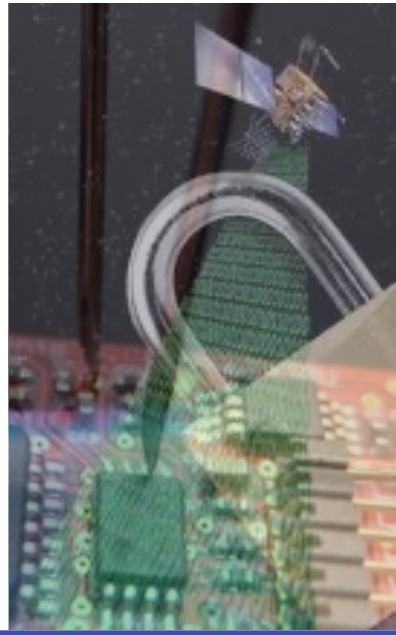
Garbled Circuits for Leakage-Resilience: Hardware Implementation and Evaluation of One-Time Programs

Kimmo Järvinen
Aalto University, Finland

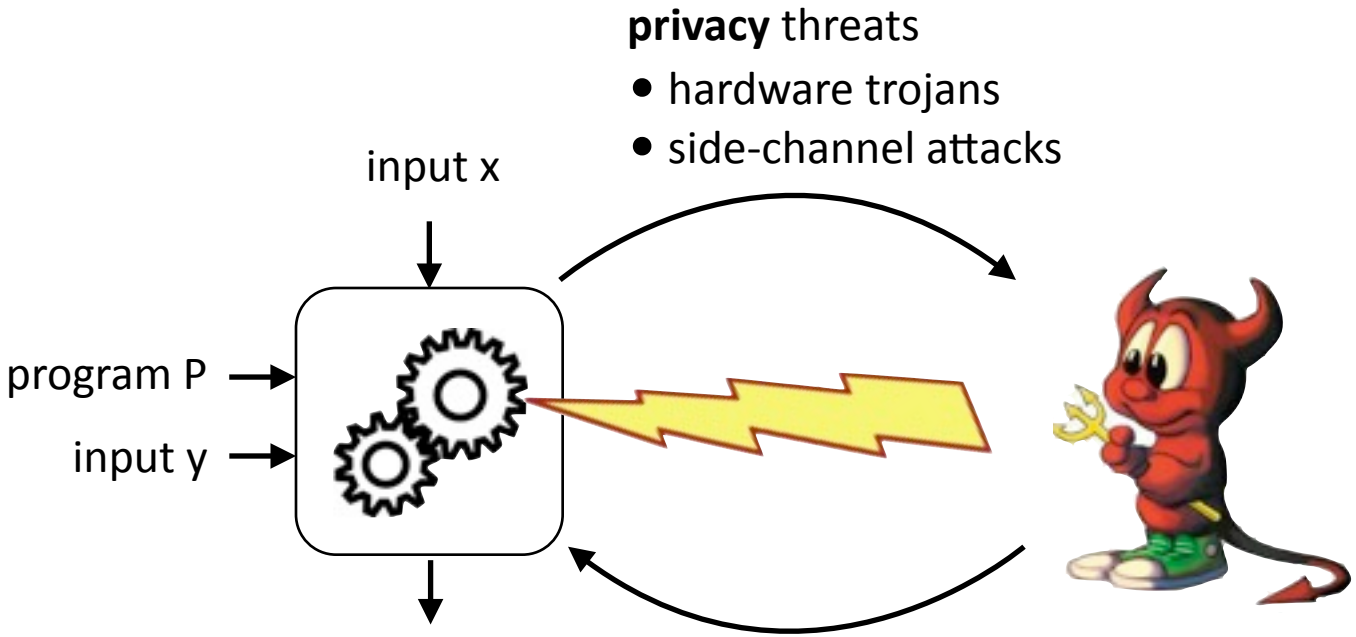
Vladimir Kolesnikov
Alcatel-Lucent Bell Laboratories, USA

