FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Stealthy Integration of Software Protections

prof. Bjorn De Sutter

dr. Bart Coppens

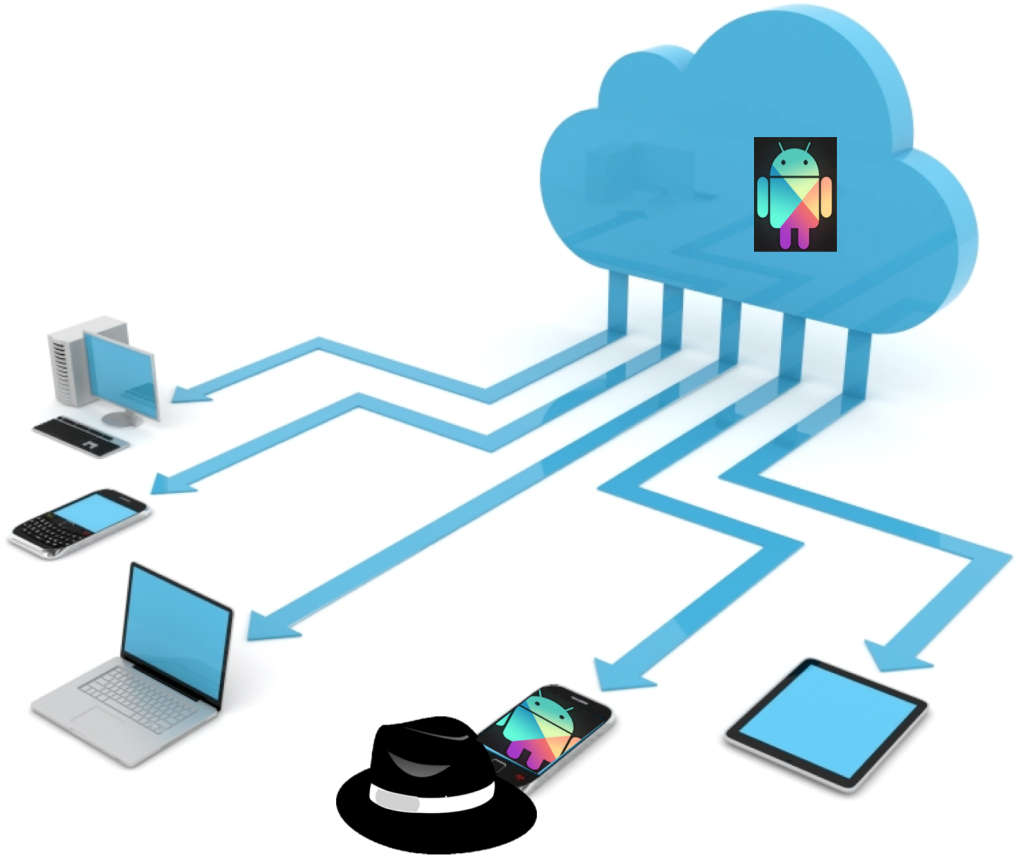
ir. Jens Van den Broeck

UNIVERSITEIT GENT

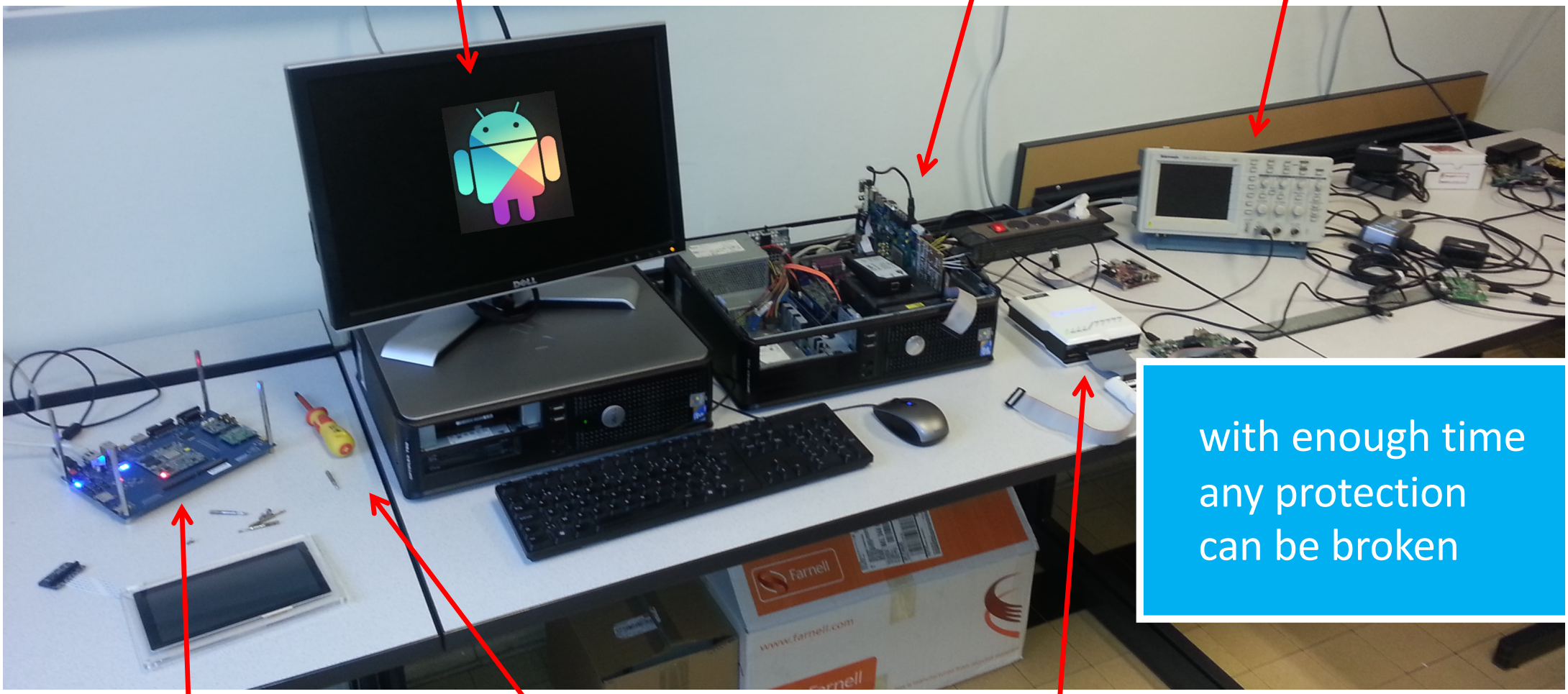
bjorn.desutter@ugent.be

St. Pölten, 8 Nov 2019

