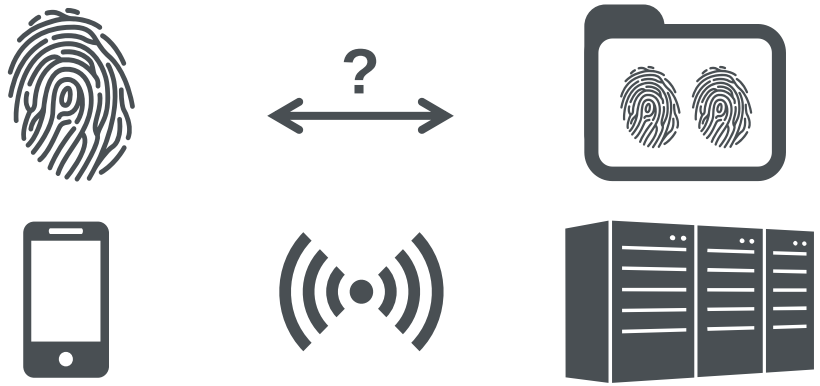
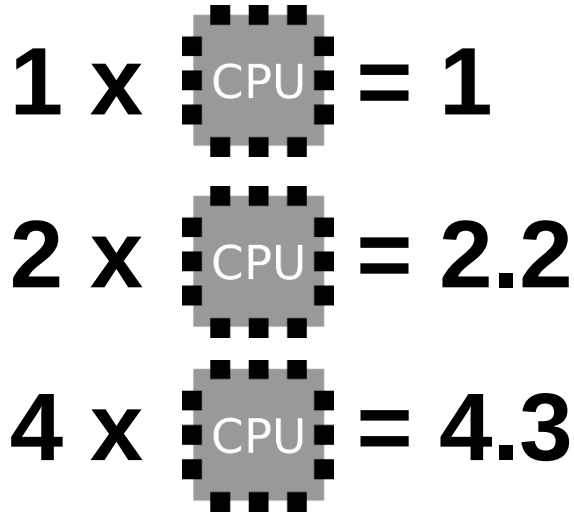
Faster Secure Computation through Automatic Parallelization

Niklas Buescher, Stefan Katzenbeissser

USENIX Security Symposium`15

TECHNISCHE UNIVERSITÄT DARMSTADT



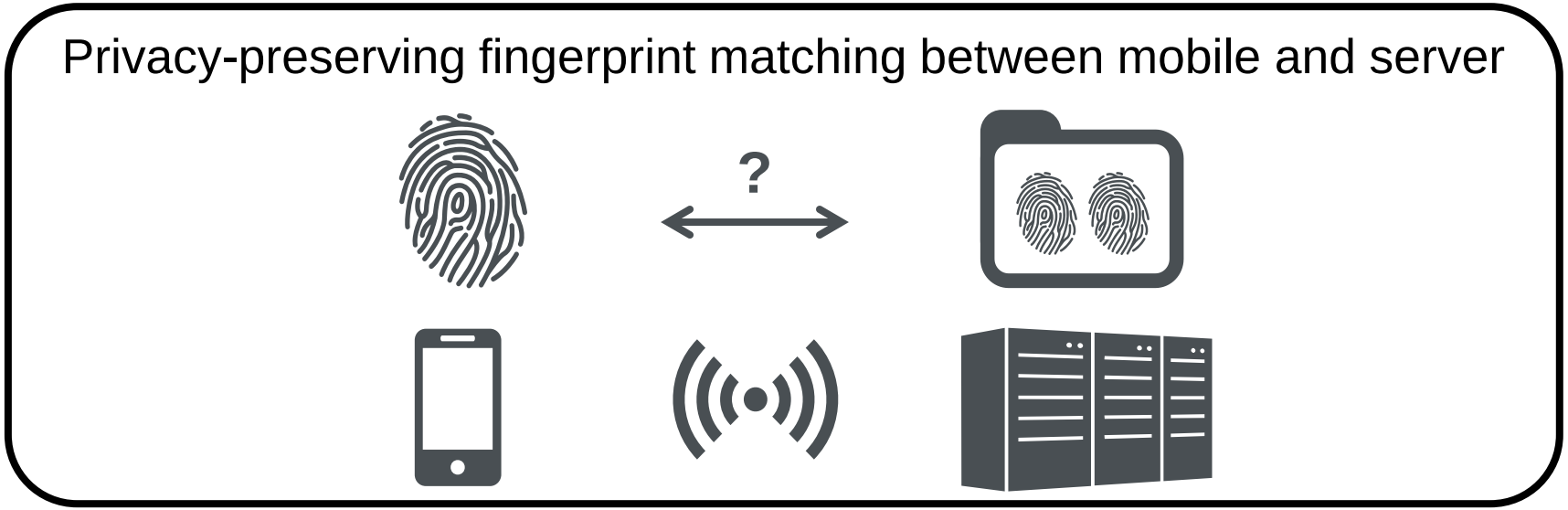


EC SPRIDE

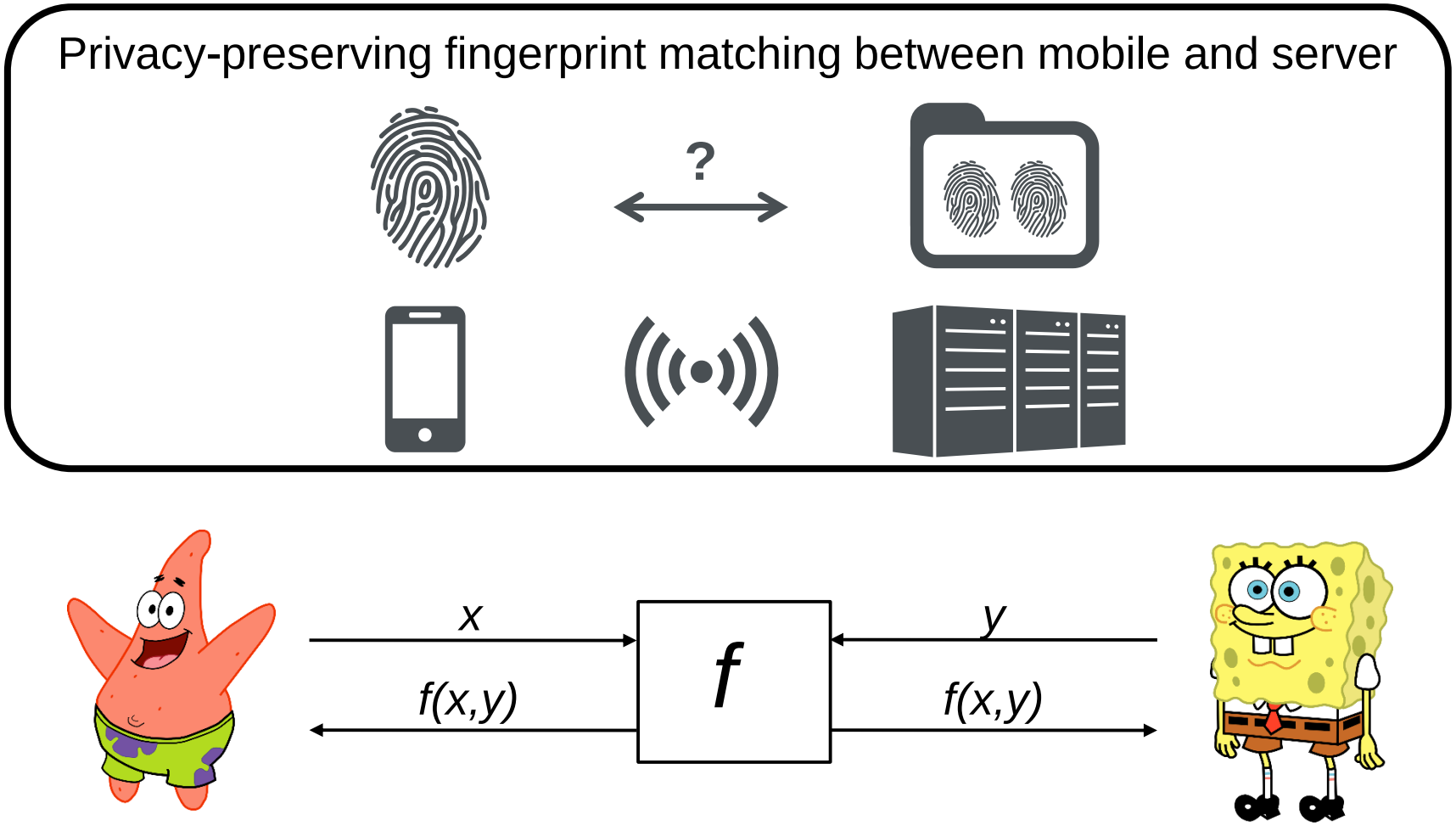
CROSSING

SECENG SECURITY ENGINEERING

Preliminaries - Secure Two-party Computation (STC)



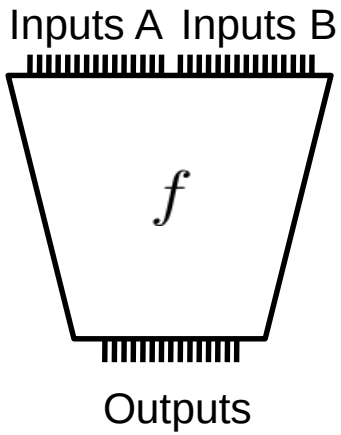
Preliminaries - Secure Two-party Computation (STC)

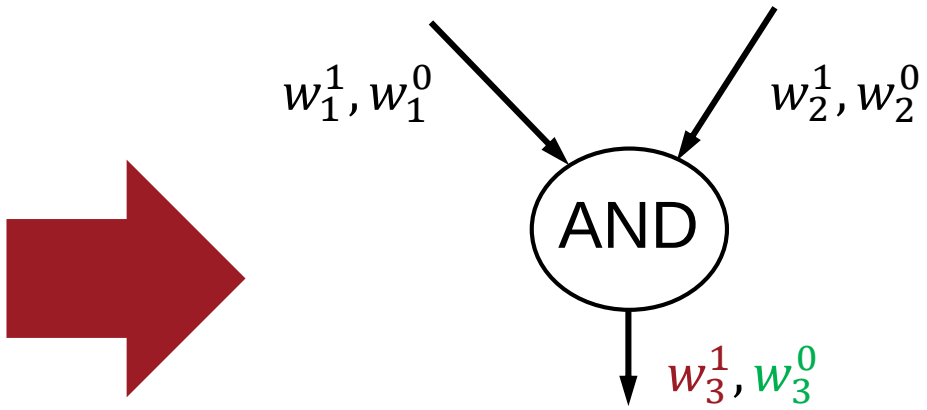


Preliminaries - Yao's Garbled Circuits
Andrew Yao

Idea

- Functionalities as Boolean circuits
- Bits → random wire labels
- Gates → garbled truth tables (GTT)



