BitML^x

Cross-chain Smart Contracts for Bitcoin-style Cryptocurrencies

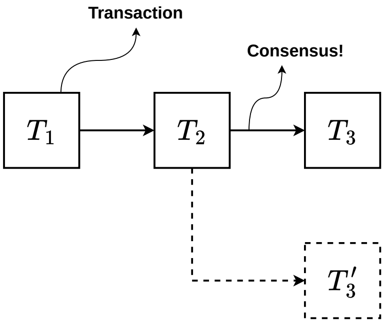
Federico Badaloni* Chrysoula Oikonomou**
Sebastian Holler* Clara Schneidewind* Pedro Moreno-Sanchez**

*Max Planck Institute for Security and Privacy

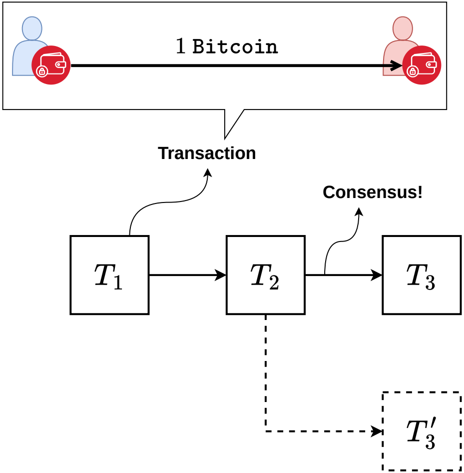
**IMDEA Software Institute

CSF 2025
June 19, 2025

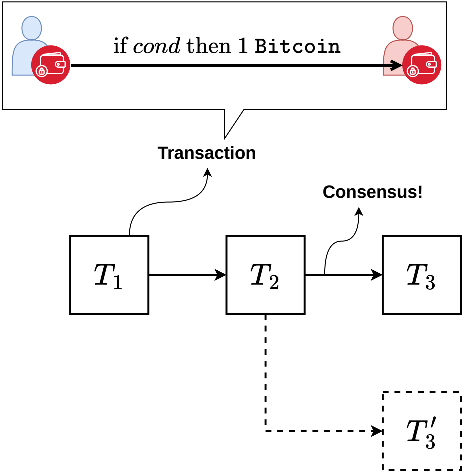
Blockchains



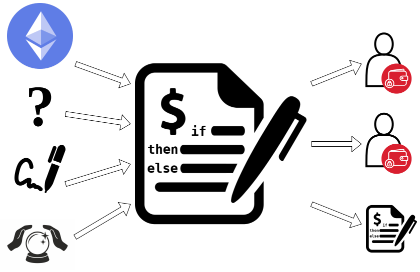
Blockchains



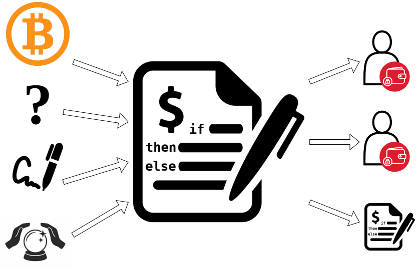
Blockchains



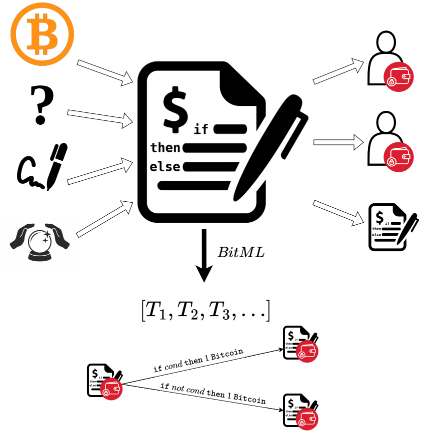
Smart Contracts



BitML: From Transactions To Smart Contracts

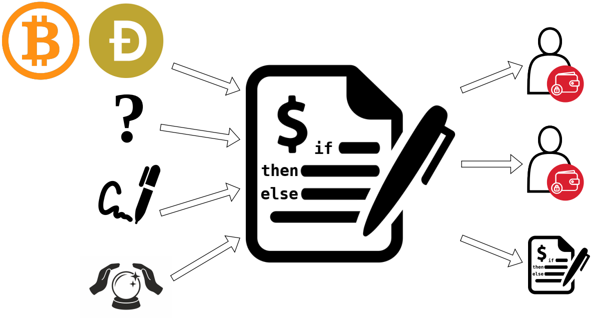


BitML: From Transactions To Smart Contracts

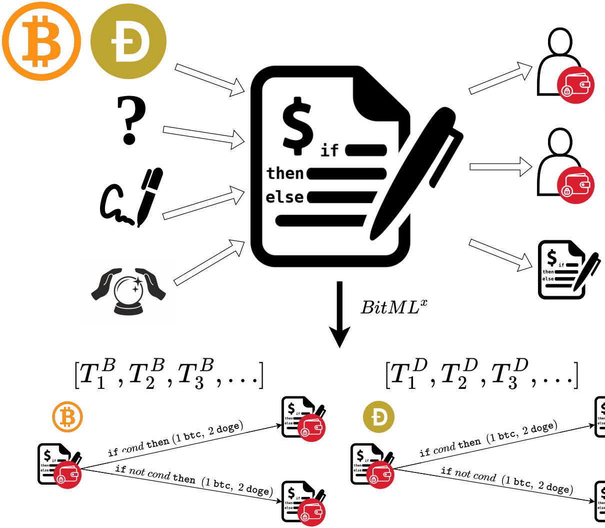


- Bartoletti & Zunino. (2018). BitML: A Calculus for Bitcoin Smart Contracts.