Ahmad-Reza Sadeghi
System Security Lab,
Ruhr-University Bochum, Germany

Thomas Schneider
System Security Lab,
Ruhr-University Bochum, Germany



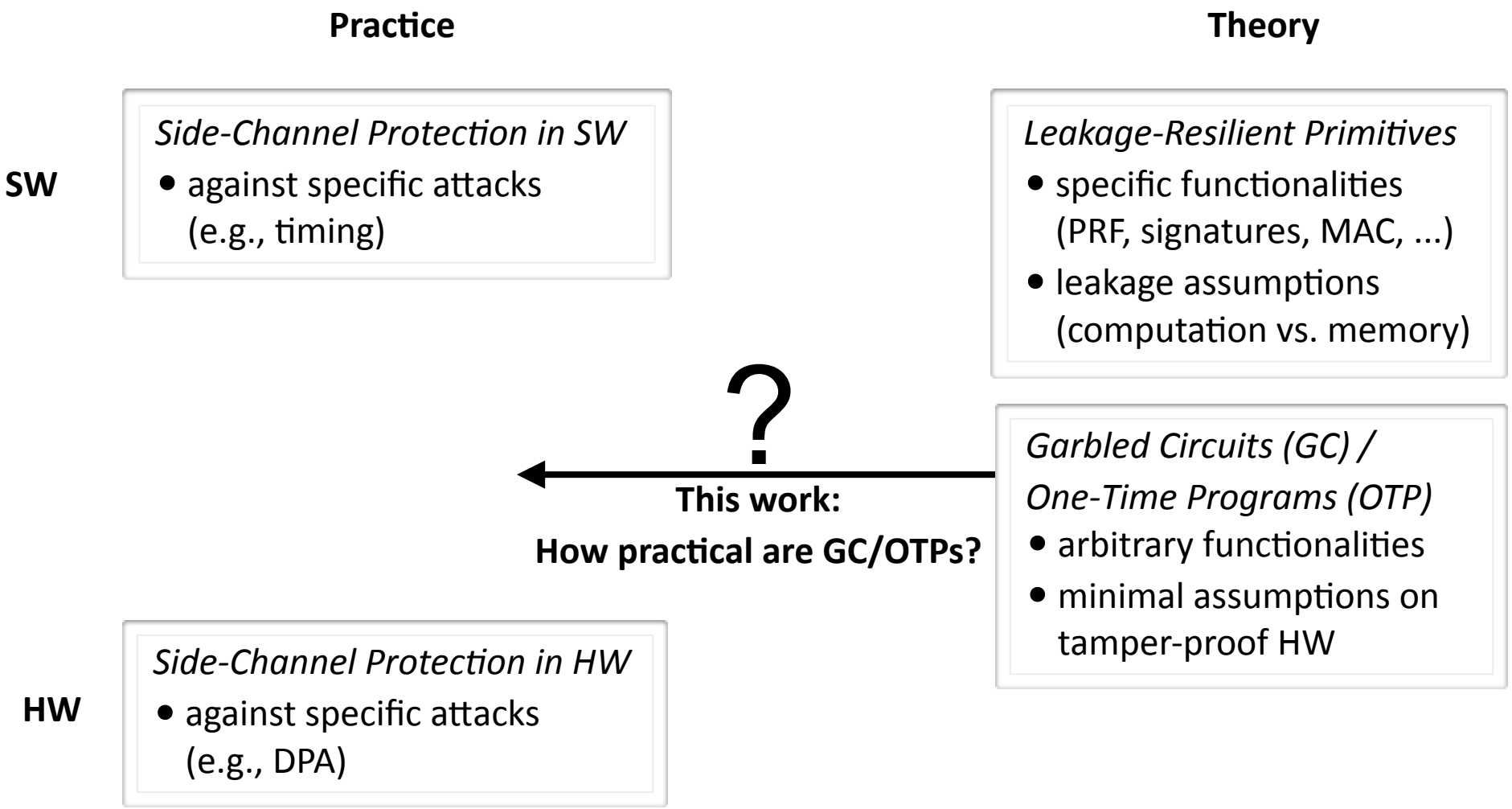
Scenario: Compute in Hostile Environment



Goal: Guarantee <u>privacy & correctness</u>

in the presence of <u>malicious/attacked HW</u> !

Methods for Leakage-Resilient Computation



Our Goal & Contribution

Evaluate practicality of OTP:

- Improved GC/OTP for leakage-resilience
  - Adapt OTPs for practice
  - Generic architecture: GCs for leakage-resilience

- First GC/OTP evaluation in Hardware
  - HW architectures
  - Implementation on FPGA: GC/OTP of AES
    - 10x faster than existing SW implementations
    - slower than unprotected / DPA protected implementations

