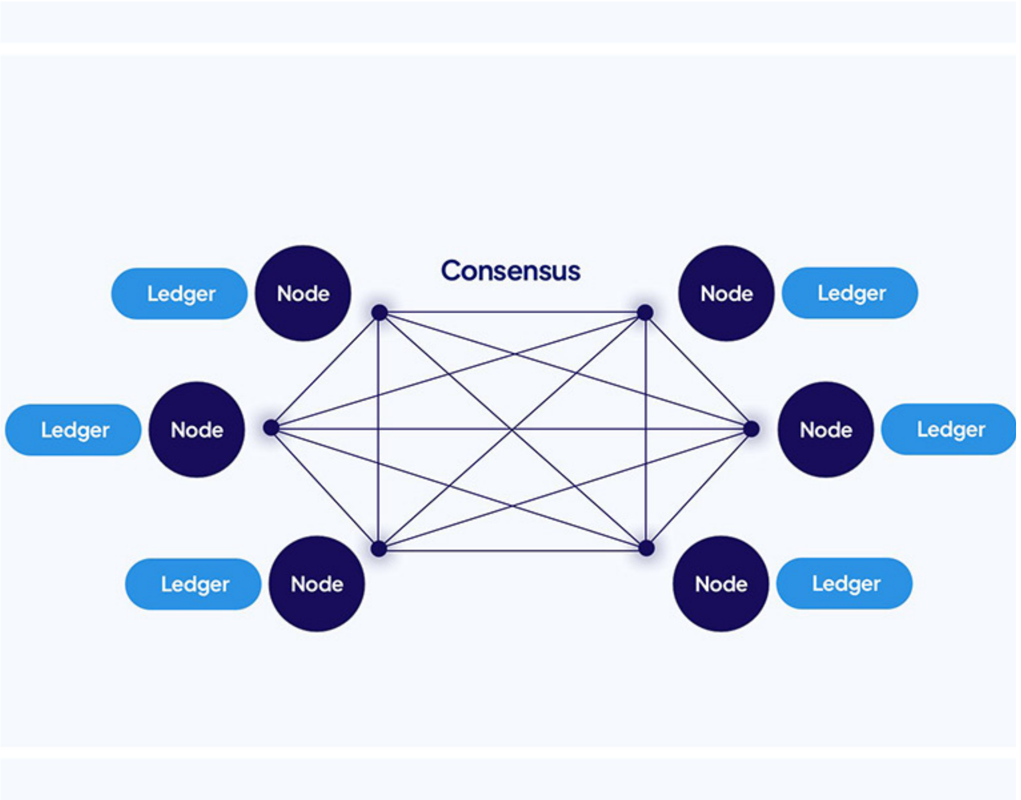
Sabina Rossi
Università Ca' Foscari di Venezia

Semia Guesmi
Università degli Studi di Camerino

Carla Piazza
Università degli Studi di Udine

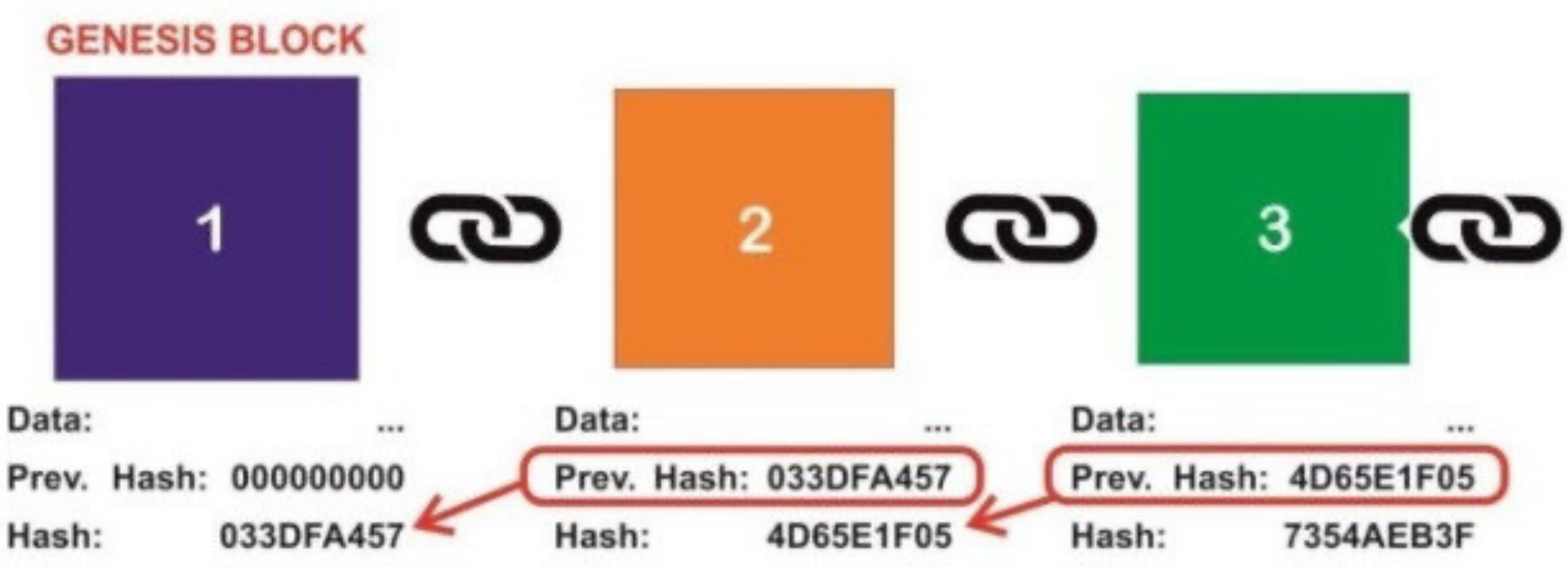
39th Italian Conference on Computational Logic JUNE 26-28, 2024 - ROME, ITALY

Distributed System: Blockchain



A digital database or ledger that is distributed among the nodes of a peer-to-peer network.

Distributed System: Blockchain



Blocks are cryptographically linked together. Blockchains are collection of blocks. A block is a collection of all the transactions with a cryptographic hash of the previous block.

Problematic

Maximal Extractable Value "MEV"

What is MEV ?

