Robust MPC: Asynchronous Responsiveness yet Synchronous Security

Chen-Da Liu-Zhang — ETH Zurich

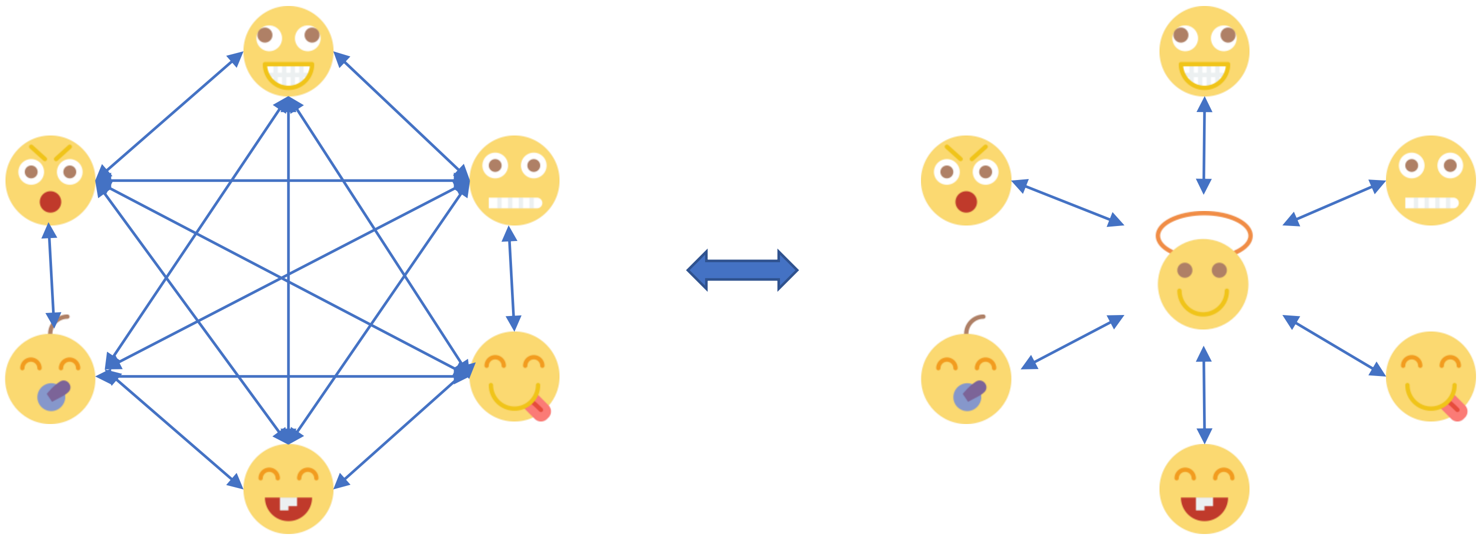
Julian Loss — RUB

Ueli Maurer — ETH Zurich

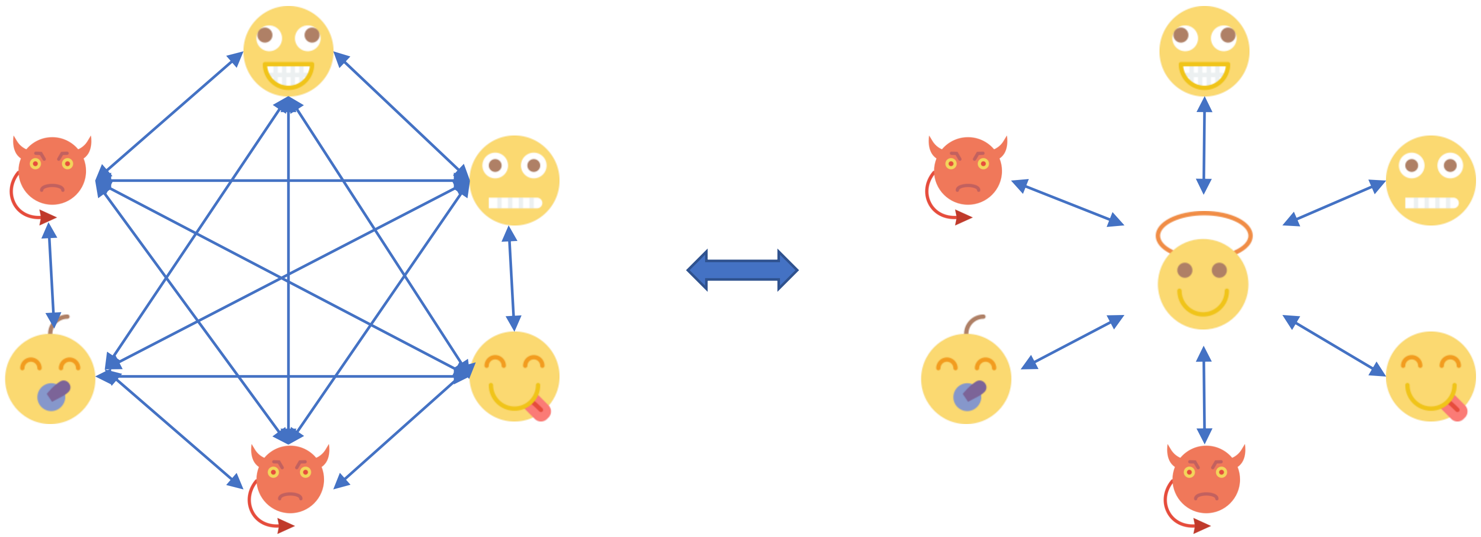
Tal Moran — IDC Herzliya

Daniel Tschudi — Aarhus University

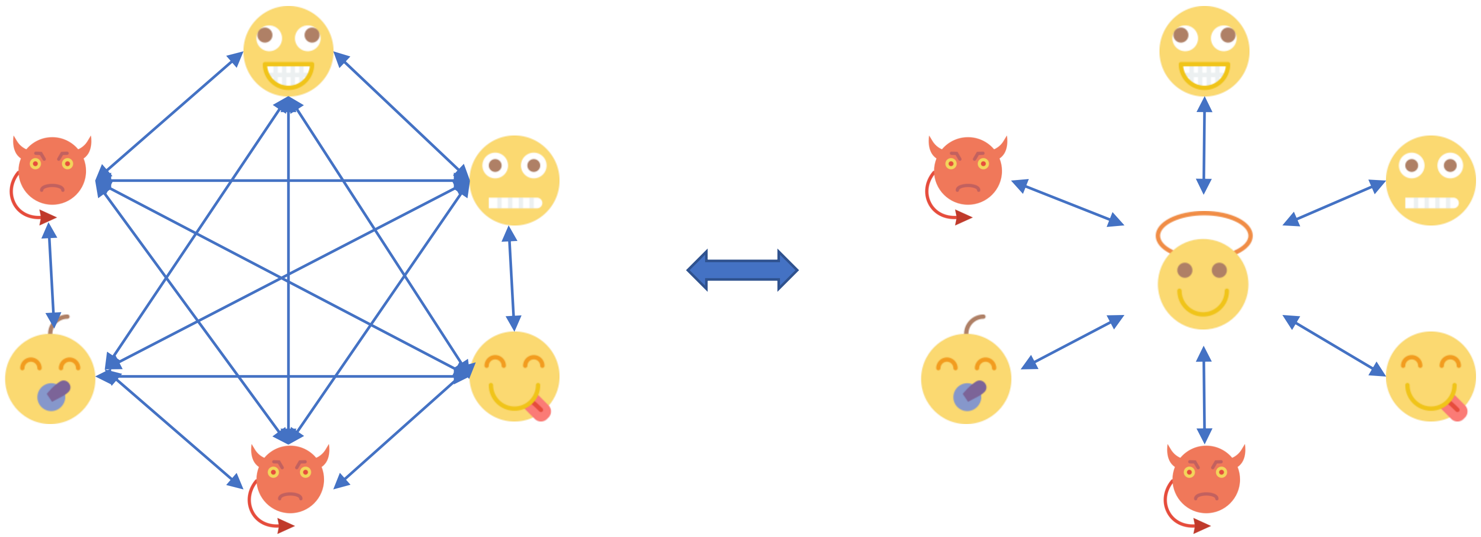
Multiparty Computation



Multiparty Computation



Multiparty Computation



Correctness: Output is correct
Privacy: Inputs remain private
Termination: Parties obtain output

Synchronous Model

Round structure:

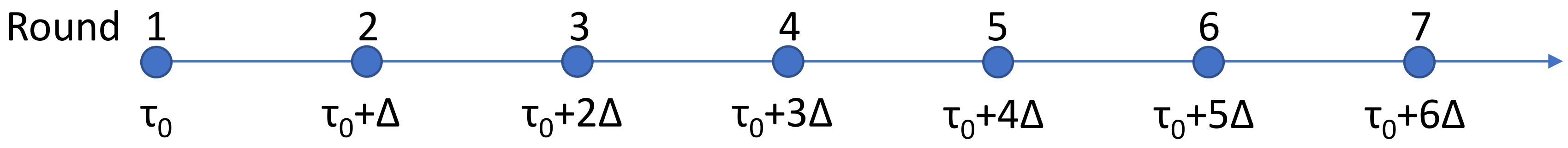
- Each P_i knows the current round
- Round r: P_i reads round r-1 messages
  P_i computes/sends round r messages
- **Round r messages are guaranteed to be delivered by round r+1**

Synchronous Model

Round structure:

- Each P_i knows the current round
- Round r: P_i reads round r-1 messages
  P_i computes/sends round r messages
- **Round r messages are guaranteed to be delivered by round r+1**

Can be achieved with synchronized clocks and channels with known delay upper bound Δ



Synchronous Protocols

$t < \frac{n}{2}$

Input completeness

Synchronous Protocols

$t < \frac{n}{2}$

Input completeness

Time: T(Δ)

$\Delta \gg \delta$, for network delay δ

Asynchronous Protocols

Greedy approach: P_i continues as soon as it gets enough messages

Asynchronous Protocols

Greedy approach: P_i continues as soon as it gets enough messages

Time: T(δ)

δ network delay

Asynchronous Protocols

Greedy approach: P_i continues as soon as it gets enough messages

Time: T(δ)

δ network delay

$t < \frac{n}{3}$

Take into account $n - t$ inputs

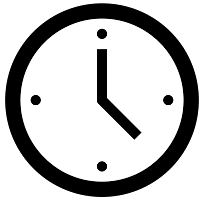
Synchronous vs Asynchronous

	Correctness	Privacy	Termination		Number of Inputs
			Slow Output T(Δ)	Fast Output T(δ)	
Synchronous	$n/2$	$n/2$	n	-	n
Asynchronous	$n/3$	$n/3$	-	$n/3$	$2n/3$

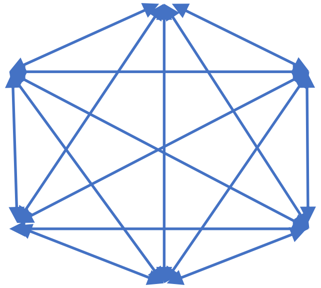
Synchronous vs Asynchronous

	Correctness	Privacy	Termination		Number of Inputs
			Slow Output T(Δ)	Fast Output T(δ)	
Synchronous	$n/2$	$n/2$	n	-	n
Asynchronous	$n/3$	$n/3$	-	$n/3$	$2n/3$
Our work	$n/2$	$n/2$	n	$n/4$	n or $3n/4$

Model



Global clock
[KMTZ13]



Network
unknown delay δ

Time out

Corrupt up to t parties arbitrarily
Schedule messages arbitrarily within δ clock ticks

Protocols can use $\Delta \gg \delta$

Synchronous over δ-network

Each P_i appends to each message its round number
Upon receiving all messages from round r:

- Send messages for round r+1
- Notify every party with round r

Upon receiving notifications from all parties for round r, continue with round r+1

Synchronous over δ-network

Each P_i appends to each message its round number
Upon receiving all messages from round r:

- Send messages for round r+1
- Notify every party with round r

Upon receiving notifications from all parties for round r, continue with round r+1

Problem: Corrupted P_j does not send a message, parties need to wait Δ clock ticks to deduce P_j is corrupted

Synchronous over δ-network

Each P_i appends to each message its round number

Upon receiving all messages from round r:

- Send messages for round r+1
- Notify every party with round r

Upon receiving notifications from all parties for round r, continue with round r+1

Problem: Corrupted P_j does not send a message, parties need to wait Δ clock ticks to deduce P_j is corrupted

Single corruption prevents Fast outputs!

