KU LEUVEN

ProSpeCT: Provably Secure Speculation for the Constant-Time Policy

September 29th 2023

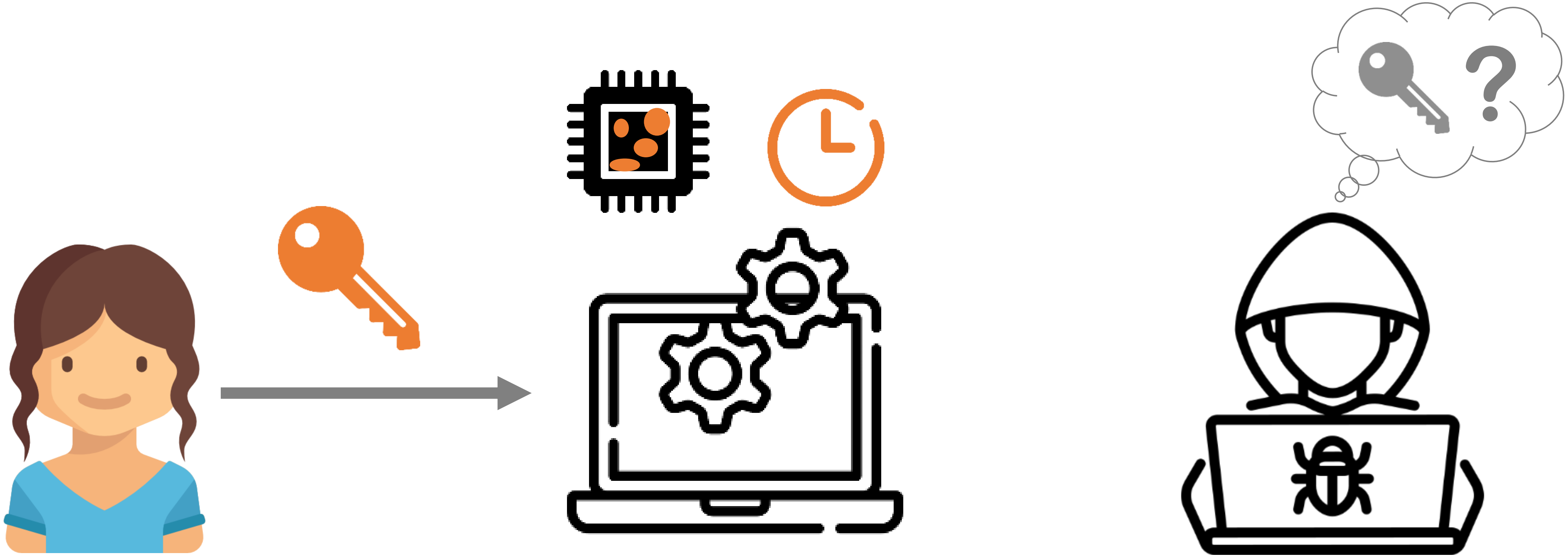
SPLiTS Security Workshop

32ND USENIX SECURITY SYMPOSIUM

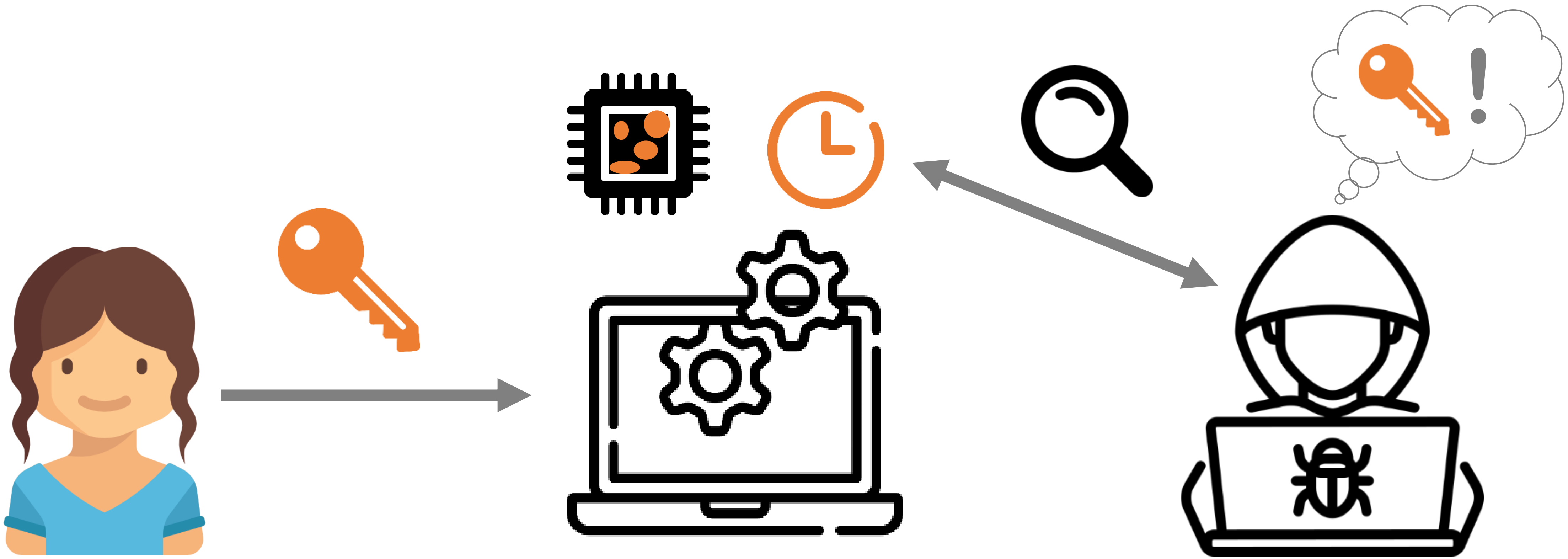
AUGUST 9–11, 2023
ANAHEIM, CA, USA

Lesly-Ann Daniel	Marton Bognar	Job Noorman	Sébastien Bardin	Tamara Rezk	Frank Piessens
KU Leuven	KU Leuven	KU Leuven	CEA List	INRIA	KU Leuven