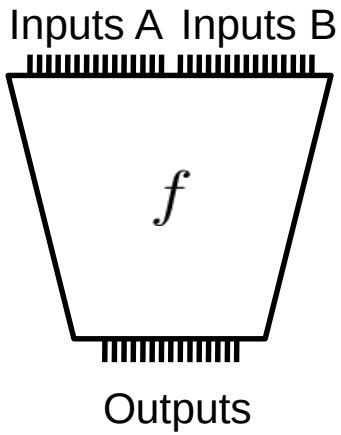


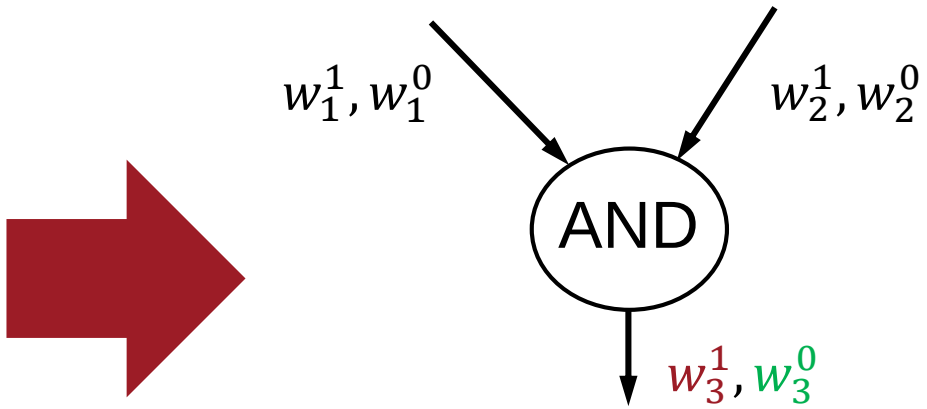
$\wedge$	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

Preliminaries - Yao's Garbled Circuits
Andrew Yao

Idea

- Functionalities as Boolean circuits
- Bits → random wire labels
- Gates → garbled truth tables (GTT)



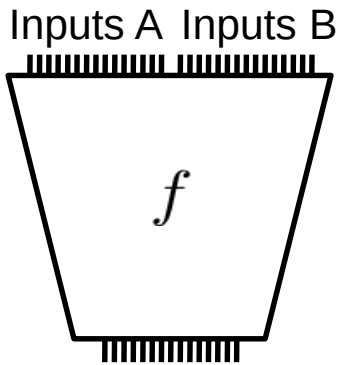


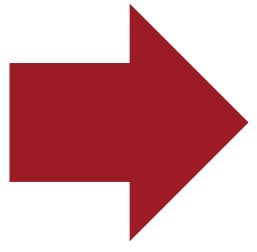
$\wedge$	$x=1$	$x=0$
$y=1$	1	0
$y=0$	0	0

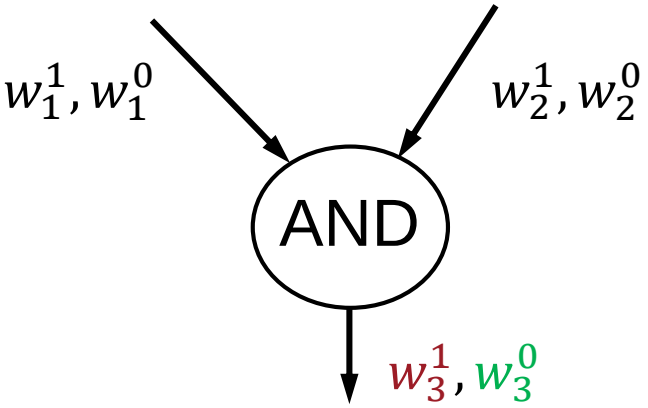
	$x=1$	$x=0$
$y=1$	$E(w_1^1, E(w_2^1, w_3^1))$	$E(w_1^0, E(w_2^1, w_3^0))$
$y=0$	$E(w_1^1, E(w_2^0, w_3^0))$	$E(w_1^0, E(w_2^0, w_3^0))$

Preliminaries - Yao's Garbled Circuits
Andrew Yao

Idea
- Functionalities as Boolean circuits
- Bits → random wire labels
- Gates → garbled truth tables (GTT)







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



Generator



GTTs + Input



Evaluator

	$x=1$	$x=0$
$y=1$	$E(w_1^1, E(w_2^1, w_3^1))$	$E(w_1^0, E(w_2^1, w_3^0))$
$y=0$	$E(w_1^1, E(w_2^0, w_3^0))$	$E(w_1^0, E(w_2^0, w_3^0))$

Secure in the semi-honest model

Motivation - Practical Yao's Garbled Circuits

TECHNISCHE UNIVERSITÄT DARMSTADT

2004 ***Fairplay*** **[MNP04]** – Garbled row reduction (GRR), Point-and-permute

2008 ***Free-XOR*** **[KS08] –** XOR gates require no encryption/communication

2011 ***FastGC*** **[HEK11]** – Streaming

2013 ***Fixed-key Garbling*** **[BHK13]** – Garbling from a fixed-key blockcipher with AES-NI

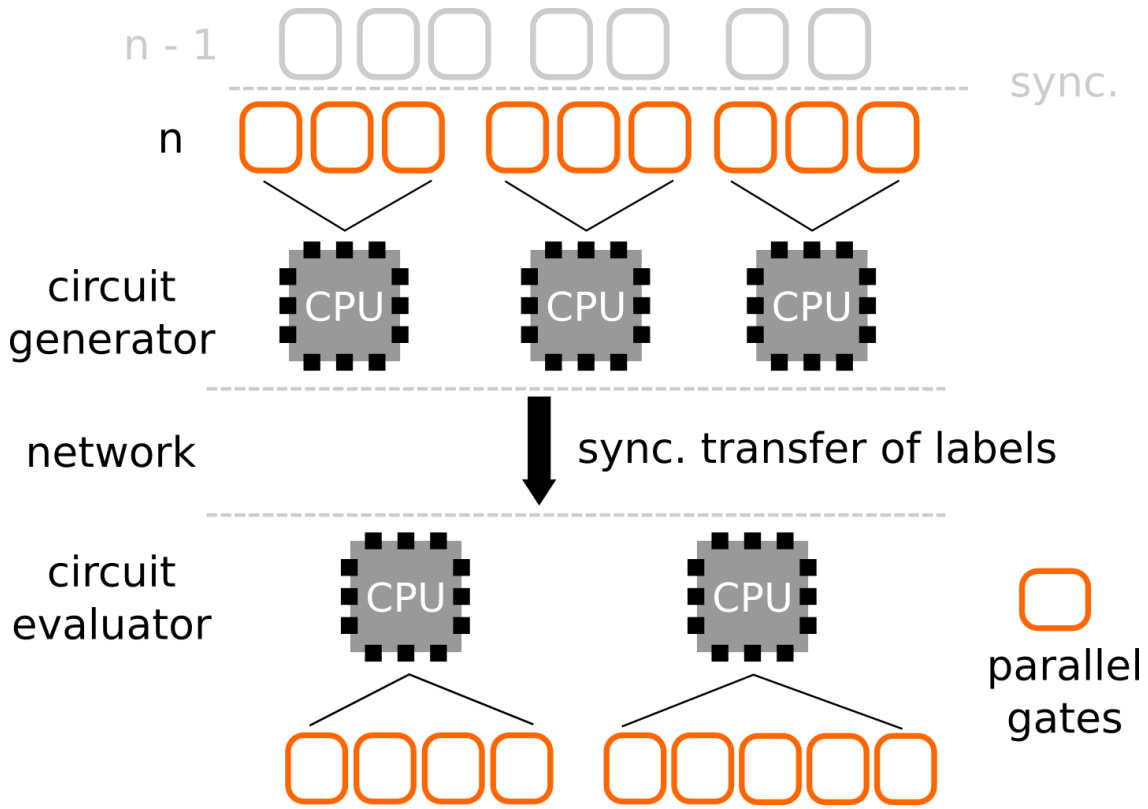
2015 ***Half-Gates*** **[ZRE15]** – Optimizations: GRR2 with Free-XOR

→ Proof: **Lower bound** on the number of ciphertexts

Still a huge gap between STC and generic computation ➔ This work: parallelization

SECENG Security Engineering

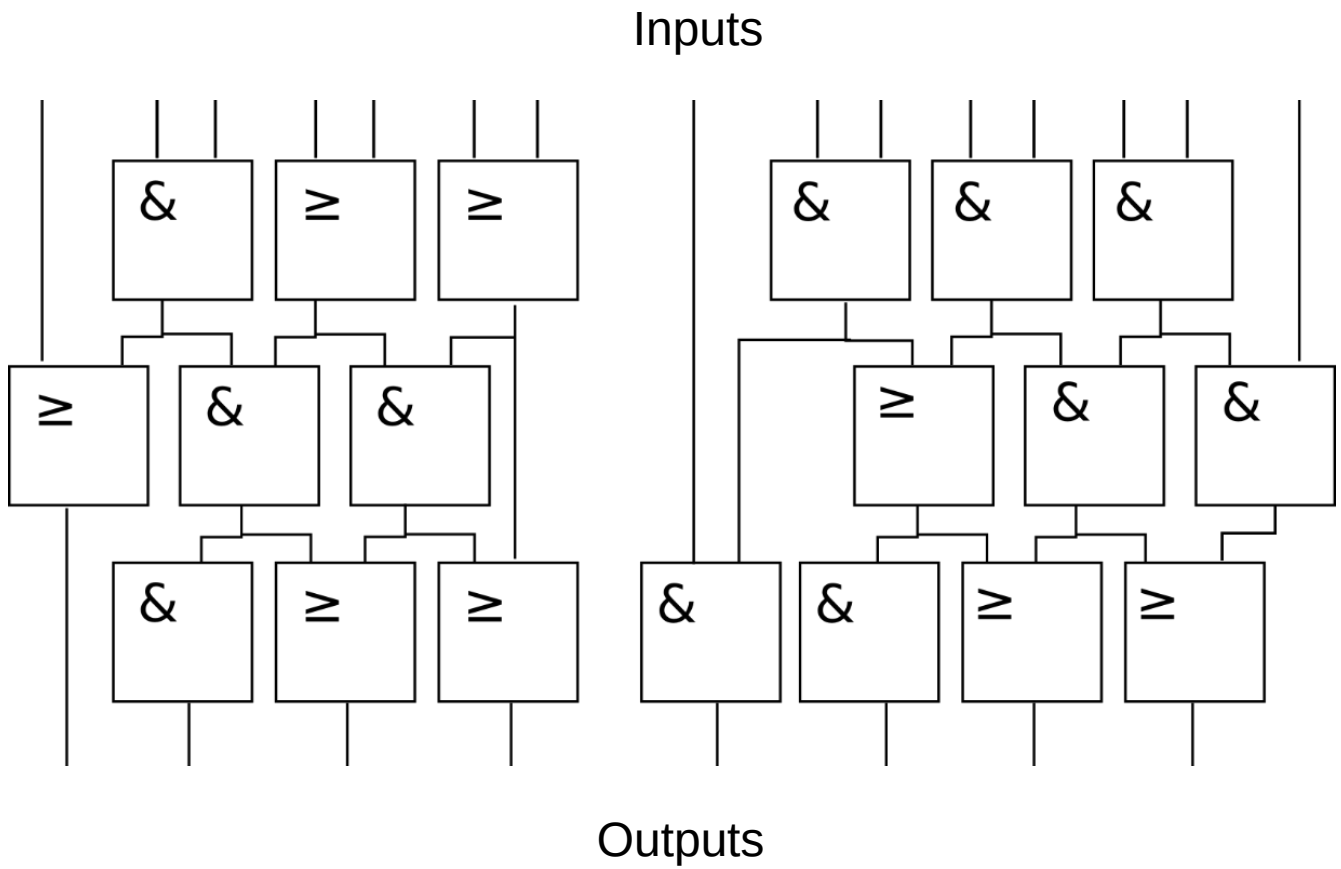
Parallel Yao Garbled Circuits



Parallel garbling: Independent gates or partitions of gates can be garbled and evaluated by any thread in any order.

