CBMC-GC: An ANSI-C Compiler for Secure Two-Party Computations

Martin Franz
Deutsche Bank

Andreas Holzer
TU Wien

Stefan Katzenbeisser
TU Darmstadt & CASED

Christian Schallhart
Oxford University

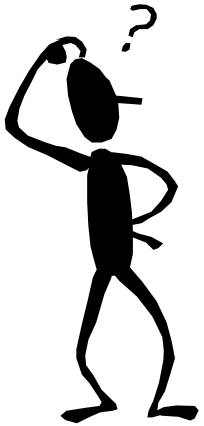
Helmut Veith
TU Wien

Outline

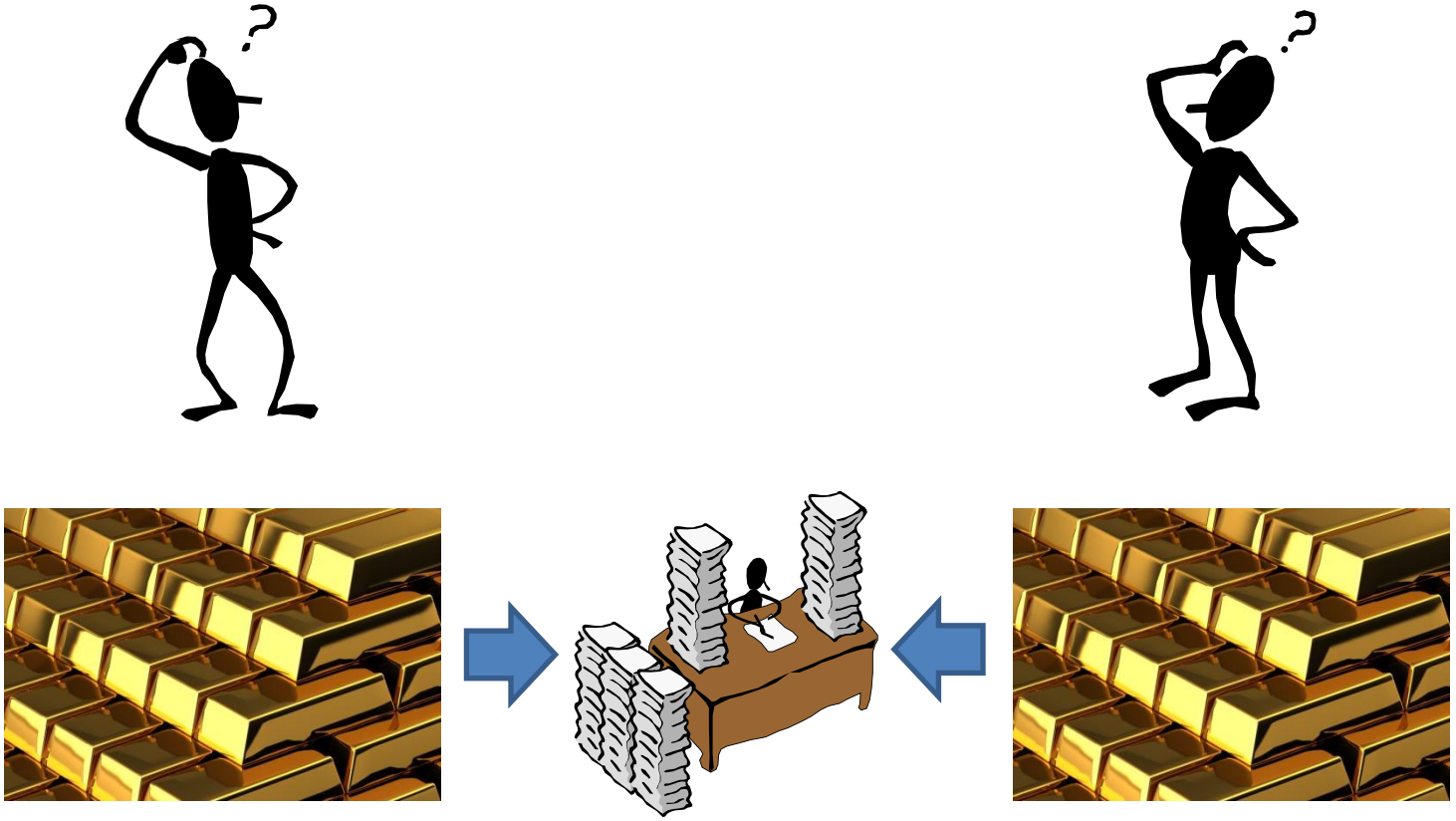
1. Introduction to Secure Two-Party Computations
2. CBMC-GC
3. Demo
4. Research Challenges