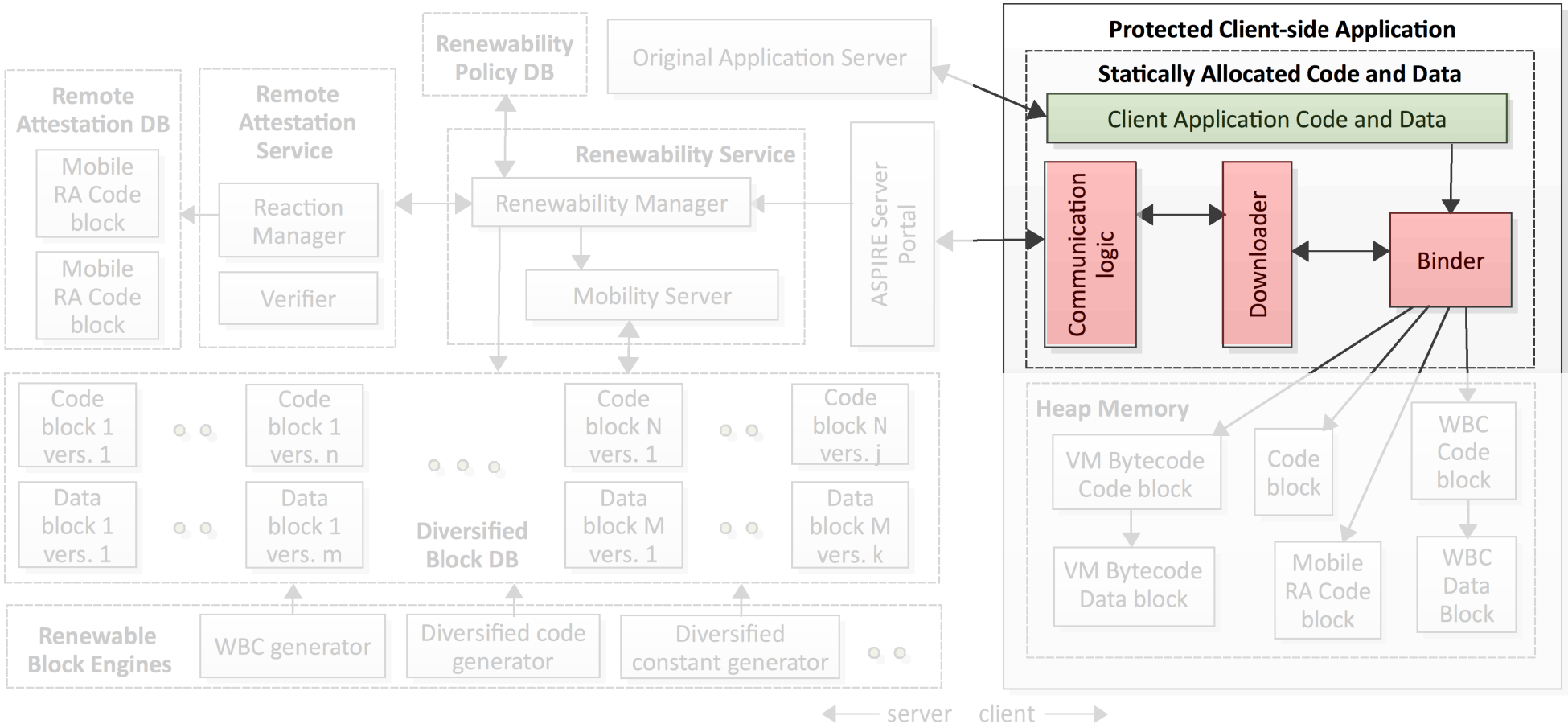
Man At The End Attacks



assets in software

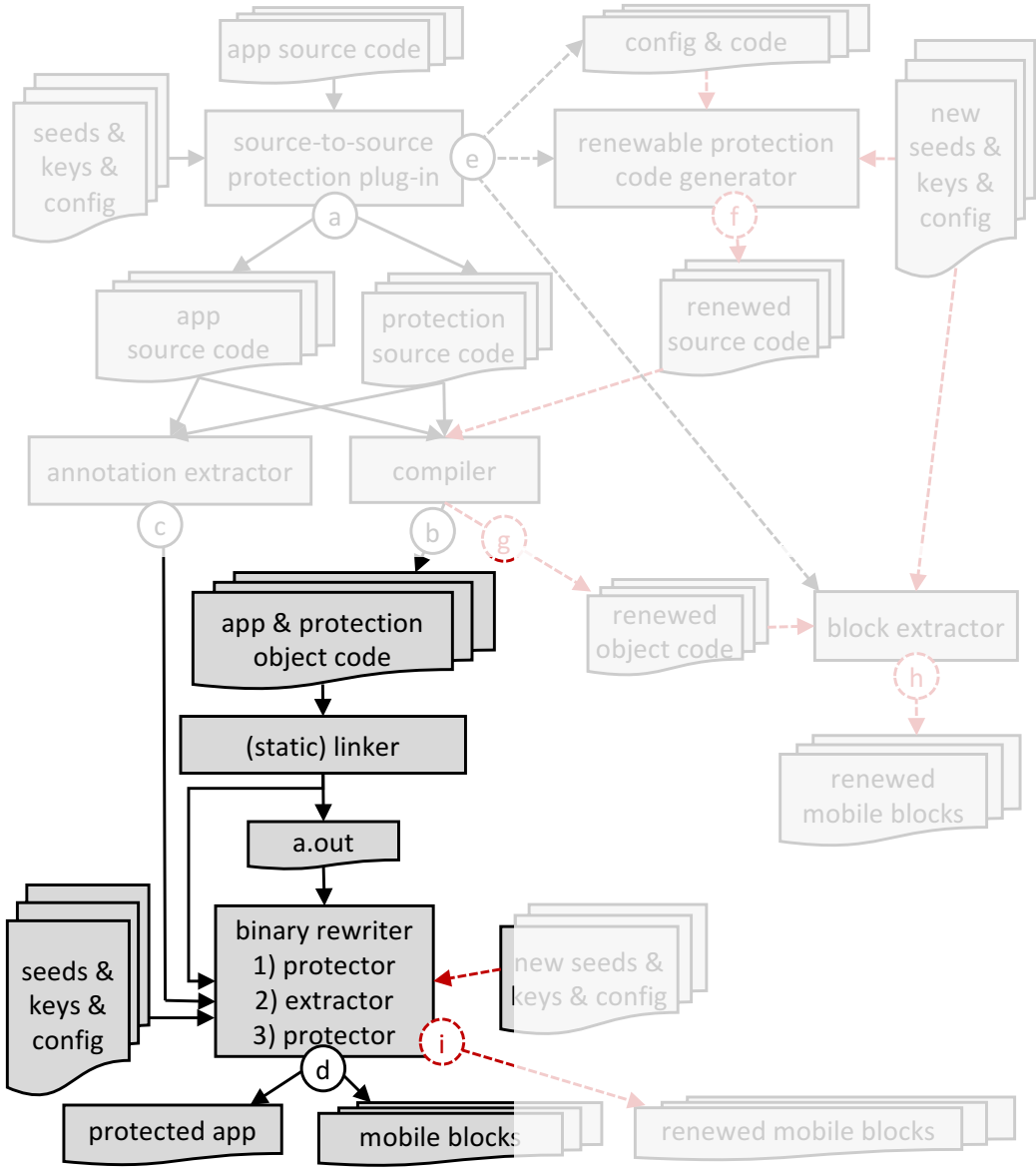

software analysis tools
FPGA sampler
oscilloscope
with enough time
any protection
can be broken
developer boards
screwdrivers
JTAG debugger

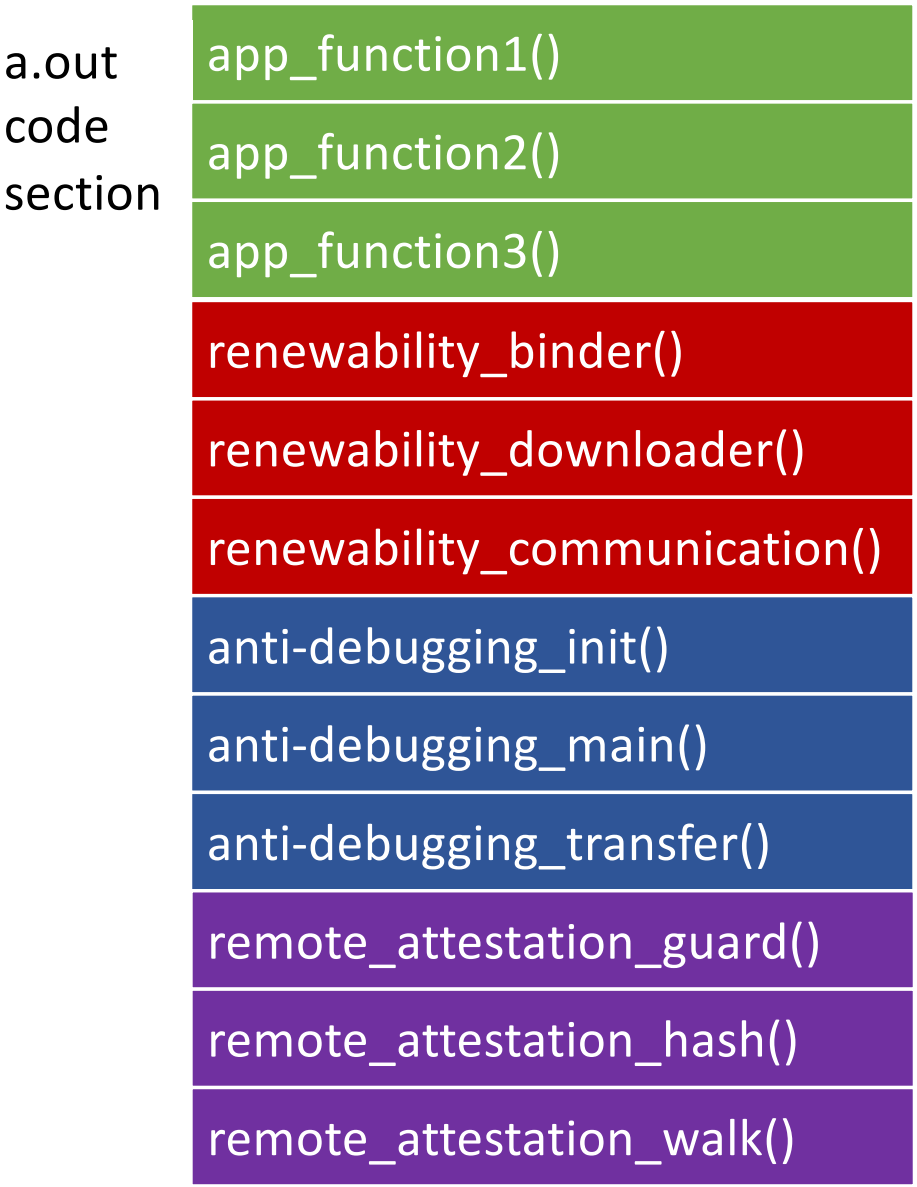
Software Protections



a.out
code
section

app_function1()
app_function2()
app_function3()
renewability_binder()
renewability_downloader()
renewability_communication()
anti-debugging_init()
anti-debugging_main()
anti-debugging_transfer()
remote_attestation_guard()
remote_attestation_hash()
remote_attestation_walk()

