UNIHAVEGE

De l’ «UNIversalité » du générateur d’aléa irreproductible HAVEGE

projet ACI sécurité nov 2003 – nov 2006

André Seznec
Caps Team
IRISA/INRIA

Unpredictable random numbers

- Unpredictable = irreproducible + uniformly distributed
  - ➔ no bias
  - ➔ no way to replay the sequence

- Needs for cryptographic purpose:
  - ➔ key generation, paddings, zero-knowledge protocols, ..
  - ➔ secure deletion

André Seznec
Caps Team
Irisa

Unpredictable random numbers: where are they needed?

- Ideally, on every computing appliance that needs to store data and/or communicate:
  - Servers
  - PCs
  - PDAs
  - Intelligent cell phones
  - Smart cards
  - Network routers
  - ..

André Seznec
Caps Team
Irisa

Current solutions

- **hardware: exploiting some non deterministic physical process**
  - ➔ **10-100 Kbits/s**
  - ➔ **not cost-effective for 50 € appliances**
- **software: exploiting the occurences of *(pseudo)* non deterministic external events**
  - ➔ **known as *software entropy gathering***
    - **10-100 bits/s on PCs and workstations**
    - **less on PDAs**

André Seznec
Caps Team
Irisa

HAVEGE (2002)

HArdware Volatile Entropy Gathering and Expansion

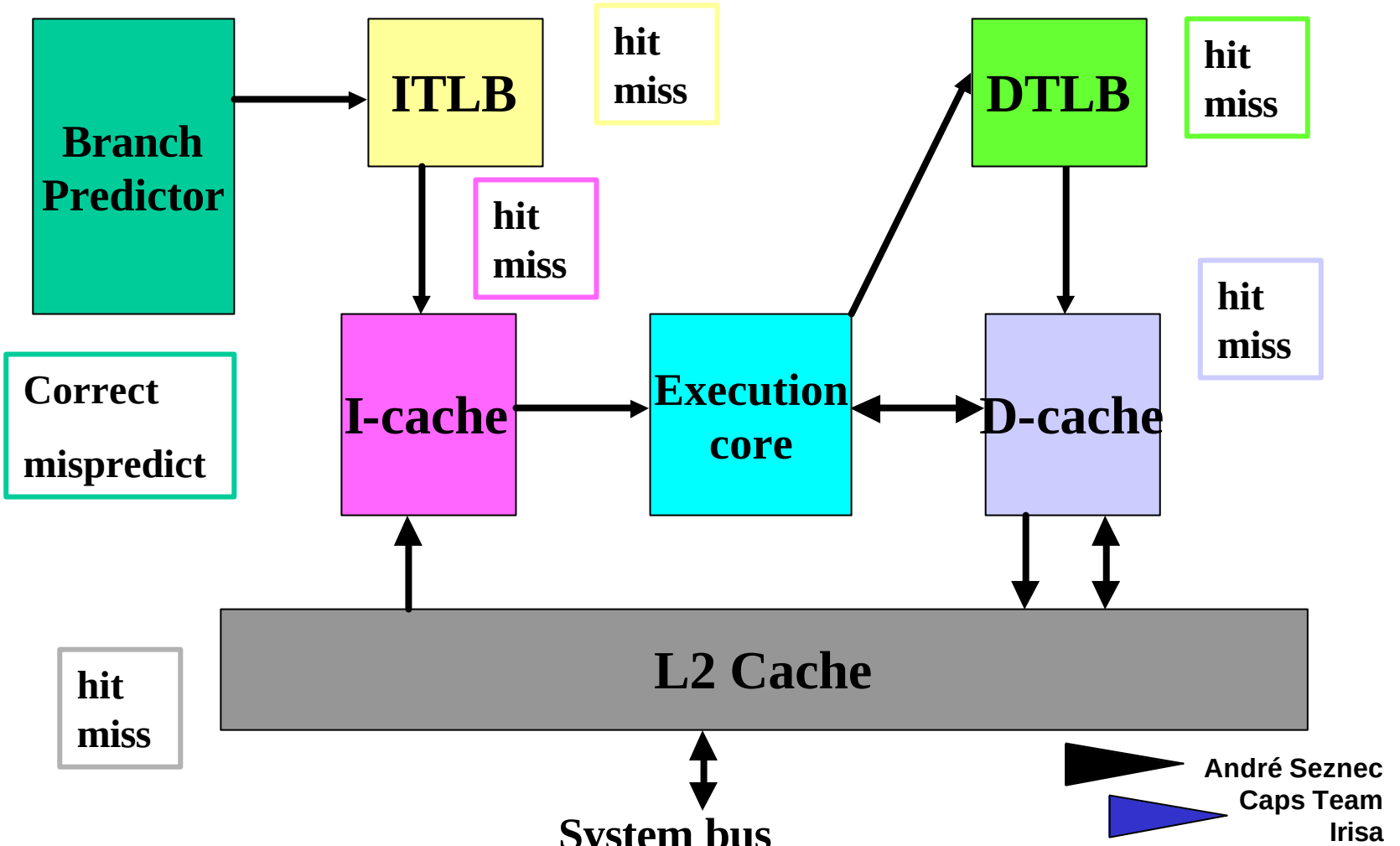
Thousands of hardware states for performance improvement in modern processors