Guarantees

Guarantees

Correctness and Privacy
$^{n}/_{2}$

Guarantees

Correctness and Privacy
$^{n}/_{2}$

Output guarantees depend on 

Guarantees

Correctness and Privacy
$n/2$

Output guarantees depend on

Fast network &
$t \le \frac{n}{4}$

Guarantees

Correctness and Privacy
$^{n}/_{2}$

Output guarantees depend on ⏰

Fast network &
$t \leq \frac{n}{4}$

Guarantees

Correctness and Privacy
$^{n}/_{2}$

Output guarantees depend on 

Fast network &
$t \leq \frac{n}{4}$

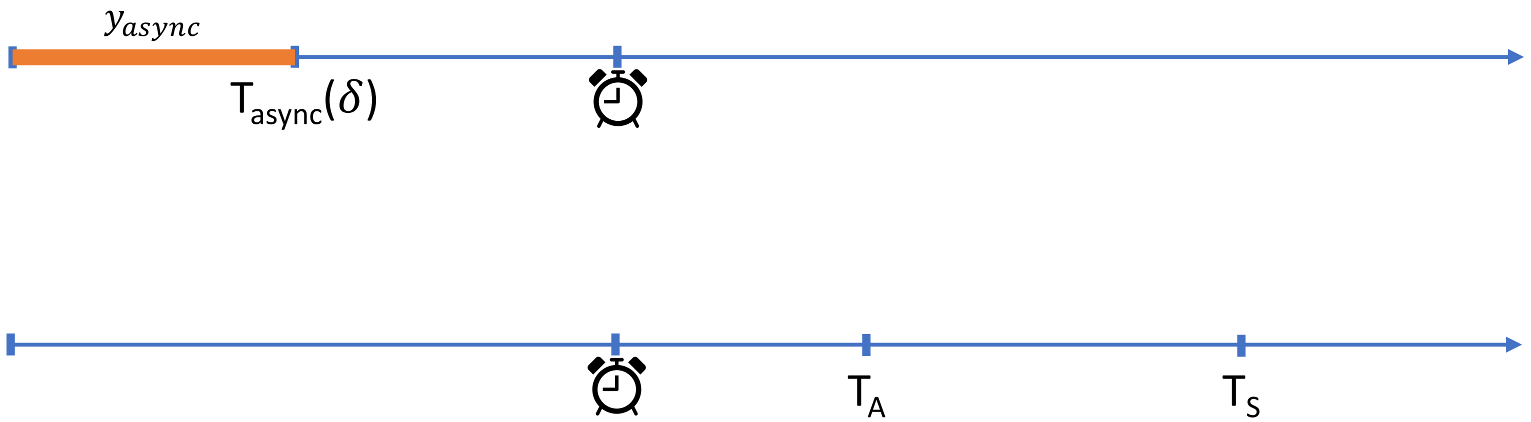
Slow network or
$t > \frac{n}{4}$

Guarantees

Correctness and Privacy
$^{n}/_{2}$

Output guarantees depend on 

Fast network &
$t \leq \frac{n}{4}$

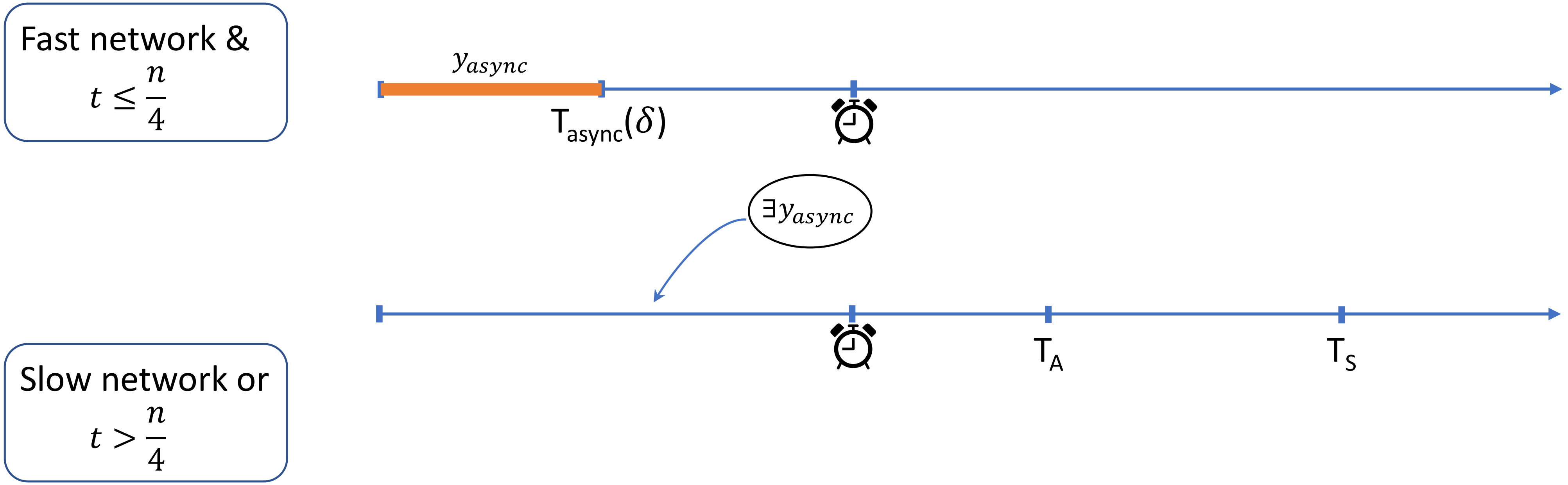


Slow network or
$t > \frac{n}{4}$

Guarantees

Correctness and Privacy
$n/2$

Output guarantees depend on ⏰



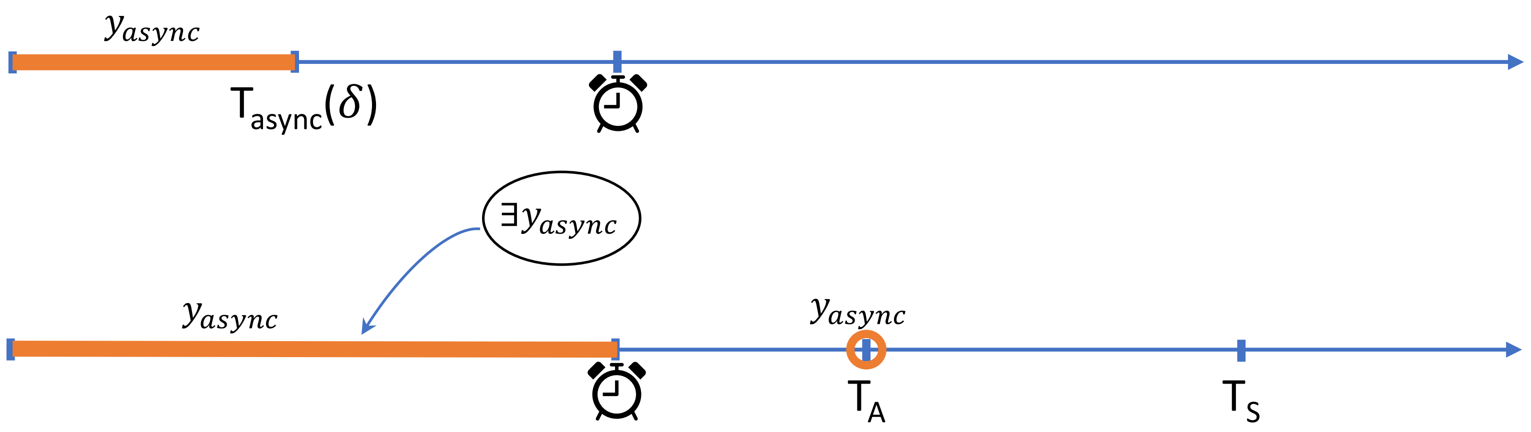
Guarantees

Correctness and Privacy
$^n/_2$

Output guarantees depend on ⏰

Fast network &
$t \leq \frac{n}{4}$

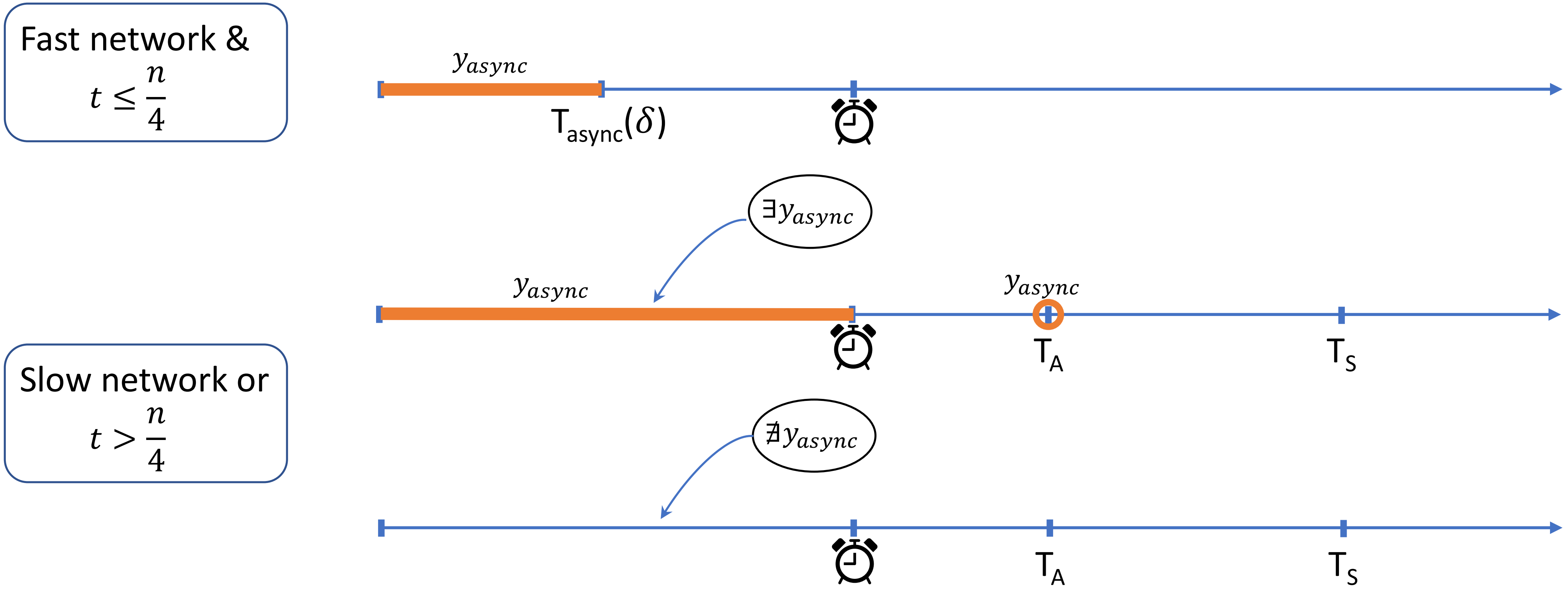
Slow network or
$t > \frac{n}{4}$



Guarantees

Correctness and Privacy
$^{n}/_{2}$

Output guarantees depend on ⏰



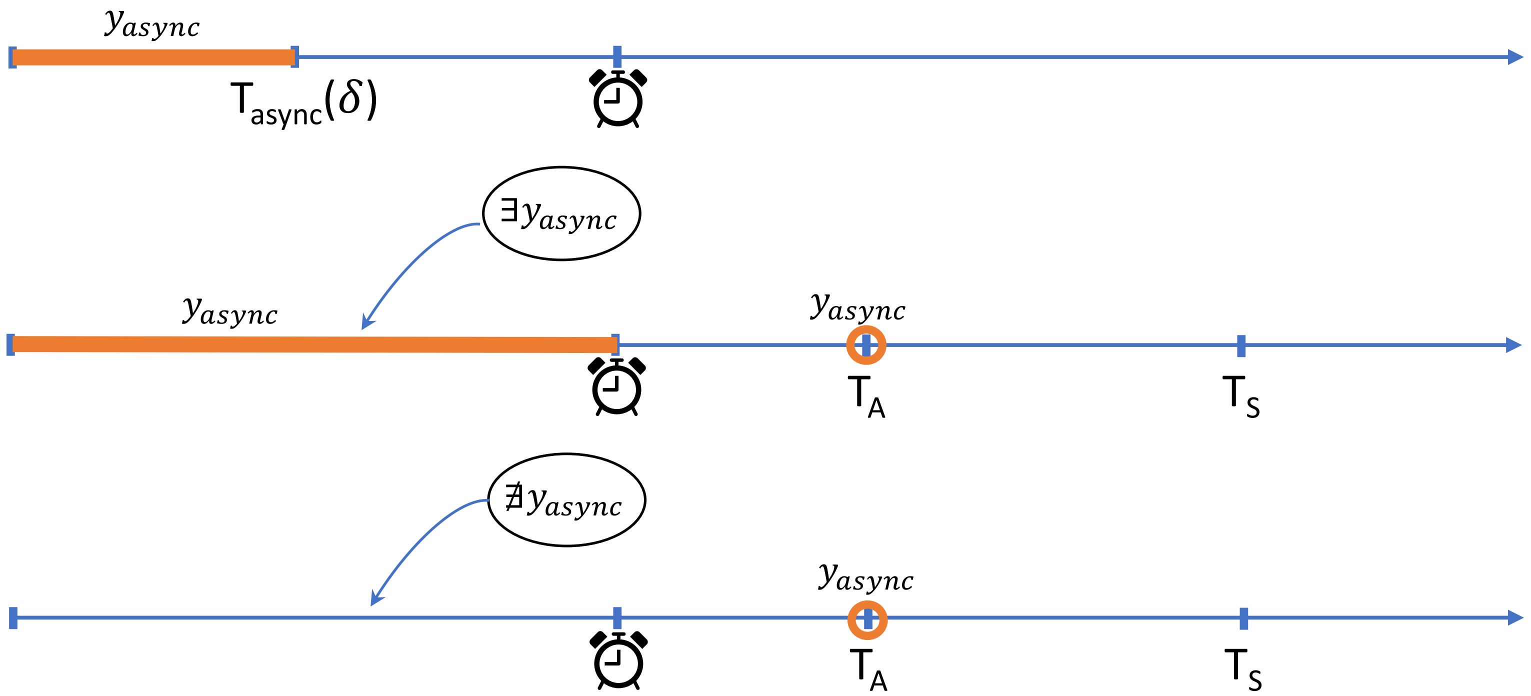
Guarantees

Correctness and Privacy
$^{n}/_{2}$