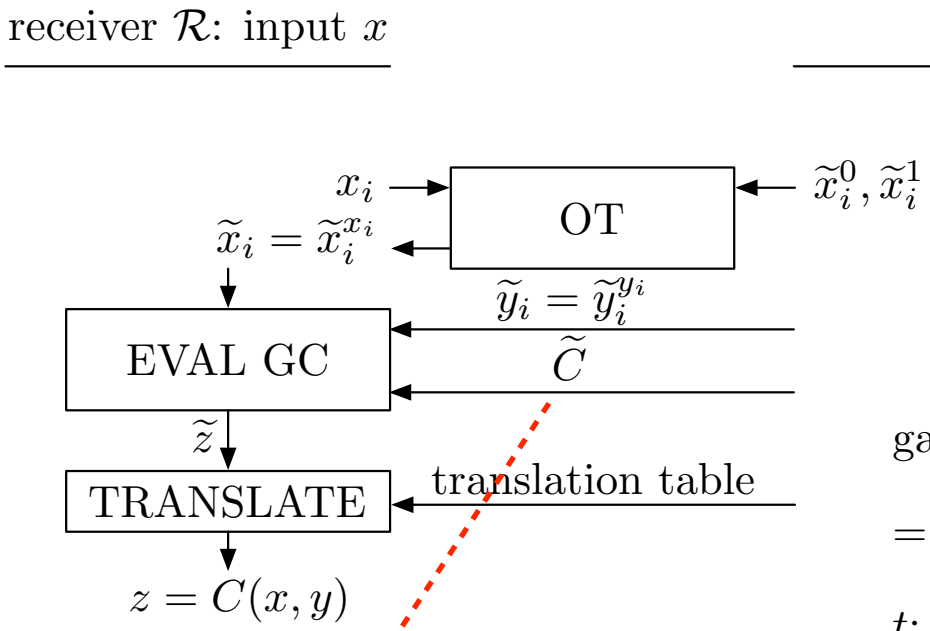
Related Work

GC/OTP for Leakage-Resilience

Related Work	Interaction	Attacks	Security
[Yao FOCS'86] "**Garbled Circuits (GC)**"	interactive	passive	computational
[Gunupudi,Tate FC'08] "**Mobile Agents**"	**non-interactive**	passive	computational
[Goldwasser,Kalai,Rothblum CRYPTO'08] "**One-Time Programs (OTP)**"	**non-interactive**	**active**	computational
[Goyal,Ishai,Sahai,Venkatesan,Wadia TCC'10] "**Non-Interactive Secure Computation**"	**non-interactive**	**active**	**unconditional**

This work: computational security

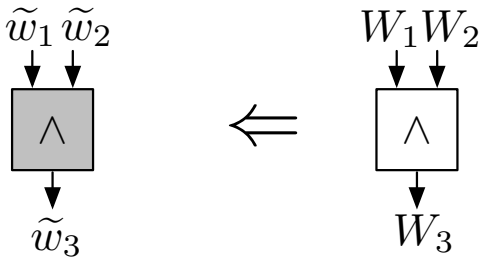
Garbled Circuits (GC) [Yao FOCS'86]



GC cannot be reused !

Improved GC constructions:

- [Naor,Pinkas,Sumner ACM EC'99]: remove 1 entry from garbled table
- [Kolesnikov,Schneider ICALP'08]: free XOR gates



garbled values $\widetilde{w}_i \in \{0,1\}^t$ gate G_i

$$= \begin{cases} \widetilde{w}_i^0 & \text{for plain value 0} \\ \widetilde{w}_i^1 & \text{for plain value 1} \end{cases}$$

t: symmetric security parameter (e.g., $t = 128$)

garbled table $\widetilde{T}_i$

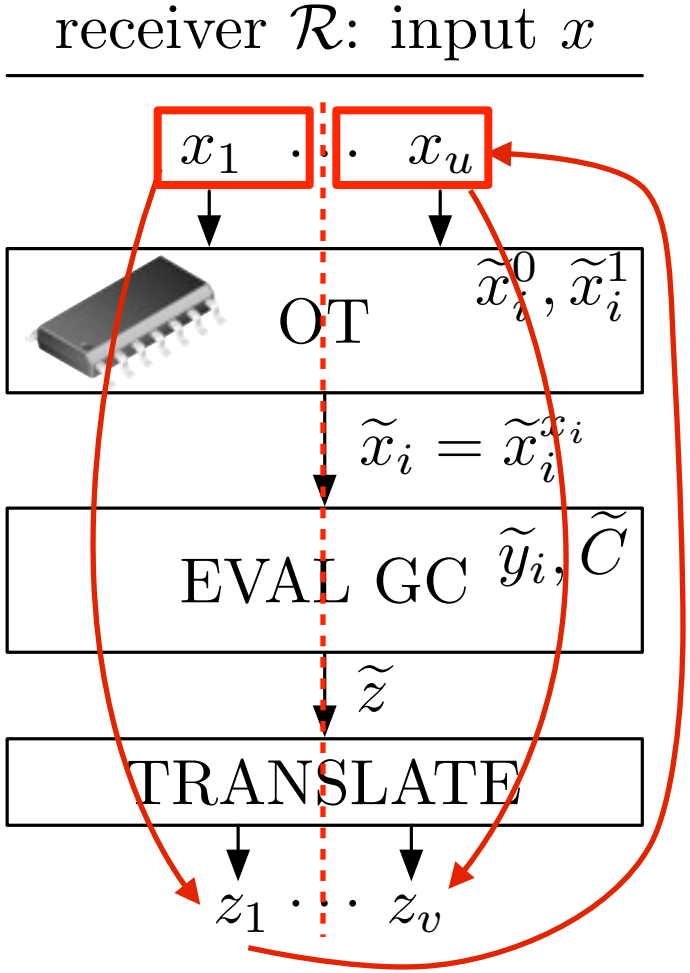
$$\begin{bmatrix} E_{\widetilde{w}_1^0, \widetilde{w}_2^0}(\widetilde{w}_3^0) \\ E_{\widetilde{w}_1^0, \widetilde{w}_2^1}(\widetilde{w}_3^0) \\ E_{\widetilde{w}_1^1, \widetilde{w}_2^0}(\widetilde{w}_3^0) \\ E_{\widetilde{w}_1^1, \widetilde{w}_2^1}(\widetilde{w}_3^1) \end{bmatrix}$$

