TWO CHALLENGES OF STEALTHY HYPERVISORS DETECTION: TIME CHEATING & DATA FLUCTUATIONS

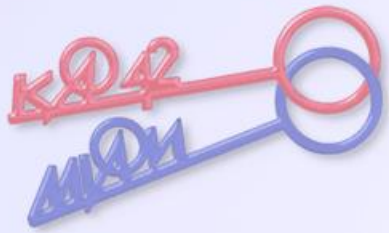
Igor Korkin

CDFSL 2015

Agenda

- Hypervisor (or HYP) as a security threat
- Ways of HYPs detection & their drawbacks
- Time-based detection methods

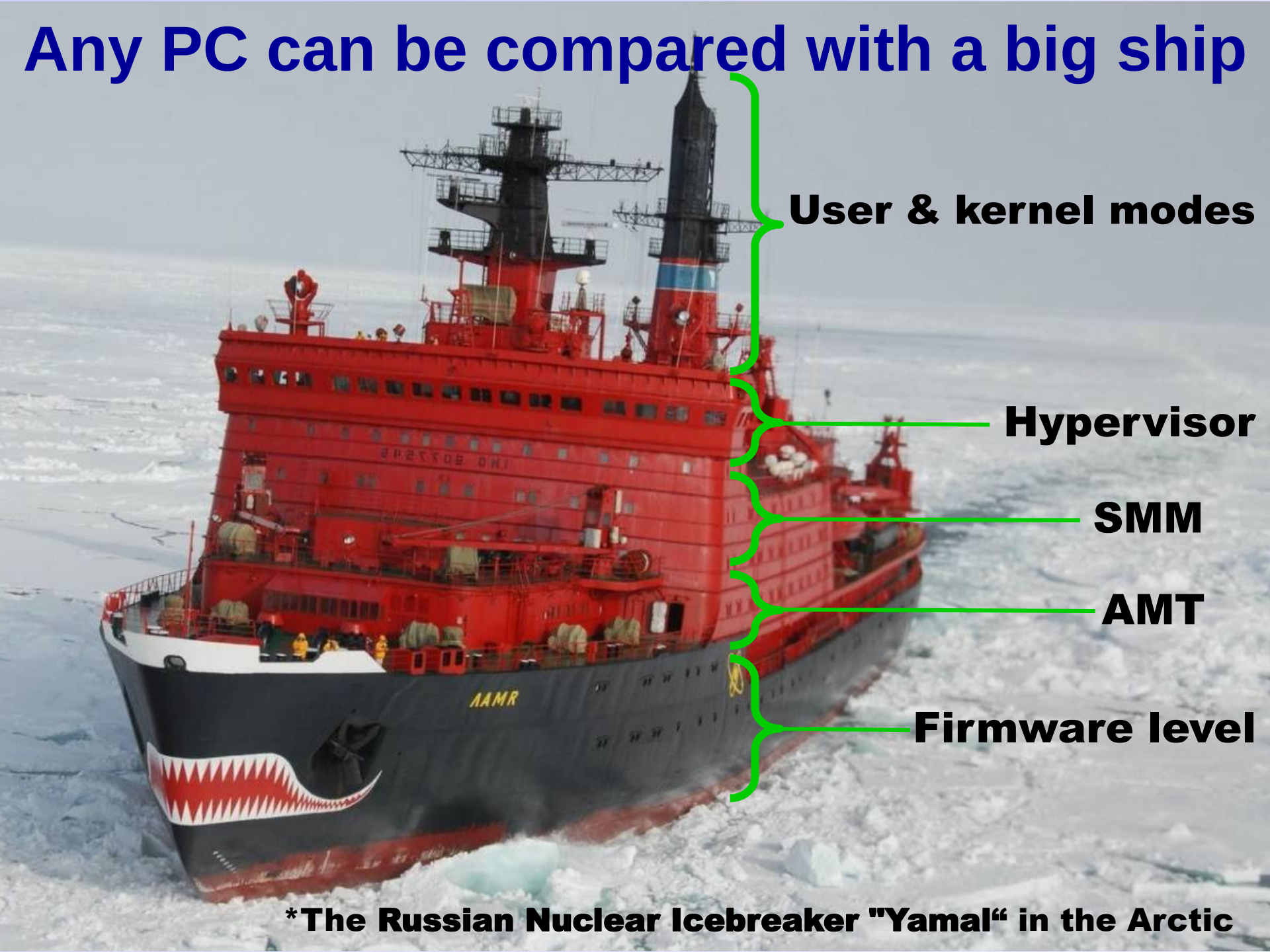
  improvements & its challenges



Any PC can be compared with a big ship

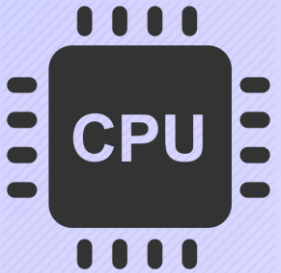
*The Russian Nuclear Icebreaker "Yamal“ in the Arctic

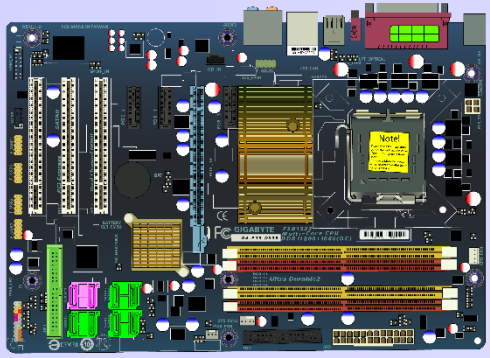
Any PC can be compared with a big ship

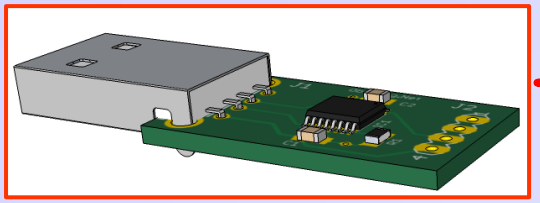


*The Russian Nuclear Icebreaker "Yamal“ in the Arctic

The existing places to plant the backdoor







User & kernel modes (VMX non root mode)

ADFSL 2014

Hypervisor (VMX root mode)

ADFSL 2015

System Management Mode (SMM)

SMM keylogger by Wecherowski, 2009

Active Management Technology (AMT)

AMT keylogger by Stewin & Seifert, 2011

Firmware level e.g. BADUSB, 2014

GPU-based Keylogger by Koromilas, 2013

What & where is a hypervisor?

Image source: http://pngimg.com/download/5932

What & where is a hypervisor?

Image source: http://pngimg.com/download/5932

What & where is a hypervisor?

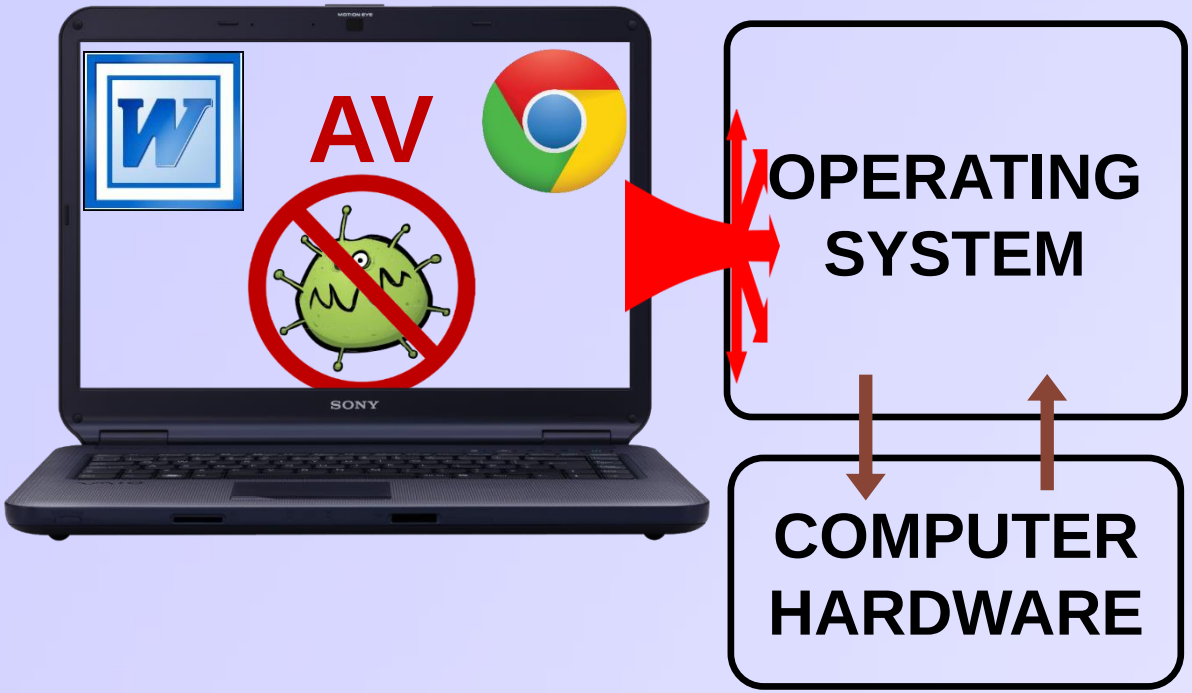


Image source: http://pngimg.com/download/5932

What & where is a hypervisor?

1990-2005

OPERATING SYSTEM

COMPUTER HARDWARE

2005-now

OPERATING SYSTEM

HYPERVISOR*

COMPUTER HARDWARE

CPU

- VT-x
- AMD-V

Image source: http://pngimg.com/download/5932

What & where is a hypervisor?

1990-2005

OPERATING SYSTEM

COMPUTER HARDWARE

2005-now

OPERATING SYSTEM

HYPERVISOR*

COMPUTER HARDWARE

CPU

- VT-x
- AMD-V

*Hypervisor (or HYP) is a code run by CPU in a more privileged mode than OS

Image source: http://pngimg.com/download/5932

What computers support hardware virtualization?

CPU without VT-x low performance computers	CPU with VT-x high performance computers
 mini PC **no**	Server **VT-x** 
 Netbook & Ultrabook **no**	Laptop **VT-x** 
 tablet PC **no**	Workstation **VT-x** 

Does your CPU support Hardware Virtualization?

Check on ark.intel.com or use CPU-Z

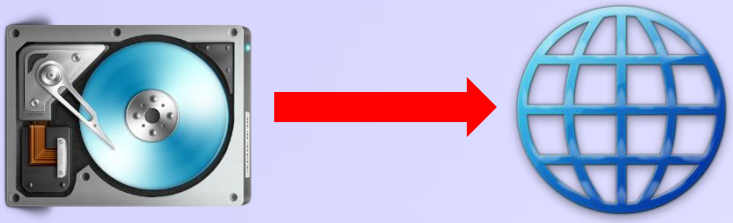
Five features of HYP & the area of its application

Features	1. HYP can control access to memory, HDD etc 2. Impossible to block or delete HYP by OS 3. There is no built-in tool for HYP detection 4. HYP can prevent its detection = stealthy HYP e.g. by using time cheating 5. HYP installs invisibly for both users & AVs
Areas	

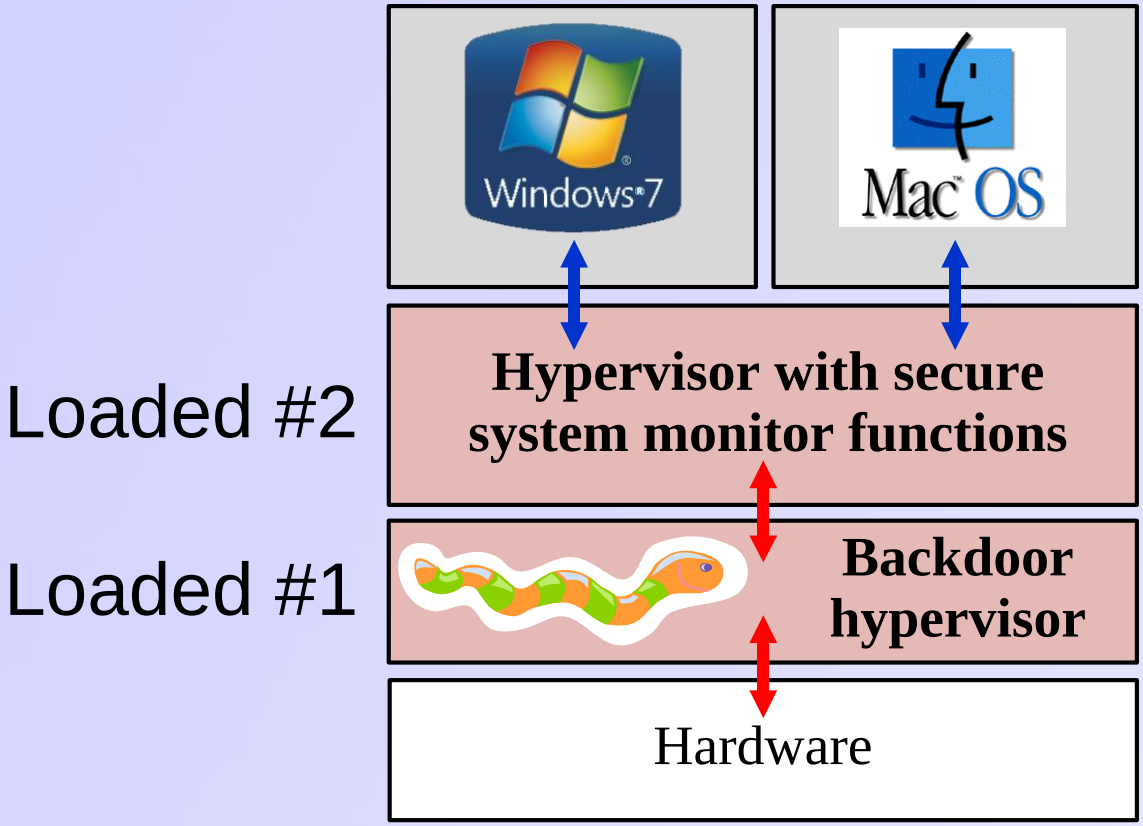
Five features of HYP & the area of its application

Features	1. HYP can control access to memory, HDD etc 2. Impossible to block or delete HYP by OS 3. There is no built-in tool for HYP detection 4. HYP can prevent its detection = stealthy HYP e.g. by using time cheating 5. HYP installs invisibly for both users & AVs
Areas	1 + 2 = for security 1 + 2 + 3 + 4 + 5 = for backdoor

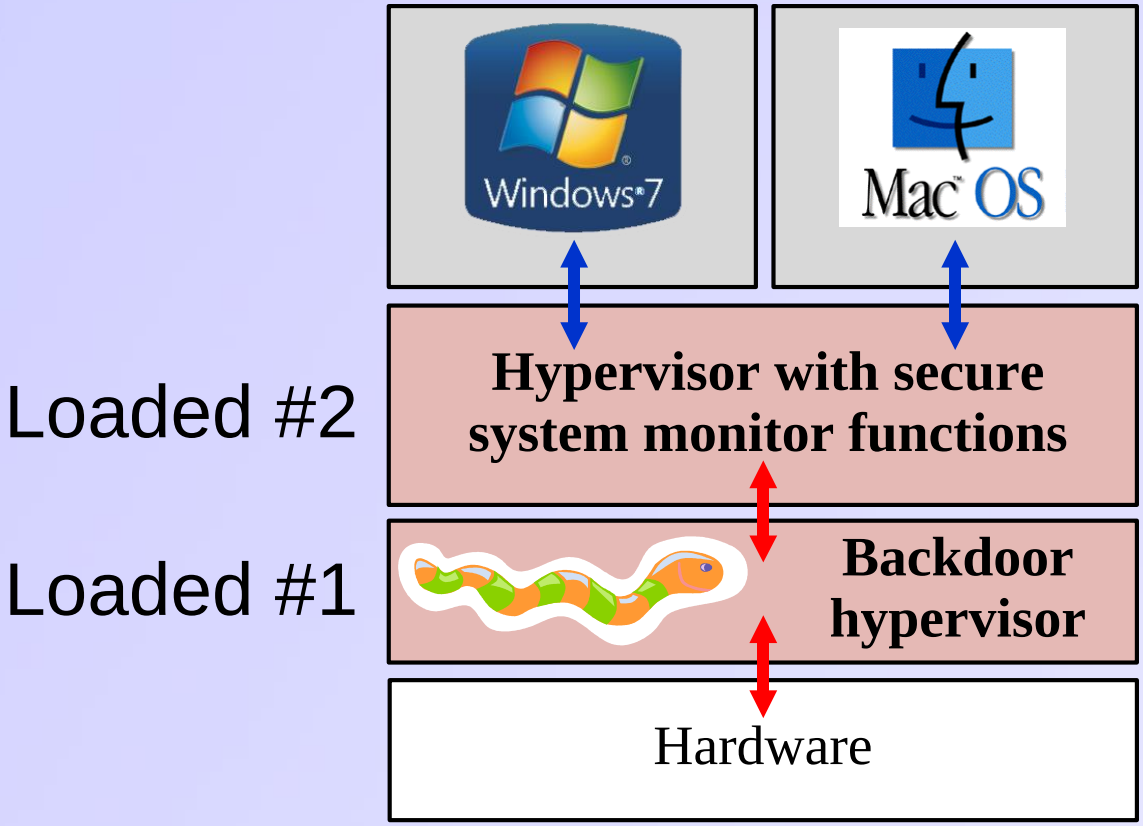
Overview of a backdoor HYP facilities

Backdoor HYP can	Ways to plant a HYP
• record keystrokes 	• using OS vulnerabilities to load a driver-based HYP
• steal all data 	• using BIOS-based approach to infect a motherboard
• block PC 	

Backdoor HYP & well-known examples



Backdoor HYP & well-known examples



HYP example	Author	HYP is loaded by	CPU
Blue Pill	Invisible Things Lab	Windows driver	AMD
Vitriol	Matasano Security	MAC OS driver	intel
Russian Ghost	M.Utin by DeepSec14	BIOS	intel

Analysis of hypervisor detection tools

	Tool	Detection method	Resi-lient?	Easy to distribute?
Hardware	Copilot 2004	Signature based	+	—
	Deep Watch 2008			
Software	Symantec EndPoint Protection 2012	Based on the trusted HYP	—	+
	McAfee Deep Defender 2012			
	Actaeon 2013	Signature based		
	Proof of Concepts 2008 - 2015	Behavior based & Time based		
	New proposal tool	**Time based**	**+**	**+**

Analysis of hypervisor detection tools

	Tool	Detection method	Resi-lient?	Easy to distribute?
Hardware	Copilot 2004	Signature based	+	—
	Deep Watch 2008			
Software	Symantec EndPoint Protection 2012	Based on the trusted HYP	—	+
	McAfee Deep Defender 2012			
	Actaeon 2013	Signature based		
	Proof of Concepts 2008 - 2015	Behavior based & Time based		
	New proposal tool	**Time based**	**+**	**+**

Analysis of hypervisor detection tools

	Tool	Detection method	Resi-lient?	Easy to distribute?
Hardware	Copilot 2004	Signature based	+	—
	Deep Watch 2008			
Software	Symantec EndPoint Protection 2012	Based on the trusted HYP	—	+
	McAfee Deep Defender 2012			
	Actaeon 2013	Signature based		
	Proof of Concepts 2008 - 2015	Behavior based & Time based		
	New proposal tool	**Time based**	**+**	**+**

Analysis of hypervisor detection tools

	Tool	Detection method	Resi-lient?	Easy to distribute?
Hardware	Copilot 2004	Signature based	+	—
	Deep Watch 2008			
Software	Symantec EndPoint Protection 2012	Based on the trusted HYP	—	+
	McAfee Deep Defender 2012			
	Actaeon 2013	Signature based		
	Proof of Concepts 2008 - 2015	Behavior based & Time based		
	New proposal tool	**Time based**	**+**	**+**

What are the principles of these software detection tools?