No impact on security [LIP09]

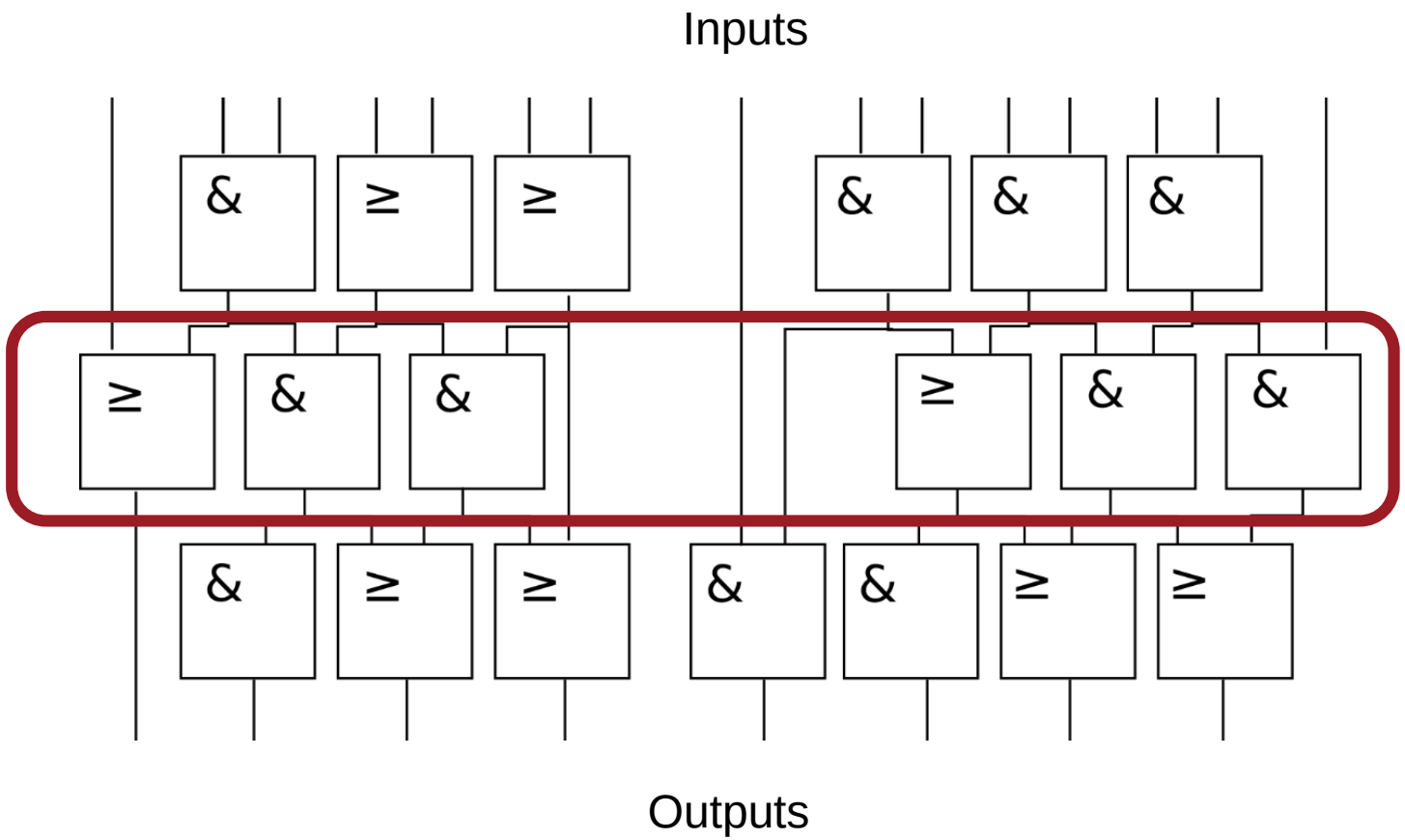
Parallelization - Circuit Decomposition



Parallelization - Circuit Decomposition

Fine-grained parallelization (FGP)

- All gates within a level (circuit depth) are independent
- Synchronization is needed after every level



Parallelization - Circuit Decomposition

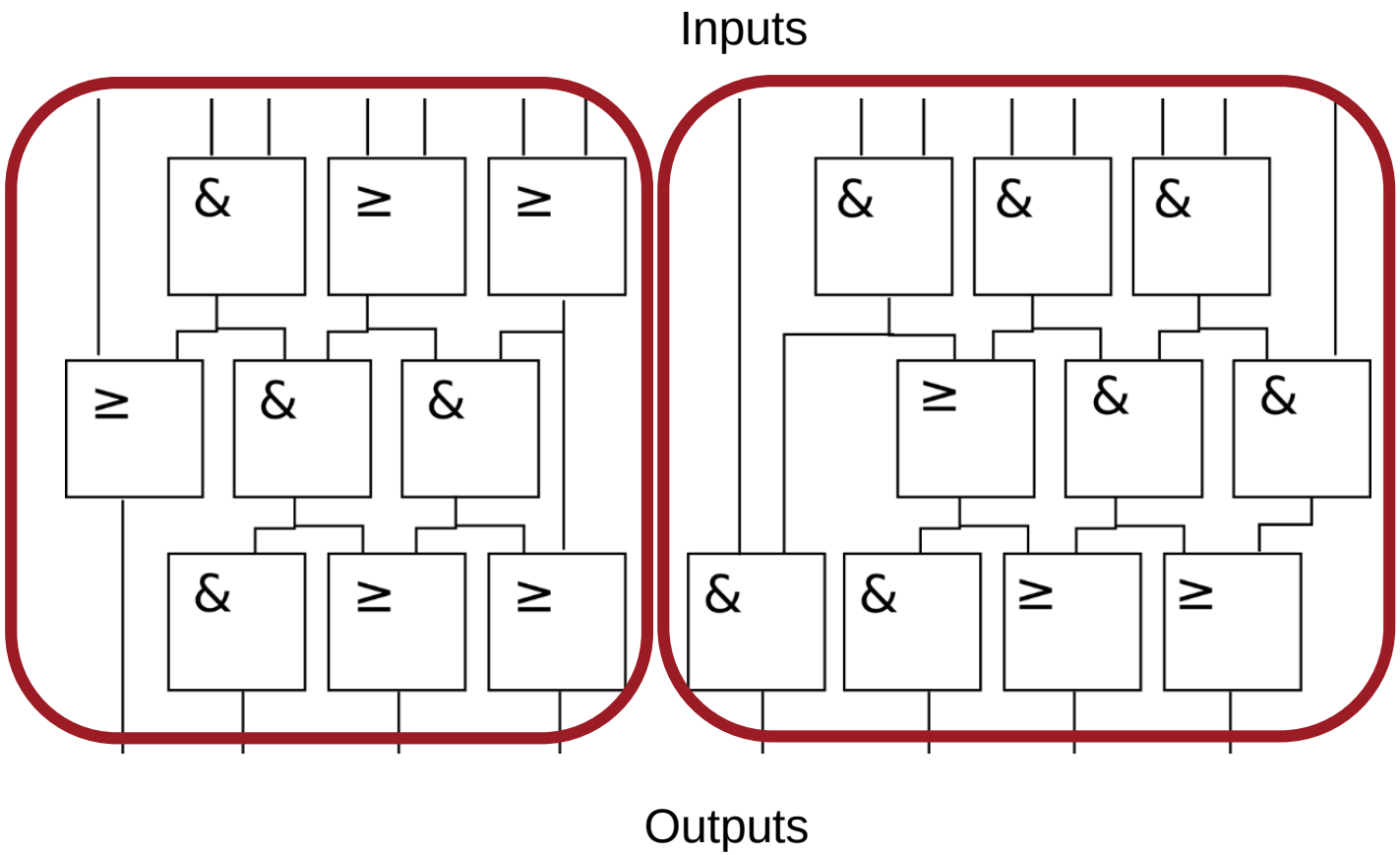
Fine-grained parallelization (FGP)

- All gates within a level (circuit depth) are independent
- Synchronization is needed after every level

Coarse grained parallelization (CGP)

