Covert Channels Through Branch Predictors: a Feasibility Study

Dmitry Evtyushkin[1]

Dmitry Ponomarev[1] Nael Abu-Ghazaleh[2]

[1]*State University of New York at Binghamton*
Department of Computer Science

[2]*University of California at Riverside*
Department of Computer Science & Engineering

Hardware and Architectural Support for Security and Privacy (HASP)

June 14, 2015
Portland, OR, USA
in conjunction with ISCA 2015

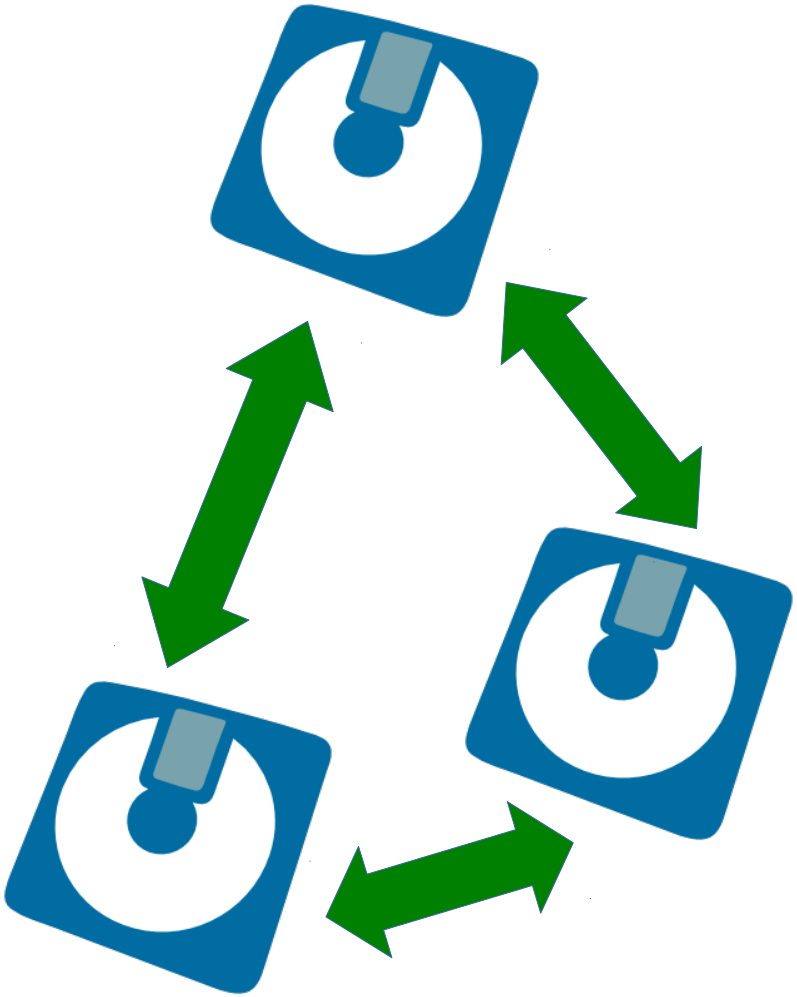
BINGHAMTON UNIVERSITY
STATE UNIVERSITY OF NEW YORK

UC RIVERSIDE
UNIVERSITY OF CALIFORNIA

Presentation Outline

- *What is covert channel? How can it be used?*
- *Architectural covert channels*
- *Branch Predictor in CPU*
- *Constructing covert channel through Branch Predictor*
- *Results*
- *Optimizations*
- *Conclusion*

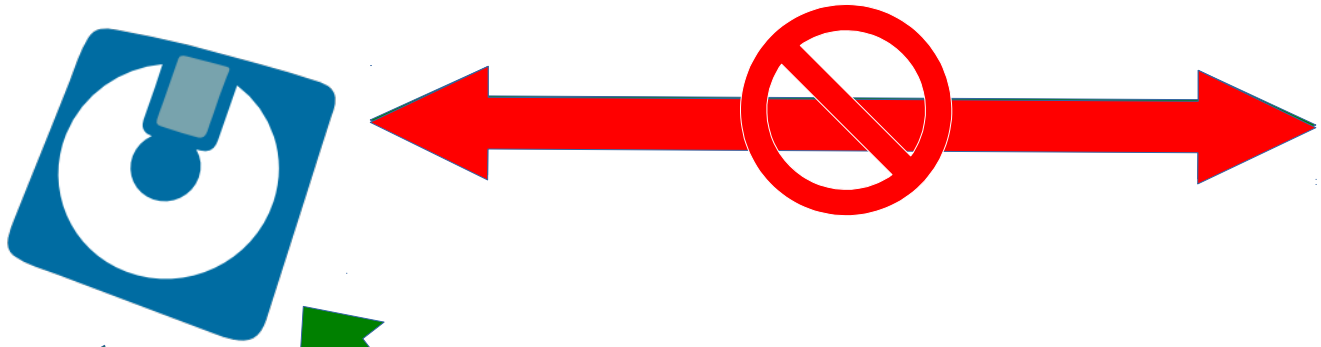
Covert Channels



Data Sharing Mechanisms:

