Superfolding Garbled Circuits with Logic Synthesis

Farinaz Koushanfar
Rice University,
USA

Ahmad-Reza Sadeghi
Technische Universität Darmstadt,
Germany

Abstract

Designing practical secure 2-party computation based on Yao's garbled circuit (GC) has been subject of intensive research in the recent past with surprising progress on efficiency.

In this talk we present a novel methodology for automatic generation of highly compact and scalable Boolean circuits that further pushes down the limits. We achieve this by leveraging a new folded function description and powerful existing *real world* hardware synthesis methods and tools along with our created custom libraries.

For instance, we improve the results of the best known recently proposed automatic tool for GC generation by several orders of magnitude in storage while reducing the number of non-XOR gates more than 3x in average on benchmark circuits.

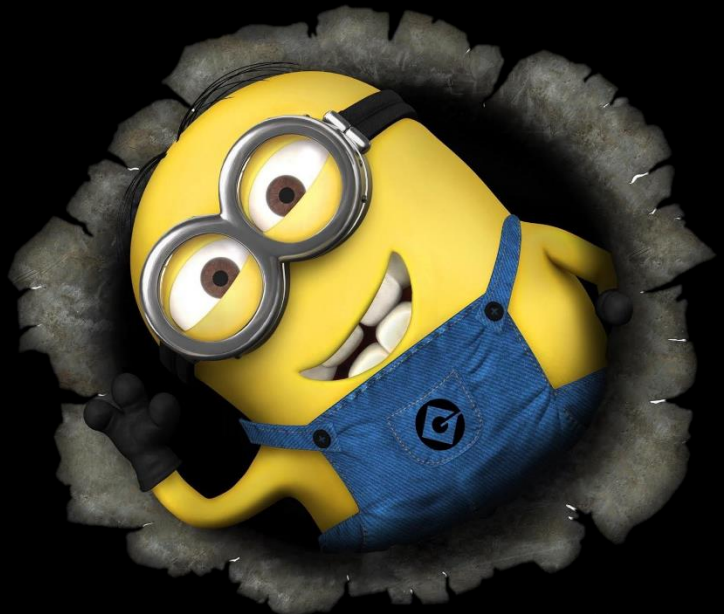
Furthermore, our approach enables us to implement functions that have not been reported before, such as RSA-8192 and SHA-3, which circuits require less than 8MB and 0.16MB of memory respectively on an Intel processor. We discuss how several exciting applications and known classical challenges can be practicably addressed be practicably addressed by our new findings.

Disclaimer

Copyright notice:

These slides were presented at the scientific conference „Real World Crypto“ and provide a short overview of our current research on secure computation. These slides and the materials they contain must only be used for non-commercial, educational and personal purposes. In particular, some of these slides contain copyrighted material „minions“ from the Chiquita company http://minionslovebananas.com/ to which we have referred wherever they used.

Motivation

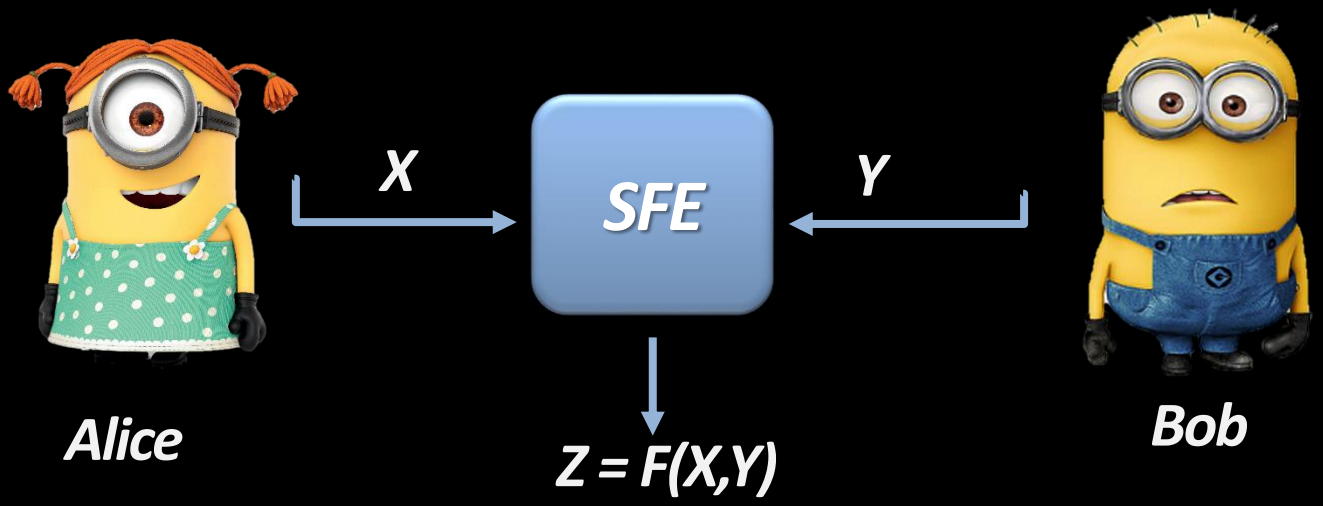


Copyright minions: The Chiquita company http://minionslovebananas.com/

Secure Function Evaluation

- **Goal: Compute function F() on private inputs X and Y**

Copyright minions: The Chiquita company http://minionslovebananas.com/



- **Focus:**
  - 2-party computation based on (Yao's) Garbled Circuits (GC) Protocol [Yao1986]
  - Garbling as a stand-alone primitive (JustGarble [BHKR, USENIX 2013])

Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

(private data) X

$X = (110)_2$

$F(X, Y)$

Y (private data)

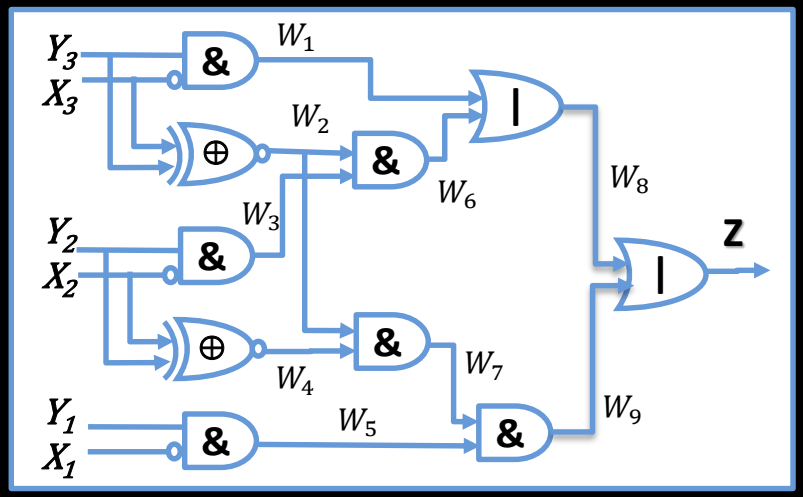
$Y = (100)_2$

Z

Bob (Circuit Evaluator)

Copyright minions: The Chiquita company http://minionslovebananas.com/

Generate Logic Circuit C



Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

Bob (Circuit Evaluator)

(private data) X

$X = (110)_2$

F(X ,Y)

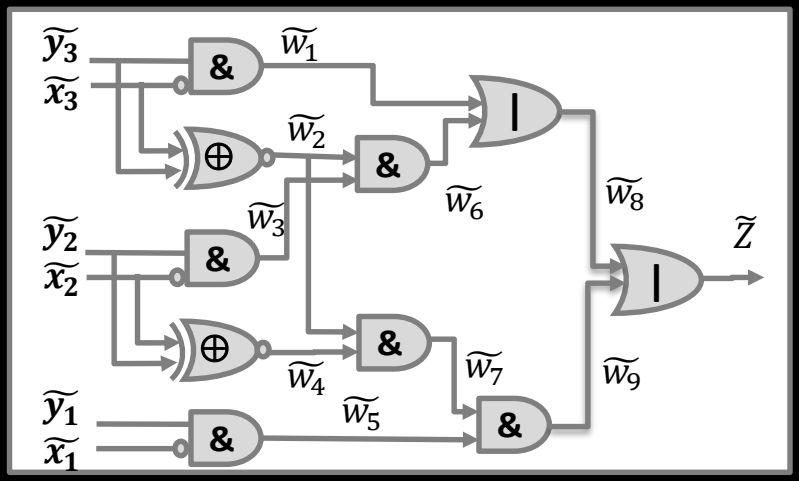
Y (private data)