Hypervisor detection methods

Signature based

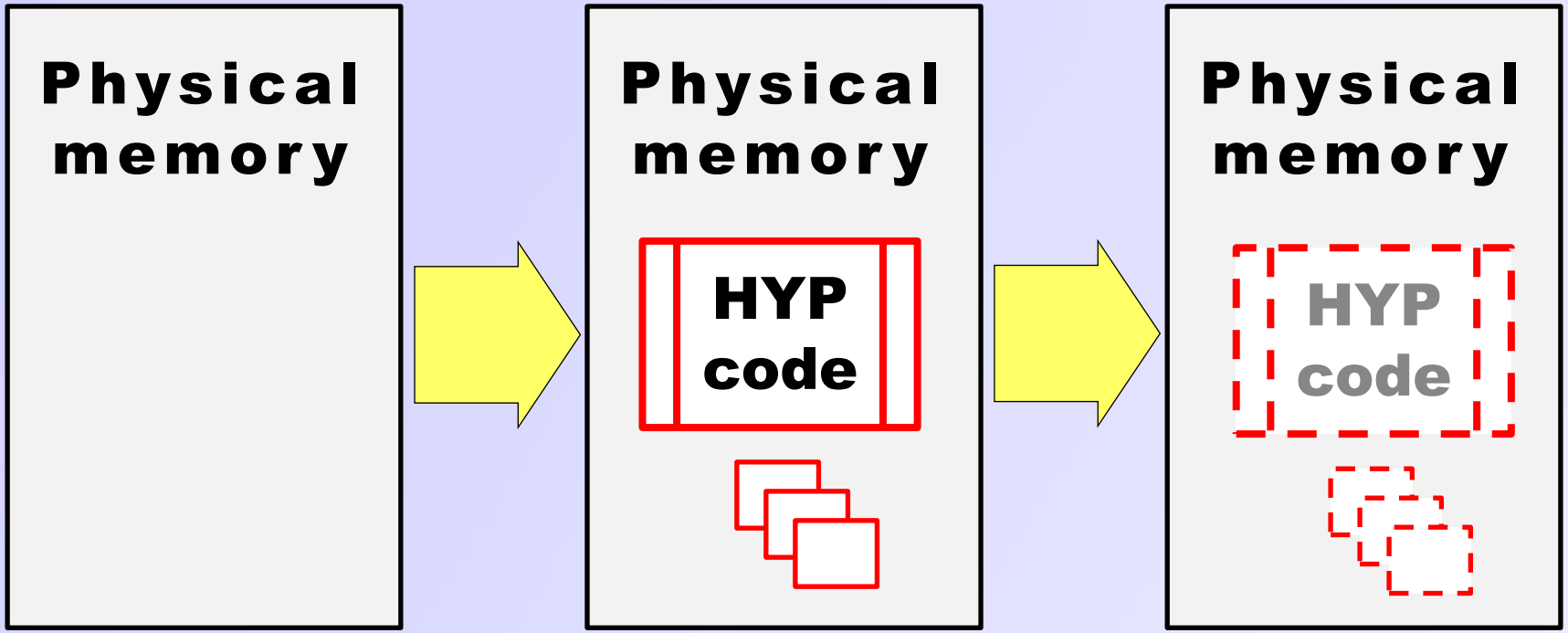
Based on the trusted HYP

Behavior based

Time based

Signature based detection

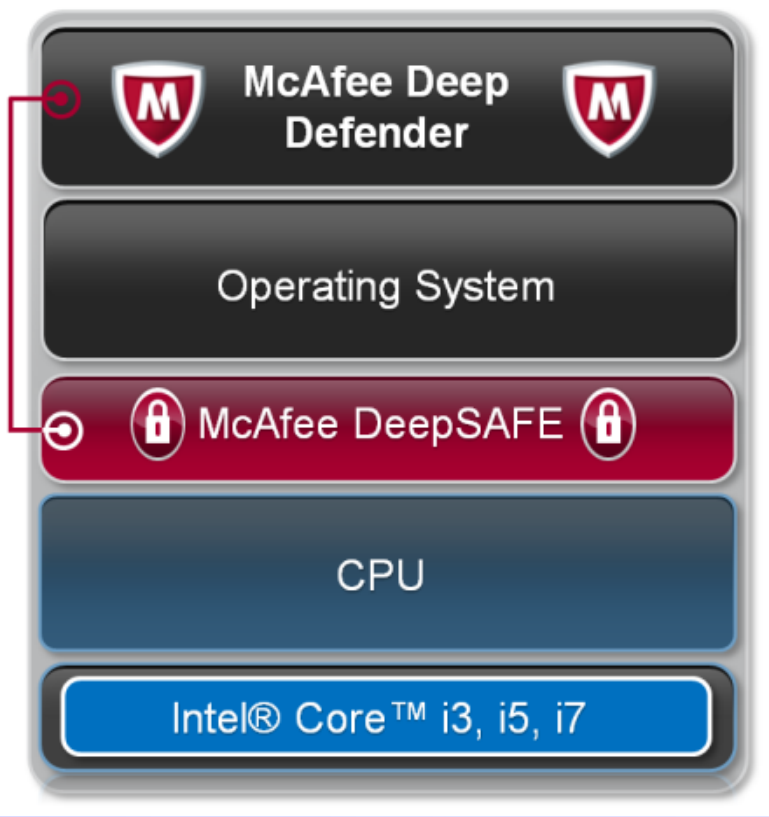
Without HYP | Non-stealthy HYP | Stealthy HYP



- HYP is loaded to memory
- We can detect a HYP using a search in the mem dump

- HYP hides memory areas
- HYP prevents acquiring a real memory dump from OS

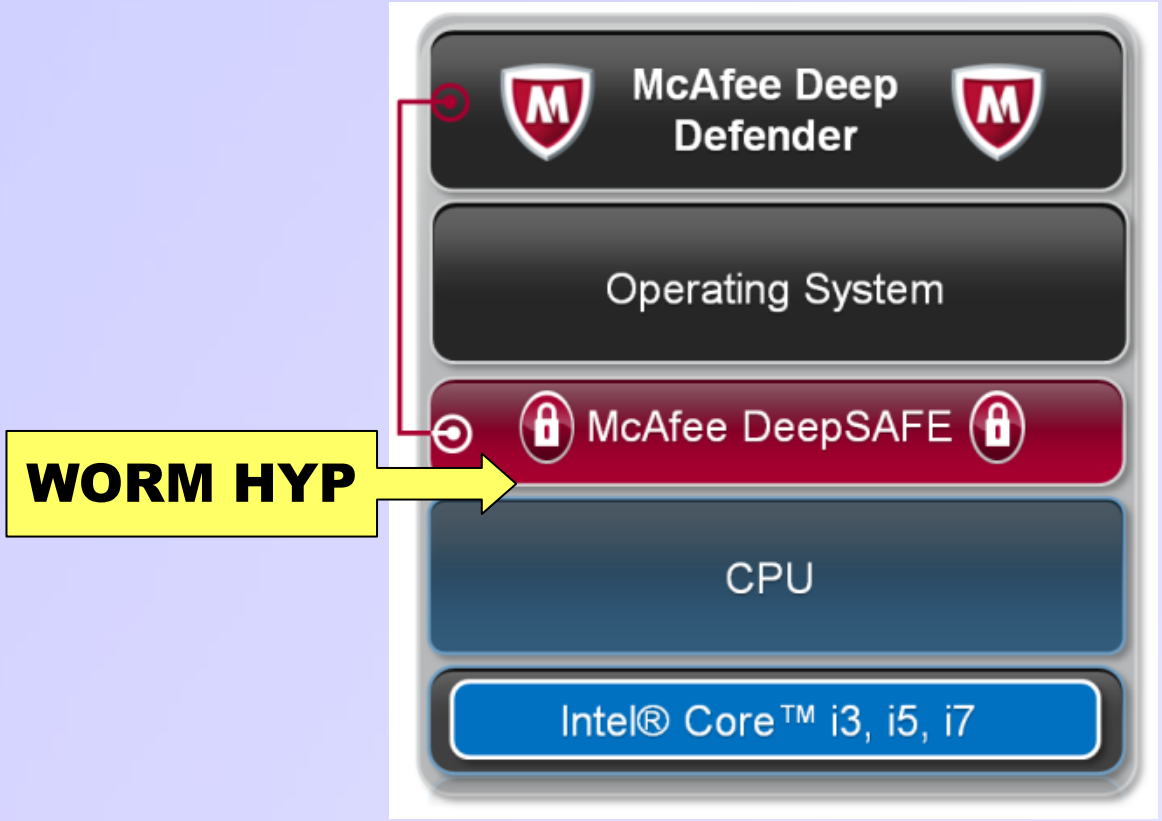
Detection based on the trusted HYP



The boot process with McAfee Deep Defender

***McAfee Deep Defender Technical Evaluation and Best Practices Guide**

Detection based on the trusted HYP



Vulnerability:

- If worm HYP is loaded first it blocks Deep Defender
- Exp. BIOS-based HYP

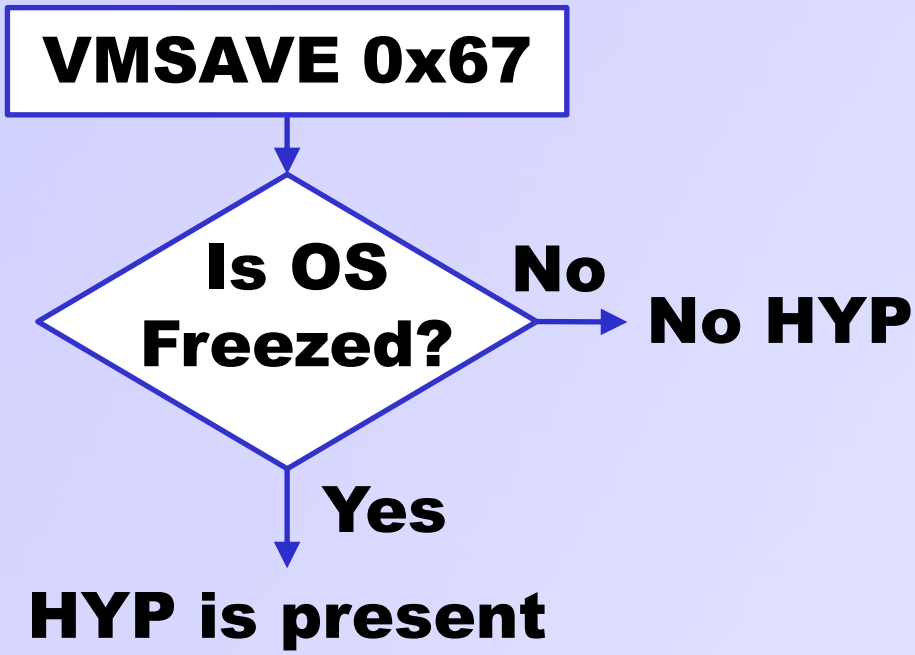
WORM HYP

The boot process with McAfee Deep Defender

***McAfee Deep Defender Technical Evaluation and Best Practices Guide**

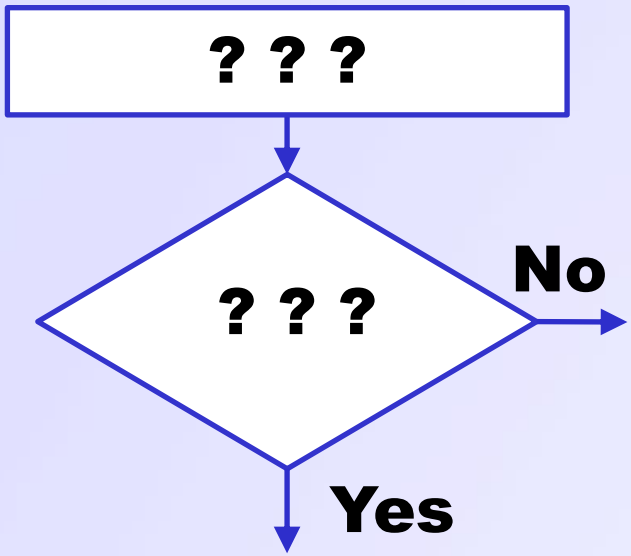
Behavior based detection

Old CPU & HYP 2007-2011



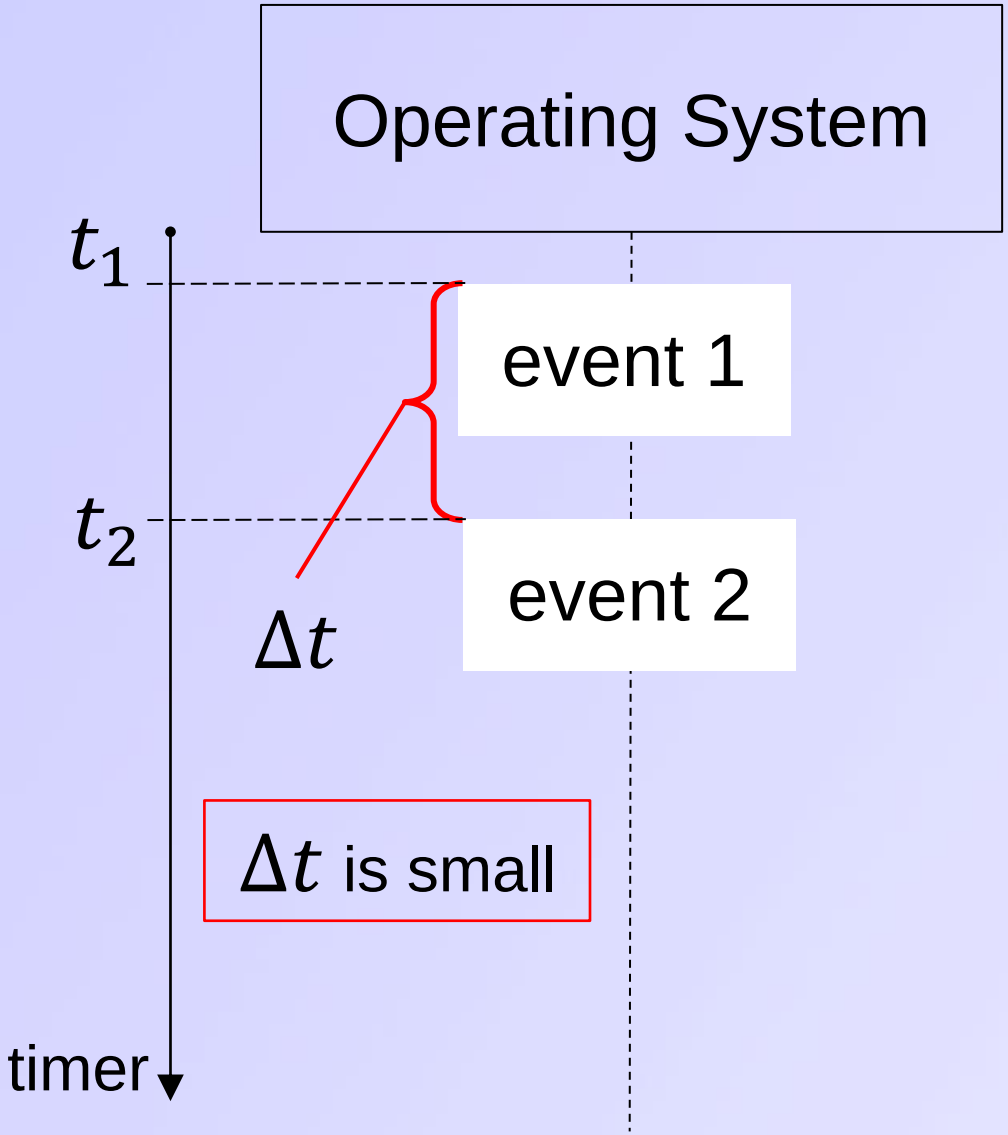
VMSAVE 0x67 is a “bug” instruction presented by Barbosa in the 2007