- *Shared Memory*
- *Networking*
- *File system*
- *IPC*
- *etc...*

Covert Channels



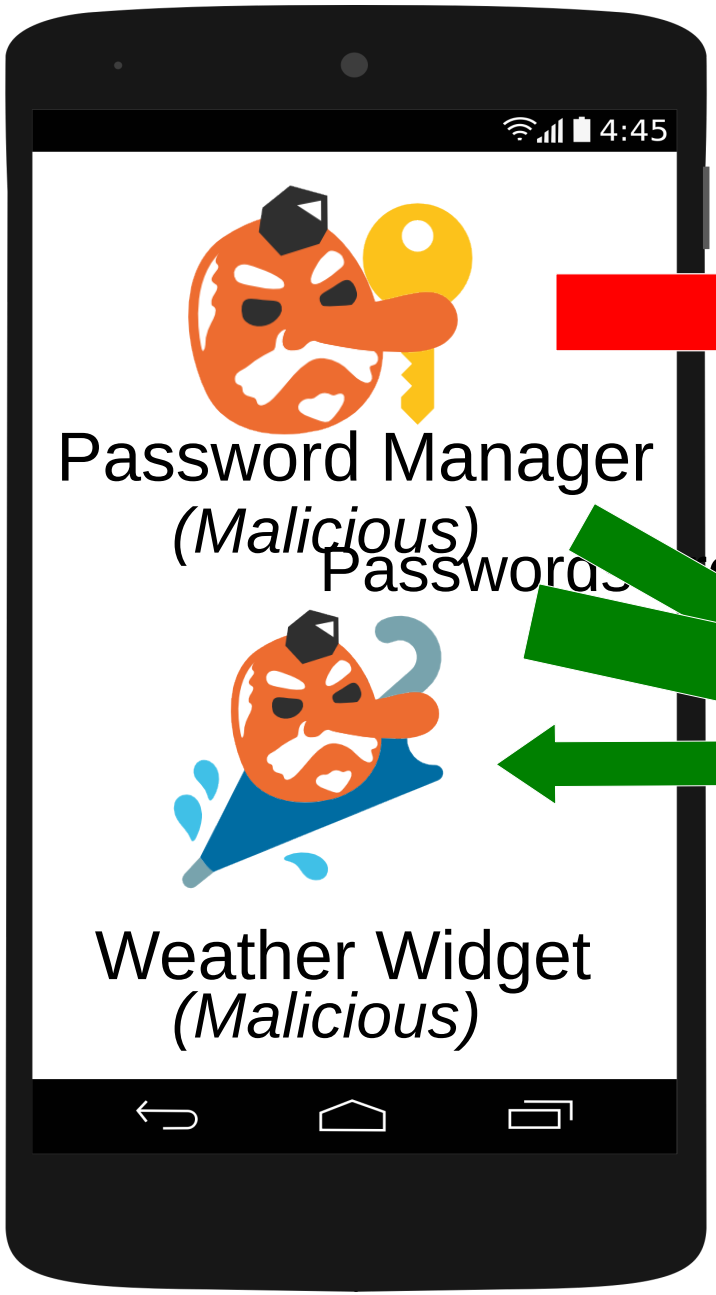
Sharing is not desirable:

- *Violates security policy*
  - *Information Flow Control (IFC) schemes*
    - *TaintDroid (OSDI '10)*
  - *Applications in different* ***Security Domains***

Covert Channels:

Transfer information through channels not intended for information transfer

Covert Channel Usage Example



Application is isolated

or password data is tainted

Are passwords secure?