The Additional profit that participants in the blockchain ecosystem can extract through technically legitimate but potentially malicious actions.

Logic Reasoning to analyse malicious scenarios

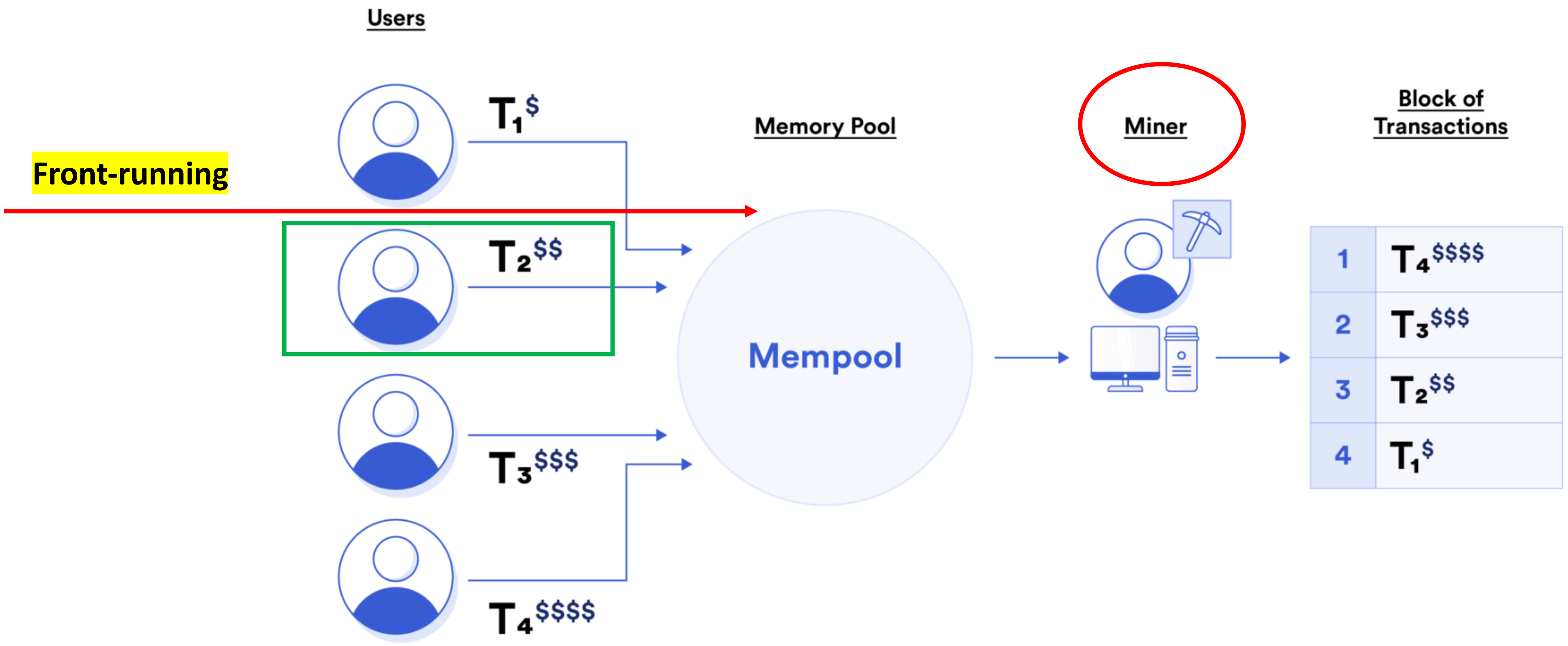
Formal Verification

- Ensuring integrity checks smart contracts to confirm they execute exactly as intended.

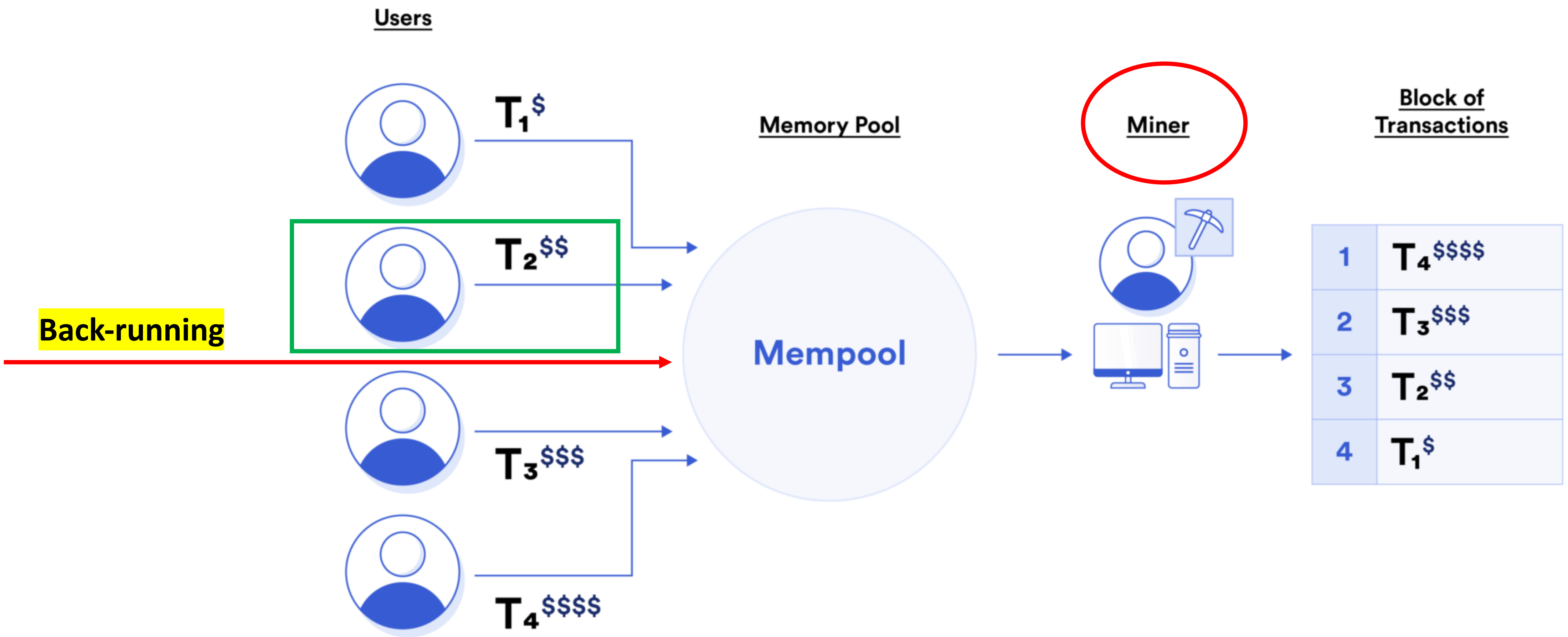
Model Checking

- Verifying Consistency: Model checking simulates all possible states of smart contracts and blockchain protocols to ensure they behave correctly.