These states are touched by all external events

a good source of entropy/uncertainty !

André Seznec
Caps Team
Irisa

Execution time of a short instruction sequence is a complex function !



Gathering hardware volatile entropy/uncertainty ?

Collecting the complete hardware state of a processor:

- **requires freezing the clock**
- not accessible on off-the-shelf PCs or workstations

Indirect access through timing:

- use of **the hardware clock counter** *at a very low granularity*
- Heisenberg ’s criteria:
  indirect access to a particular state (e.g. status of a branch predictor entry) modifies many others

André Seznec
Caps Team
Irisa

But a processor is built to be deterministic !?!

Yes but:

- Not the response time !
- External events: peripherals , IOs
- Operating System
- Fault tolerance

OS interruptions and some volatile hardware states

Solaris on an UltraSparc II (non loaded machine)

- **L1 data cache: 80-200 blocks displaced**
- **L1 instruction cache: around 250 blocks displaced**
- **L2 cache: 850-950 blocks displaced**
- **data TLB: 16-52 entries displaced**
- **instruction TLB: 6 entries displaced**

Thousands of modified hardware states

- **+ that ’s a minimum**
- **+ distribution is erratic**

Similar for other OS and other processors

André Seznec
Caps Team
Irisa

HAVEGE= hardware entropy gathering + pseudo-random number generation

Embed an hardware entropy gathering algorithm in a pseudo-random number generator

A very simple PRNG:

-two concurrent walks in a table

-random number is the exclusive-OR of the two read data

But the table is continuously modified using the hardware clock counter

André Seznec
Caps Team
Irisa

Code is designed to exercise the hardware states (*just an example)*

```
if (pt & 0x4000){ PT2 = PT2 ^ 1;}
if (pt & 0x8000){ PT2 = PT2 + 7;}

PT=pt & 0x1fff; pt= Walk[PT];
PT2=Walk[(PT2 & 0xfff) ^
          ((PT ^ 0x1000) & 0x1000)];

RESULT[i] ^ =  PT2 ^ pt  ; i++;

T=((T<< 11) ^ (T>> 21)) + HardClock();
pt = pt ^ T; Walk[PT]= pt;
```

Tests to exercise the branch predictor

The two concurrent walks

Output generation

Entropy gathering and table update

Read the hardware clock counter

André Seznec
Caps Team
Irisa

HAVEGE internal state

The usual memory state of any PRNG

+

Internal volatile hardware states:

branch predictor

I-cache

data cache

data TLB

miscelleanous, ..

On a Solaris UltraSparcII

(2406) * (2**304) states**

7256 states**

7512 states**

128!/64! States

..

André Seznec
Caps Team
Irisa

Maintaining unpredictable *hidden* volatile states

```
if (pt & 0x4000){ PT2 = PT2 ^ 1;}
if (pt & 0x8000){ PT2 = PT2 + 7;}

PT=pt & 0x1fff; pt= Walk[PT];
PT2=Walk[(PT2 & 0xfff) ^
          ((PT ^ 0x1000) & 0x1000)];

RESULT[i] ^ =  PT2 ^ pt  ; i++;

T=((T<< 11) ^ (T>> 21)) + HardClock();
pt = pt ^ T; Walk[PT]= pt;
```