New CPU & HYP nowadays

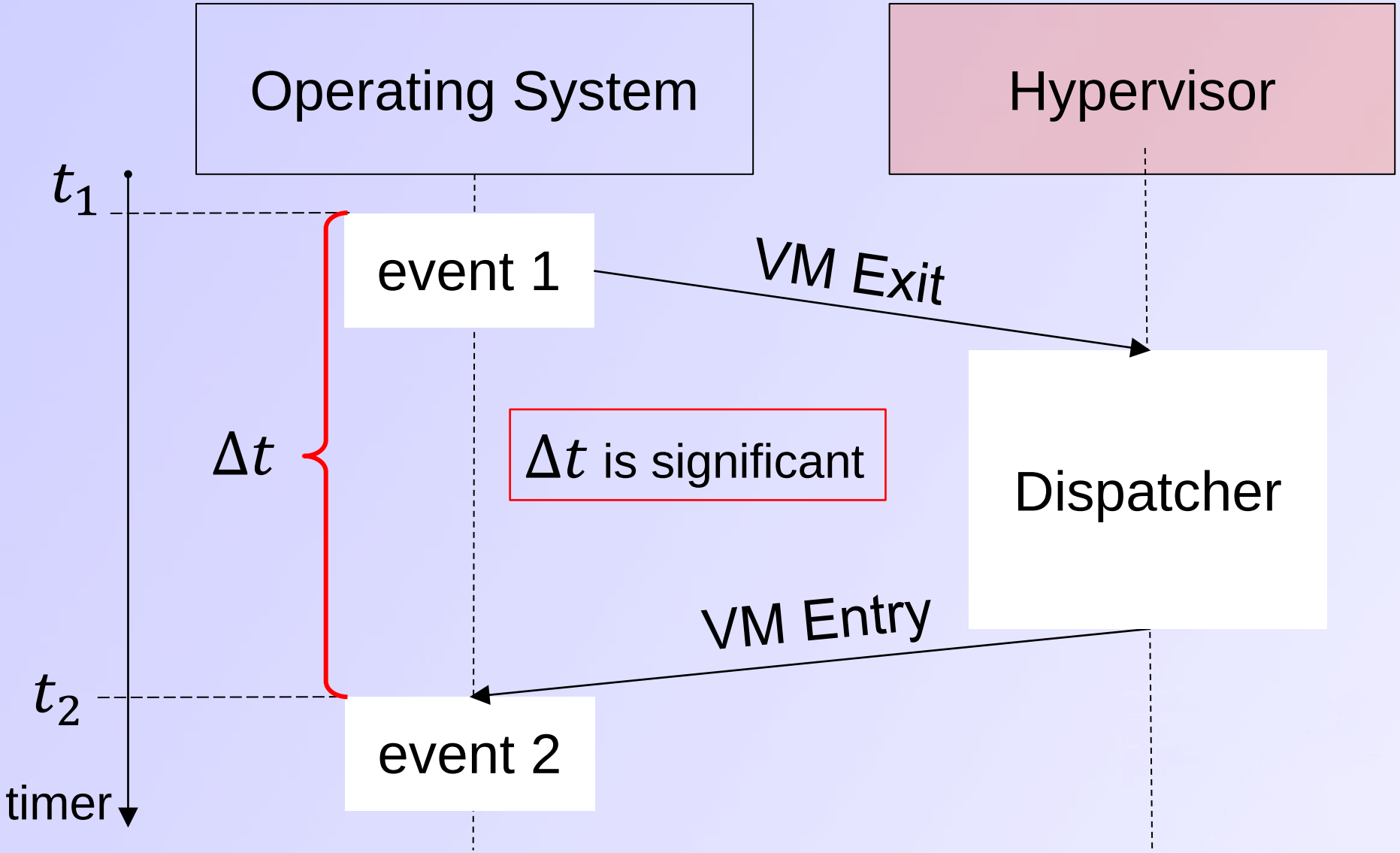


There is no such “bug” instruction for new CPU

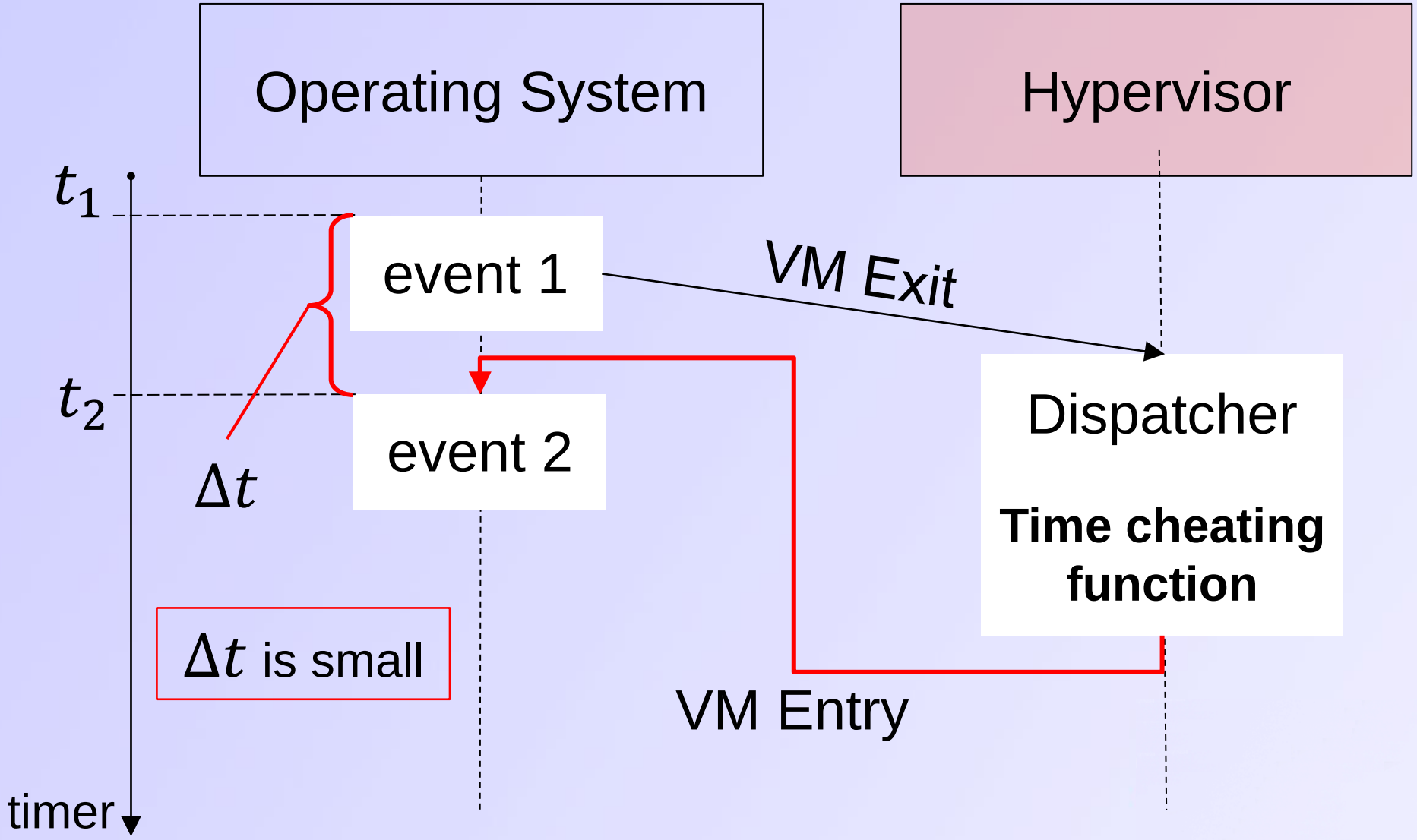
Time based detection



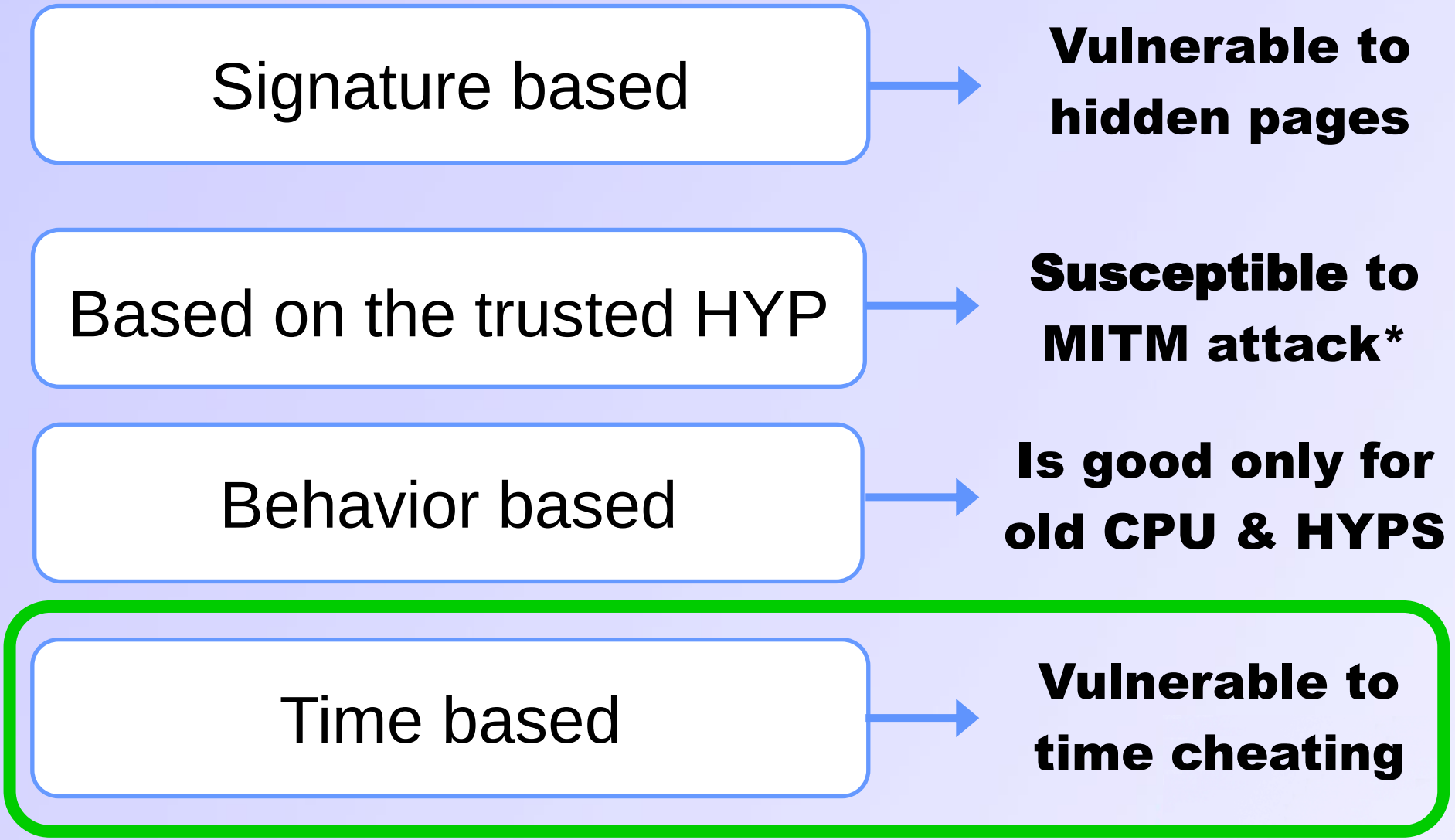
Time based detection



Time based detection

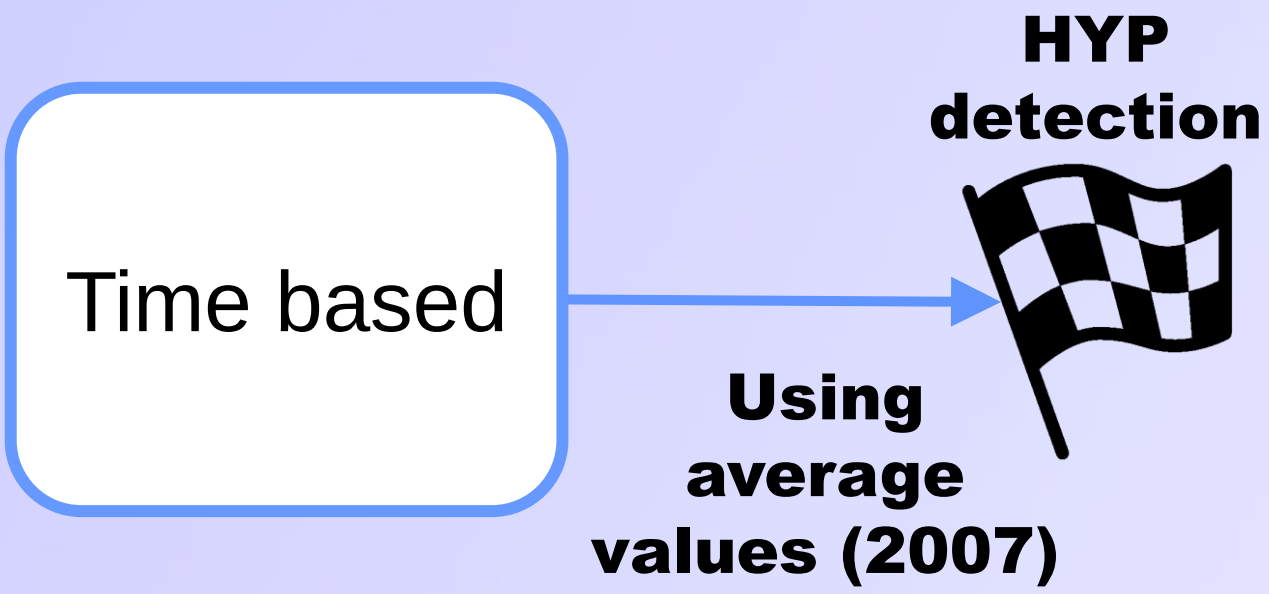


Drawbacks of HYP detection methods

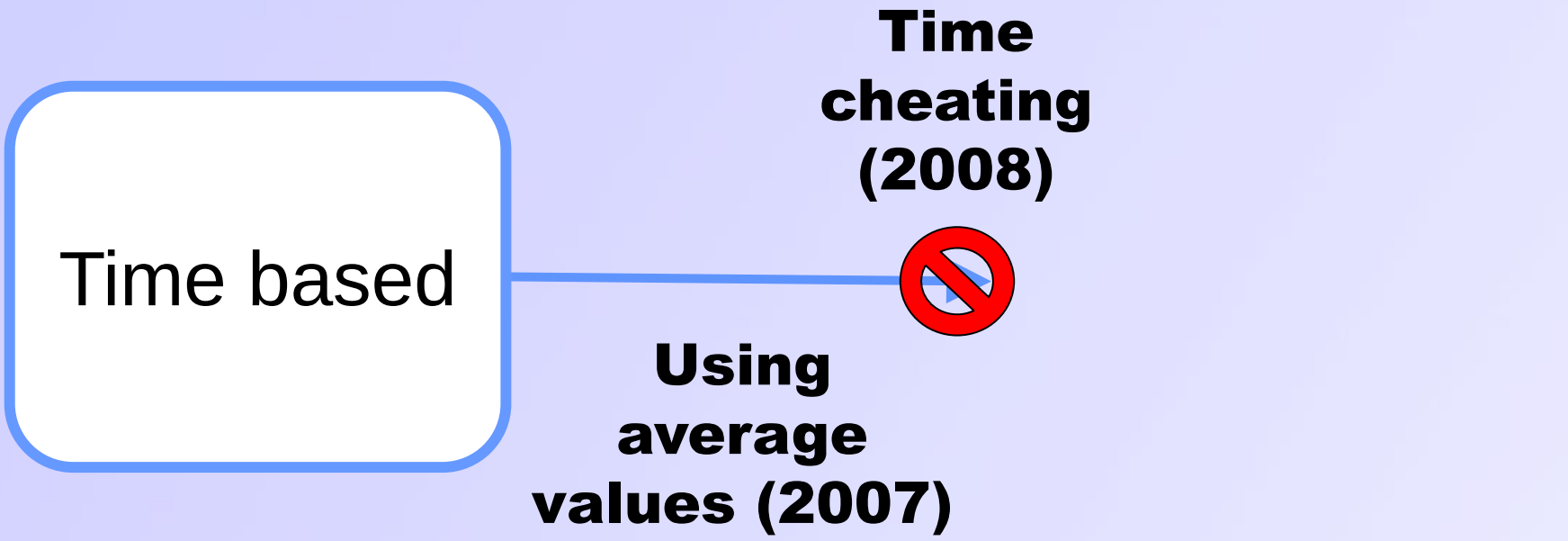


*MITM attack - man in the middle attack

Time based detection. Yesterday.



Time based detection. Today.



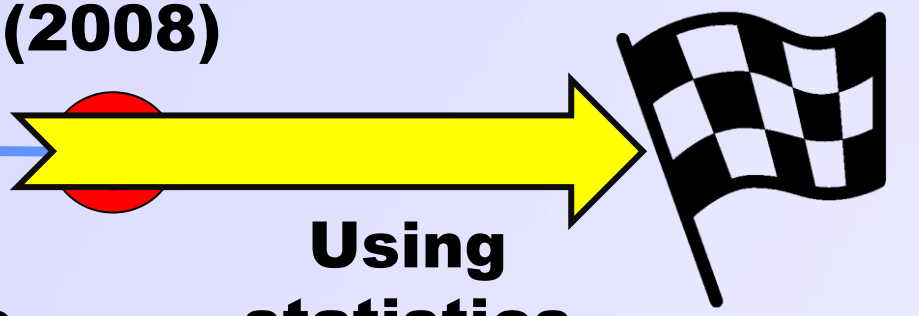
#1 How to detect a HYP that applies time cheating?