E: semantically secure symmetric encryption (e.g., using SHA-256)

Non-Interactive Oblivious Transfer (OT)

[Gunupudi,Tate FC'08]

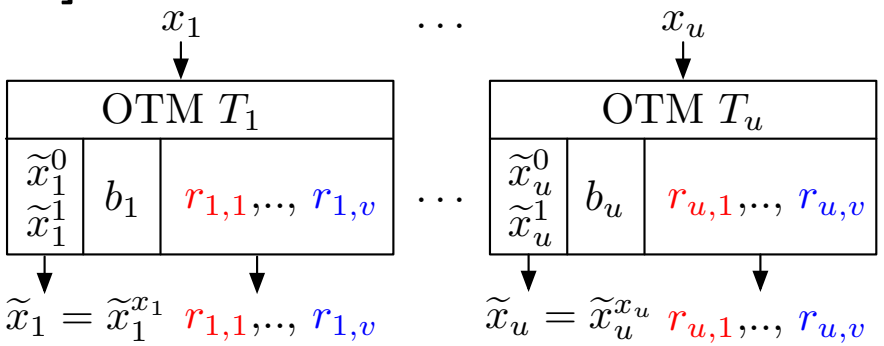
- implement non-interactive OT with trusted hardware
- use Trusted Platform Module (TPM)
- secure only against passive attacks as **active adversary can query adaptively**



One-Time Programs (OTP)

[Goldwasser,Kalai,Rothblum CRYPTO'08]

- **Minimal tamper-proof HW:**
  One-Time Memory (OTM):
  on input x_i, OTM T_i:
  - verifies tamper-proof bit b_i is unset
  - sets b_i, outputs $\widetilde{x}_i^{x_i}$
  - never touches or deletes $\widetilde{x}_i^{1-x_i}$



- **Prevent active attacks by receiver R**
  - R can decrypt output only after he has queried all OTMs
  - proposed technique: secret-sharing + one-time pad
    use $r_1 = r_{1,1} \oplus \cdots \oplus r_{u,1}$ to mask output bit $z_1, \ldots,$
    use $r_v = r_{1,v} \oplus \cdots \oplus r_{u,v}$ to mask output bit z_v
  - **Problem: OTMs depend on number of outputs v**