Need to protect against microarchitectural attacks

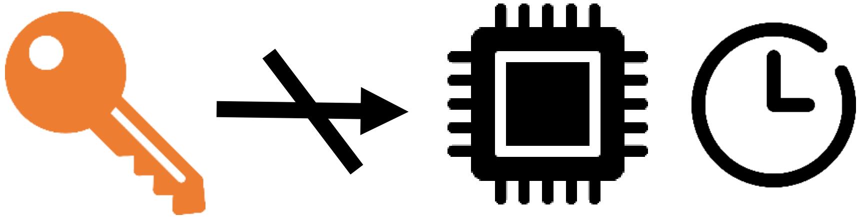


Need to protect against microarchitectural attacks



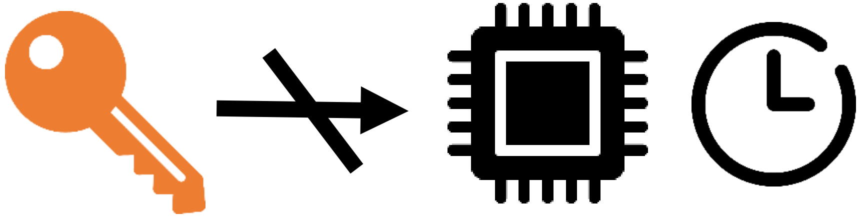
Easy: constant-time programming!

```
char array[len]
char mysecret
if (idx < len)
    x = array[idx]
    leak(x)
```



Easy: constant-time programming!

```
char array[len]
char mysecret
if (idx < len)
    x = array[idx]
    leak(x)
```