BitML^x: Cross-chain Smart Contracts



BitML^x: Cross-chain Smart Contracts

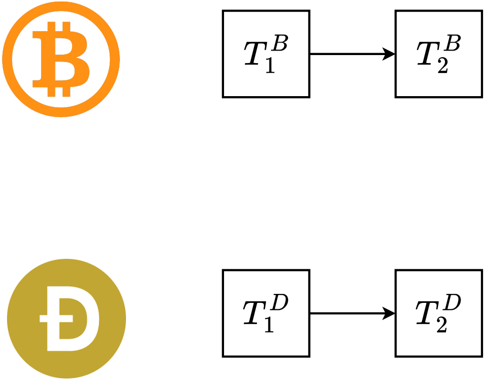


Cross-chain & Consensus

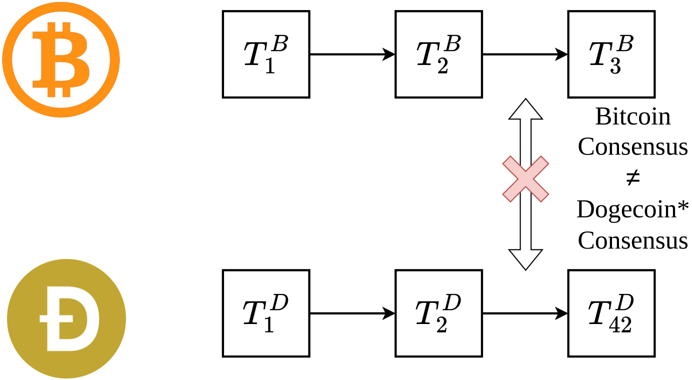
T_1^B

T_1^D

Cross-chain & Consensus



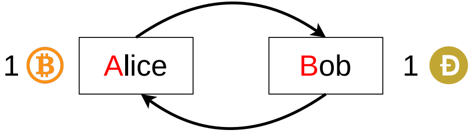
Cross-chain & Consensus



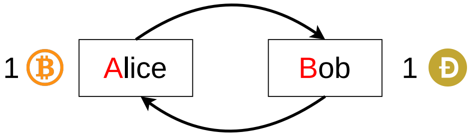
(*): Elon Musk ruined my slides 😠

BitML and Synchroincity

Example: Alice Wants To Swap Coins

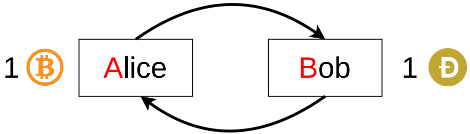


Example: Alice Wants To Swap Coins



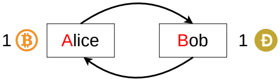
$$\overbrace{\{\ \underbrace{A : !\, 1Ƀ}_{\text{user deposits}} \ \mid\ \underbrace{B : \texttt{secret}\ s}_{\text{secret commitments}}\ \}}^{\text{preconditions}}\ \overbrace{Swap^{Ƀ}}^{\text{contract}}$$

Example: Alice Wants To Swap Coins



$$\{\overbrace{\underbrace{A \;:!\; 1\text{₿}}_{\text{user deposits}} \;\mid\; \underbrace{B : \texttt{secret}\ s}_{\text{secret commitments}}}^{\text{preconditions}}\}\;\overbrace{Swap^{\text{₿}}}^{\text{contract}} \qquad \{B \;:!\; 1\text{Đ} \mid B : \texttt{secret}\ s\}\,Swap^{\text{Đ}}$$

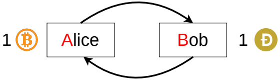
Example: Alice Wants To Swap Coins



$\{A :! 1\text{฿} \mid B : \texttt{secret}\ s\}Swap^{\text{฿}}$

$\{B :! 1\text{Đ} \mid B : \texttt{secret}\ s\}Swap^{\text{Đ}}$

Example: Alice Wants To Swap Coins

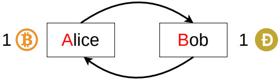


$$\{A :! 1₿ \mid B : \texttt{secret } s\}\mathit{Swap}^{₿}$$

$$\{B :! 1Ɖ \mid B : \texttt{secret } s\}\mathit{Swap}^{Ɖ}$$

$$\mathit{Swap}^{₿} = \mathit{Exchange}^{₿} + \mathit{Refund}^{₿}$$

Example: Alice Wants To Swap Coins



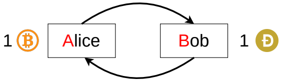
$$\{A :! 1\text{Ƀ} \mid B : \texttt{secret } s\}\mathit{Swap}^{\text{Ƀ}}$$

$$\{B :! 1\text{Đ} \mid B : \texttt{secret } s\}\mathit{Swap}^{\text{Đ}}$$

$$\mathit{Swap}^{\text{Ƀ}} = \mathit{Exchange}^{\text{Ƀ}} + \mathit{Refund}^{\text{Ƀ}}$$

$$\mathit{Exchange}^{\text{Ƀ}} = \texttt{reveal } s \texttt{ . withdraw } B$$

Example: Alice Wants To Swap Coins



$$\{A :! 1\text{฿} \mid B : \texttt{secret}\ s\}\mathit{Swap}^{\text{฿}}$$

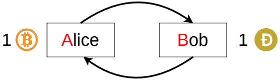
$$\{B :! 1\text{Đ} \mid B : \texttt{secret}\ s\}\mathit{Swap}^{\text{Đ}}$$

$$\mathit{Swap}^{\text{฿}} = \mathit{Exchange}^{\text{฿}} + \mathit{Refund}^{\text{฿}}$$

$$\mathit{Exchange}^{\text{฿}} = \texttt{reveal}\ s\ .\ \texttt{withdraw}\ B$$

$$\mathit{Refund}^{\text{฿}} = \texttt{after}\ t_0 :\ \texttt{withdraw}\ A$$

Example: Alice Wants To Swap Coins



$$\{A :! 1\text{฿} \mid B : \texttt{secret } s\}\mathit{Swap}^{\text{฿}}$$

$$\{B :! 1\text{Ɖ} \mid B : \texttt{secret } s\}\mathit{Swap}^{\text{Ɖ}}$$

$$\mathit{Swap}^{\text{฿}} = \mathit{Exchange}^{\text{฿}} + \mathit{Refund}^{\text{฿}}$$

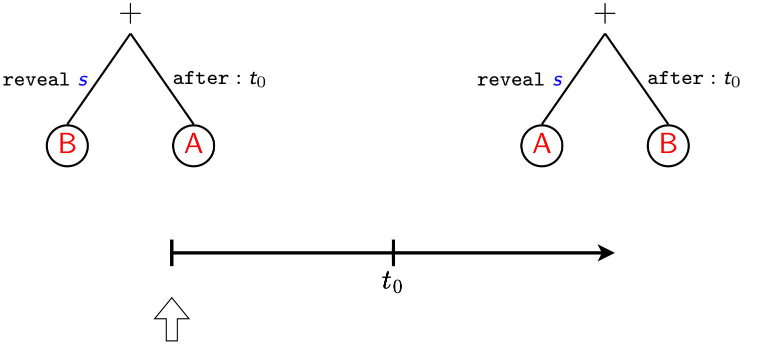
$$\mathit{Exchange}^{\text{฿}} = \texttt{reveal } s \,.\, \texttt{withdraw } B$$

$$\mathit{Refund}^{\text{฿}} = \texttt{after } t_0 : \texttt{withdraw } A$$

$$\mathit{Swap}^{\text{Ɖ}} = \texttt{reveal } s \,.\, \texttt{withdraw } A$$
$$+ \texttt{after } t_0 : \texttt{withdraw } B$$

Successful Swap

Alice | Bob

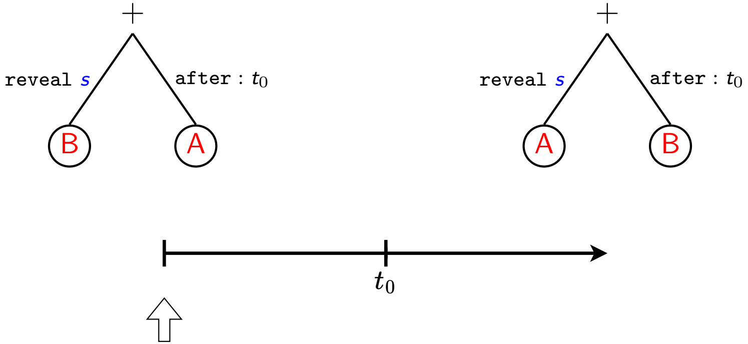


Successful Swap

Alice

- Wanna swap coins?

