

# *Verifying liquidity of Bitcoin contracts*

*(oral communication)*

Massimo Bartoletti  
Università di Cagliari



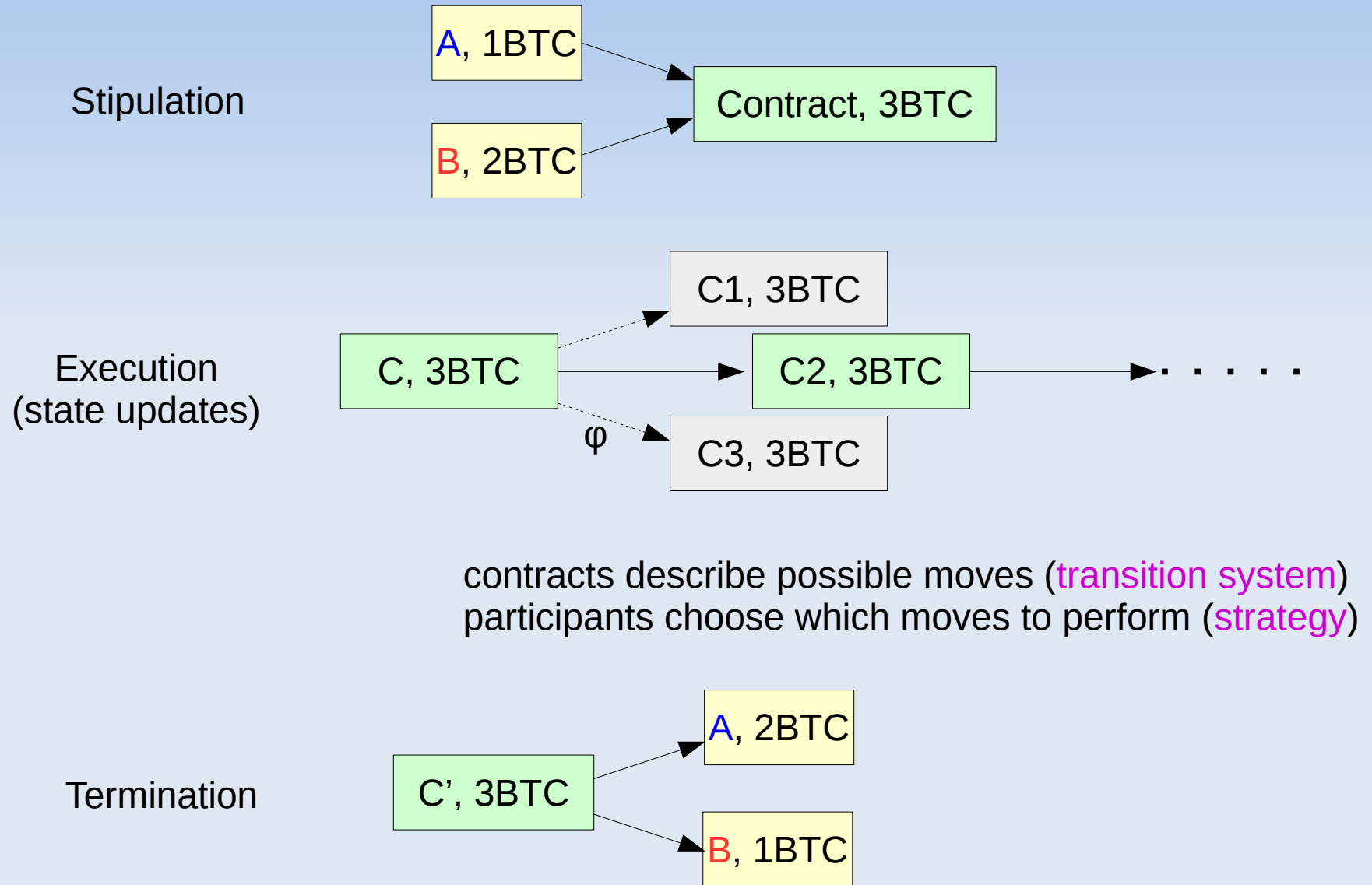
Roberto Zunino  
Università di Trento



DLT 2019, Pisa, 2019-02-12

The life of smart contracts, simply put

# Smart contracts: basic workflow



# Designing low-level smart contracts is hard!

**Win**( $\pi, a$ ) with  $\epsilon \neq \pi \sqsubset a$   
certifies that  $a$  has won all the rounds until  $\pi$  (included)

