POLITECNICO DI MILANO

Dipartimento di
Elettronica, Informazione
e Bioingegneria

Compiler Technologies for Extra-Functional Properties

Giovanni Agosta

April 23, 2018

Compilers as a commodity?

- *Perception*: compilers come for free, as a result of 30+ years of FOSS development
- *Reality*: major companies (Apple, Google, Sony, ARM, Intel, etc.) still dedicate large, highly qualified teams to compiler development
- *Rationale*:
  - Software ages through functionality bloating
  - Inherent trade-offs (e.g., quality of optimization vs ease of debugging optimized code) lead to circular development patterns
  - Increasing resources to compensate for poor code not a viable strategy any longer

Extra-functional properties

Besides functionality (semantical equivalence of source and generated code):

- Security: mathematical security of cipher primitives not sufficient when implementation attacks are feasible
- Energy-efficiency: battery-powered embedded/mobile systems
- Power-efficiency: limited power envelope for HPC centers, data-centres

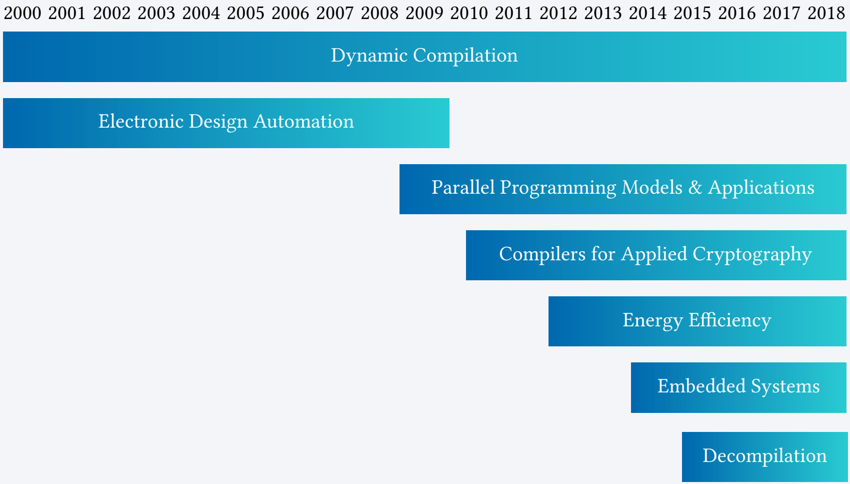
Compilers can help!

Manual solutions are error-prone and tedious, compilers offer:

- Understanding program behaviour through static and dynamic analysis
- Code transformation and insertion, while preserving desired semantics

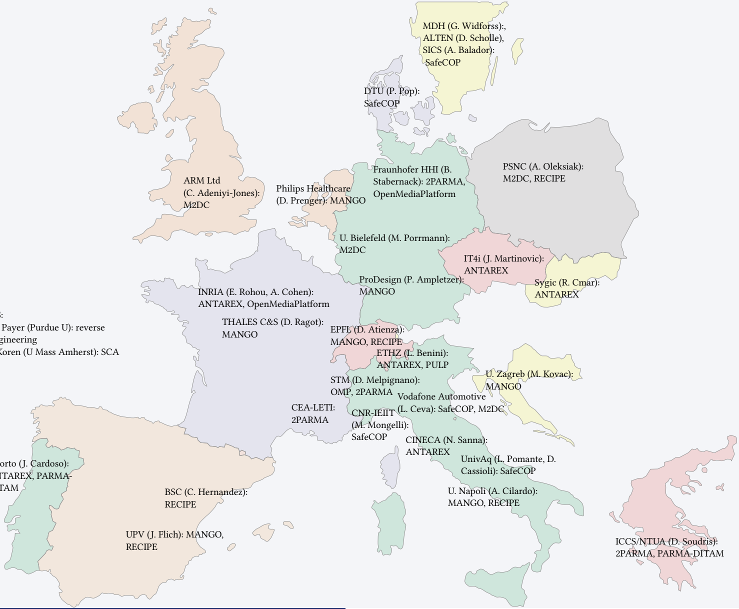
Compiler Technology in the III Millennium