Bob



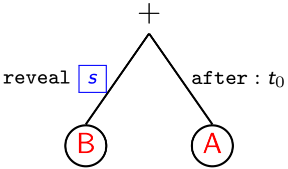
Successful Swap

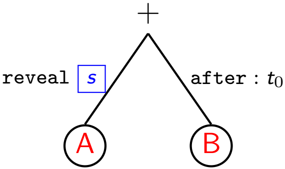
Alice

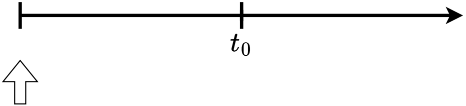
- Wanna swap coins?

Bob

- Sure! Here is s.



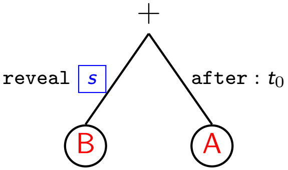




Successful Swap

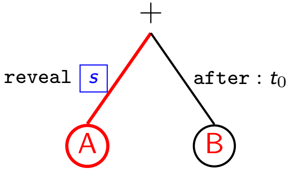
Alice

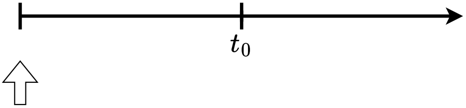
- Wanna swap coins?
- Thanks! I'm taking ₿😄



Bob

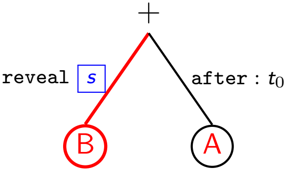
- Sure! Here is s.





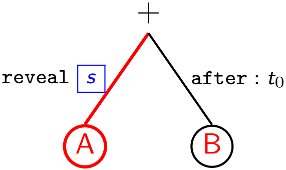
Alice

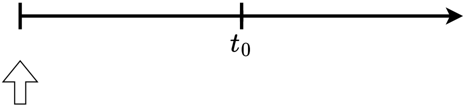
- Wanna swap coins?
- Thanks! I'm taking Ð😄



Bob

- Sure! Here is s.
- And I'm taking ₿😄

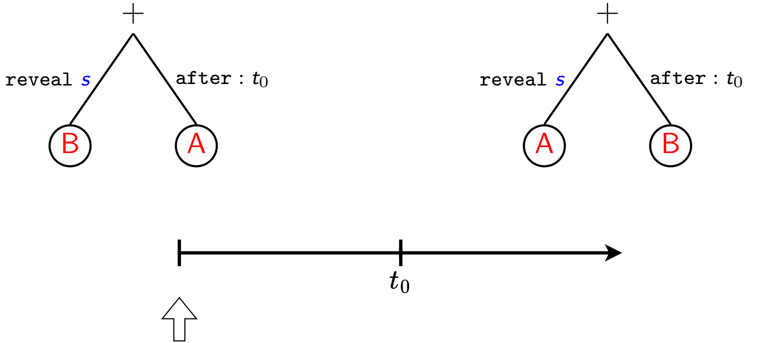




Successful Refund

Alice

Bob

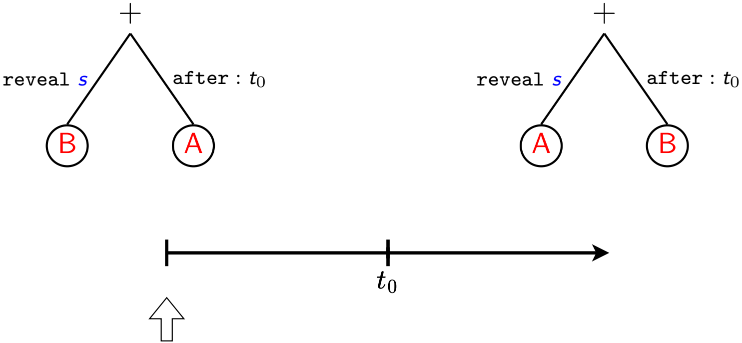


Successful Refund

Alice

Bob

- Wanna swap coins?



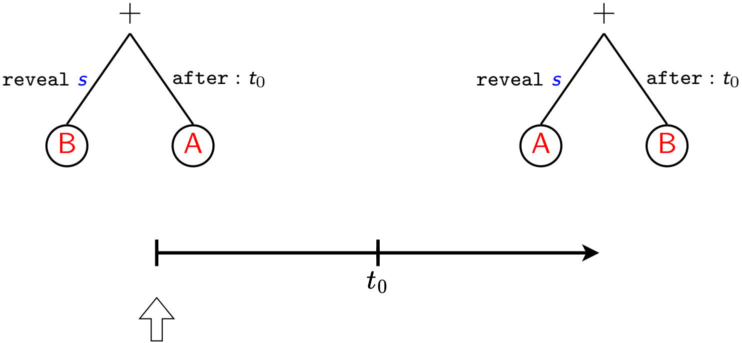
Successful Refund

Alice

- Wanna swap coins?

Bob

- Not really.



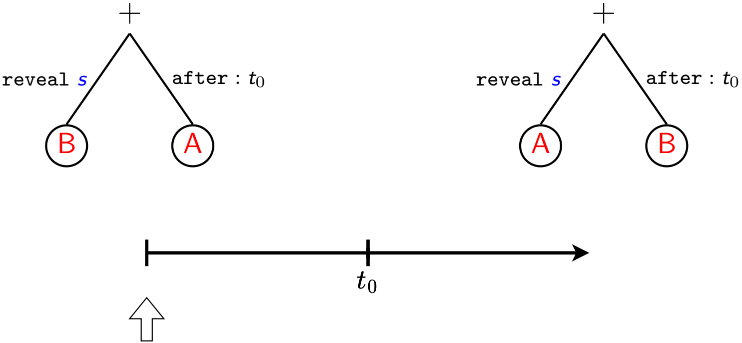
Successful Refund

Alice

- Wanna swap coins?
- Oh. That's ok. 🥲

Bob

- Not really.



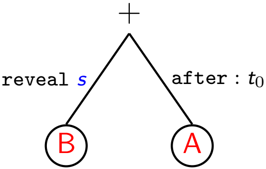
Successful Refund

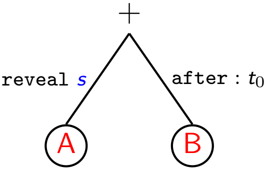
Alice

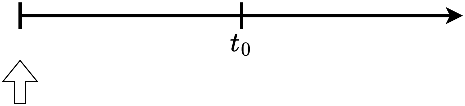
- Wanna swap coins?
- Oh. That's ok. 🥲

Bob

- Not really.
- Yeah, sorry. 😅







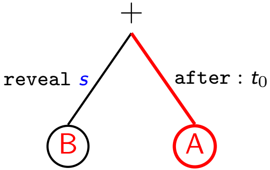
Successful Refund

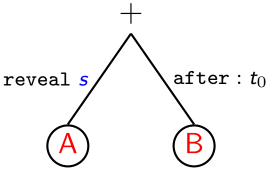
Alice

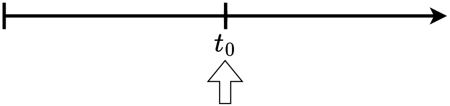
- Wanna swap coins?
- Oh. That's ok. 🥲
- I'll take back my ₿.