$Y = (100)_2$

Z

Copyright minions: The Chiquita company http://minionslovebananas.com/

Generate Garbled Circuit $\widetilde{GC}$



Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

(private data) X → $X = (110)_2$

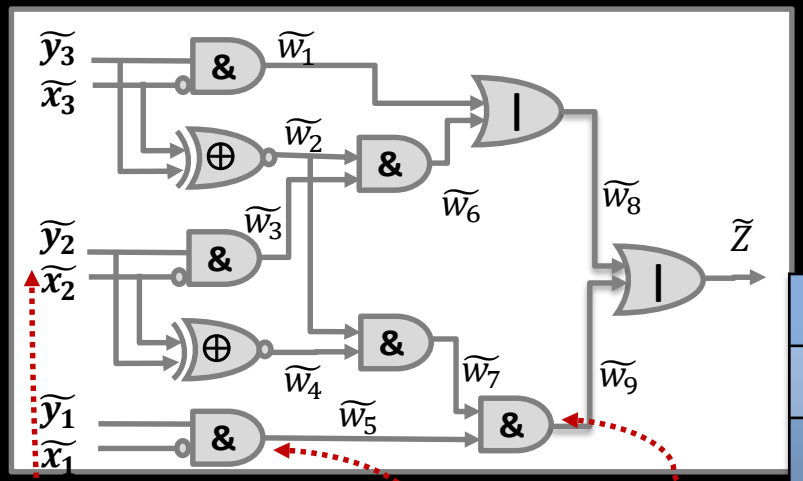
$F(X, Y)$

Y (private data) ← $Y = (100)_2$

Z

***Bob* (Circuit Evaluator)**

Generate Garbled Circuit $\widetilde{GC}$



Garbled Tables

w_7^0	w_5^0	w_9^0	$\mathrm{E}(w_7^0, w_5^0; w_9^0)$
w_7^0	w_5^1	w_9^0	$\mathrm{E}(w_7^0, w_5^1; w_9^0)$
w_7^1	w_5^0	w_9^0	$\mathrm{E}(w_7^1, w_5^0; w_9^0)$
w_7^1	w_5^1	w_9^1	$\mathrm{E}(w_7^1, w_5^1; w_9^1)$

x_1^0	y_1^0	w_5^0	$\mathrm{E}(x_1^0, y_1^0; w_5^0)$
x_1^0	y_1^1	w_5^0	$\mathrm{E}(x_1^0, y_1^1; w_5^0)$
x_1^1	y_1^0	w_5^0	$\mathrm{E}(x_1^1, y_1^0; w_5^0)$
x_1^1	y_1^1	w_5^1	$\mathrm{E}(x_1^1, y_1^1; w_5^1)$

Garbled inputs

$$\widetilde{x_i} = \begin{cases} x_i^0, & X_i = 0 \\ x_i^1, & X_i = 1 \end{cases}$$

Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

Bob (Circuit Evaluator)

(private data) X

$X = (110)_2$

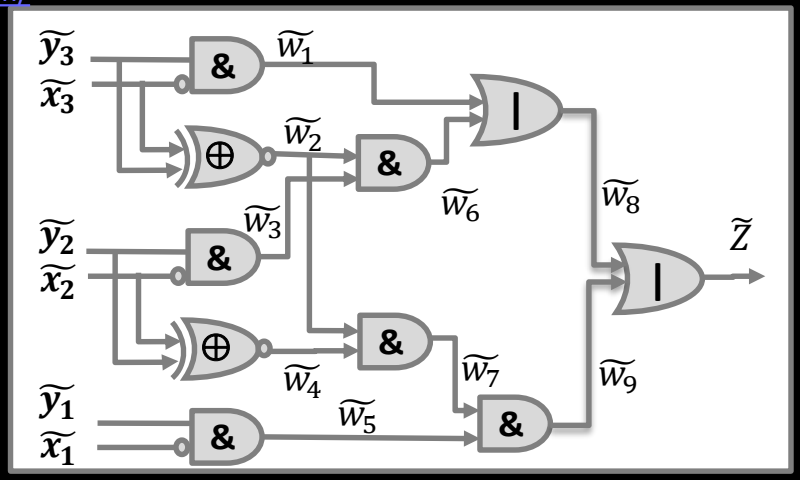
F(X ,Y)

Y (private data)

$Y = (100)_2$

Z

Copyright minions: The Chiquita company http://minionslovebananas.com/



x_1^0	y_1^0	w_5^0	$\mathrm{E}(x_1^0, y_1^0; w_5^0)$
x_1^0	y_1^1	w_5^0	$\mathrm{E}(x_1^0, y_1^1; w_5^0)$
x_1^1	y_1^0	w_5^0	$\mathrm{E}(x_1^1, y_1^0; w_5^0)$
x_1^1	y_1^1	w_5^1	$\mathrm{E}(x_1^1, y_1^1; w_5^1)$

w_7^0	w_5^0	w_9^0	$\mathrm{E}(w_7^0, w_5^0; w_9^0)$
w_5^1	w_9^0	$\mathrm{E}(w_7^0, w_5^1; w_9^0)$	
w_5^0	w_9^0	$\mathrm{E}(w_7^1, w_5^0; w_9^0)$	
w_5^1	w_9^1	$\mathrm{E}(w_7^1, w_5^1; w_9^1)$	

Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

(private data) X → $X = (110)_2$

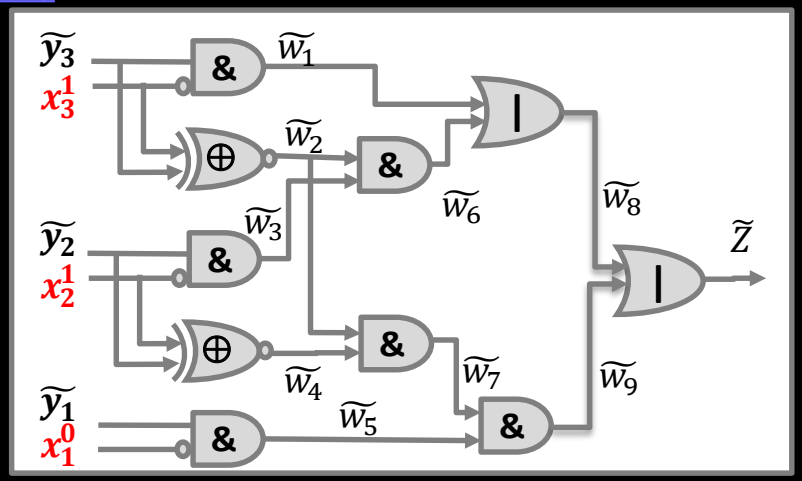
F(X ,Y)