Historical perspective



Compiler Technology in the III Millennium

Collaboration map



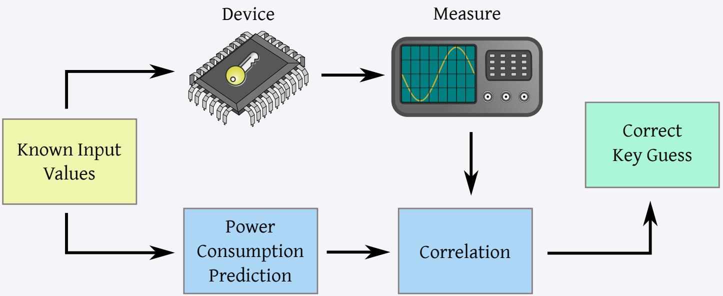
US:
M. Payer (Purdue U): reverse engineering
I. Koren (U Mass Amherst): SCA

- A cipher implementation leaks information over different *Side Channels* (power consumption, EM)
- Vast corpus of successful Side Channel Attacks against *software* implementations of ciphers on embedded μCs and CPUs

State of the Art: no general solution

- Costly manual analysis of vulnerability
- Ad-hoc countermeasures, manually designed and implemented, based on:
  - *Protected Logic Styles* provide data independent power consumption
  - *Hiding*: add noise to pollute measurements
  - *Masking*: given a sensitive operation, split each input operand in a set of randomized *shares* and perform an equivalent computation on them

Power/EM Side Channel Attack – Workflow

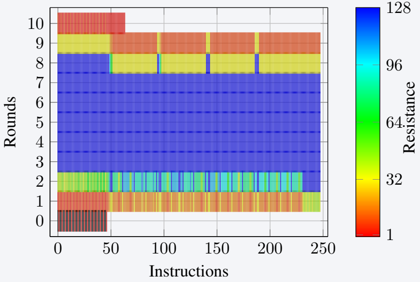


- Consider an `xor` instruction `I` involving e.g., 1-byte `k` of the key

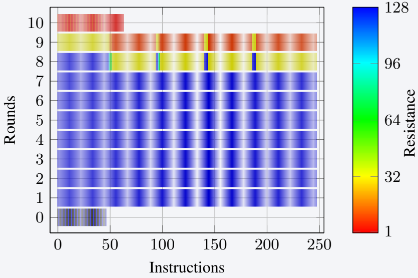
1. Given a set of input values `v`, compute a power consumption prediction for `I` for each possible value of `k`: e.g., `HW(v xor k)`
2. Correlate all the predictions with the actual power consumption
3. The prediction fitting best the actual measured side-channel is the one based on the correct key guess

Computation of a (power/EM) SCA resistance metric

- Resistance can be measured, for each bit of each intermediate value of a computation using a secret key, as the number of secret key bits involved in the computation of the intermediate value
- This information can be computed through the data flow analysis (DFA) framework, a classic compiler analysis technique
- The analysis is conservative, returning the *minimum* number of secret key bits
- Both a forward and a backward DFA are needed

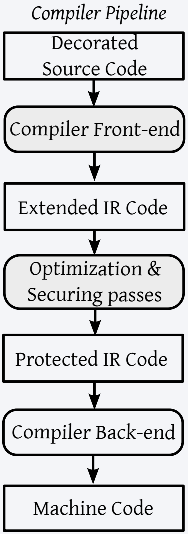


Resistance to known input attack



Resistance to known output attack

- Blue areas are secure from SCA
- Red areas are vulnerable
- We can now apply countermeasures only to the vulnerable areas



- Change the side-channel profile of cryptographic code at every run
- Key idea: morphing the execution trace

