Time-Lock Puzzles with Efficient Batch Solving

Jesko Dujmovic
CISPA and Saarland University

Rachit Garg
UT Austin

Giulio Malavolta
Bocconi University and MPI-SP

Date: May 29, 2024

Time-Lock Puzzles [May93, RSW96]

Time-Lock Puzzles [May93, RSW96]

Time-Lock Puzzles [May93, RSW96]

- Fast puzzle generation - Time to generate 

is much shorter than time T.

Time-Lock Puzzles [May93, RSW96]

- Fast puzzle generation - Time to generate  is much shorter than time T.

- Puzzle opening takes a long time - The circuit that opens  has depth at least T. Parallelism shouldn't help.

Applications

Applications

Encrypt to the future!

Applications

Encrypt to the future!

Sealed Bid Auctions

Applications

Encrypt to the future!

Sealed Bid Auctions

Non-Malleable Commitments

Applications

Encrypt to the future!

Sealed Bid Auctions

Non-Malleable Commitments

Miner extractable value prevention

Applications

Encrypt to the future!

Sealed Bid Auctions

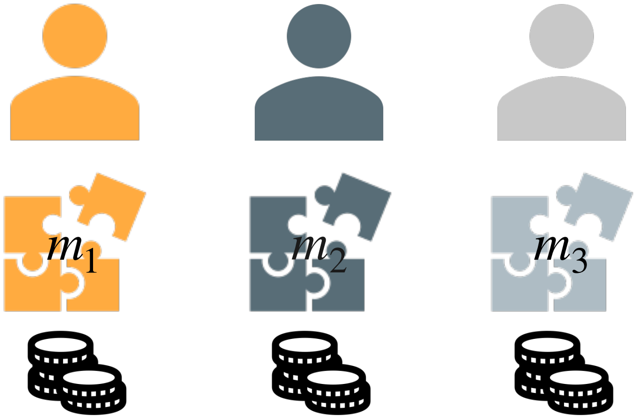
Non-Malleable Commitments

Miner extractable value prevention

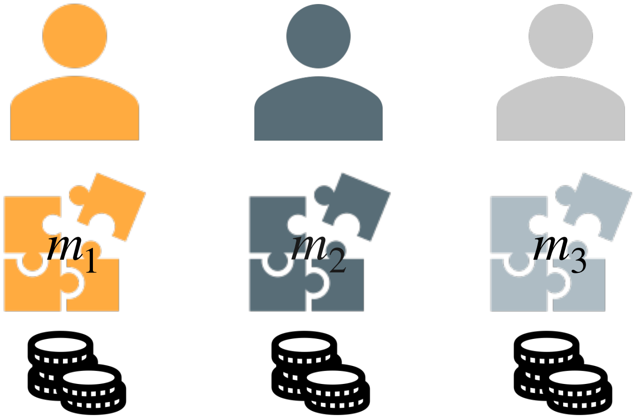
Blockchain front running prevention, fair contract signing, cryptocurrency payments, distributed consensus, more!

Applications - Batch Solving

Applications - Batch Solving

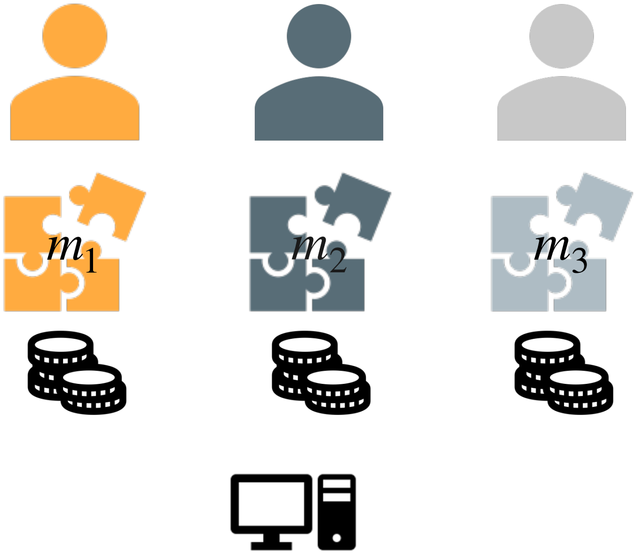


Applications - Batch Solving



Decrypt all transactions!
Solve all puzzles

Applications - Batch Solving

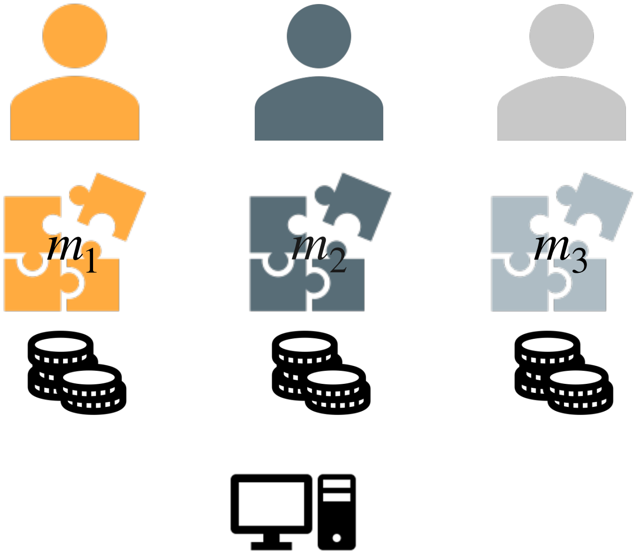


Blockchains, byzantine broadcast

Scalability - Millions of users need solving

Decrypt all transactions!
Solve all puzzles

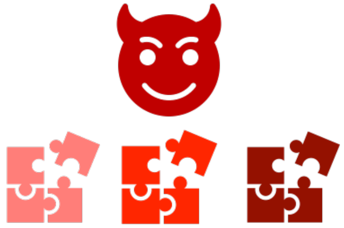
Applications - Batch Solving



Blockchains, byzantine broadcast

Scalability - Millions of users need solving

Decrypt all transactions!
Solve all puzzles



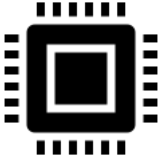
Denial of service attacks

Batching Complexity

- Fast batch solving - Time to solve  , multiple puzzles, grows with the time to solve a “single” puzzle.

Batching Complexity

- Fast batch solving - Time to solve  , multiple puzzles, grows with the time to solve a “single” puzzle.

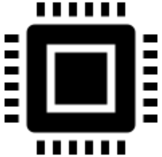


$N \cdot \text{poly}(T)$

Trivial solution

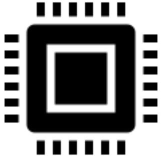
Batching Complexity

- Fast batch solving - Time to solve  , multiple puzzles, grows with the time to solve a “single” puzzle.



$N \cdot \text{poly}(T)$

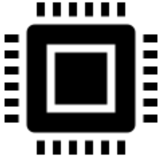
Trivial solution



$o(N) \cdot \text{poly}(T) + \text{poly}(\log T, N)$

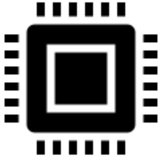
Batching Complexity

- Fast batch solving - Time to solve  , multiple puzzles, grows with the time to solve a “single” puzzle.



$N \cdot \text{poly}(T)$

Trivial solution

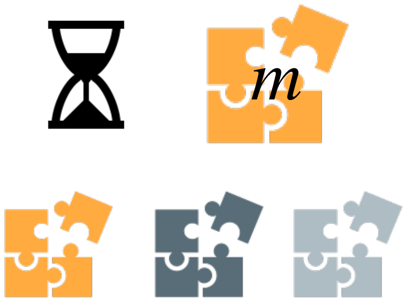


$o(N) \cdot \text{poly}(T) + \text{poly}(\log T, N)$

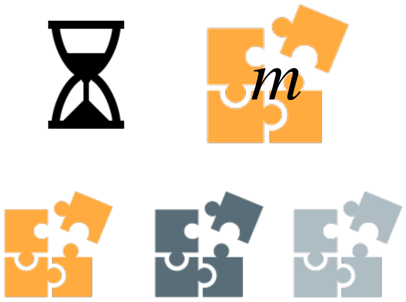
$\boxed{\text{poly}(T)} + \text{poly}(\log T, N)$

This work

Our Result



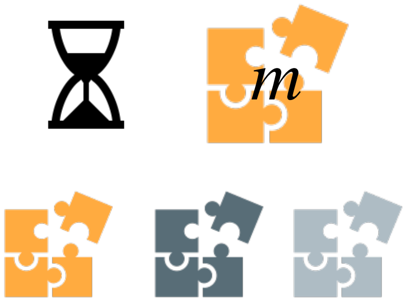
Our Result



- Generic template for constructing batchable TLPs.

  Only prior solution was based on iO [SLM+23].

Our Result

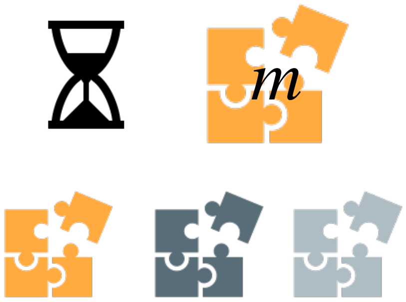


- Generic template for constructing batchable TLPs.

  Only prior solution was based on iO [SLM+23].

- We give two concrete constructions and an implementation.

Our Result

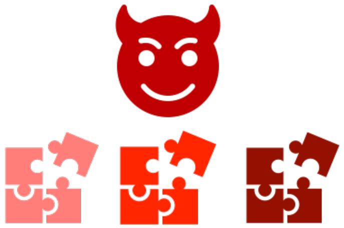


- Generic template for constructing batchable TLPs.

  Only prior solution was based on iO [SLM+23].

- We give two concrete constructions and an implementation.

- Introduce the notion of rogue batch solving.