No IPC

No File System Access

No Network Access

Passwords are secure as long as OS enforces **security poli**

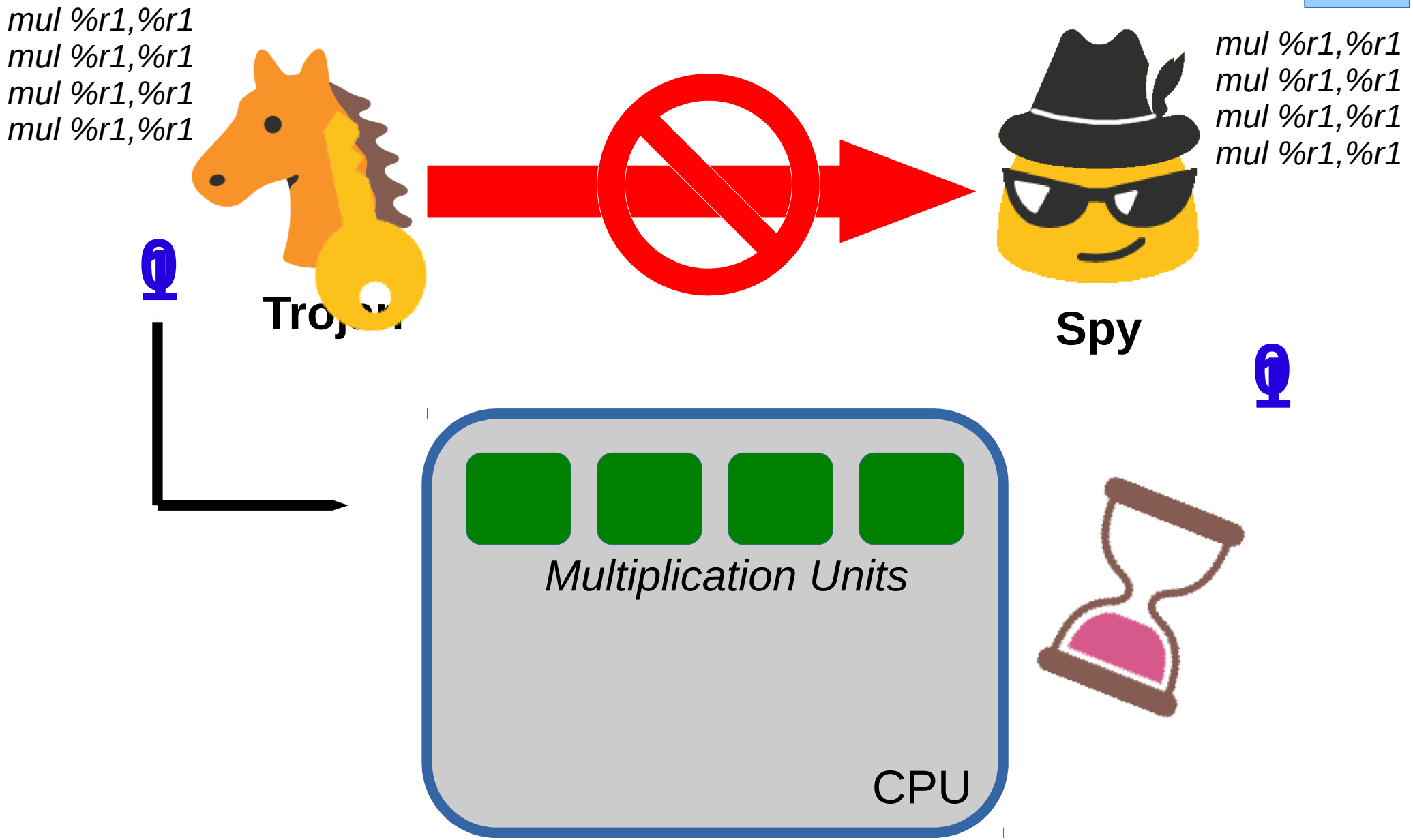
Network Access

Covert Channel Usage Example

Shared resources can be used to construct covert channels:

- *OS resources*
  - *File descriptors, free space, etc.*
- *Network latencies*
- *Power and Thermal effects*
- ***Architectural resources***
  - *Caches*
  - *Functional units*
  - ***Branch predictor***
  - *Other resources*
- *Others*