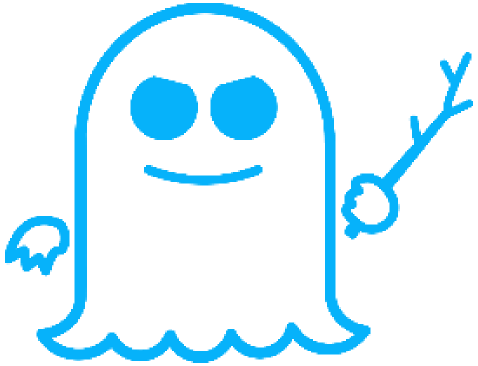
mysecret $\not\rightarrow$ leak()

De facto standard for crypto

… still vulnerable to Spectre attacks

```
char array[len]
char mysecret
if (idx < len)
    x = array[idx]
    leak(x)
```

Predict branch taken



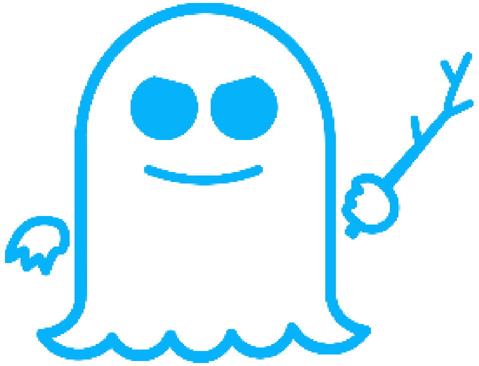
… still vulnerable to Spectre attacks

```
char array[len]
char mysecret
if (idx < len)
    x = array[idx]
    leak(x)
```

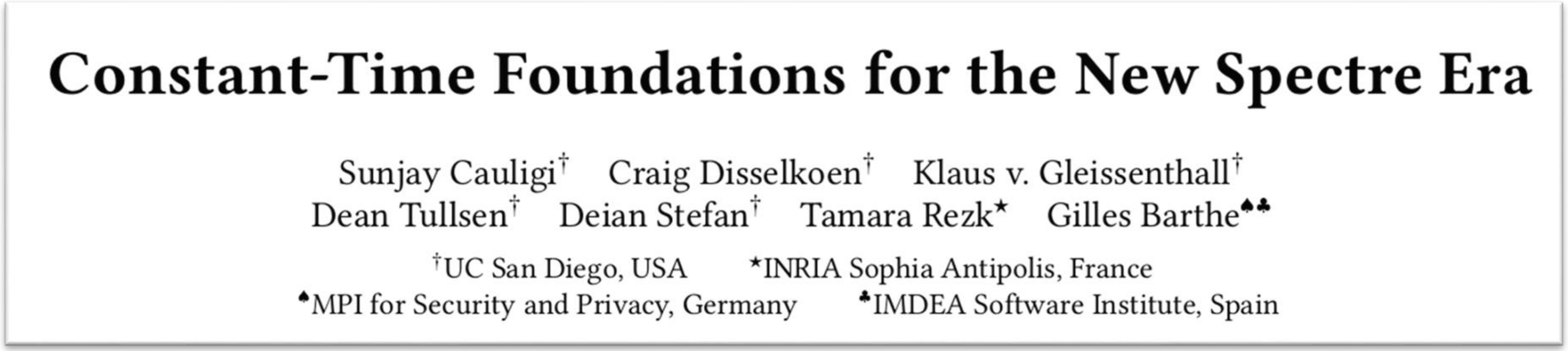
Predict branch taken

`x = mysecret`

Leaks `mysecret` to microarchitecture!



How can I protect my code?