Roadmap

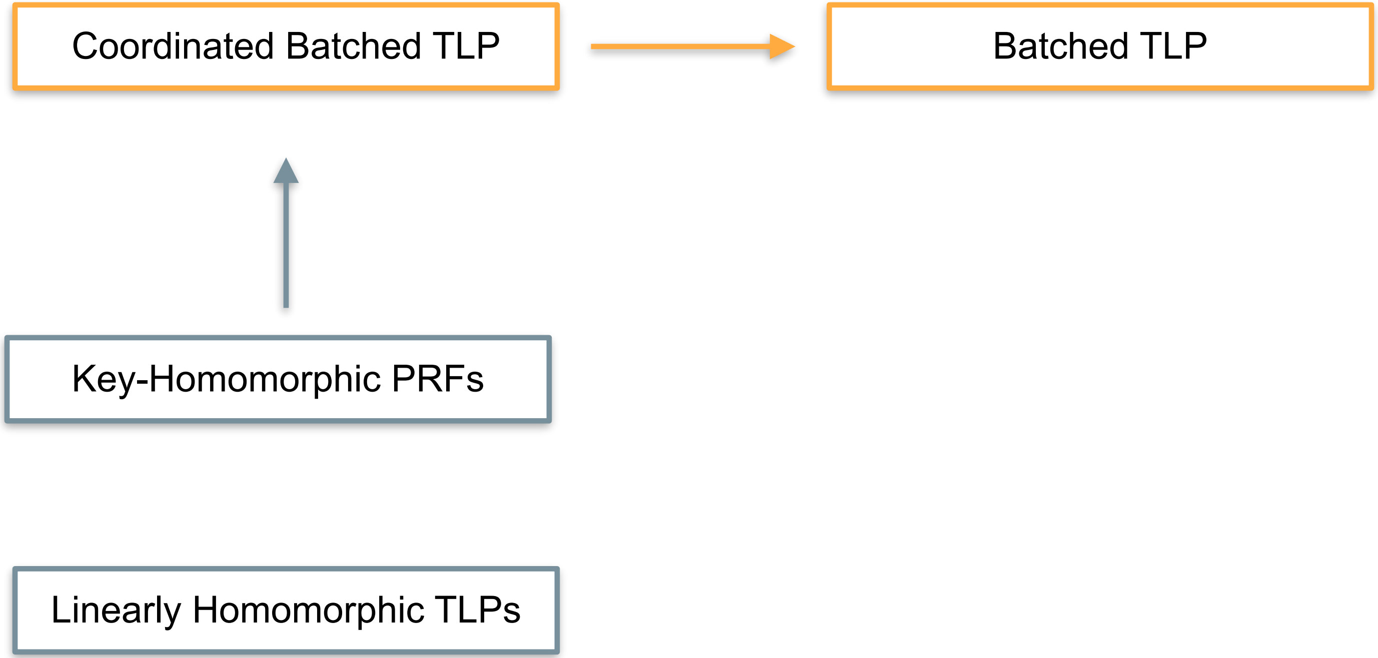
Roadmap

Batched TLP

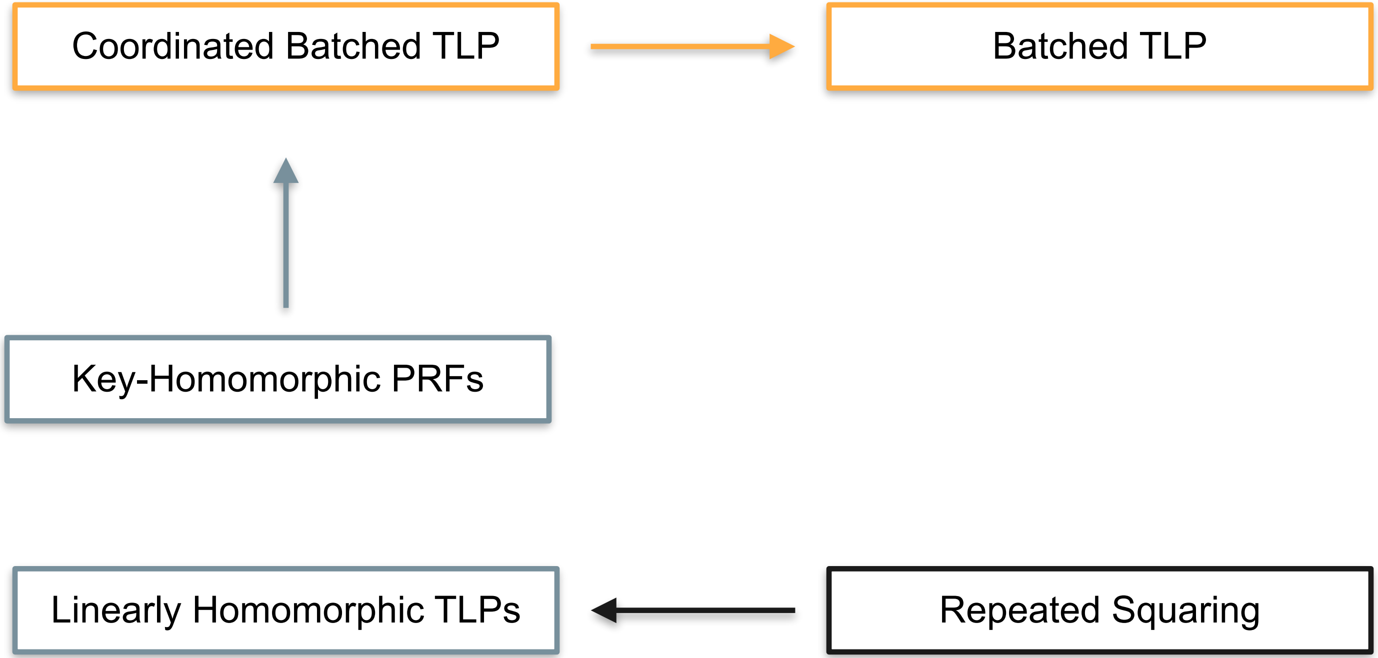
Roadmap

Coordinated Batched TLP → Batched TLP

Roadmap

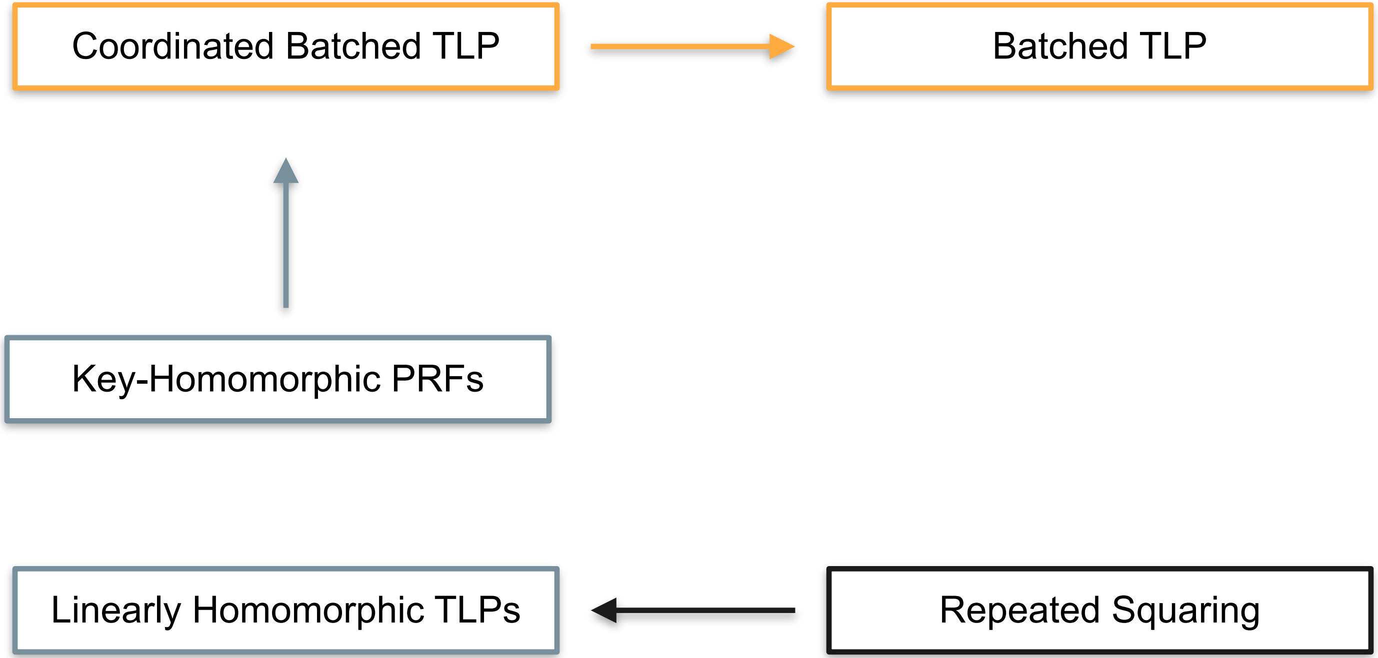

Coordinated Batched TLP
Batched TLP
Key-Homomorphic PRFs
Linearly Homomorphic TLPs

Roadmap



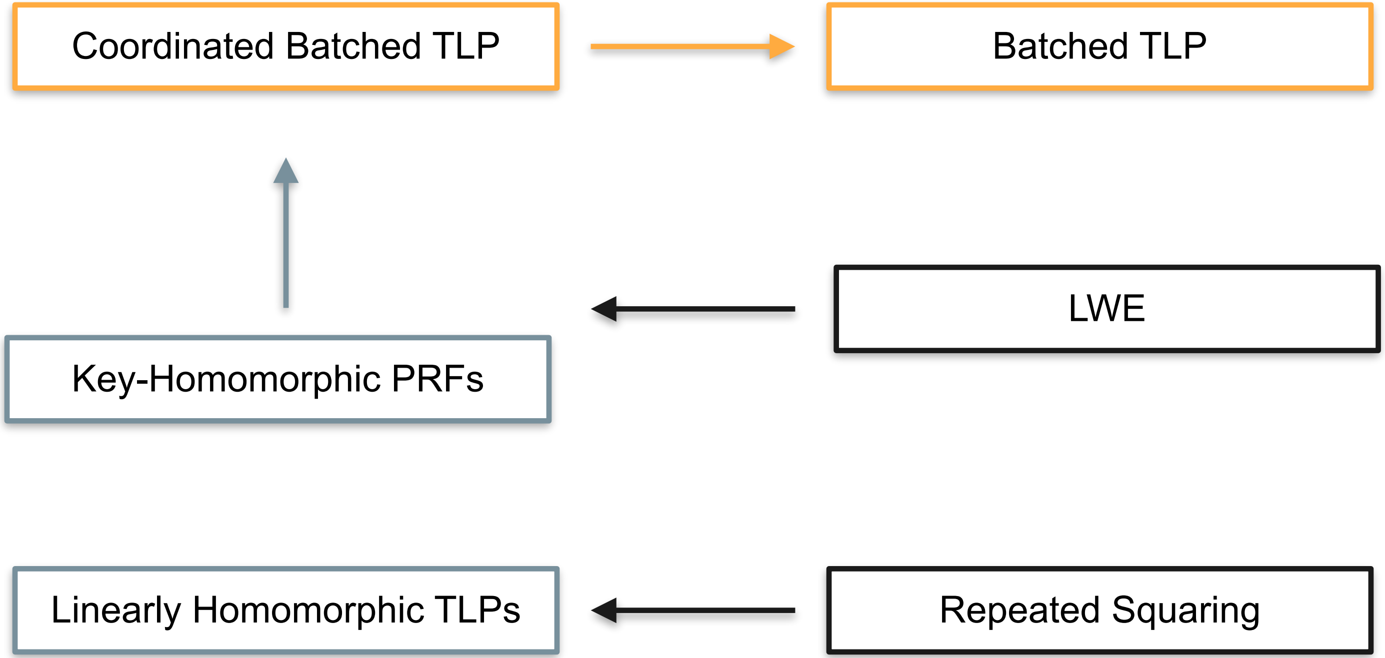
[MT19, TCLM21]

Roadmap



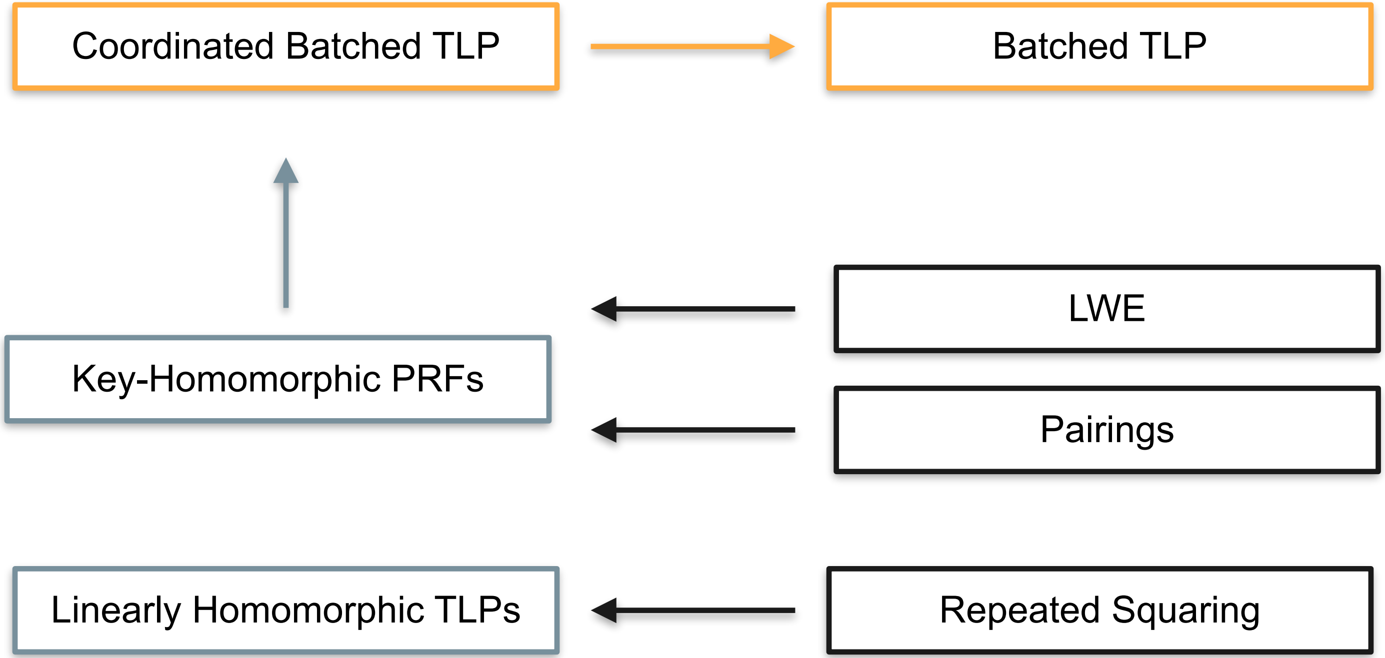
[MT19, TCLM21] $x, x^2, x^4 = (x^2)^2, x^8, \ldots, x^{2^T}$