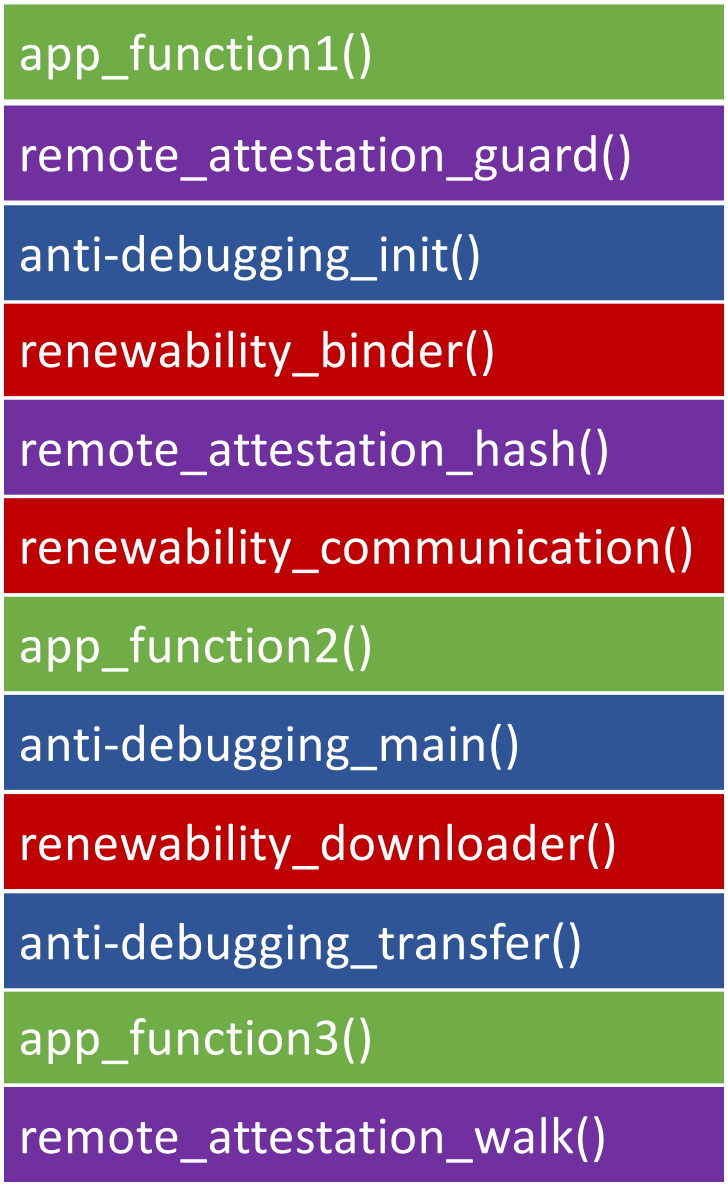




- Two fundamental problems
  - related code is grouped
  - each code fragment implements only one semantics
- This eases attacker's job
  - identifying interesting fragments
  - code comprehension
  - tampering
  - overcoming protections
- Our solution
  1. code layout randomization
  2. inserting fake edges
  3. deduplicating code fragments

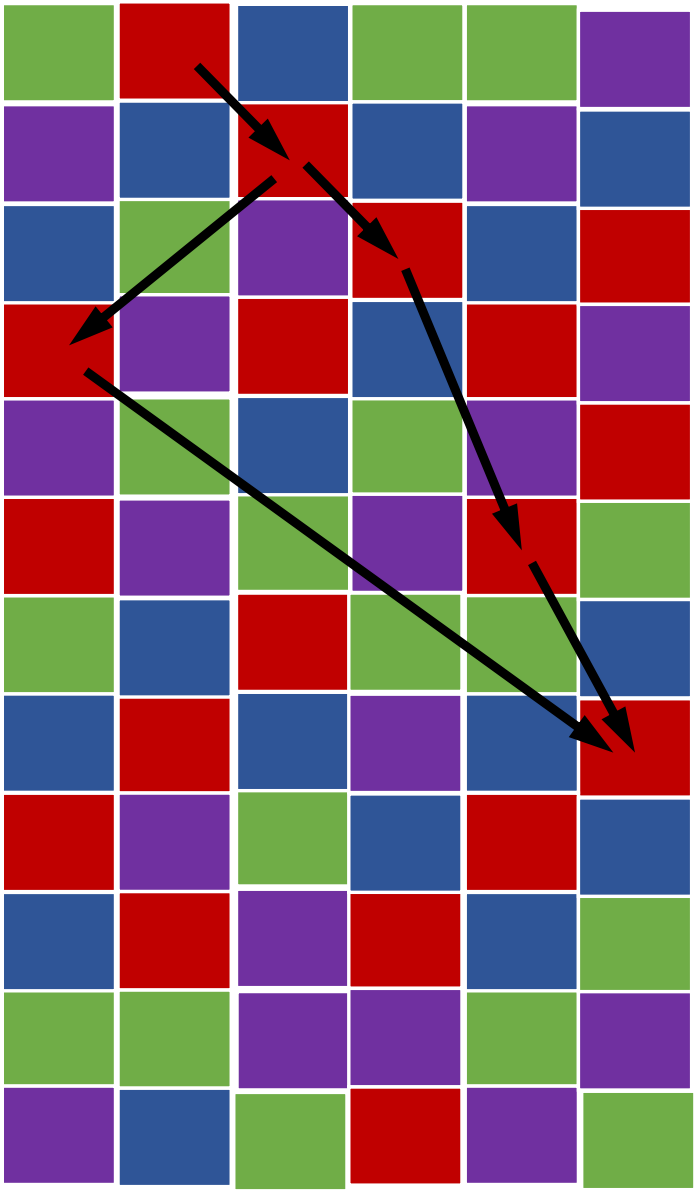
1) Code Layout Randomization

- Simple at function level
- Relatively weak
- Improvement
  - replace direct calls by indirect obfuscated ones



1) Code Layout Randomization

- Harder at basic block level
  - requires (post) link-time code rewriter
- Still relatively weak
  - recursive descent disassemblers
- Improvement
  - replace direct jumps by indirect ones
- Limitation
  - hides information but does not mislead the attacker