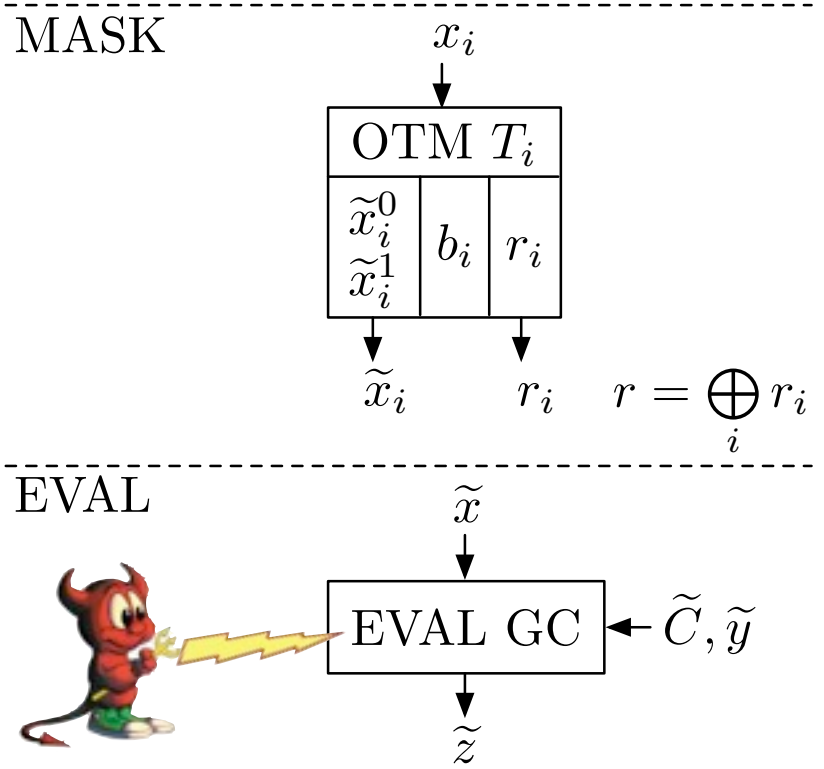
Theoretical Contribution

Improved GC/OTP for leakage-resilience

Our Improved One-Time Programs

- **Make OTMs independent of number of outputs**

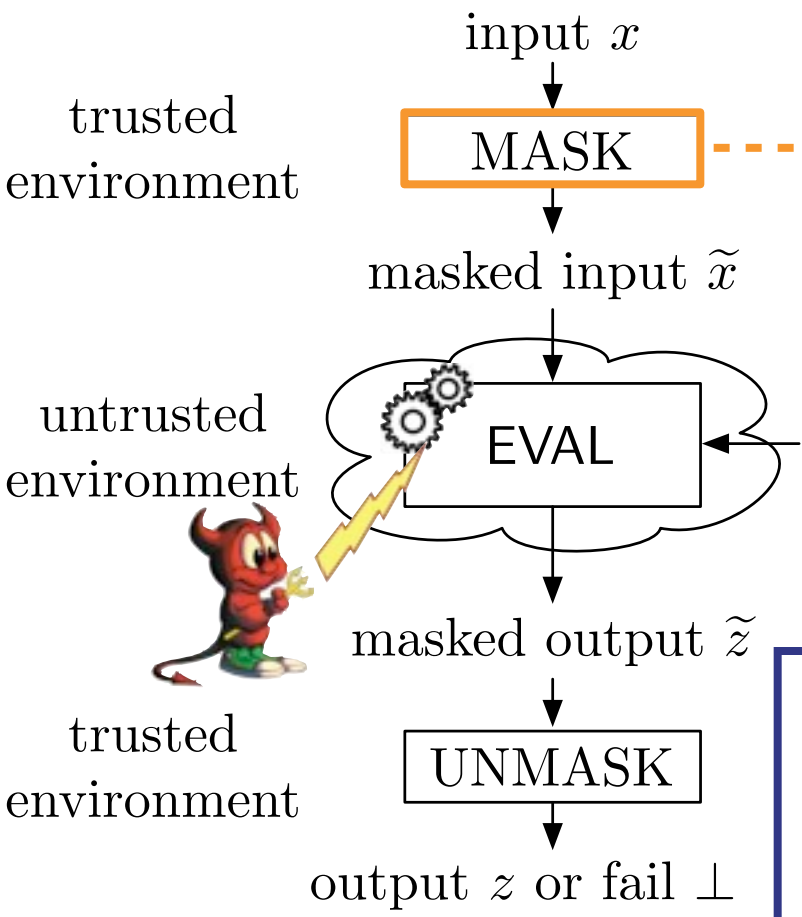
  OTM T_i releases **single key** $r_i \in \{0,1\}^t$
  t: symmetric security parameter



- **Output Verifiability**

  H: Random Oracle (e.g., SHA-256)