Roadmap



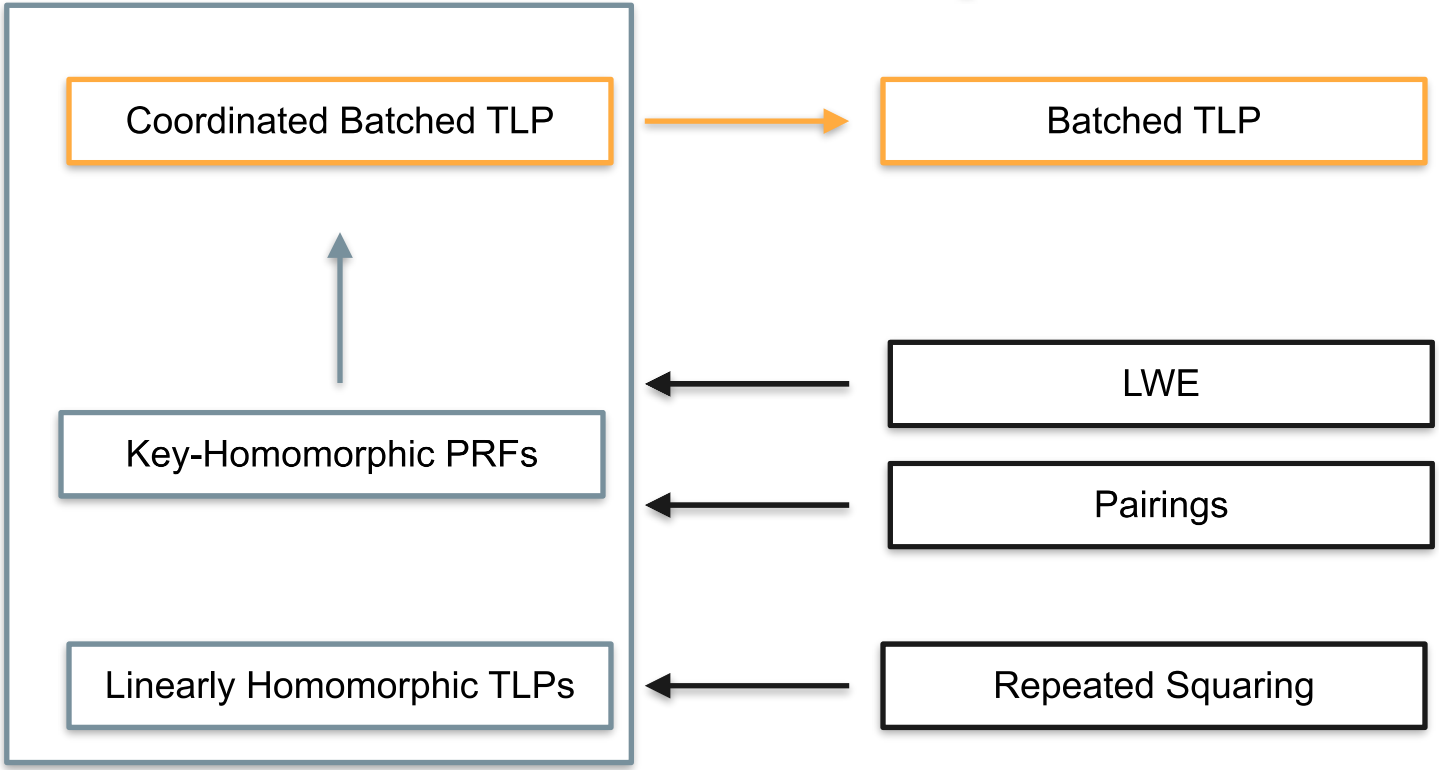
[MT19, TCLM21] $x, x^2, x^4 = (x^2)^2, x^8, \ldots, x^{2^T}$

Roadmap



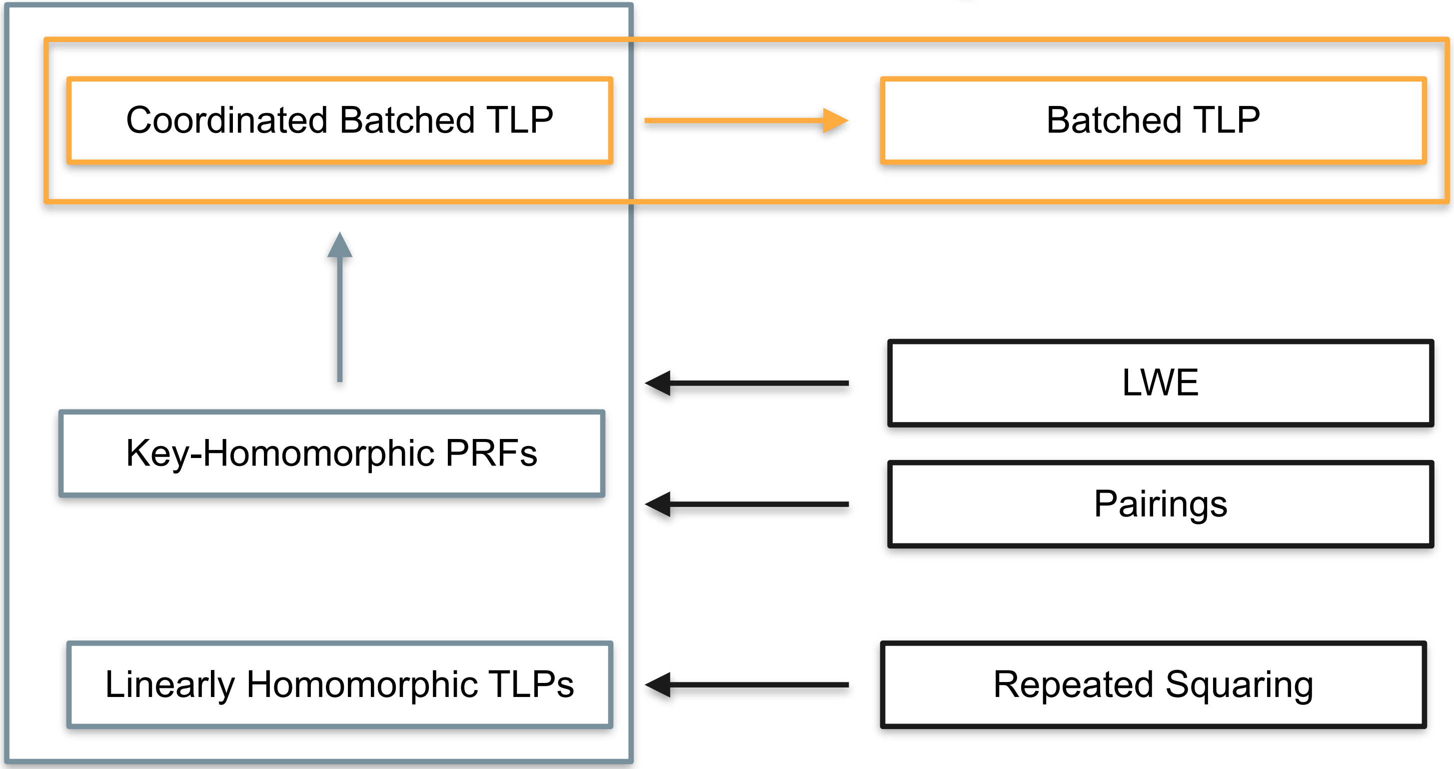
[MT19, TCLM21] $x, x^2, x^4 = (x^2)^2, x^8, \ldots, x^{2^T}$