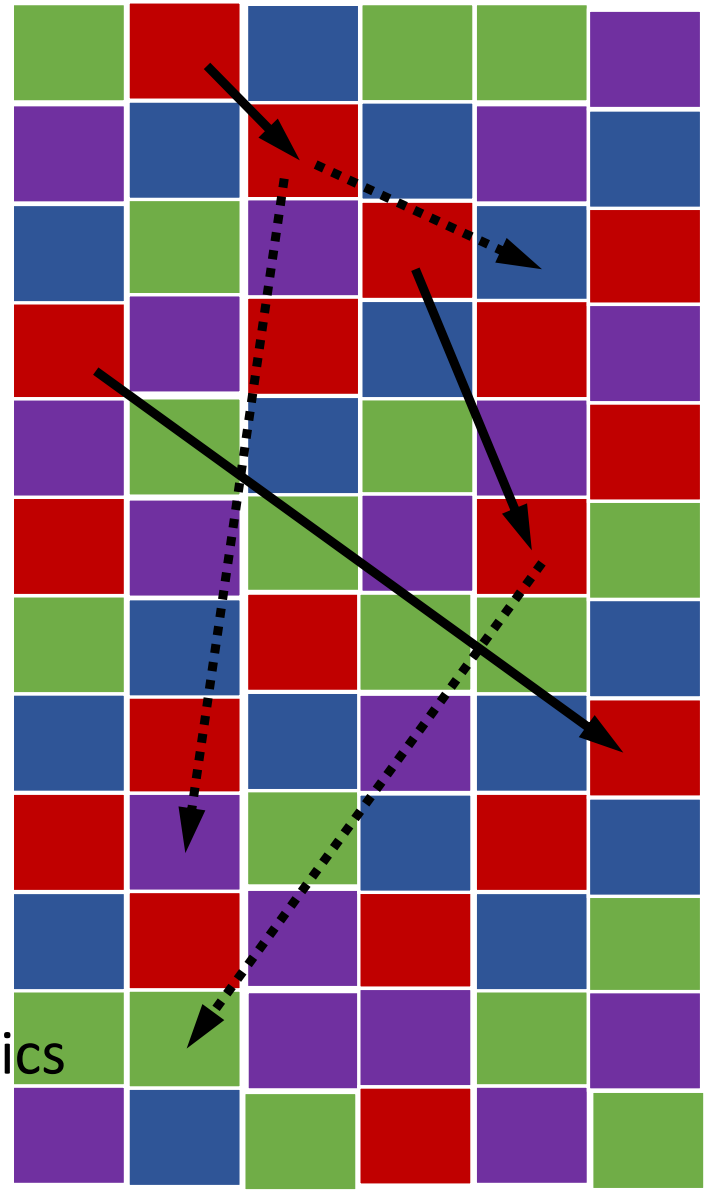


2) Opaque predicates

- Insert interprocedural fake direct CF edges with intraprocedural CF idioms

- Attacks are still possible
  - detect opaque predicates
    - pattern matching
    - abstract interpretation
    - symbolic execution
  - detect invariant behavior
    - generic deobfuscation

- Each fragments still implements only one semantics



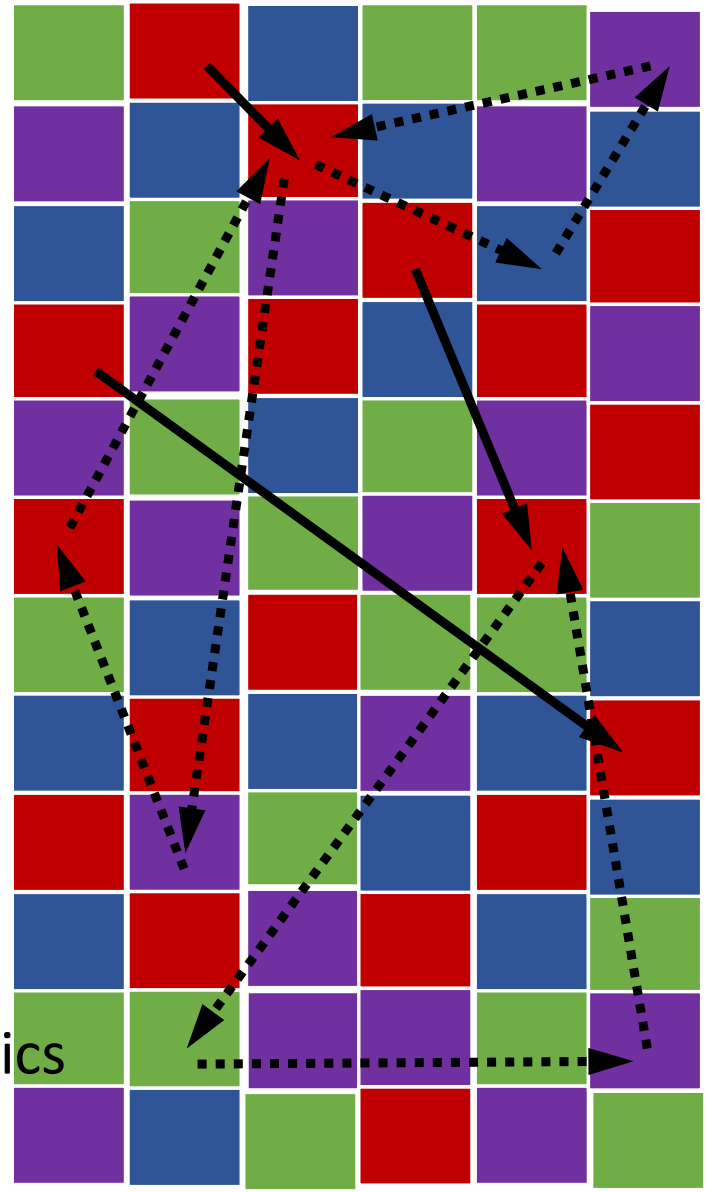
2) Opaque predicates

- Insert interprocedural fake direct CF edges with intraprocedural CF idioms

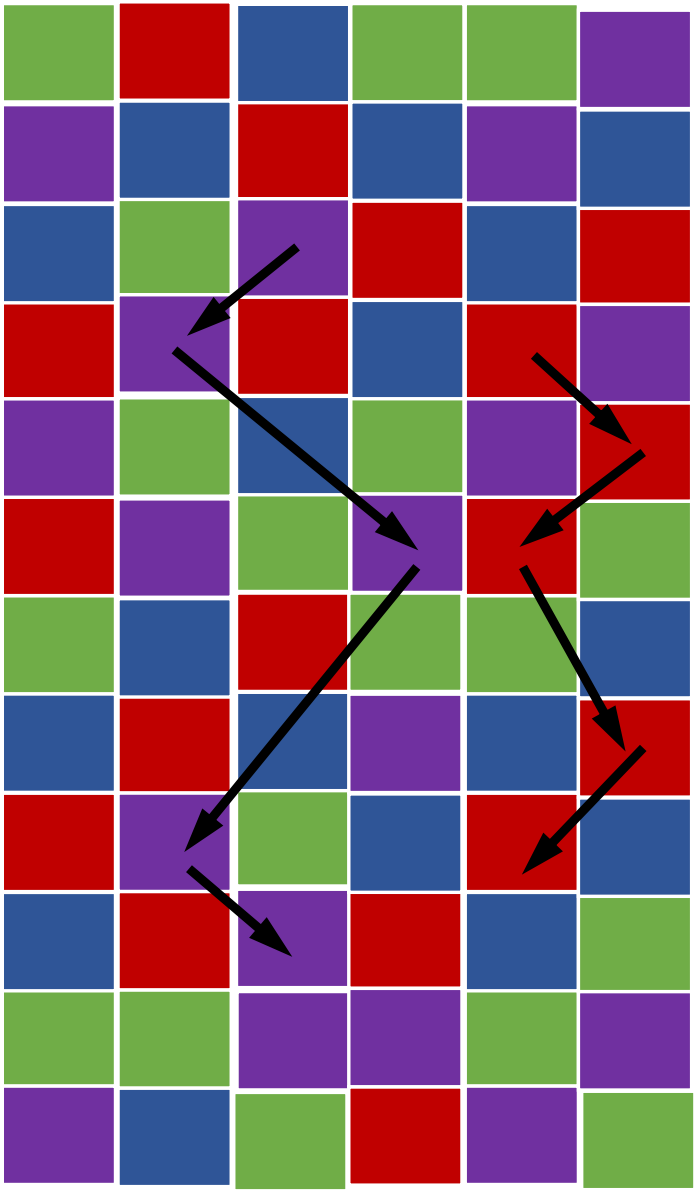
- Attacks are still possible
  - detect opaque predicates
    - pattern matching
    - abstract interpretation
    - symbolic execution
  - detect invariant behavior
    - generic deobfuscation