Architecture: GCs for Leakage-Resilience

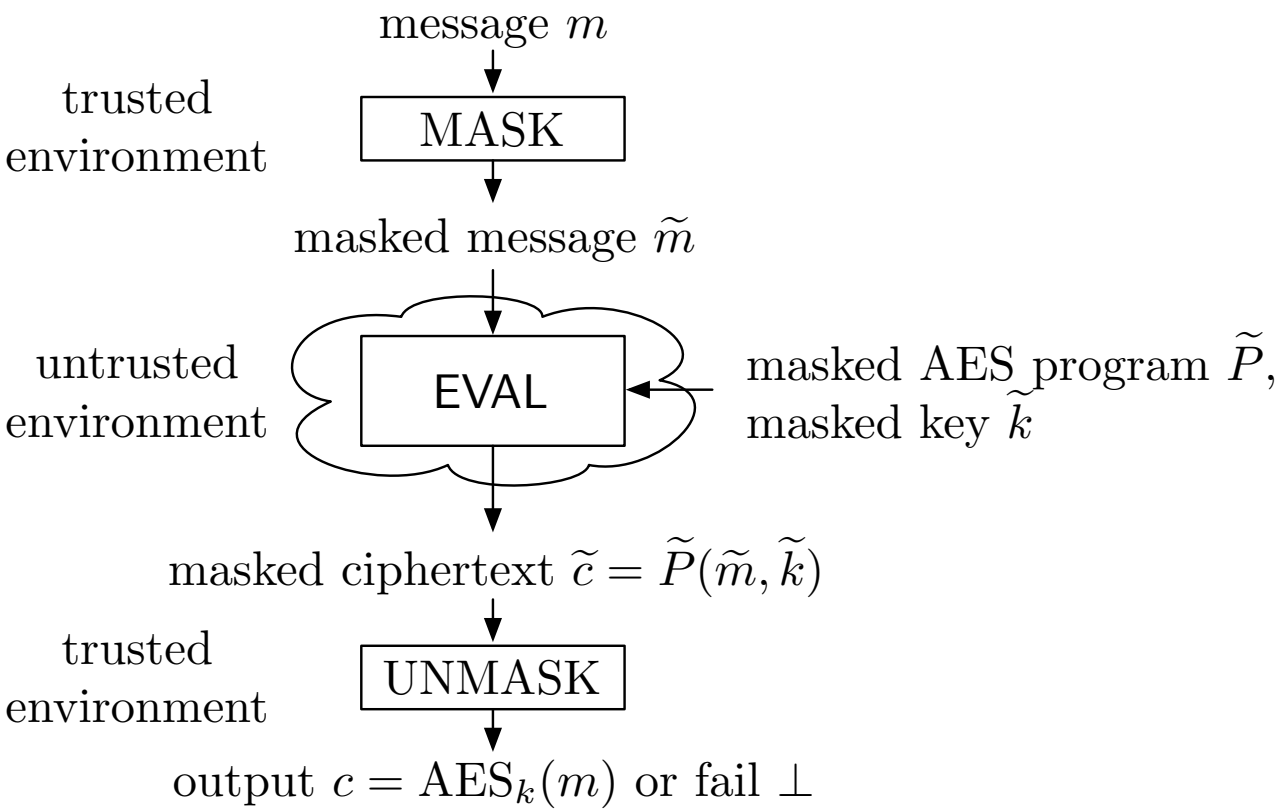


Possible implementations:
- interactive OT
- non-interactive OT with TPM
- OTMs
- smartcard

Possible implementations:
- created & sent by S
- local on-the-fly generation with smartcard [Järvinen,Kolesnikov,Sadeghi,Schneider FC'10]

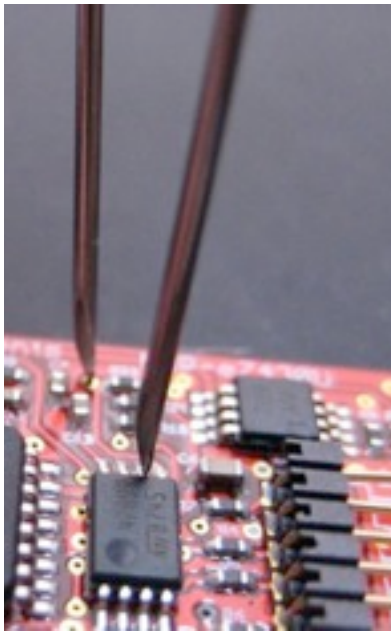
Use case: OTP for leakage-resilient AES

- AES is relatively complex function
- Allows comparison with previous works
- Application: encrypt message m with key k in untrusted environment

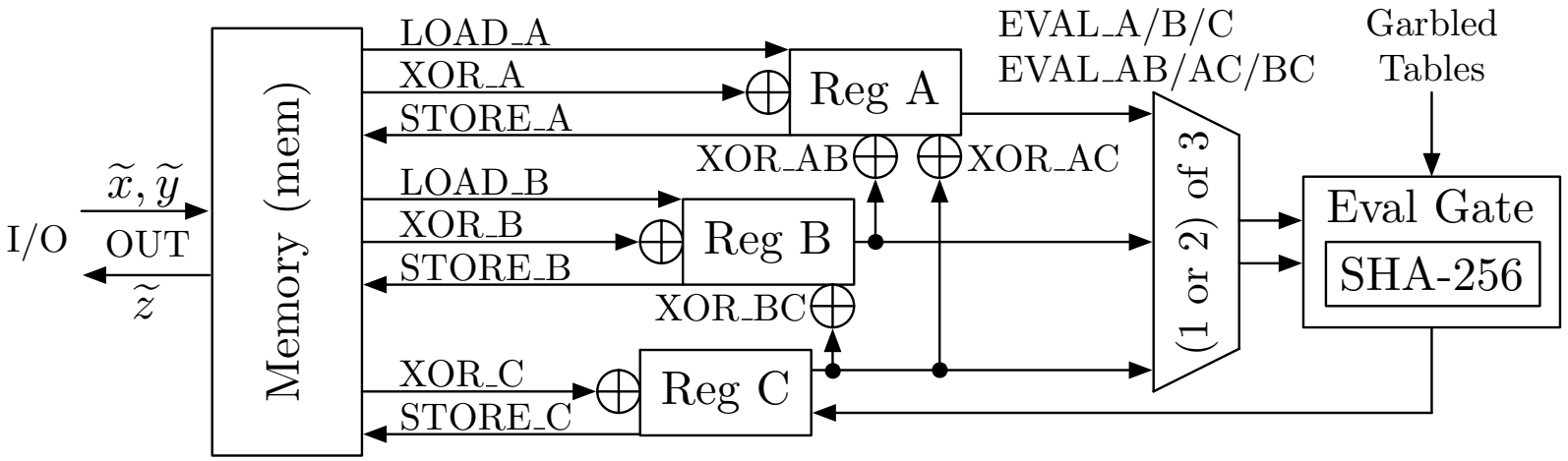


Practical Contribution

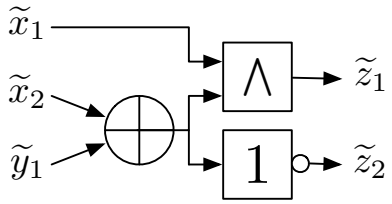
Hardware implementation of GC/OTP evaluation