Y (private data) ← $Y = (100)_2$

Z

Bob (Circuit Evaluator)

Copyright minions: The Chiquita company http://minionslovebananas.com/



$X_3 = 1$
$X_2 = 1$
$X_1 = 0$

$x_3^1 \quad x_2^1 \quad x_1^0$

Bob has no clue if x_i^0 corresponds to X_i= 0 or 1

x_1^0	y_1^0	w_5^0	$\mathrm{E}(x_1^0, y_1^0; w_5^0)$
x_1^0	y_1^1	w_5^0	$\mathrm{E}(x_1^0, y_1^1; w_5^0)$
x_1^1	y_1^0	w_5^0	$\mathrm{E}(x_1^1, y_1^0; w_5^0)$
x_1^1	y_1^1	w_5^1	$\mathrm{E}(x_1^1, y_1^1; w_5^1)$

w_7^0	w_5^0	w_9^0	$\mathrm{E}(w_7^0, w_5^0; w_9^0)$
	w_5^1	w_9^0	$\mathrm{E}(w_7^0, w_5^1; w_9^0)$
	w_5^0	w_9^0	$\mathrm{E}(w_7^1, w_5^0; w_9^0)$
	w_5^1	w_9^1	$\mathrm{E}(w_7^1, w_5^1; w_9^1)$

Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

Bob (Circuit Evaluator)

(private data) X

$X = (110)_2$

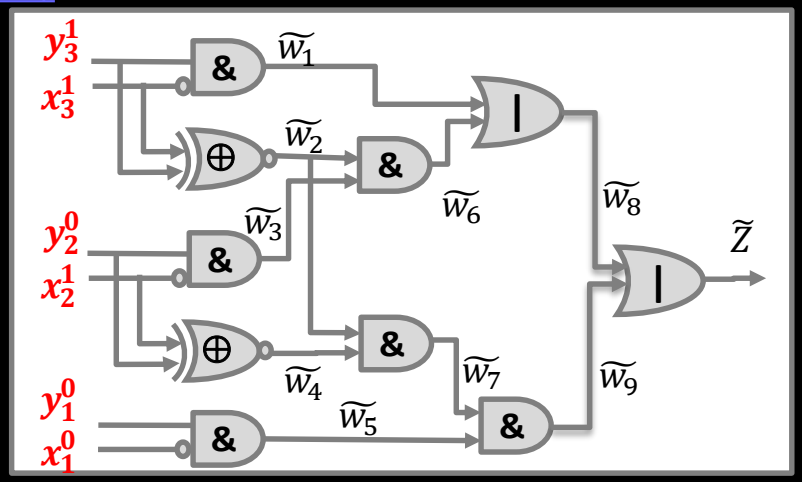
F(X ,Y)

Y (private data)

$Y = (100)_2$

Z

Copyright minions: The Chiquita company http://minionslovebananas.com/



Alice has no clue if Bob picked y_i^0 or y_i^1 up

y_1^0

y_1^0

y_1^1

OT

$Y_1 = 0$

x_1^0	y_1^0	w_5^0	$\mathrm{E}(x_1^0, y_1^0; w_5^0)$
x_1^0	y_1^1	w_5^0	$\mathrm{E}(x_1^0, y_1^1; w_5^0)$
x_1^1	y_1^0	w_5^0	$\mathrm{E}(x_1^1, y_1^0; w_5^0)$
x_1^1	y_1^1	w_5^1	$\mathrm{E}(x_1^1, y_1^1; w_5^1)$

w_7^0	w_5^0	w_9^0	$\mathrm{E}(w_7^0, w_5^0; w_9^0)$
	w_5^1	w_9^0	$\mathrm{E}(w_7^0, w_5^1; w_9^0)$
	w_5^0	w_9^0	$\mathrm{E}(w_7^1, w_5^0; w_9^0)$
	w_5^1	w_9^1	$\mathrm{E}(w_7^1, w_5^1; w_9^1)$

Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

(private data) X → $X = (110)_2$

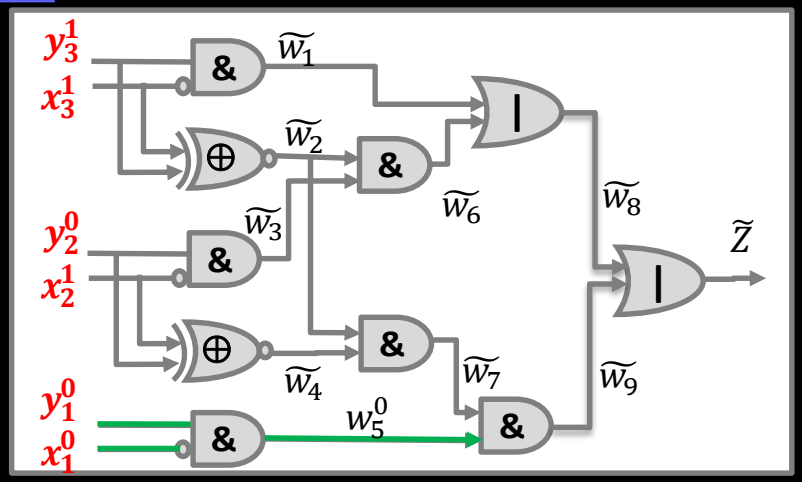
$F(X, Y)$

Y (private data) ← $Y = (100)_2$

Z

***Bob* (Circuit Evaluator)**

Copyright minions: The Chiquita company http://minionslovebananas.com/



				w_7^0	w_5^0	w_9^0	$\mathrm{E}(w_7^0, w_5^0; w_9^0)$
x_1^0	y_1^0	w_5^0	$\mathrm{E}(x_1^0, y_1^0; w_5^0)$		w_5^1	w_9^0	$\mathrm{E}(w_7^0, w_5^1; w_9^0)$
x_1^0	y_1^1	w_5^0	$\mathrm{E}(x_1^0, y_1^1; w_5^0)$	w_5^0	w_9^0	$\mathrm{E}(w_7^1, w_5^0; w_9^0)$	
x_1^1	y_1^0	w_5^0	$\mathrm{E}(x_1^1, y_1^0; w_5^0)$	w_5^1	w_9^1	$\mathrm{E}(w_7^1, w_5^1; w_9^1)$	
x_1^1	y_1^1	w_5^1	$\mathrm{E}(x_1^1, y_1^1; w_5^1)$				

Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

(private data) X

$X = (110)_2$

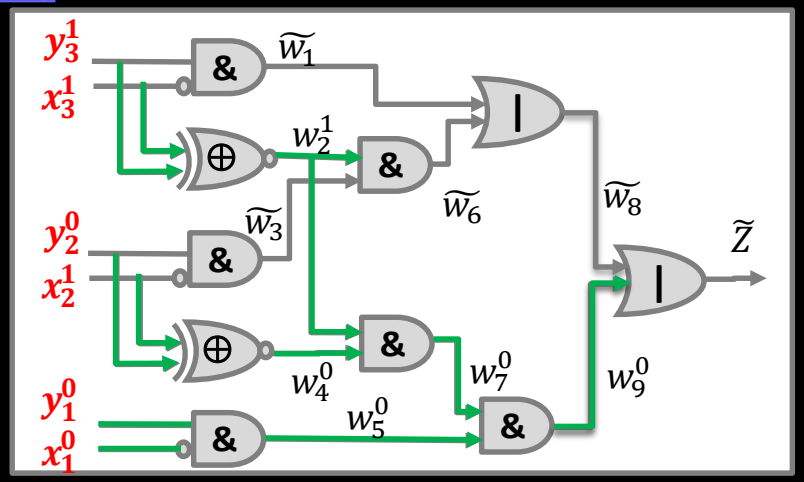
F(X ,Y)

Z

Y (private data)

$Y = (100)_2$

***Bob* (Circuit Evaluator)**



x_1^0	y_1^0	w_5^0	$\mathrm{E}(x_1^0, y_1^0; w_5^0)$
x_1^0	y_1^1	w_5^0	$\mathrm{E}(x_1^0, y_1^1; w_5^0)$
x_1^1	y_1^0	w_5^0	$\mathrm{E}(x_1^1, y_1^0; w_5^0)$
x_1^1	y_1^1	w_5^1	$\mathrm{E}(x_1^1, y_1^1; w_5^1)$

w_7^0	w_5^0	w_9^0	$\mathrm{E}(w_7^0, w_5^0; w_9^0)$
w_5^1	w_9^0	$\mathrm{E}(w_7^0, w_5^1; w_9^0)$	
w_5^0	w_9^0	$\mathrm{E}(w_7^1, w_5^0; w_9^0)$	
w_5^1	w_9^1	$\mathrm{E}(w_7^1, w_5^1; w_9^1)$	

Yao's Protocol: Garbled Circuits (GC)

Alice (Circuit Generator)

e.g. F(X < Y) Millionaire Problem

(private data) X

$X = (110)_2$

F(X ,Y)