Bob

- Not really.
- Yeah, sorry. 😅



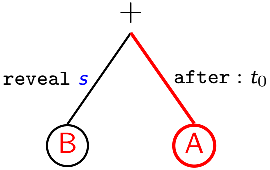




Successful Refund

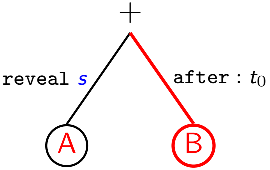
Alice

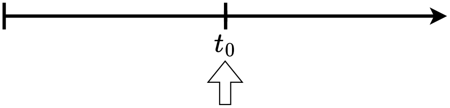
- Wanna swap coins?
- Oh. That's ok. 🥲
- I'll take back my ₿.



Bob

- Not really.
- Yeah, sorry. 😅
- And I'll take back my Ḓ.

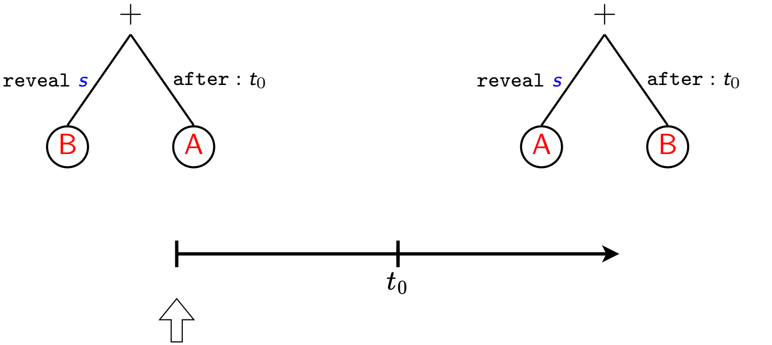




But...

Alice

Bob

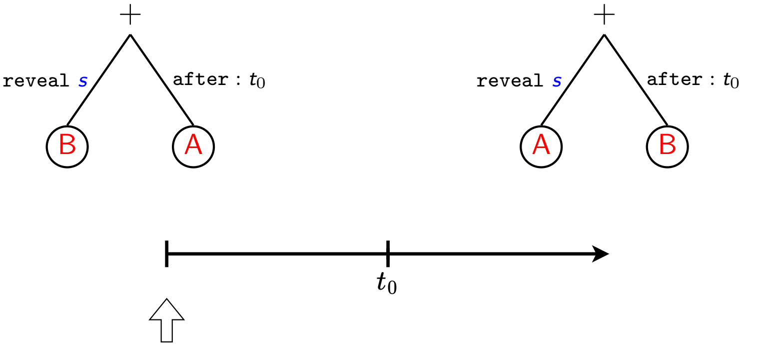


But...

Alice

- Wanna swap coins?

Bob



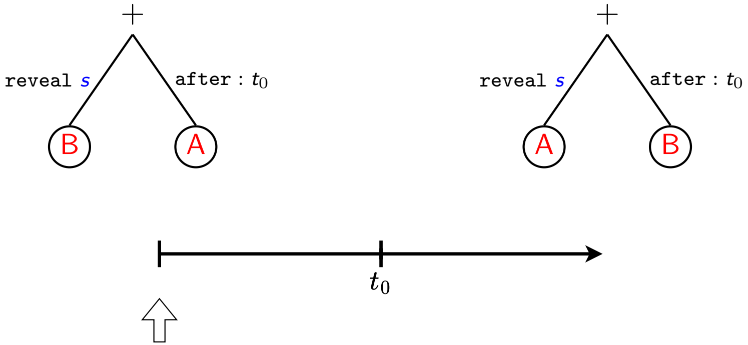
But...

Alice

- Wanna swap coins?

Bob

- Not really.



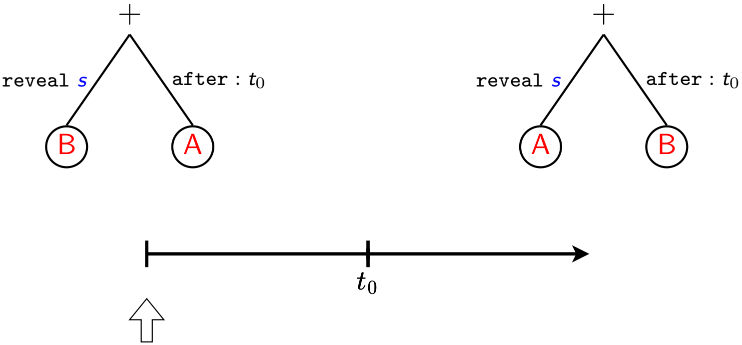
But...

Alice

- Wanna swap coins?
- Oh. That's ok. 🥲

Bob

- Not really.



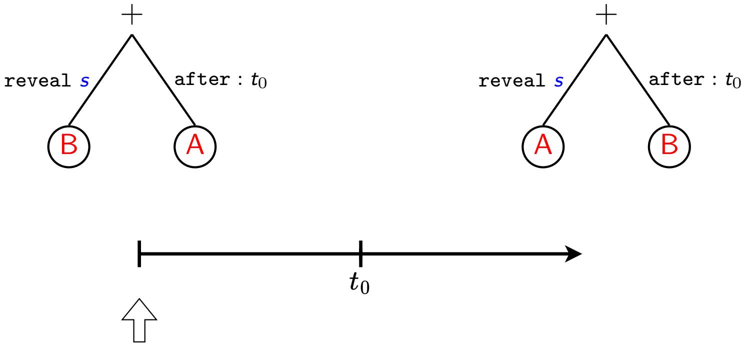
But...

Alice

- Wanna swap coins?
- Oh. That's ok. 🥲

Bob

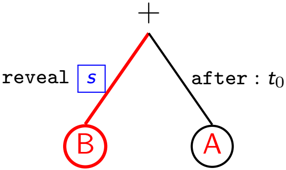
- Not really.
- Yes... totally ok. 😶



But...

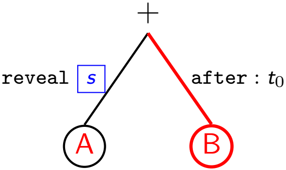
Alice

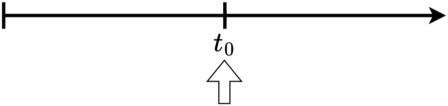
- Wanna swap coins?
- Oh. That's ok. 🥲
- Bob, WTF? 😱



Bob

- Not really.
- Yes... totally ok. 😶

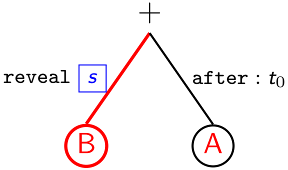




But...