- Larger coherent partitions
- NP-hard graph partitioning problem

➔ Our approach: Detect parallelism on source code level



ParCC - Parallel Circuit Compiler

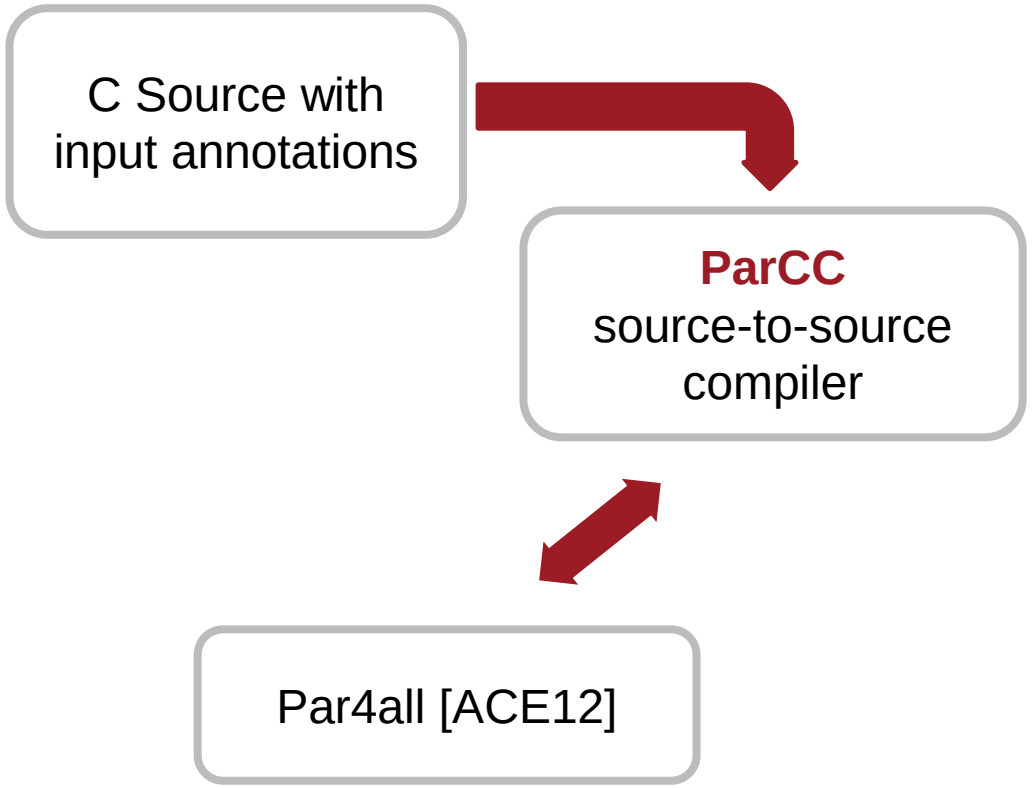
TECHNISCHE UNIVERSITÄT DARMSTADT

C Source with input annotations

```
void millionaires() {
  int INPUT_A_wealth, INPUT_B_wealth;
  int OUPUT_res;
  if(INPUT_A_wealth > INPUT_B_wealth)
    OUTPUT_res = 1;
  else
    OUTPUT_res = 0;
}
```

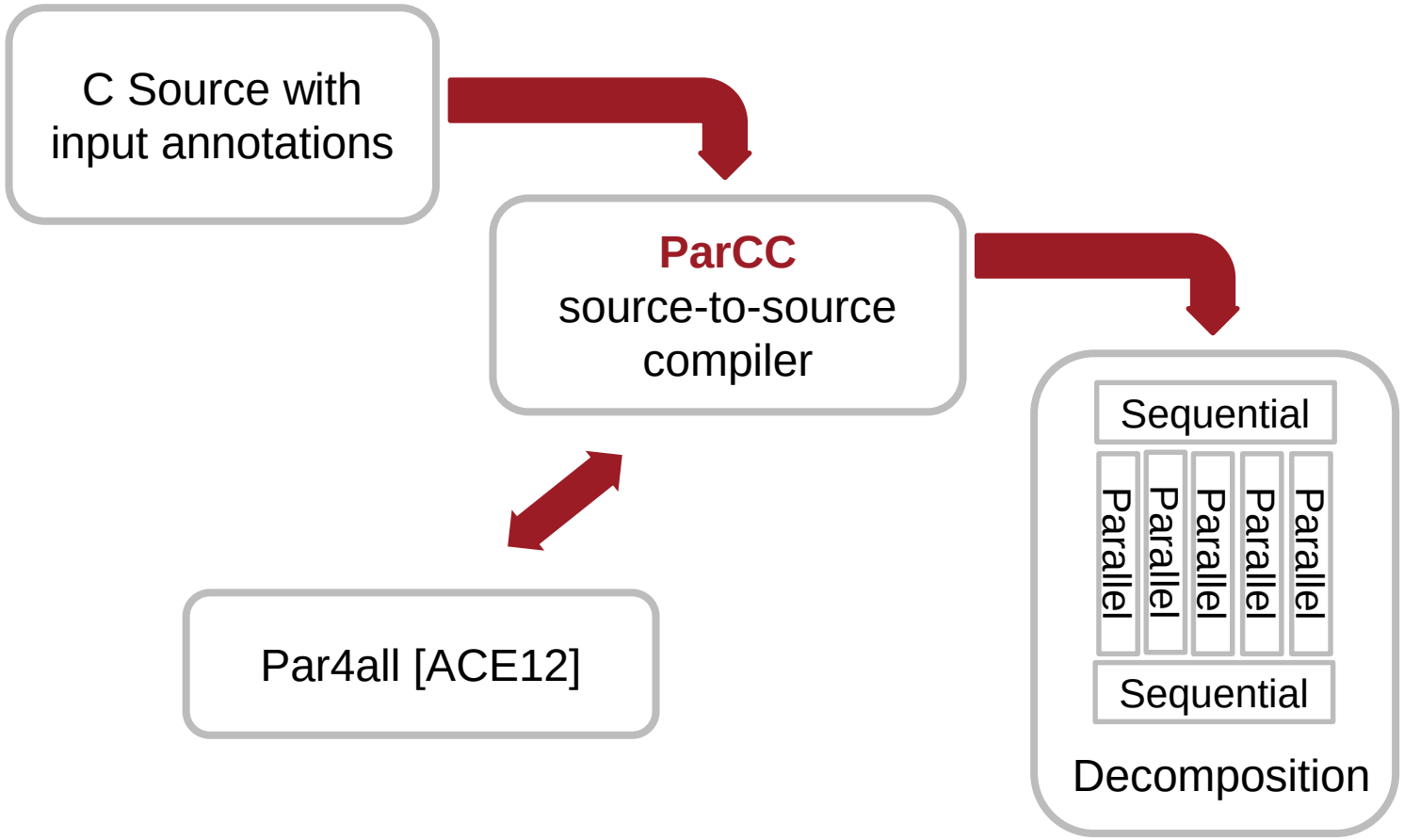

SECENG Security Engineering

ParCC - Parallel Circuit Compiler



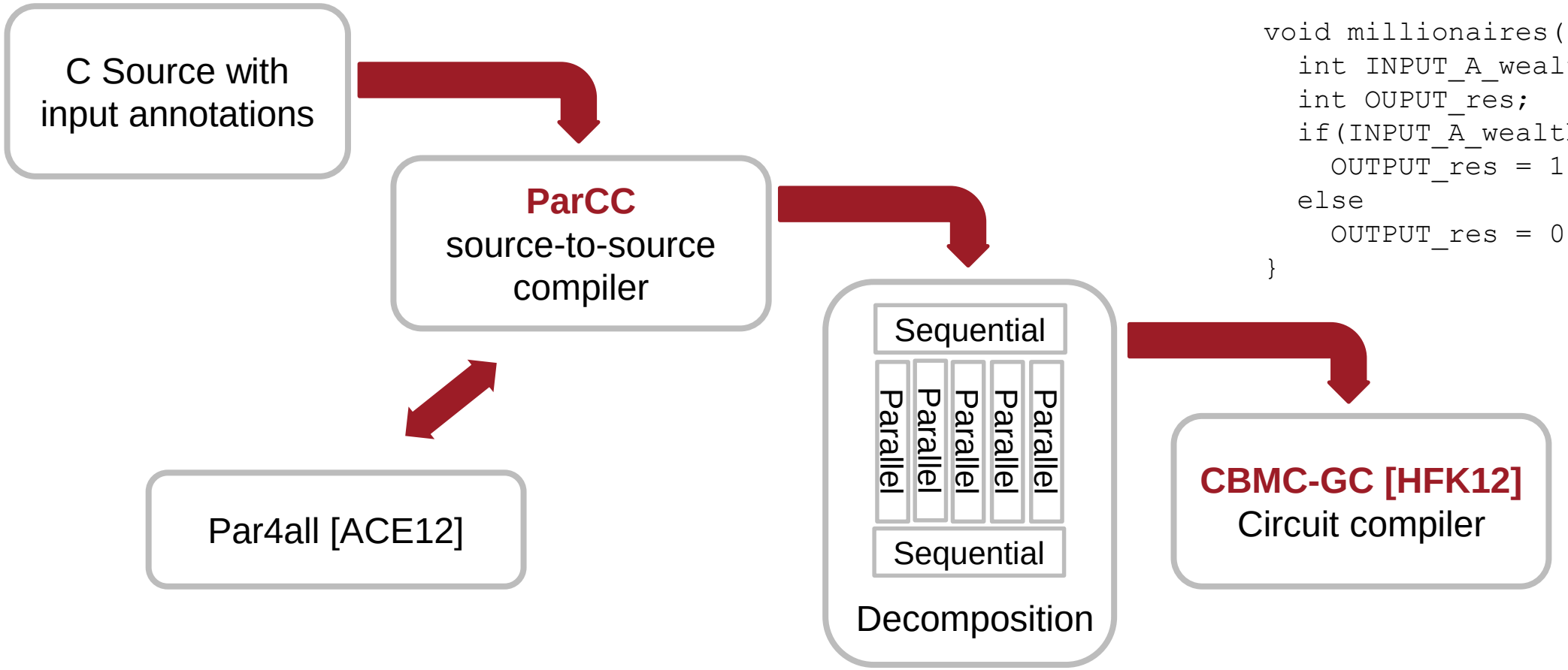

```
void millionaires() {
  int INPUT_A_wealth, INPUT_B_wealth;
  int OUPUT_res;
  if(INPUT_A_wealth > INPUT_B_wealth)
    OUTPUT_res = 1;
  else
    OUTPUT_res = 0;
}
```