Output guarantees depend on ⏰

Fast network &
$t \leq \frac{n}{4}$

Slow network or
$t > \frac{n}{4}$



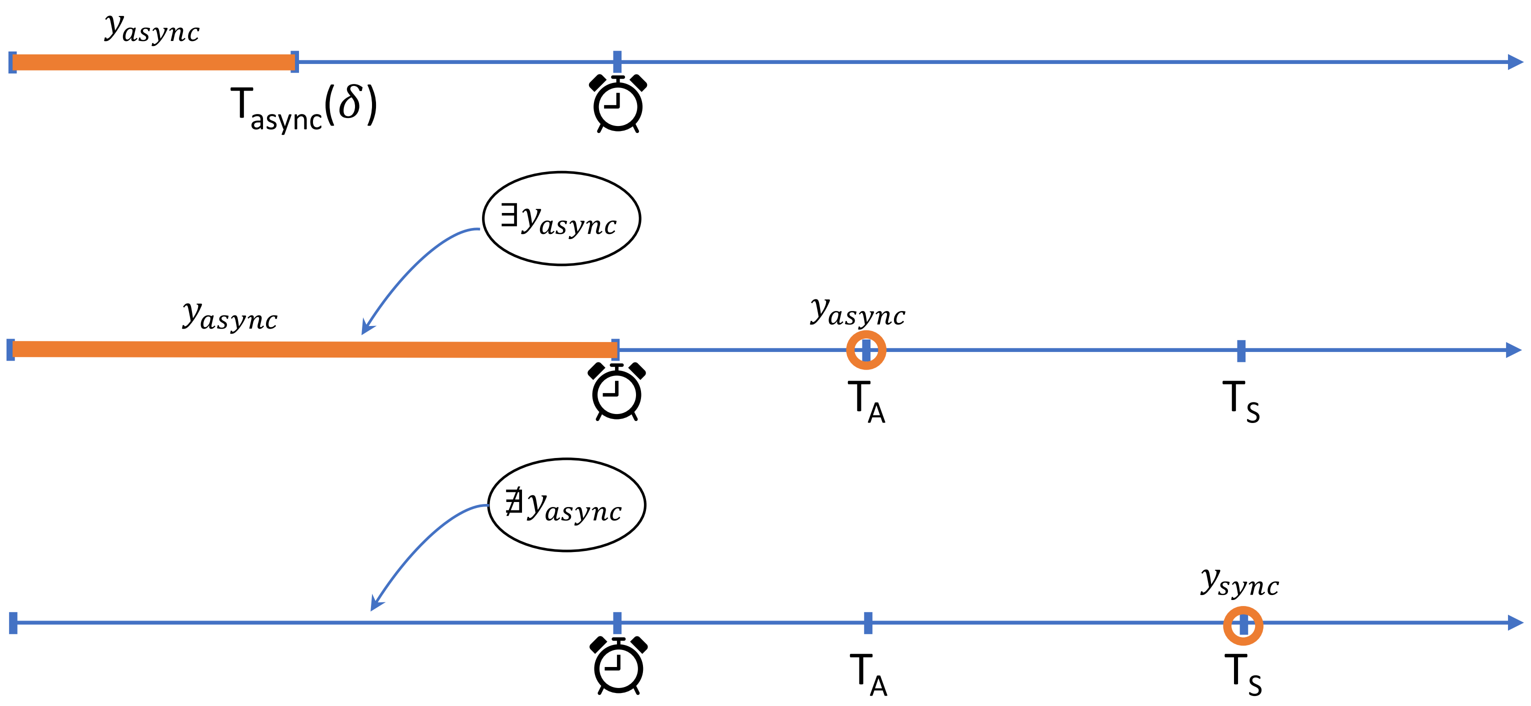
Guarantees

Correctness and Privacy
$^{n}/_{2}$

Output guarantees depend on ⏰

Fast network & $t \le \frac{n}{4}$

Slow network or $t > \frac{n}{4}$



Road map

Π_{ASYNC}

Correctness/Privacy: $\frac{n}{2}$

Fast Output: $\frac{n}{4}$

Road map

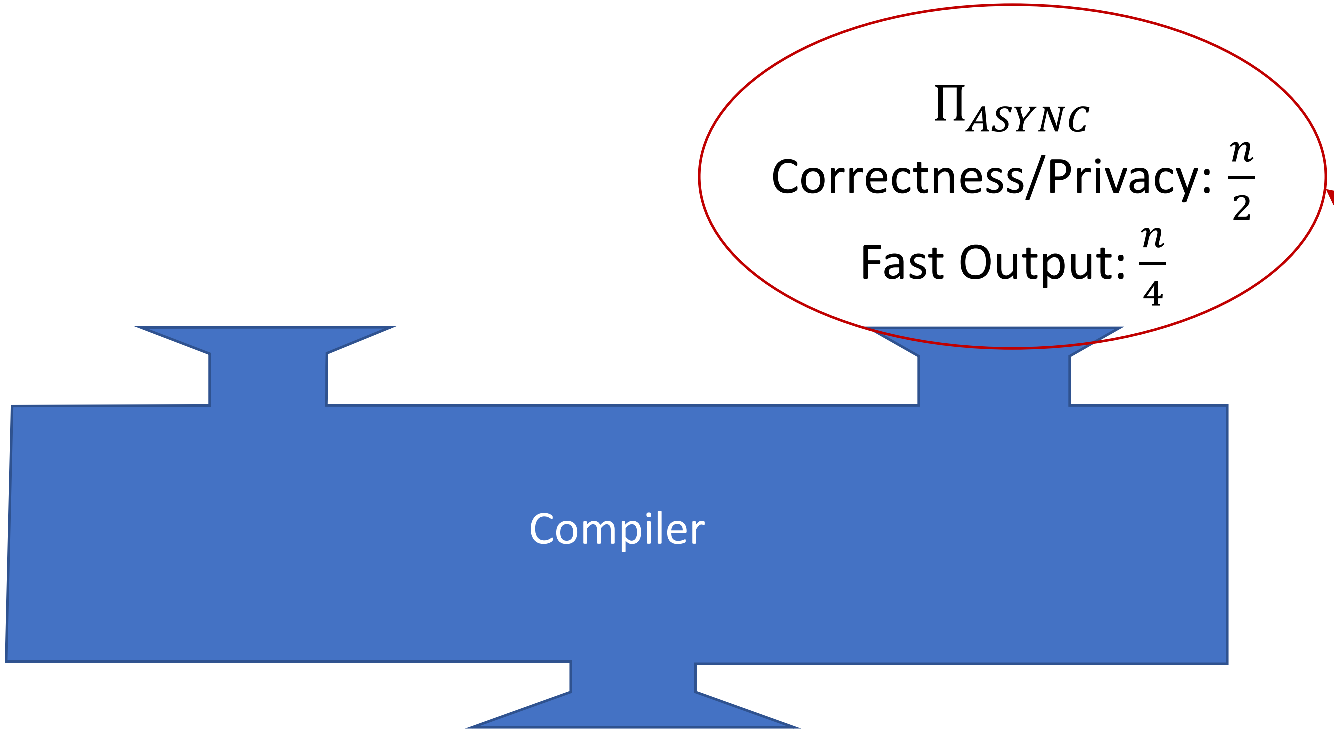
Π_{ASYNC}

Correctness/Privacy: $\frac{n}{2}$

Fast Output: $\frac{n}{4}$

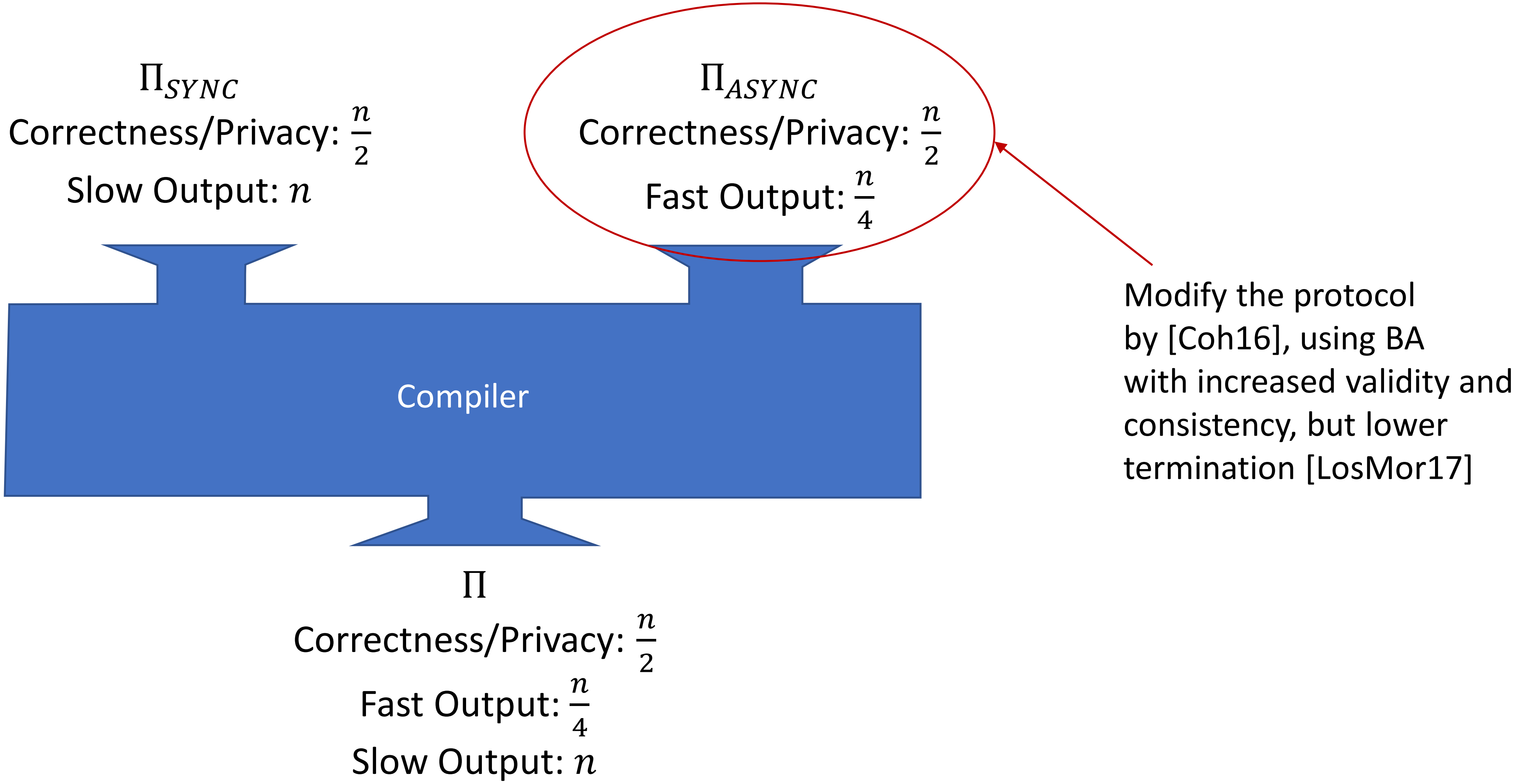
Modify the protocol
by [Coh16], using BA
with increased validity and
consistency, but lower
termination [LosMor17]

Road map

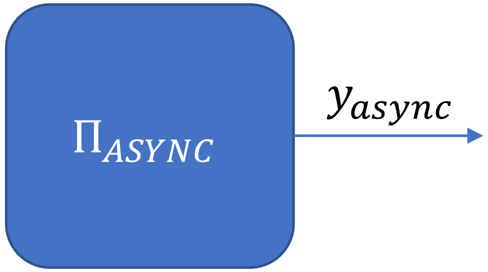


Modify the protocol
by [Coh16], using BA
with increased validity and
consistency, but lower
termination [LosMor17]

Road map

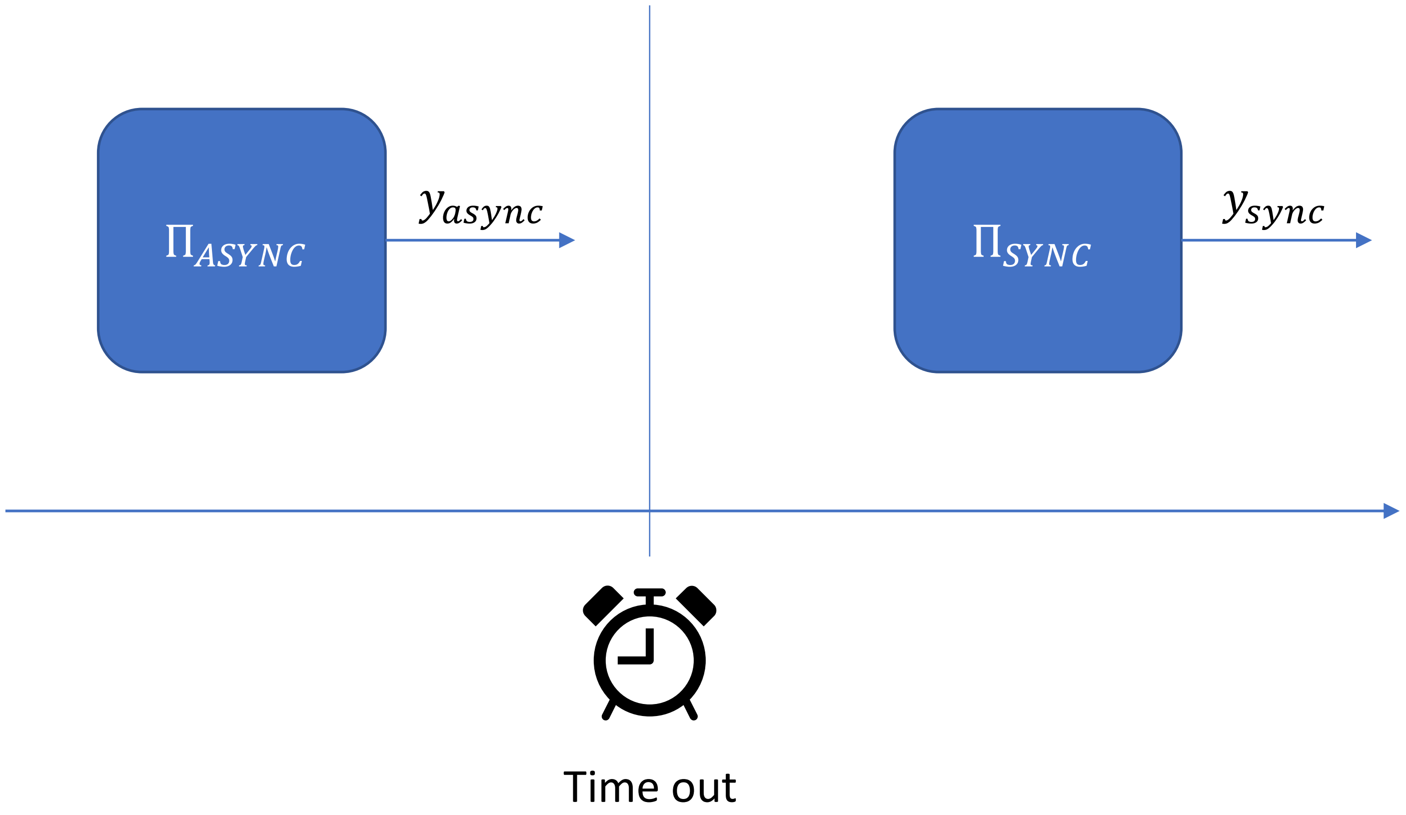

Π_{SYNC}
Correctness/Privacy: $\frac{n}{2}$
Slow Output: n
Π_{ASYNC}
Correctness/Privacy: $\frac{n}{2}$
Fast Output: $\frac{n}{4}$
Compiler
Modify the protocol by [Coh16], using BA with increased validity and consistency, but lower termination [LosMor17]
Π
Correctness/Privacy: $\frac{n}{2}$
Fast Output: $\frac{n}{4}$
Slow Output: n