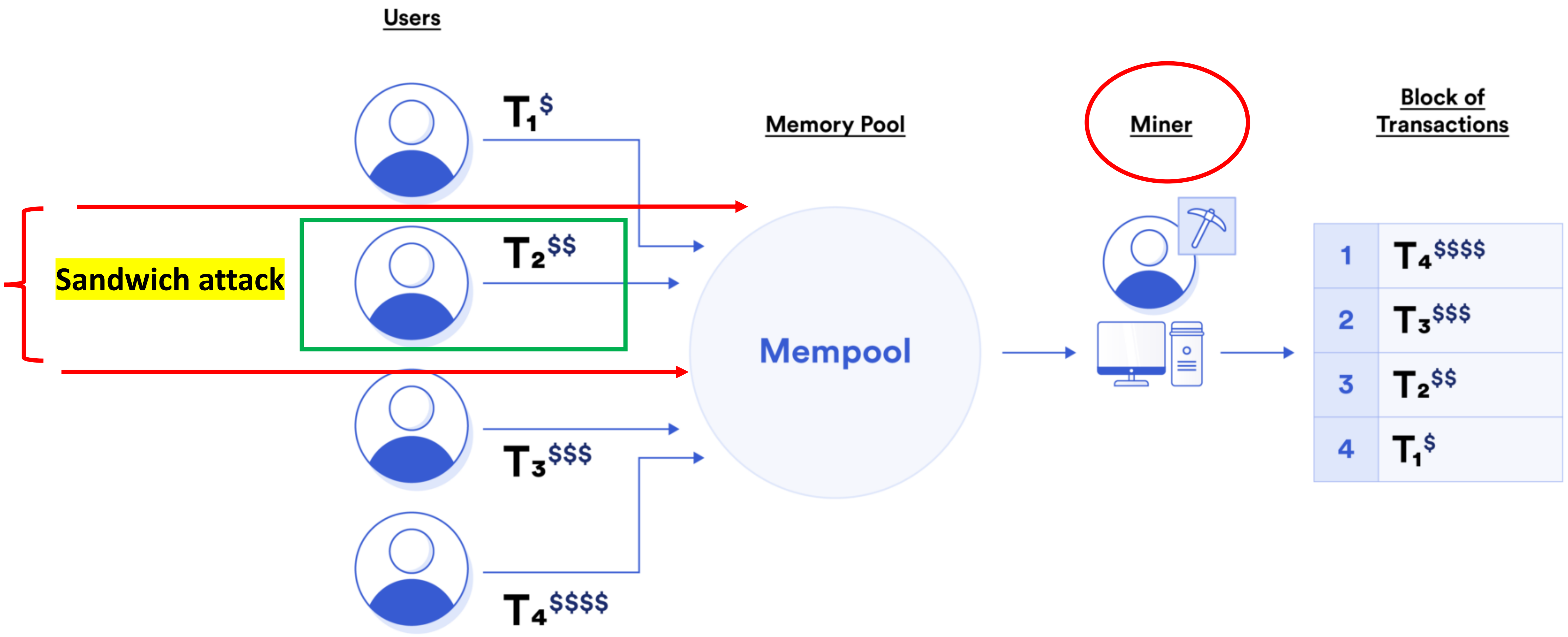
MEV Vulnerabilities



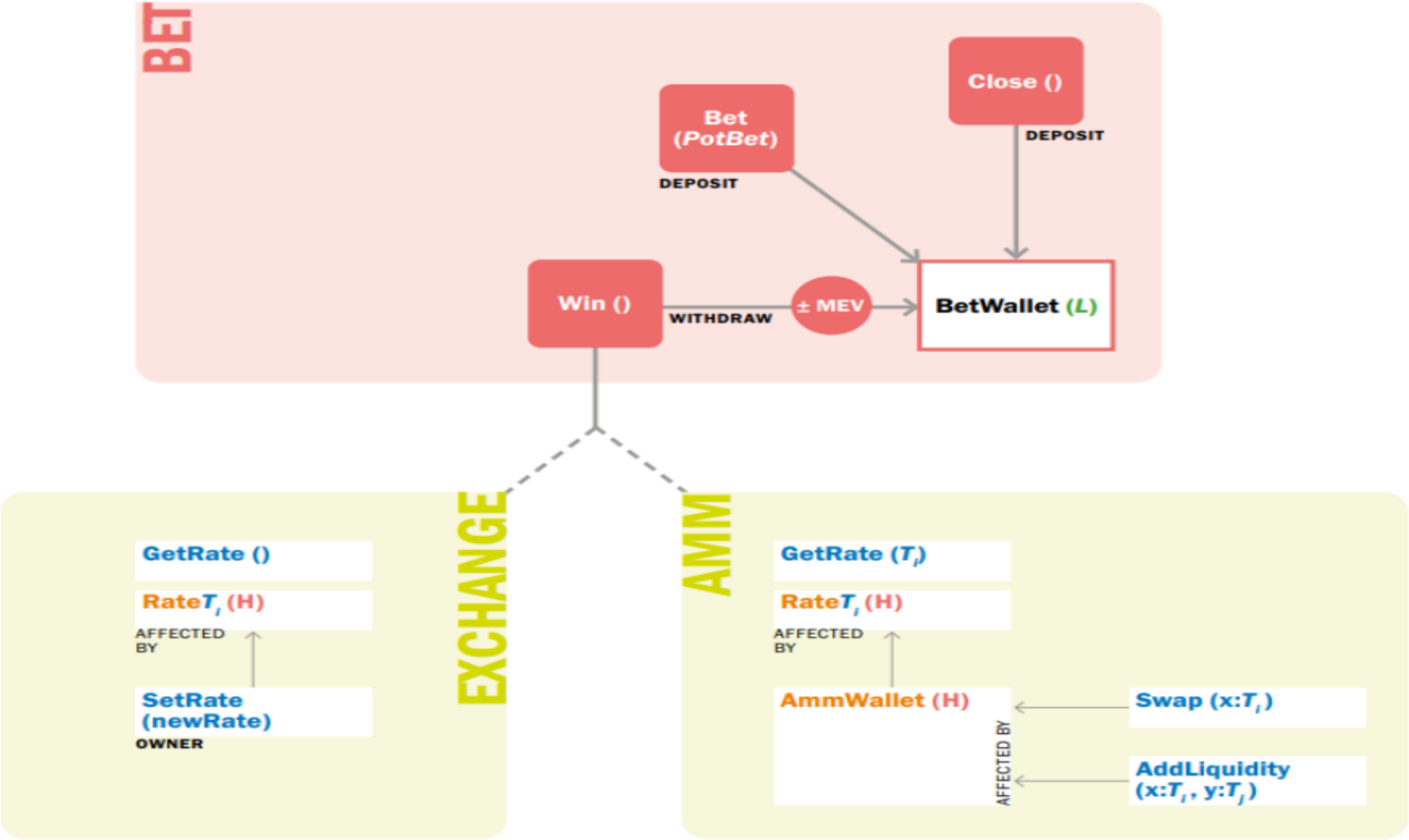
MEV Vulnerabilities



MEV Vulnerabilities