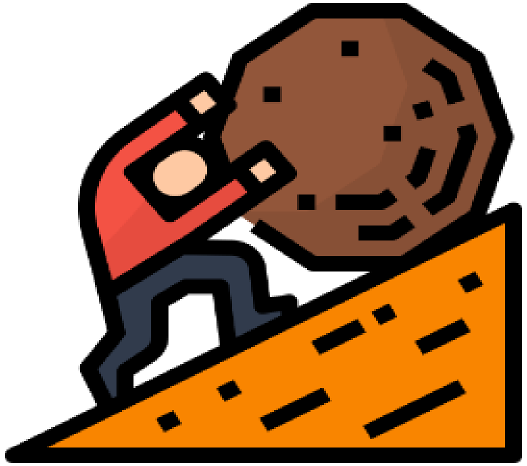
Constant-Time Foundations for the New Spectre Era

Sunjay Cauligi[†] Craig Disselkoen[†] Klaus v. Gleissenthall[†]
Dean Tullsen[†] Deian Stefan[†] Tamara Rezk[⋆] Gilles Barthe[♠♣]

[†]UC San Diego, USA [⋆]INRIA Sophia Antipolis, France
[♠]MPI for Security and Privacy, Germany [♣]IMDEA Software Institute, Spain

Speculative constant-time

- Hard to reason about
- New speculation mechanisms?



We need Secure Speculation for Constant-Time!

Developers should not care about speculations

Hardware shall not speculatively leak secrets

But still be efficient and enable speculation

Hardware defense:

Secure speculation for constant-time!

Hardware Secrecy Tracking

Software side

- Label secrets
- Constant-time program

Hardware side

- Track security labels
- Secrets do not speculatively flow to insecure instructions

ConTExT: A Generic Approach for Mitigating Spectre

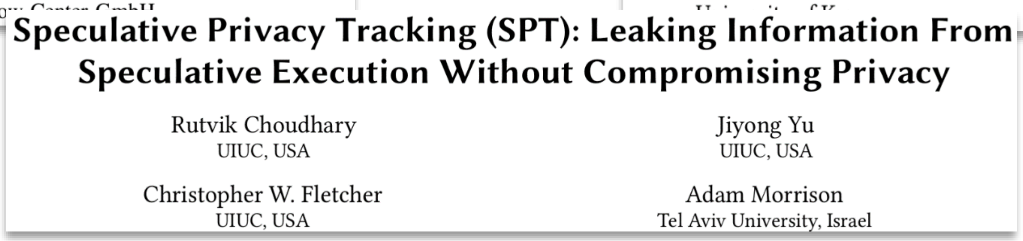
Michael Schwarz[1], Moritz Lipp[1], Claudio Canella[1], Robert Schilling[1,2], Florian Kargl[1], Daniel Gruss[1]
[1]Graz University of Technology [2]Kno

SpectreGuard: An Efficient Data-centric Defense Mechanism against Spectre Attacks

Jacob Fustos

Farzad Farshchi
University of Kansas

Heechul Yun
University of Kansas



Speculative Privacy Tracking (SPT): Leaking Information From Speculative Execution Without Compromising Privacy

Rutvik Choudhary
UIUC, USA

Jiyong Yu
UIUC, USA

Christopher W. Fletcher
UIUC, USA

Adam Morrison
Tel Aviv University, Israel

Illustration with Spectre-v1

```
char array[len]
char mysecret
if (idx < len)
    x = array[idx]
    leak(x)
```

Consider `idx = len`

Illustration with Spectre-v1

```
char array[len]
secret char mysecret
if (idx < len)
    x = array[idx]
    leak(x)
```