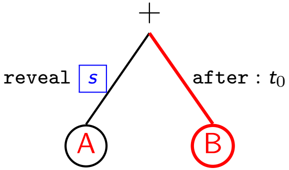
Alice

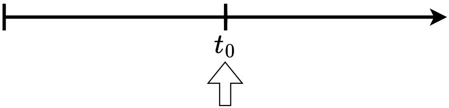
- Wanna swap coins?
- Oh. That's ok. 🥲
- Bob, WTF? 😱



Bob

- Not really.
- Yes... totally ok. 😶
- See ya, looser. 😈



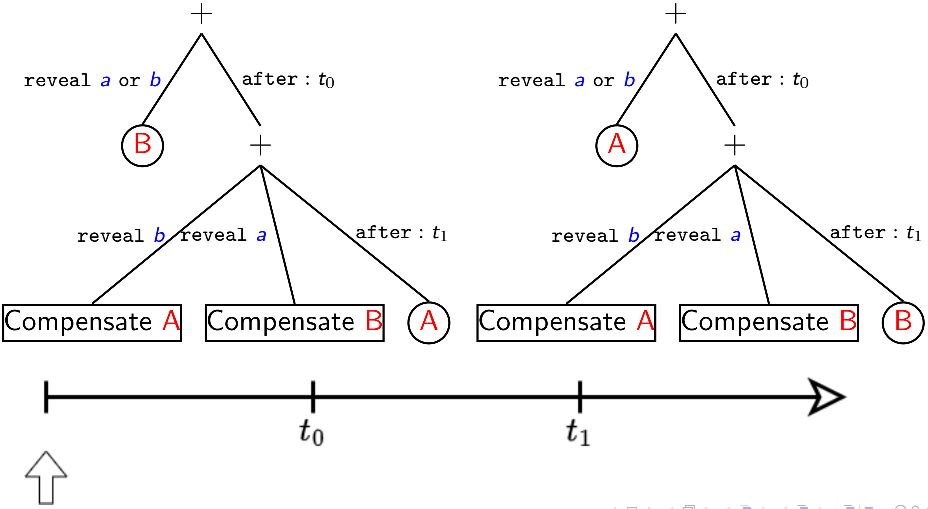


BitML^{x}compilation

Improved Swap

Alice has 1 ₿ and knows secret *a*

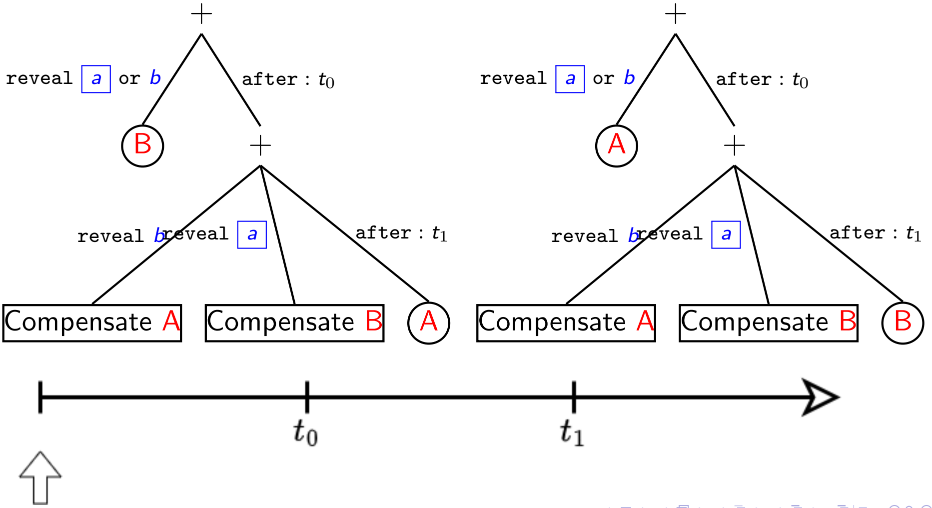
Bob has 1 Ð and knows secret *b*



Improved Swap

Alice has 1 ₿ and knows secret *a*

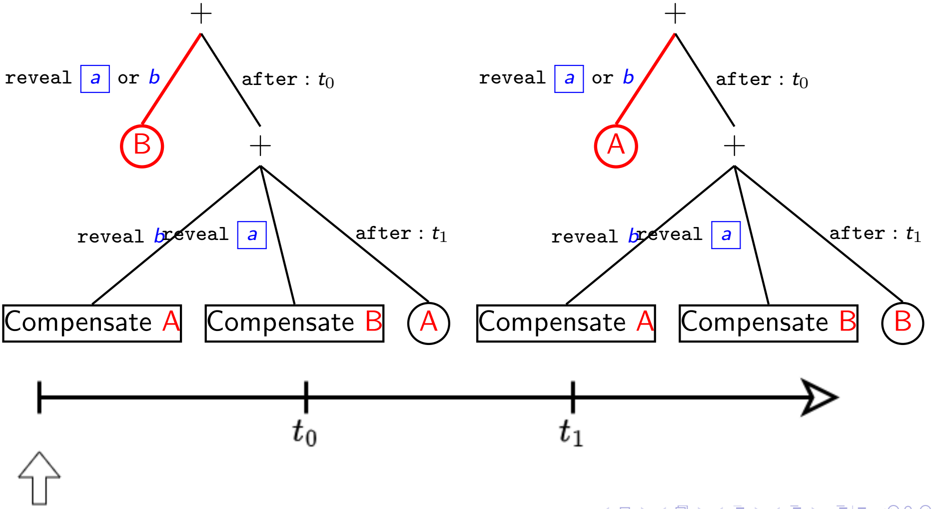
Bob has 1 Ð and knows secret *b*



Improved Swap

Alice has 1 ₿ and knows secret *a*

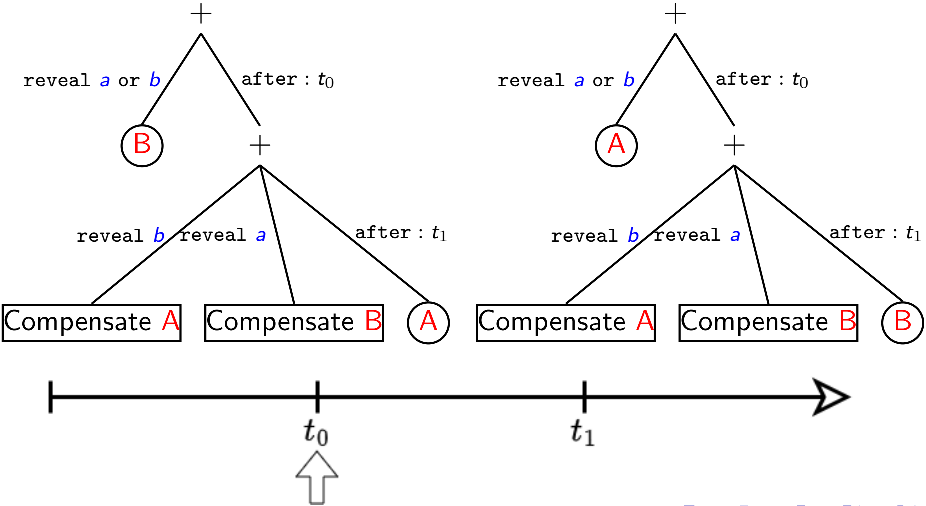