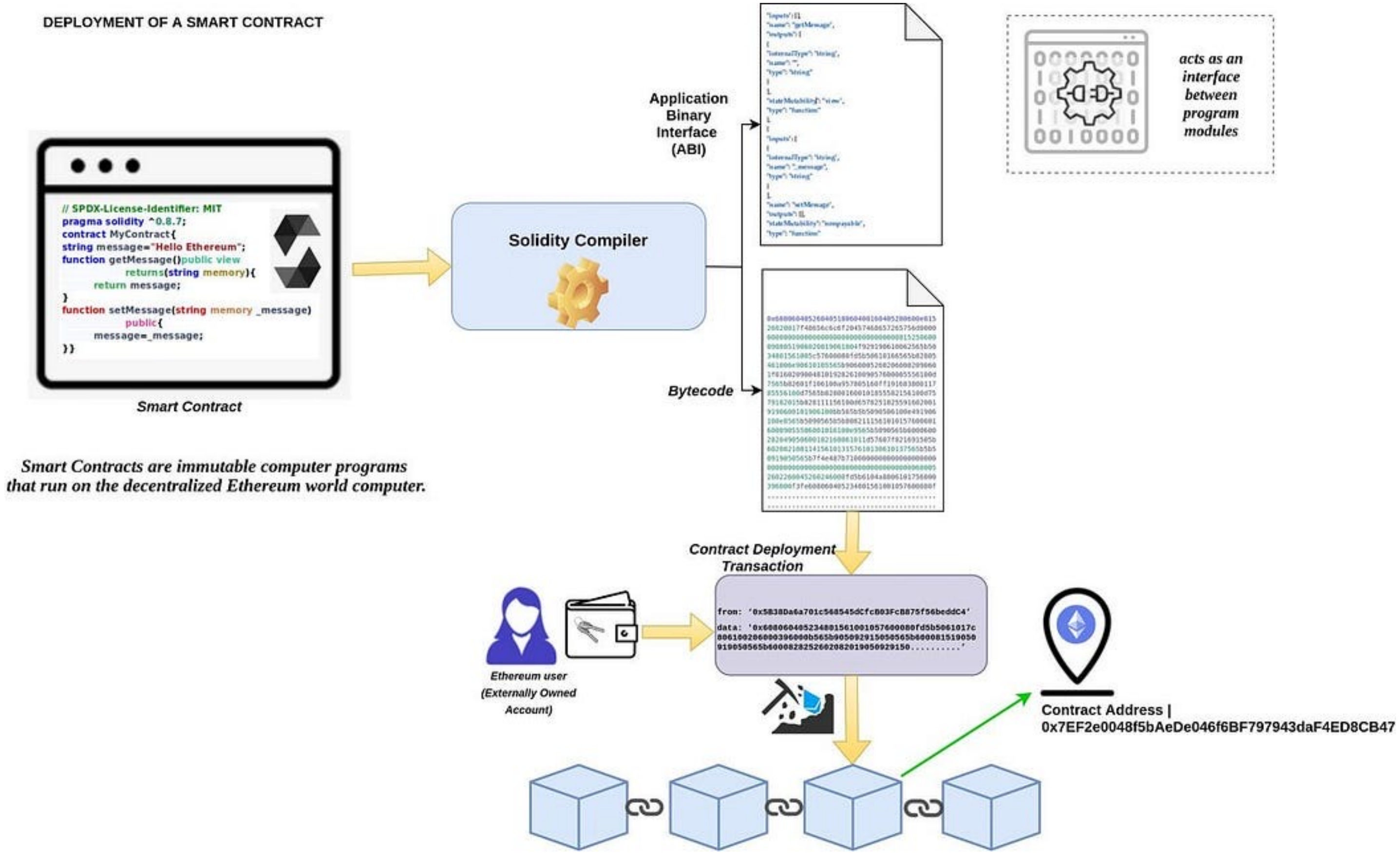


MEV Example: Bet Contract

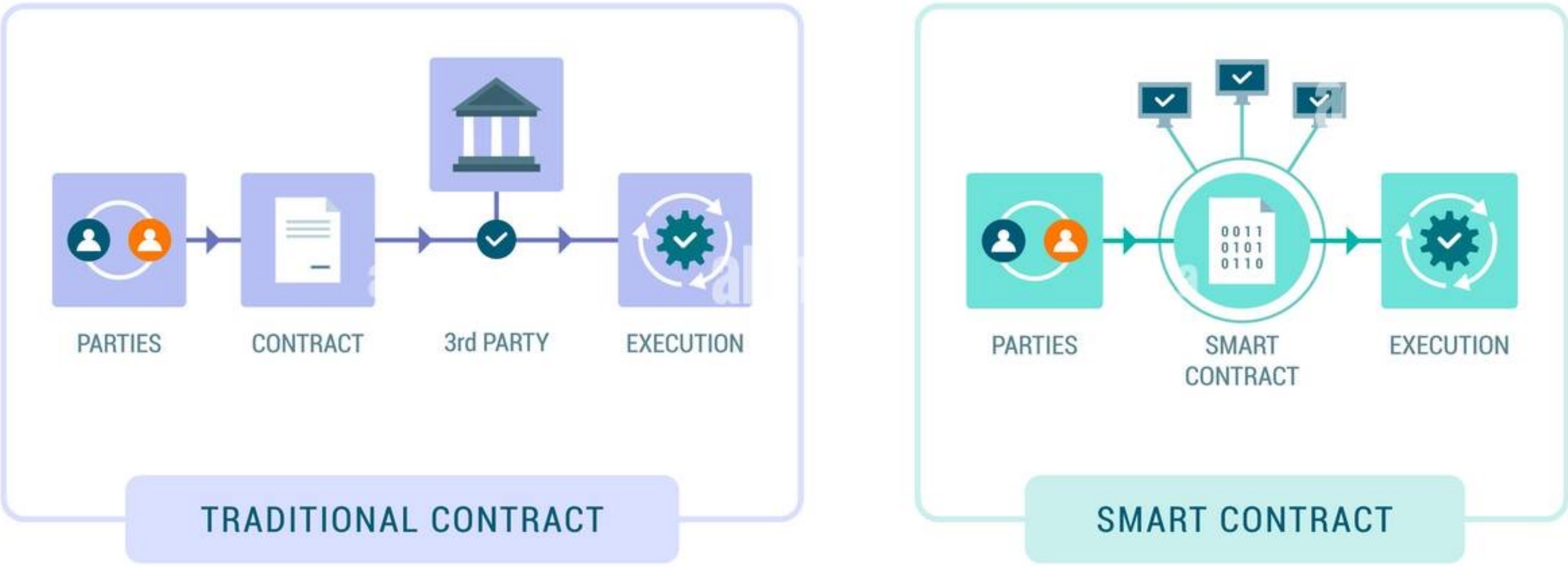



What are Smart Contracts?