Sketch of the Transformation

1. Locate a sensitive instruction in the original code
2. Look-up the set of semantically equivalent code-fragment templates
3. Instantiate the fragment templates in separate basic blocks
4. Bind the basic blocks together in a `switch-case` construct driven by a *run-time* generated *nonce*

A successful basic research line

- Publication at Design Automation Conference in 2012, 2013, 2014, 2015, ICCAD 2016, and several more conferences
- Achieved 4 HiPEAC paper awards
- Achieved best paper award (cryptography section and overall) at SIN 2014
- Publication on IEEE TCAD, IET Comp. & Dig. Tech., Inf. Proc. Lett. (Elsevier)

Technology maturation & exploitation

- Technology integrated in industry-grade LLVM compiler framework
- Adopted as *technology brick* of ECSEL SafeCOP project (2016-2019)

Current research lines

Energy-Efficiency & Performance

Basic research line

Achieve energy- and time-to-solution improvements by means of

Memoization Define sufficiently *pure* functions to apply automatically memoization

Architecture-Compiler Co-Optimization Explore compiler mid- and back-end analyses and transformations to detect and exploit specific patterns that are amenable to optimized hardware implementation

Precision tuning Explore reduction of computation precision (e.g., from floating point to fixed point)

Technology maturation & exploitation

- Precision tuning technology developed within FET-HPC ANTAREX project (2015-2018)

Current research lines

Parallel Programming Models

Basic research line

Current programming models are difficult to use in deeply heterogeneous HPC architectures

- Explored CUDA and OpenCL by designing several *record-setting* implementations of encryption primitives on GPGPU
- Investigated the impact of architectural features on performance
- Defined an easy to use *programming model* that supports integrated resource management

Technology maturation & exploitation

- Resource managed programming model developed within FET-HPC MANGO project (2015-2018)
- Adopted as baseline technology of FET-HPC RECIPE project (2018-2020)

Current research lines

Reverse Engineering

Basic research line

Core idea: leverage existing compiler frameworks (LLVM, QEMU)

- Binary-to-binary for legacy code support
- Binary-to-intermediate for binary analysis
- Contributions:
  - ISA-independent DFA to recover indirect branch targets by tracking value ranges
  - Accurate recovery of function boundaries

Technology maturation & exploitation

- Two post-doc researchers active on technology maturation

Research Directions

Parallel Programming Models

- Exposing runtime managed resources in OpenCL and other industrial programming models
- Extending runtime managed programming model to non-HPC scenarios

Energy-Efficiency

- Improving precision tuning with stronger error guarantees and reliable prediction of performance improvements
- Integrating precision tuning with other approximate computing techniques for overall precision/performance/energy tradeoff

Research Directions

Reverse Engineering

- Automated recovery of high-level constructs
- Production of C code for manual and automated analyses
- Applications: plagiarism-detection, vulnerability analysis

Applied Cryptography

- Extend SDFA to VHDL for vulnerability analysis of hardware specifications

Acknowledgements

PoliMi staff co-authors

- A. Barenghi
- C. Brandolese
- L. Breveglieri
- F. Bruschi
- S. Crespi Reghizzi
- W. Fornaciari
- C. Francalanci
- F. Garzotto
- A. Miele
- G. Palermo
- G. Pelosi
- M. Santambrogio
- D. Sciuto
- C. Silvano

Current Post-Doc & PhD Students

- A. Di Federico
- S. Cherubin
- P. Fezzardi
- A. Pupykina

Past PhD Students

- M. Bessi
- A. Di Biagio
- M. Scandale
- E. Speziale

and all my MS students as well!

HEAP Lab People

- L. Cremona
- D. Gadioli
- S. Libutti
- G. Massari
- N. Mainardi
- F. Reghenzani
- F. Rossi
- F. Terraneo
- E. Vitali
- M. Zanella
- D. Zoni
- and all past members