Taken or not-taken with p = 1/2

Hit/miss on the L1 cache with p = 1/2

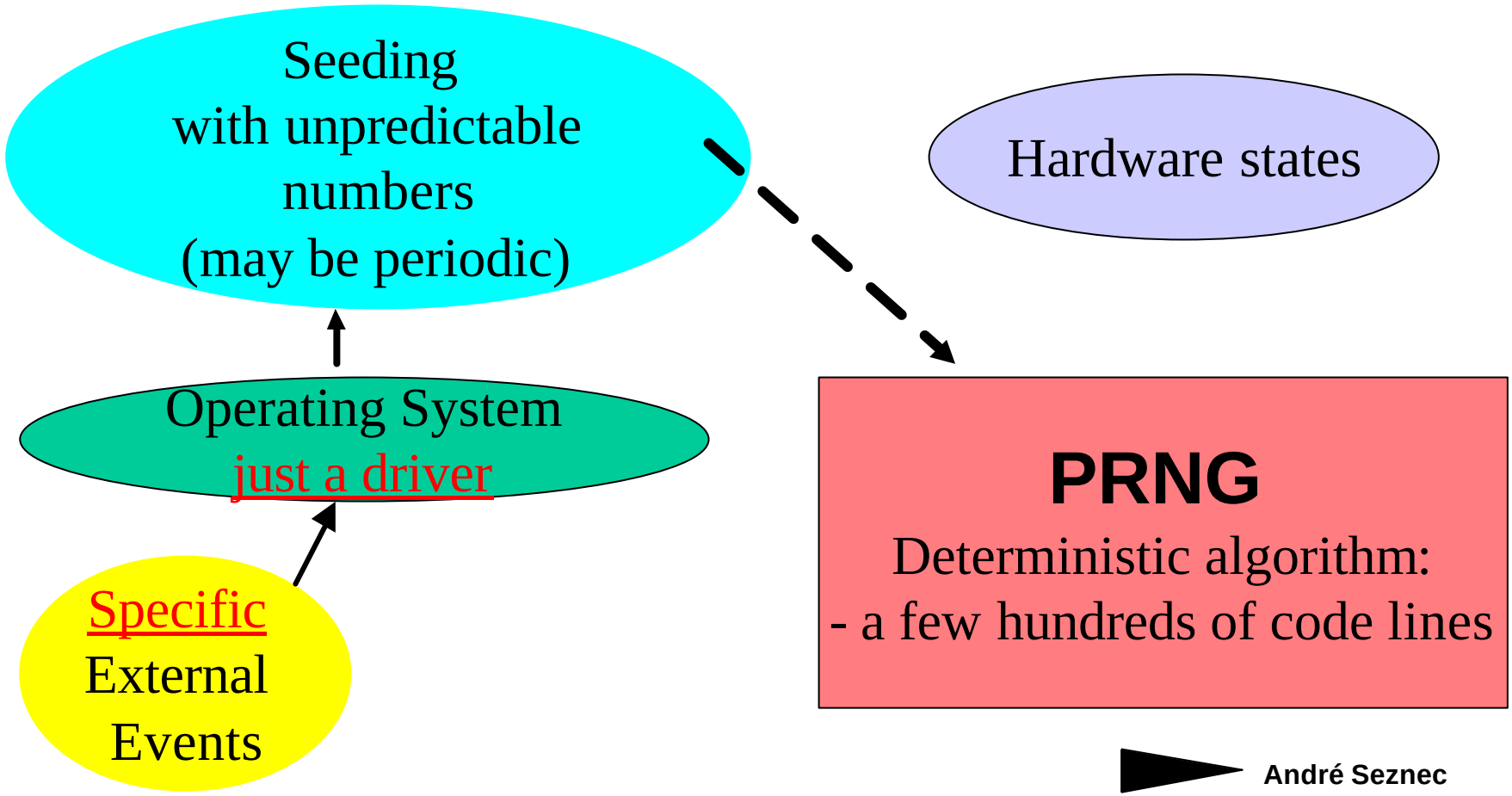
André Seznec
Caps Team
Irisa

HAVEGE continuous reseeding

- On each OS interrupt:
  - ➔ internal state of the generator is modified
    - thousands of binary states are touched
  - ➔ complex interaction between internal general state and OS servicing:
    - service time of an OS interrupt depends on the initial hardware state
- Any event on the memory system touches the state
  - ➔ asynchronous events on the memory bus !