Bob has 1 Ð and knows secret *b*



Improved Swap

Alice has 1 ₿ and knows secret *a*

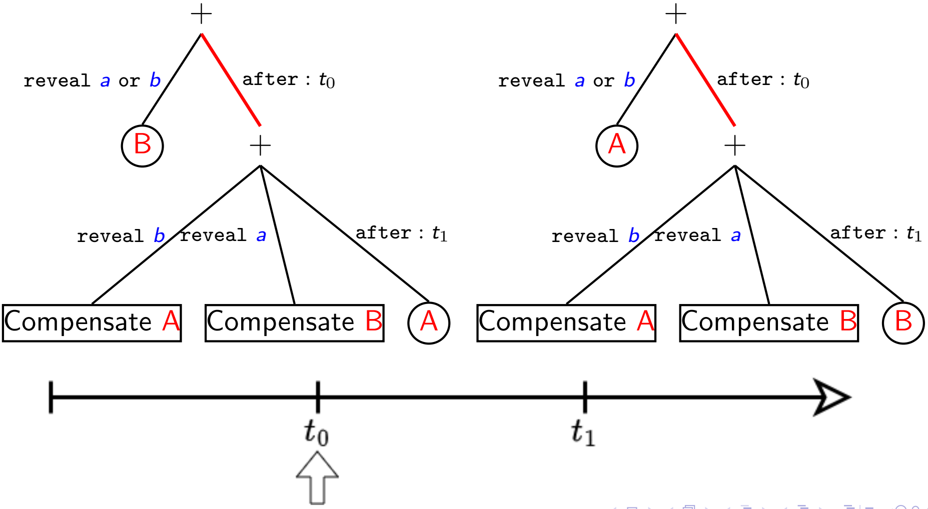
Bob has 1 Ð and knows secret *b*



Improved Swap

Alice has 1 ₿ and knows secret *a*

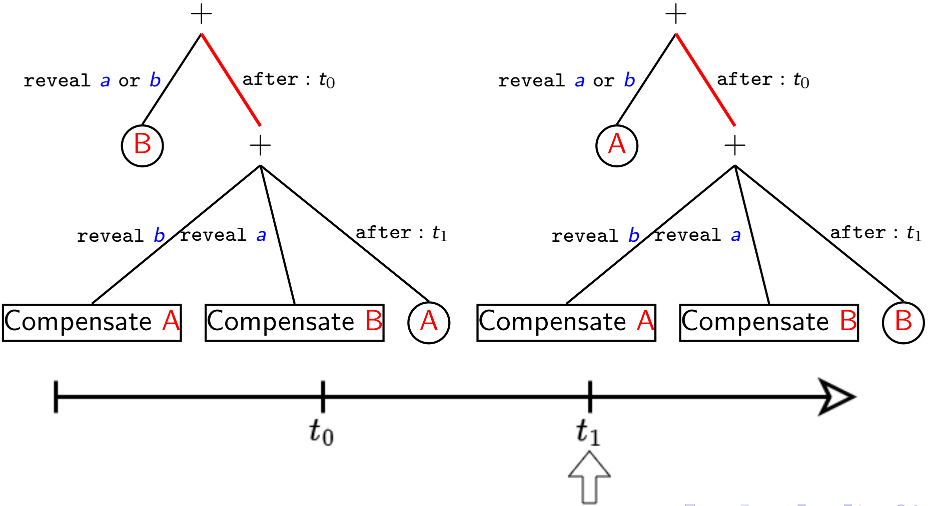
Bob has 1 Ð and knows secret *b*



Improved Swap

Alice has 1 ₿ and knows secret a

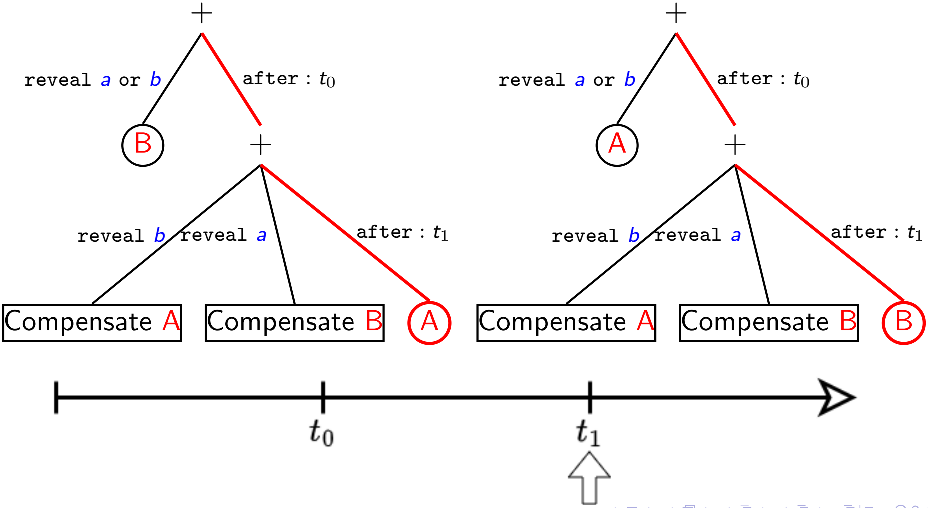
Bob has 1 Ð and knows secret b



Improved Swap

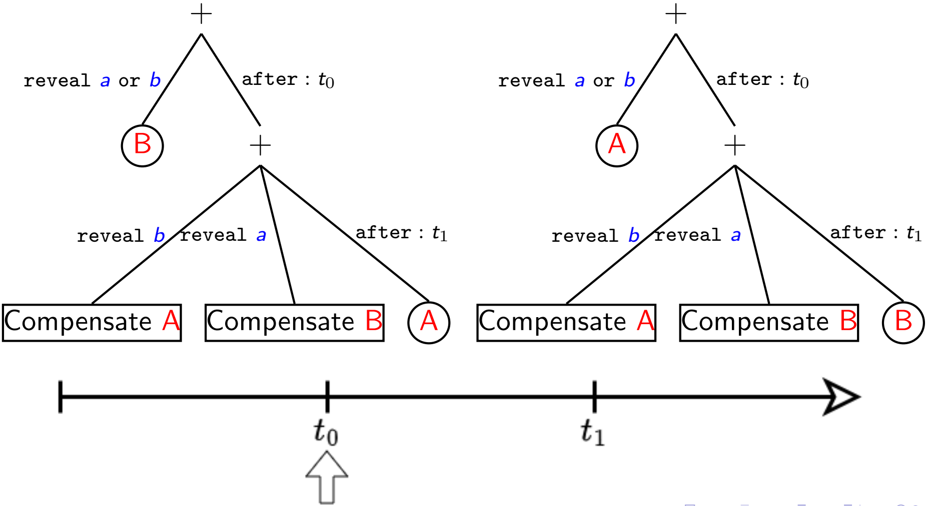
Alice has 1 ₿ and knows secret *a*

Bob has 1 Ð and knows secret *b*



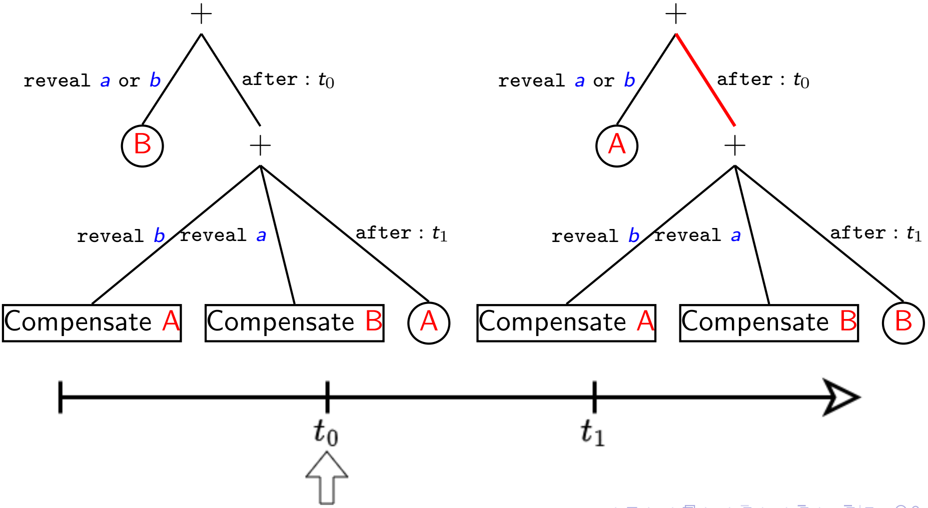
Improved Swap