Developer marks secrets

Consider `idx = len`

Illustration with Spectre-v1

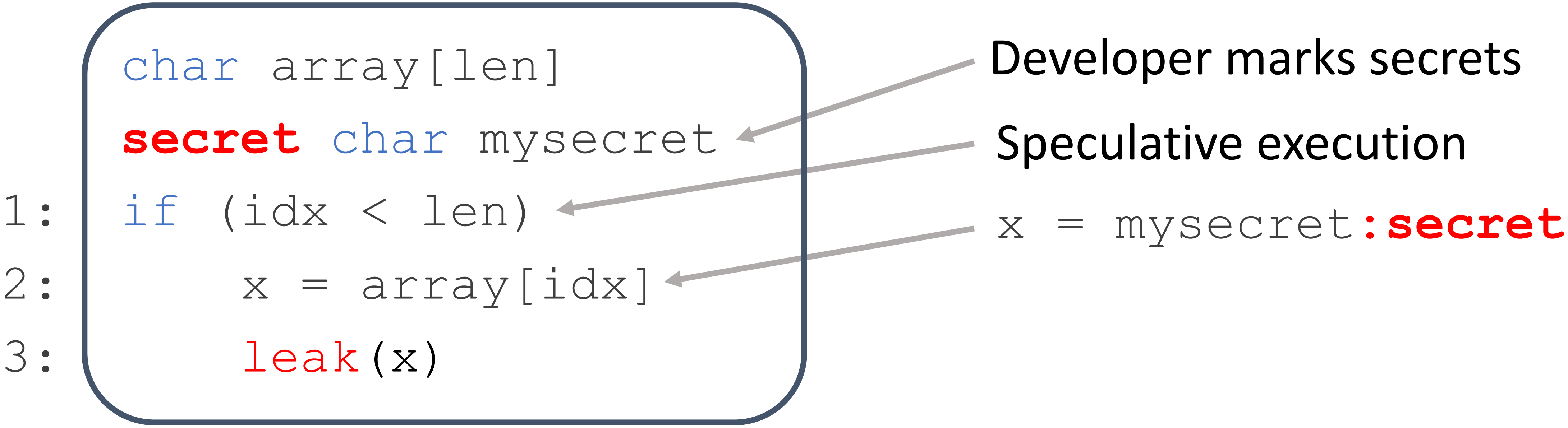
```
char array[len]
secret char mysecret
if (idx < len)
    x = array[idx]
    leak(x)
```

Developer marks secrets

Speculative execution

Consider `idx = len`

Illustration with Spectre-v1



```
char array[len]
secret char mysecret
if (idx < len)
    x = array[idx]
    leak(x)
```