Contention-based Architectural Covert Channels

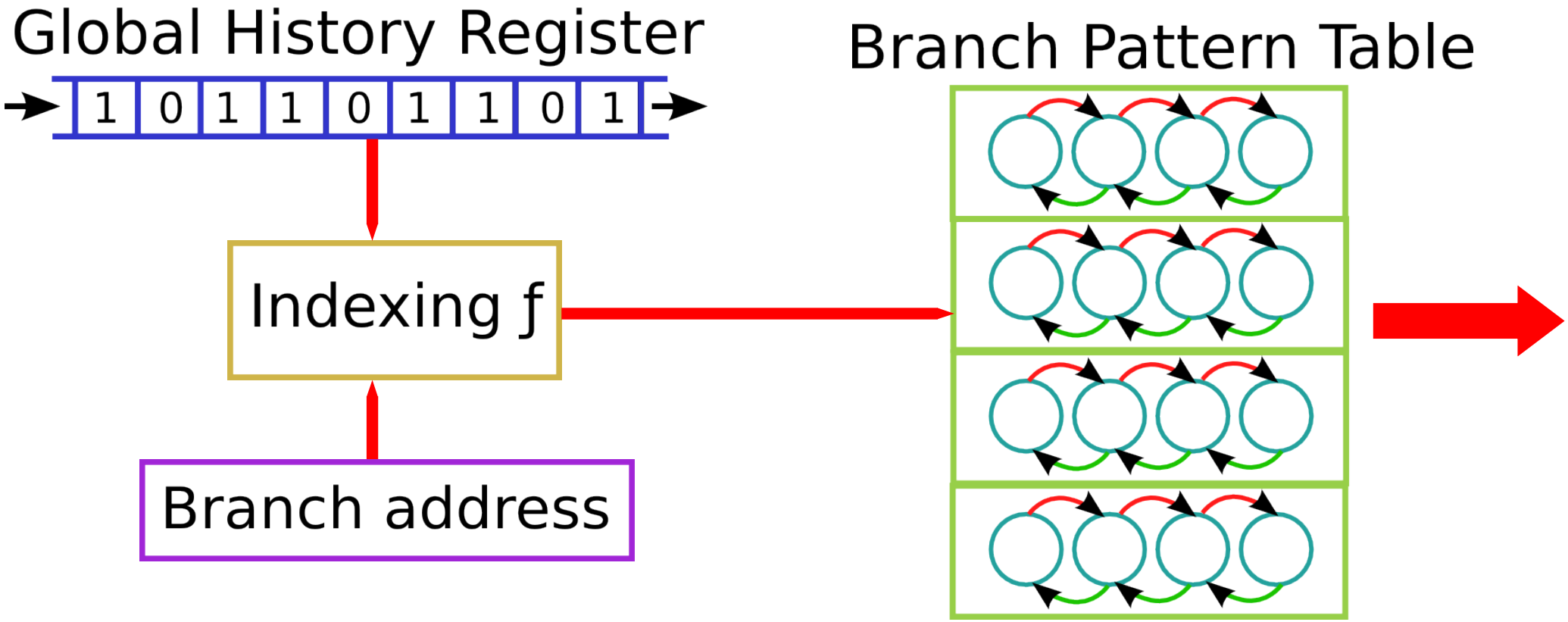


Branch Prediction Unit

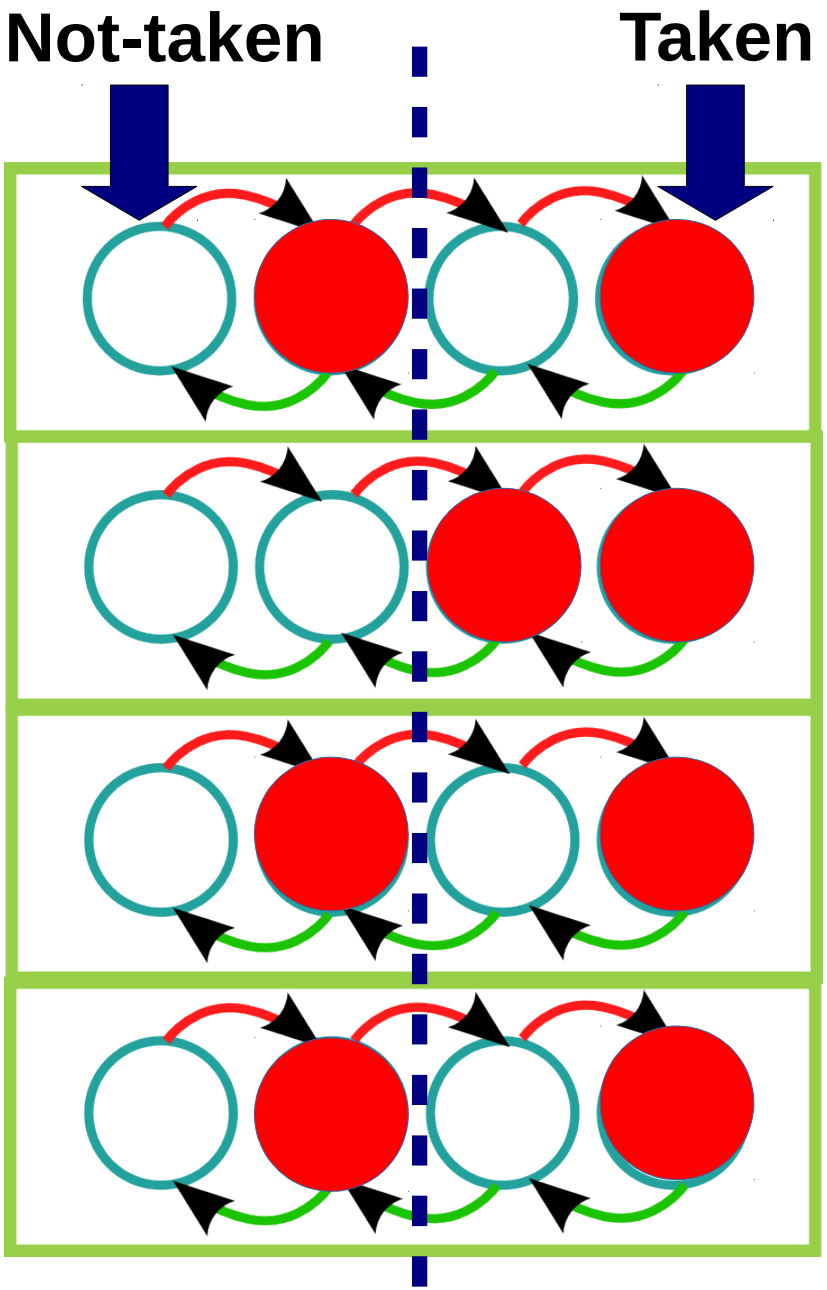
Properties that make covert channel possible:

- *During execution branch predictor accumulates state*
- *BP is shared among all processes on core*
- *BP is not flushed on context switches*
- *Parallel threads in SMT share same BP*
- *Branch mispredictions have high cost*

Branch Predictor Operation



Constructing Covert Channel




```
while(1)
   if (time(0)%2){
     taken();
   }else{
     nottaken();
}
```

```
for (int i=0; i <
   MAX_PROBES;i++){
   usleep(SLEEP_T);
   start_t=rdtsc();
   taken();
   end_t=rdtsc();
}
```

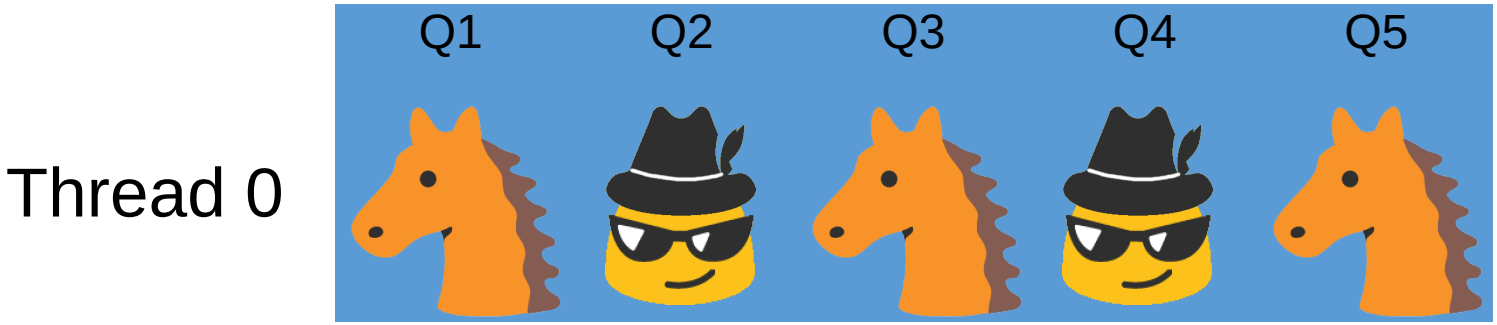
1

Scheduling Trojan and Spy

Single Threaded Scenario:

