AUTOMATED PROGRAM ANALYSIS: FROM SAFETY TO HYPERSAFETY

Lesly-Ann Daniel

CEA LIST, Université Côte d'Azur

INTRODUCTION

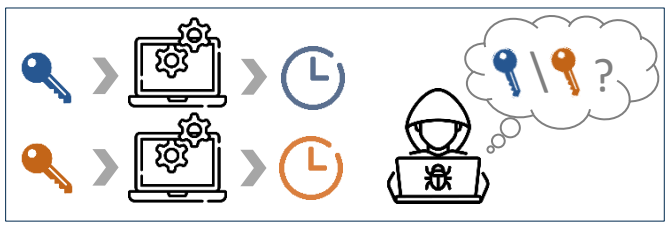
Software take an increasing place in our society and are used in many critical systems:

- encrypt our communications
- manipulate health data
- secure banking transactions, etc.

It is crucial to ensure not only that these software are bug-free (safety), but also that they preserve the **confidentiality of secret data** they manipulate (security).

Safety vs. Security.

- **Safety**: no bugs (e.g. crash due to a division by 0) along *one execution* of the program.
- **Security**: a program does not leak secret (e.g. crypto keys) to an attacker.
  Relates *pairs of executions* (**2-hypersafety**).



Problem.
We have automated bug-finding tools for safety, but we lack automated bug-finding tools for 2-hypersafety.

Goal.
Adapt automated bug-finding tools for safety to security (2-hypersafety).

We focus on a crucial 2-hypersafety property to protect against timing attacks: **constant-time**.

BINARY ANALYSIS AGAINST TIMING ATTACKS

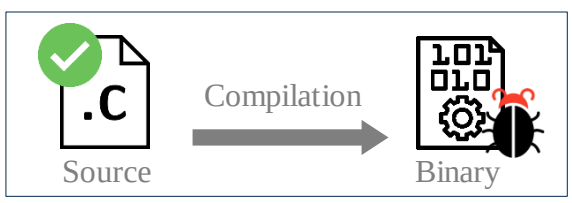
Timing attacks exploit the execution time of a program to leak secret data.

```
bool password_check(guess, pass, length) {
  for (int i = 0; i < length; i++) {
    if (guess[i] != pass[i]) return false;
  }
  return true;
}
```

Constant-time programming ensures that execution time is independent from secrets. Implemented in cryptographic libraries like OpenSSL, BearSSL, Libsodium, etc.

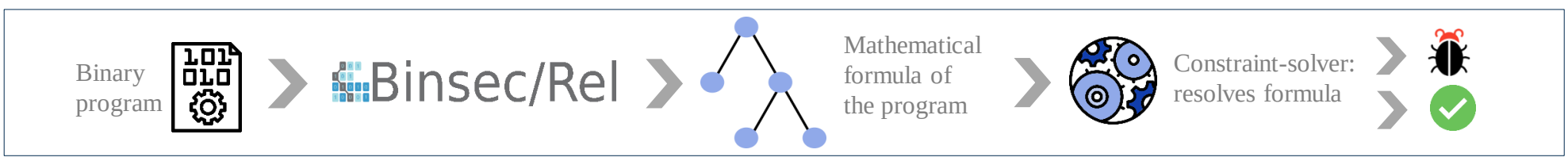
Challenges of constant-time analysis:

- 2-hypersafety ⇒ requires to reason about pairs of executions efficiently
- Not necessarily preserved by compilers ⇒ requires binary analysis



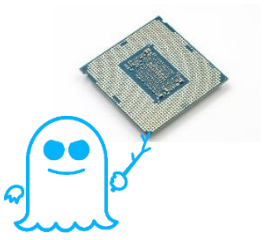
Our contributions [1]:

- **Binary-level RelSE,** a new relational symbolic execution technique for constant-time analysis at binary level
- Based on **dedicated optimizations** (speedup of 2 orders of magnitude)
- Implementation in the **Binsec/Rel tool**: found 2 **new bugs** introduced by the compiler & **new security proofs** at binary-level for 296 crypto binaries



WHEN PROCESSORS SPECULATE AGAINST US

In 2018, **Spectre attacks** [2] exploit optimizations based on *speculative execution* in processors to open new possibilities for timing attacks, even in constant-time programs.



New challenge: Efficiently model the speculative behavior of the processor to protect software against Spectre attacks.

Our contributions (under submission):

- **HauntedRelSE**: new optimizations for constant-time **analysis under speculation**
- Implementation & experiments on crypto
- New **attacks** & **countermeasures**

CONCLUSION

We work on closing the gap in **automated bug-finding techniques** between safety and **hypersafety**.

Applications to **security analysis** of **cryptographic programs** against timing attacks.

We developed a tool, **Binsec/Rel**, for **constant-time** analysis at **binary-level** and extended it to encompass new classes of **Spectre** attacks.

REFERENCES

[1] Daniel, L., Bardin, S., & Rezk, T., *Binsec/Rel: Efficient Relational Symbolic Execution for Constant-Time at Binary-Level*, IEEE Symposium Security and Privacy, 2020.
[2] Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., Yarom, Y., *Spectre Attacks: Exploiting Speculative Execution*, IEEE Symposium on Security and Privacy, 2019.
Icons made by Freepik, bqlqn, and becris from Flaticon.

For Women in Science
FONDATION L'ORÉAL