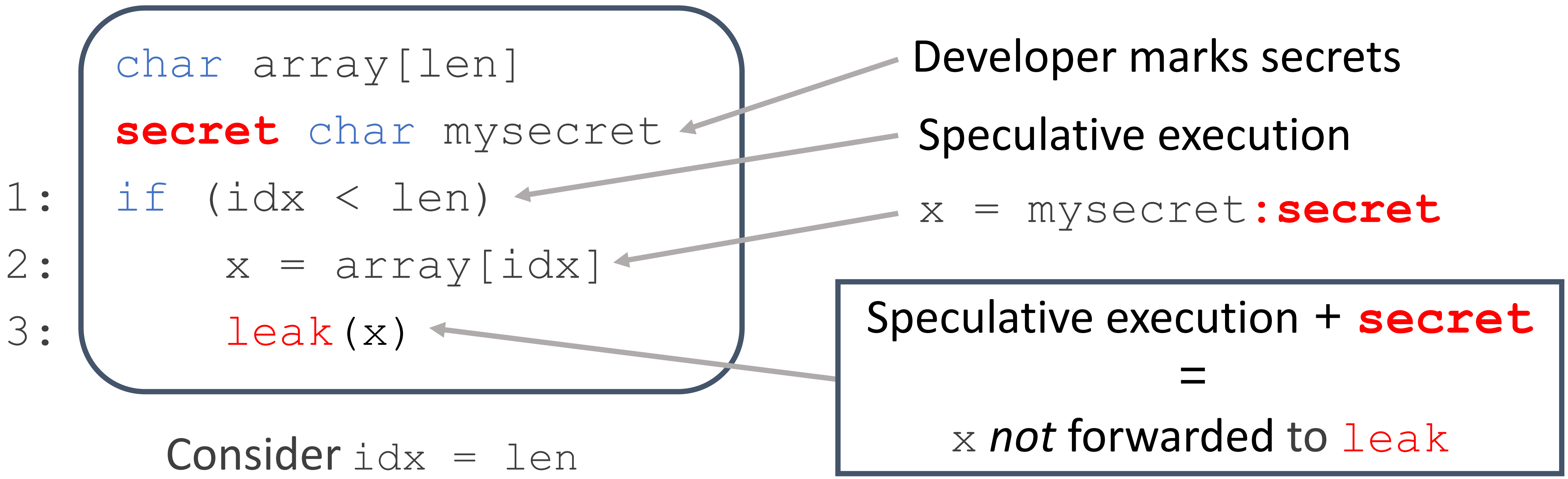
Developer marks secrets

Speculative execution

`x = mysecret:secret`

Consider `idx = len`

Illustration with Spectre-v1



```
    char array[len]
    secret char mysecret
1:  if (idx < len)
2:      x = array[idx]
3:      leak(x)
```

Consider `idx = len`

Developer marks secrets

Speculative execution

`x = mysecret`:**secret**

Speculative execution + **secret**

=

`x` *not* forwarded to `leak`

How do I know that my defense works?

How do I know that my defense works?

Hardware-Software Contracts for Secure Speculation

Marco Guarnieri*, Boris Köpf†, Jan Reineke‡, and Pepe Vila*
**IMDEA Software Institute* †*Microsoft Research* ‡*Saarland University*

Challenges

Adapt HW/SW contract framework to account for

- All existing speculation mechanisms (Spectre, LVI)
- Futuristic speculation mechanisms (value prediction)
- Declassification

Our contributions

- ProSpeCT: Formal processor model with HST
  - Proof: constant-time programs do not leak secrets
  - Allows for declassification
  - Generic: all Spectre variants / LVI

- First to consider (Load) Value Speculation
  - Novel insight: sometimes need to rollback *correct* speculations for security

- Implementation in a RISC-V microarchitecture
  - First synthesizable implementation
  - Evaluation: hardware cost, performance, annotations