Yao‘s Millionaires Problem

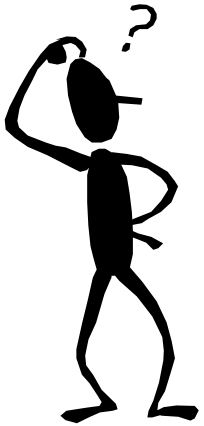
Yao‘s Millionaires Problem



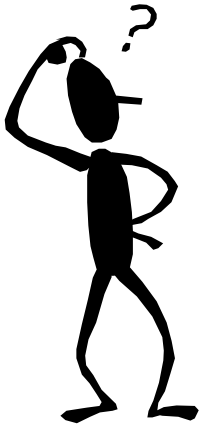
Yao‘s Millionaires Problem



Yao‘s Millionaires Problem



Yao‘s Millionaires Problem



Secure Two-Party Computation

Yao‘s Millionaires Problem

Yao‘s Millionaires Problem

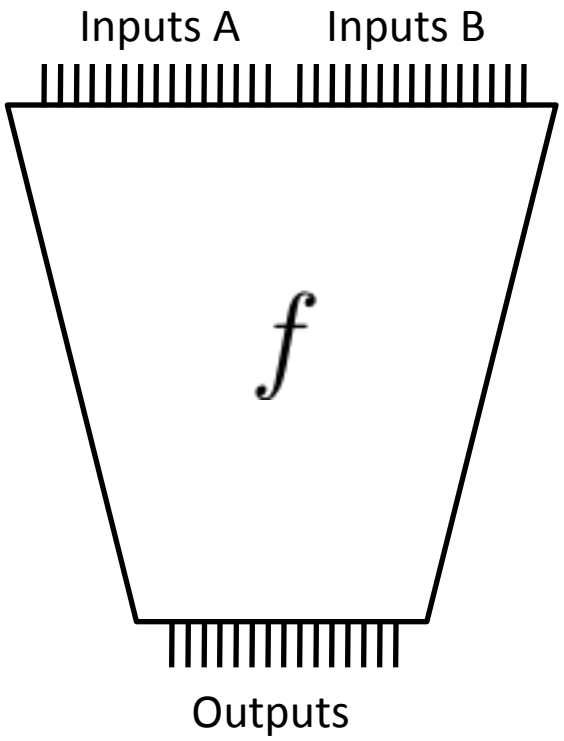
Secure Two-Party Computation

Yao‘s Millionaires Problem

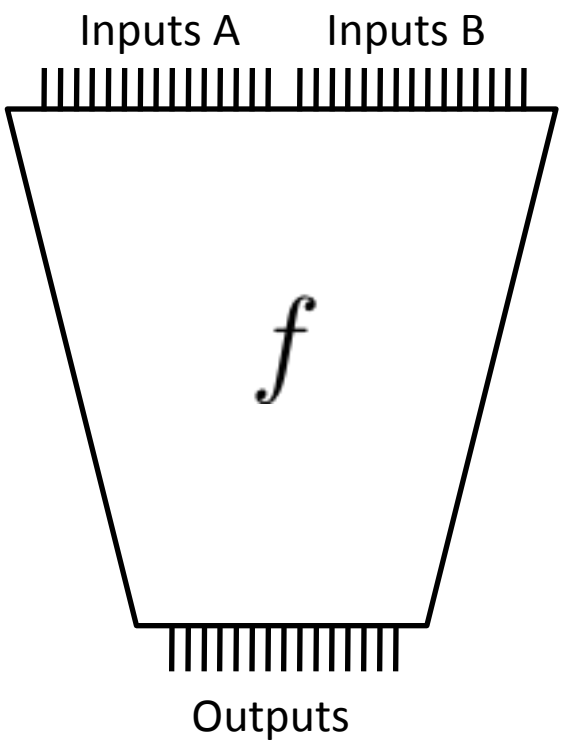
Secure Two-Party Computation

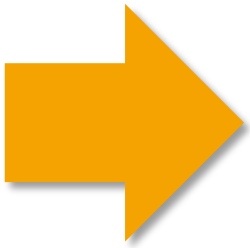
- Auctions
- DNA Analysis
- Face Recognition
- …

Yao‘s Garbled Circuits

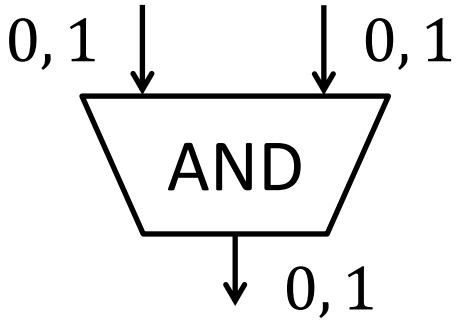


Yao‘s Garbled Circuits



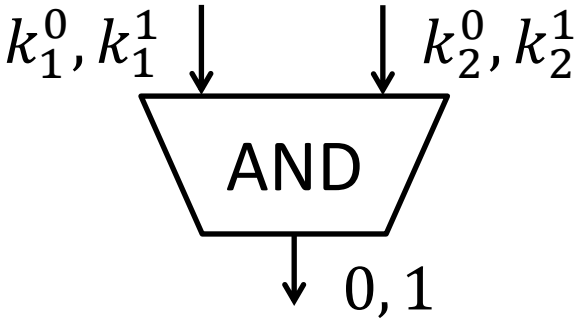


Yao‘s Garbled Circuits



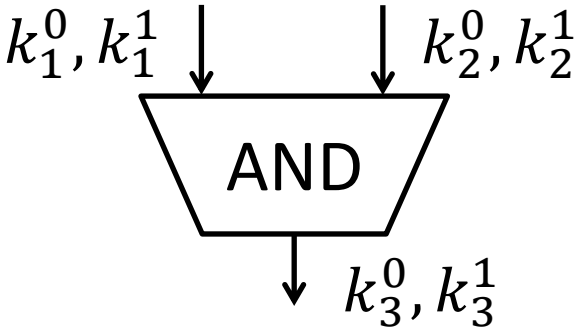
$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

Yao‘s Garbled Circuits



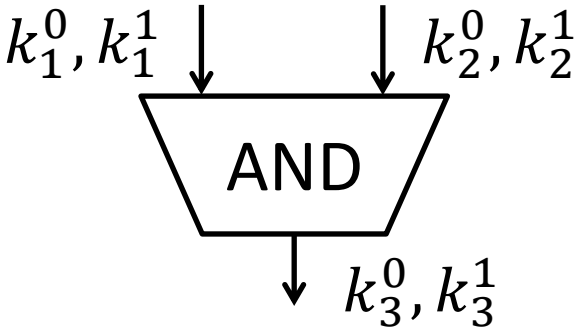
$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

