A calculus for Bitcoin smart contracts

Massimo Bartoletti
Università di Cagliari

Roberto Zunino
Università di Trento

Perugia, 2018-02-01

Motivation

- Designing secure smart contracts is hard
  - Ethereum attacks: TheDAO, Parity
- Bugs can have very large consequences
  - TheDAO: 3M ETH / ~50M$ then / ~3000M$ now / fork
  - Parity: ~150M$ recently
- How to guarantee bug-free contracts?

Our Approach

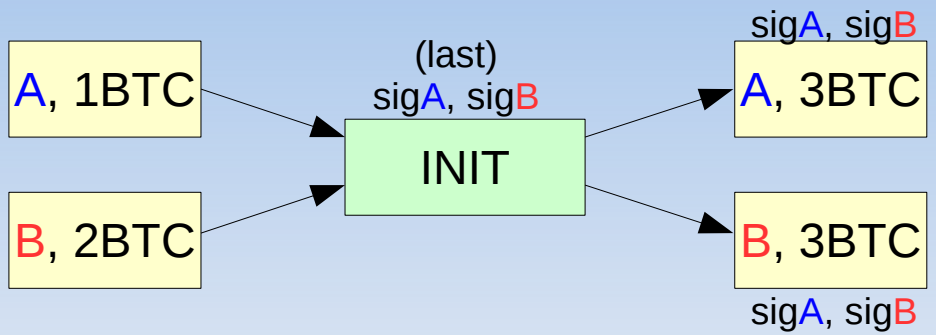
- Identify a class of smart contracts on Bitcoin
  - Simple enough to study
  - General enough for applications
- Design a specification language for that class
  - BitML
- Build a “compiler” from the language to Bitcoin
  - Symbolic specification to computational implementation
  - Leverage our formal model for Bitcoin (Financial Cryptography 2018)

BitML

A simple process calculus featuring:

- Stipulation: initial deposits & secret commitment
- Running a stipulated contract
  - additional deposit
  - withdrawal
- Constraints:
  - signature
  - secret reveal
  - time deadline

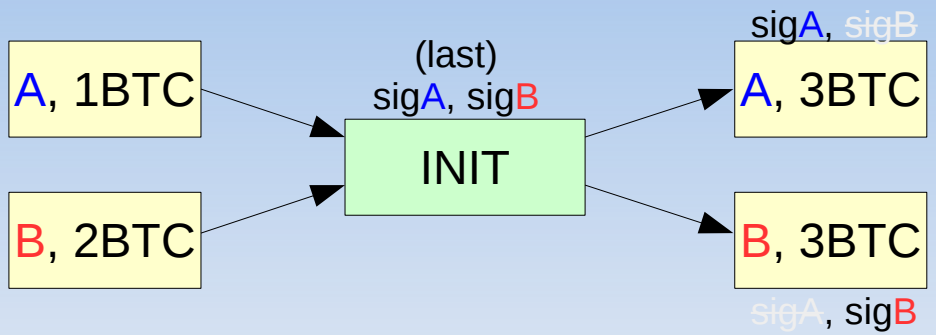
Example: “Far West”



$$\text{init}\,\{A : 1\,฿ \quad B : 2\,฿\}$$

$$(\text{withdraw}\ A \\ +\text{withdraw}\ B)$$

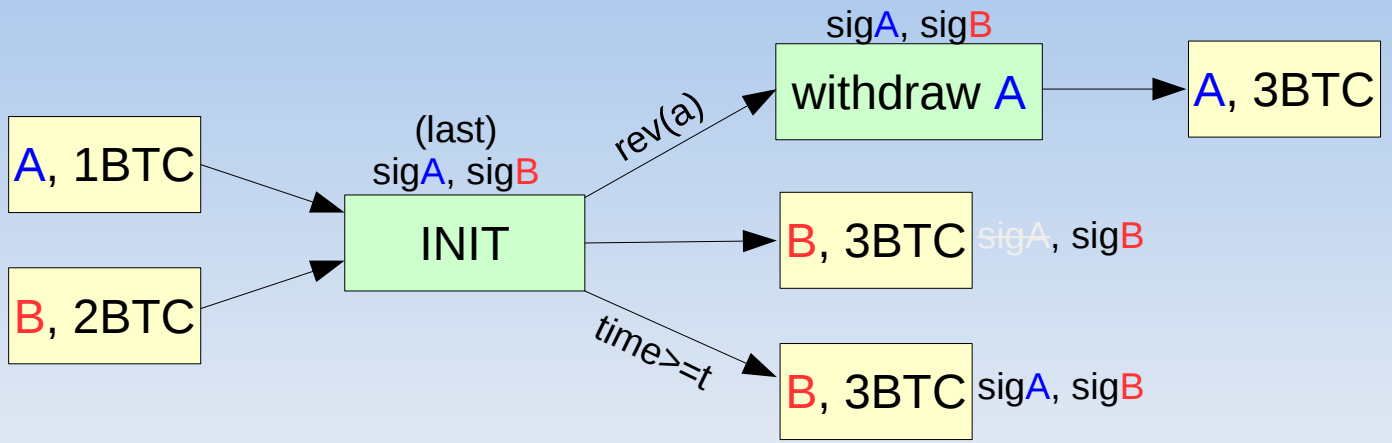
Example: Authorization



$$\text{init}\,\{A : 1\,฿ \quad B : 2\,฿\}$$

$$(B : \text{withdraw}\ A$$
$$+A : \text{withdraw}\ B)$$

Example: Incentive to Reveal



$$\text{init}\,\{A : 1\,\text{฿}, \text{secret}\,a \qquad B : 2\,\text{฿}\}$$

$$(\text{reveal}\,a.\ \text{withdraw}\,A$$
$$+A :\ \ \text{withdraw}\,B$$
$$+\text{after}\,\mathrm{t} :\ \ \text{withdraw}\,B)$$

Example: Fair Lottery

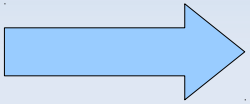
(general protocol: Bartoletti, Zunino - Bitcoin Workshop 2017)

$$
\begin{aligned}
&\mathsf{init}\,\{A : 1\,\text{฿}, \mathsf{secret}\,a\\
&\qquad B : 1\,\text{฿}, \mathsf{secret}\,b\}\\
&\\
&(\mathsf{reveal}\,a.\\
&\qquad (\ \mathsf{reveal}\,b.\ \mathsf{if}\,(a+b)\%2 = 0\\
&\qquad\qquad\qquad \mathsf{then\ withdraw}\,A\\
&\qquad\qquad\qquad \mathsf{else\ withdraw}\,B\\
&\qquad +\mathsf{after}\,2\cdot \mathrm{t} :\ \ \mathsf{withdraw}\,A)\\
&+\mathsf{after}\,\mathrm{t} :\ \ \mathsf{withdraw}\,B)
\end{aligned}
$$

Results

- Prove the compiler computationally sound

attacks at the Bitcoin level



attacks at the BitML level

- We can look for attacks in the simpler model, only

- This enables formal verification techniques

Thank you

Computational vs Formal Models

	Computational	Formal
Messages	bit strings	symbolic terms (e.g. enc(x,k))
Network	controlled by the adversary	controlled by the adversary
Adversary operations	anything	fixed set (enc, dec, ...)
Adversary limits	complexity probability	no limits
Protocol verification	hard	easier tool-supported (& bridge results)