Compiler



Compiler

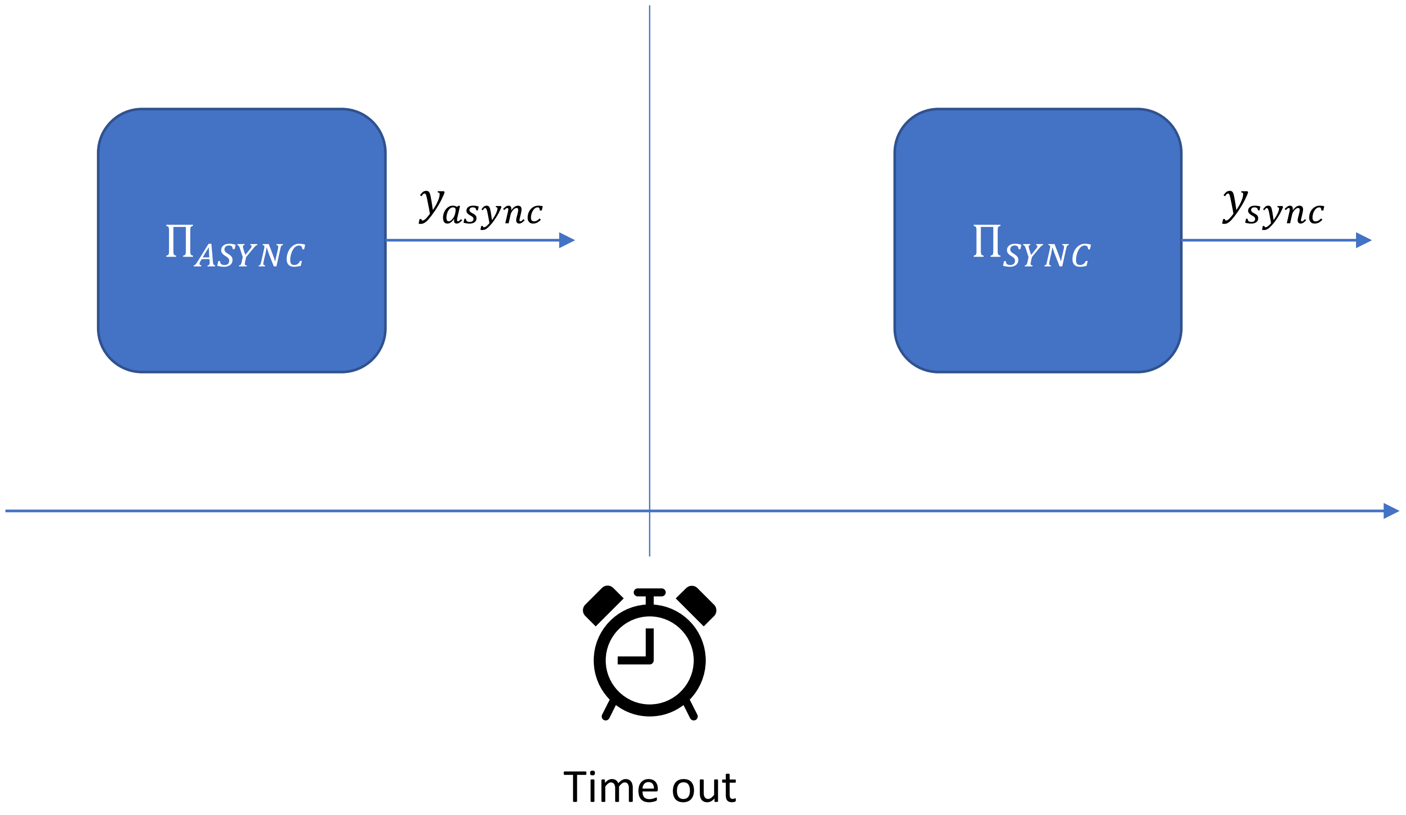
Use an asynchronous protocol and a synchronous protocol as fall back*



*[PasShi17,LosMor17,GPS19]

Compiler

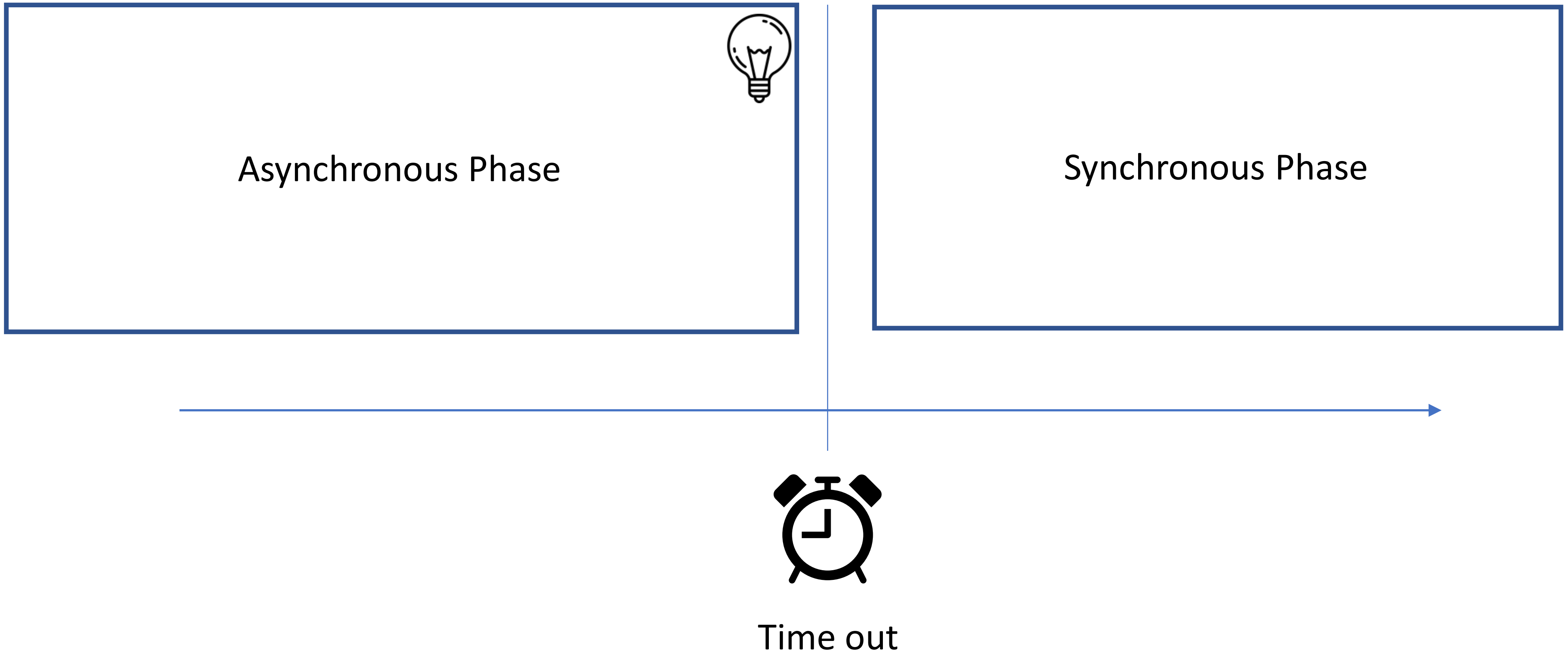
Use an asynchronous protocol and a synchronous protocol as fall back*



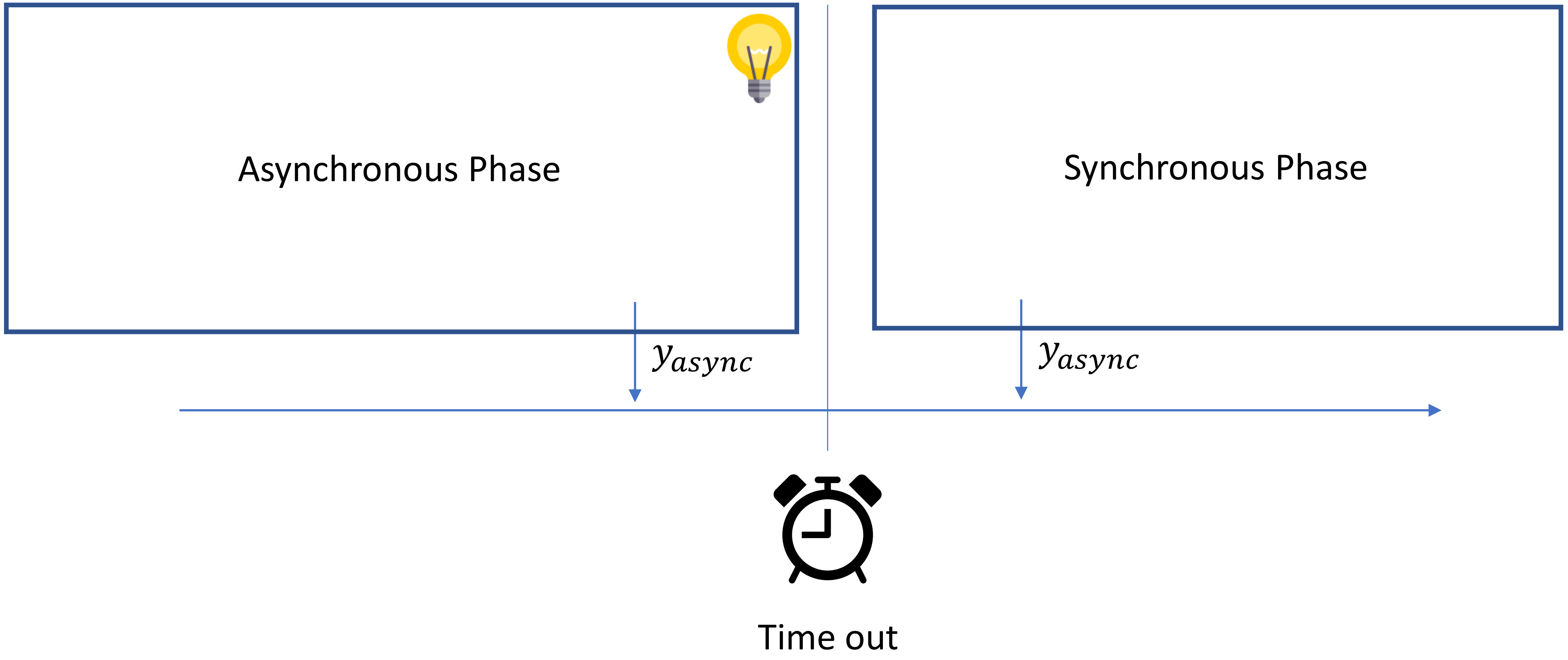
Problem:
Adversary can learn two outputs, y_{async} and y_{sync}!