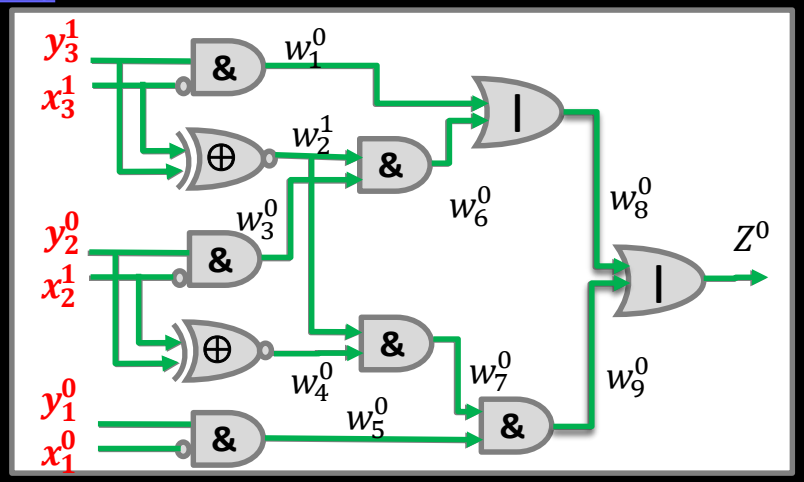
Z

Y (private data)

$Y = (100)_2$

***Bob* (Circuit Evaluator)**

Copyright minions: The Chiquita company http://minionslovebananas.com/



x_1^0	y_1^0	w_5^0	$\mathrm{E}(x_1^0, y_1^0; w_5^0)$
x_1^0	y_1^1	w_5^0	$\mathrm{E}(x_1^0, y_1^1; w_5^0)$
x_1^1	y_1^0	w_5^0	$\mathrm{E}(x_1^1, y_1^0; w_5^0)$
x_1^1	y_1^1	w_5^1	$\mathrm{E}(x_1^1, y_1^1; w_5^1)$

w_7^0	w_5^0	w_9^0	$\mathrm{E}(w_7^0, w_5^0; w_9^0)$
w_5^1	w_9^0	$\mathrm{E}(w_7^0, w_5^1; w_9^0)$	
w_5^0	w_9^0	$\mathrm{E}(w_7^1, w_5^0; w_9^0)$	
w_5^1	w_9^1	$\mathrm{E}(w_7^1, w_5^1; w_9^1)$	

Can Yao‘s GC protocol be practical?

Copyright minions: The Chiquita company http://minionslovebananas.com/

Adversary Model

- **Honest but curious**
- **Malicious**
- **Covert**

Copyright minions: The Chiquita company http://minionslovebananas.com/

Garbled Circuit Optimizations

- **Row-reduction**
  - Reduce size of garbled truth table for non-XOR gates by 25%
    [Naor et al., ACMEC 1999]
- **Free-XOR**
  - No garbled truth table for XOR gate needed
    [Kolesinkov et al., ICALP 2008], [Kolesnikov et al. Crypto 2014]
- **Garbling with fixed-key block cipher**
  - No additional keys for gate output (unique tweak T per gate)
    [Bellare et al., S&P 2013]
- **Execution optimization**
  - Fast table lookups, pipelining
    [Järvinen et al. CHES 2010], [Haung, et al. USENIX 2011]

Main Approach for GC

- **Compiler-based**
  - Compile high-level description of functionality to optimized circuits
  - e.g., FairPlay, TASTY, PFC, etc.
- **Library-based**
  - Custom-libraries with special fucntions for emitting Boolean circuits, built-in boolean circuits
  - e.g., FastGC, VMCrypt, etc.
- **Hardware-assisted**
  - GPU based, AES-NI

1986 — Yao GC Protocol [Yao, FOCS]

SELECTED

2004 — FairPlay Compiler-based [MNPS, USENIX]

2009 — 2-Party SFE is practical Compiler-based [PSNW, AISACCS]

2010 — TASTY Compiler-based [HKSSW, CCS]; GC for One-Time Prog. HW-based [CHES]

2011 — FastGC Library-based [HEKM, USENIX]; VMCrypt Library-based [Malka, CCS]

2012 — Bill. Gate 2-Party SFE Compiler-based [KSS, USENIX]; 2-Party SFE in ANSI C Compiler-based [HFAK, CCS]

2013 — Portable Circuit Format (PCF) Compiler-based [KSMB, USENIX]; AES-NI JustGarble Hardware-based [BHKR, USENIX]; Circuit Structures Library-based [ZE, S&P]; 2-Party SFE on GPU HW-based [HMSG, ACSAC]; 2-Party SFE Appl. on GPU HW-based [PL, eprint]; 2-Party SFE on GPU HW-based [FN, ACNS]

2014 — TinyGarble Sequential Logic Synthesis (this work)

Problems

- **Manual circuit-level optimizations**
  - No global optimization
- **Lack of practical utility**
  - Users cannot comprehend the final circuit organization and therefore cannot apply finer circuit optimizations
- **Poor Scalability**
  - Memory exhaustion
  - Circuit generation/evaluation time may exceed real-time constraints
  - Loops unrolling and subroutines inlining
  - High-level programming abstraction: circuits not compact and well optimized
- **Combinational logic**
  - Prevents synthesis of large circuits (e.g., SHA3, RSA)
- **Only moderate size circuits are handled**
  - Some circuit sizes not feasible for embedded devices