Yao‘s Garbled Circuits



$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

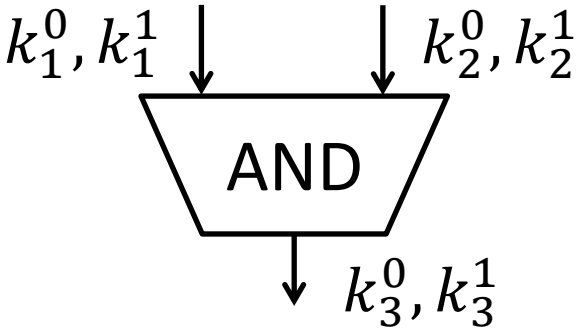
Yao‘s Garbled Circuits



$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

$\wedge$	$x = 1$	$x = 0$
$y = 1$		
$y = 0$		

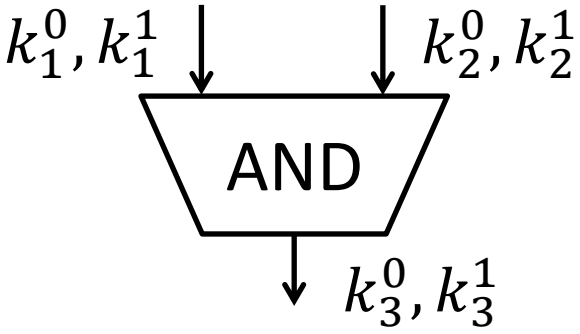
Yao‘s Garbled Circuits



$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

$\wedge$	$x = 1$	$x = 0$
$y = 1$	$E(k_1^1, E(k_2^1, k_3^1))$	
$y = 0$		

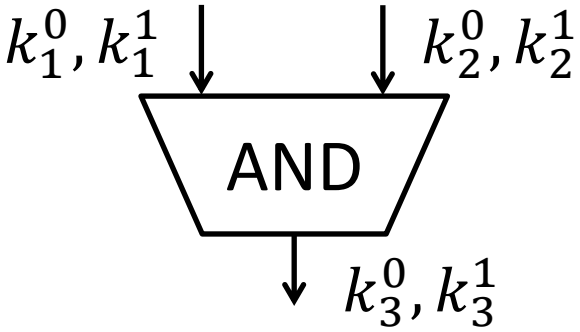
Yao‘s Garbled Circuits



$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

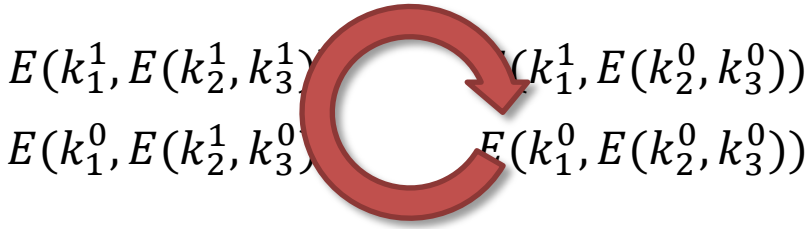
$\wedge$	$x = 1$	$x = 0$
$y = 1$	$E(k_1^1, E(k_2^1, k_3^1))$	$E(k_1^1, E(k_2^0, k_3^0))$
$y = 0$	$E(k_1^0, E(k_2^1, k_3^0))$	$E(k_1^0, E(k_2^0, k_3^0))$

Yao‘s Garbled Circuits

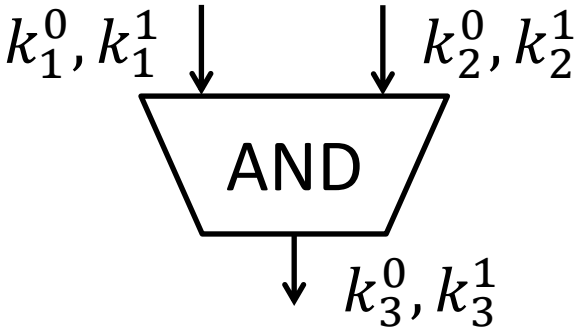


$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

Shuffle



Yao‘s Garbled Circuits

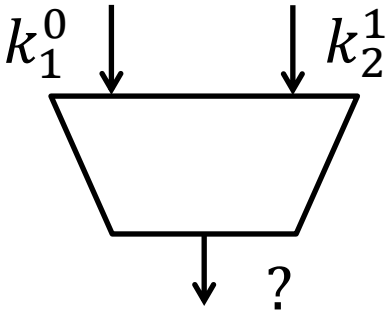


$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

$E(k_1^1, E(k_2^0, k_3^0))$ $E(k_1^0, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^1, k_3^0))$ $E(k_1^1, E(k_2^1, k_3^1))$

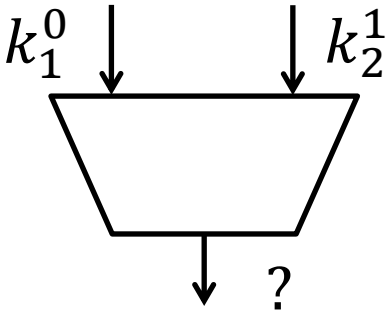
Yao‘s Garbled Circuits



$E(k_1^1, E(k_2^0, k_3^0))$ $E(k_1^0, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^1, k_3^0))$ $E(k_1^1, E(k_2^1, k_3^1))$