Roadmap



[MT19, TCLM21] $x, x^2, x^4 = (x^2)^2, x^8, \ldots, x^{2^T}$

Roadmap



[MT19, TCLM21] $x, x^2, x^4 = (x^2)^2, x^8, \ldots, x^{2^T}$

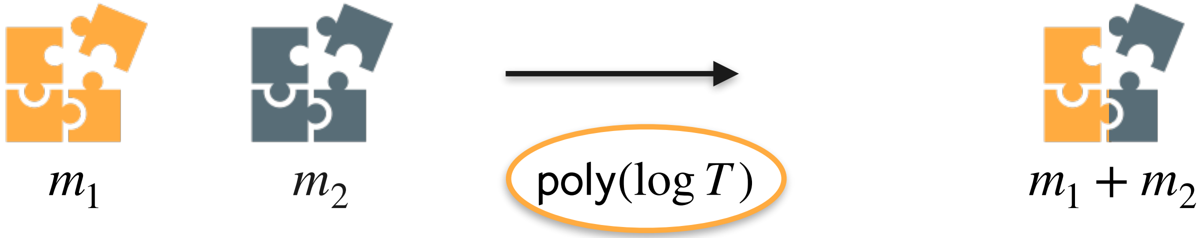
Linearly Homomorphic TLP

Linearly Homomorphic TLP

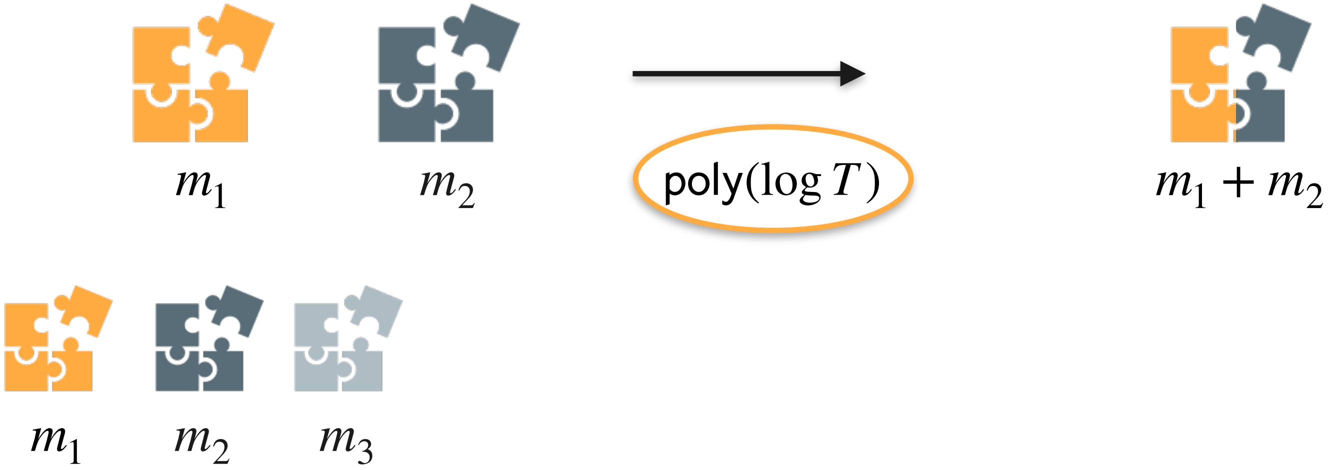
m_1

m_2

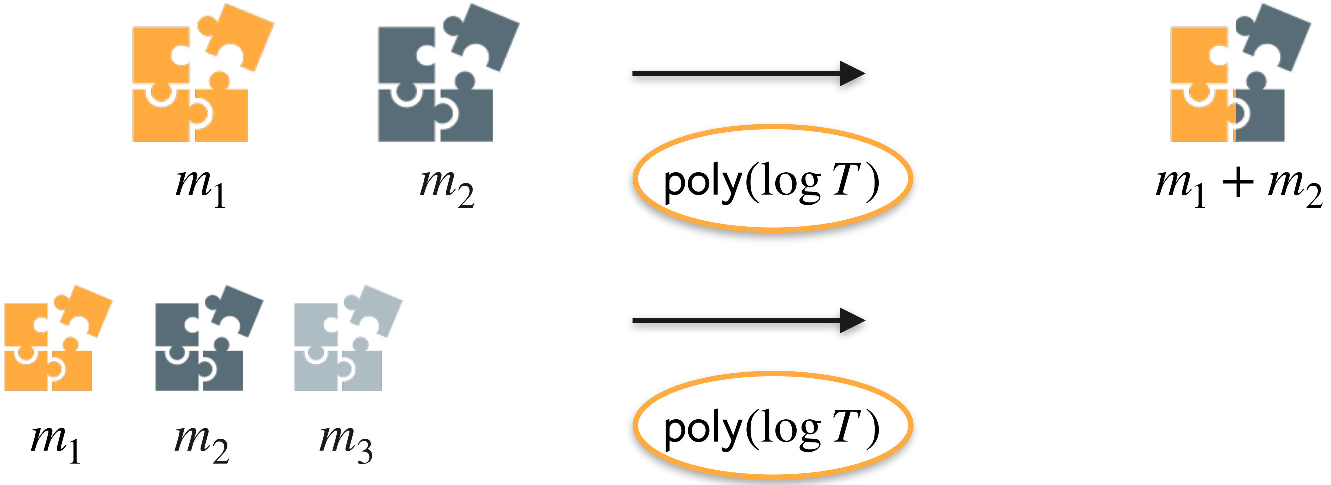
Linearly Homomorphic TLP



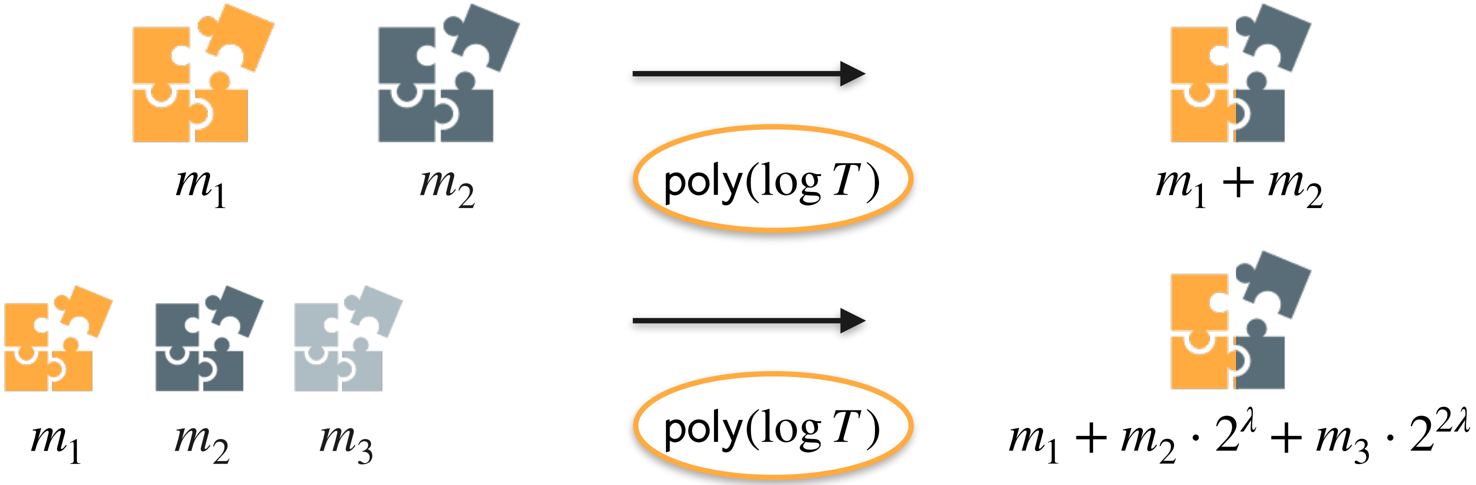
Linearly Homomorphic TLP



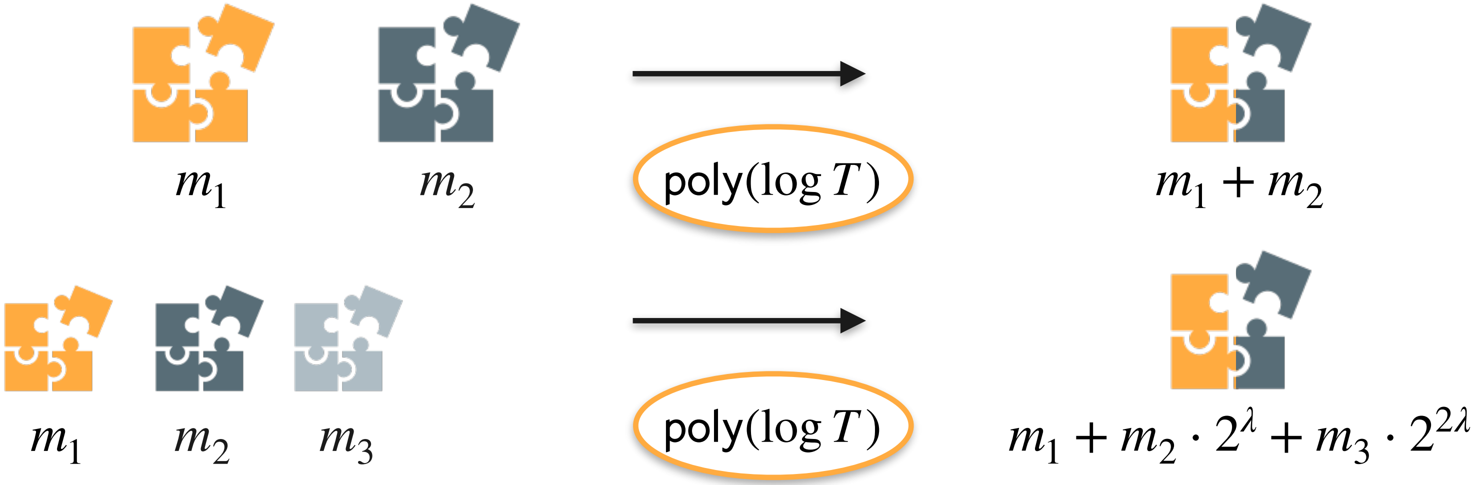
Linearly Homomorphic TLP



Linearly Homomorphic TLP

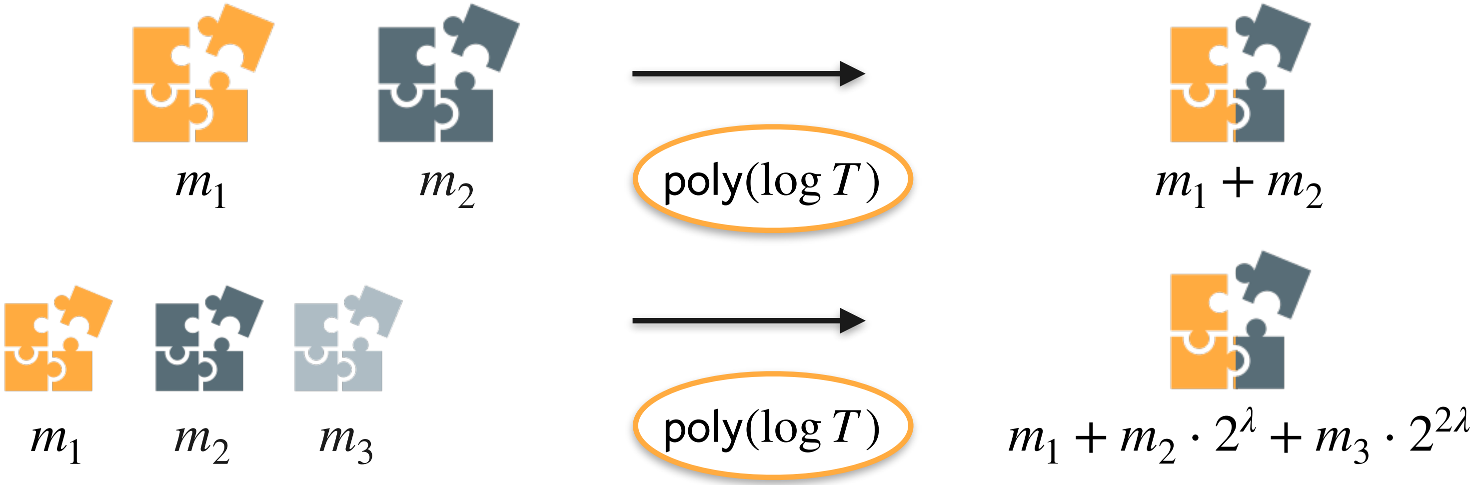


Linearly Homomorphic TLP



Bounded Batching only

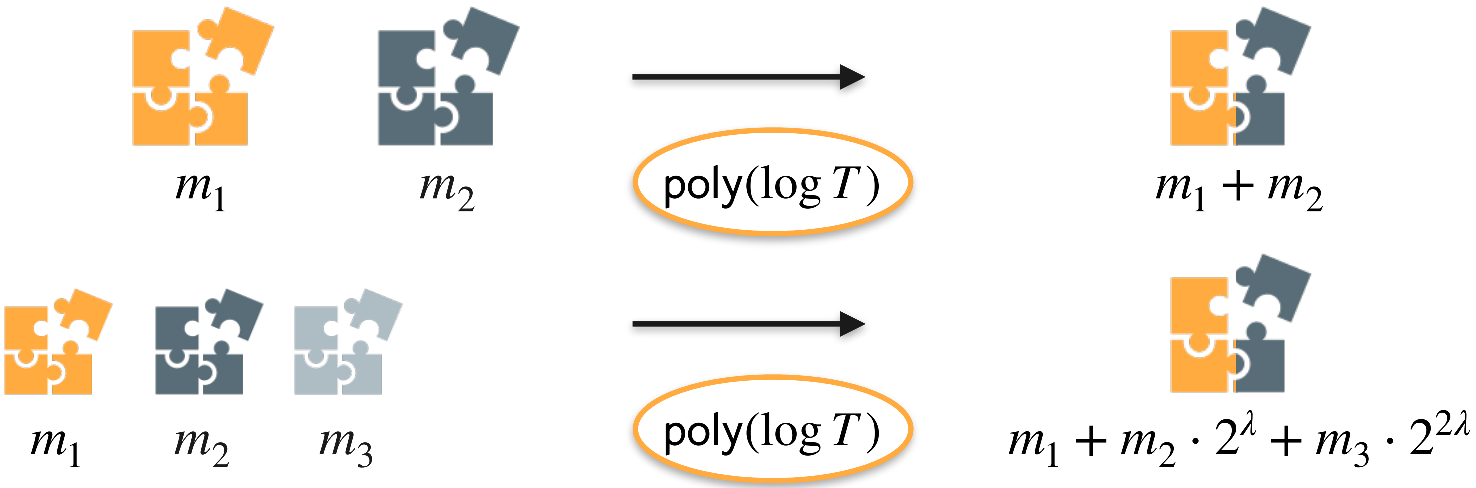
Linearly Homomorphic TLP



Bounded Batching only

Homomorphism over $\{0,1\}^{3\lambda}$

Linearly Homomorphic TLP



Bounded Batching only

Homomorphism over $\{0,1\}^{3\lambda}$

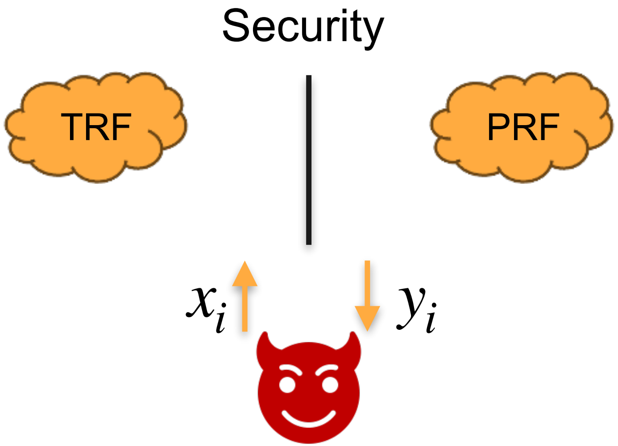
Key Homomorphic PRFs

Key Homomorphic PRFs

- PRF Setup - $\mathsf{Setup}(1^\lambda) \to k$.
- PRF Evaluation - $\mathsf{Eval}(k, x) \to y$.

Key Homomorphic PRFs

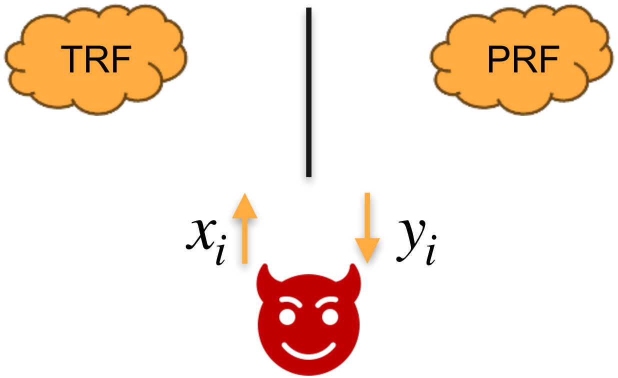
- PRF Setup - $\mathsf{Setup}(1^\lambda) \to k$.
- PRF Evaluation - $\mathsf{Eval}(k, x) \to y$.



Key Homomorphic PRFs

- PRF Setup - Setup$(1^\lambda) \rightarrow k$.
- PRF Evaluation - Eval$(k, x) \rightarrow y$.

Security

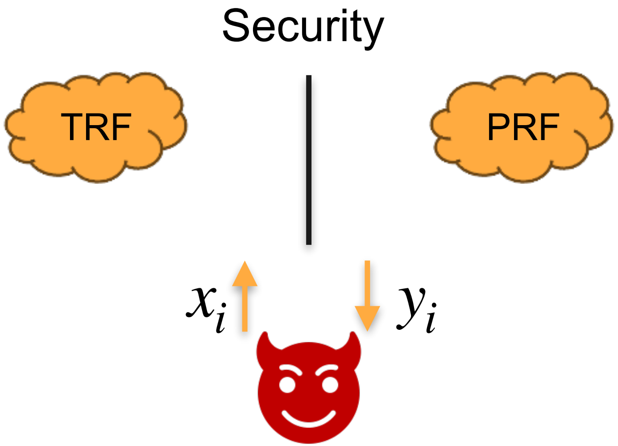


Eval(k_1, x)

Eval(k_2, x)

Key Homomorphic PRFs

- PRF Setup - $\mathsf{Setup}(1^\lambda) \to k$.
- PRF Evaluation - $\mathsf{Eval}(k, x) \to y$.



$\mathsf{Eval}(k_1, x)$

$\mathsf{Eval}(k_2, x)$

$\longrightarrow$ $\mathsf{Eval}(k_1 + k_2, x)$

Key Homomorphic PPRFs

Key Homomorphic PPRFs

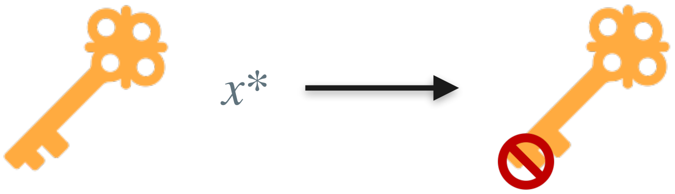
- PRF Setup - $\mathsf{Setup}(1^{\lambda}) \to k$.
- PRF Evaluation - $\mathsf{Eval}(k, x) \to y$.
- PRF key homomorphism.