more, coupled fake edges!

- Each fragments still implements only one semantics



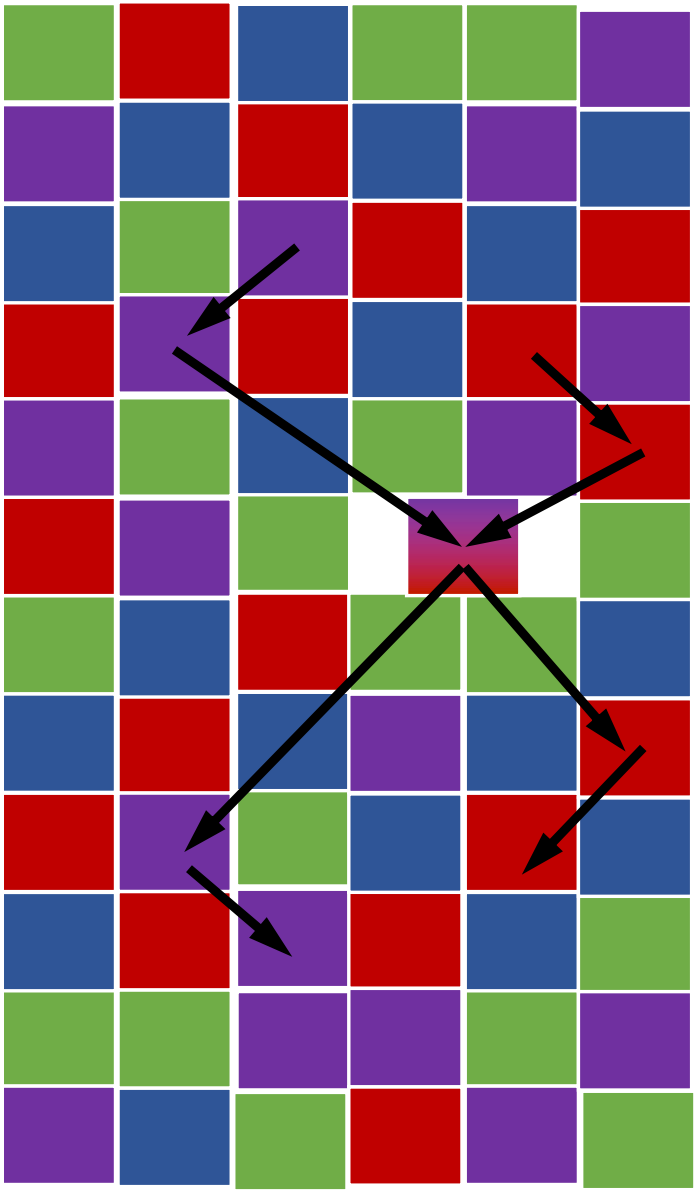
- So far, different components are not connected by true CF edges