Setting affinity mask to run Trojan and Spy on the same core

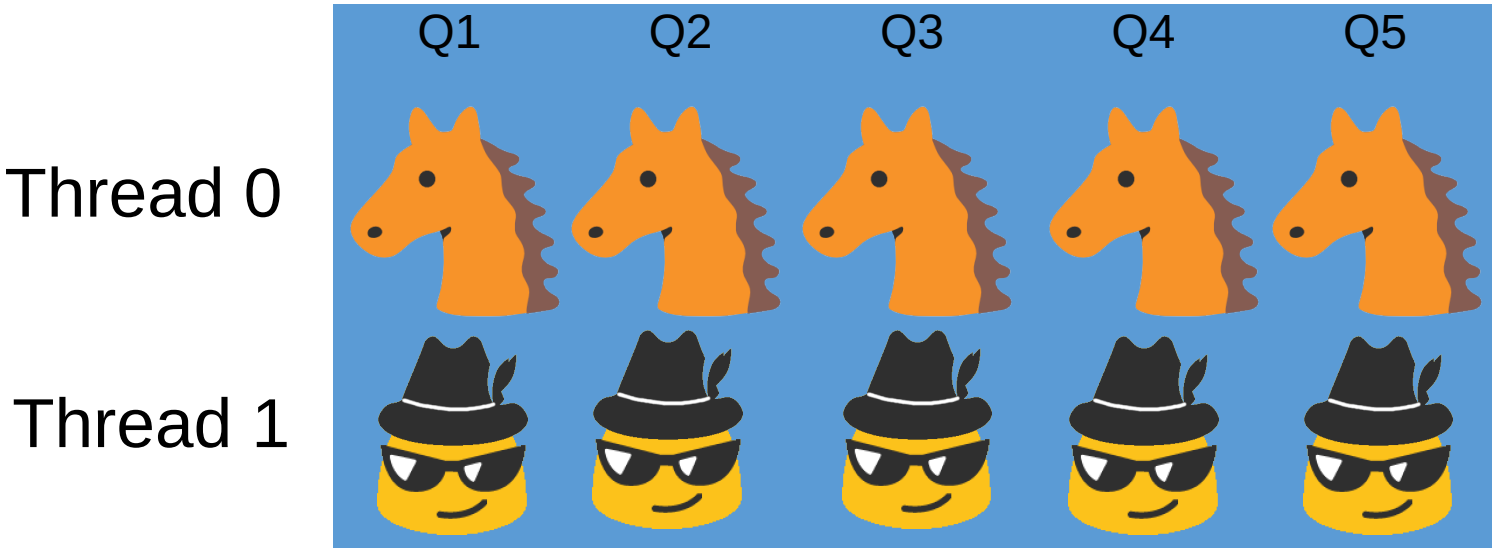
Quanta



Multi Threaded Scenario (SMT):

Setting affinity mask to run Trojan and Spy on parallel virtual cores

Quanta



Branch Code


```
br_taken:
push   %rbp
movl   $0x1,-0x8(%rbp)
cmpl   $0x0,-0x8(%rbp)
jne .L2
nop
nop
.L2:
cmpl   $0x0,-0x8(%rbp)
jne .L3
nop
nop
nop
.L3:
.....................
```

Notes:

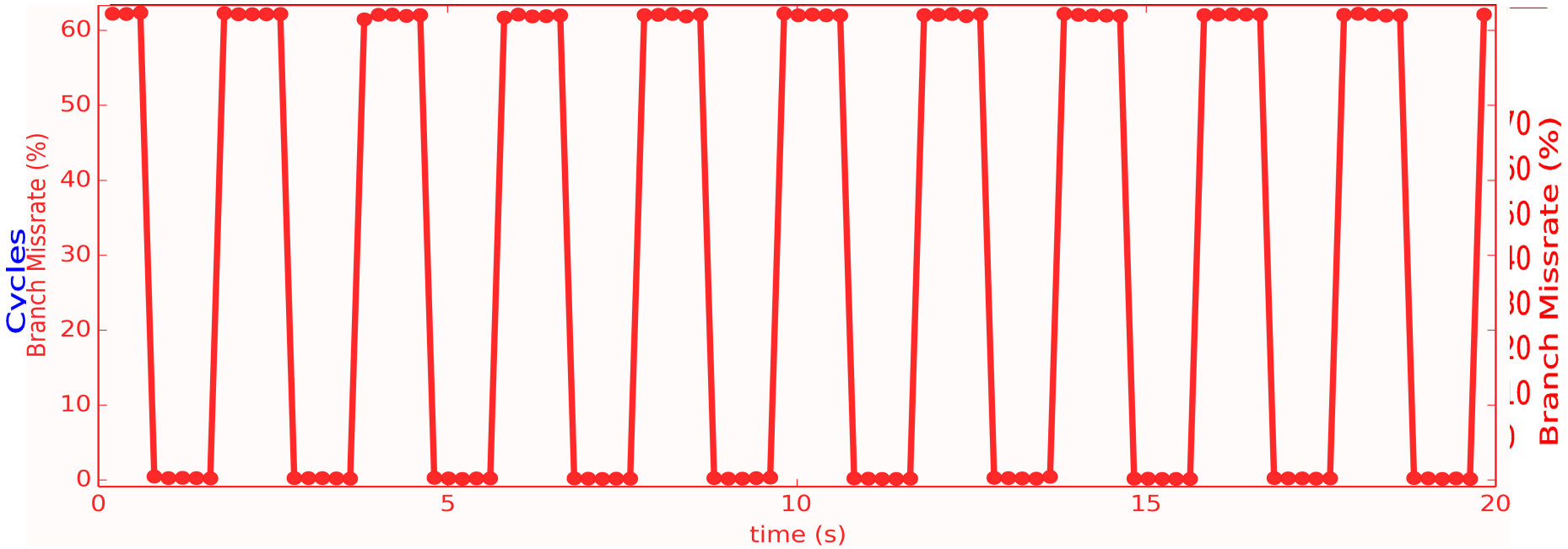
- *Trojan and Spy execute large number of branch instructions*
  - *500K for Trojan*
  - *30K for Spy*
- *Trojan and Spy execute the similar code*
- *Nop instructions to randomize layout*