Time based detection. Today & tomorrow

Time based

Time cheating (2008)

HYP detection

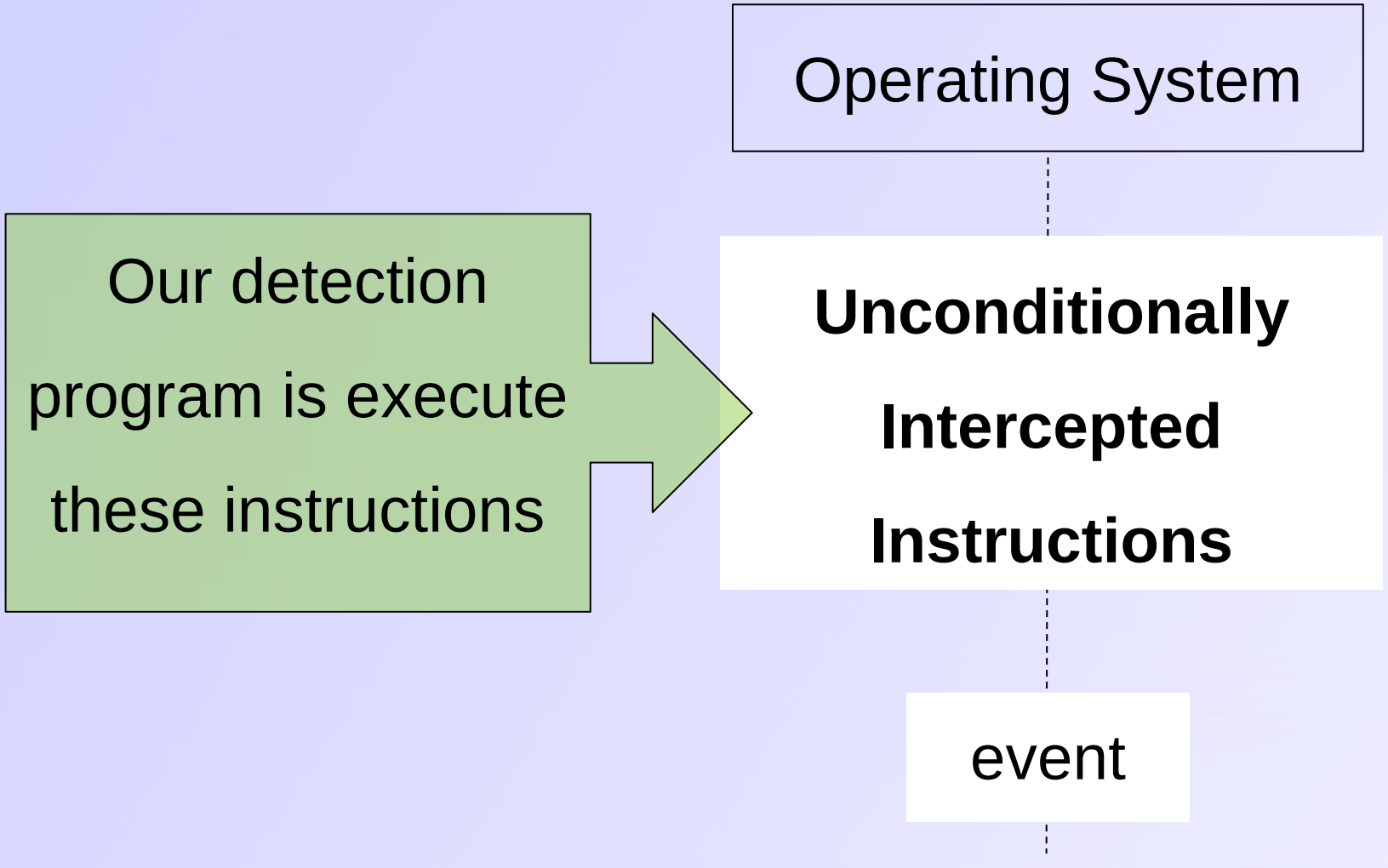


Using average values (2007)

Using statistics (CDFSL 2015)

#1 How to detect a HYP that applies time cheating?

Let's focus on the time-based detection by unconditionally intercepted instructions



Time based detection by Unconditionally Intercepted Instructions

What are these? → Their execution is always trapped by HYP
e.g. CPUID instruction

How to detect a HYP using them? →

Average IET values →

Time based detection by Unconditionally Intercepted Instructions

What are these?

Their execution is always trapped by HYP

e.g. CPUID instruction

How to detect a HYP using them?

1. T1 = get_time()
2. execute CPUIDs
3. T2 = get_time()

Instructions Execution Time (IET) = T2 - T1

Average IET values

Time based detection by Unconditionally Intercepted Instructions

What are these?

Their execution is always trapped by HYP

e.g. CPUID instruction

How to detect a HYP using them?

1. T1 = get_time()
2. execute CPUIDs
3. T2 = get_time()

Instructions Execution Time (IET) = T2 - T1

Average IET values

	Non Stealthy
Without HYP	**~2,000**
With HYP	**~20,000**

* Lifebook E752 Core i5, Windows Live CD XP DDD

Time based detection by Unconditionally Intercepted Instructions

What are these?

Their execution is always trapped by HYP

e.g. CPUID instruction

How to detect a HYP using them?

1. T1 = get_time()
2. execute CPUIDs
3. T2 = get_time()

Instructions Execution Time (IET) = T2 - T1

Average IET values

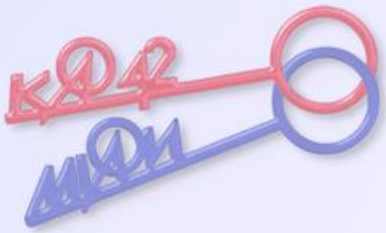
	Non Stealthy	Stealthy HYP
Without HYP	**~2,000**	**~2,000**
With HYP	**~20,000**	**~2,000**

!

* Lifebook E752 Core i5, Windows Live CD XP DDD

How do we want to detect a HYP?

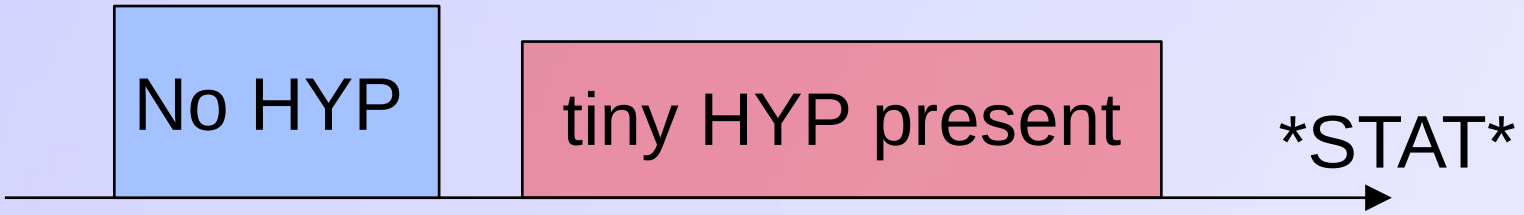
What steps can be used to detect a stealthy hypervisor?



What are the steps for time-based detection?

Preliminary stage

1. Load a clear PC without any HYP
2. Measure time for no HYP and for HYP present
3. Calculate *STAT* value (now it is average)
4. Achieve intervals for each of two cases:



Detection stage

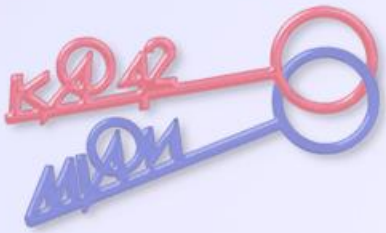
5. Measure time & calculate *STAT* value
6. Check if *STAT* value is belongs to the intervals:

$$If\ {}^{*}\mathrm{STAT}^{*} \in NoHYP\ \therefore\ PC\ is\ clear$$
$$If\ {}^{*}\mathrm{STAT}^{*} \in HYP\ present\ \therefore\ HYP\ is\ present$$

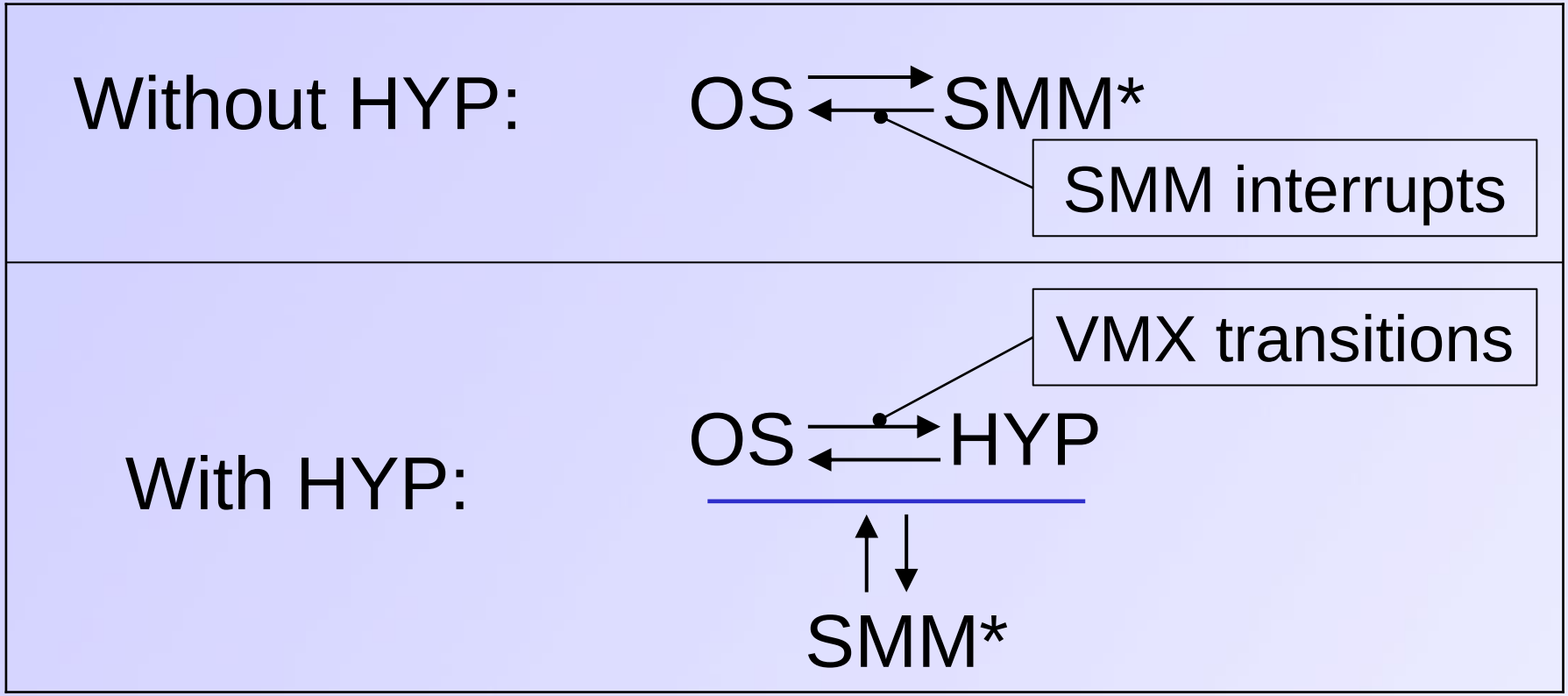
STAT = ?

How to find the appropriate statistics?

What is happening to the PC during time measurements?



What is happening to the computer during time measurements?



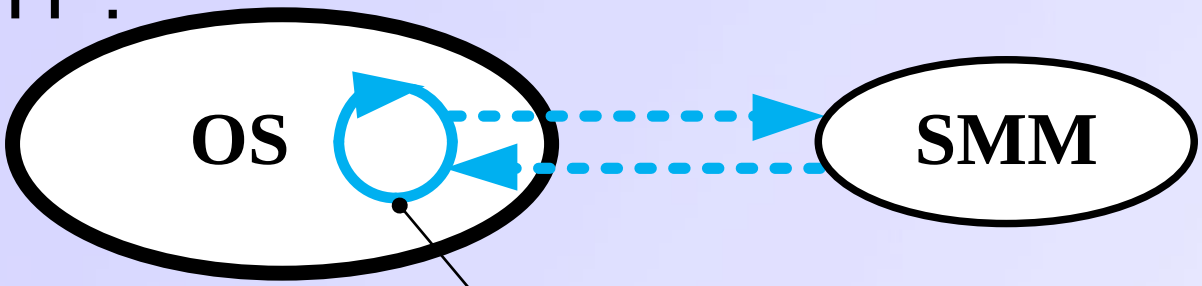
SMM — System Management Mode, works lower than HYP & OS

SMM interrupts — occur randomly & suspend PC for a short time

VMX transitions — catch execution of every CPUID instruction

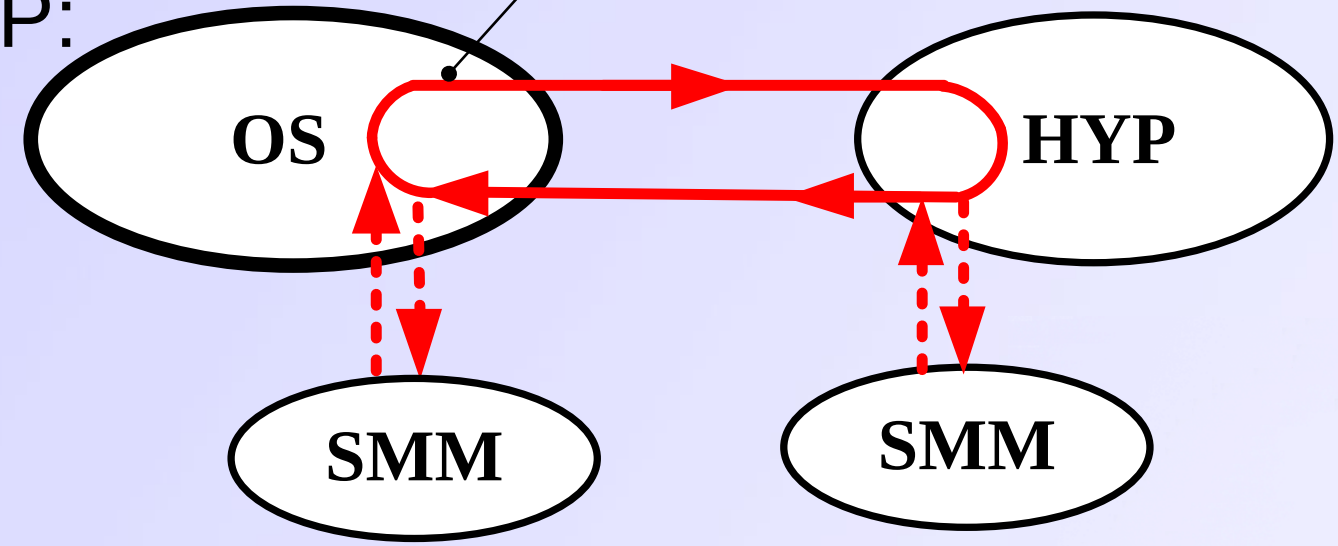
Switching between CPU modes during time measurements of CPUID execution

1. Without HYP:



Process of execution CPUID instructions

2. With HYP:



Theoretic analysis of switches between modes

- CPU works as a stochastic system → **IET is a random variable**
- SMM interrupts both OS & HYP → **IET has a layered structure**
- IET indexes are increased after HYP is loaded:

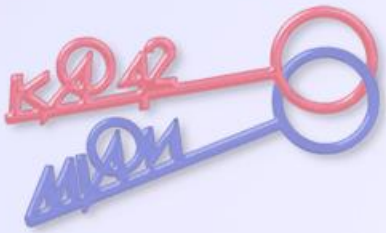
Average
Number of layers
Variance & 4^{th} order moment

Theoretic analysis of switches between modes

- CPU works as a stochastic system → **IET is a random variable**
- SMM interrupts both OS & HYP → **IET has a layered structure**
- IET indexes are increased after HYP is loaded:

Average	Time-cheating by HYP
Number of layers	**Both are possible for stealth HYP detection**
Variance & 4^{th} order moment	

Let’s check these three ideas by experiment



Scheme of the experiment

1. Run a tiny HYP with time cheating
2. Measure IET by the own driver:

```
for ( 10 ) /*< outer loop */
{
        for ( 1000 ) /*< inner loop */
        {
                T1 = read_tsc()
                CPUID // #1
                …
                CPUID // # 10
                T2 = read_tsc()
                save_one_IET_value(T2-T1)
        }
        Sleep( 2 sec )
}
```