Key Homomorphic PPRFs

- PRF Setup - $\mathsf{Setup}(1^\lambda) \to k$.
- PRF Evaluation - $\mathsf{Eval}(k, x) \to y$.
- PRF key homomorphism.
- PRF puncturing

Key Homomorphic PPRFs

- PRF Setup - $\mathsf{Setup}(1^\lambda) \to k$.
- PRF Evaluation - $\mathsf{Eval}(k, x) \to y$.
- PRF key homomorphism.
- PRF puncturing

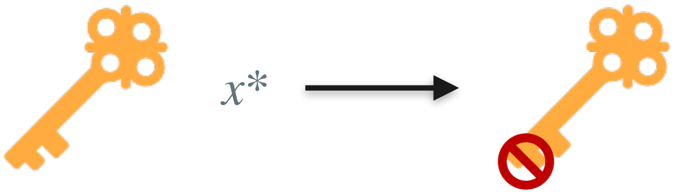


Key Homomorphic PPRFs

- PRF Setup - Setup$(1^\lambda) \to k$.
- PRF Evaluation - Eval$(k, x) \to y$.
- PRF key homomorphism.
- PRF puncturing

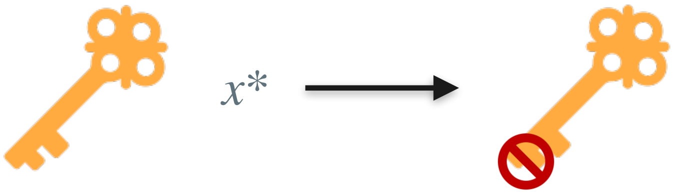
Eval( $, x) =$ Eval( $, x)$

$x \neq x^*$



Key Homomorphic PPRFs

- PRF Setup - $\text{Setup}(1^\lambda) \rightarrow k$.
- PRF Evaluation - $\text{Eval}(k, x) \rightarrow y$.
- PRF key homomorphism.
- PRF puncturing

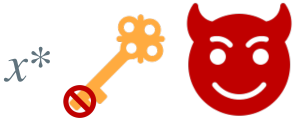


Eval( , x) = Eval( , x)