Alice has 1 and knows secret a
Bob has 1 and knows secret b



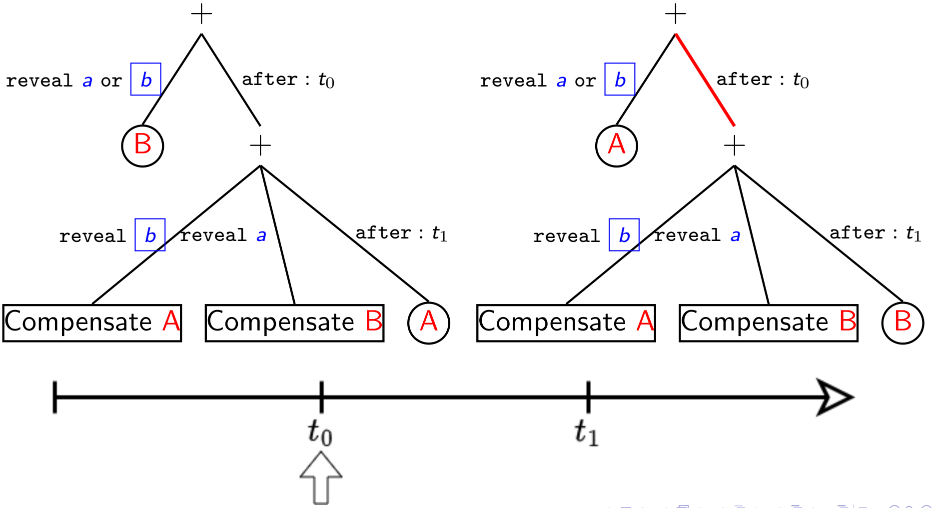
Improved Swap

Alice has 1 ₿ and knows secret a
Bob has 1 Ð and knows secret b



Improved Swap

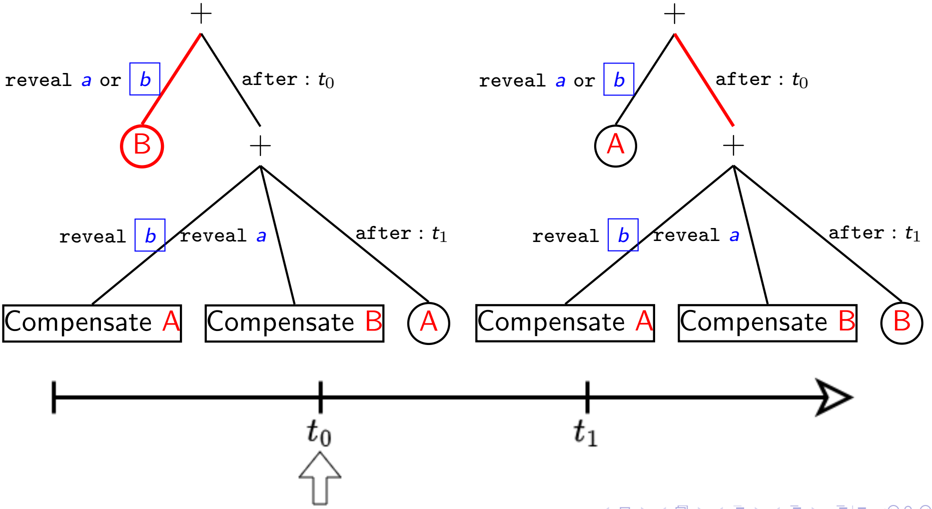
Alice has 1 ₿ and knows secret *a*
Bob has 1 Ð and knows secret *b*



Improved Swap

Alice has 1 and knows secret *a*

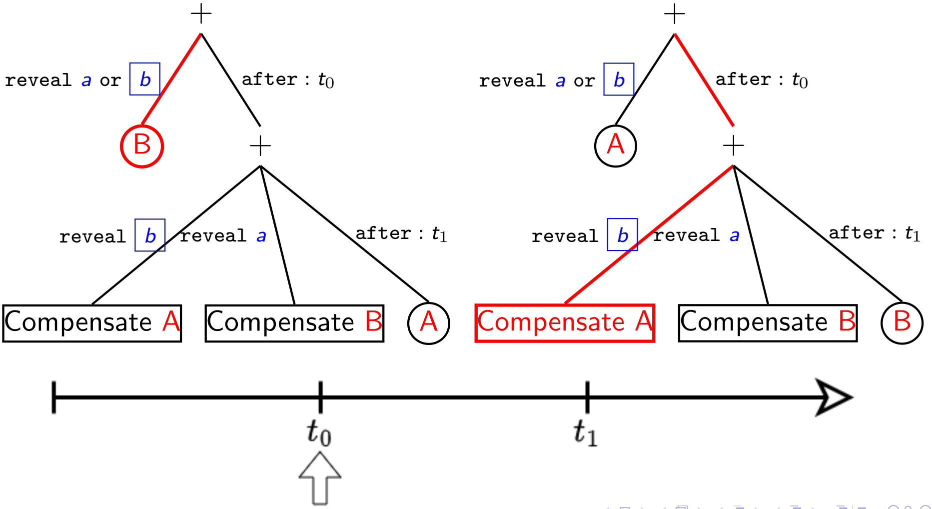
Bob has 1 and knows secret *b*



Improved Swap

Alice has 1 ₿ and knows secret *a*

Bob has 1 Ð and knows secret *b*



Swap In BitML^x

Alice should have read the bibliography on sound cryptographic protocol designs.