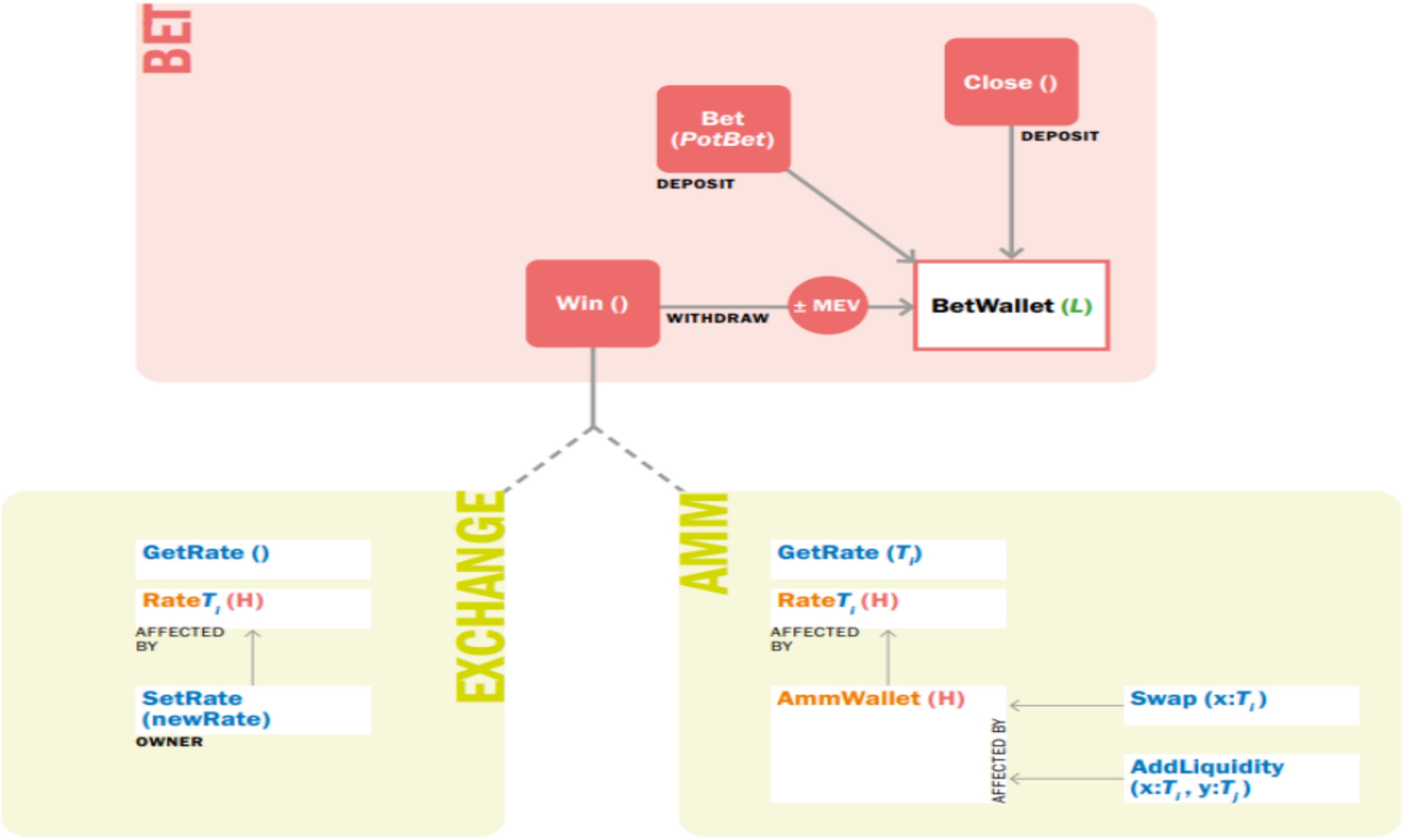
Smart contract



MEV Example: Bet Contract



MEV Example: Bet Contract



MEV Example: Bet Contract

Initial state:

Adversary Wallet

AMM wallet

AMM Exchange Rate (ETH) = $\frac{600}{600}$ = 1 (T)

$$S = \mathtt{M}[310:\mathtt{ETH}] \mid \mathtt{AMM}[600:\mathtt{ETH}, 600:\mathtt{T}] \mid \mathtt{block.num} = n - k \mid \cdots$$

$$\Delta = \mathtt{Bet}[10:\mathtt{ETH}, \mathtt{tok} = \mathtt{T}, \mathtt{rate} = 2, \mathtt{owner} = \mathtt{A}, \mathtt{deadline} = n]$$

Fixed Bet Rate = 2

Bet Wallet

MEV Example: Bet Contract

attack scenario

$$S \mid \Delta \xrightarrow{\texttt{M:Bet.bet}(?\,10:\texttt{ETH})} \texttt{M}[300:\texttt{ETH}] \mid \texttt{AMM}[600:\texttt{ETH}, 600:\texttt{T}] \mid \texttt{Bet}[20:\texttt{ETH}, \cdots]$$

$$\xrightarrow{\texttt{M:AMM.swap}(?\,300:\texttt{ETH},0)} \texttt{M}[200:\texttt{T}] \mid \texttt{AMM}[900:\texttt{ETH}, 400:\texttt{T}] \mid \texttt{Bet}[20:\texttt{ETH}, \cdots]$$

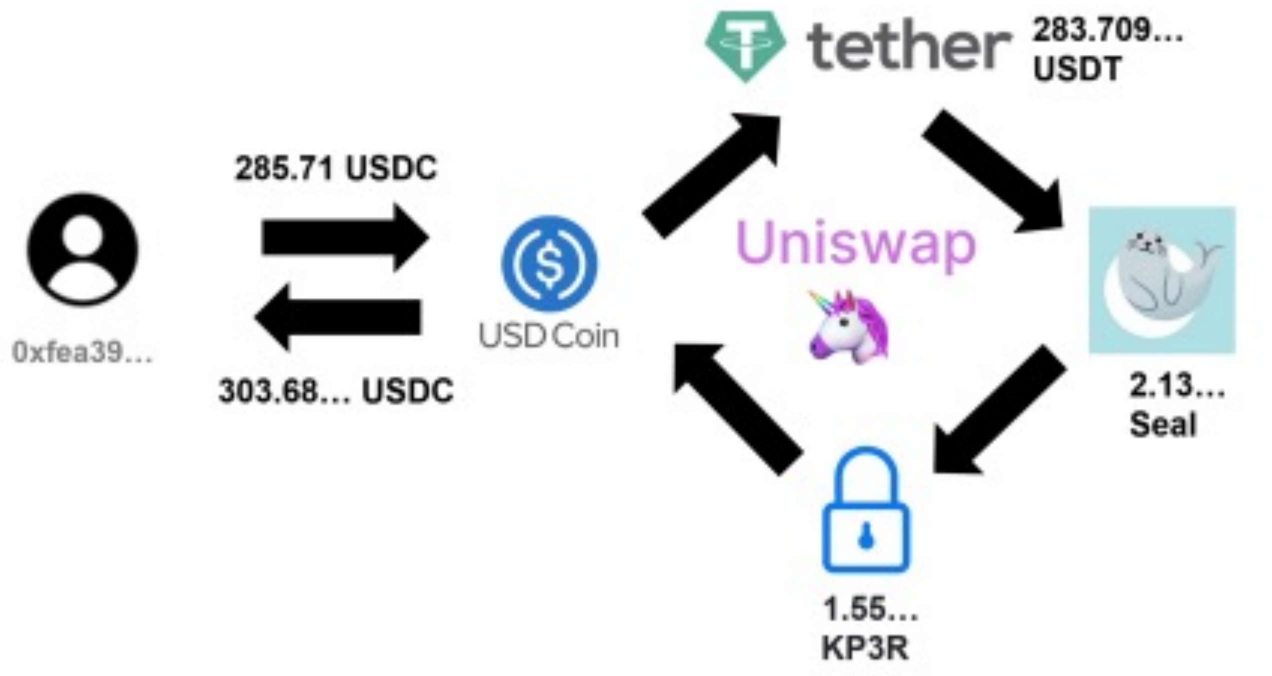
MEV :Front-running Transaction

AMM Exchange Rate (ETH) = $\frac{900}{400}$ > Fixed Bet Rate = 2

$$\xrightarrow{\texttt{M:Bet.win}()} \texttt{M}[20:\texttt{ETH}, 200:\texttt{T}] \mid \texttt{AMM}[900:\texttt{ETH}, 400:\texttt{T}] \mid \texttt{Bet}[0:\texttt{ETH}, \cdots]$$

$$\xrightarrow{\texttt{M:AMM.swap}(?\,200:\texttt{T},0)} \texttt{M}[320:\texttt{ETH}] \mid \texttt{AMM}[600:\texttt{ETH}, 600:\texttt{T}] \mid \texttt{Bet}[0:\texttt{ETH}, \cdots]$$

MEV Vulnerabilities



Cyclic Arbitrage in Decentralized Exchange Protocols

Our aim

Detect | Allocate | Analyze

Identify and mitigate MEV vulnerabilities within the instructions of smart contracts in the decentralized finance ecosystem.

⟶ Formal Verification methods

State of the art

Adversary perspective: Secure if the global MEV does not significantly increase.

Global MEV $MEV(S) = \max\left\{gain_{Adv}\,(S, \underline{X})\ \middle|\ \underline{X} \in K(Adv)^*\right\}$

Δ interacts safely with S $\leftrightarrow$ $MEV\ (S|\Delta) \leq (1+\varepsilon)\ MEV(S)$

$(\varepsilon - composability, see\ Clockwork\ Finance\ bt\ Babel, Daian, Kelkar, Juels)$

Contract perspective: Secure if being in a composition does not cause loss.