$x \neq x^*$

Security

Eval( , x^*) | random



Construction

Construction

Party i

m_i

Construction

Party i

m_i

k_i

$\text{PRF\_Eval}(k_i, i) + m_i$

Construction

Party i

m_i

k_i

$\text{PRF\_Eval}(k_i, i) + m_i$

Puncture k_i at i

Construction

Party i

m_i

k_i

$\text{PRF\_Eval}(k_i, i) + m_i$

Puncture k_i at i

m_1 m_2 m_3

Construction

Party i

m_i

k_i

PRF_Eval$(k_i, i) + m_i$

Puncture k_i at i

m_1 m_2 m_3

PRF_Eval$(k_1,1) + m_1$

Construction

Party i

m_i

k_i

PRF_Eval$(k_i, i) + m_i$

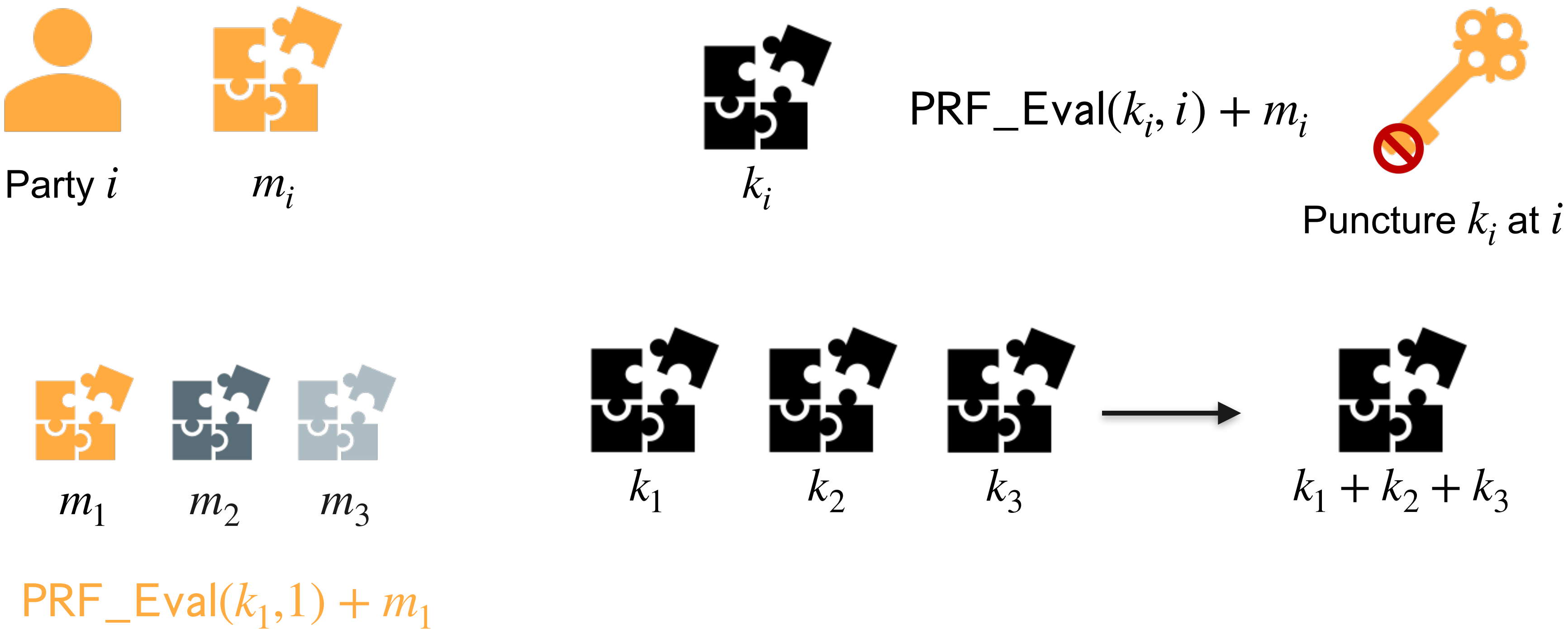
Puncture k_i at i

m_1 m_2 m_3

k_1 k_2 k_3

PRF_Eval$(k_1, 1) + m_1$

Construction



Construction

Party i

m_i

k_i

PRF_Eval$(k_i, i) + m_i$

Puncture k_i at i

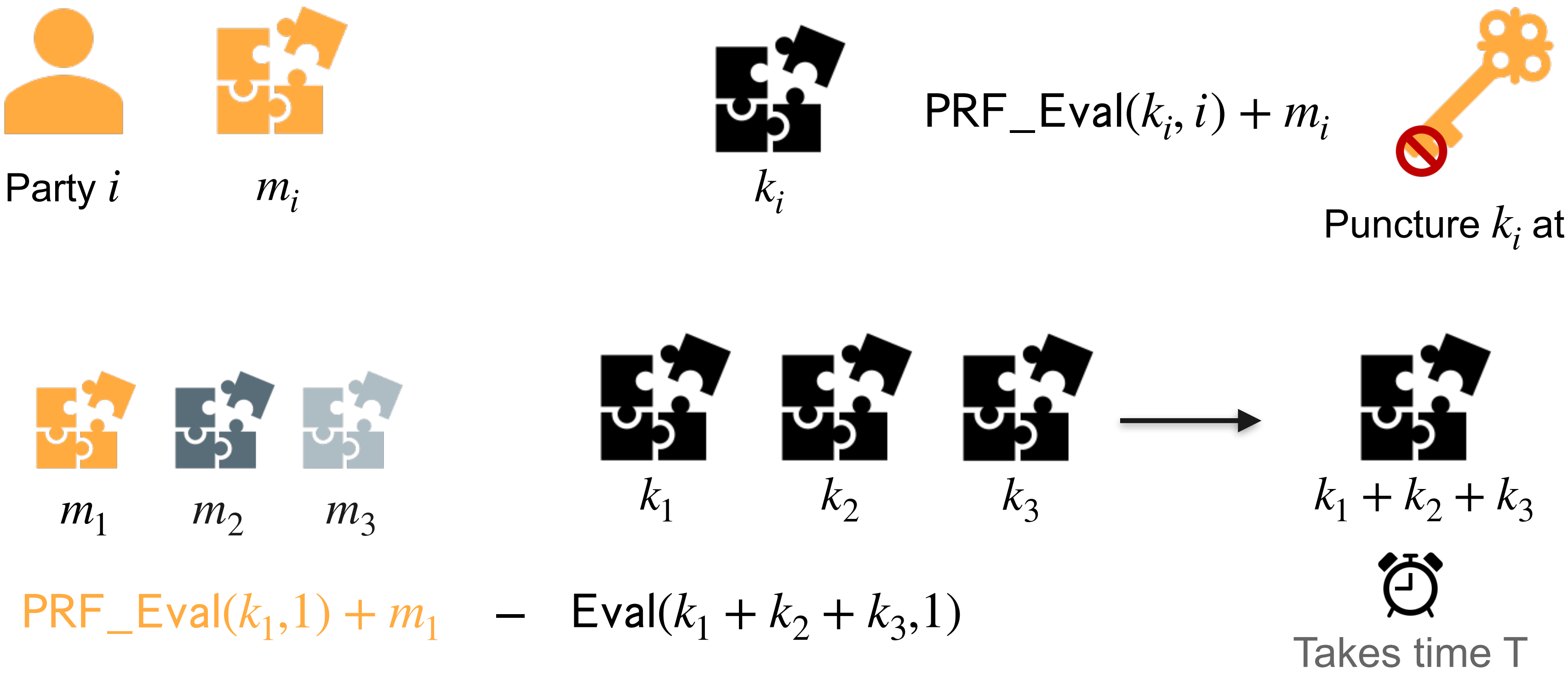
m_1 m_2 m_3

PRF_Eval$(k_1, 1) + m_1$

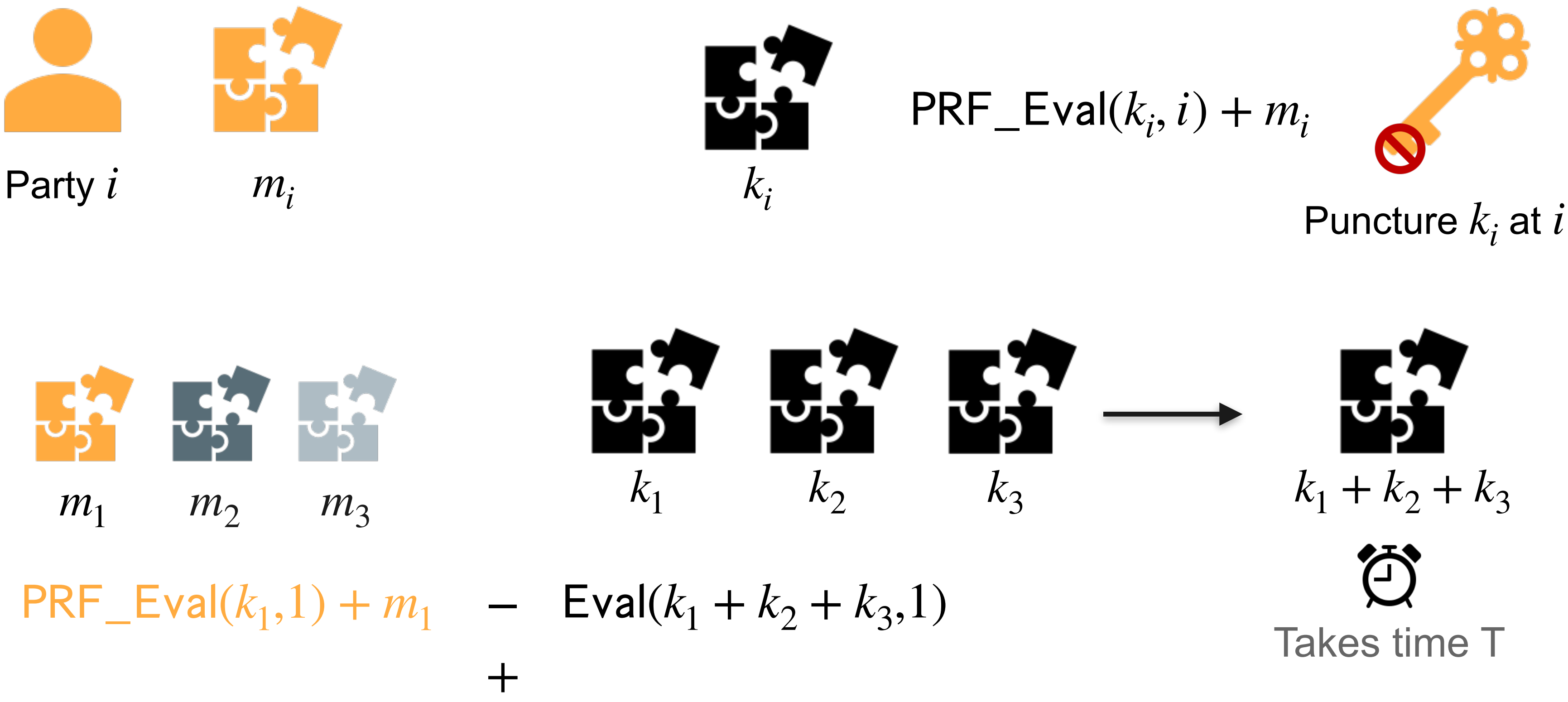
k_1 k_2 k_3 → $k_1 + k_2 + k_3$

Takes time T

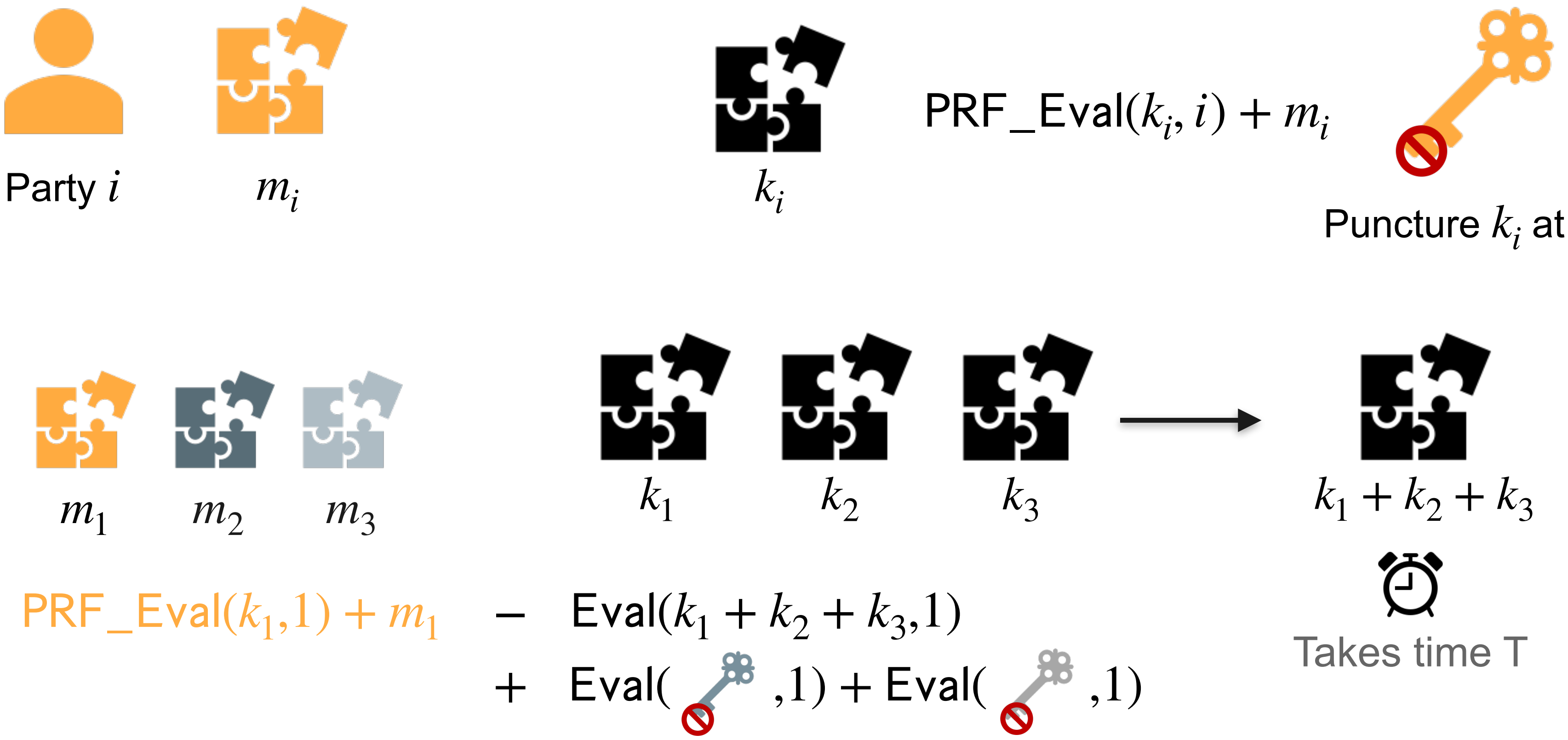
Construction



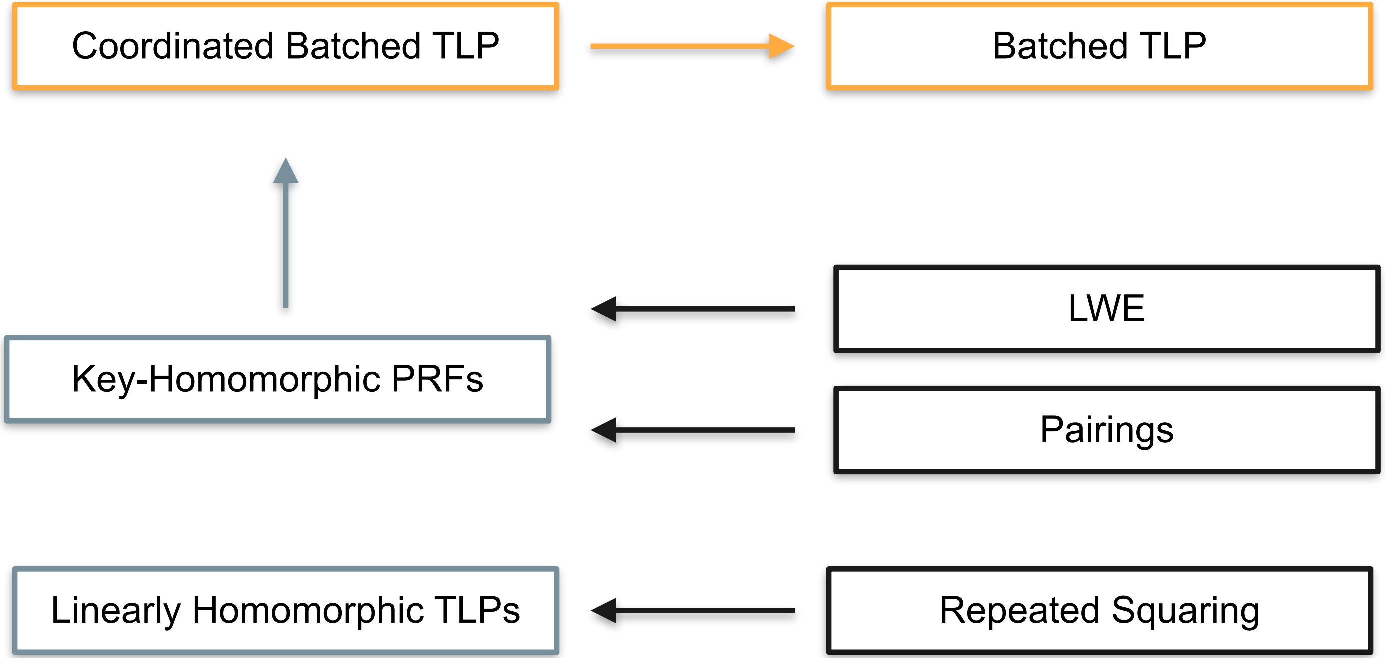
Construction



Construction

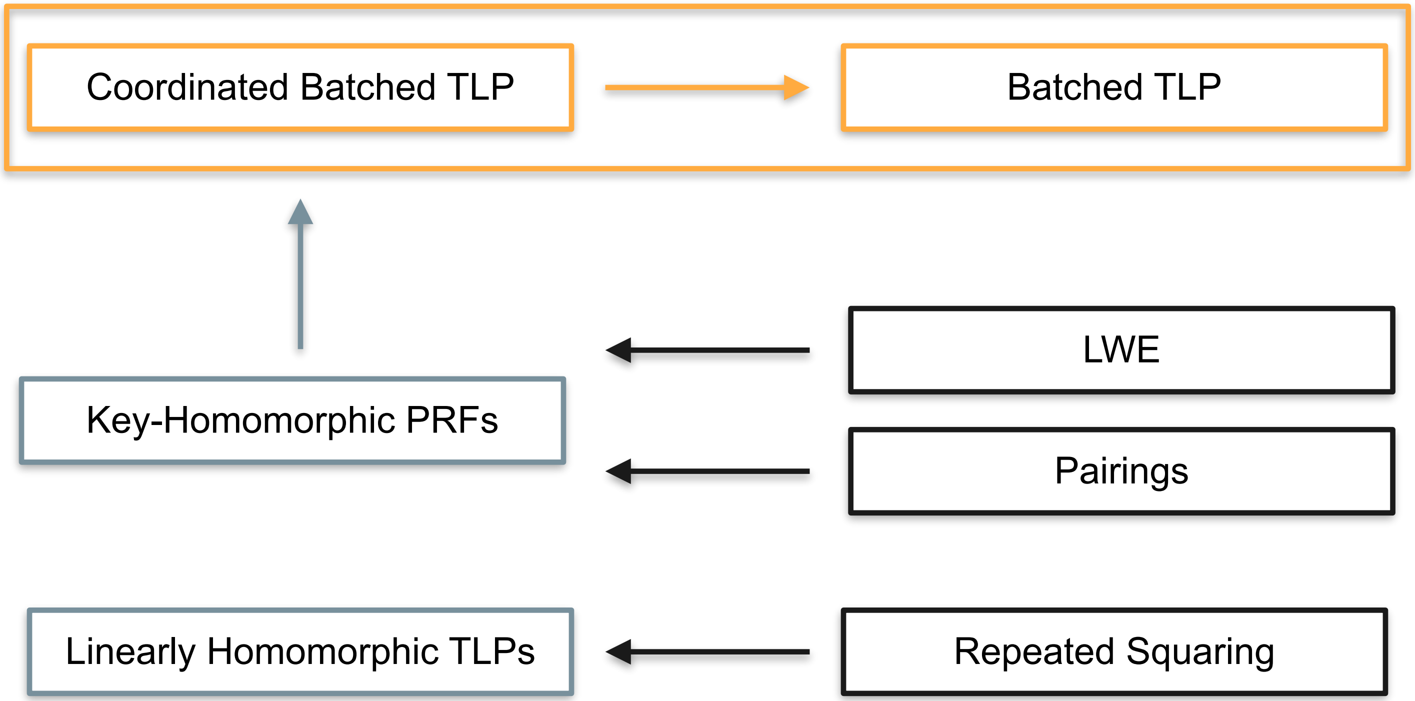


Roadmap



Coordinated Batched TLP
Batched TLP
LWE
Key-Homomorphic PRFs
Pairings
Linearly Homomorphic TLPs
Repeated Squaring

Roadmap


Coordinated Batched TLP
Batched TLP
LWE
Key-Homomorphic PRFs
Pairings
Linearly Homomorphic TLPs
Repeated Squaring

Transformation

Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

Slots

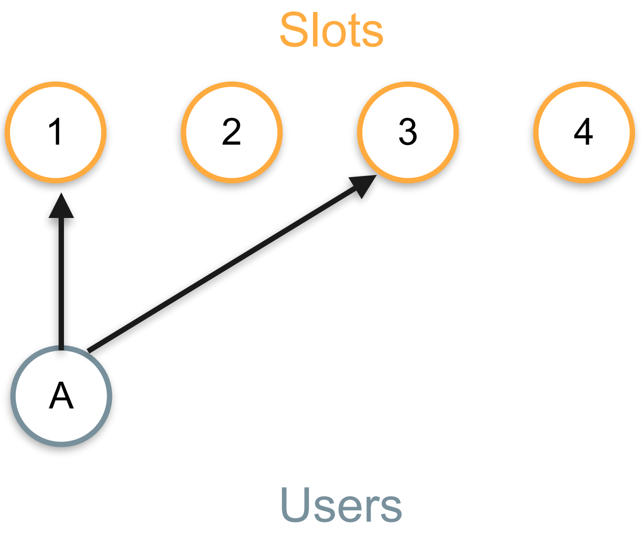
Setup$(1^{\lambda}, T, 1^{4}) \rightarrow$ pp

Users

Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

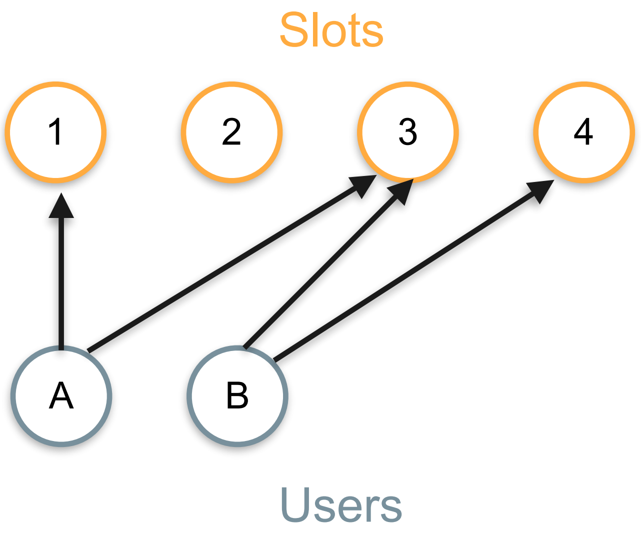
$\text{Setup}(1^\lambda, T, 1^4) \rightarrow \text{pp}$



Transformation

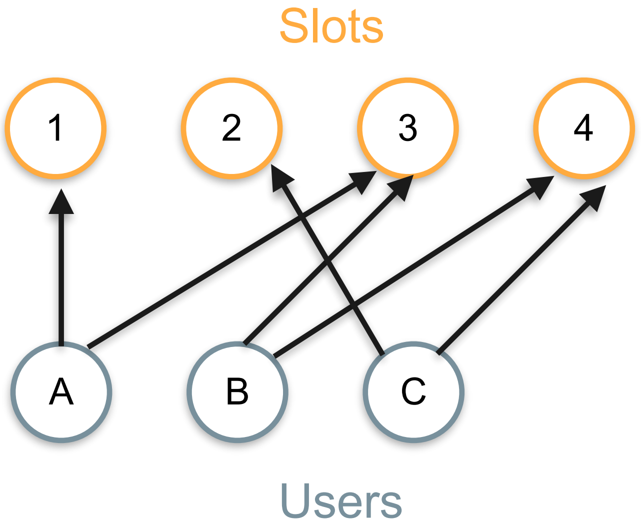
M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

