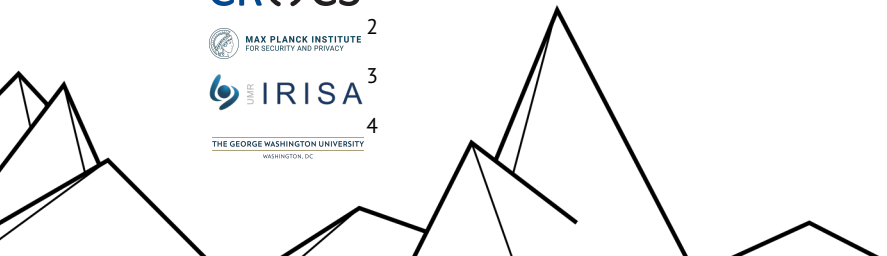
“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks

Jan Jancar[1], Marcel Fourné[2], Daniel De Almeida Braga[3], Mohamed Sabt[3], Peter Schwabe[2], Gilles Barthe[2], Pierre-Alain Fouque[3] and Yasemin Acar[2,4]

CROCS [1]
MAX PLANCK INSTITUTE FOR SECURITY AND PRIVACY [2]
UMR IRISA [3]
THE GEORGE WASHINGTON UNIVERSITY WASHINGTON, DC [4]

IEEE S&P 2022
bit.ly/3riKHWB



Timing attacks

Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher **1996**

- **When?** 25+ years old

Timing attacks

Timing Attacks on Implementations of
Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher **1996**

- **When?** 25+ years old
- **What?** Duration of operation leaks information on secrets

Timing attacks

Timing Attacks on Implementations of
Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher **1996**

- **When?** 25+ years old
- **What?** Duration of operation leaks information on secrets
- **Why?** Branches or memory accesses on secret-derived values

Timing attacks

Timing Attacks on Implementations of
Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher **1996**

- **When?** 25+ years old
- **What?** Duration of operation leaks information on secrets
- **Why?** Branches or memory accesses on secret-derived values
- They are still around

Timing attacks

Resistance and tools for verification

- Constant-time code practice
- Tools to verify constant-timeness

Tool	Target	Technique
ABPV13	C	Formal
Binsec/Rel	Binary	Symbolic
Blazer	Java	Formal
BPT17	C	Symbolic
CacheAudit	Binary	Formal
CacheD	Trace	Symbolic
COCO-CHANNEL	Java	Symbolic
ctgrind	Binary	Dynamic
ct-fuzz	LLVM	Dynamic
ct-verif	LLVM	Formal
CT-WASM	WASM	Formal
DATA	Binary	Dynamic
dudect	Binary	Statistics
FaCT	DSL	Formal
FlowTracker	LLVM	Formal
haybale-pitchfork	LLVM	Symbolic
KMO12	Binary	Formal
MemSan	LLVM	Dynamic
MicroWalk	Binary	Dynamic
SC-Eliminator	LLVM	Formal
SideTrail	LLVM	Formal
Themis	Java	Formal
timecop	Binary	Dynamic
tis-ct	C	Symbolic
VirtualCert	x86	Formal

Why are timing attacks still around?

Are timing attacks part of threat models of libraries?

How do libraries protect against timing attacks?

Are developers aware of the tools?

Why are the tools not being used effectively?

Why are timing attacks still around?

- Let's ask the crypto library developers!
- They are the ones that would fix them

Are timing attacks part of threat models of libraries?

How do libraries protect against timing attacks?

Are developers aware of the tools?

Why are the tools not being used effectively?

Ran a survey

Sample

- Targeted open-source cryptographic libraries
- Most-active contributors
  - number of commits
- Invited 201 developers from 36 libraries
  - 44 valid responses
  - 27 libraries
- Thanks to our participants!

Libraries

OpenSSL, LibreSSL,
Amazon s2n, libgcrypt,
RustCrypto, libsecp256k1
…

Developers

11 core developers,
19 maintainers,
11 committers,
…

Ran a survey

Content

1. Participant background

Asked about

- Background in cryptography
- Experience developing cryptographic code
- Academic / Industry background

Ran a survey

Content

1. Participant background

↓

2. Library / Primitive properties and decisions

Asked about

- ⚥ role in library
- Library design decisions
- Library threat model
- Timing attack protections in library
- Testing of timing attack resistance of library