Local MEV $MEV(S, \Delta) = \max\left\{loss_{\Delta}\,(S, \underline{X})\ \middle|\ \underline{X} \in K(Adv)^*\right\}$

S does not interfere with the new contract Δ if: $MEV\ (S \mid \Delta, \Delta) = MEV_{\Delta}\ (S \mid \Delta, \Delta)$

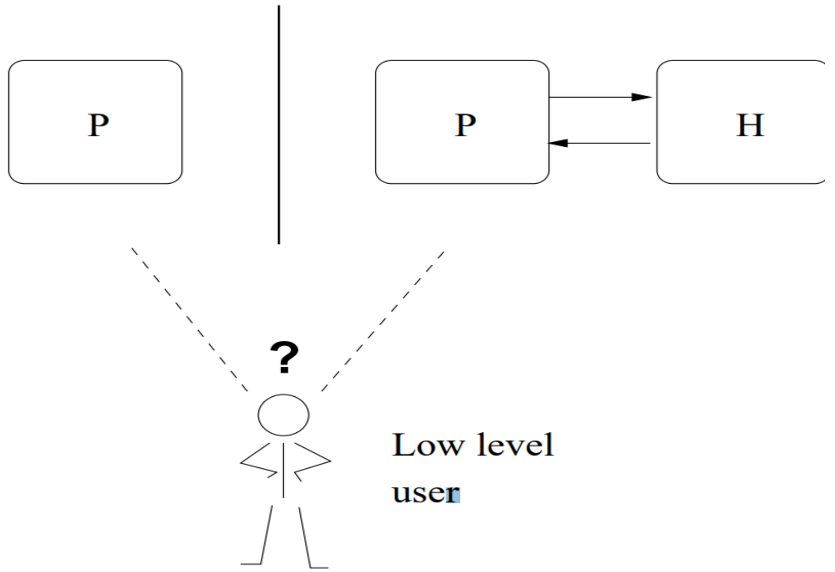
$(DeFi\ composability\ as\ MEV\ non-interference\ Bartoletti, Marchesin, Roberto\ )$

Noninterference

Noninterference aims to capture unwanted information flows in multi-level systems.

The notion of confidentiality: High and low levels.

A flow of information from high to low could represent the public disclosure of private data.



Generalized Unwinding Condition $\mathcal{W}(\doteq, \mathcal{R}, \div)$

Contract perspective in Computational framework:

- ✓ formalizing noninterference through unwinding conditions to analyze MEV.
- ✓ Guarantees that any reachable state resulting from high-level interactions still maintains indistinguishability with respect to low-level observations.

$$
\begin{array}{ccccc}
\langle F, \psi \rangle & \xrightarrow{\text{high}} & \langle G, \varphi \rangle & \rightsquigarrow & \langle \text{end}, \varphi' \rangle \\
\updownarrow & & & & \updownarrow \\
\pi =_l \psi & & & & \varphi' =_l \rho' \\
\updownarrow & & & & \updownarrow \\
\langle F, \pi \rangle & \rightsquigarrow & \cdots & \rightsquigarrow & \langle \text{end}, \rho' \rangle
\end{array}
$$

A pictorial representation of the unwinding condition

(Paper DLT2024: Noninterference Analysis for Smart Contracts: Would you Bet on it Samia G, Carla P, Sabina)

Information Flow

attack scenario: Interference from High level to Low level

$$S \mid \Delta \xrightarrow{\texttt{M:Bet.bet}(?\,10:\texttt{ETH})} \texttt{M}[300:\texttt{ETH}] \mid \texttt{AMM}[600:\texttt{ETH}, 600:\texttt{T}] \mid \texttt{Bet}[20:\texttt{ETH}, \cdots]$$

$$\xrightarrow{\texttt{M:AMM.swap}(?\,300:\texttt{ETH},0)} \texttt{M}[200:\texttt{T}] \mid \texttt{AMM}[900:\texttt{ETH}, 400:\texttt{T}] \mid \texttt{Bet}[20:\texttt{ETH}, \cdots]$$

$$\xrightarrow{\texttt{M:Bet.win}()} \texttt{M}[20:\texttt{ETH}, 200:\texttt{T}] \mid \texttt{AMM}[900:\texttt{ETH}, 400:\texttt{T}] \mid \texttt{Bet}[0:\texttt{ETH}, \cdots]$$

$$\xrightarrow{\texttt{M:AMM.swap}(?\,200:\texttt{T},0)} \texttt{M}[320:\texttt{ETH}] \mid \texttt{AMM}[600:\texttt{ETH}, 600:\texttt{T}] \mid \texttt{Bet}[0:\texttt{ETH}, \cdots]$$

HIGH level action

LOW level variable

LOW level action

HIGH level variable

Operational Semantics: Concurrent Imperative Language

```
Program Bet
    while (Deadline > BlockNum) do
        await (Player = 'NULL' ∧ PotBet ≠0 ) do
            skip
        if (PotBet = BetWallet) then
            Player := SenderBet;
            PlayerWalletEther := PlayerWalletEther − PotBet;
            BetWallet := BetWallet + PotBet
        else
            PotBet := 0;
            SenderBet := 0
```

```
Program Win
    while (Deadline > BlockNum) do
        await ( SenderWin = Player ) do
            skip
        if (BetRate < AmmRateEther) then
            PlayerWalletEther := PlayerWalletEther + BetWallet;
            BetWallet := 0
        else
            SenderWin := 'NULL'
```

Operational Semantics: Concurrent Imperative Language

```
Program Swap
    while true do
        await (AmountToSwap ≠0 ∧ TokenToSwap ≠'NULL' ) do
            skip
        K := AmmWalletT1 * AmmWalletT2
        if (TokenToSwap = 'T1') then
            Y := AmountToSwap * AmmRateT2;
            if (Y < AmmWalletT2) then
                AmmWalletT1 := AmmWalletT1 + AmountToSwap;
                AmmWalletT2 := K / AmmWalletT1;
                ZeroAMM
            else
                ZeroAMM
        else if (TokenToSwap = 'T2' ) then
            Y := AmountToSwap * AmmRateT1;
            if (Y < AmmWalletT1) then
                AmmWalletT2 := AmmWalletT2 + AmountToSwap;
                AmmWalletT1 := K / AmmWalletT2;
                ZeroAMM
            else
                ZeroAMM
```

```
Program GetRate
    while true do
        await ( T ≠'NULL') do
            skip
        if (T = 'T1') then
            AmmRateT1 := AmmWalletT1 / AmmWalletT2;
            T := 'NULL'
        else if (T = 'T2') then
            AmmRateT2 := AmmWalletT2 / AmmWalletT1;
            T := 'NULL'
        else
            T := 'NULL'
```