ParCC - Parallel Circuit Compiler



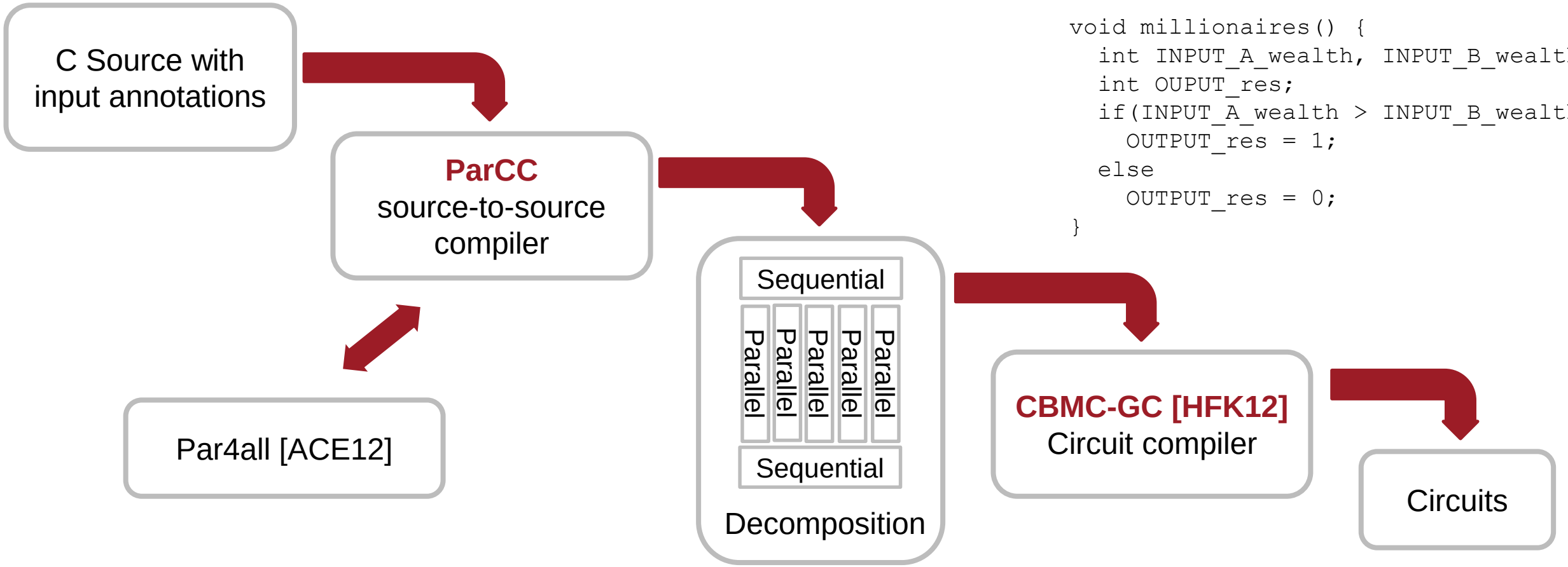

```
void millionaires() {
  int INPUT_A_wealth, INPUT_B_wealth;
  int OUPUT_res;
  if(INPUT_A_wealth > INPUT_B_wealth)
    OUTPUT_res = 1;
  else
    OUTPUT_res = 0;
}
```

ParCC - Parallel Circuit Compiler




```
void millionaires() {
  int INPUT_A_wealth, INPUT_B_wealth;
  int OUPUT_res;
  if(INPUT_A_wealth > INPUT_B_wealth)
    OUTPUT_res = 1;
  else
    OUTPUT_res = 0;
}
```

ParCC - Parallel Circuit Compiler




```
void millionaires() {
  int INPUT_A_wealth, INPUT_B_wealth;
  int OUPUT_res;
  if(INPUT_A_wealth > INPUT_B_wealth)
    OUTPUT_res = 1;
  else
    OUTPUT_res = 0;
}
```

Evaluation - Example Applications and Circuit Properties

Biometric Matching (BioMatch):

Database with n = 512 samples of degree d = 4; 64 bit integers

$$min\left(\sum_{i=1}^{d}(s_{1,i}-e_i)^2, \ldots, \sum_{i=1}^{d}(s_{n,i}-e_i)^2\right)$$

Parallel Modular Exponentiation (MExp):

32 x 32 bit modular exponentiation

Matrix-Vector Multiplication (MVMul):

16 x 16 Matrix with 64 bit integers

	BioMatch	MExp	MVMul
Circuit Size	66M	21.5M	3.3M
%AND	25%	41%	37%
Inputs bits	512 / 112K	1K / 1K	17K / 1K
Offline garbling time	2.07s	1.136s	0.154s

Parallel Framework – Implementation and Testbed

TECHNISCHE UNIVERSITÄT DARMSTADT

UltraSFE:

- Based on: *JustGarble, ME_SFE (FastGC)*
- Focus: Parallelization, low memory footprint
- Garbled row reduction, pipe-lining, fixed-key garbling with AES-NI, half-gates
- Written in C++, OpenMP and pthreads, SSE4.2

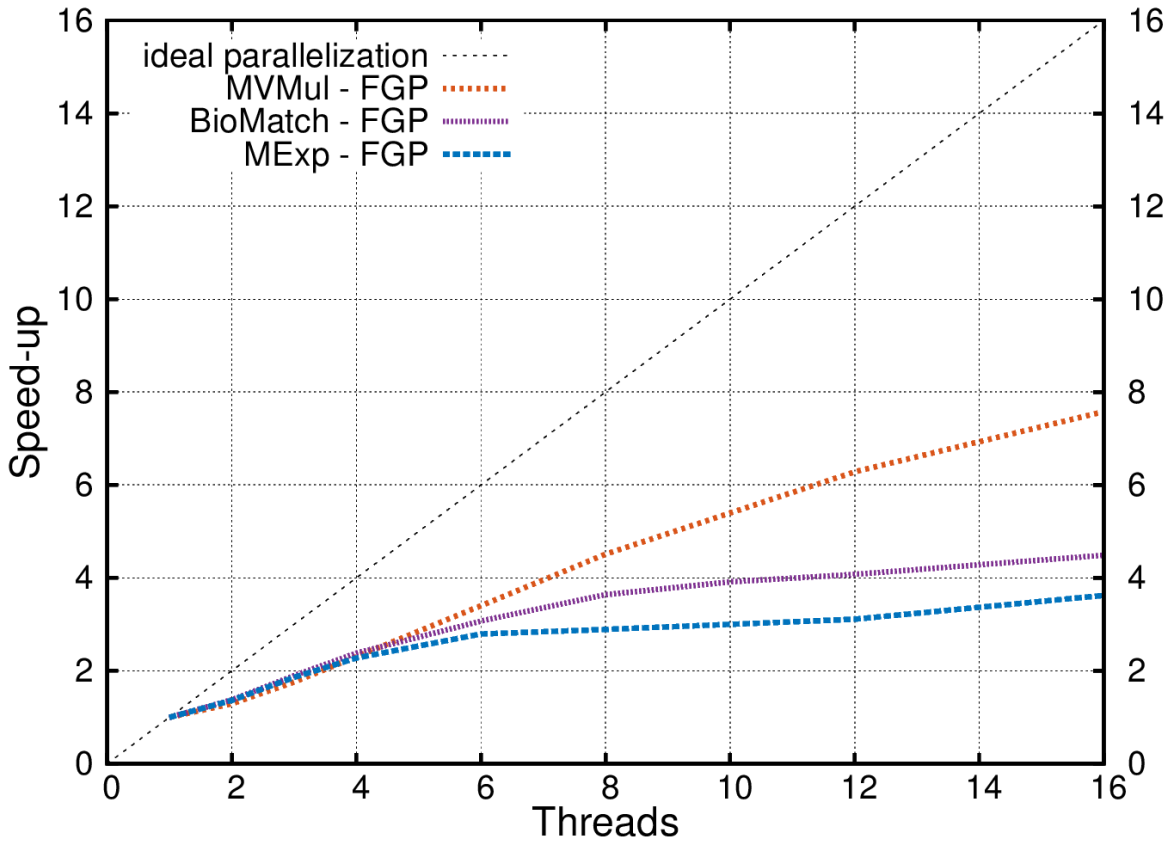


Testbed:

- Amazon EC2 – *c3.8xlarge* instance (report 16 physical cores)
- Serial garbling speed ~8M AND gates / s

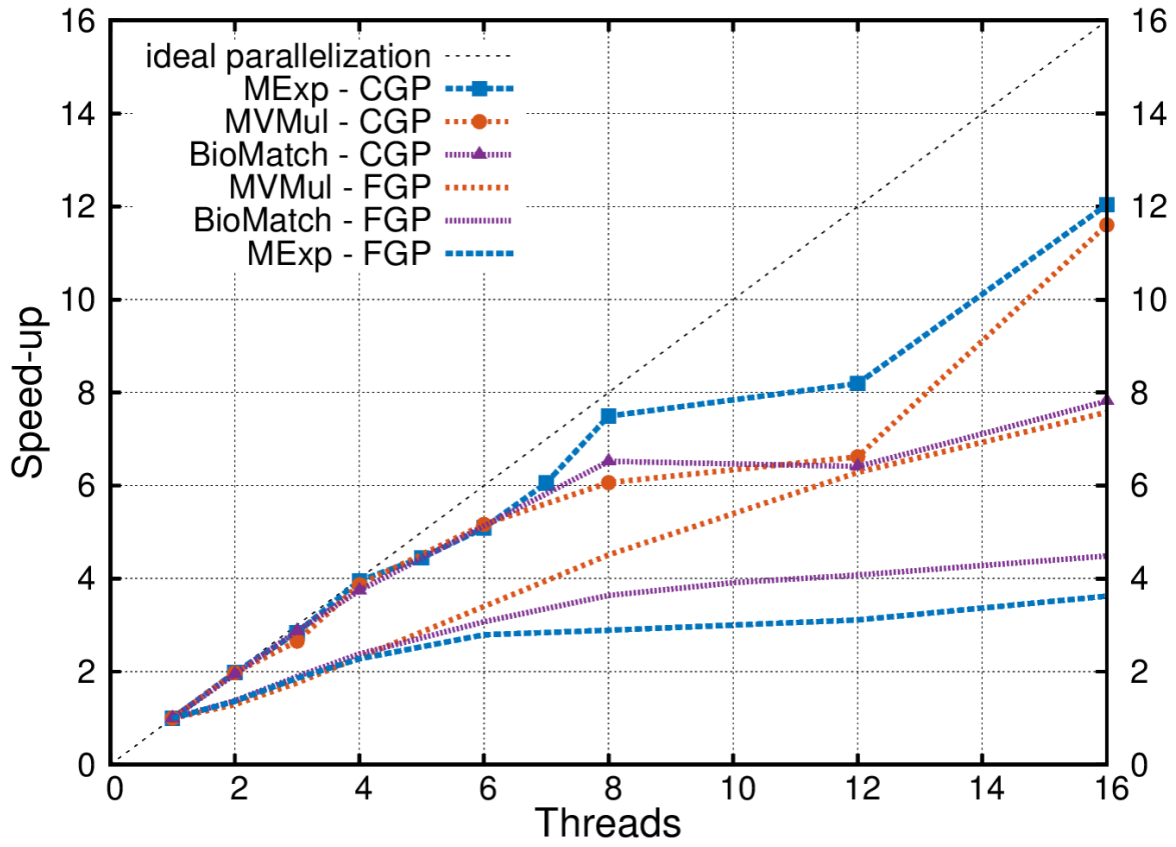
SECENG Security Engineering

Results – Circuit Garbling (offline)



Speed-up of FGP for a different number of cores.