Ran a survey

Content

1. Participant background

↓

2. Library / Primitive properties and decisions

↓

3. Tool awareness

Asked about

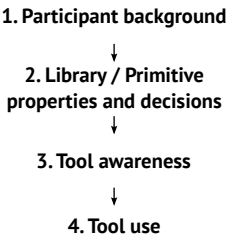
- ♀ awareness of tools
- How ♀ learned about them

Ran a survey

Content

Asked about

- Experience with using tools

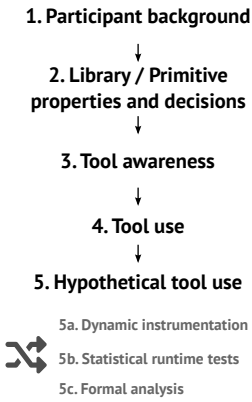


Ran a survey

Content

Presented 🧍 properties of three groups of tools

- Groups of tools
  - Dynamic instrumentation
  - Statistical runtime tests
  - Formal analysis
- Properties
  - Requirements on code
  - Guarantees on the results
- Asked about likeliness of use and reasoning

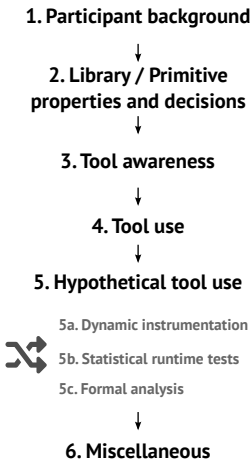


Ran a survey

Content

Asked about general thoughts on

- Timing attacks
- Our survey



Developers know about timing attacks...

- 100% knew about timing attacks
- Opinions varied

*"They're **not that hard to mitigate**, at least with the compilers I'm using right now."*

*"It was totally obvious for everybody right from the start that **protection** against timing attacks is **necessary**."*

*"For many cases there **aren't enough real world attacks** to justify spending time on preventing timing leaks."*

. . . and consider them a threat. . .

- Threat models of libraries
  - Included timing attacks: 23
  - Did not include: 2
- Libraries differentiate between local and remote attacks
  - Include remote: 20
  - Include local: 16

*"We worry mostly about timing now. These can vary, **remote observation is obviously a bigger issue**, local observation cannot be discounted either."*

. . . and consider them a threat. . .

- Threat models of libraries
  - Included timing attacks: 23
  - Did not include: 2
- Libraries differentiate between local and remote attacks
  - Include remote: 20
  - Include local: 16
- Reasoning varied
  - Timing attacks relevance, loss of reputation, target environment of library

*“We worry mostly about timing now. These can vary, **remote observation is obviously a bigger issue**, local observation cannot be discounted either.”*

*“Yes. They [timing attacks] are a concern for some users. And **it is never fun to be the bug** that a new research paper is talking about exploiting :)”*

. . . that is worth protecting against.

- Claimed resistance against timing attacks
  - Yes, fully: 13
  - **Partially**: 10
  - No: 3

“It’s just how you write cryptographic code, every other way is the wrong approach (unless in very specific circumstances or if no constant-time algorithm is known).”

. . . that is worth protecting against.

- Claimed resistance against timing attacks
  - Yes, fully: 13
  - **Partially**: 10
  - No: 3
- Various protection techniques
  - Constant-time code practice: 21
  - Constant-time algorithm: 9
  - Blinding, slicing, assembly, hardware features, **random delays**

"It's just how you write cryptographic code, every other way is the wrong approach (unless in very specific circumstances or if no constant-time algorithm is known)."

"Conditional branches and lookups are avoided on secrets. Assembly code and common tricks are used to prevent compiler optimizations."

Most heard about the tools...

- 33 🧍 heard of at least one
- Well-known tools:
  - ct-grind: 27 🧍
  - ct-verif: 17 🧍
  - MemSan: 8 🧍
- **Most of the tools were unknown**

> *"We independently came up with this approach and were using it [before we] knew ctgrind existed."*

Most heard about the tools...