Architecture for Embedded GC Evaluation

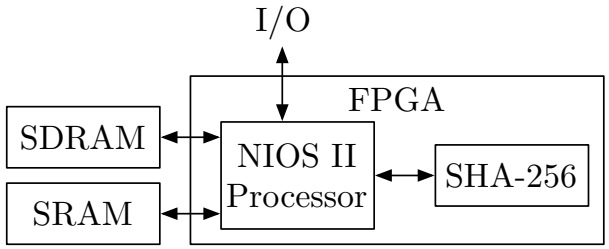


Example Circuit and Instruction Sequence:

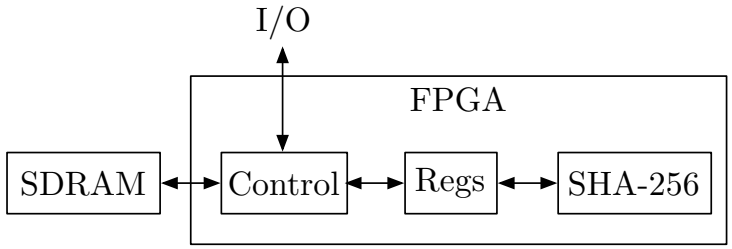


LOAD_A	0x0	// $A \leftarrow \text{mem}[0x0] = \widetilde{x}_1$
LOAD_B	0x1	// $B \leftarrow \text{mem}[0x1] = \widetilde{x}_2$
XOR_B	0x2	// $B \leftarrow B \oplus \text{mem}[0x2] = \widetilde{x}_2 \oplus \widetilde{y}_1$
EVAL_AB		// $C \leftarrow A \wedge B$
STORE_C	0x0	// $\text{mem}[0x0] \leftarrow C$
EVAL_B		// $C \leftarrow \text{not } B$
STORE_C	0x1	// $\text{mem}[0x1] \leftarrow C$
OUT	0x0	// $\widetilde{z}_1 \leftarrow \text{mem}[0x0]$
OUT	0x1	// $\widetilde{z}_2 \leftarrow \text{mem}[0x1]$

Hardware Architectures for GC Evaluation



(a) System-on-a-Programmable-Chip



(b) Stand-Alone Unit

Resources for GC Evaluation on Altera Cyclone II FPGA

Design	LC	FF	M4K
SOPC	7501	4364	22
NIOS II	1104	493	4
SHA-256	2918	2300	8
Stand-Alone Unit	6252	3274	8
SHA-256	3161	2300	8
AES (unprotected)	2418	431	0

Timings of Instructions (average #clock cycles)

Instruction	*SOPC*	*Stand-Alone Unit*
LOAD	291.43	87.63
XOR1	395.30	87.65
XOR2	252.00	1.00
STORE	242.00	27.15
EVAL1	1,282.30	109.95
EVAL2	1,491.68	135.05
OUT	581.48	135.09

Memory access almost as expensive as gate evaluation.

Optimize Circuits for Embedded GC/OTPs

- Memory access slower than computation
  ⇒ cache values in registers to minimize #read/write operations
- XOR gates faster than non-XOR gates ⇒ reduce #non-XOR gates
- Memory expensive ⇒ reduce memory footprint

Table 1. Optimized AES Circuits (Sizes in kB)

	Garbled Circuit $\widetilde{C}$				Program P		Memory for GC Evaluation			
Circuit	non-XOR	1-input	XOR	Size	Instr.	Size	Read	Write	Entries	Size
Baseline	11,286	0	22,594	529	113,054	442	67,760	33,880	34,136	533
Optimized	7,200	40	26,680	338	73,583	287	42,853	22,650	17,315	271

Baseline: circuit of [Pinkas,Schneider,Smart,Williams ASIACRYPT'09]
Optimized: see paper for optimizations applied