Results – Circuit Garbling (offline)



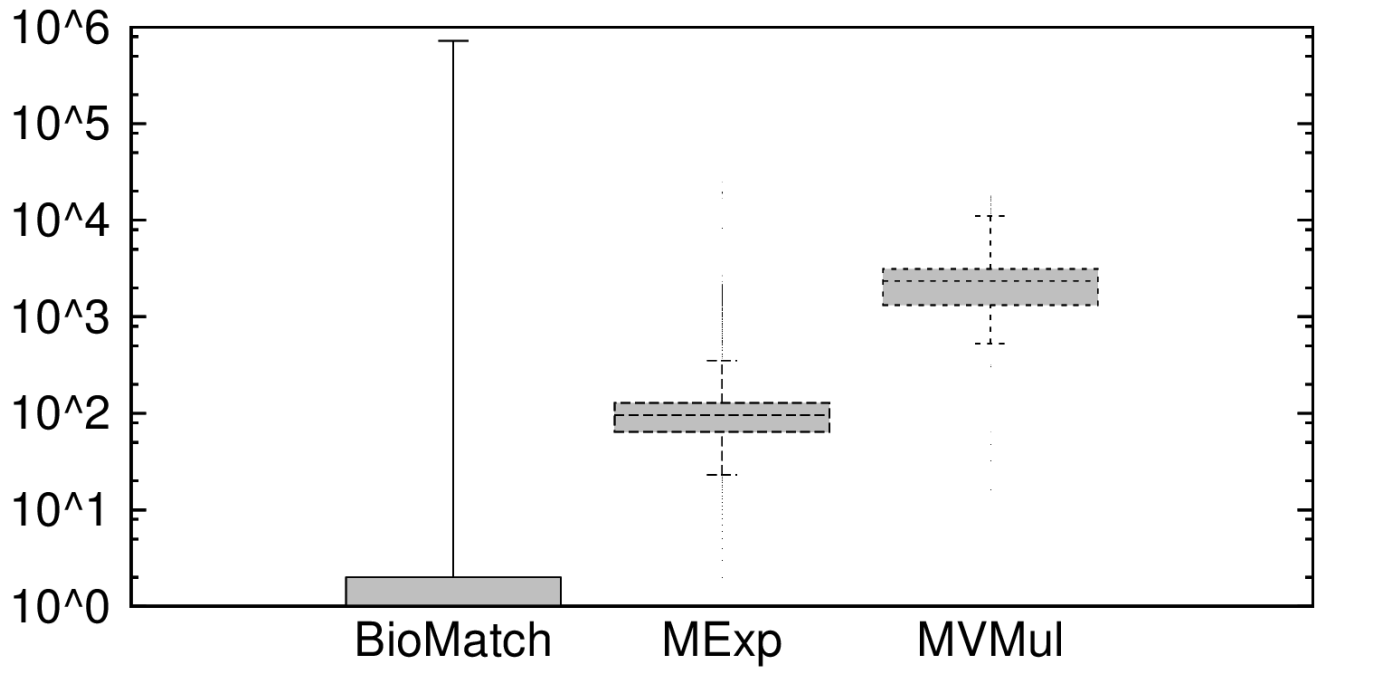
Speed-up of FGP and CGP for a different number of cores.

CGP significantly outperforms FGP

The Limits of Fine-Grained Parallelization

TECHNISCHE UNIVERSITÄT DARMSTADT

"Increasing input sizes should overcome limits of FGP" ➔ Not really:



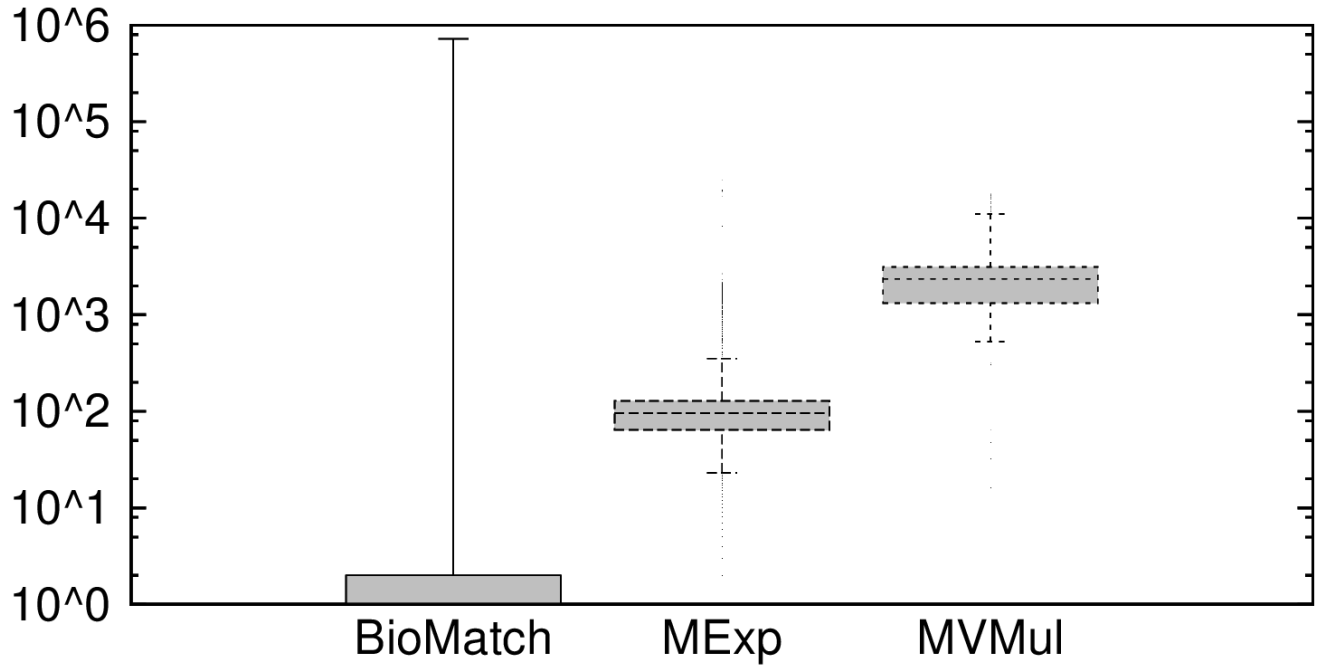
Distribution of level widths in number of non-linear gates when compiled with CBMC-GC

SECENG Security Engineering

The Limits of Fine-Grained Parallelization

TECHNISCHE UNIVERSITÄT DARMSTADT

"Increasing input sizes should overcome limits of FGP" ➔ Not really:



Distribution of level widths in number of non-linear gates when compiled with CBMC-GC

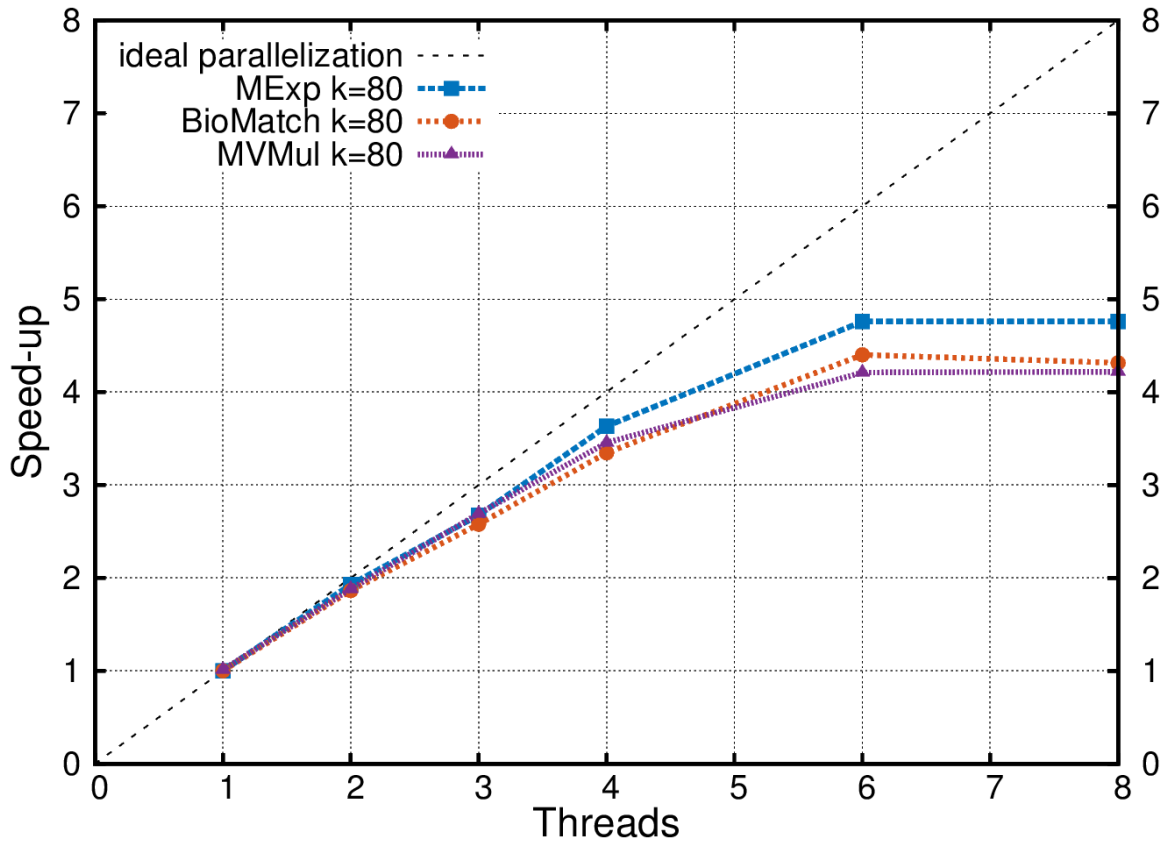
Current high level compiler are optimized for Free-XOR but not width

Examples:

- Array accesses compile to multiplexers with constant width
- Comparisons compile to circuits with constant width