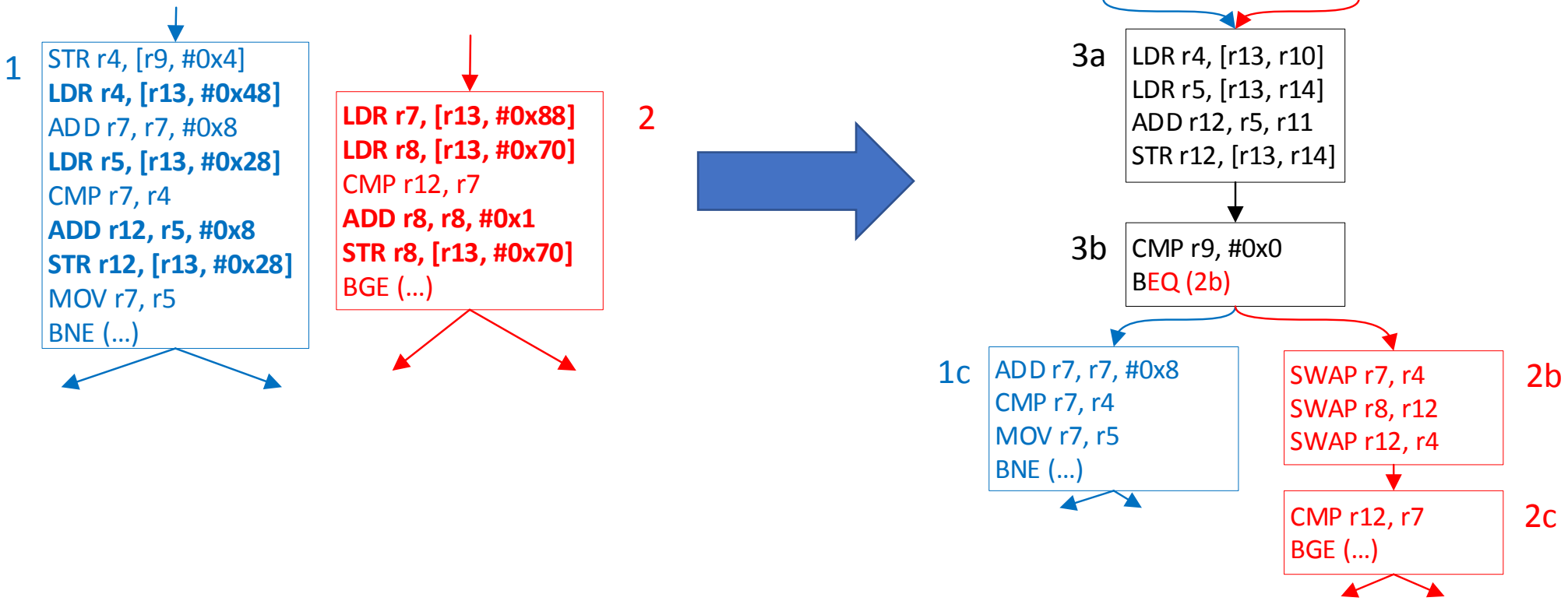
3) Code deduplication

- So far, different components are not connected by true CF edges

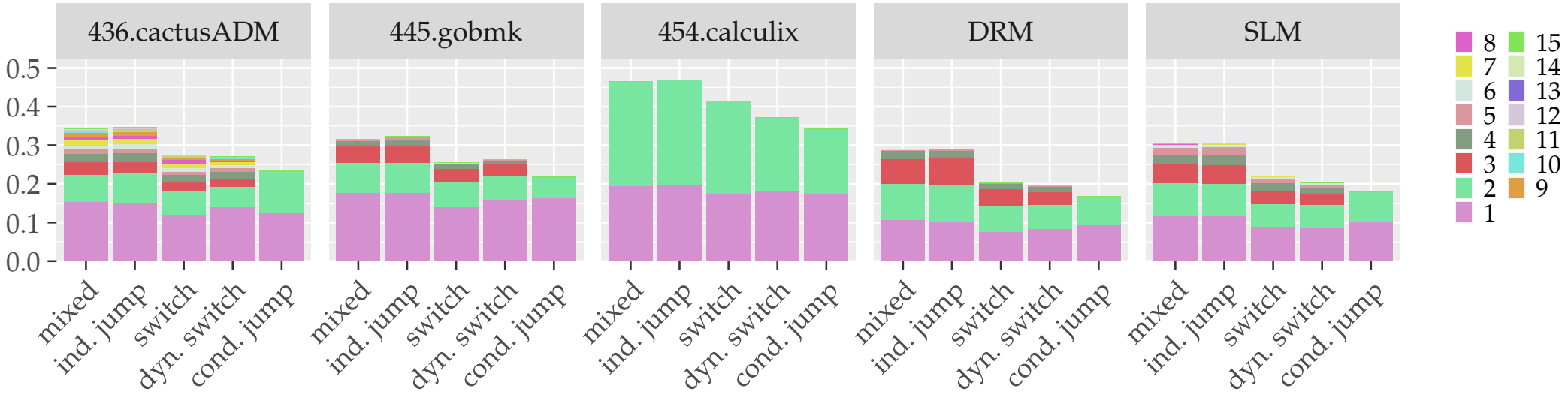
- Now components are connected by true edges
- Now code implements multiple semantics
- Now outgoing edges can both be taken
- If both are covered, we have variant behavior