*[PasShi17,LosMor17,GPS19]

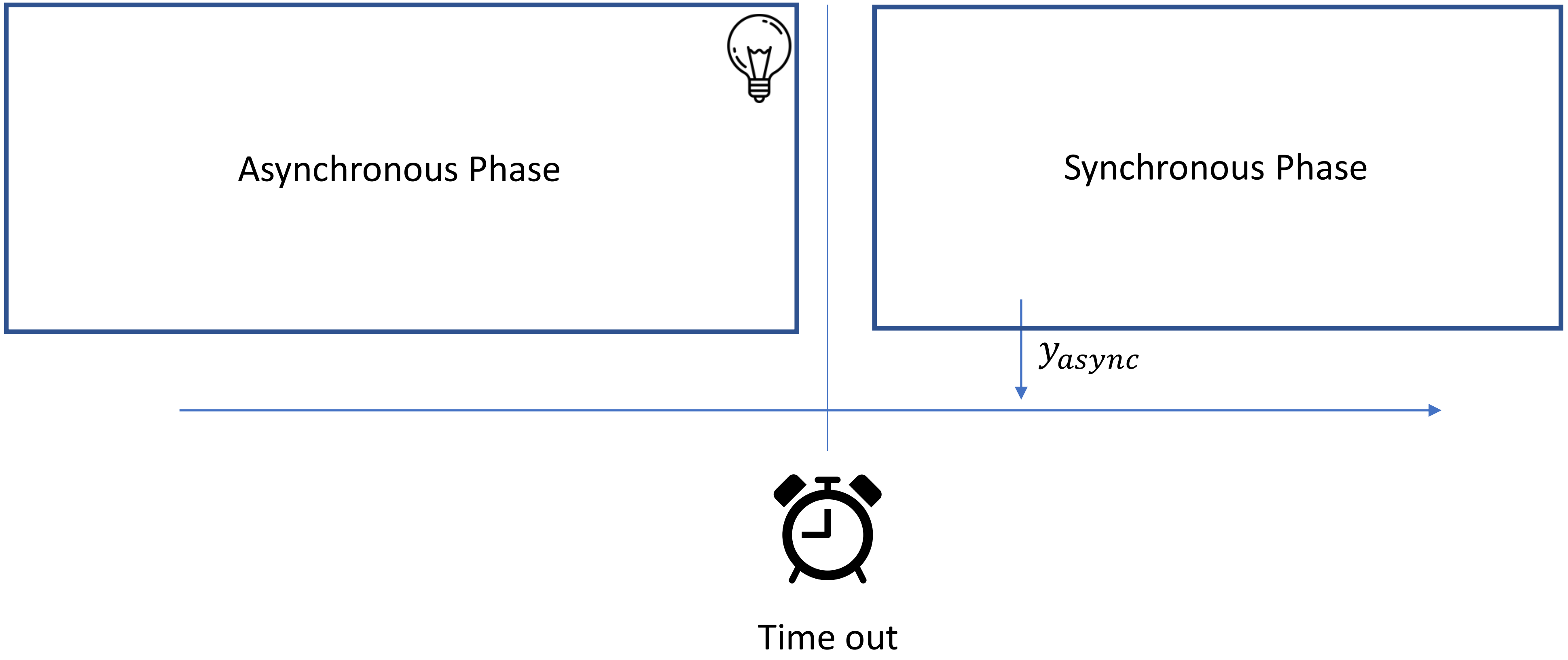
Compiler



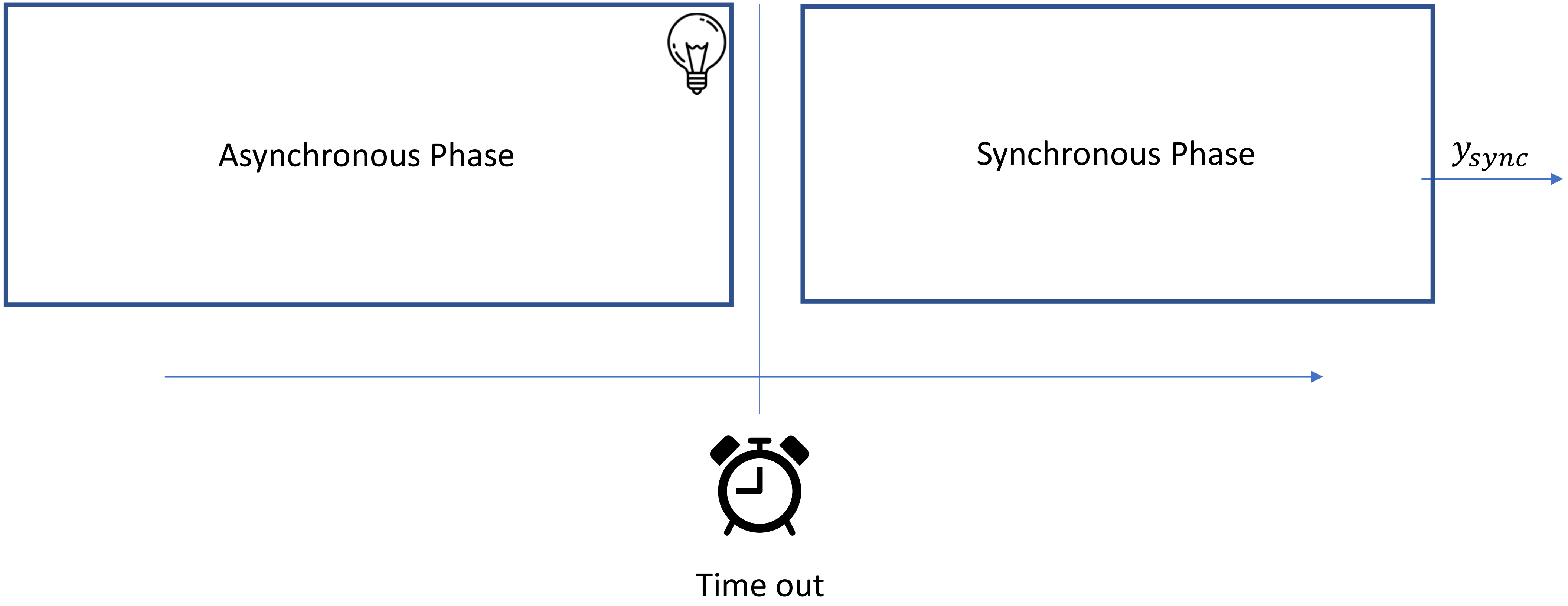
Compiler


Asynchronous Phase
Synchronous Phase
y_{async}
y_{async}
Time out

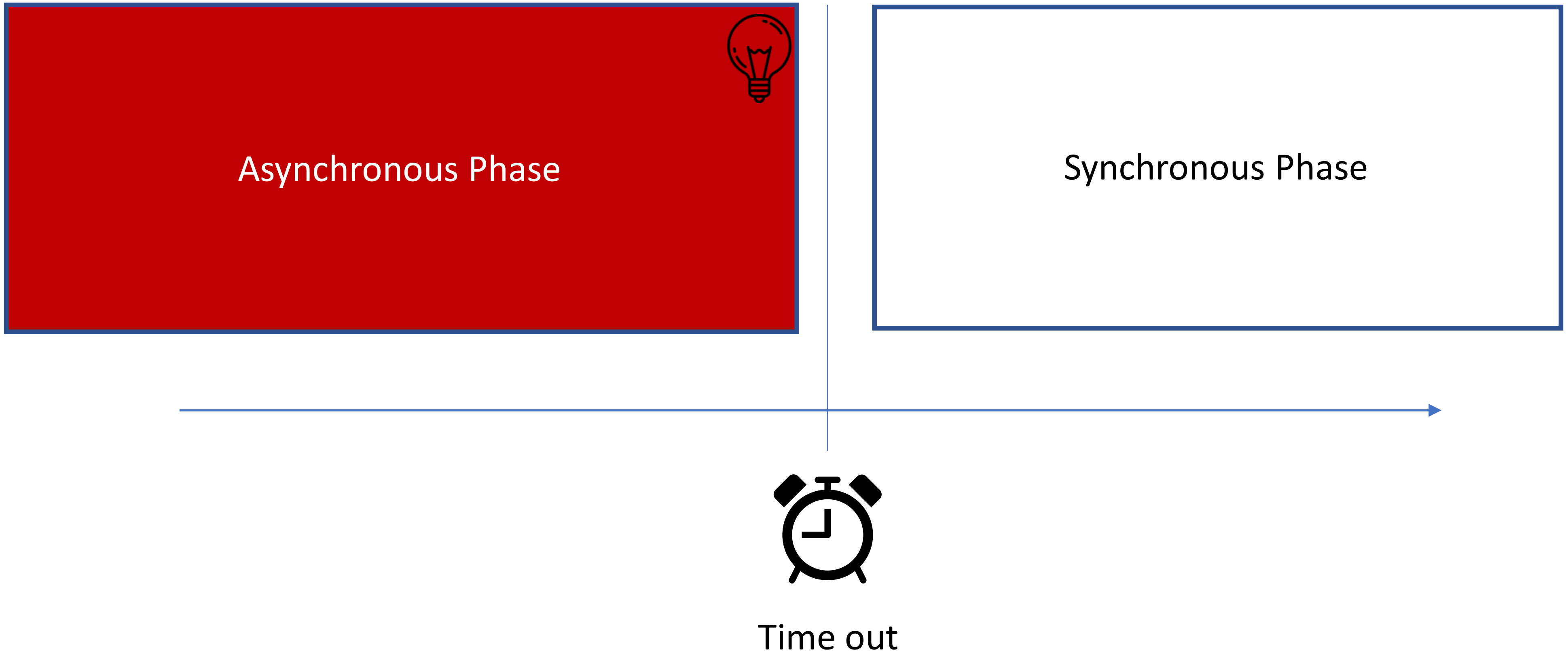
Compiler



Compiler

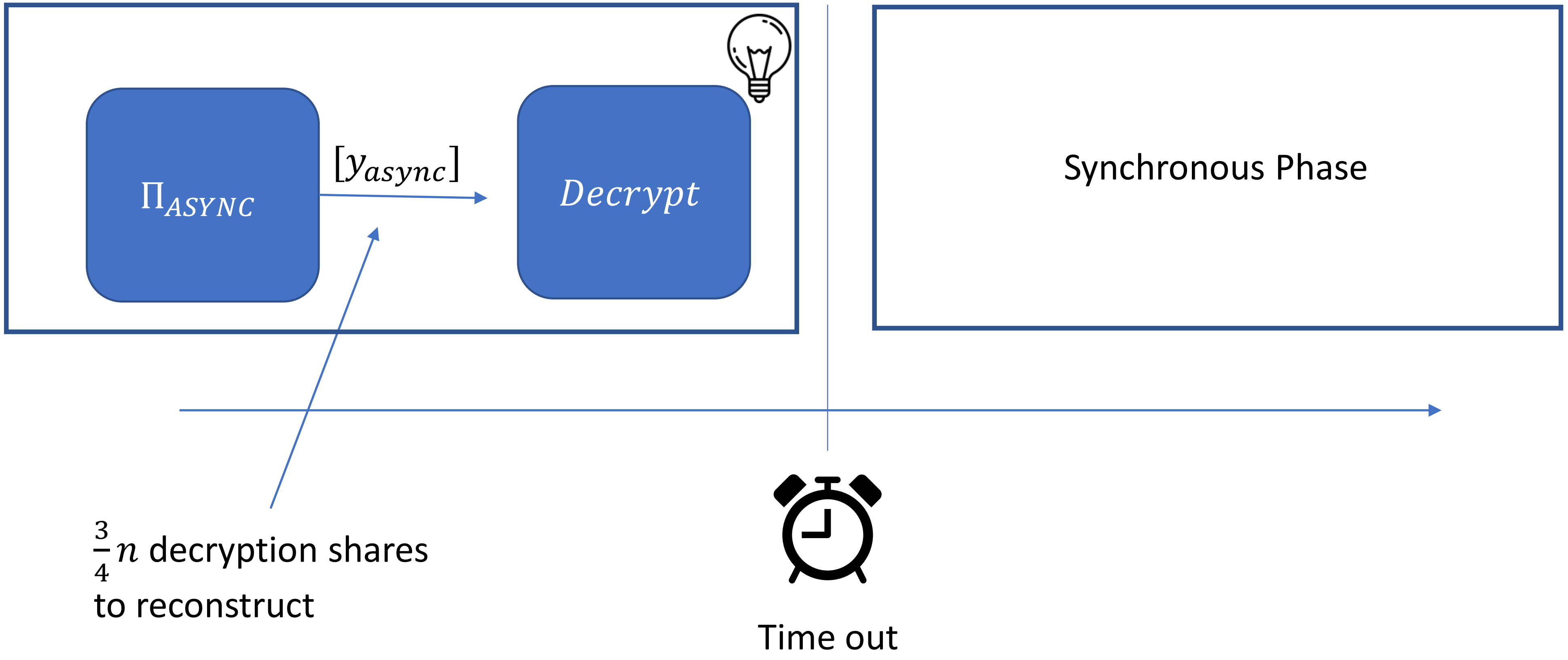


Compiler



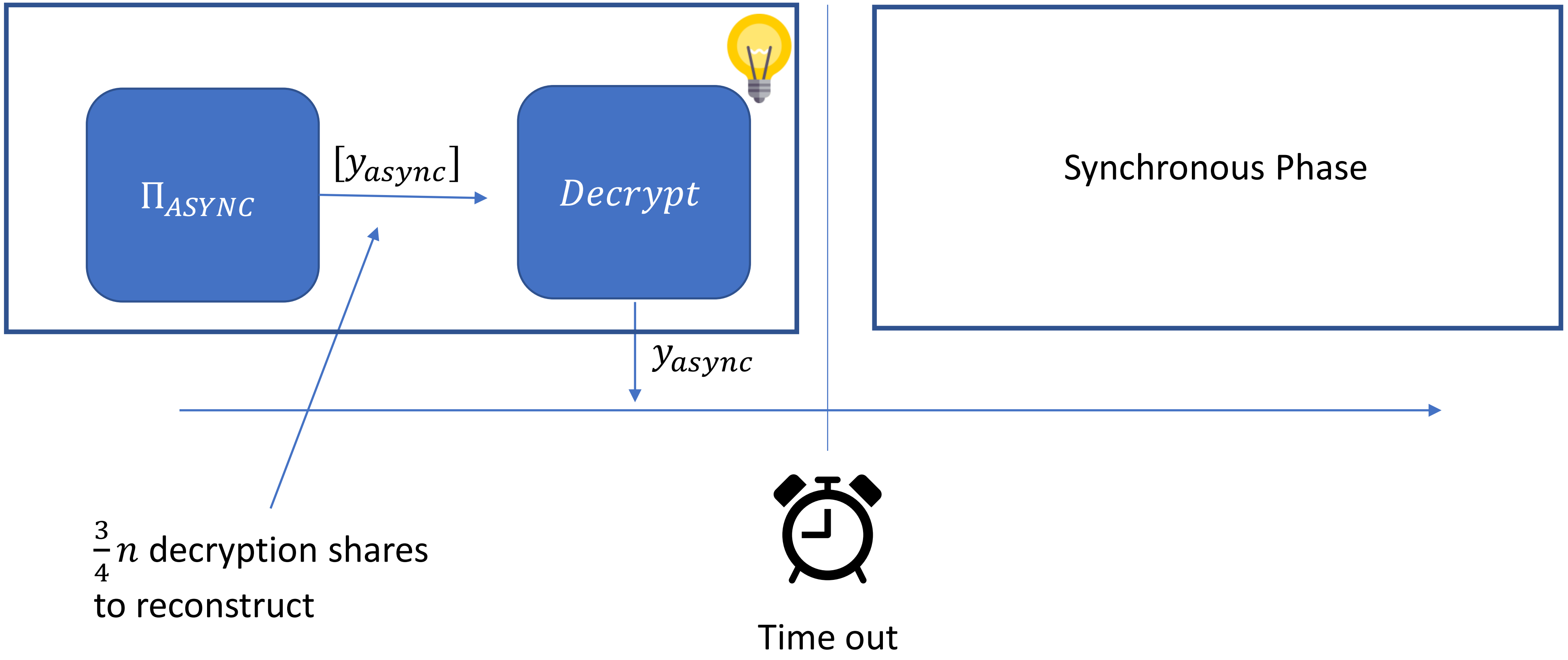
Compiler

Setup: Threshold encryption and digital signatures



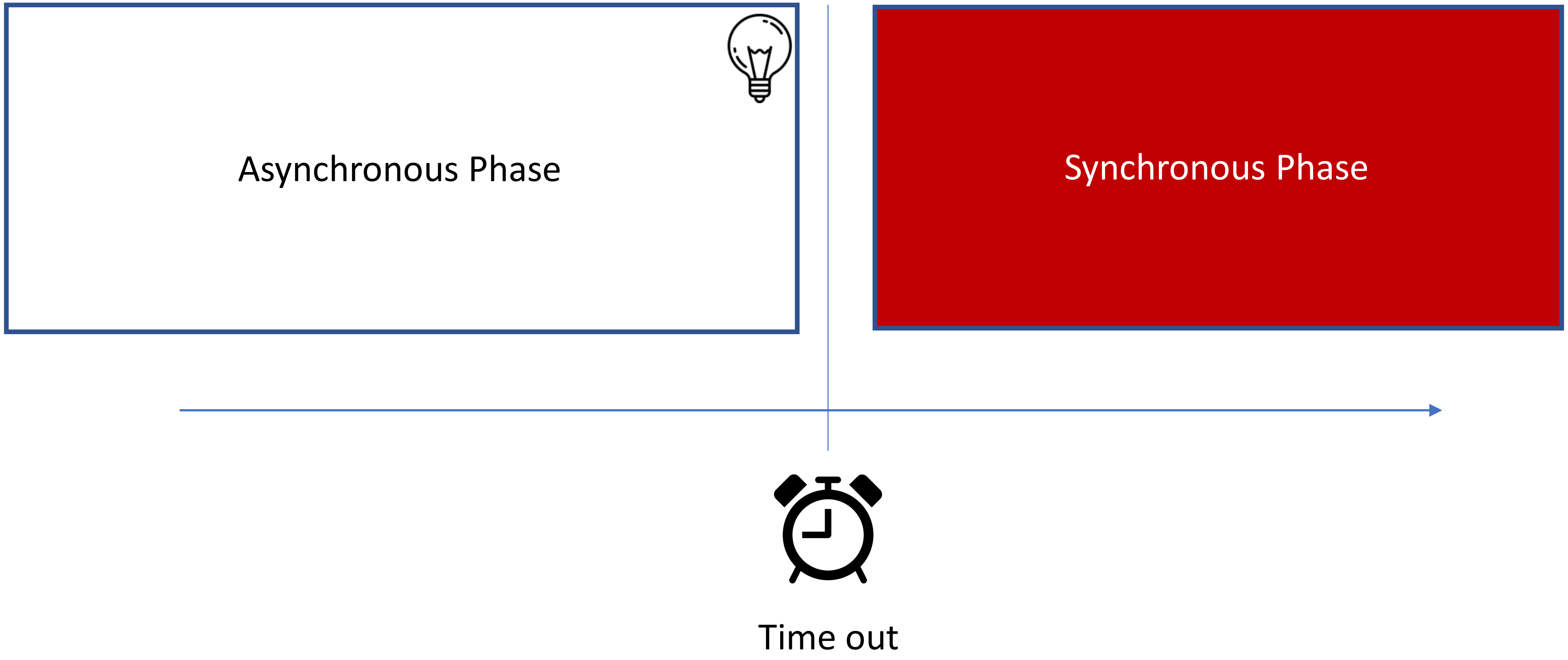
Compiler

Setup: Threshold encryption and digital signatures



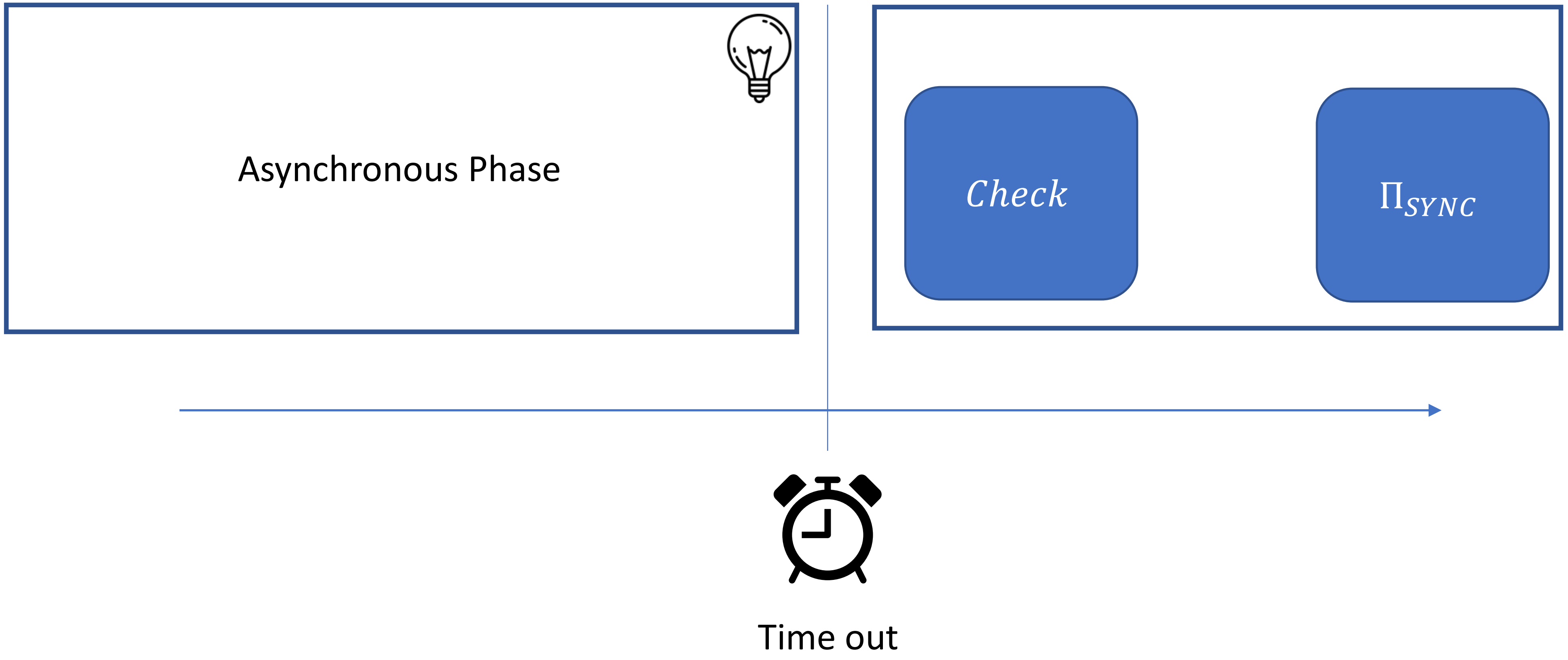
Compiler

Setup: Threshold encryption and digital signatures



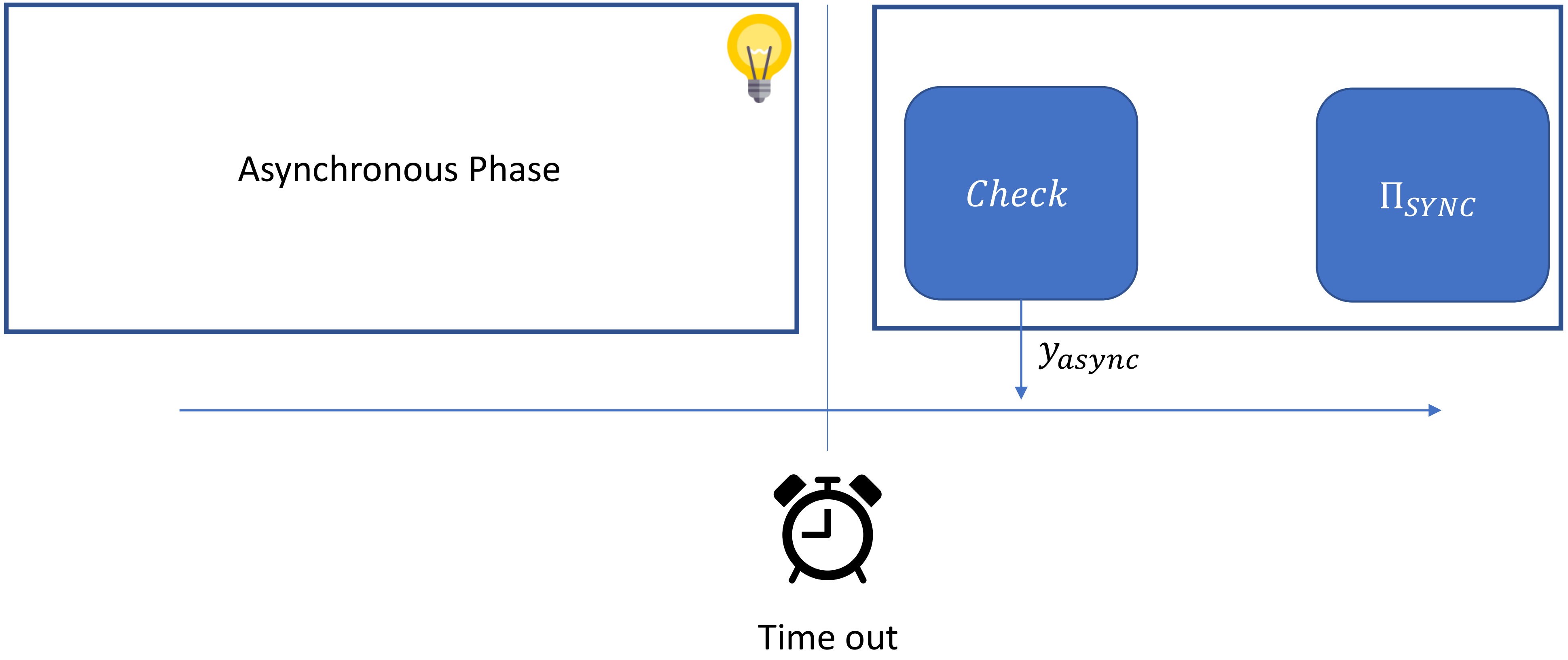
Compiler

Setup: Threshold encryption and digital signatures



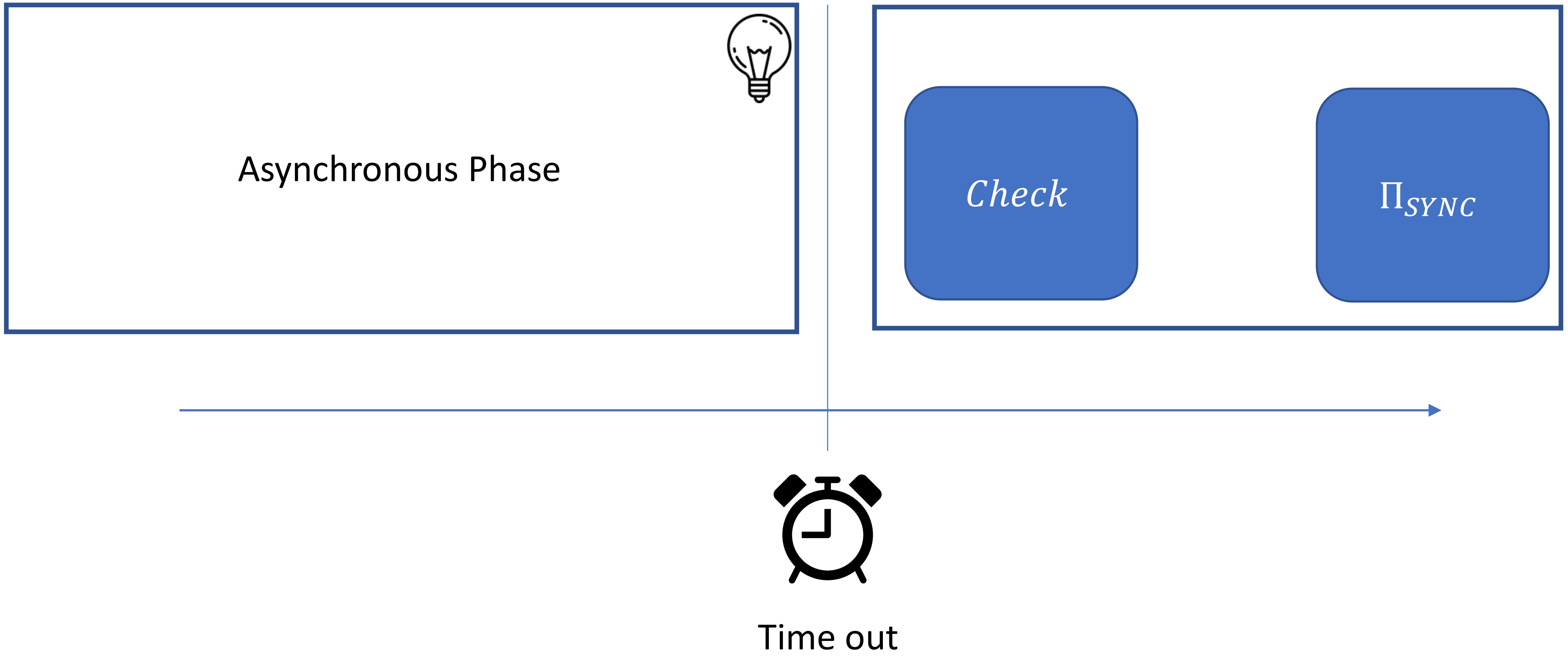
Compiler

Setup: Threshold encryption and digital signatures



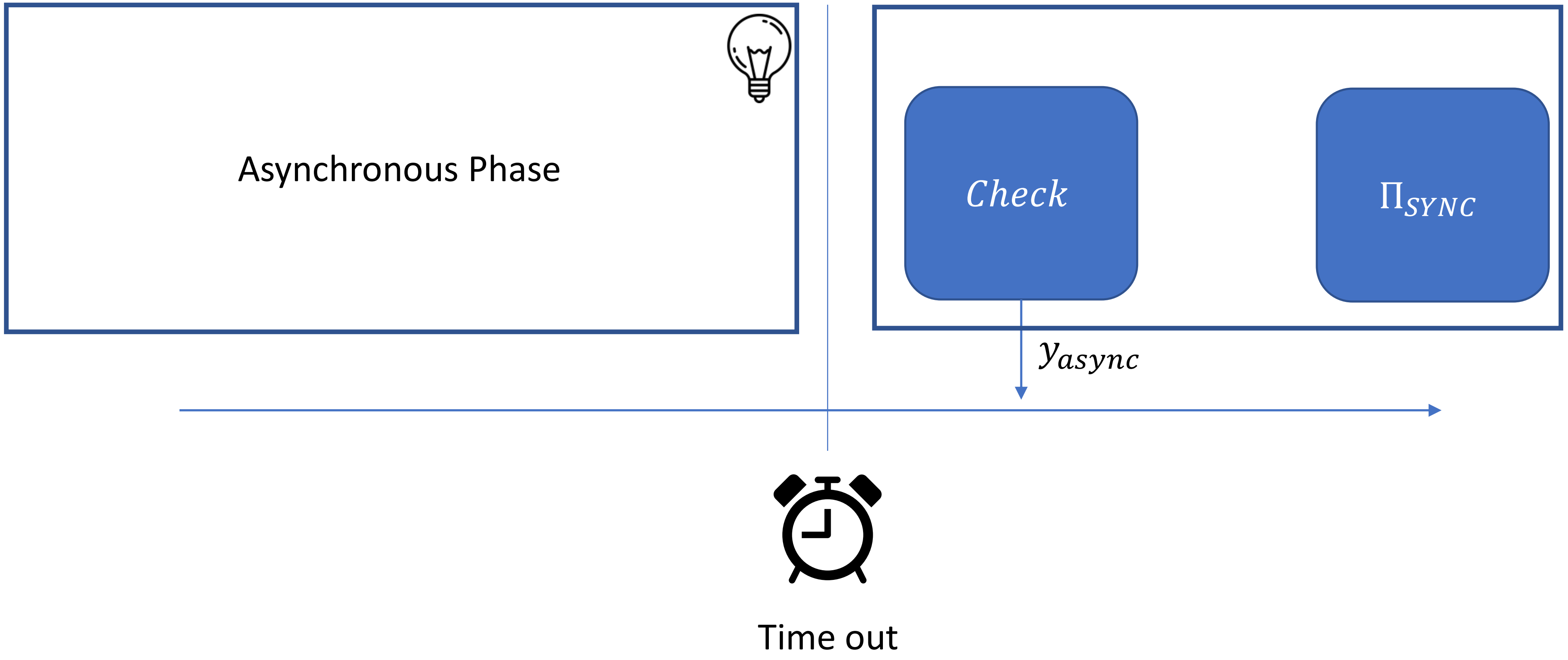
Compiler

Setup: Threshold encryption and digital signatures



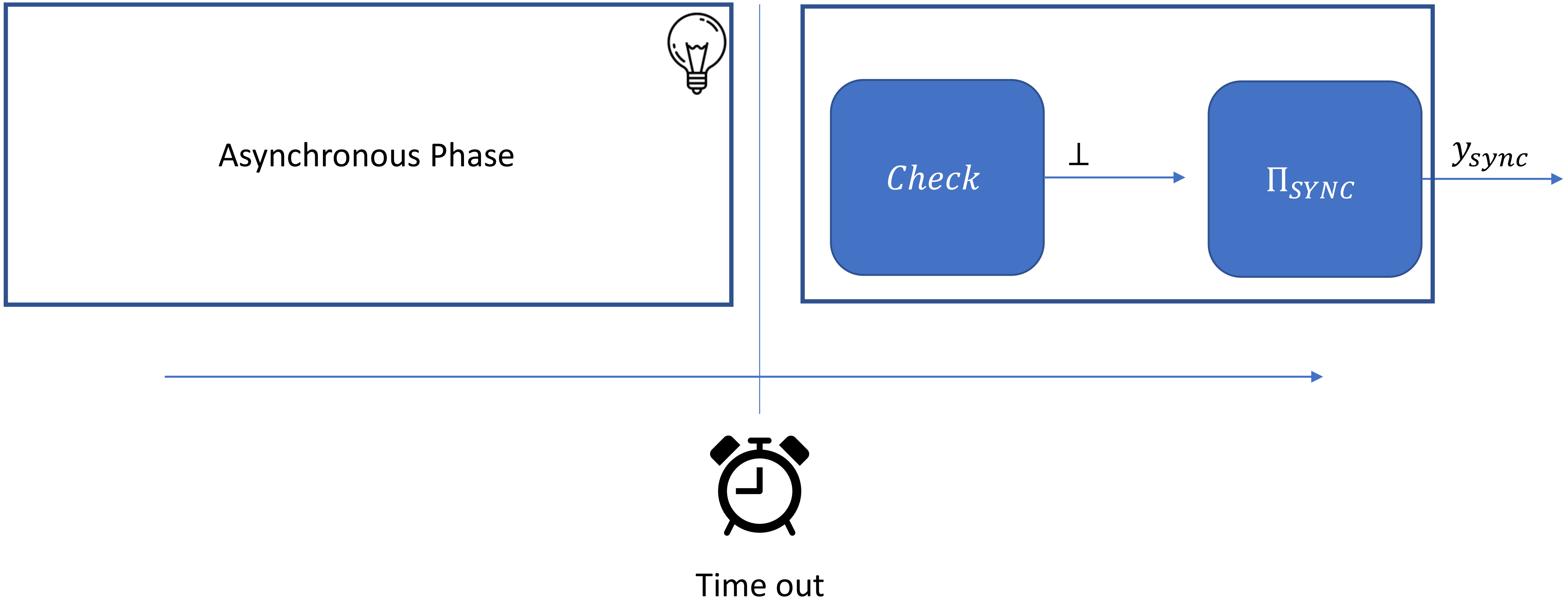