Our Approach:

Generating super compact and scalable circuits by

- sequential logic description for functionality
- introducing new transforms/libraries to enable adapting classic HW synthesis techniques

Copyright minions: The Chiquita company http://minionslovebananas.com/

Our Contributions

- Sequential logic description for generating functions
- Enable adaption of established HDL synthesis techniques/tools
- Creation of new custom synthesis libraries optimized for GC
  - used with standard HDL synthesis tools for minimizing non-XOR gates
- New degree of freedom: designer can control degree of circuit folding
- Improving best reported results by several orders of magnitude
- Enabling implementation of circuits never reported before
  - SHA3, RSA-8192

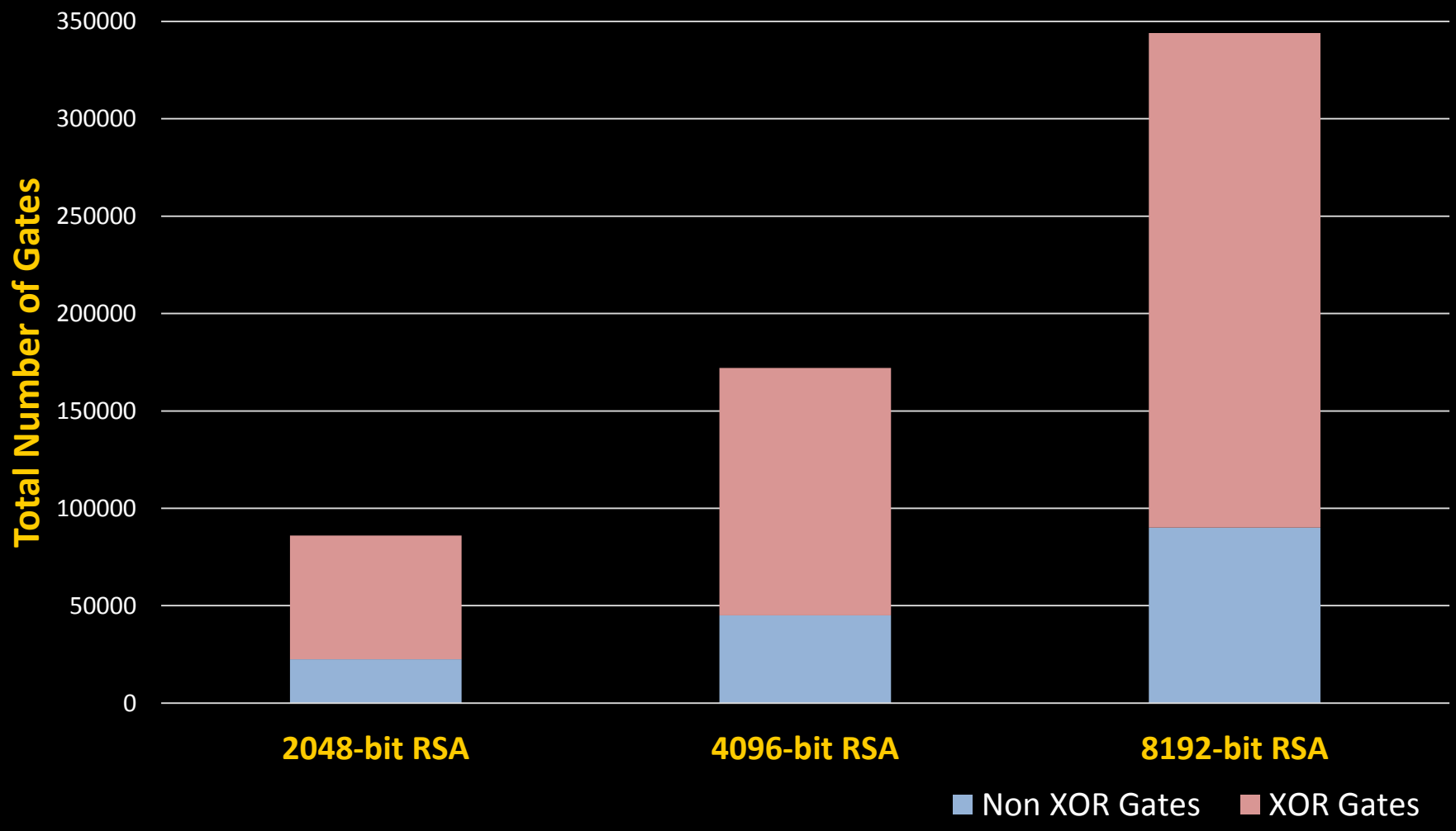
Evaluation

Copyright minions: The Chiquita company http://minionslovebananas.com/

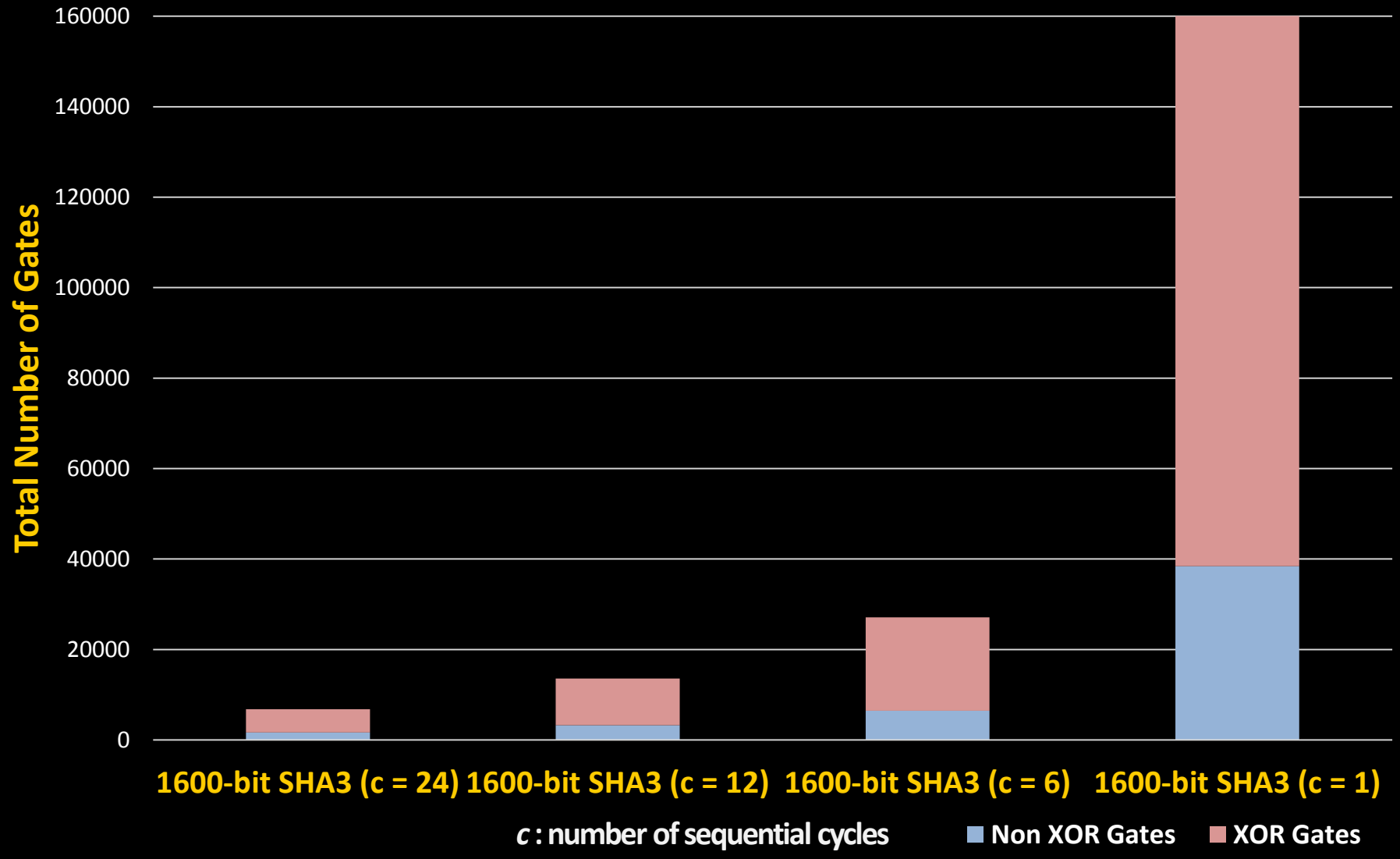
Benchmark Functions

- **Combinational circuit implementations**
- **Sequential circuit implementations**
  - Compared with equivalent combinational circuit
  - Compared with sequential circuit implementations of different circuit folding
  - Compared with reference circuit implementations in other works
- **Report functions not implementable earlier**
  - e.g. SHA3 function

Circuit Size: Evaluation of RSA



Circuit Size: Evaluation of SHA3



Conclusion and Ongoing Work

- **GC Boolean circuit generation viewed as atypical logic synthesis**
- **Many exciting applications enabled**
  - Scalable mapping for many relevant privacy-enhancing primitives
- **Integration to mobile devices**
    - Super compact circuit sizes requires low amount of storage
    - Using Trusted Execution Environment on Smartphones
- **Scalable Semi Private SFE**
  - Circuit size independent of the number of instructions

You need more details? Talk to us!

Farinaz Koushanfar
Rice University, USA
farinaz@rice.edu

Ahmad-Reza Sadeghi
Technische Universität Darmstadt, Germany
ahmad.sadeghi@trust.cased.de