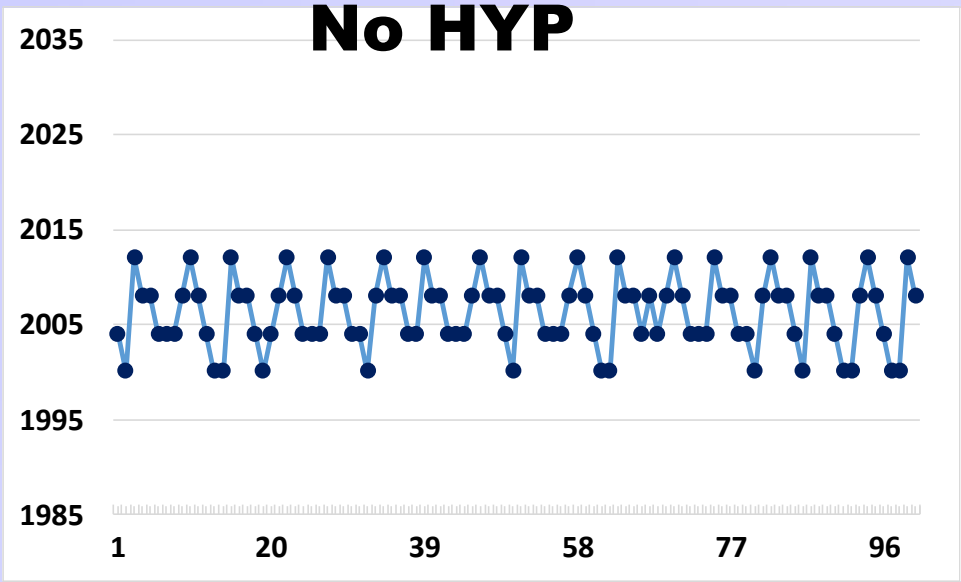
→ **matrix 1000 x 10**

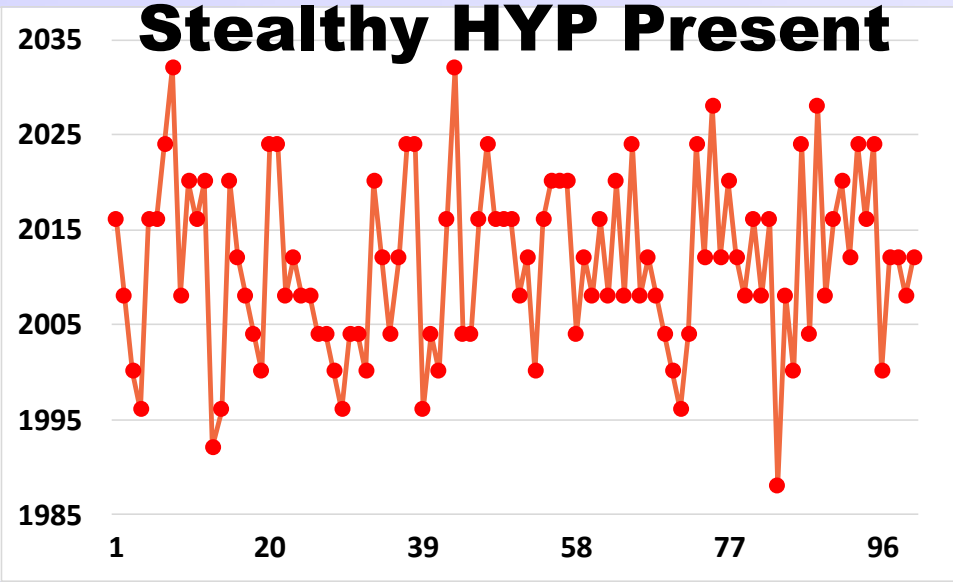
Instruction Execution Time in CPU ticks*

Number of inner loop interactions	Number of outer loop interactions: 1	2	3	…	10
1	**2004**	2008	2048	…	2044
2	**2000**	2008	2048	…	2048
3	**2012**	2004	2048	…	2044
4	**2008**	2000	2048	…	2048
5	**2008**	2004	2044	…	2040
…	…	…	…	…	…
1000	**2008**	2000	2040	…	2036

* without HYP, Lifebook E752 Core i5, Windows Live CD XP DDD

Analysis of the experimental results

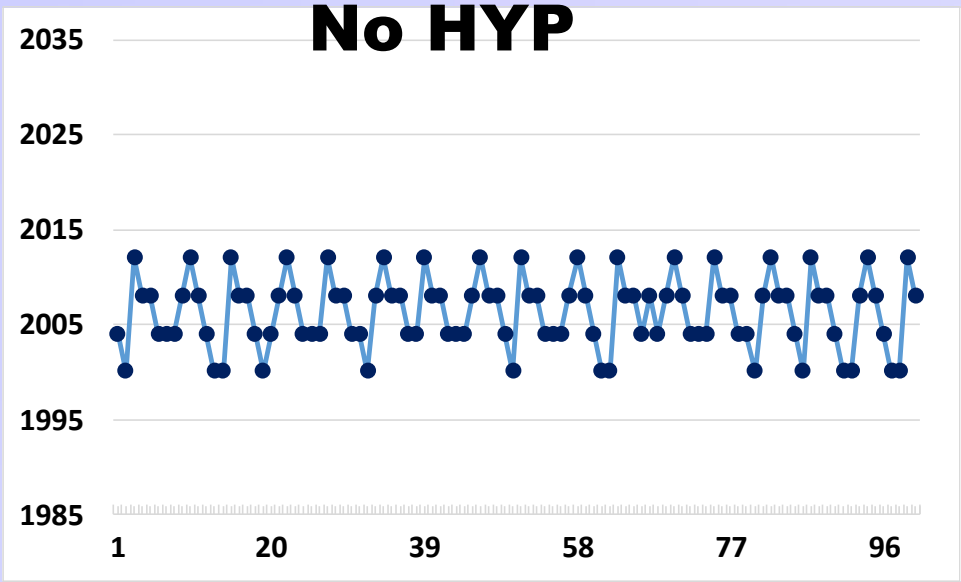


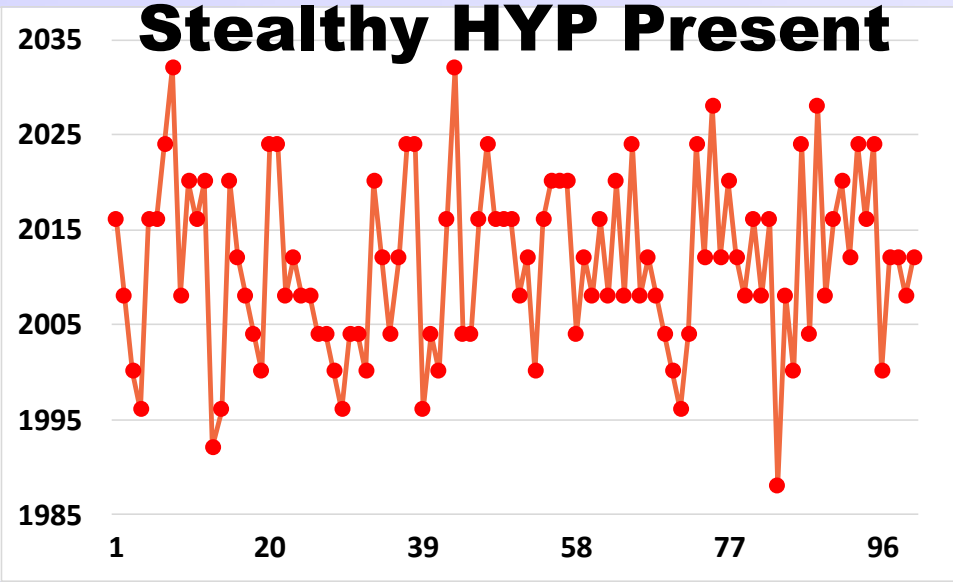


Comparison of statistical indexes values

Are averages values the same?

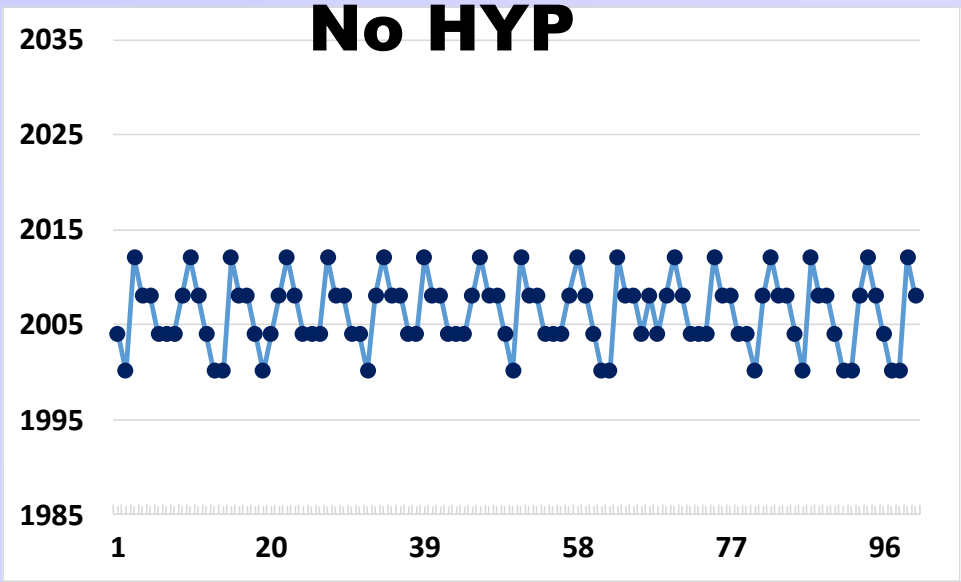
Analysis of the experimental results

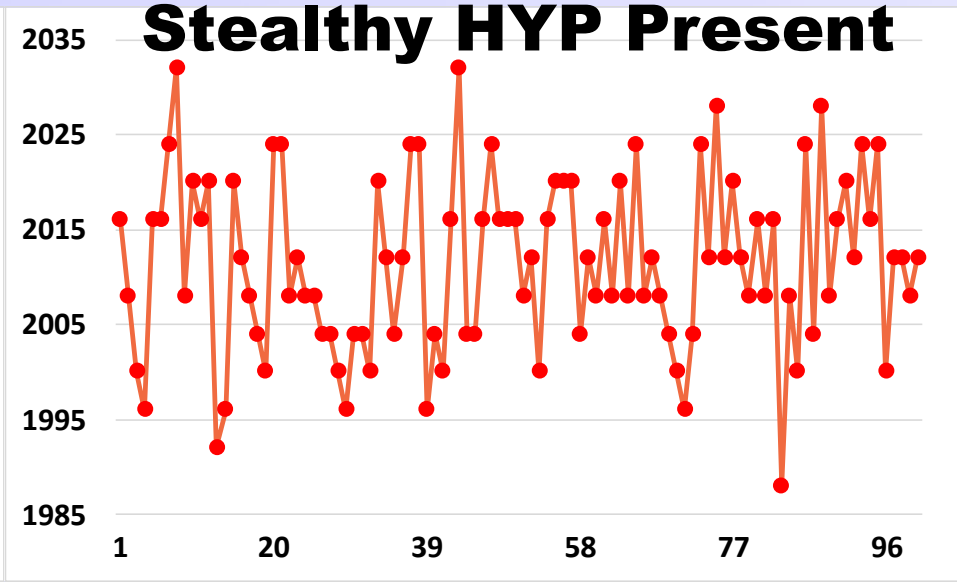




Comparison of statistical indexes values	
Yes, averages values are the same	☑
Does IET have a layered nature?	☐

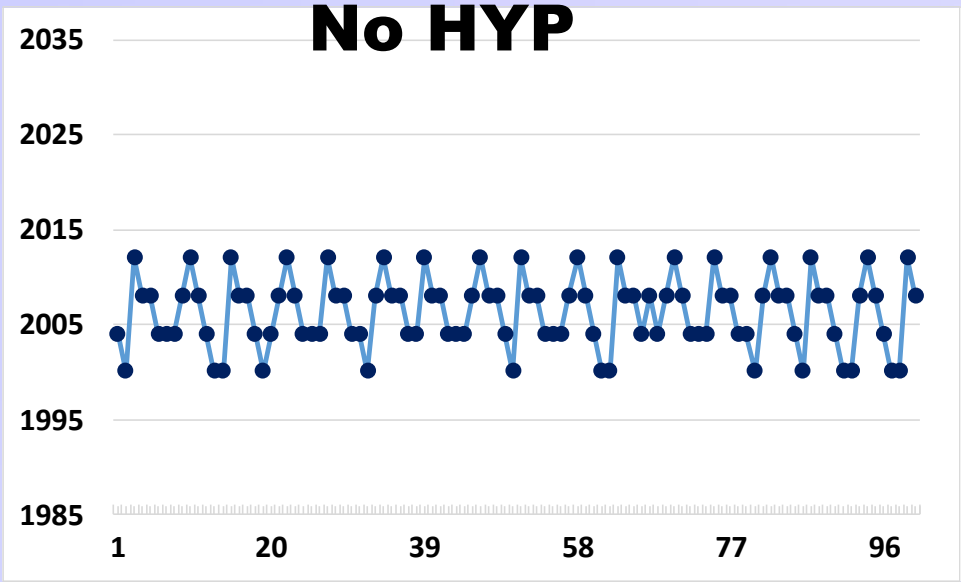
Analysis of the experimental results

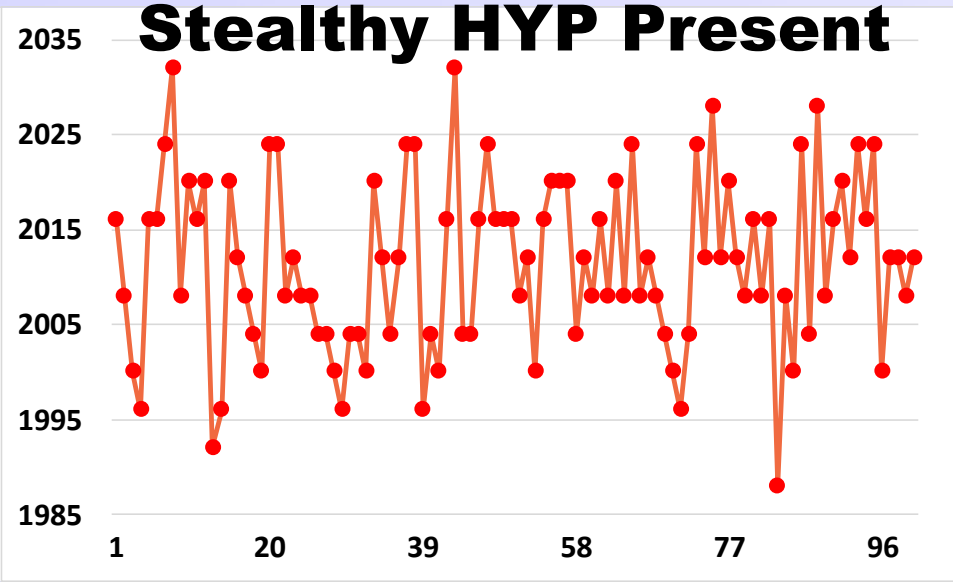




Comparison of statistical indexes values	
Yes, averages values are the same	☑
Yes, IET has a layered nature	☑
Is the number of layers increased? Is the variance increased ?	☐

Analysis of the experimental results





Comparison of statistical indexes values	
Yes, averages values are the same	☑
Yes, IET has a layered nature	☑
The number of layers is increased: 4 < 12 The variance is increased: 14 < 85	☑

Yeah! We've done it!

We've found the following "resilient" statistics:

- number of horizontal layers
- variance $\Rightarrow V = \frac{\sum(x_i - \bar{X})^2}{n}$
- 4th order moment $\Rightarrow \bar{M}_4 = \frac{\sum(x_i - \bar{X})^4}{n}$

Let's use statistical tests to complete samples

But also IET has the following anomalies:

- IET samples include noise
- IET samples statistics fluctuate daily
- IET random variable is not normally distributed

What statistical tests are appropriate to compare these samples?

Possible ways to compare the samples

Classical parametric tests	Non-parametric tests
• Student's t-test • ANOVA & ANCOVA	• Wilcoxon test
Require normal distribution	Give bad approximation

Possible ways to compare the samples

Classical parametric tests	Non-parametric tests
• Student's t-test • ANOVA & ANCOVA	• Wilcoxon test
Require normal distribution	Give bad approximation

Kornfeld (USSR'65) or Strellen (GER'01) method:

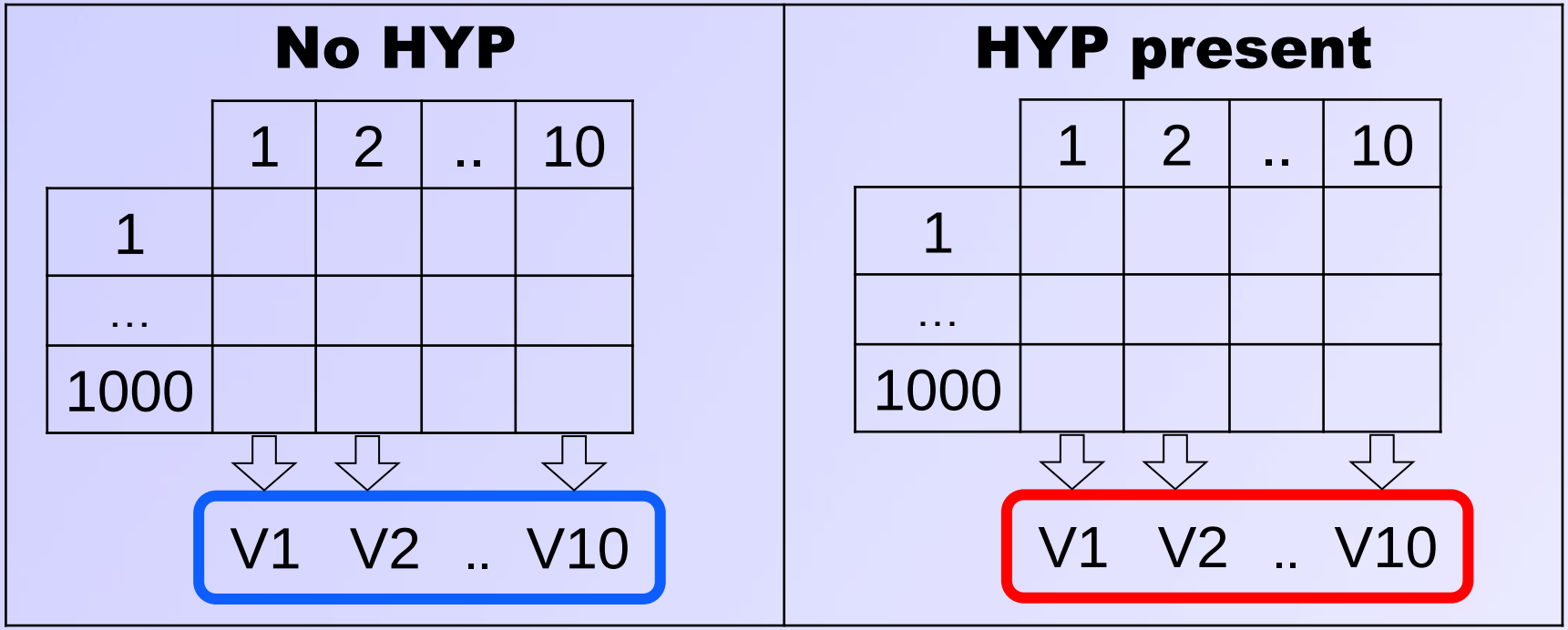
$$Let \;\; T_1, T_2, .. T_n \;\; is\ a\ sample, \qquad therefore$$

$$confidence\ interval: \;\; (T_{MIN}, T_{MAX})$$

$$confidence\ level: \;\; P = 1 - 0.5^{n-1}$$

Calculate statistics & variation intervals

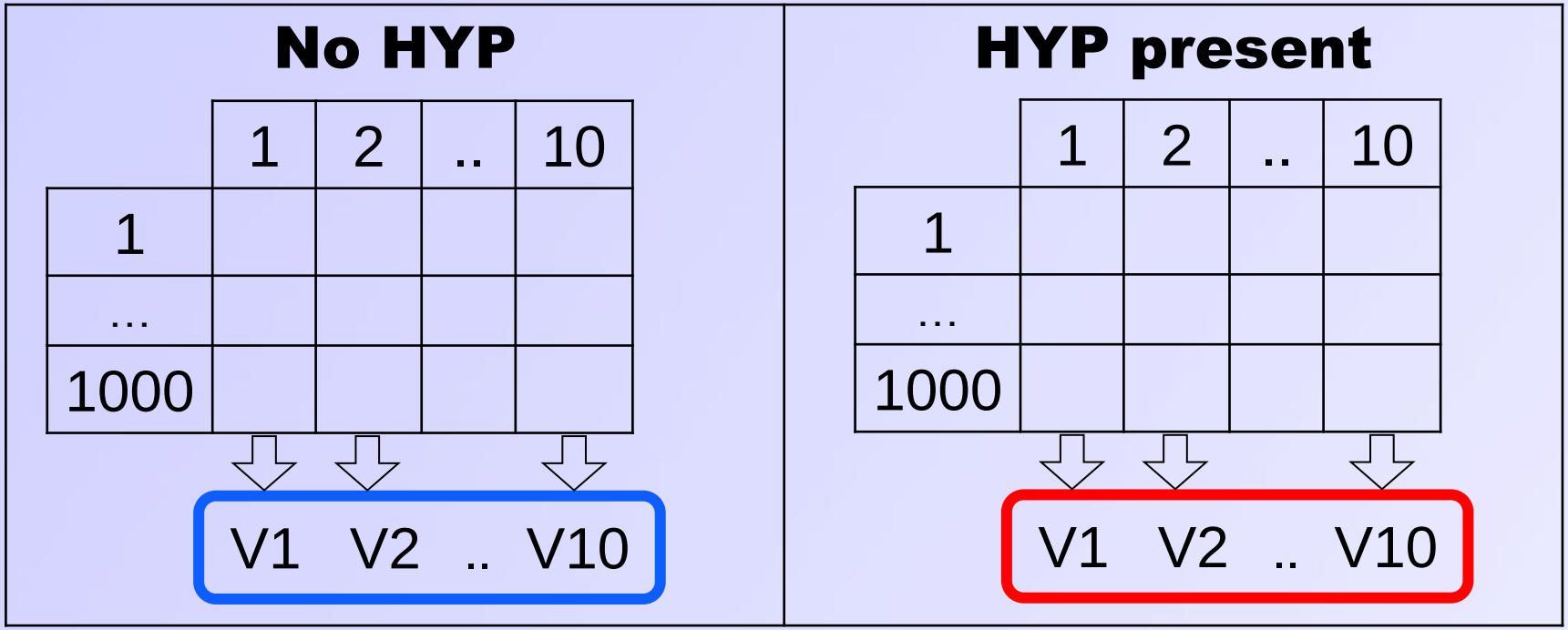
1. Calculate variances for each matrixes of IET values:



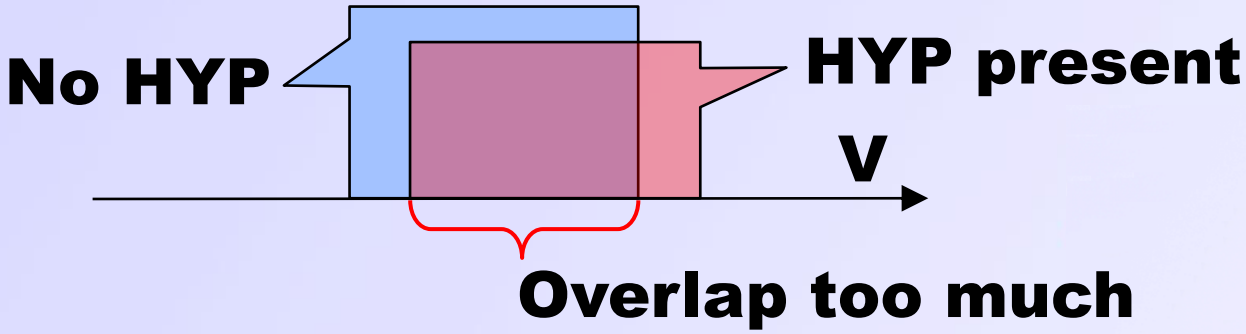
2. The result:

Calculate statistics & variation intervals

1. Calculate variances for each matrixes of IET values:

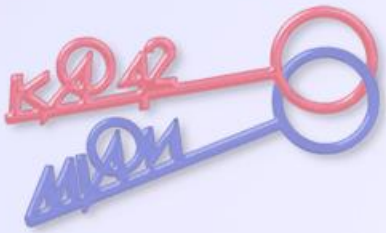


2. The result: instability of statistics values

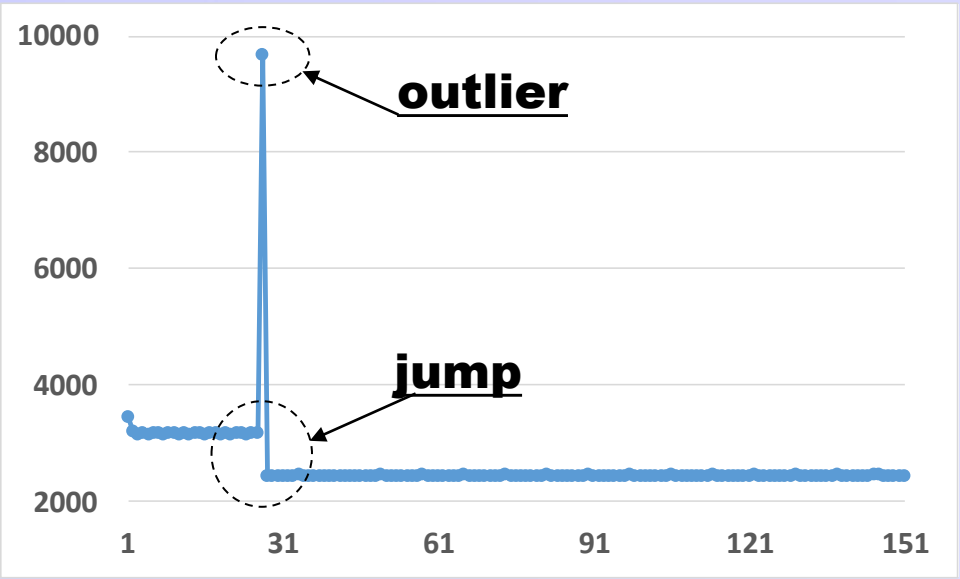


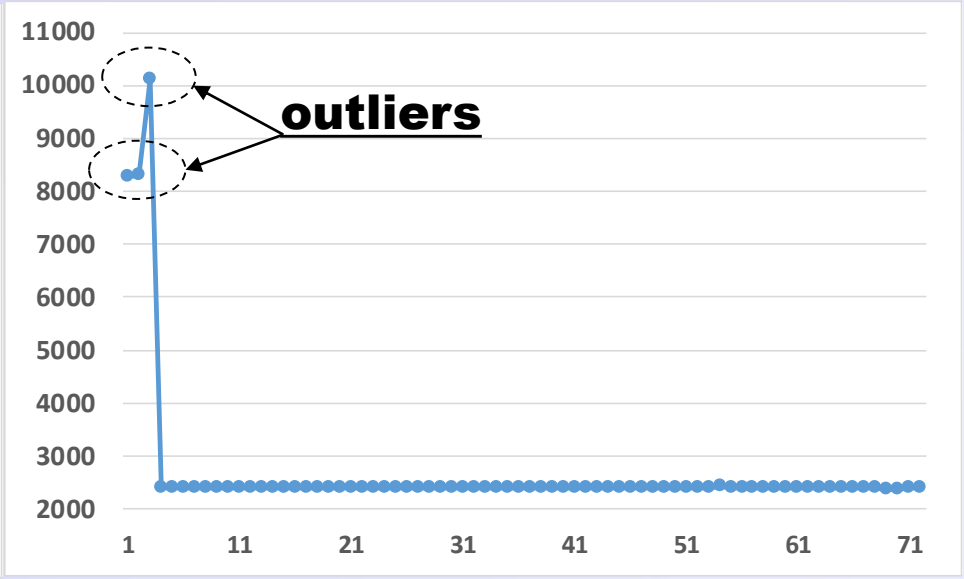
Data fluctuation: instability of statistics

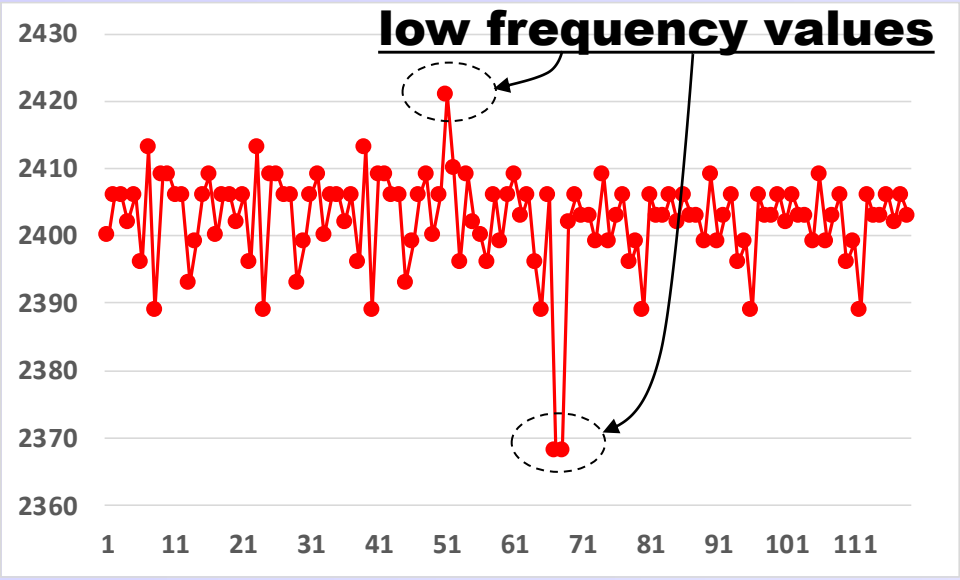
What are the reasons for the instability of statistics?



Reasons for the data instability or data fluctuations are outliers & jumps





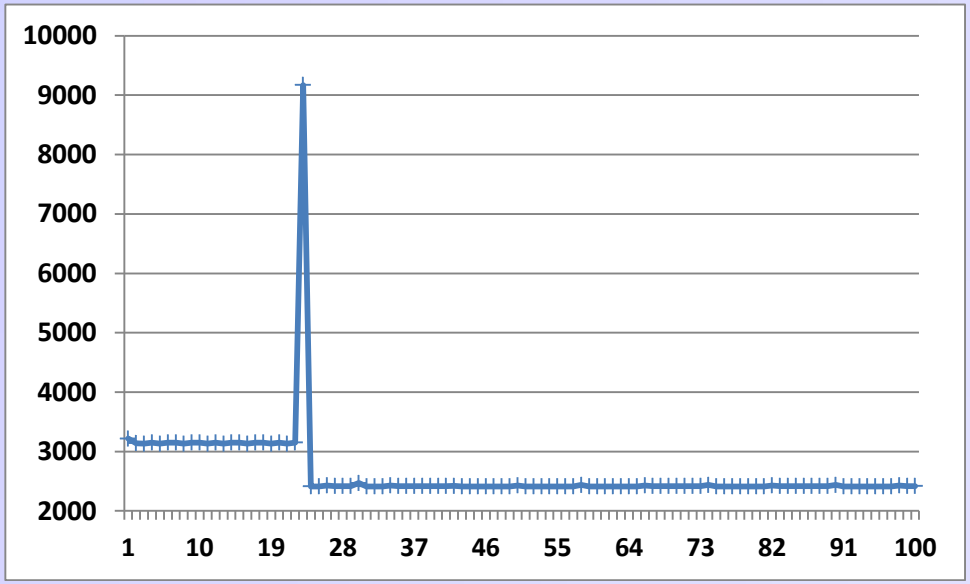


$$V = \frac{\sum (X_i - \bar{X})^2}{n}$$

Variance is significantly increased because of outliers and jumps

How to overcome the negative influence of outliers & jumps

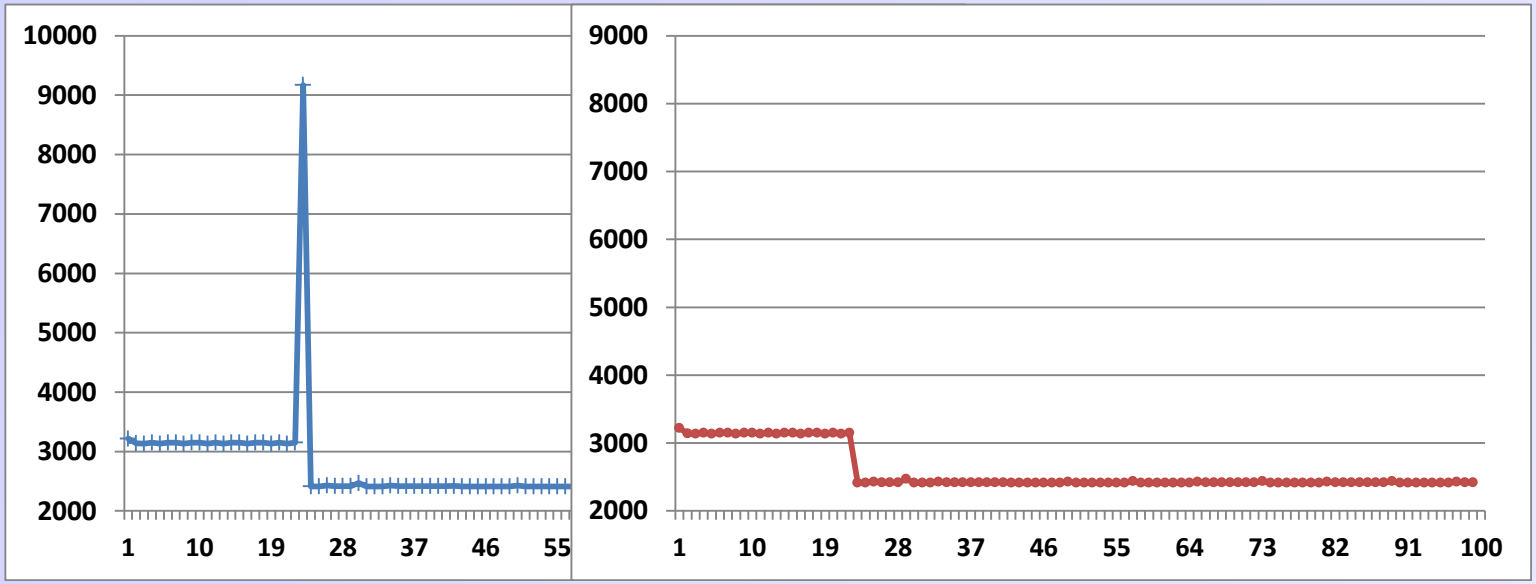
Type of noise	Outlier, low frequency value	Jump
TO DO	Low frequency filtration with 2-10%	



VAR = 526,000

How to overcome the negative influence of outliers & jumps

Type of noise	Outlier, low frequency value	Jump
TO DO	Low frequency filtration with 2-10%	