Compiler

Setup: Threshold encryption and digital signatures



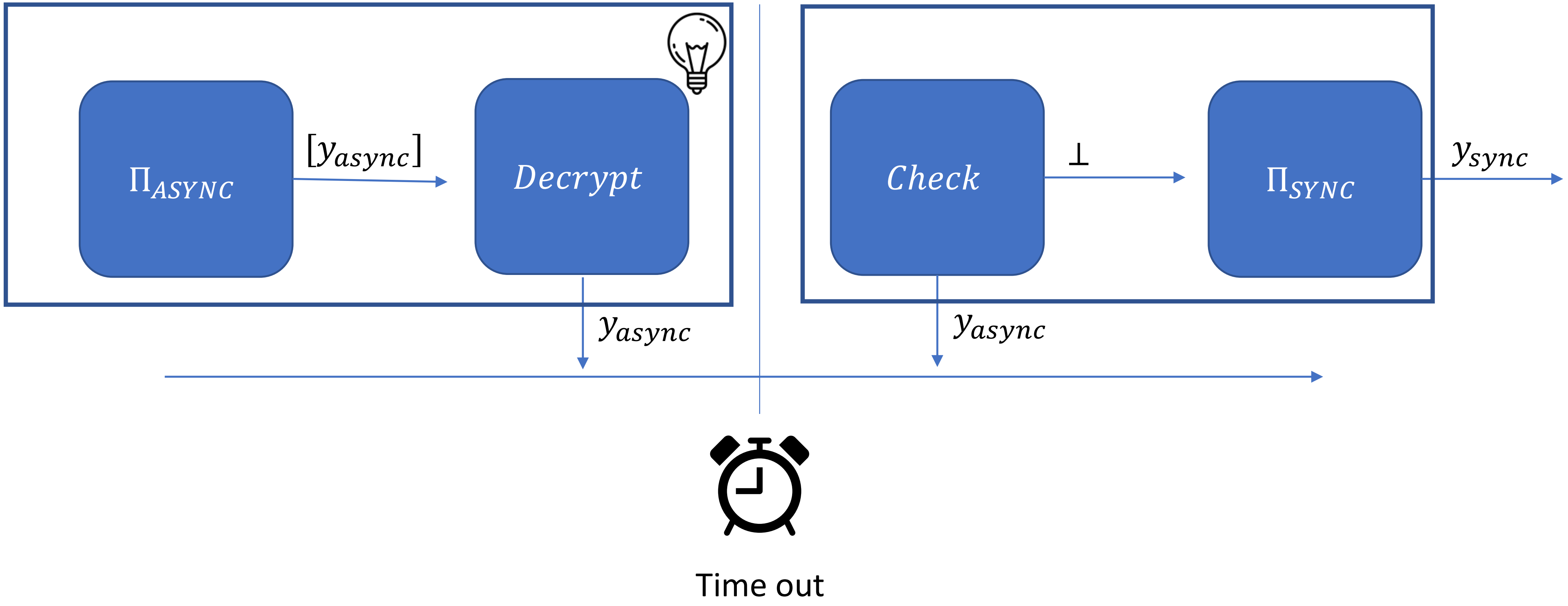
Compiler

Setup: Threshold encryption and digital signatures



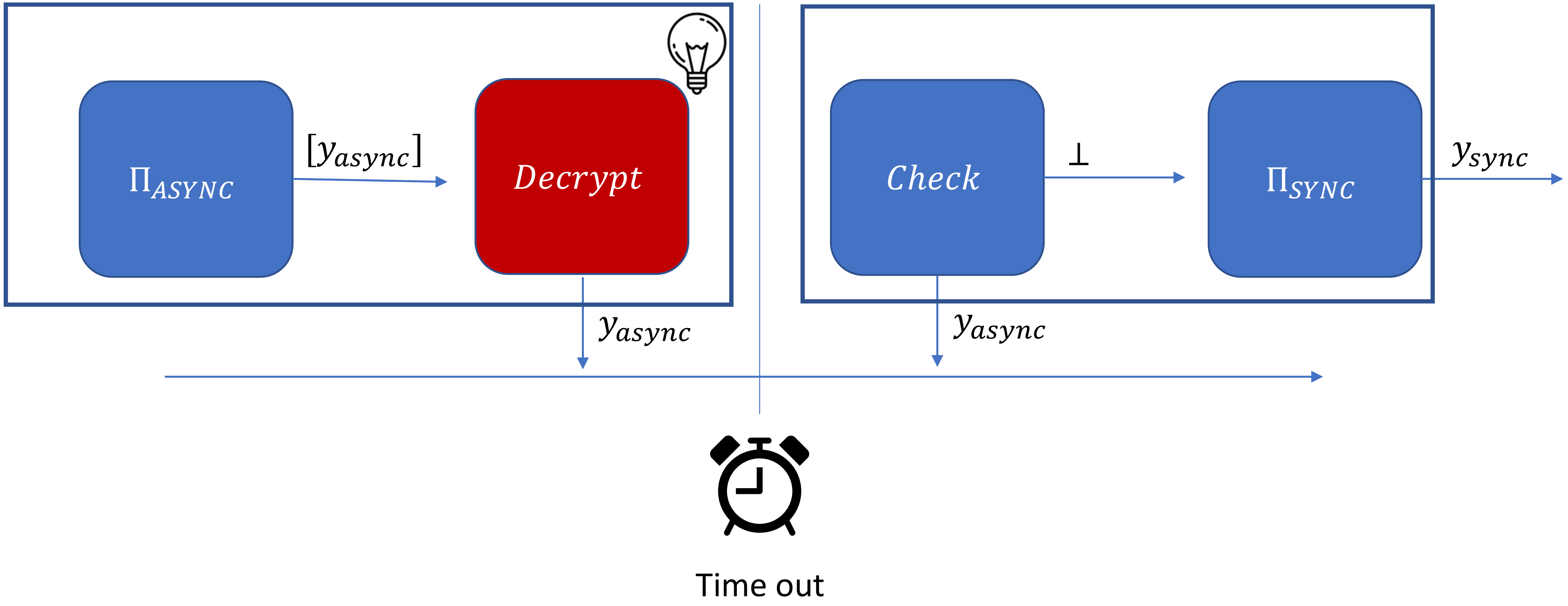
Compiler

Setup: Threshold encryption and digital signatures

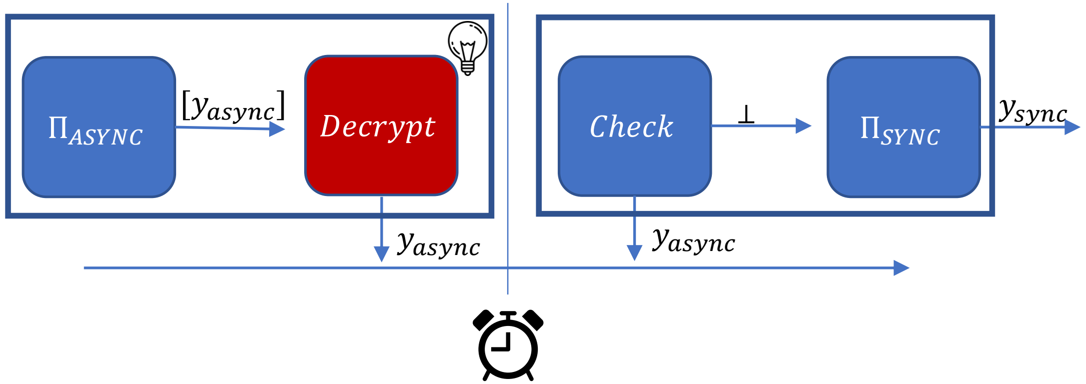


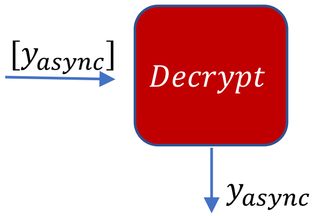
Compiler

Setup: Threshold encryption and digital signatures



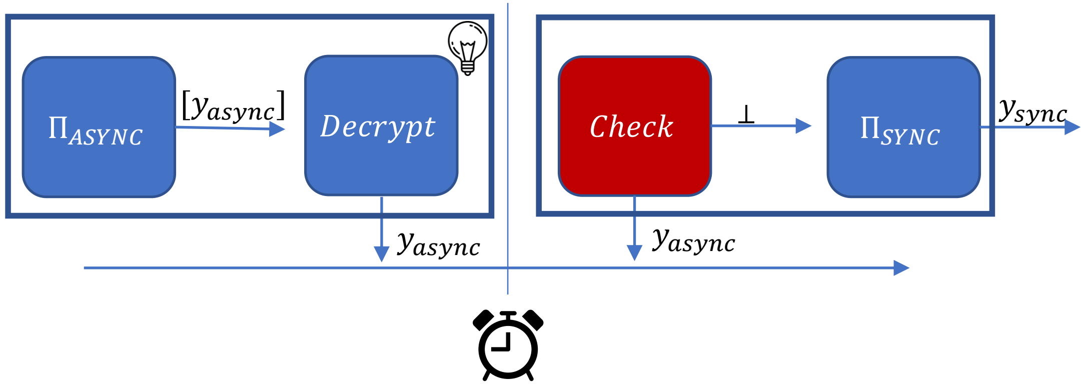
Compiler

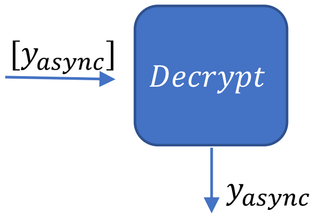




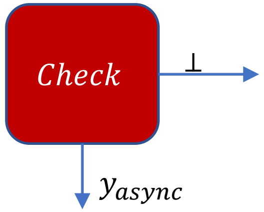
1. Sign $[y_{async}]$ and send it to each P_j
2. Once collected list L of $\frac{3}{4}n$ signatures on c, send decryption share $d_i = Dec_{dki}(c)$
3. Once received $\frac{3}{4}n$ shares, reconstruct the output y_{async}

Compiler





1. Sign $[y_{async}]$ and send it to each P_j
2. Once collected list L of $\frac{3}{4}n$ signatures on c, send decryption share $d_i = Dec_{dki}(c)$