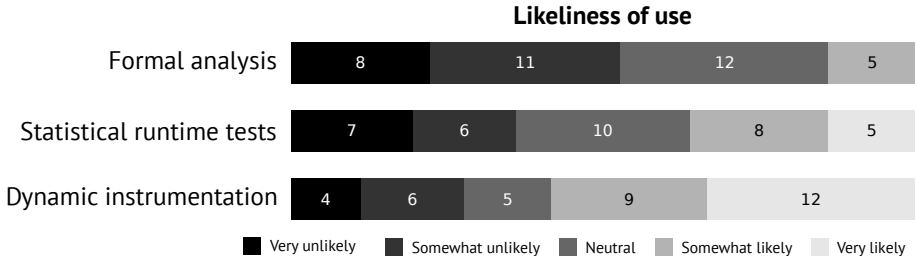
- 33 🧍 heard of at least one
- Well-known tools:
  - ct-grind: 27 🧍
  - ct-verif: 17 🧍
  - MemSan: 8 🧍
- **Most of the tools were unknown**

...but haven't tried using them,

- Only 19 🧍 tried to use
- Why not?
  - **Lack of time**: 26
  - Inability to ignore issues: 8
  - Tool not maintained: 5
  - Tool not available: 4

"We independently came up with this approach and were using it [before we] knew ctgrind existed."

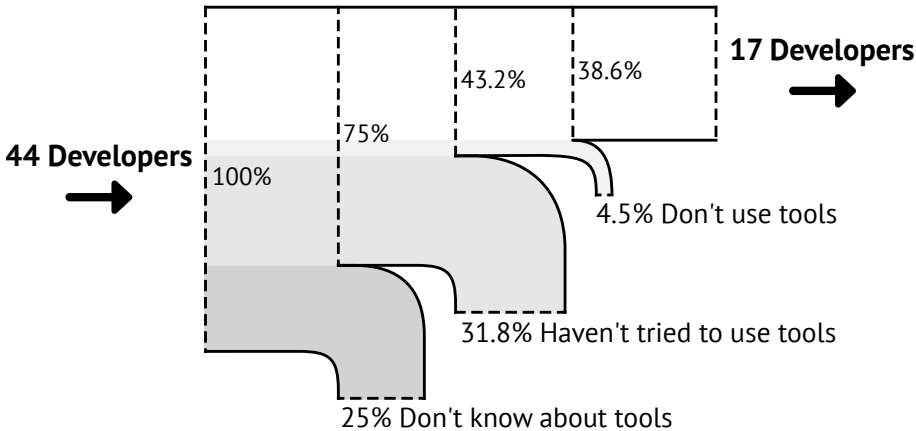
. . . and are unlikely to use some of them.



- Formal analysis
  - Perceived as too much effort: 22 ♀
- Dynamic & Statistical tools
  - Acceptable trade-off between effort and guarantees: 10 ♀

"I'm very interested in these sorts of tools, but so far it seems formal analysis tools (at least where we've tried to apply it to correctness) are ***not really usable by mere mortals yet****."*

There is a leaky pipeline of developers using tools.



Recommendations

Tool developers

- Make tools usable
  - Available
  - Easy to install
  - Documentation, examples
- Promote tools at appropriate venues

Recommendations

Tool developers

- Make tools usable
  - Available
  - Easy to install
  - Documentation, examples
- Promote tools at appropriate venues

Crypto developers

- Use the tools, automate, include in CI
- Eliminate all timing leaks
- Mark secrets in code

Recommendations

Tool developers

- Make tools usable
  - Available
  - Easy to install
  - Documentation, examples
- Promote tools at appropriate venues

Crypto developers

- Use the tools, automate, include in CI
- Eliminate all timing leaks
- Mark secrets in code

Compiler writers

- Support secret types
  - Do not introduce timing leaks
- Give more control to developers
  - To stop introduction of timing leaks

Recommendations

Tool developers

- Make tools usable
  - Available
  - Easy to install
  - Documentation, examples
- Promote tools at appropriate venues

Crypto developers

- Use the tools, automate, include in CI
- Eliminate all timing leaks
- Mark secrets in code

Compiler writers

- Support secret types
  - Do not introduce timing leaks
- Give more control to developers
  - To stop introduction of timing leaks

Standardization bodies

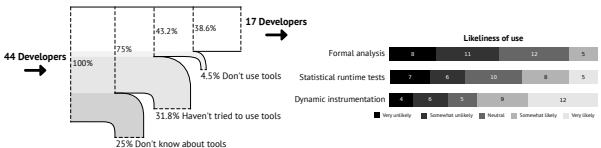
- Encourage submitters to use tools
- Require constant-time code

“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks

Jan Jancar, Marcel Fourné, Daniel De Almeida Braga, Mohamed Sabt, Peter Schwabe, Gilles Barthe, Pierre-Alain Fouque and Yasemin Acar

Developers

- Know and care about timing attacks
- Do not know most tools for verifying constant-timeness
- Do not use tools, mostly due to lack of time



Questions?

J08nY

bit.ly/3riKHWB