**Timeout1**  $\langle b \rangle$   
in: **Timeout1**( $\pi, b, a$ )  
in-script:  $\text{sig}_{\mathbf{K}(Timeout1, \pi, b, a)}(\bullet)$

**Timeout2**  $\langle b \rangle$   
in: **Timeout2**( $\pi, a, b$ )  
in-script:  $\text{sig}_{\mathbf{K}(Timeout2, \pi, a, b)}(\bullet)$

**Turn2fst**  $\langle b, \hat{s}_a, \hat{s}_b \rangle$   
in: **Turn2**( $\pi, a, b$ )  
in-script:  $\hat{s}_a, \hat{s}_b, \text{sig}_{\mathbf{K}(Turn2, \pi, a)}(\bullet)$

**Turn2snd**  $\langle b, \hat{s}_a, \hat{s}_b \rangle$   
in: **Turn2**( $\pi, b, a$ )  
in-script:  $\hat{s}_b, \hat{s}_a, \text{sig}_{\mathbf{K}(Turn2, \pi, a)}(\bullet)$

out-script( $\mathbb{T}, \sigma$ ):  $\text{ver}_{\mathbf{K}(Win, \pi, a)}(\mathbb{T}, \sigma) \vee \text{ver}_{\mathbf{K}(WinTO, \pi, a)}(\mathbb{T}, \sigma)$   
value:  $(1 + d) 2^{L - |\pi|} \mathbb{B}$

**Turn1**( $\pi, a, b$ ) with  $\pi \sqsubset a, b$   
certifies that  $a$  and  $b$  are playing in match  $\pi$ , where it is  $a$ 's turn to reveal her secret

in[0]: **Win**( $\pi 0, a$ )  
in-script[0]:  $\text{sig}_{\mathbf{K}(Win, \pi 0, a)}(\bullet)$   
in[1]: **Win**( $\pi 1, b$ )  
in-script[1]:  $\text{sig}_{\mathbf{K}(Win, \pi 1, b)}(\bullet)$

out-script( $\mathbb{T}, \hat{s}_a, \sigma$ ):  
(  $H(\hat{s}_a) = h_a^\pi \wedge \text{ver}_{\mathbf{K}(Turn1, \pi, a, b)}(\mathbb{T}, \sigma)$  )  
 $\vee \text{ver}_{\mathbf{K}(Turn1TO, \pi, a, b)}(\mathbb{T}, \sigma)$   
value:  $(1 + d) 2^{L - |\pi|} \mathbb{B}$

**Timeout1**( $\pi, a, b$ ) with  $\pi \sqsubset a, b$   
certifies that  $a$  lost against  $b$  in match  $\pi$  because she did not reveal her secret in time

in: **Turn1**( $\pi, a, b$ )  
in-script:  $\perp, \text{sig}_{\mathbf{K}(Turn1TO, \pi, a, b)}(\bullet)$

out-script( $\mathbb{T}, \sigma$ ):  $\text{ver}_{\mathbf{K}(Timeout1, \pi, a, b)}(\mathbb{T}, \sigma)$   
value:  $(1 + d) 2^{L - |\pi|} \mathbb{B}$   
lockTime:  $\tau_1 + (L - |\pi| - 1)\tau_{Round} + 2\tau_{Ledger}$

**Turn2**( $\pi, a, b$ ) with  $\pi \sqsubset a, b$   
certifies that  $a$  and  $b$  are playing in match  $\pi$ , where  $a$  has revealed her secret, and now it is  $b$ 's turn

**Secret**  $\langle \hat{s}_a \rangle$   
in: **Turn1**( $\pi, a, b$ )  
in-script:  $\hat{s}_a, \text{sig}_{\mathbf{K}(Turn1, \pi, a, b)}(\bullet)$

out-script( $\mathbb{T}, \hat{s}_a, \hat{s}_b, \sigma$ ):  
(  $H(\hat{s}_a) = h_a^\pi \wedge H(\hat{s}_b) = h_b^\pi$  )  
 $\wedge \text{ver}_{\mathbf{K}(Turn2, \pi, winner(a, b, \hat{s}_a, \hat{s}_b))}(\mathbb{T}, \sigma)$   
 $\vee \text{ver}_{\mathbf{K}(Turn2TO, \pi, a, b)}(\mathbb{T}, \sigma)$   
value:  $(1 + d) 2^{L - |\pi|} \mathbb{B}$