3. Once received $\frac{3}{4}n$ shares, reconstruct the output y_{async}

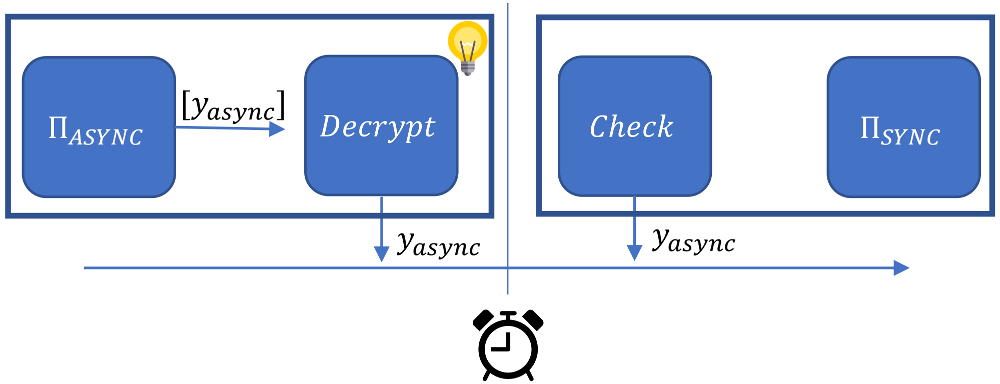


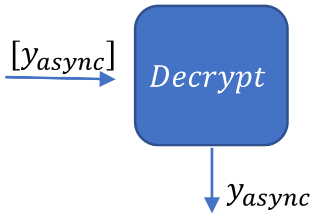
⏰ : Synchronously BC (L,c), if received from Step 2.

After BC: If any (L,c) received, send d_i and reconstruct y_{async}

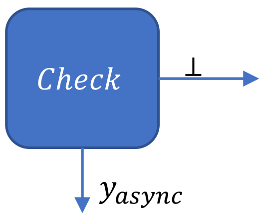
Otherwise, output ⊥

Compiler





1. Sign $[y_{async}]$ and send it to each P_j
2. Once collected list L of $\frac{3}{4}n$ signatures on c, send decryption share $d_i = Dec_{dki}(c)$
3. Once received $\frac{3}{4}n$ shares, reconstruct the output y_{async}

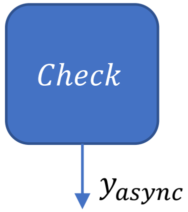


: Synchronously BC (L,c), if received from Step 2.

After BC: If any (L,c) received, send d_i and reconstruct y_{async}

Otherwise, output ⊥

⟹ ∃honest P_i collected list L ⟹



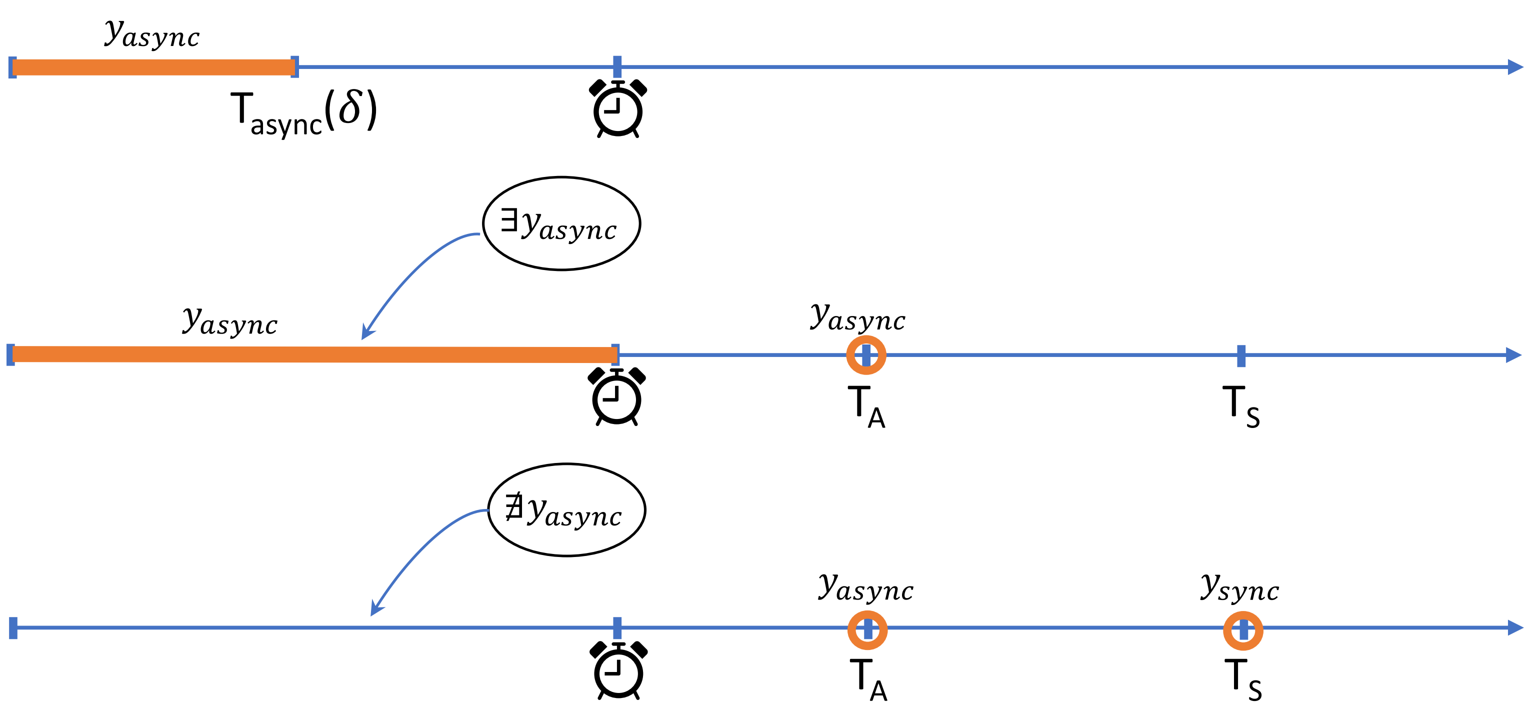
Guarantees

Correctness and Privacy
$^{n}/_{2}$

Output guarantees depend on ⏰

Fast network & $t \le \frac{n}{4}$

Slow network or $t > \frac{n}{4}$