Setup$(1^\lambda, T, 1^4) \rightarrow$ pp



Transformation

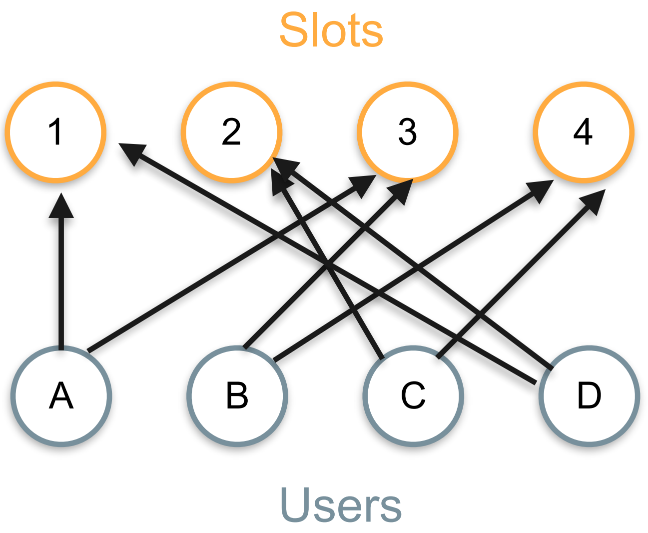
M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree



Setup$(1^{\lambda}, T, 1^{4}) \rightarrow$ pp

Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

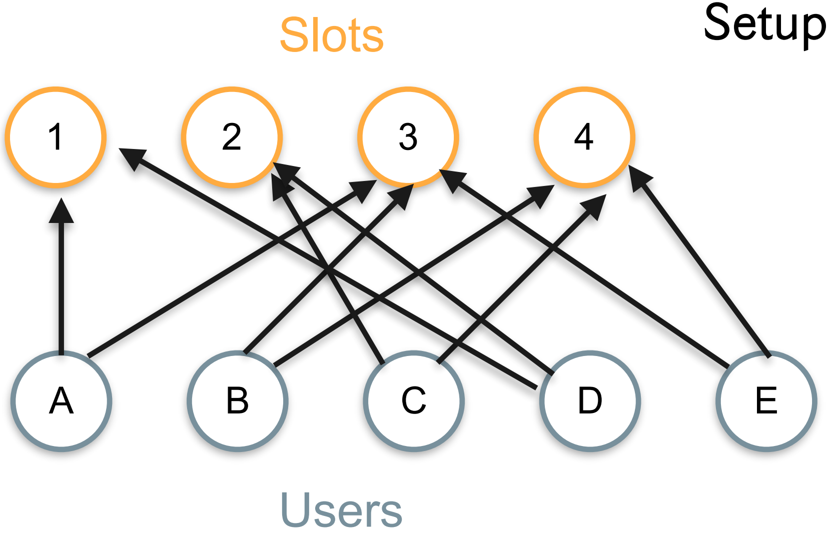


Setup$(1^\lambda, T, 1^4) \rightarrow$ pp

Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

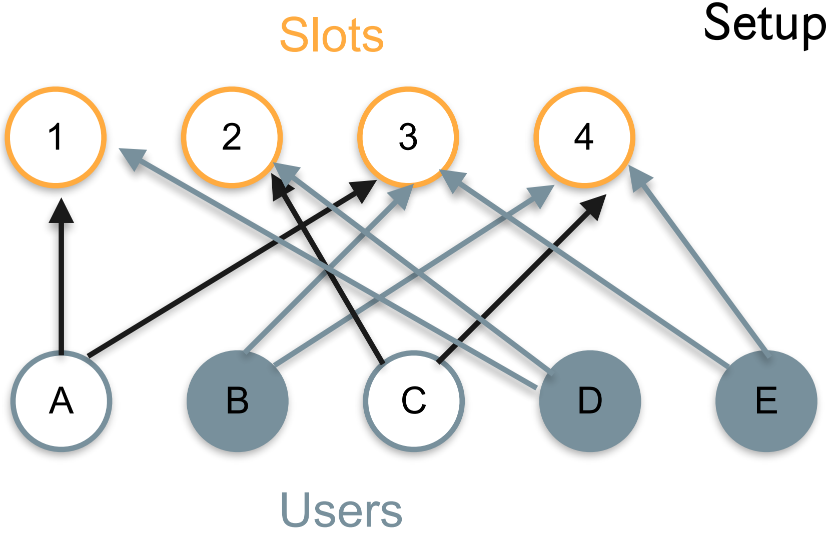
Setup$(1^\lambda, T, 1^4) \rightarrow$ pp



Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

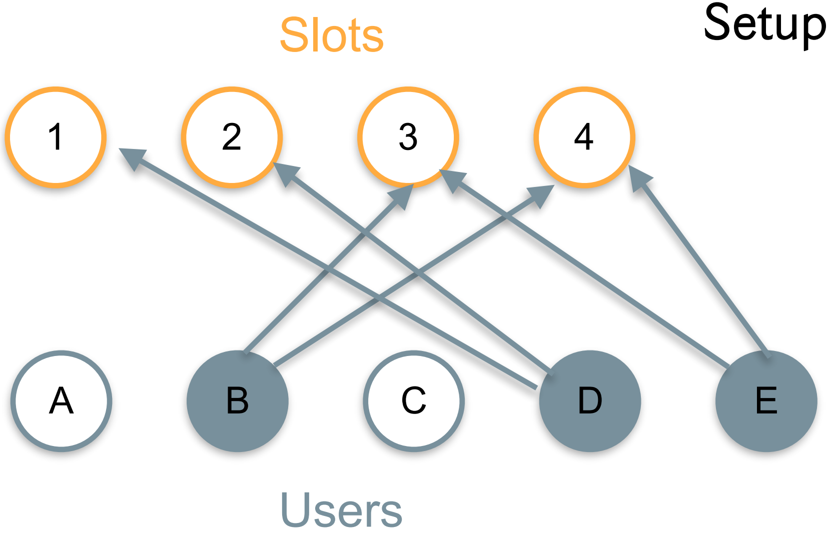
Setup$(1^\lambda, T, 1^4) \rightarrow$ pp



Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

Setup$(1^\lambda, T, 1^4) \rightarrow$ pp

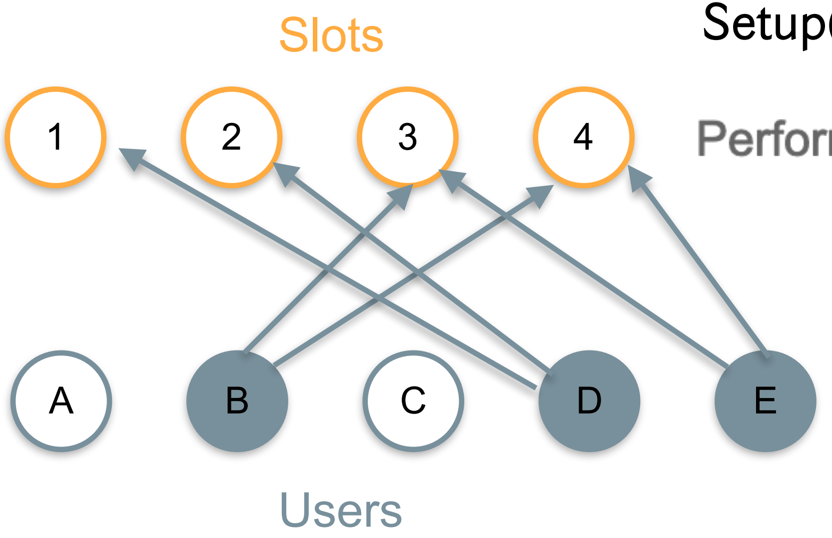


Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

$\text{Setup}(1^\lambda, T, 1^4) \rightarrow \text{pp}$

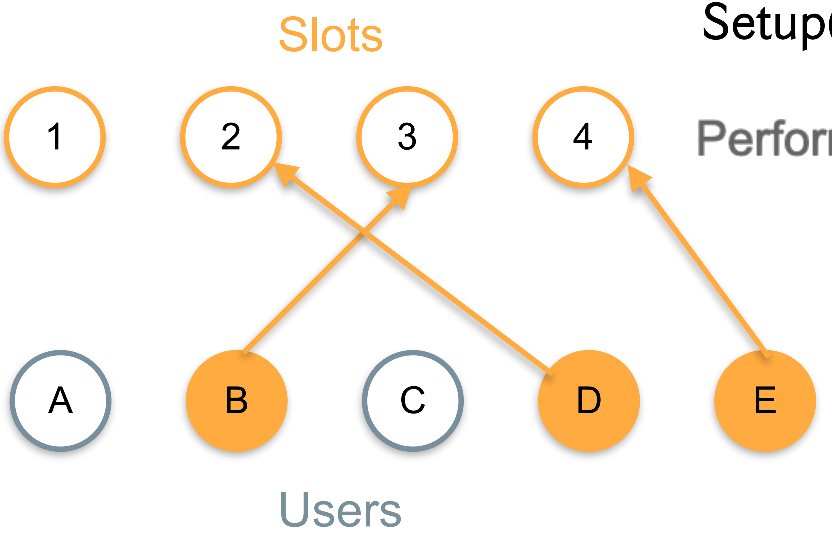
Perform a matching



Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree

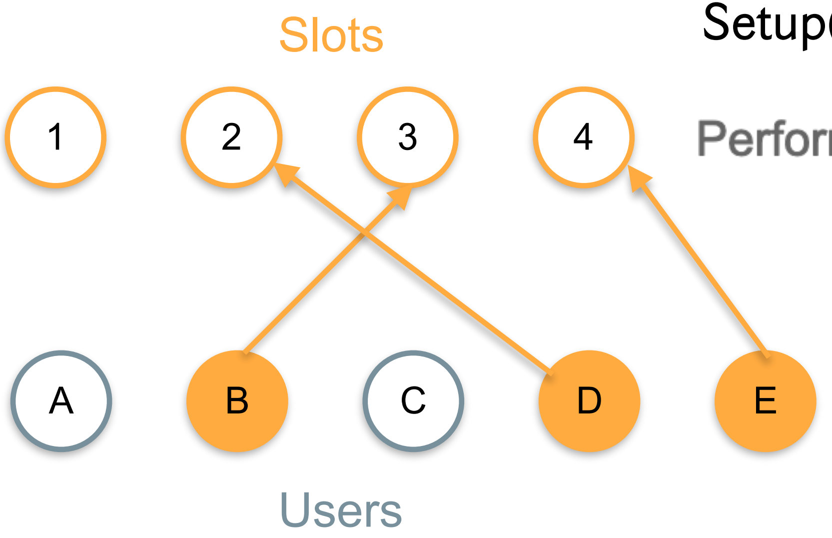
$\text{Setup}(1^\lambda, T, 1^4) \rightarrow \text{pp}$



Perform a matching

Transformation

M - 5 - number of users
N - 3 - Batch to at-most 3 puzzles
D - 2 - degree



Setup$(1^{\lambda}, T, 1^4) \rightarrow$ pp

Perform a matching

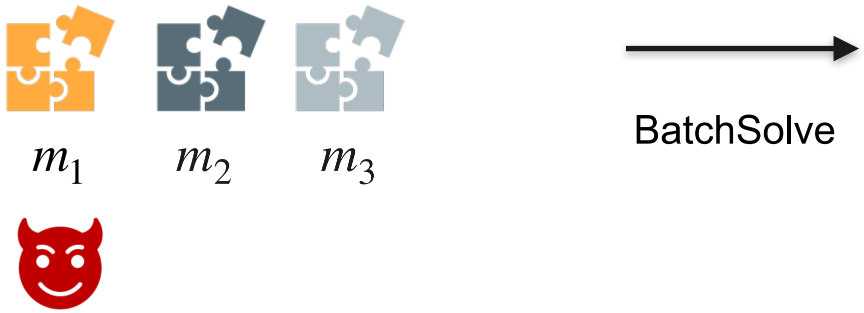
Theorem (informal) - Set slots $\geq 2e \cdot N$, degree $\geq \dfrac{\omega(\log \lambda)}{\log N}$, then we can build an uncoordinated batch TLP.