Yao‘s Garbled Circuits



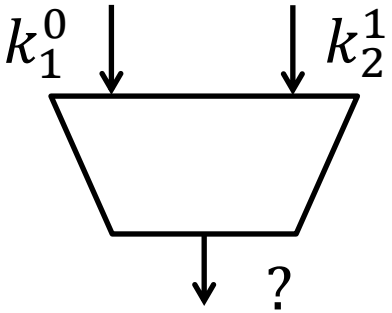
$E(k_1^1, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^1, k_3^0))$

$E(k_1^1, E(k_2^1, k_3^1))$

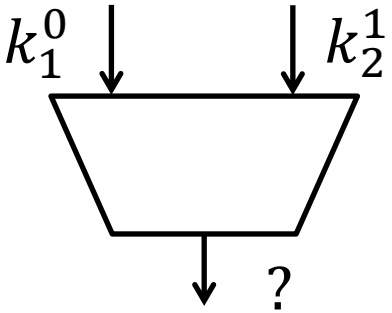
Yao‘s Garbled Circuits



$$D\left(k_2^1, D\left(k_1^0, \boldsymbol{E}\left(\boldsymbol{k_1^1}, \boldsymbol{E}\left(\boldsymbol{k_2^0}, \boldsymbol{k_3^0}\right)\right)\right)\right)$$

$E(k_1^1, E(k_2^0, k_3^0))$	$E(k_1^0, E(k_2^0, k_3^0))$
$E(k_1^0, E(k_2^1, k_3^0))$	$E(k_1^1, E(k_2^1, k_3^1))$

Yao‘s Garbled Circuits



$$D\left(k_2^1, D\left(k_1^0, \boldsymbol{E}\left(\boldsymbol{k_1^1}, \boldsymbol{E}\left(\boldsymbol{k_2^0}, \boldsymbol{k_3^0}\right)\right)\right)\right)$$

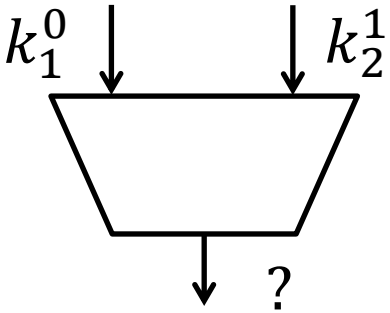
✖ $E(k_1^1, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^1, k_3^0))$

$E(k_1^1, E(k_2^1, k_3^1))$

Yao‘s Garbled Circuits



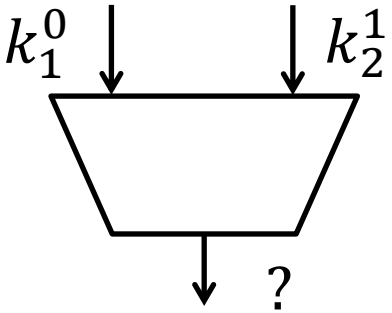
✖ $E(k_1^1, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^1, k_3^0))$

$E(k_1^1, E(k_2^1, k_3^1))$

Yao‘s Garbled Circuits



$$D\left(k_2^1, D\left(k_1^0, \boldsymbol{E}\left(\boldsymbol{k_1^0}, \boldsymbol{E}\left(\boldsymbol{k_2^1}, \boldsymbol{k_3^0}\right)\right)\right)\right)$$

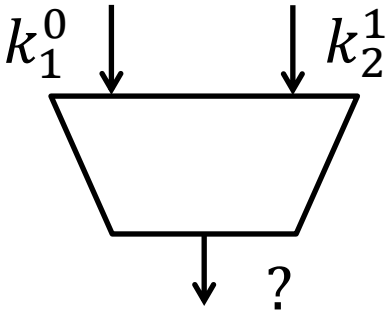
✖ $E(k_1^1, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^0, k_3^0))$

$E(k_1^0, E(k_2^1, k_3^0))$

$E(k_1^1, E(k_2^1, k_3^1))$

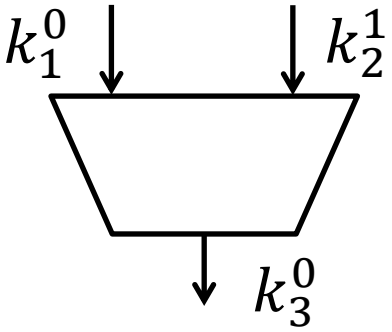
Yao‘s Garbled Circuits



$$D\left(k_2^1, D\left(k_1^0, \boldsymbol{E}\left(\boldsymbol{k_1^0}, \boldsymbol{E}\left(\boldsymbol{k_2^1}, \boldsymbol{k_3^0}\right)\right)\right)\right)$$

✗	$E(k_1^1, E(k_2^0, k_3^0))$	$E(k_1^0, E(k_2^0, k_3^0))$
✓	$E(k_1^0, E(k_2^1, k_3^0))$	$E(k_1^1, E(k_2^1, k_3^1))$

Yao‘s Garbled Circuits



$$D\left(k_2^1, D\left(k_1^0, \boldsymbol{E}\left(\boldsymbol{k_1^0}, \boldsymbol{E}\left(\boldsymbol{k_2^1}, \boldsymbol{k_3^0}\right)\right)\right)\right)$$

✗ $E(k_1^1, E(k_2^0, k_3^0))$ $E(k_1^0, E(k_2^0, k_3^0))$

✓ $E(k_1^0, E(k_2^1, k_3^0))$ $E(k_1^1, E(k_2^1, k_3^1))$

Yao‘s Garbled Circuits

Secure Two-Party Computation (STC)

Party A

Party B

Yao‘s Garbled Circuits

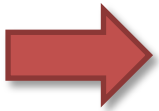
Secure Two-Party Computation (STC)