Performance of AES OTP

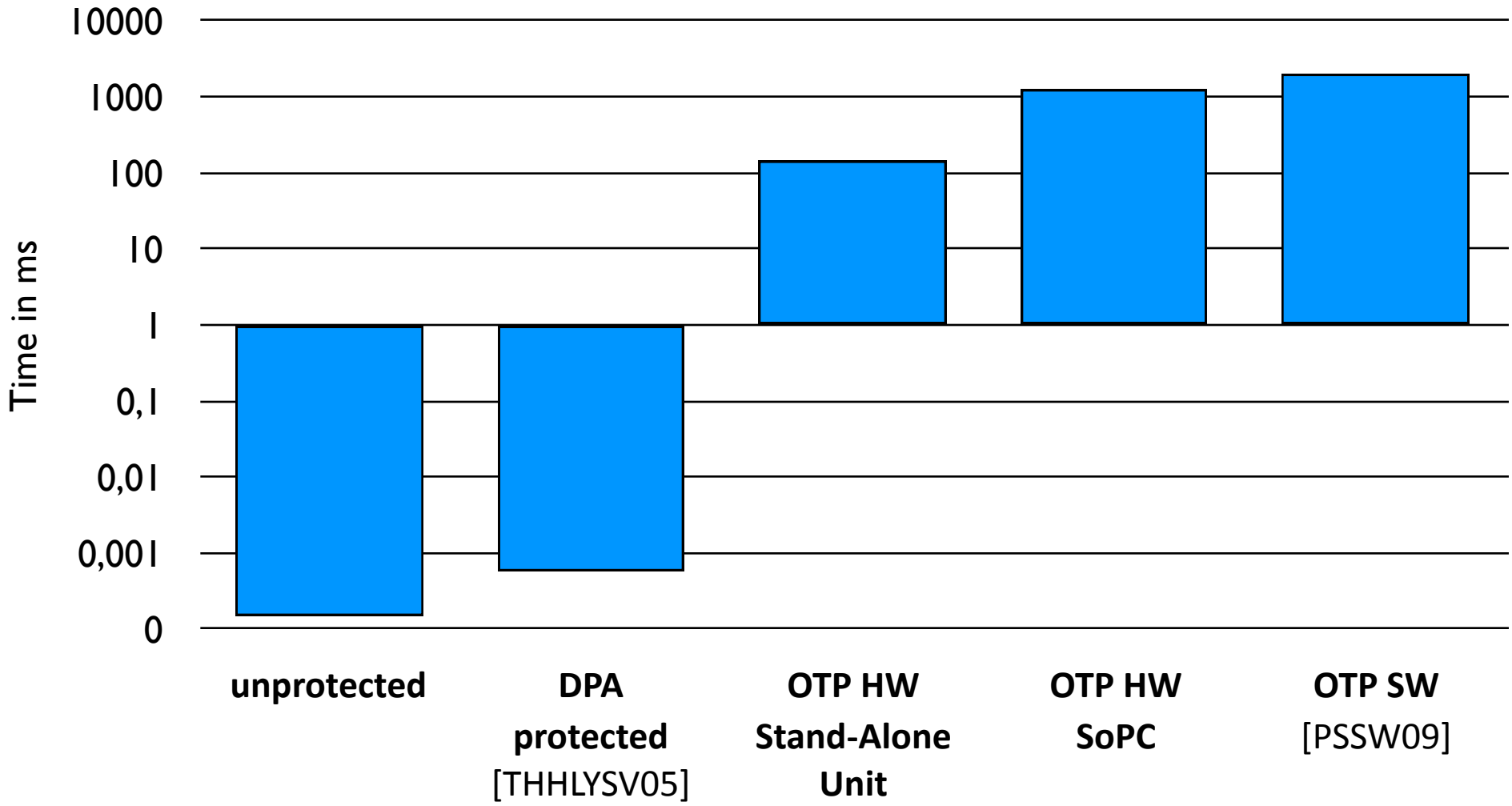
	System-on-a-Programmable-Chip				*Stand-Alone Unit*			
	Clock cycles		Timings (ms)		Clock cycles		Timings (ms)	
Circuit	SHA	Total	SHA	Total	SHA	Total	SHA	Total
Baseline	744,876	94,675,402	14.898	1,893.508	744,876	11,235,118	14.898	224,702
Optimized	477,840	62,629,261	9.557	1,252.585	477,840	7,201,150	9.557	144.023

Overall times dominated by memory access ⇒ key for future improvements

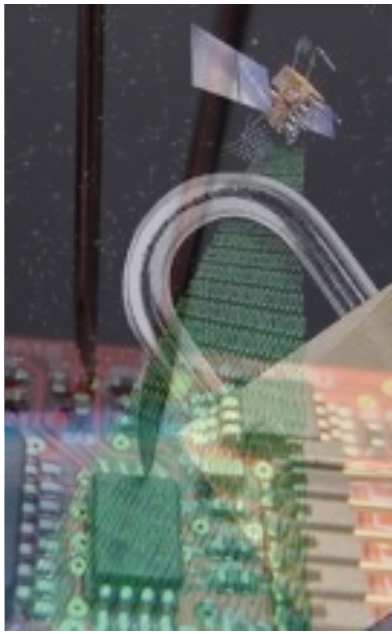
Performance comparison with other AES implementations:

- Unprotected AES: 10 clock cycles = 0.15µs@66MHz
- AES Protected against DPA attacks: ≈ 3.88 · 0.15µs = 0.58µs
  [Tiri,Hwang,Hodjat,Lai,Yang,Schaumont,Verbauwhede CHES'05]
- GC evaluation in Software: 2s on Intel Core 2 Duo 3.0 GHz, 4GB RAM
  [Pinkas,Schneider,Smart,Williams ASIACRYPT'09]

Performance of leakage-protected AES



Summary



Summary: GC/OTPs with improvements

- allow provably secure computations in hostile environment
- can be implemented efficiently in HW
  - 10x faster than SW implementation

- have several restrictions
  - each evaluation requires fresh:
    - GC (AES: 338 kB)
    - masking (e.g., one OTM for each input bit)
  - much slower than unprotected implementations

⇒ for highly security-critical applications only!

Full Version:
http://eprint.iacr.org/2010/276

Contact:
http://www.trust.rub.de