SECENG Security Engineering

Results - CGP in the Online Setting

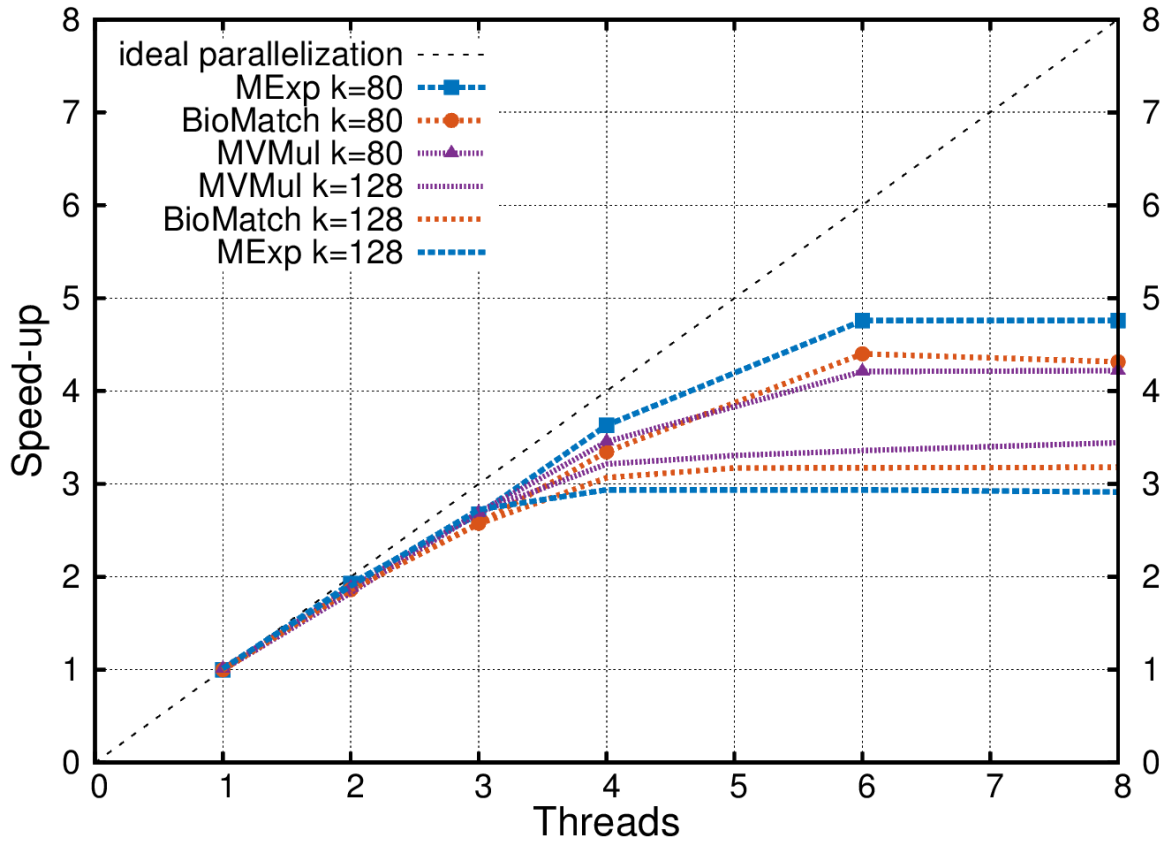


Security levels: 80bit and 128bit

Throughput: ~7M AND gates

Bandwidth req.: ~1 – 1.5 Gbit / core

Results - CGP in the Online Setting



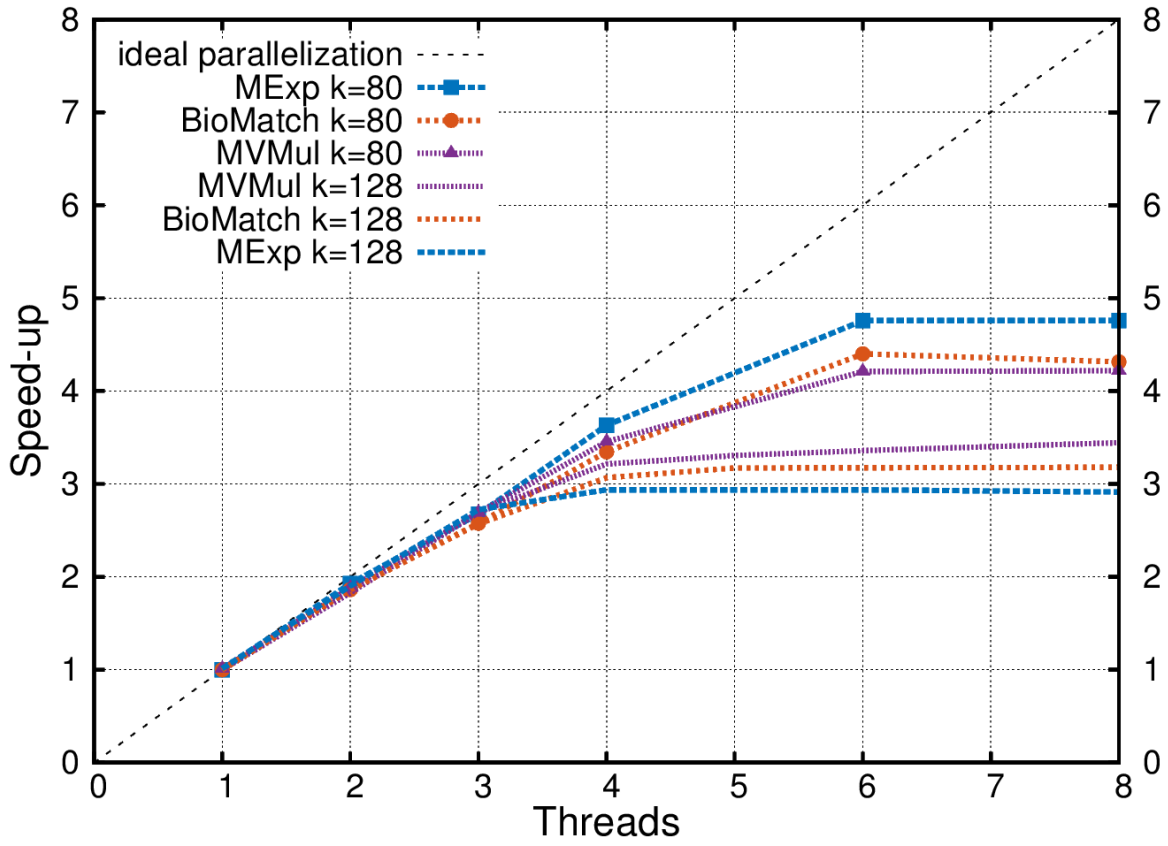
Security levels: 80bit and 128bit

Throughput: ~7M AND gates

Bandwidth req.: ~1 – 1.5 Gbit / core

Can we overcome the boundary?

Results - CGP in the Online Setting



Security levels: 80bit and 128bit

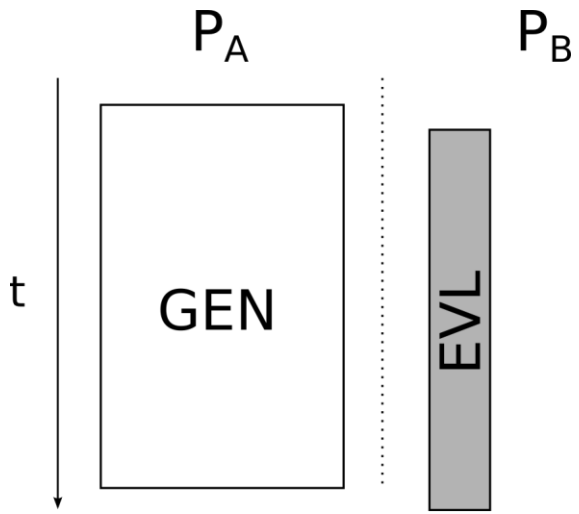
Throughput: ~7M AND gates

Bandwidth req.: ~1 – 1.5 Gbit / core

Can we overcome the boundary?

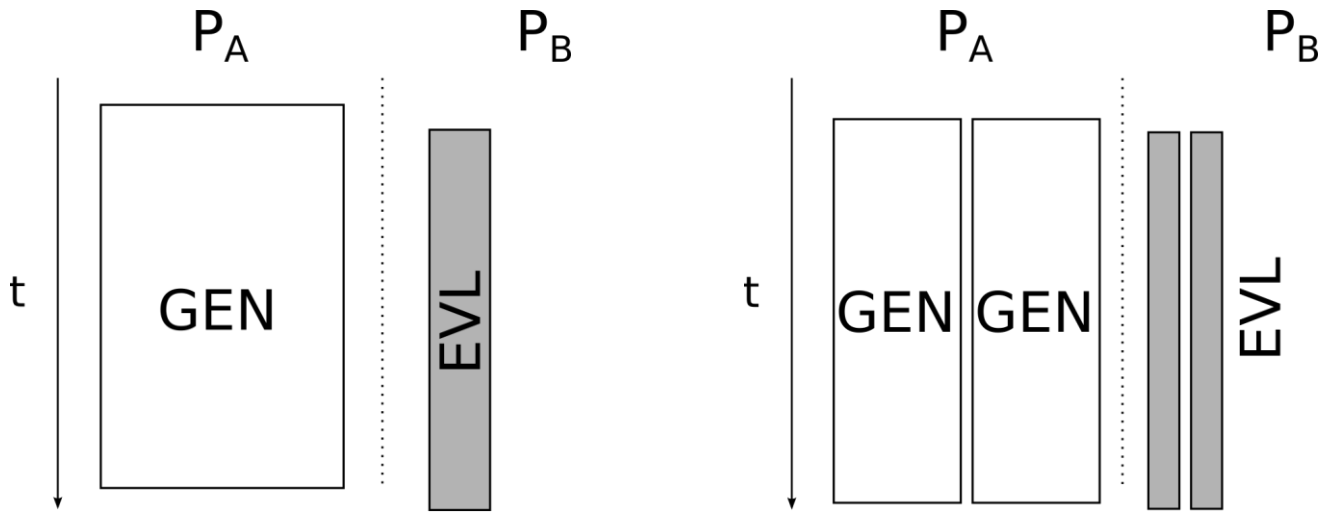
➔ Yes! See next slides

Inter-Party Parallelization (IPP)



a.) Yao’s Garbled Circuits

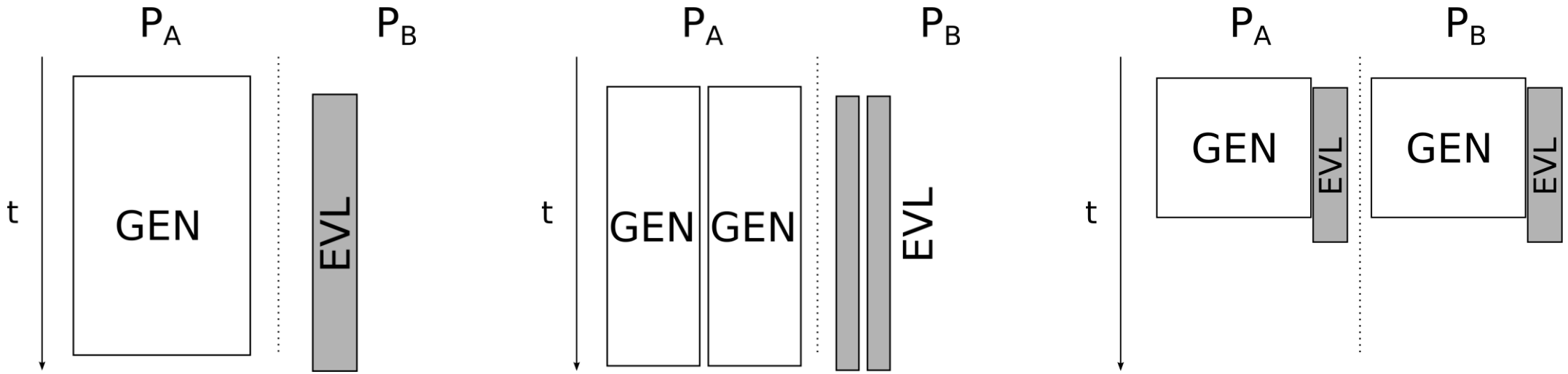
Inter-Party Parallelization (IPP)



a.) Yao's Garbled Circuits

b.) Coarse-grained decomposition

Inter-Party Parallelization (IPP)