Party A

Party B

Yao‘s Garbled Circuits

Secure Two-Party Computation (STC)



Party A

Keys for Inputs of Party B/Outputs

Garbled Circuit
+ Encrypted Input for Party A

Party B

Yao‘s Garbled Circuits

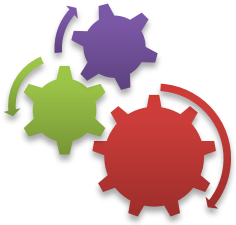
Secure Two-Party Computation (STC)

Party A

Keys for Inputs of Party B/Outputs

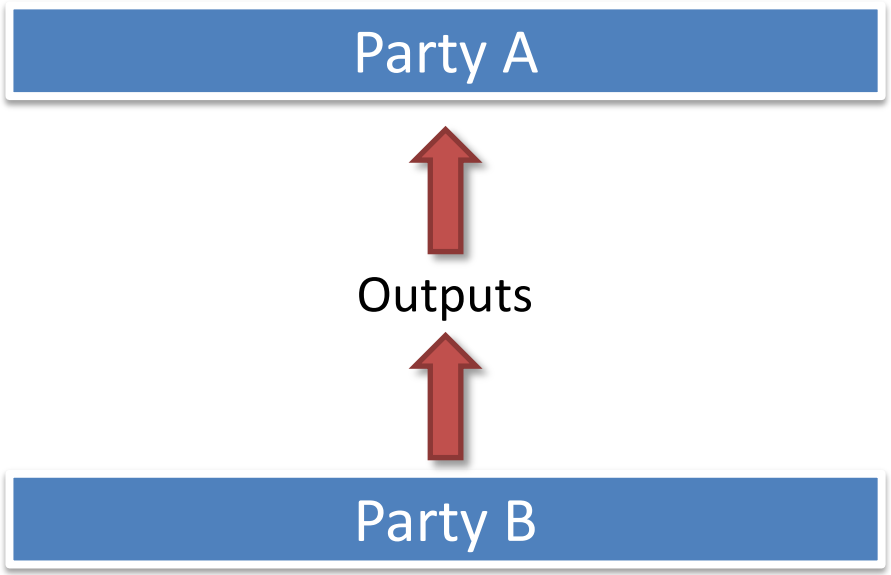
Garbled Circuit
+ Encrypted Input for Party A

Party B



Yao‘s Garbled Circuits

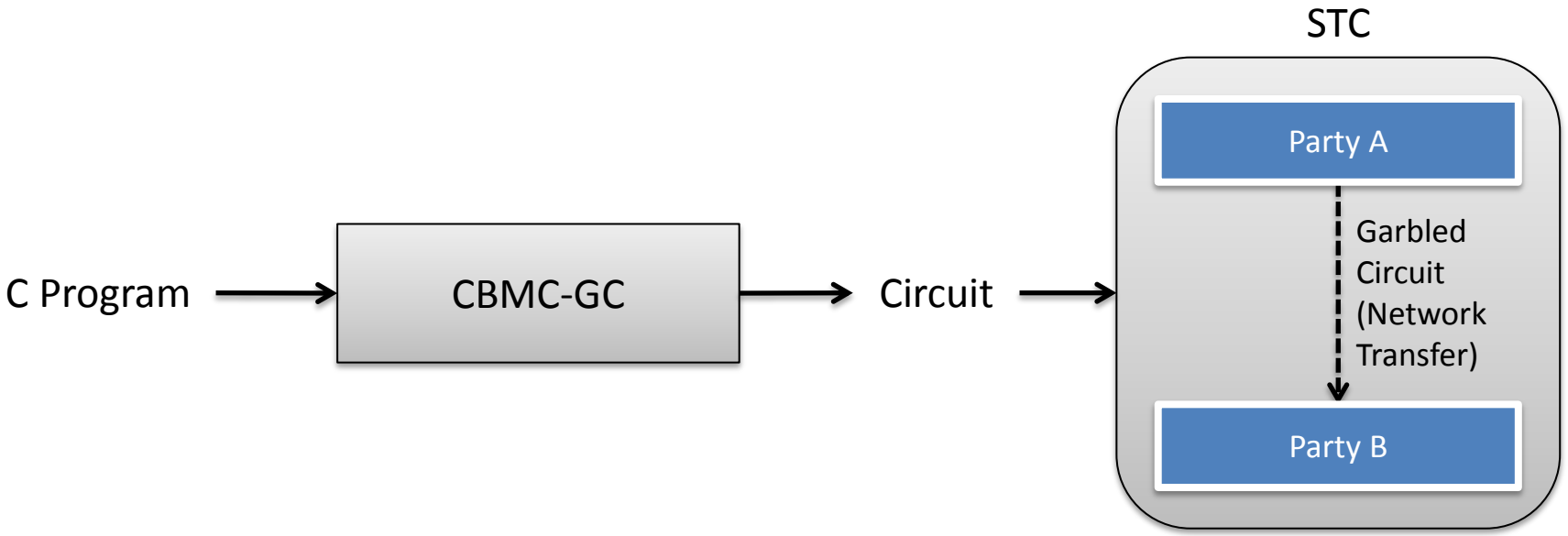
Secure Two-Party Computation (STC)