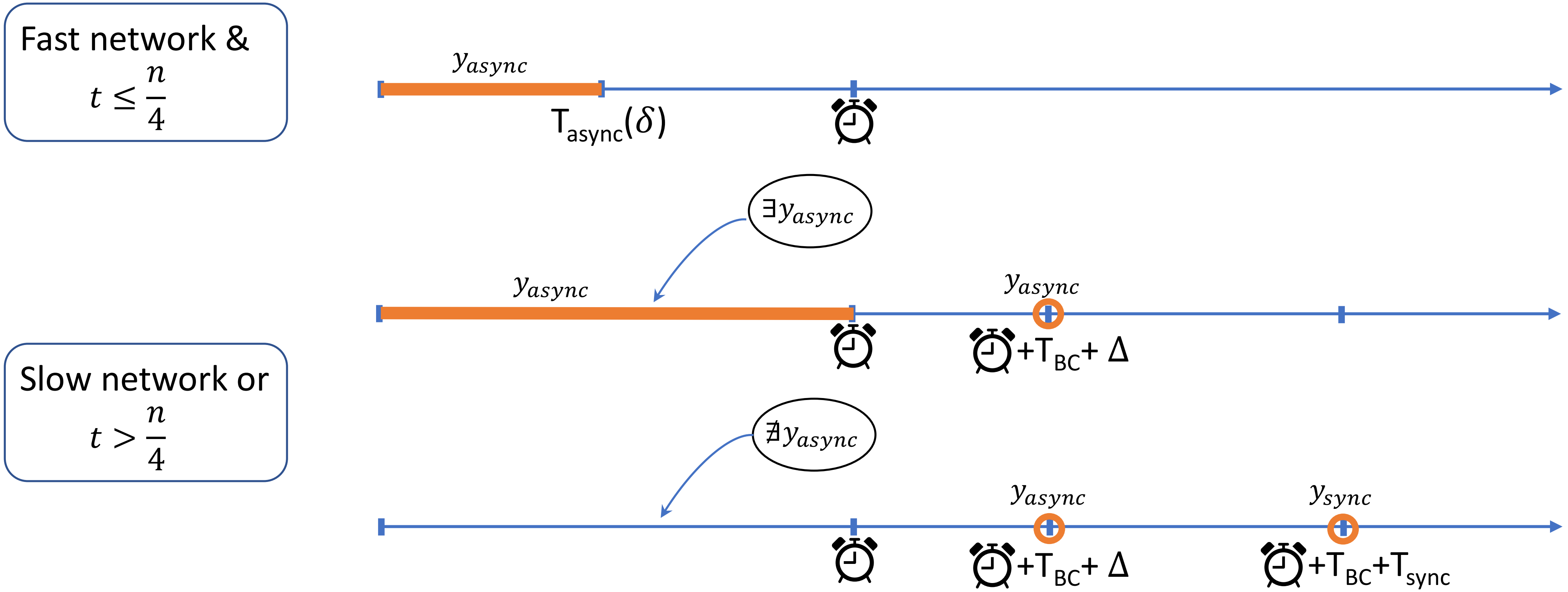


Guarantees

Correctness and Privacy

$^{n}/_{2}$

Output guarantees depend on ⏰



References and Credits

ETH Zurich, Department of Computer Science
lichen@inf.ethz.ch

References:

[LLMMT19]: C. Liu-Zhang, J. Loss, U. Maurer, T. Moran, D. Tschudi. Robust MPC: Asynchronous Responsiveness yet Synchronous Security. Full version: https://eprint.iacr.org/2019/159

[KMTZ13]: Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498. Springer, Heidelberg, March 2013.

[PasShi17]: Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 91. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[LosMor17]: Julian Loss and Tal Moran. Combining asynchronous and synchronous byzantine agreement: The best of both worlds. Cryptology ePrint Archive, Report 2018/235, 2018. https://eprint.iacr.org/2018/235.

[GPS19]: Yue Guo, Rafael Pass, and Elaine Shi. Synchronous, with a chance of partition tolerance. CRYPTO 2019.

Credits:

Icons: https://www.flaticon.com/