ProSpeCT: Generic formal processor model for HST

Semantics of out-of-order speculative processor with HST

- → Abstract microarchitectural context
- → Functions $update, predict, next$

Attacker observations/influence

All public values are leaked / influence predictions

Generic/Powerful predictors

Declassify = write secrets to public memory

- → Beware unintentional declassification

ProSpeCT: Generic formal processor model for HST

Semantics of out-of-order speculative processor with HST

- → Abstract microarchitectural context
- → Functions $update, predict, next$

Attacker observations/influence

All public values are leaked / influence predictions

Generic/Powerful predictors

Declassify = write secrets to public memory

- → Beware unintentional declassification

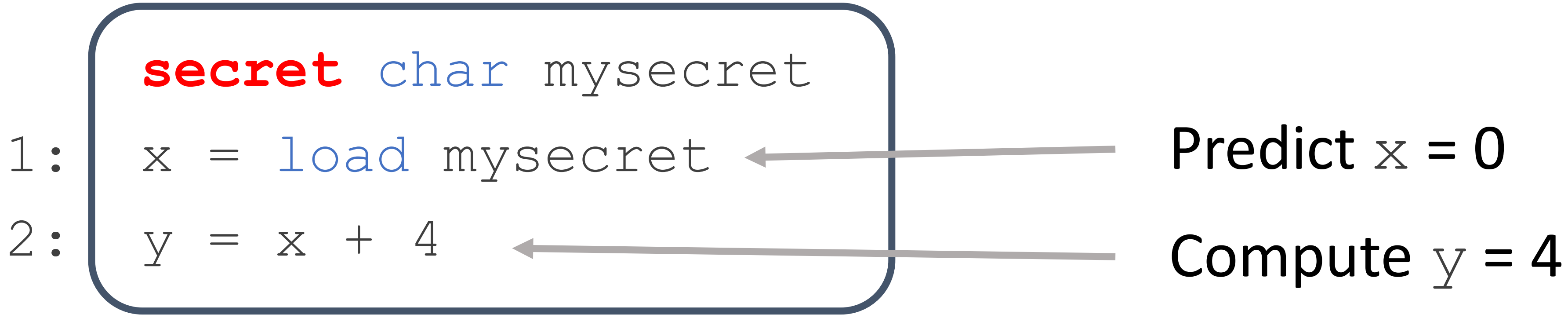
Security proof

Constant-time programs (ISA semantics)
do not leak secrets (micro-arch. semantics)

Load Prediction: Rollback correct executions?

```
secret char mysecret
x = load mysecret
y = x + 4
```

Load Prediction: Rollback correct executions?

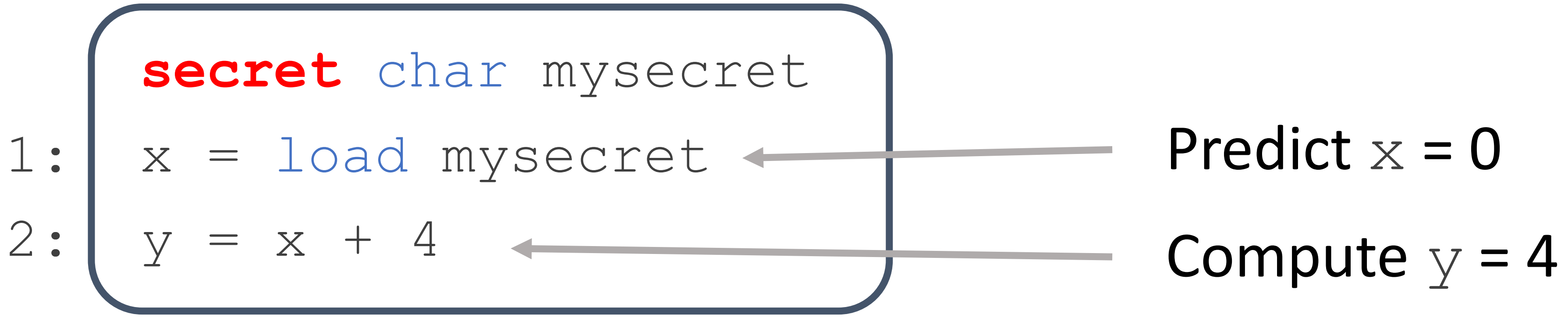


```
secret char mysecret
x = load mysecret
y = x + 4
```

Predict `x` = 0

Compute `y` = 4

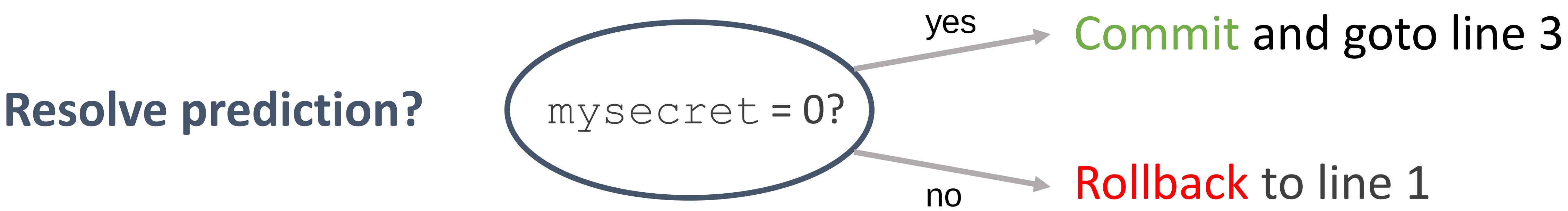
Load Prediction: Rollback correct executions?



```
    secret char mysecret
1:  x = load mysecret
2:  y = x + 4
```

Predict x = 0

Compute y = 4



Resolve prediction?

mysecret = 0?

yes

Commit and goto line 3

no

Rollback to line 1

Implicit resolution-based channel!