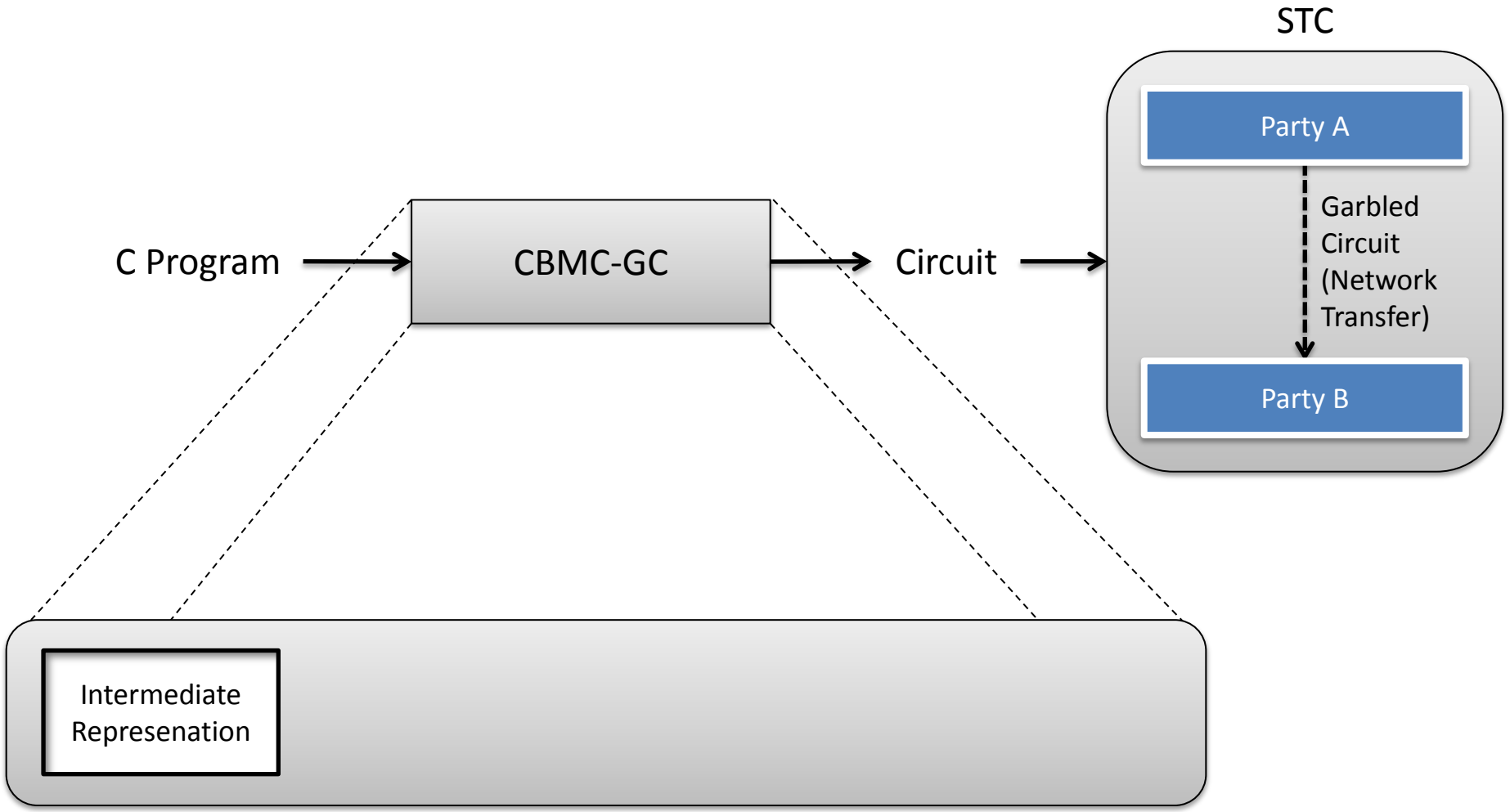
CBMC-GC

CBMC-GC

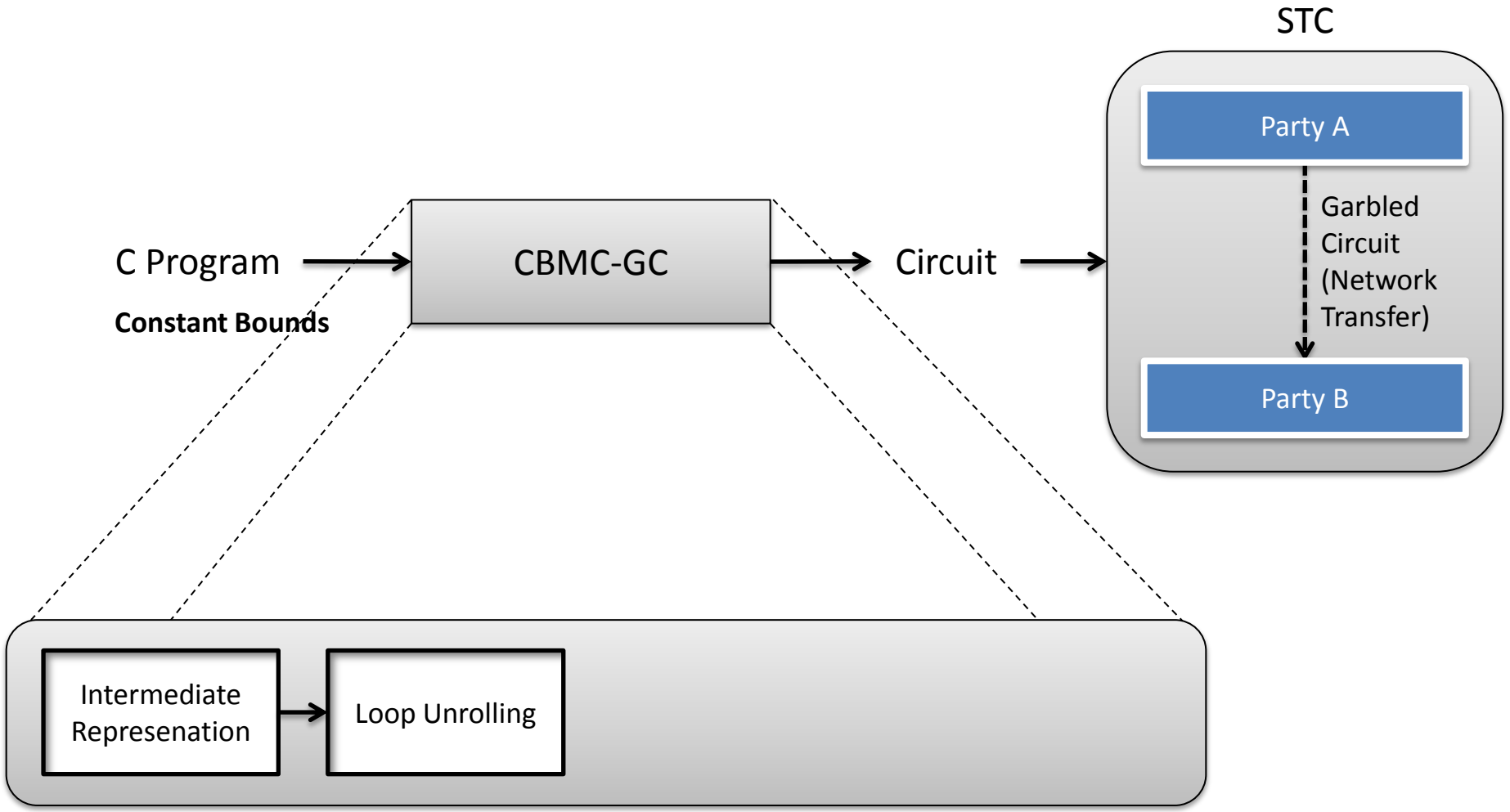
CBMC-GC


C Program
CBMC-GC
Circuit
STC
Party A
Garbled
Circuit
(Network