**Timeout2**( $\pi, a, b$ ) with  $\pi \sqsubset a, b$   
certifies that  $b$  lost against  $a$  in match  $\pi$  because she did not reveal her secret in time

in: **Turn2**( $\pi, a, b$ )  
in-script:  $\perp, \perp, \text{sig}_{\mathbf{K}(Turn2TO, \pi, a, b)}(\bullet)$

out-script( $\mathbb{T}, \sigma$ ):  $\text{ver}_{\mathbf{K}(Timeout2, \pi, a, b)}(\mathbb{T}, \sigma)$   
value:  $(1 + d) 2^{L - |\pi|} \mathbb{B}$   
lockTime:  $\tau_1 + (L - |\pi| - 1)\tau_{Round} + 4\tau_{Ledger}$

**Init**  
certifies that all players have placed their bets (and deposits)

$\forall p \in \mathcal{P} : \left\{ \begin{array}{l} \text{in}[p]: \text{Bet}_p \\ \text{in-script}[p]: \text{sig}_{K_p(\text{Bet}_p)}(\bullet) \end{array} \right.$

$\forall p \in \mathcal{P} : \left\{ \begin{array}{l} \text{out-script}[p](\mathbb{T}, \sigma): \text{ver}_{\mathbf{K}(\text{Init}, p)}(\mathbb{T}, \sigma) \\ \text{value}[p]: 1 + d\mathbb{B} \end{array} \right.$

**Win**( $a, a$ ) (leaf)  
contains the bet (and deposit) of  $a$  at the first round

in: **Init**[ $a$ ]  
in-script:  $\text{sig}_{\mathbf{K}(\text{Init}, a)}(\bullet)$

out-script( $\mathbb{T}, \sigma$ ):  $\text{ver}_{\mathbf{K}(Win, a, a)}(\mathbb{T}, \sigma)$   
value:  $1 + d\mathbb{B}$

**Win**( $\epsilon, a$ ) (root)  
certifies that  $a$  has won the lottery

(Variants as for **Win**( $\pi, a$ ))

out-script[ $a$ ]( $\mathbb{T}, \sigma$ ):  $\text{ver}_{K_a(\text{Collect})}(\mathbb{T}, \sigma)$   
value[ $a$ ]:  $N + d\mathbb{B}$

$\forall p \neq a : \left\{ \begin{array}{l} \text{out-script}[p](\mathbb{T}, \sigma): \text{ver}_{K_p(\text{Collect})}(\mathbb{T}, \sigma) \\ \text{value}[p]: d\mathbb{B} \end{array} \right.$

**CollectOrphanWin**( $\pi, a$ ) with  $\epsilon \neq \pi \sqsubset a$   
certifies that  $a$  was prevented by an adversary to participate in the rounds after  $\pi$ , but she can collect her winnings so far (see Theorem 5 for details)

## A lottery smart contract on Bitcoin

[BITCOIN 2017]

Very **error-prone** to design!  
Hard to guarantee **security**

We need **high level** languages:

EVM  $\rightarrow$  Solidity

Bitcoin  $\rightarrow$  BitML

# BitML in a nutshell

- A high-level language for **smart contracts** on Bitcoin

[ACM CCS 2018]

- Main features:
  - Depositing / withdrawing cryptocurrency
  - Committing to secrets (& revealing them)
  - Time constraints
  - Authorization-enabled actions
- Not Turing-complete, but can model timed commitment, escrow contracts, micropayment channels, lotteries, ...

# BitML Contract Example

init {  $A : 1 \text{ €}$ , secret  $a$   
       $B : 1 \text{ €}$ , secret  $b$  }

(reveal  $a$ .

  ( reveal  $b$ . if  $(a + b) \% 2 = 0$   
      then withdraw  $A$   
      else withdraw  $B$

  +after  $2 \cdot t$  : withdraw  $A$ )

+after  $t$  : withdraw  $B$ )

# BitML security

- Computationally sound compilation to Bitcoin  
no BitML attacks  $\Rightarrow$  no Bitcoin attacks
- To guarantee Bitcoin-level security, we still need to **verify** BitML code against desirable properties
- **Liquidity** is a desirable general property of smart contracts