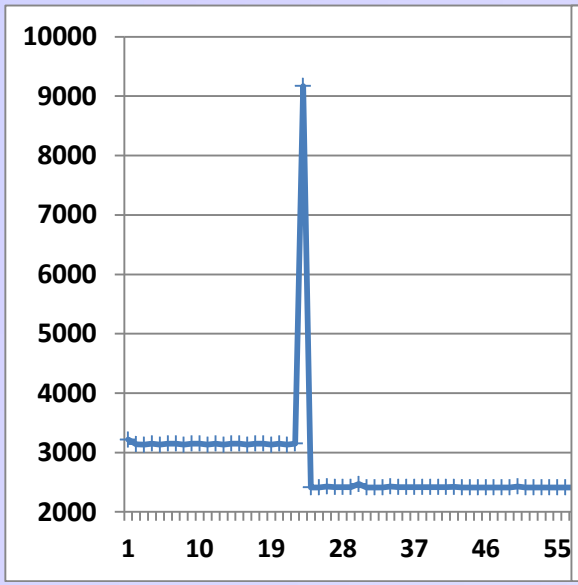


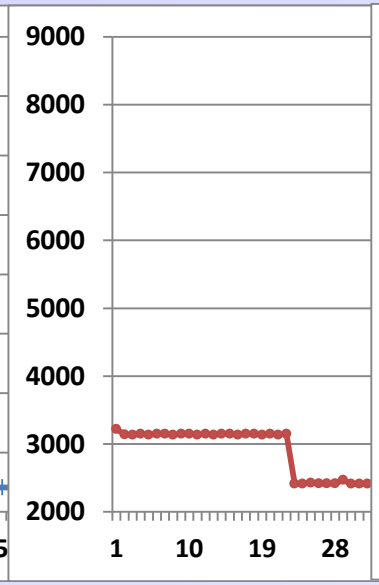
VAR = 526,000 **VAR = 93,000**

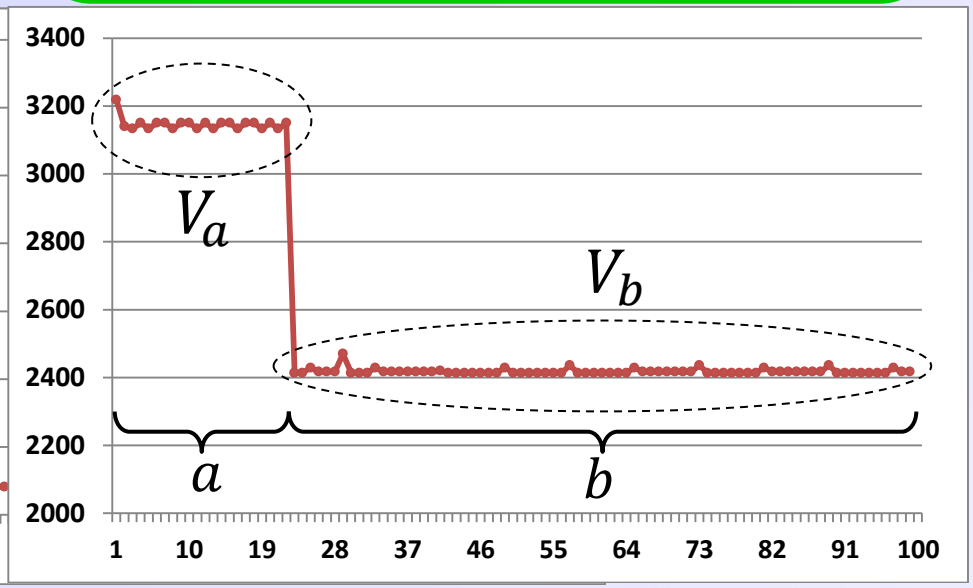
without an outlier

How to overcome the negative influence of outliers & jumps

Type of noise	Outlier, low frequency value	Jump
TO DO	Low frequency filtration with 2-10%	$V = \frac{a}{a+b} * V_a + \frac{b}{a+b} * V_b$



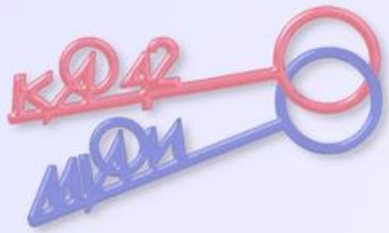




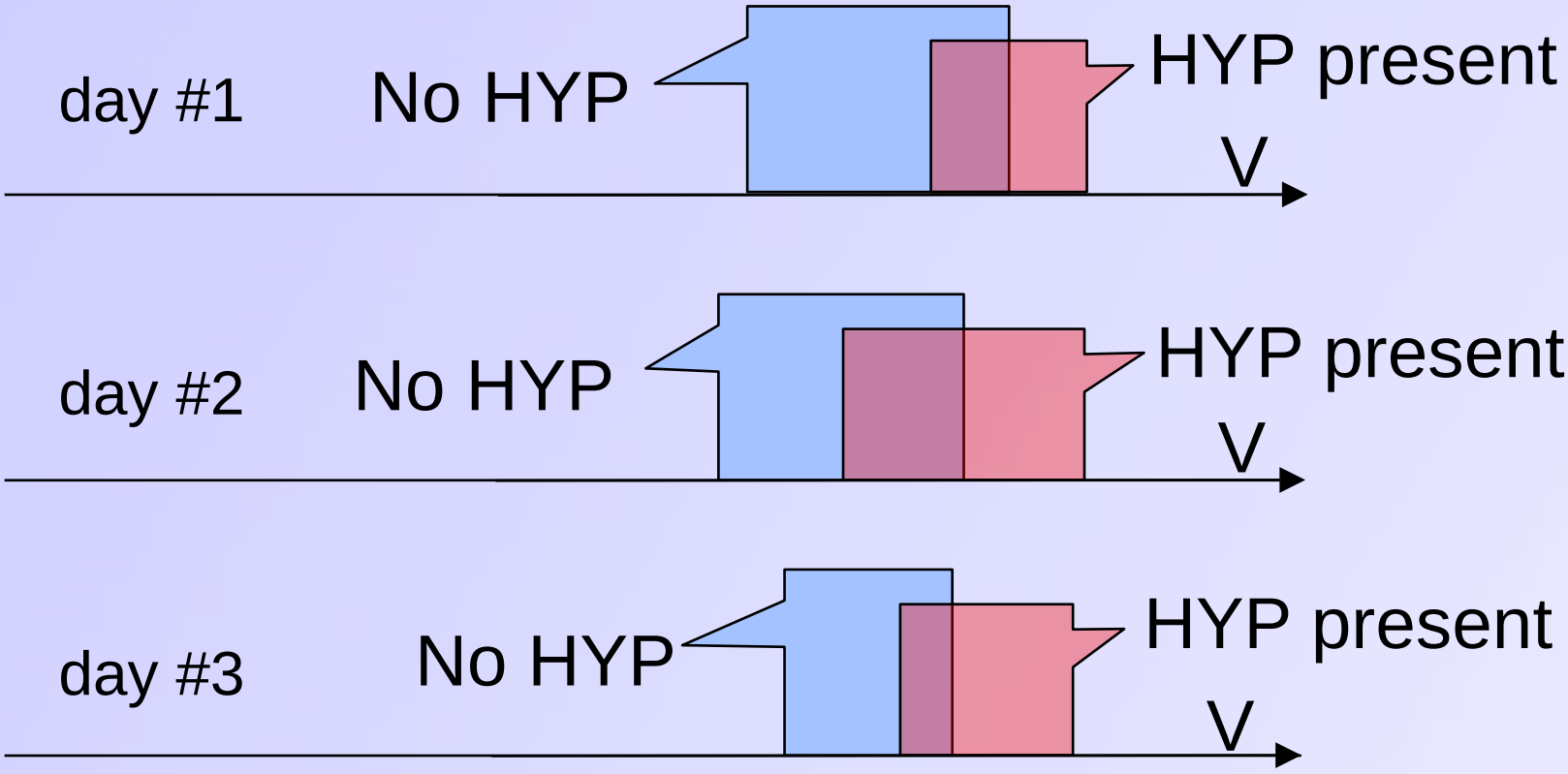
VAR = 526,000 → **without an outlier** → **VAR = 93,000** → **without a jump** → **VAR = 123**

I decided to test these ideas &

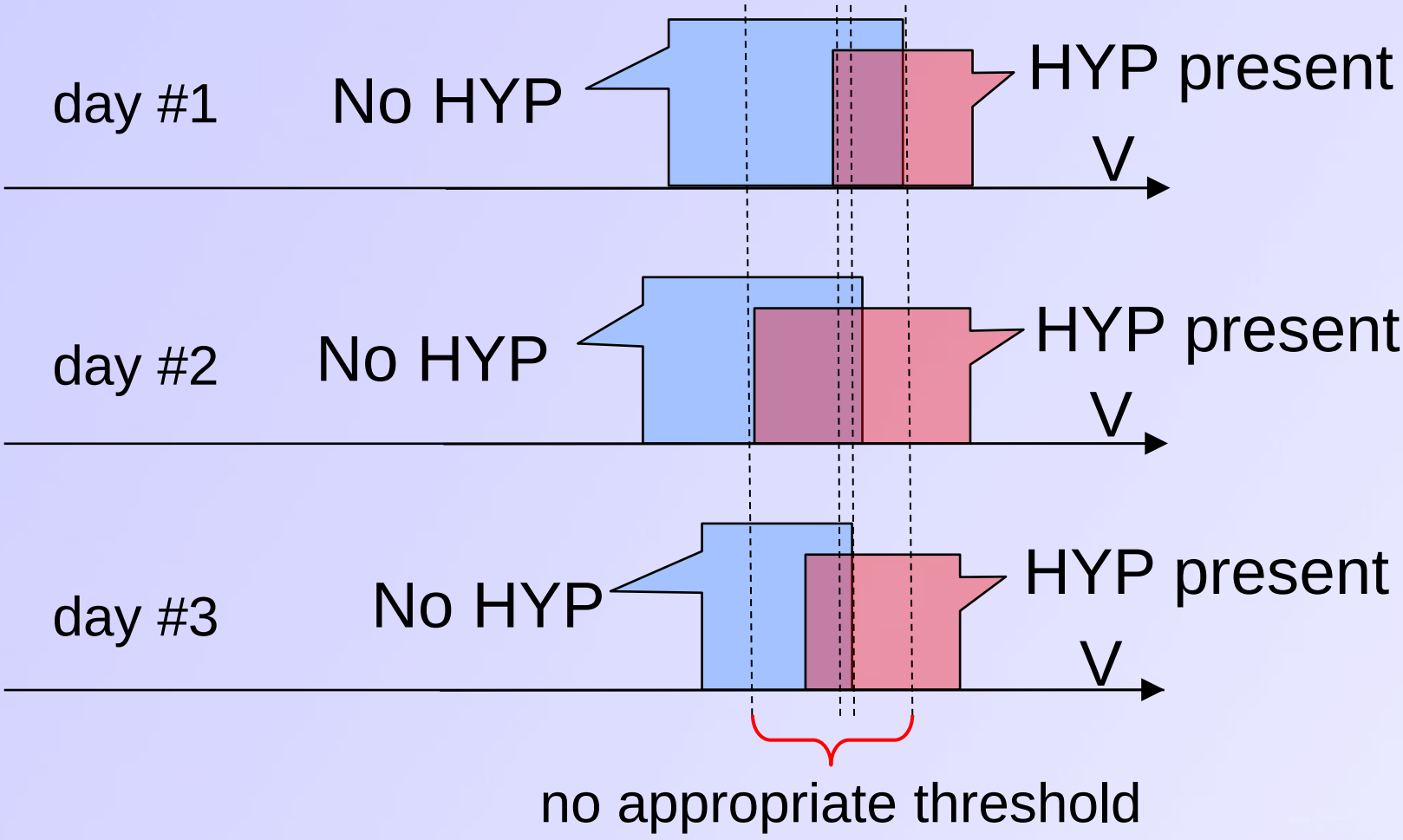
try to detect a HYP <u>every day</u>



Obtain different statistical values on different days



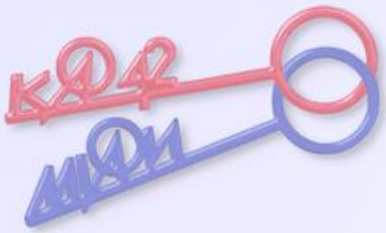
Obtain different statistical values on different days



Data fluctuation: lack of repeatability

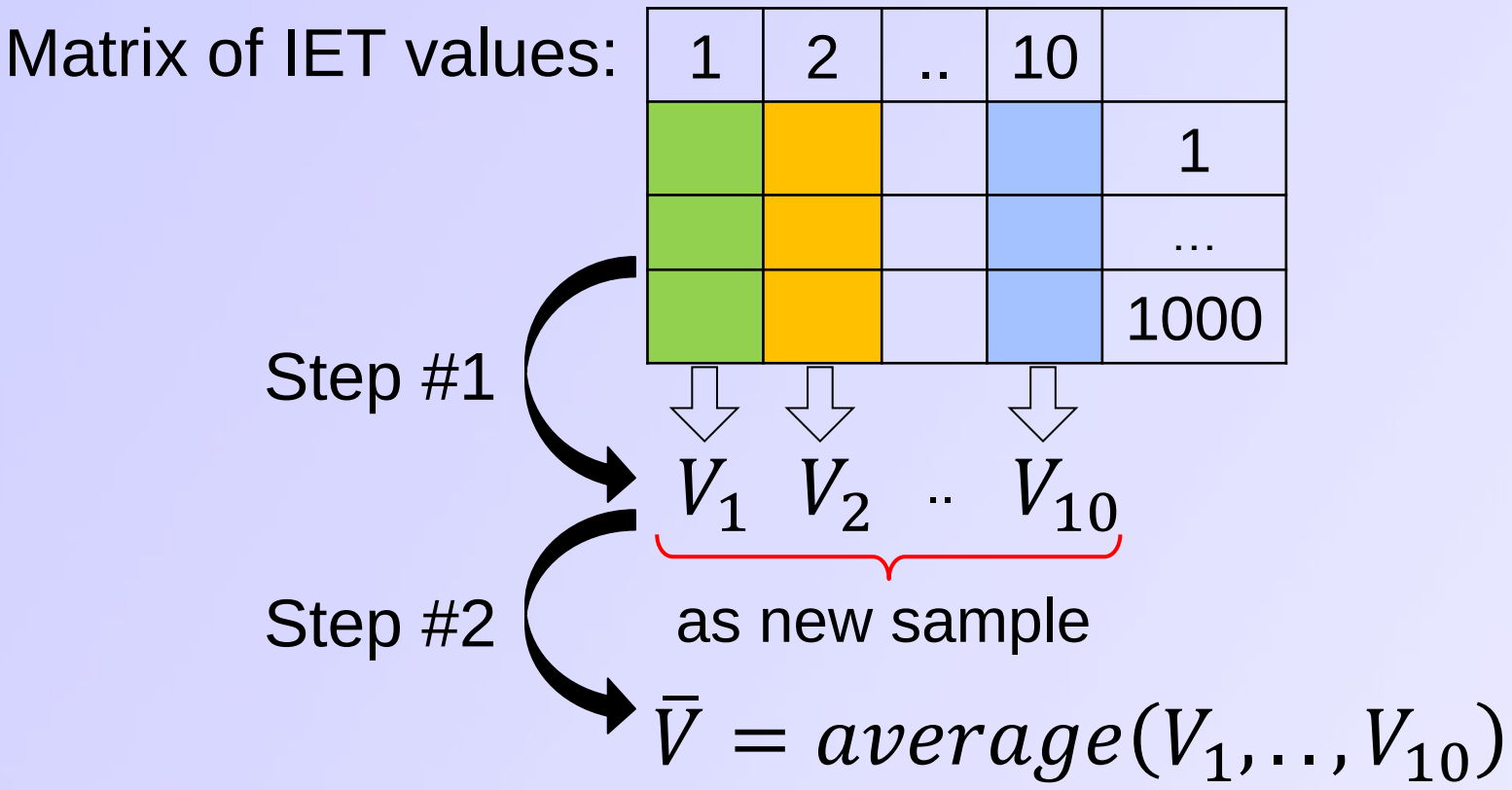
How to overcome this

data fluctuation every day?