Transfer)
Party B

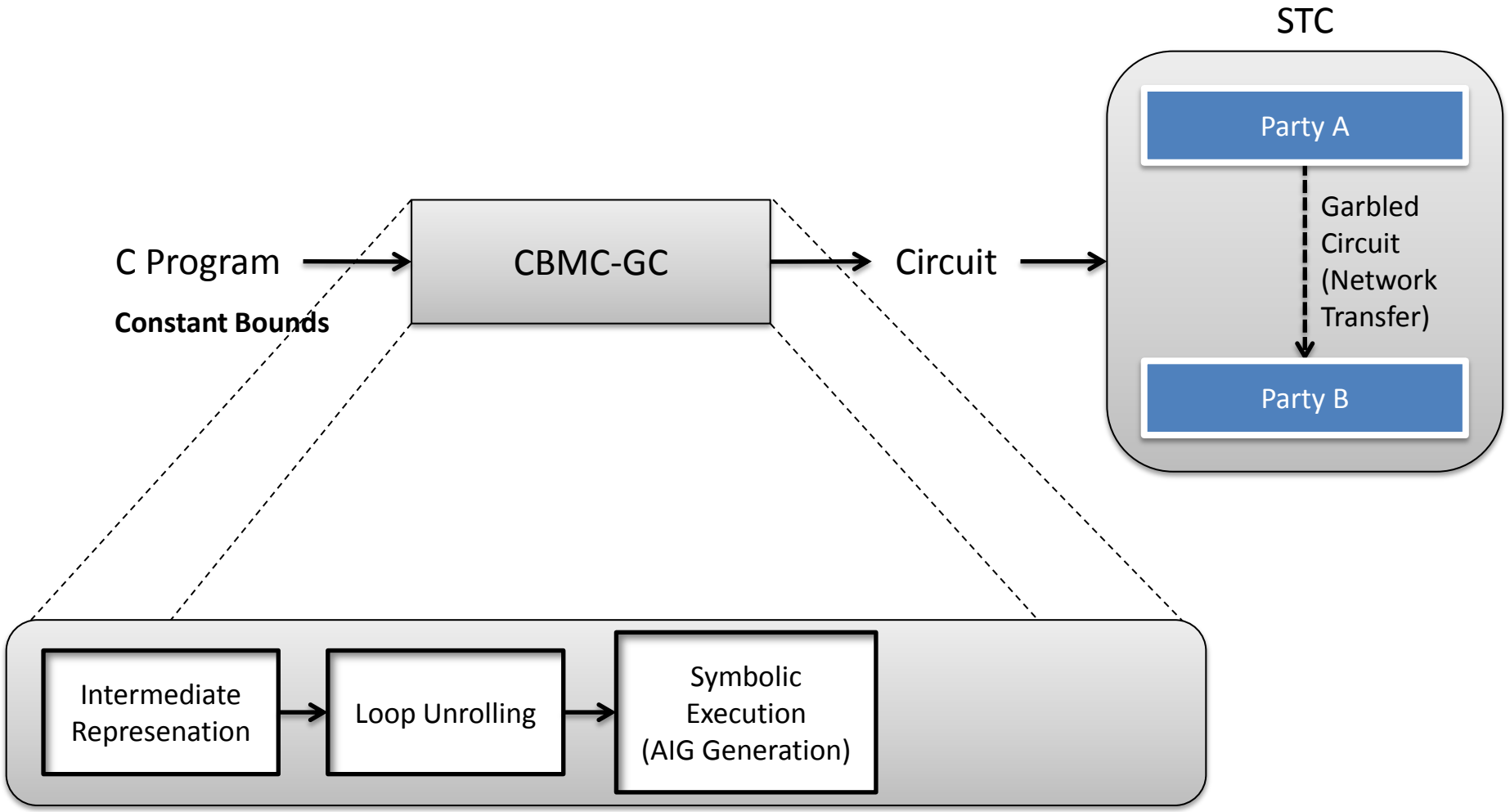
CBMC-GC


STC
Party A
Garbled Circuit (Network Transfer)
Party B
C Program
CBMC-GC
Circuit
Intermediate Represenation