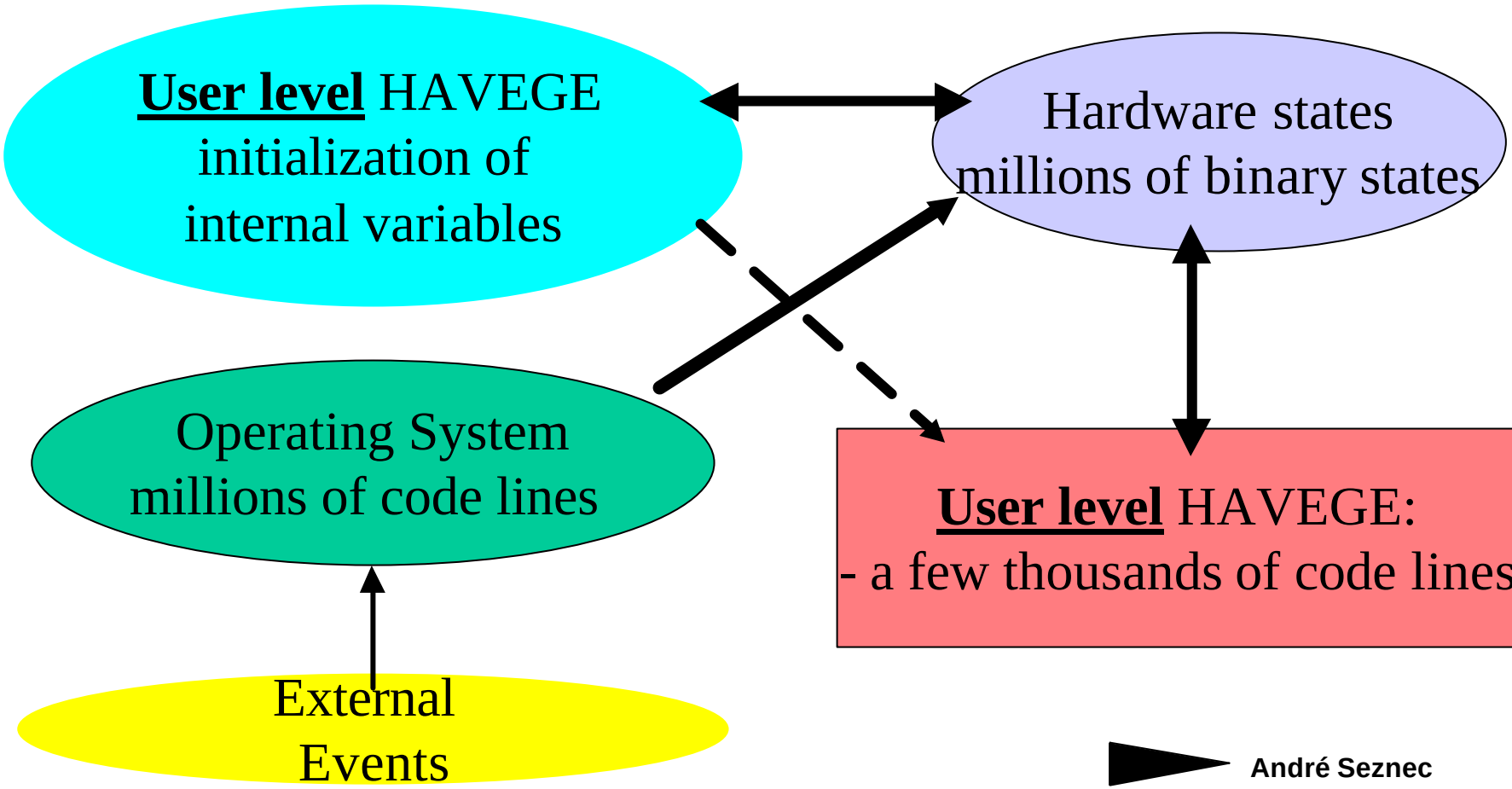
André Seznec
Caps Team
Irisa

Entropy Gathering + PRNG



André Seznec
Caps Team
Irisa

HAVEGE



André Seznec
Caps Team
Irisa

HAVEGE status

- Distributed since january 2002 (non-commercial usage)
  - ➔ http://www.irisa.fr/caps/projects/hipsor/HAVEGE.html
- Platforms:
  - ➔ UltraSparc II and III, Solaris
  - ➔ Pentium III, Pentium 4, Athlon - Windows, Linux
  - ➔ Itanium, Linux
  - ➔ PowerPC G4, MacOS 10
- more than 100 Mbits/s

UNIHAVEGE objective

Provide evidence that,

any *« reasonably complex »* modern computing appliance can be its own source of unpredictable random number

« reasonably complex »:

features caches, branch predictors, ..

uses some operating system

André Seznec
Caps Team
Irisa

UNIHAVEGE partners

- CAPS team, IRISA/INRIA
  - ➔ A. Seznec, J.C Hernandez (postdoc till 03-2004)
- CODES team
  - ➔ N. Sendrier, C. Lauradoux (Ph D. student)
- Ponctually: V. Issarny, I. Puaut, FT R & D (LASR)

UNIHAVEGE actions

- Cryptanalysis and improvement on HAVEGE
  - ➔ Intensive testing on HAVEGE (degraded forms)
  - ➔ Passive and active attacks
  - ➔ Stronger HAVEGE versions
- New platforms for HAVEGE
  - ➔ PDAs, cell phone
- Towards new applications for unpredictable random number
  - ➔ Unprecedented and cheap throughput should allow new usages

André Seznec
Caps Team
Irisa