Rogue puzzles

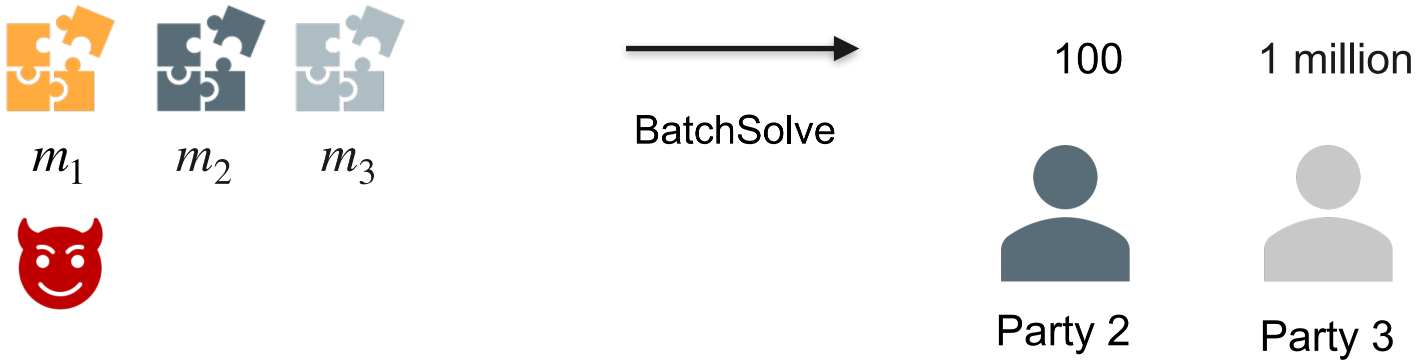
Rogue puzzles

m_1 m_2 m_3

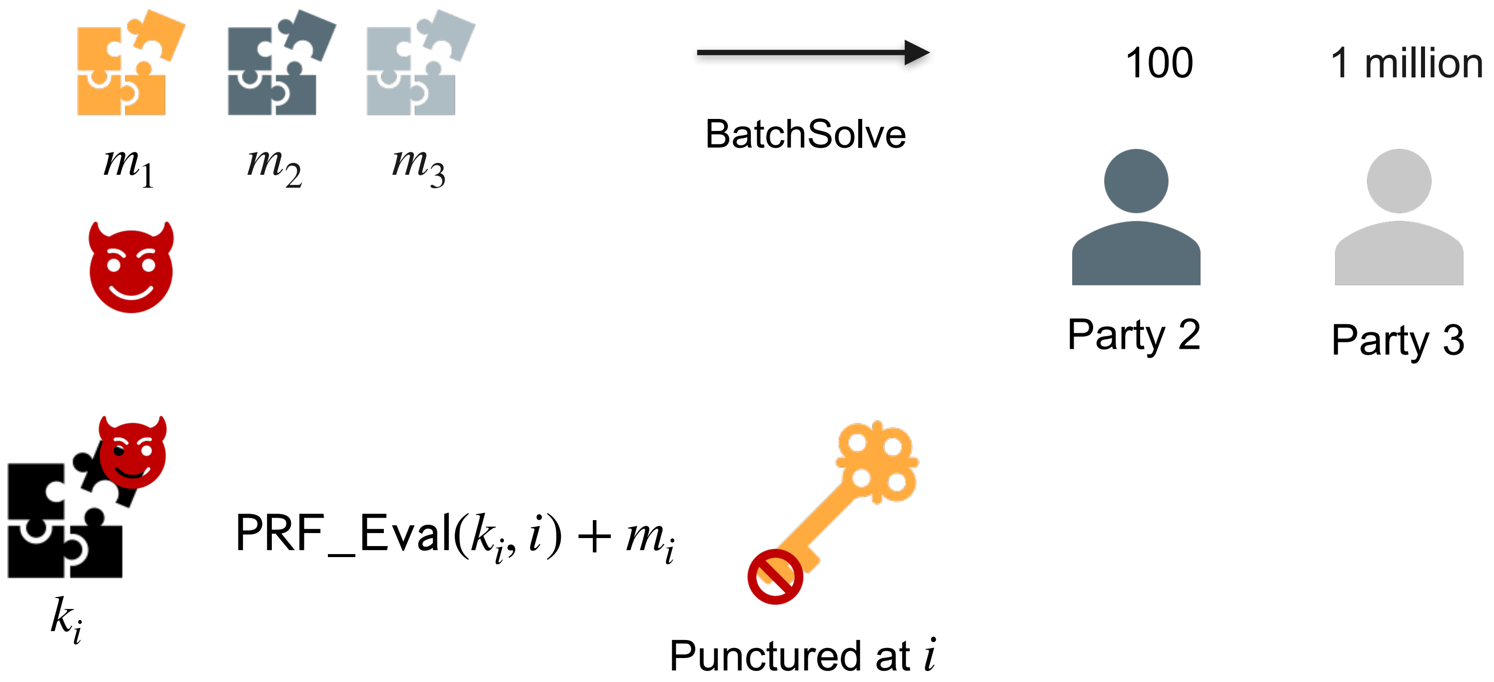
Rogue puzzles



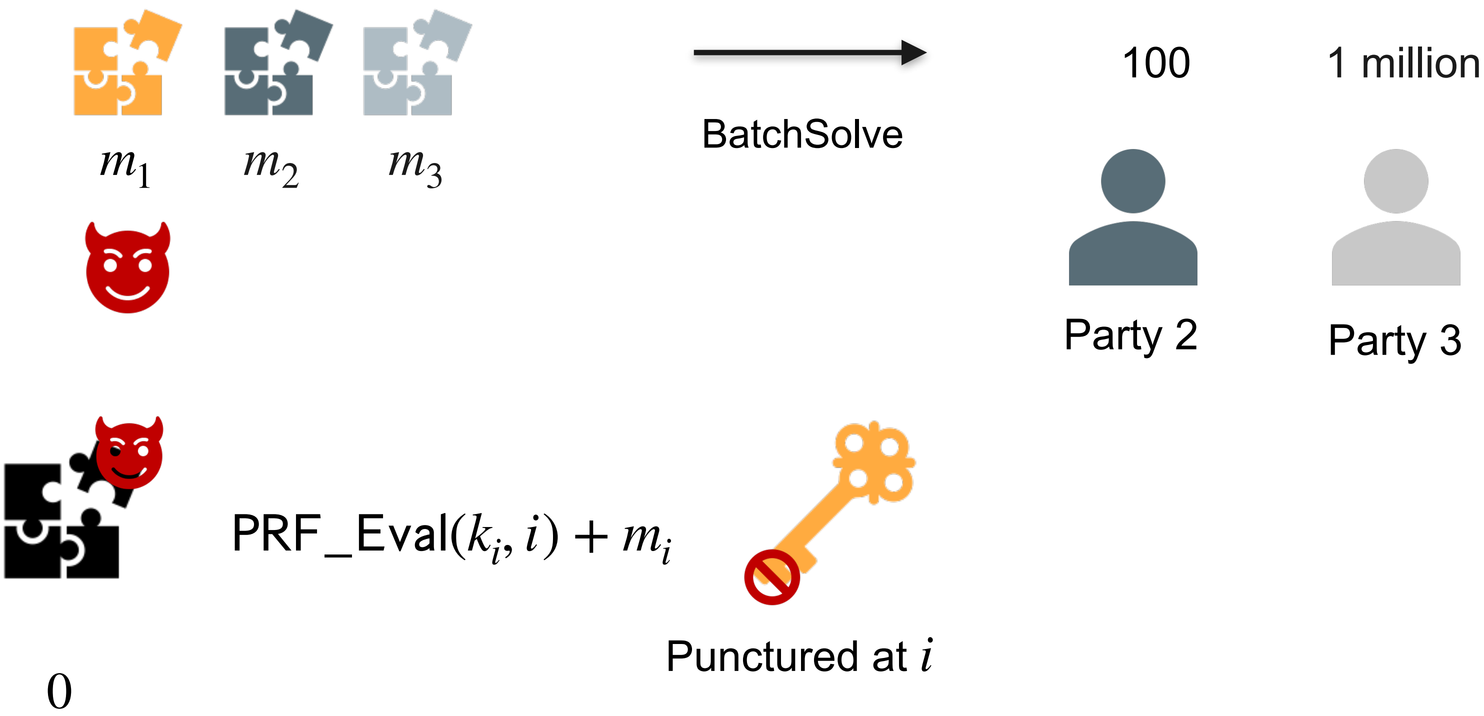
Rogue puzzles



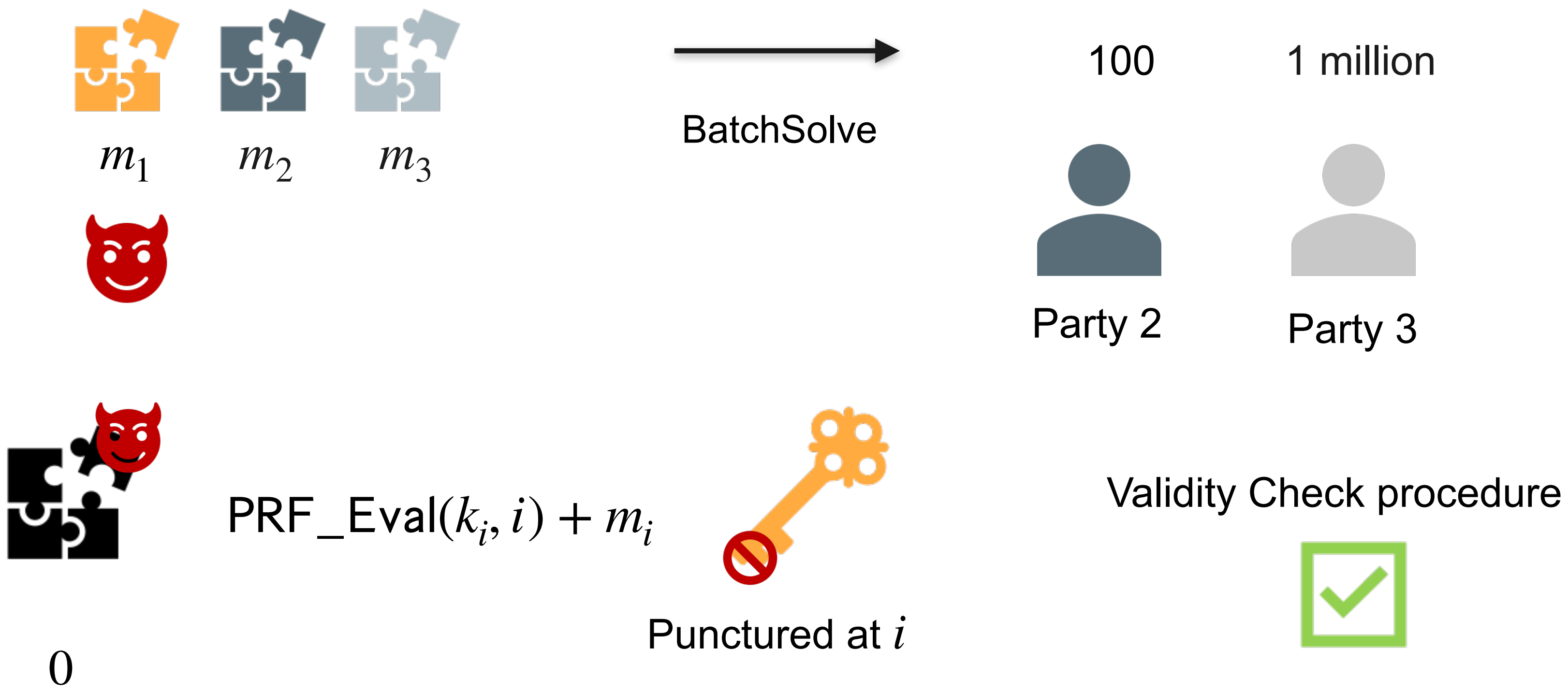
Rogue puzzles



Rogue puzzles



Rogue puzzles



Prototype Evaluation

Prototype Evaluation

- For T = 50 million sequential computations*, and batching 500 puzzles, the batching time trivially would take 15 hours, while our solution takes close to 6 minutes (we did not use any parallelism for our experiments).

*the time to do 50 million sequential computations on the test machine is 5 minutes

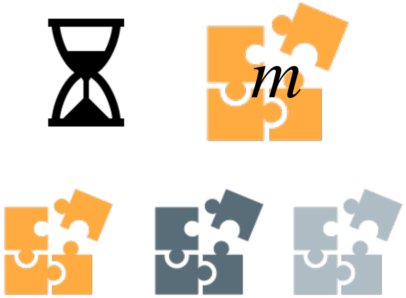
Prototype Evaluation

- For T = 50 million sequential computations*, and batching 500 puzzles, the batching time trivially would take 15 hours, while our solution takes close to 6 minutes (we did not use any parallelism for our experiments).

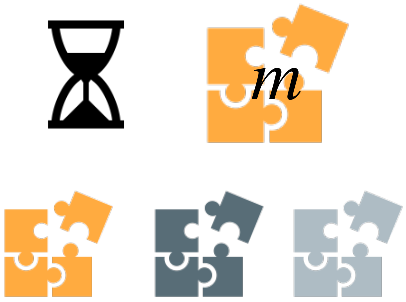
- For T = 50 million computations, and batching 7000 puzzles, the size of a single puzzle is 8 MB trivially, 37 MB using our solution and would be 790 MB using the linearly homomorphic solution.

*the time to do 50 million sequential computations on the test machine is 5 minutes

Conclusion!

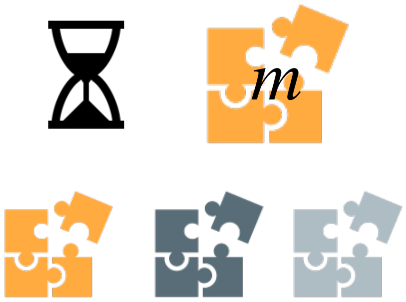


Conclusion!

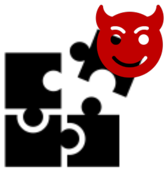


- We gave a solution template for batch solving of time-lock puzzles.

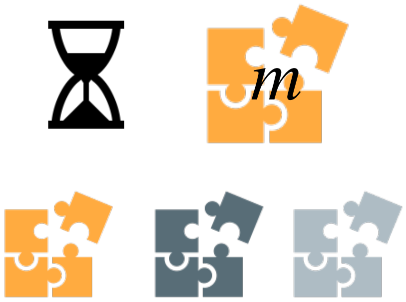
Conclusion!



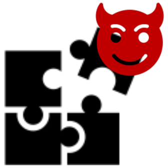
- We gave a solution template for batch solving of time-lock puzzles.
- Introduction of rogue puzzle attacks.



Conclusion!



- We gave a solution template for batch solving of time-lock puzzles.
- Introduction of rogue puzzle attacks.



- Give a concrete implementation and numbers.