Swap In BitML^x

Alice should have ~~read the bibliography on sound cryptographic protocol designs.~~ switched to BitML^x!

$$\{A \;:!\, (1\text{B}, 0\text{Ɖ}) \mid B \;:!\, (0\text{B}, 1\text{Ɖ})\}\mathit{Swap}^x$$
$$\mathit{Swap}^x \; = \mathit{Exchange}^x \nrightarrow \mathit{Refund}^x$$

Swap In BitML^x

Alice should have ~~read the bibliography on sound cryptographic protocol designs.~~ switched to BitML^x!

$$\{A :! (1 Ƀ, 0 Đ) \mid B :! (0 Ƀ, 1 Đ)\} \mathit{Swap}^x$$
$$\mathit{Swap}^x = \mathit{Exchange}^x +\!\!> \mathit{Refund}^x$$

$$\mathit{Exchange}^x = \texttt{withdraw}($$
$$(0 Ƀ, 1 Đ) \rightarrow A,$$
$$(1 Ƀ, 0 Đ) \rightarrow B$$
$$)$$

Swap In BitML^x

Alice should have ~~read the bibliography on sound cryptographic protocol designs.~~ switched to BitML^x!

$$\{A :! (1Ƀ, 0Đ) \mid B :! (0Ƀ, 1Đ)\}\mathit{Swap}^x$$
$$\mathit{Swap}^x = \mathit{Exchange}^x \mathrel{+\!\!>} \mathit{Refund}^x$$

$$\mathit{Exchange}^x = \texttt{withdraw}((0Ƀ, 1Đ) \to A, (1Ƀ, 0Đ) \to B)$$

$$\mathit{Refund}^x = \texttt{withdraw}((1Ƀ, 0Đ) \to A, (0Ƀ, 1Đ) \to B)$$

BitML^x Compilation Idea

$$\mathit{Swap}^x \xrightarrow{???} \begin{cases} \vec{T}^x_{\text{₿}} \\ \vec{T}^x_{\text{Đ}} \end{cases}$$

BitML^x Compilation Idea

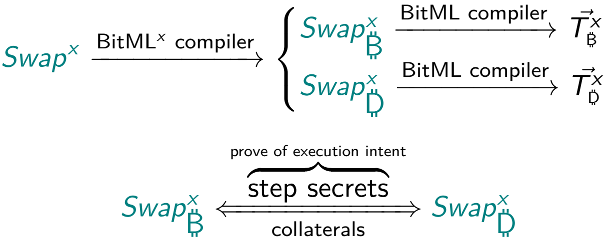
$$\mathit{Swap}^x \xrightarrow{\text{BitML}^x\text{ compiler}} \begin{cases} \mathit{Swap}^x_{₿} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{₿} \\ \mathit{Swap}^x_{Ð} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{Ð} \end{cases}$$

BitML^x Compilation Idea

$$\mathit{Swap}^x \xrightarrow{\text{BitML}^x\text{ compiler}} \begin{cases} \mathit{Swap}^x_{\text{₿}} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{\text{₿}} \\ \mathit{Swap}^x_{\text{Đ}} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{\text{Đ}} \end{cases}$$

$$\mathit{Swap}^x_{\text{₿}} \underset{\text{collaterals}}{\overset{\text{step secrets}}{\Longleftrightarrow}} \mathit{Swap}^x_{\text{Đ}}$$

BitML^x Compilation Idea



BitML^x Compilation Idea

$$\mathit{Swap}^x \xrightarrow{\text{BitML}^x\text{ compiler}} \begin{cases} \mathit{Swap}^x_{₿} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{₿} \\ \mathit{Swap}^x_{Ð} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{Ð} \end{cases}$$

$$\mathit{Swap}^x_{₿} \xLeftrightarrow[\underbrace{\text{collaterals}}_{\text{to compensate asynchronous behaviours}}]{\overbrace{\text{step secrets}}^{\text{to prove execution intent}}} \mathit{Swap}^x_{Ð}$$

User Strategies

Suppose Alice follows a strategy S_A^x s.t.

$$S_A^x(Swap^x) = Refund^x$$

User Strategies

Suppose Alice follows a strategy S^x_A s.t.

$$S^x_A(Swap^x) = Refund^x$$

$$Swap^x \xrightarrow{\text{BitML}^x \text{ compiler}} \begin{cases} Swap^x_{₿} \\ Swap^x_{Ð} \end{cases}$$

User Strategies

Suppose Alice follows a strategy S_A^x s.t.

$$S_A^x(Swap^x) = Refund^x$$

$$Swap^x \xrightarrow{\text{BitML}^x \text{ compiler}} \begin{cases} Swap_{฿}^x \\ Swap_{Đ}^x \end{cases}$$

$$S_A^x \xrightarrow{\text{and strategy compiler!}} \begin{cases} S_A^{฿} \\ S_A^{Đ} \end{cases}$$

User Strategies

Suppose Alice follows a strategy S_A^x s.t.

$$S_A^x(Swap^x) = Refund^x$$

$$Swap^x \xrightarrow{\text{BitML}^x \text{ compiler}} \begin{cases} Swap^x_{₿} \\ Swap^x_{Ð} \end{cases}$$

$$S_A^x \xrightarrow{\text{and strategy compiler!}} \begin{cases} S_A^{₿} \\ S_A^{Ð} \end{cases}$$

$$S_A^{₿}(Swap^x_{₿}) = \texttt{if revealed } b$$
$$\texttt{then } Compensate\ A$$
$$\texttt{else wait until } Refund^x_{₿}$$

Correctness

Theorem (Compiler correctness, informal)

Each strategy of an honest user A on a BitML^x contract C translates into a strategy on k concurrently executing compiled BitML contracts $C_{\mathbb{B}_1} | \ldots | C_{\mathbb{B}_k}$ that allows A to extract at least as many assets from $C_{\mathbb{B}_1} | \ldots | C_{\mathbb{B}_k}$ as from C with the original strategy.

Full BitML^x Syntax

$\mathrm{B} ::= [v_1\mathbb{B}_1, \ldots, v_k\mathbb{B}_k]$	balance
$G ::= A :! \mathrm{B}$	user deposit (in all chains)
$\mid A : \texttt{secret}\ s$	secret commitment
$C ::= D \mathrel{+\!\!>} C$	choose D or skip to C
$\mid \texttt{withdraw}\ \vec{\mathrm{B}} \to \vec{A}$	last choice is always `withdraw`
$D ::= \texttt{withdraw}\ \vec{\mathrm{B}} \to \vec{A}$	distribute the balance among users
$\mid \texttt{split}\ \vec{\mathrm{B}} \to \vec{C}$	split into many contract
$\mid \texttt{reveal}\ s\ \texttt{then}\ C$	reveal secrets before executing C
$\mid A :\ D$	A needs to authorize executing C

Thanks!

- BitML^x allows you to write cross-blockchain smart contracts.
- Compiled to concurrently executing BitML contracts.
- Proven security by mechanisms of step secrets and collaterals.
- PoC BitML^x compiler in Haskell.

Download slides
and PoC compiler:

Collaterals: How Much?

Every user locks, on each blockchain $\mathbb{B}$, an extra collateral deposit of value:

$$c_{\mathbb{B}} \;=\; b_{\mathbb{B}} \;\times\; (n-2)$$

where $b_{\mathbb{B}}$ is the contract balance in that blockchain, and n is the number of participants.

Adversarial Scheduling

<u>BitML</u>

$C_1 + \cdots + C_k$

- Users can always execute a valid option.
- Guaranteed to meet deadlines.
- In case of many valid options, adversary decides.

<u>BitML^x</u>

$C_1^x \mathrel{+\!\!>} \ldots \mathrel{+\!\!>} C_k^x$

- Only one valid option at a time.
- Round-based execution.
- Users can act before a subcontract is skipped.