Operational Semantics: Concurrent Imperative Language

- Bet Contract $\notin \mathcal{W}(\doteq, \mathcal{R}, \doteqdot)$

$Bet\ \&\ Amm \equiv \boldsymbol{co}\ Bet\_Contract \mid Amm\_Contract\ \boldsymbol{oc}$

$$\langle Bet\ \&\ Amm, \sigma \rangle$$

$$\downarrow$$

$$\ldots\ \%\ execution\ of\ Swap$$ **(HIGH level action)**

$$\downarrow$$

$$\langle Bet\&Amm', \sigma\ [AmmWalletEther/900, AmmWalletT2/400] \rangle$$

$$\downarrow$$

$$\ldots\ \%\ execution\ of\ GateRate$$

$$\downarrow$$

$$\langle Bet, \sigma\ [AmmWalletEther/900, AmmWalletT2/400, AmmRateEther/2.25] \rangle$$

$$\downarrow$$

$$\ldots\ \%\ execution\ of\ Win$$ **(LOW level action)**

$$\downarrow$$

$$\langle end, \sigma\ [AmmWalletEther/900, AmmWalletT2/400, AmmRateEther/2.25, BetWallet/\mathbf{0}] \rangle$$

$$\langle Bet\ \&\ Amm, \sigma \rangle$$

$$\downarrow$$

$$\ldots\ \%\ execution\ of\ Win$$ **(LOW level action)**

$$\downarrow$$

$$\langle Amm, \sigma\ [SenderWin/\text{'}NULL\text{'}\rangle$$

$$\downarrow$$

$$\ldots\ \%\ execution\ of\ GateRate$$

$$\downarrow$$

$$\langle end, \sigma\ [SenderWin/\text{'}NULL\text{'}, AmmWalletEther/600, AmmWalletT2/600, BetWallet/\mathbf{10}]\rangle$$

- ✓ Identify the precise instructions and variables within the code that could potentially lead to information flows.
- ✓ Identify the specific dependencies of the contract that require deeper analysis.

Future Work

✓ Conducting in-depth investigations into the relationships between unwinding conditions and MEV and applying this method to analyze other case studies involving MEV attacks.

✓ Define this framework on fragments of languages for smart contracts, such as solidity.

✓ Model blockchain problem using program logic

✓ Machine Learning for MEV vulnerability detection in Ethereum smart contracts

Unwinding conditions for security in imperative languages

$$\frac{}{\langle \mathtt{skip}, \sigma\rangle \xrightarrow{\text{low}} \langle \text{end}, \sigma\rangle}$$

$$\frac{\langle a, \sigma\rangle \to n}{\langle X := a, \sigma\rangle \xrightarrow{\epsilon} \langle \text{end}, \sigma[X/n]\rangle}\ a \in \epsilon$$

$$\frac{\langle P_0, \sigma\rangle \xrightarrow{\epsilon} \langle P_0', \sigma'\rangle}{\langle P_0; P_1, \sigma\rangle \xrightarrow{\epsilon} \langle P_0'; P_1, \sigma'\rangle}\ P_0' \not\equiv \text{end}$$

$$\frac{\langle P_0, \sigma\rangle \xrightarrow{\epsilon} \langle \text{end}, \sigma'\rangle}{\langle P_0; P_1, \sigma\rangle \xrightarrow{\epsilon} \langle P_1, \sigma'\rangle}$$

$$\frac{\langle b, \sigma\rangle \to \text{true}}{\langle \mathtt{if}(b)\,\{P_0\}\ \mathtt{else}\ \{P_1\}, \sigma\rangle \xrightarrow{\epsilon} \langle P_0, \sigma\rangle}\ b \in \epsilon$$

$$\frac{\langle b, \sigma\rangle \to \text{false}}{\langle \mathtt{if}(b)\,\{P_0\}\ \mathtt{else}\ \{P_1\}, \sigma\rangle \xrightarrow{\epsilon} \langle P_1, \sigma\rangle}\ b \in \epsilon$$

$$\frac{\langle b, \sigma\rangle \to \text{true}}{\langle \mathtt{while}(b)\,\{P\}, \sigma\rangle \xrightarrow{\epsilon} \langle P; \mathtt{while}(b)\,\{P\}, \sigma\rangle}\ b \in \epsilon$$

$$\frac{\langle b, \sigma\rangle \to \text{false}}{\langle \mathtt{while}(b)\,\{P\}, \sigma\rangle \xrightarrow{\epsilon} \langle \text{end}, \sigma\rangle}\ b \in \epsilon$$

$$\frac{\langle b, \sigma\rangle \to \text{true} \qquad \langle S, \sigma\rangle \overset{\epsilon_2}{\rightsquigarrow} \langle \text{end}, \sigma'\rangle}{\langle \mathtt{await}(b)\,\{S\}, \sigma\rangle \xrightarrow{\epsilon_1 \cup \epsilon_2} \langle \text{end}, \sigma'\rangle}\ b \in \epsilon_1$$

$$\frac{\langle b, \sigma\rangle \to \text{false}}{\langle \mathtt{await}(b)\,\{S\}, \sigma\rangle \xrightarrow{\epsilon} \langle \mathtt{await}(b)\,\{S\}, \sigma\rangle}\ b \in \epsilon$$

$$\frac{\langle P_i, \sigma\rangle \xrightarrow{\epsilon} \langle P_i', \sigma'\rangle}{\langle \text{co } P_1 | \ldots | P_i | \ldots | P_n \text{ oc}, \sigma\rangle \xrightarrow{\epsilon} \langle \text{co } P_1 | \ldots | P_i' | \ldots | P_n \text{ oc}, \sigma'\rangle}$$

$$\frac{}{\langle \text{co end} | \ldots | \text{end} | \ldots | \text{end oc}, \sigma\rangle \xrightarrow{\text{low}} \langle \text{end}, \sigma\rangle}$$

$(Compositional\ Information\ Flow\ Security\ for\ Concurrent\ Programs\ Annalisa, Carla, Sabina,)$

THANKS FOR THE ATTENTION.

Samia Guesmi
semia.guesmi@unicam.it