CBMC-GC


STC
Party A
Garbled Circuit (Network Transfer)
Party B
C Program
Constant Bounds
CBMC-GC
Circuit
Intermediate Represenation
Loop Unrolling

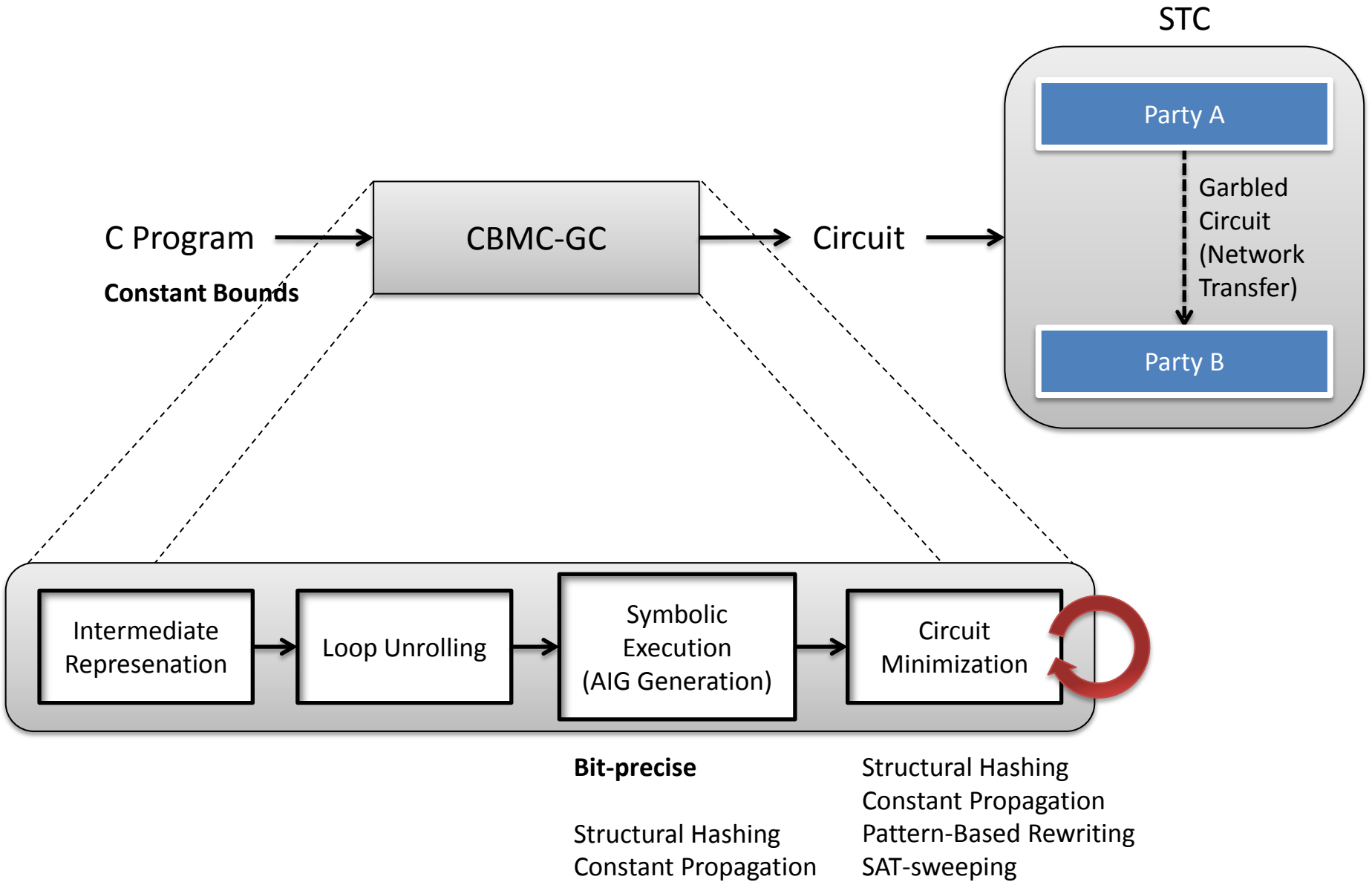
CBMC-GC



Bit-precise

Structural Hashing
Constant Propagation

CBMC-GC


STC
Party A
Garbled Circuit (Network Transfer)
Party B
C Program
Constant Bounds
CBMC-GC
Circuit
Intermediate Represenation
Loop Unrolling
Symbolic Execution (AIG Generation)
Circuit Minimization
Bit-precise
Structural Hashing
Constant Propagation
Structural Hashing
Constant Propagation
Pattern-Based Rewriting
SAT-sweeping

CBMC-GC

http://www.forsyte.at/software/cbmc-gc/

[Demo]

Research Challenges

- Circuit Optimization Algorithms
  (exact complexity unknown,
  #non-XOR gates↓, #XOR gates↑)
- Compilers for Homomorphic Encryption
- Bounds Analysis

CBMC-GC

http://www.forsyte.at/software/cbmc-gc/

THANK YOU FOR YOUR ATTENTION!