Prototype Implementation



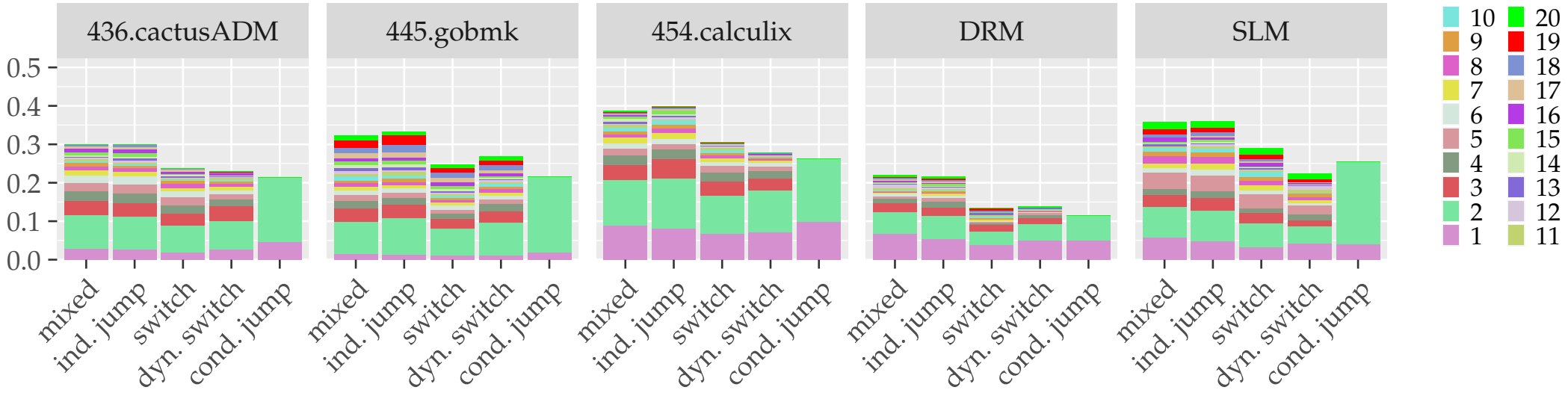
instructions that can be deduplicated



stacked colours: number of different components involved in deduplication

all instructions are taken into account

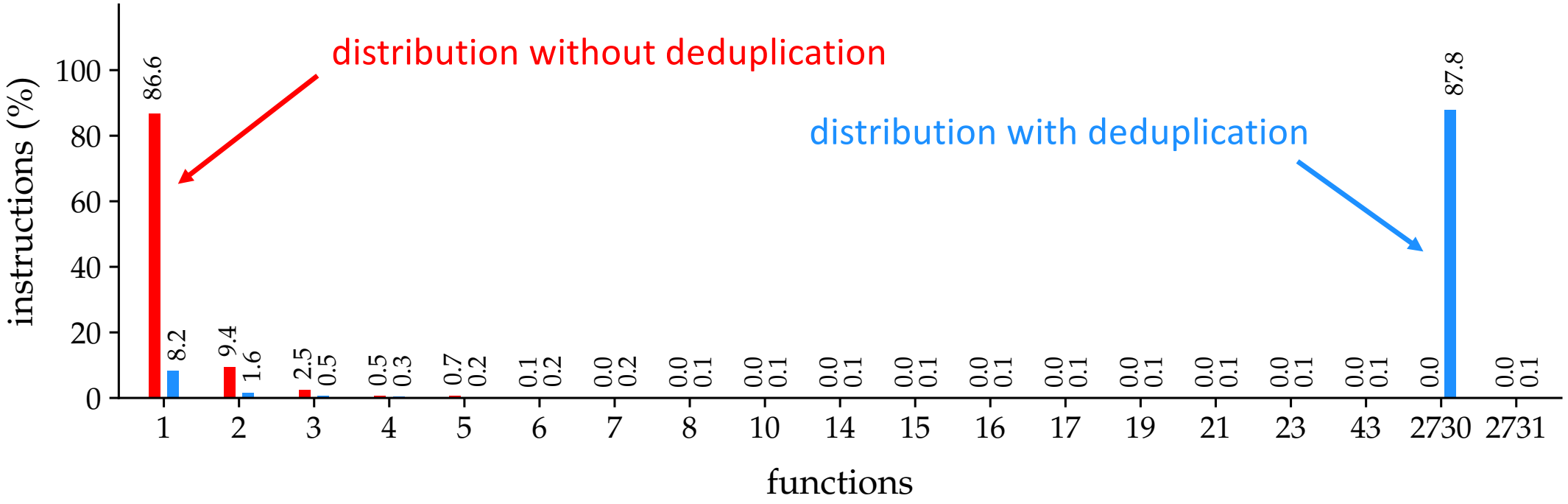
instructions that can be deduplicated



stacked colours: number of different contexts involved in deduplication

only executed instructions are taken into account

functions to which instructions belong



Reconstruction of control flow graphs in IDA Pro

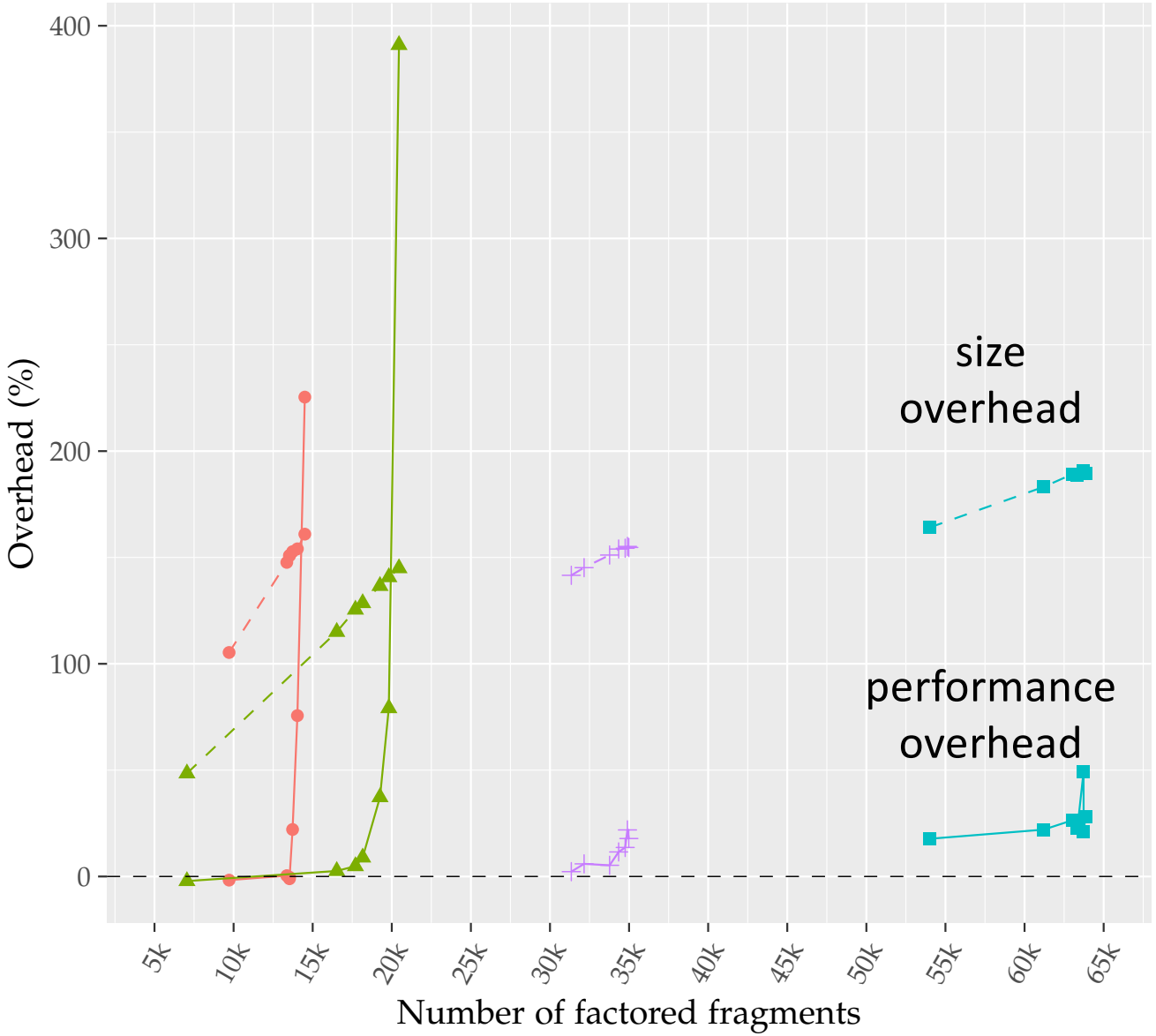
	FP/FN CFG edges drawn in GUI							FP/FN CFG edges stored in database						
	Total	**IA**	**IO**	**IF**	**iA**	**iO**	**iF**	**Total**	**IA**	**IO**	**IF**	**iA**	**iO**	**iF**
# FP	14.9k	8.3k	10.9k	11.0k	6.6k	4.0k	4.0k	18.2k	10.3k	14.1k	14.2k	7.9k	4.1k	4.0k
FPR	67%	38%	49%	49%	30%	18%	18%	82%	46%	64%	64%	36%	18%	18%
# FN	73.0k	13	448	526	73.0k	72.6k	72.5k	27.5k	0	17	20	27.5k	27.5k	27.5k
FNR	40%	0%	0%	0%	40%	40%	40%	15%	0%	0%	0%	15%	15%	15%

Pairs of fragments split by factorization		
Total	Wrong	Correct
28.4k	24.0k (85%)	4.4k (15%)

Opaque predicates	
Total	Resolved
13.3k	3.0k (22%)

unsound attack after extending IDA Pro with basic, custom heuristics

Protection vs. overhead



Conclusions

- Hide the boundaries of integrated software protection components by
  - randomizing code layout at basic block level
  - inserting coupled opaque predicates with fake edges across components
  - deduplicating (factoring) common code fragments
- We can hence integrate protection components stealthily
- The protection has configurable potency and at least some resilience