Evaluation Platform

- Real hardware
- Intel Core i7-4800MQ CPU (Haswell)
- Ubuntu 14.04.2 generic Linux kernel version 3.16.0-31

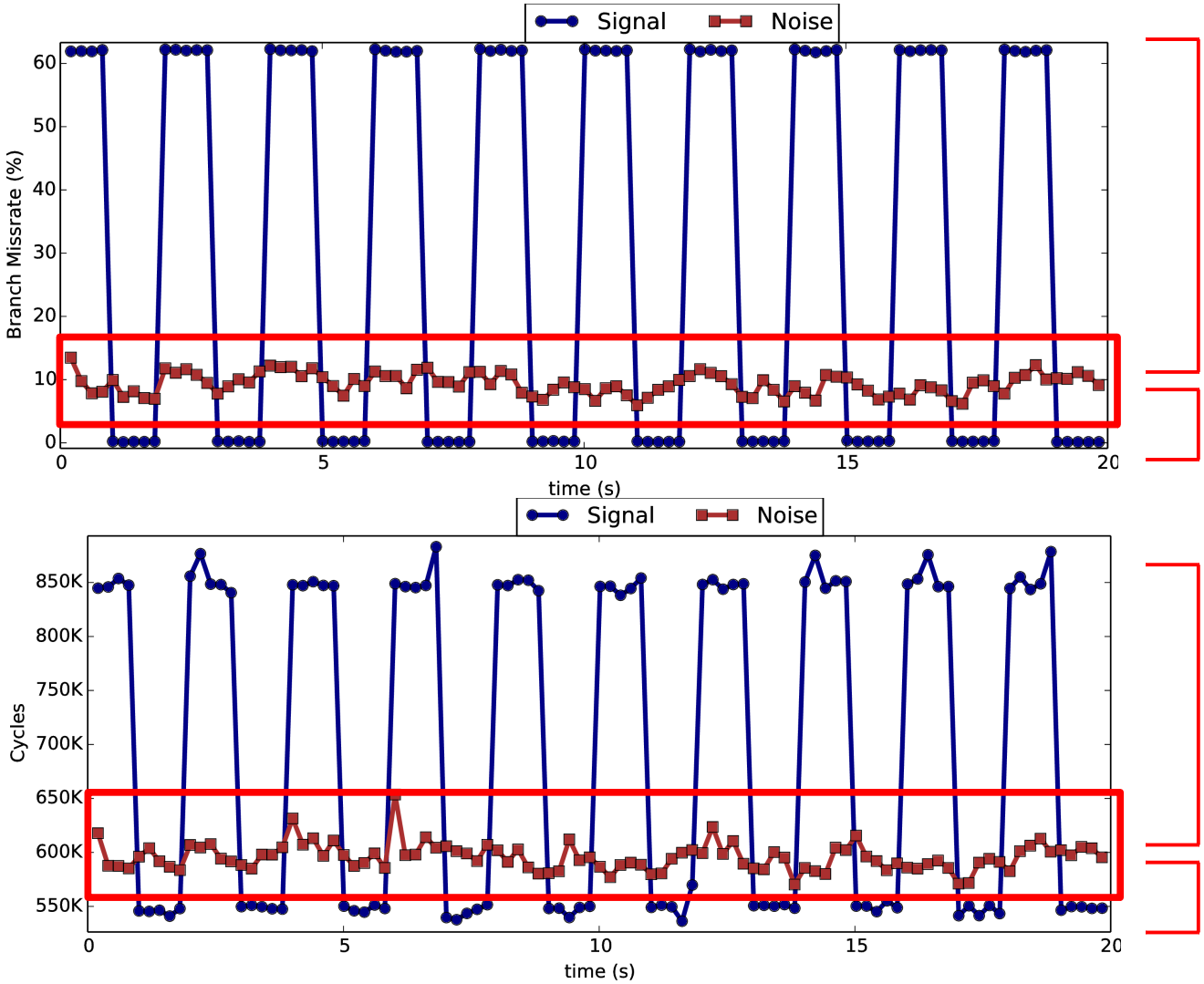
Results: Single Thread

Single Thread, no interference:



Signal vs Noise

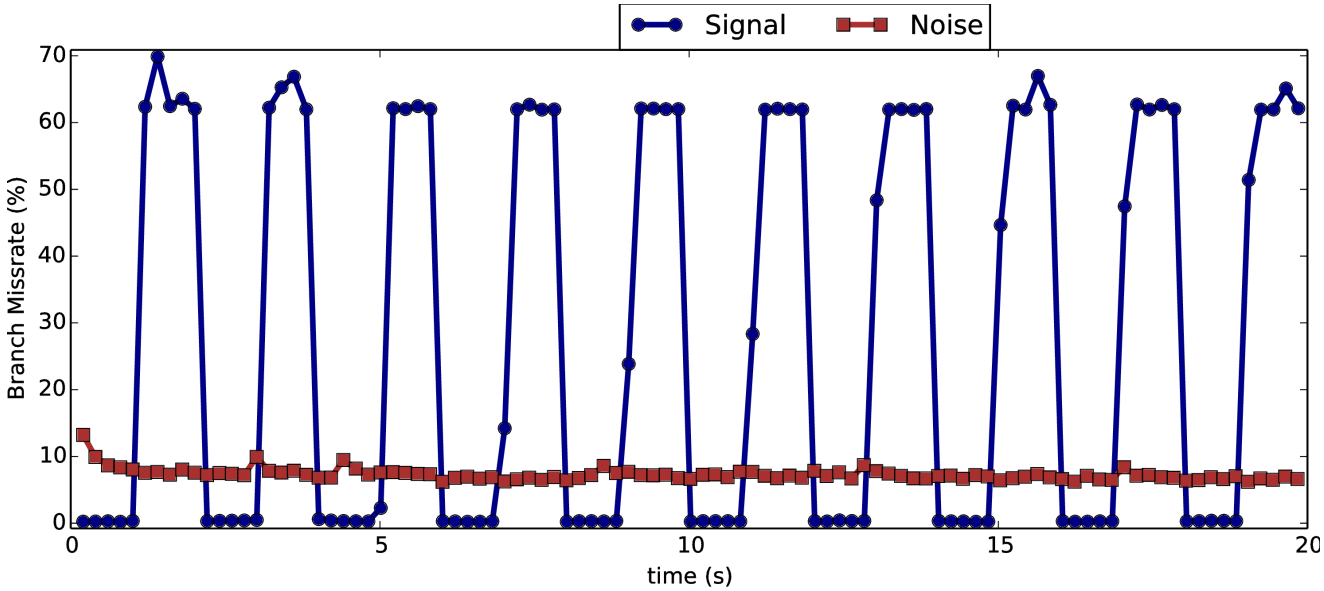
Single Thread, no interference, *cpuburn* as noise:

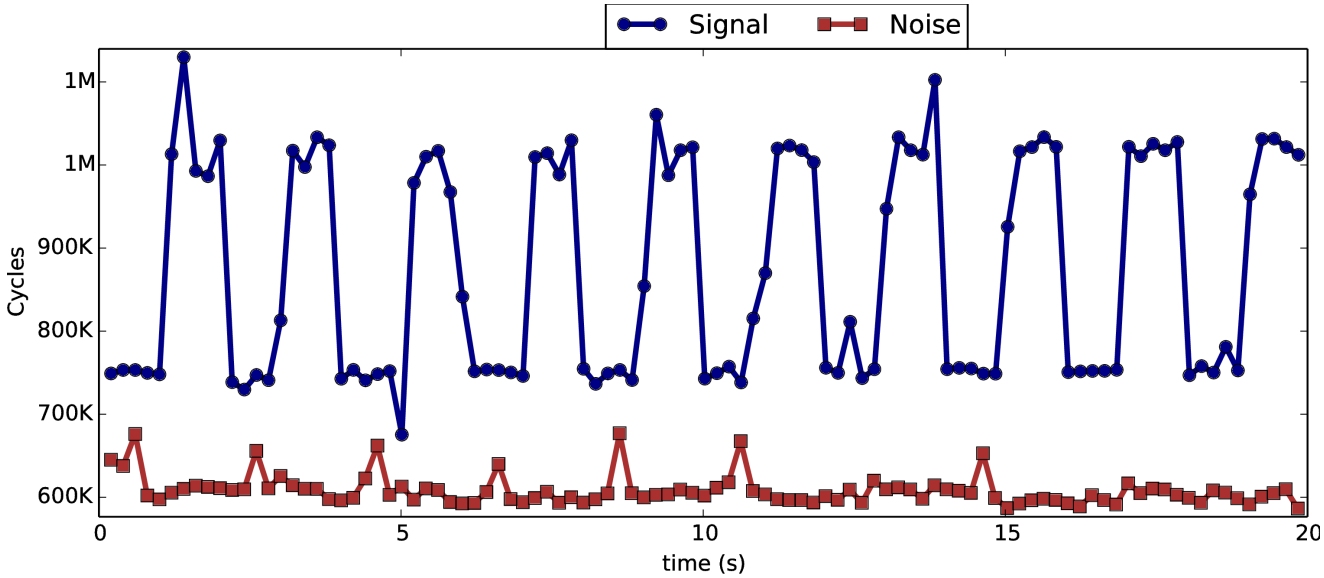


- High SNR
- Stable signal
- Possibly high capacity
- Asynchronous signal

SMT Results

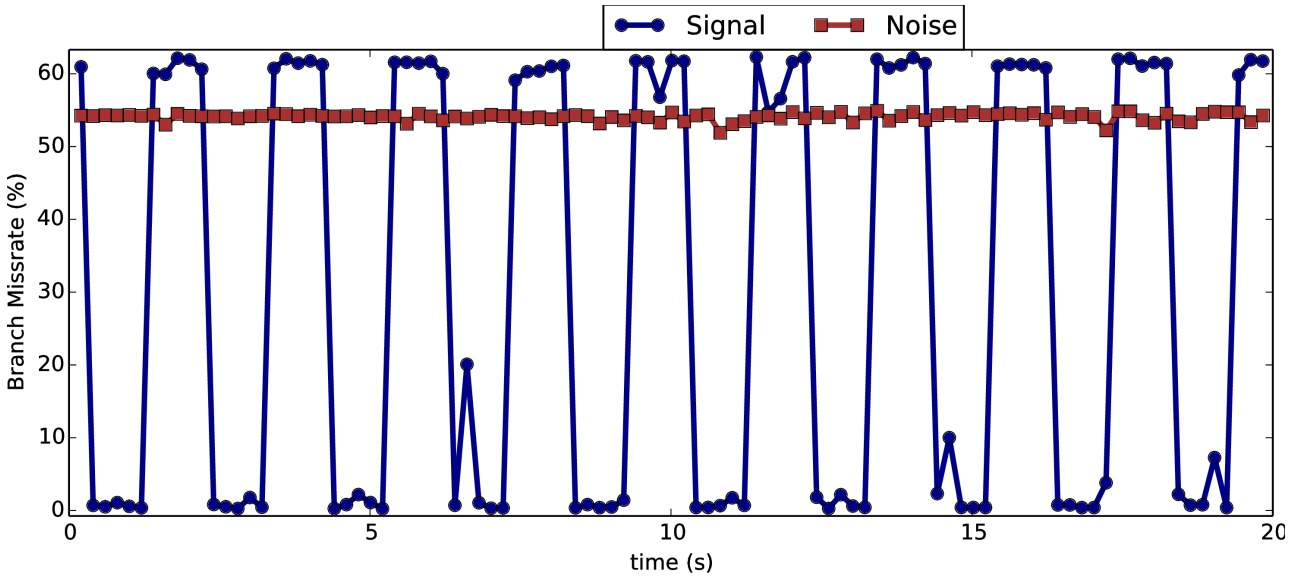
Multi Thread, no interference, *cpuburn* as noise:

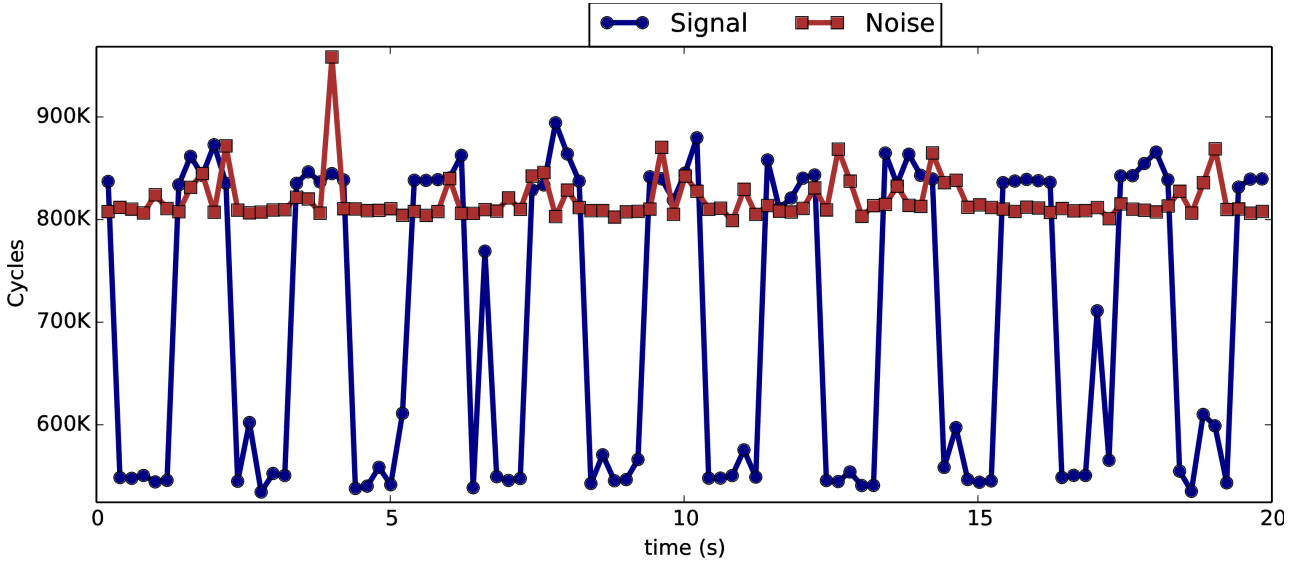




Realistic example

Single Thread, Browser with YouTube as noise/interference





Possible optimizations

- *What we present is only a **prototype***
- *Can transmit multiple bits at once*
- *Can make scheduling faster*
- *For more accuracy, can reverse-engineer BP to find optimal number of branch instructions/nops*

Conclusions

- *Branch predictor can be used as an effective covert channel media*
- *Covert channel through BP has desirable properties*
- *It should be considered when implementing secure systems free of covert and side channels*

Thank you!