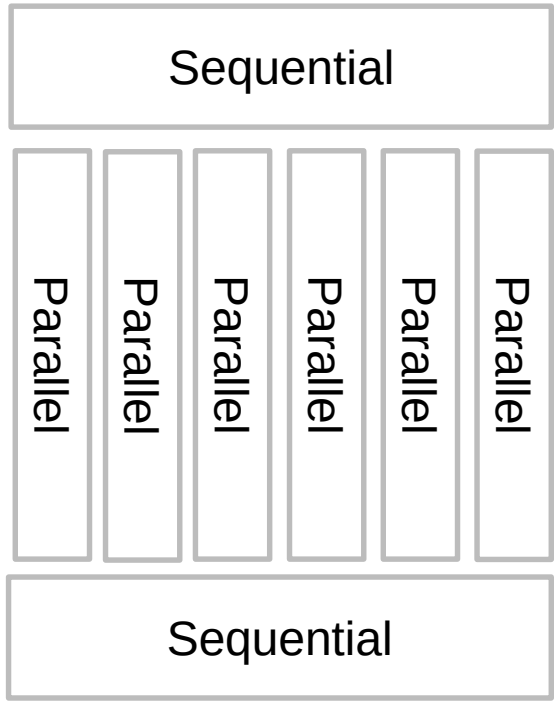
a.) Yao's Garbled Circuits

b.) Coarse-grained decomposition

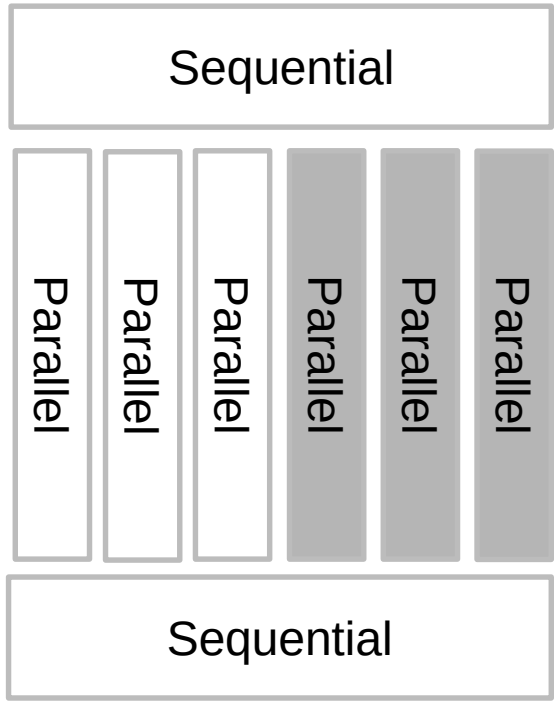
c.) Inter-party parallelization

Similar computation power → Speed-up of 1.6 / 1.33

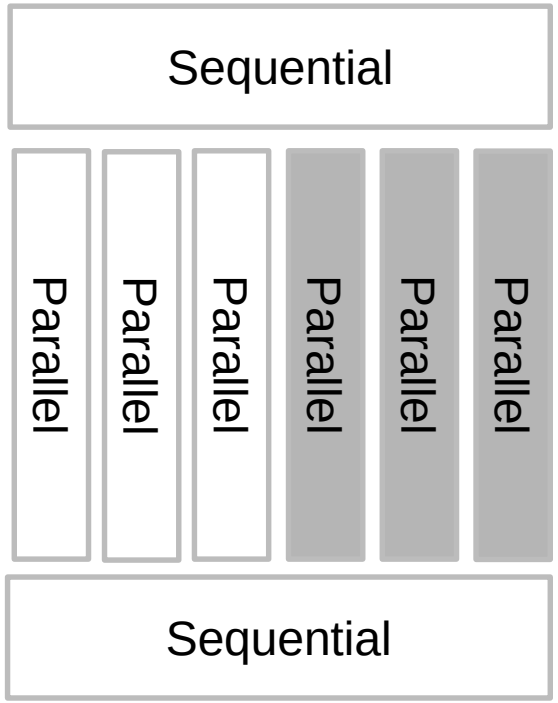
Inter-Party Parallelization (IPP) for Mixed Functionalities



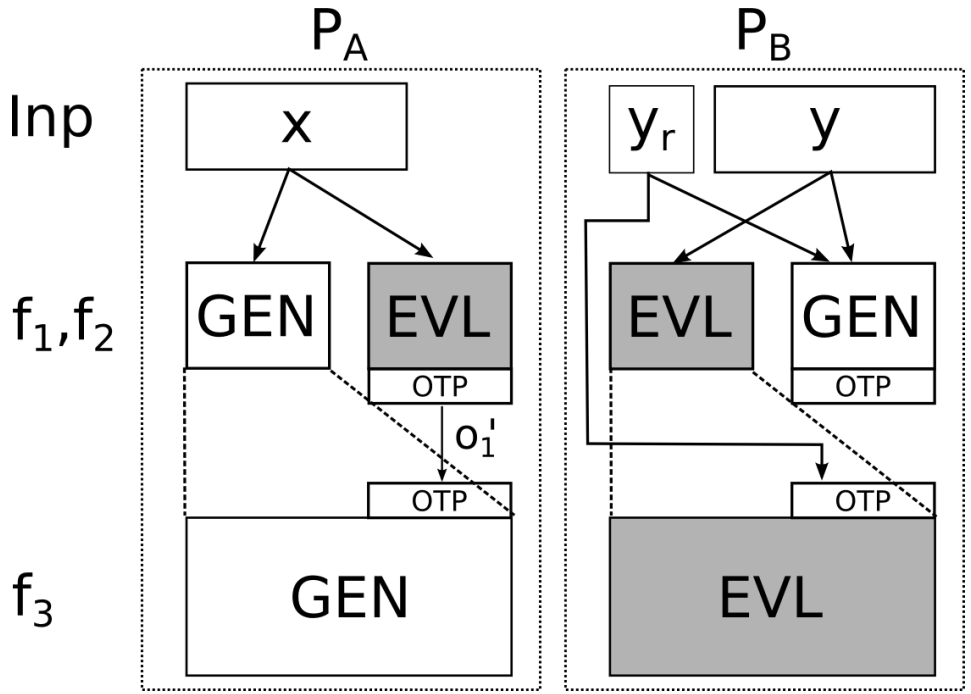
Inter-Party Parallelization (IPP) for Mixed Functionalities



Inter-Party Parallelization (IPP) for Mixed Functionalities



Transferring roles → secure state sharing



Results – Inter-Party Parallelization

TECHNISCHE UNIVERSITÄT DARMSTADT

High bandwidth environments (10Gbit)

Cores	BioMatch	MExp
1 CGP	1	1
1 IPP	1	**1.1**
2 CGP	1.8	1.9
2 IPP	1.8	**2.2**
4 CGP	3.2	3.6
4 IPP	3.1	**4.3**

Speed-up

SECENG SECURITY ENGINEERING

Results – Inter-Party Parallelization

TECHNISCHE UNIVERSITÄT DARMSTADT

High bandwidth environments (10Gbit)

Cores	BioMatch	MExp
1 CGP	1	1
1 IPP	1	**1.1**
2 CGP	1.8	1.9
2 IPP	1.8	**2.2**
4 CGP	3.2	3.6
4 IPP	3.1	**4.3**

Speed-up

Low bandwidth environments (100Mbit)

	BioMatch	MExp
raw	45.0±0.5s	24.1±0.2s
IPP	29.9±0.3s	16.1±0.1s
S	1.5	1.5

Time and speed-up

SECENG SECURITY ENGINEERING

Conclusion & Outlook

TECHNISCHE UNIVERSITÄT DARMSTADT

- Automatized toolchain for parallel circuits
- Yao’s Garbled Circuits profits from automatic parallelization
- Network boundaries have been reached

Outlook

- Open source releases of ParCC and UltraSFE
- IPP in other STC protocols

SECENG Security Engineering

Questions and Answers

TECHNISCHE UNIVERSITÄT DARMSTADT

Thank you!

SECENG SECURITY ENGINEERING

References

[ACE12] Amini, M., Creusillet, B., Even, S., Keryell, R., Goubier, O., Guelton, S., … Villalon, P. (2012). Par4All: From Convex Array Regions to Heterogeneous Computing. In Workshop on Polyhedral Compilation Techniques.

[BBL14] Barni, M., Bernaschi, M., Lazzeretti, R., Pignata, T., & Sabellico, A. (2014). Parallel Implementation of GC-Based MPC Protocols in the Semi-Honest Setting. In Data Privacy Management and Autonomous Spontaneous Security.

[BHK] Bellare, Mihir, et al. "Efficient garbling from a fixed-key blockcipher." Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013.

[FKN14] Frederiksen, T. K., Jakobsen, T. P., & Nielsen, J. B. (2014). Faster maliciously secure two-party computation using the GPU. In Security and Cryptography for Networks SCN (pp. 358–379).

[HEK11] Huang, Y., Evans, D., Katz, J., & Malka, L. (2011). Faster Secure Two-Party Computation Using Garbled Circuits. In USENIX Security Symposium (pp. 8–12).

[HMS13] Husted, N., Myers, S., Shelat, A., & Grubbs, P. (2013). GPU and CPU parallelization of honest-but-curious secure two-party computation. In Annual Computer Security Applications Conference ACSAC.

References

[HFK12] Holzer, A., Franz, M., Katzenbeisser, S., & Veith, H. (2012). Secure Two-Party Computations in ANSI C. In ACM Conference on Computer and Communications Security CCS.

[KS08] Kolesnikov, V., & Schneider, T. (2008). Improved garbled circuit: Free XOR gates and applications. In International Conference on Automata, Languages and Programming ICALP.

[MNP04] Malkhi, D., Nisan, N., Pinkas, B., & Sella, Y. (2004). Fairplay-Secure Two-Party Computation System. In USENIX Security Symposium.

[LIP09] Lindell, Y., & Pinkas, B. (2009). A proof of security of yao's protocol for two-party computation. Journal of Cryptology.

[NWI15] Nayak, K., Wang, X. S., Ioannidis, S., Weinsberg, U., Taft, N., & Shi, E. (2015). GraphSC: Parallel Secure Computation Made Easy. In IEEE Symposium on Security and Privacy S&P.

[KSS12] Kreuter, B., Shelat, A., & Shen, C. (2012). Billion-Gate Secure Computation with Malicious Adversaries. USENIX Security Symposium.

[ZRE15] Zahur, S., Rosulek, M., & Evans, D. (2015). Two halves make a whole. In Advances in Cryptology - EUROCRYPT.