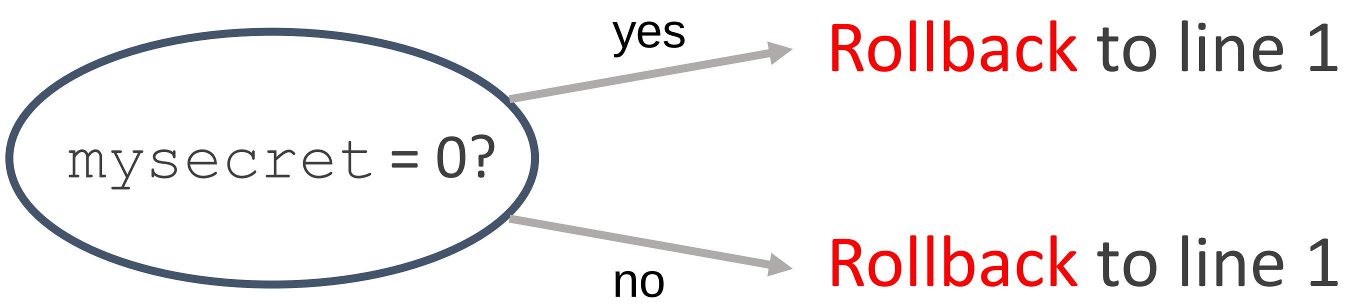
Load Prediction: Rollback correct executions?

```
    secret char mysecret
1:  x = load mysecret
2:  y = x + 4
```

Predict `x` = 0

Compute `y` = 4

Resolve prediction?



`mysecret` = 0?

yes → Rollback to line 1

no → Rollback to line 1

Solution: always rollback when value is secret

Implementation

Prototype RISC-V implementation

On top of Proteus modular RISC-V processor

- Branch target prediction
- Conservative approach
- 2 secret regions defined by CSRs

Limited Hardware Cost

- LUTs: +17%
- Registers: +6%
- Critical path: +2%

Evaluation

4 primitives (HACL*)

- Annotate secret
- Ensure no secrets spilled
- Stack public in 3/4 cases
- ≤1h / primitive

Performance overhead (benchmark from [1])

Speculation/Crypto	25/75	50/50	75/25	90/10
Precise (Key)	**0%**	**0%**	**0%**	**0%**
Conservative (All)	10%	25%	36%	45%

No overhead in software for constant-time code when secrets are precisely annotated

[1] Jacob Fustos, Farzad Farshchi, and Heechul Yun. "SpectreGuard: An Efficient Data-Centric Defense Mechanism against Spectre Attacks". In: DAC. 2019

Conclusion

Software informs hardware about secrets

Strong security guarantees

End-to-end security for constant-time programs

Low overhead

No software overhead for constant-time code

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

github.com/proteus-core/prospect

Icons made by **Freepik**, **Vectors Market, monkik** from www.flaticon.com

A step back

RISC-V open standard ISA

→ HW-SW co-design for security

- Proteus: extensible RISC-V processor
- Security extensions
  - ProSpeCT
  - ISA extension for CF balancing/linearization
  - CHERI
  - …

Future work

How to ease adoption of HW-SW co-designs?
→ Need infrastructure around HW-SW contracts

- Secure compilation/compiler support (LLVM, Jasmin?)
- Binary analysis (Binsec/angr)
- Validate HW implementation (fuzzing, verification)

⟹ Márton Bognár

Other relevant projects at KU LEUVEN DistriNet

- Attacks/Defenses for TEEs ⟹ Jo Van Bulck
- Formalization/verification of ISA security guarantees ⟹ Dominique Devriese