# Liquidity

- Let  $S$  be a strategy for a participant interacting with a given contract  $C$
- Intuition:  
 $S$  is **liquid** for  $C$  iff, even in the presence of adversaries,  $S$  can eventually cause the contract balance to be assigned to participants (in some way)

reveal  $a$ . reveal  $b$ . split( $1\text{€} \rightarrow \text{withdraw } A \mid 1\text{€} \rightarrow \text{withdraw } B$ )  
no liquid strategy for  $A$

reveal  $a$ .  
( reveal  $b$ . split( $1\text{€} \rightarrow \text{withdraw } A \mid 1\text{€} \rightarrow \text{withdraw } B$ )  
+after  $t$  . withdraw  $A$ ) liquid strategy for  $A$ : reveal and wait

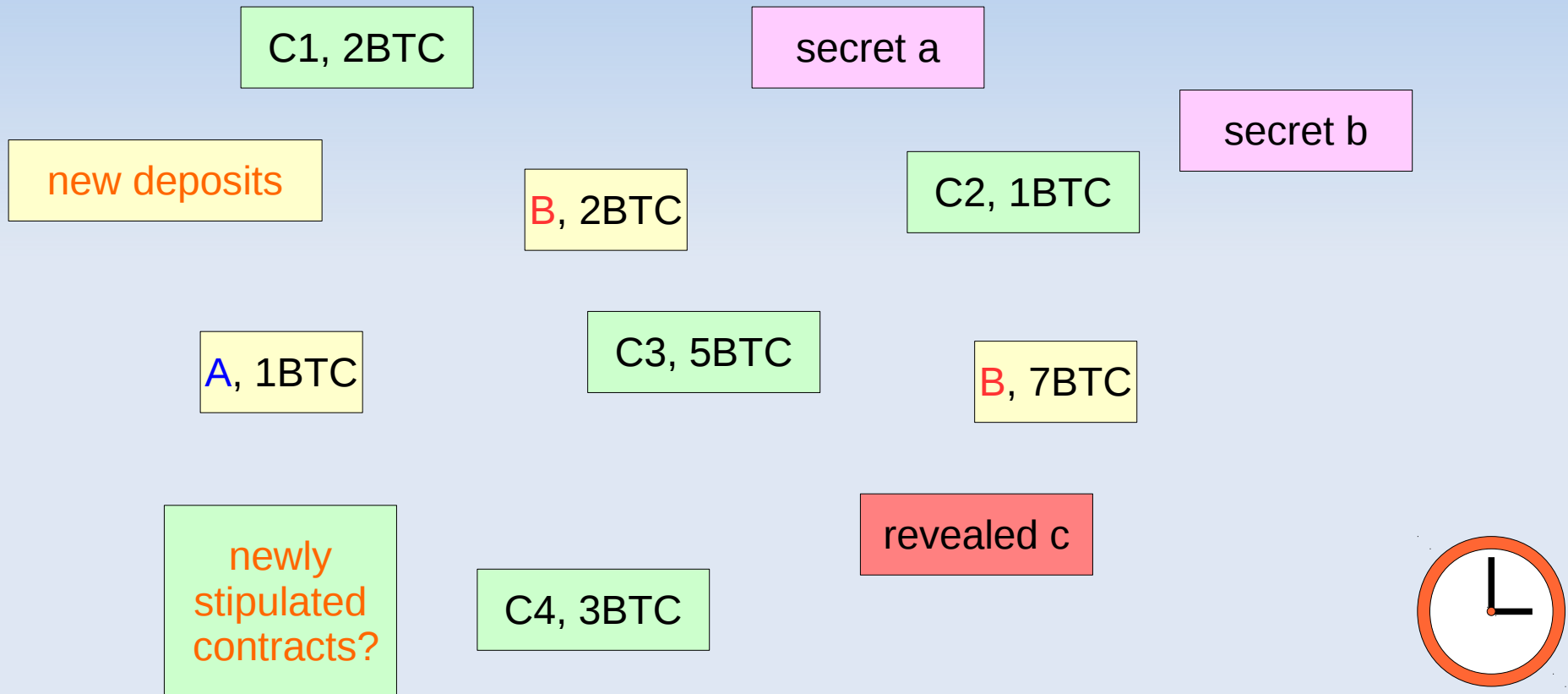
- Ethereum **Parity** attack violated liquidity



# Liquidity variants

- **Basic**: from any reachable state of C, strategy S can perform a sequence of moves “liquidating” C
- **Multiparty**: a set of participants cooperate to make C terminate
- **Quantitative**: we don't need C to terminate, as long as a large enough part of its balance is distributed
- **Known/unknown secrets**: S should be able to “liquidate” C no matter what the adversary secrets are