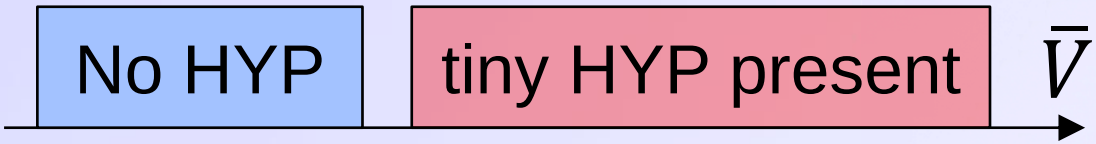
Overcoming the lack of repeatability

1. Two-step way to calculate statistics $\bar{V}$:

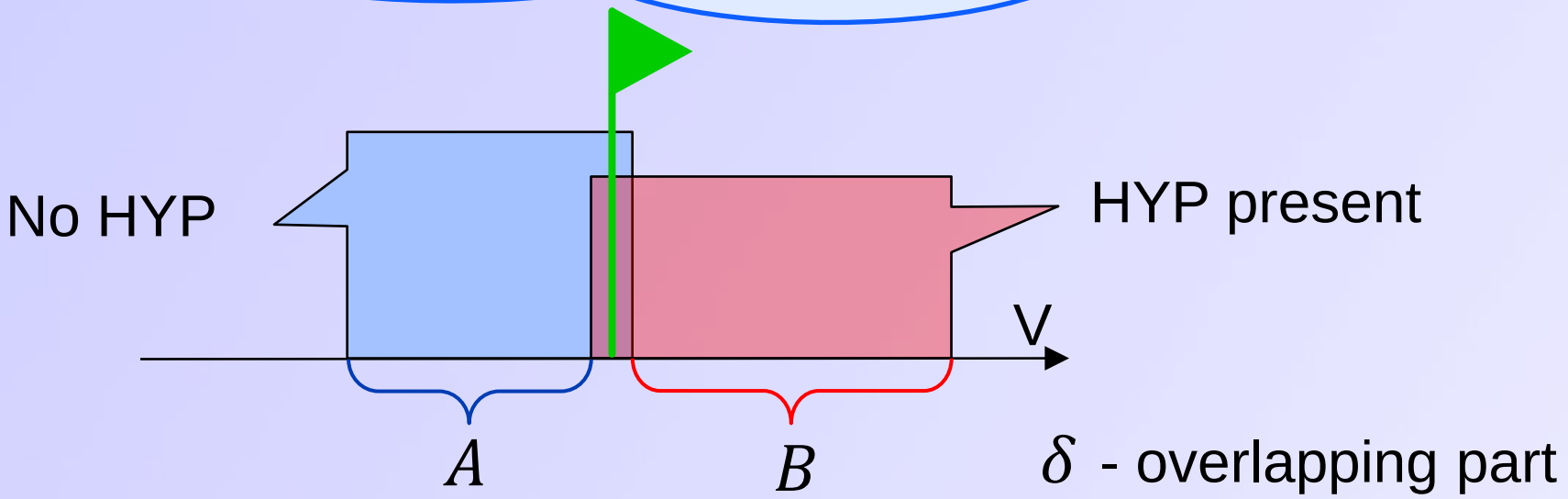


2. Repeat measurements within 10 days

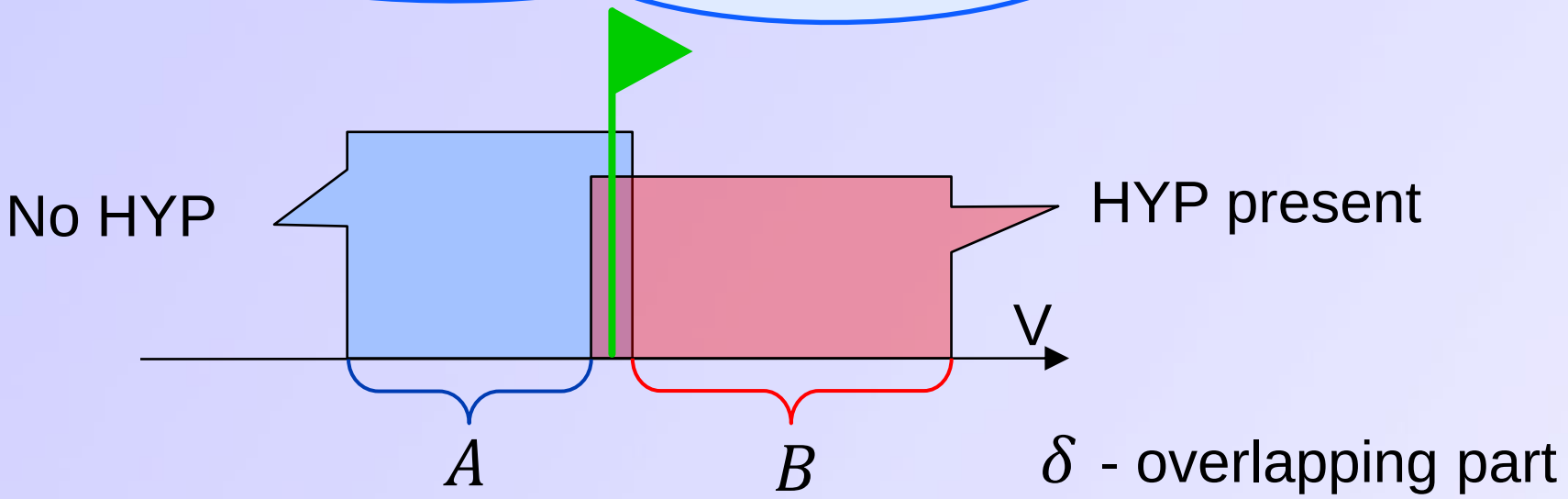
As a results:



What can we do if variation intervals keep overlapping?



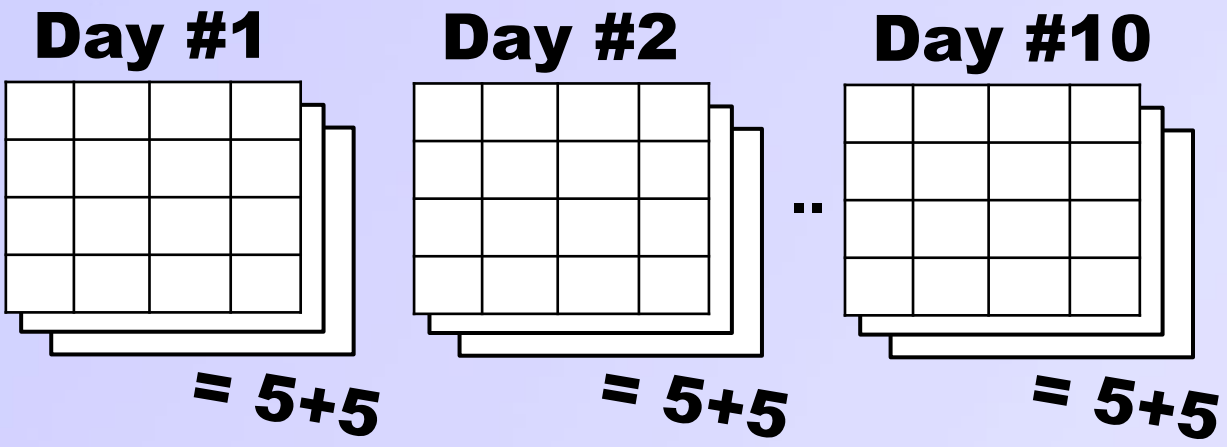
What can we do if variation intervals keep overlapping?



Type errors	Decision	Reality	Probability
I	HYP is present	no NYP	$\alpha = \delta \,/\, A$
II	no NYP	HYP is present	$\beta = \delta \,/\, B$

→ repeat data acquisition & stats calculation

Threshold values calculation

1. 

Matrices count	
no HYP	50
tiny HYP	50

2. Calculate two-step way statistics after filtration

3. Choose threshold values so that the sum of probability of type I and II errors comes to its min

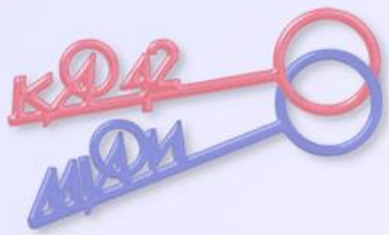
Example of threshold values

Intel Core 2 Duo E6300 + Windows 7 x32

Statistics	Filtration level	Threshold values		Type I error, %	Type II error, %
		No HYP	HYP is present		
Number of layers	0	≤ 7	≥ 8	4	0
Variance	0	≤ 14	≥ 18	2	0
Moment	0.1	≤ 679	≥ 947	2	0

How to detect stealthy hypervisors?

Step by step method:



How to detect stealthy hypervisors?

Stages	Stage description
Preliminary (calculate thresholds)	1. Flash BIOS with a trusted image or firmware 2. Install OS 3. Get threshold values in case where no HYP is present
Operational (detect a hypervisor)	4. Check in a loop if a hypervisor is present 5. Install Office etc 6. Monitor messages about a hypervisor presence 7. Go to step 3 to adapt the tool to new legitimate HYP

How to detect stealthy hypervisors?

Stages	Stage description
	1. Flash BIOS with a trusted image or firmware
	2. Install OS
	. Get thr[illegible]nt
	[illegible]k

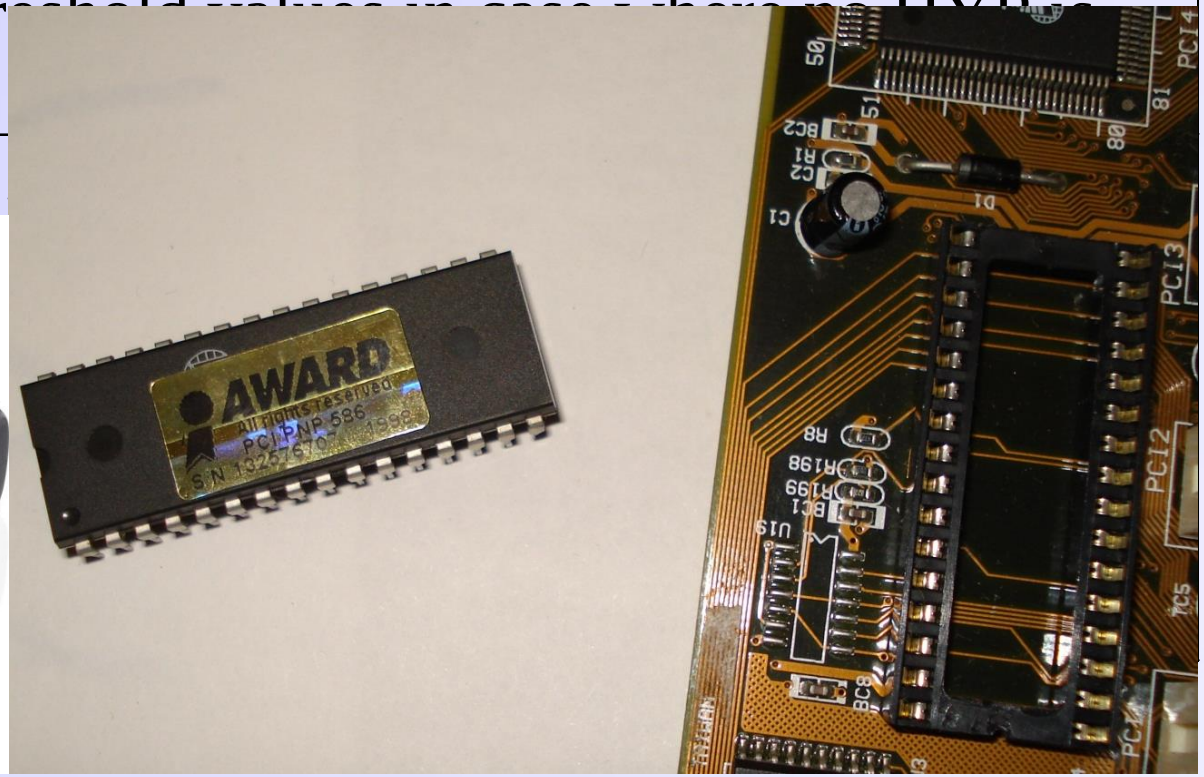
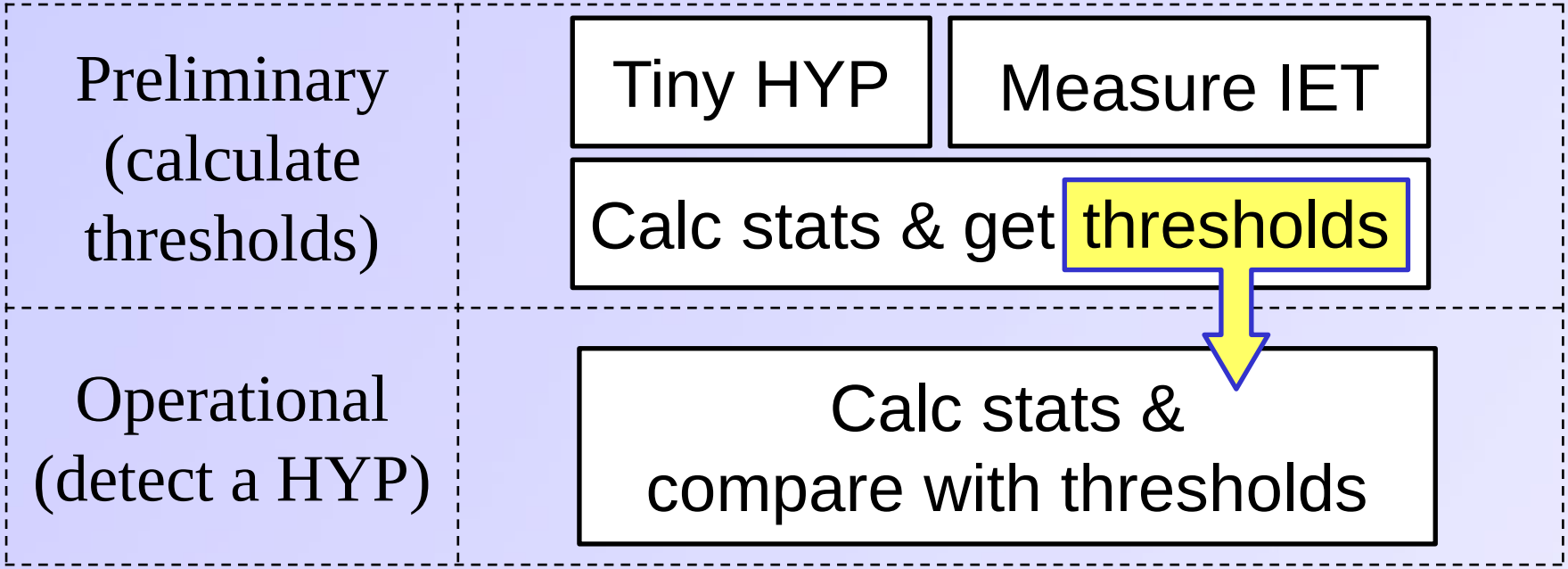


Image sources: wikipedia.org/wiki/BIOS batronix.com/versand/programmiergeraete/BX32P/index.html http://myonsitetech.ca/images/image/SoftwareUpgrade.png

How to detect stealthy hypervisors?

Stages	Stage description
Preliminary (calculate thresholds)	**1. Guarantee the absence of a HYP by checking a scatter plot (coming soon)** 2. Get threshold values in case where no HYP is present
Operational (detect a hypervisor)	3. Check in a loop if a hypervisor is present 4. Install Office etc 5. Monitor messages about a hypervisor presence 6. Go to step 3 to adapt the tool to new legitimate HYP

Detection: architecture & source code



Source code components		Details
Tiny HYP Measure IET	Windows x32 drivers & their config tools	Visual Studio & WDK, C++ asm
Calc stats & get thresholds		Matlab

http://github.com/IgorKorkin/HypervisorsDetection

Positive results on different PCs & HYPs

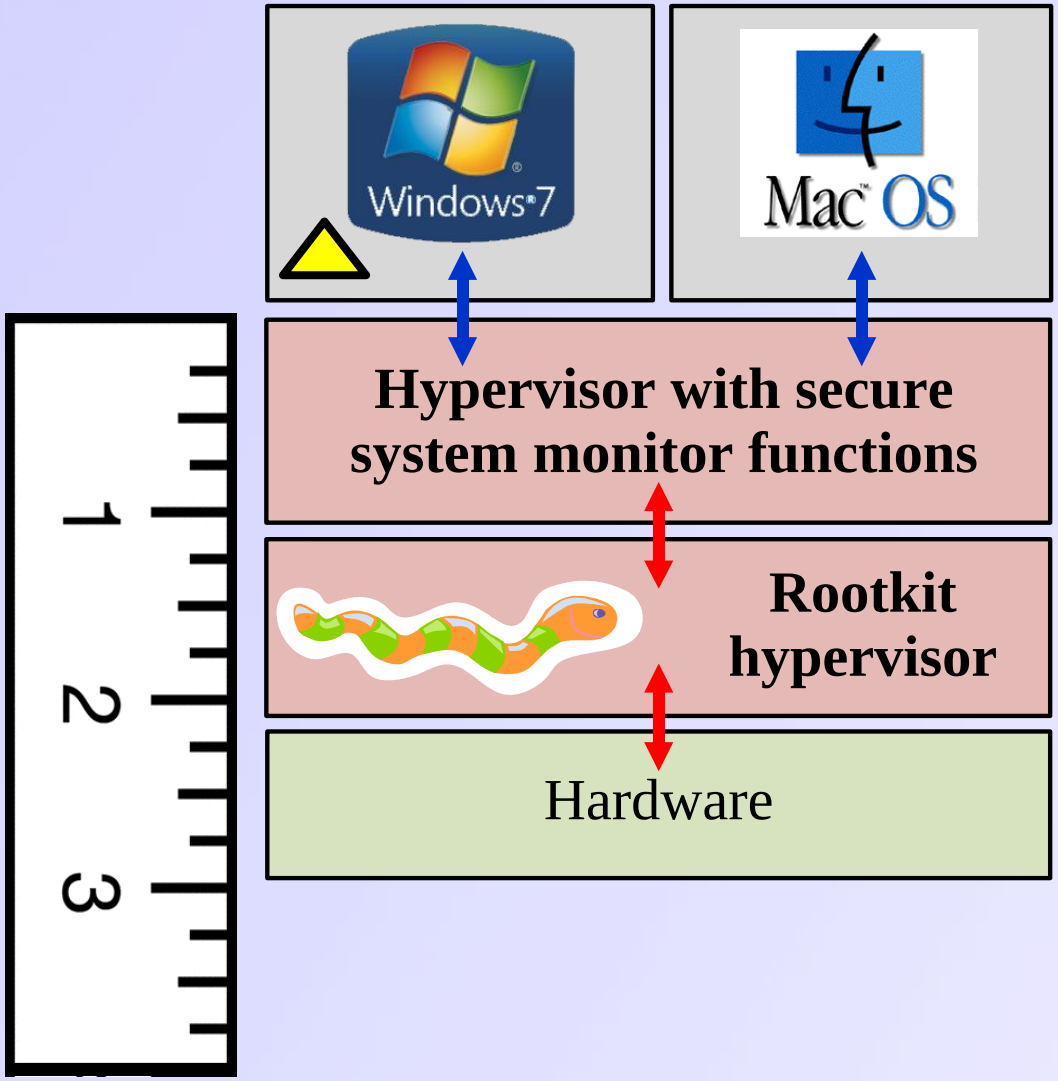
Is run by	HYP title	HYP authors & details	CPU
Driver	Only 1 tiny HYP	tested on 5 PCs	
	2 nested HYPs= ADD* + tiny HYP	ADD is loaded first, the tiny HYP is above it	
BIOS	TRace EXplorer (TREX)	A.Tichonov & A.Avetisyan (ISP RAS)	
	Russian Ghost	A.Lutsenko aka R_T_T	

ADD — Acronis Disk Director for Windows x86

List of challenges

Challenges	How to achieve
Stealthy HYP cheats time	Use variability indexes of IET
Data fluctuation: jumps & outliers	Apply filtration & two-step way statistics
Lack of repeatability	Repeat measurements within 10 days
And also:	
IET is not normally distributed & no HOV	Use Kornfeld method

“Statistical ruler” detects stealthy HYPs



- the detection tool is running in the background