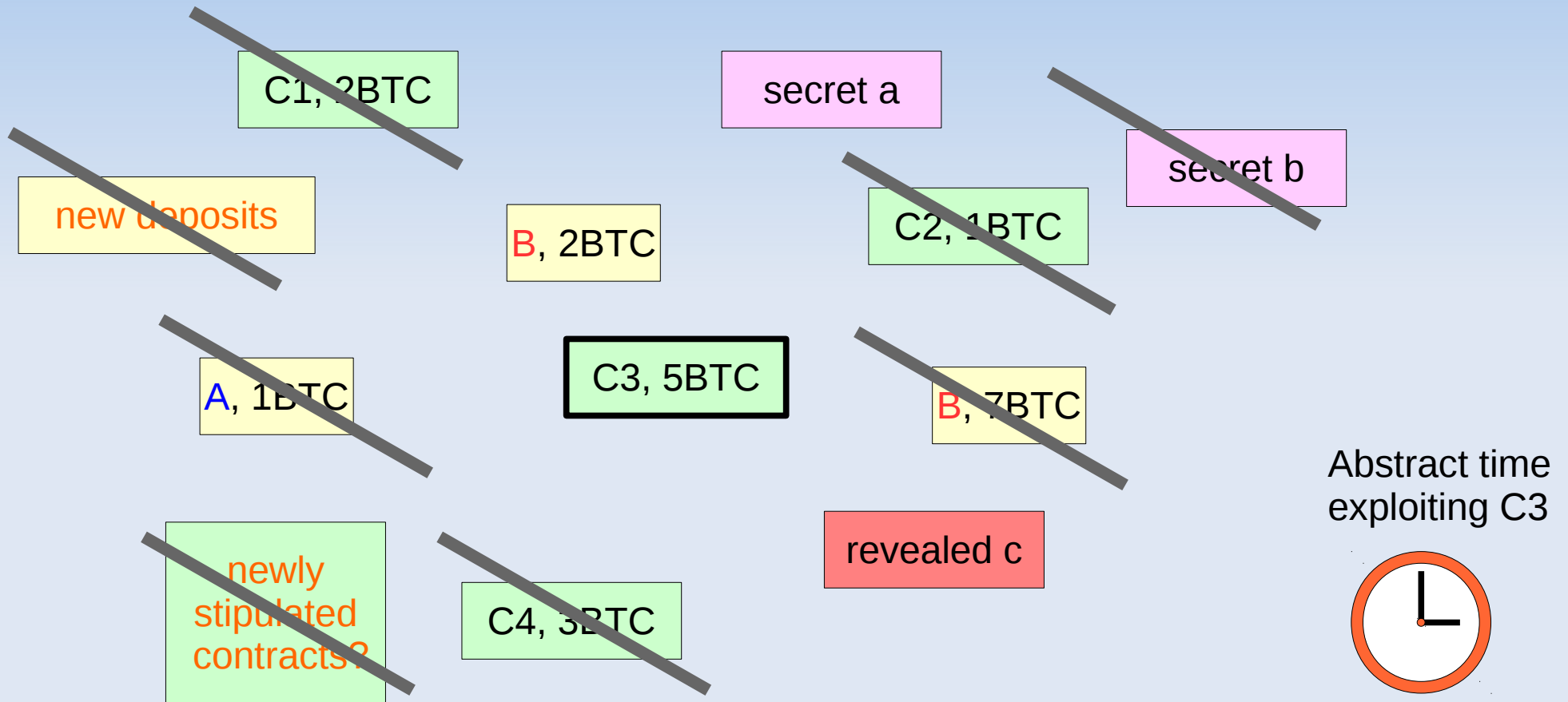
# BitML Abstraction



The BitML transition system is infinite-state, infinite branching, and timed

# BitML Abstraction

Focus on a given contract, only, and forget the irrelevant part of the configuration



The abstract transition system is now finite-state!

# Main Result

- Our abstraction is **sound and complete w.r.t. liquidity**

[to appear in POST 2019]

- Corollary: liquidity in BitML is **decidable**
- Verification **tool** in development (by UniCA)

# Further directions

- Strategy inference
  - Given a contract, find a strategy for a participant maximizing their payoff
  
- Probabilistic analysis
  - E.g. what is the average payoff?
  - Useful for lotteries

# Thank you

(all papers available on IACR)