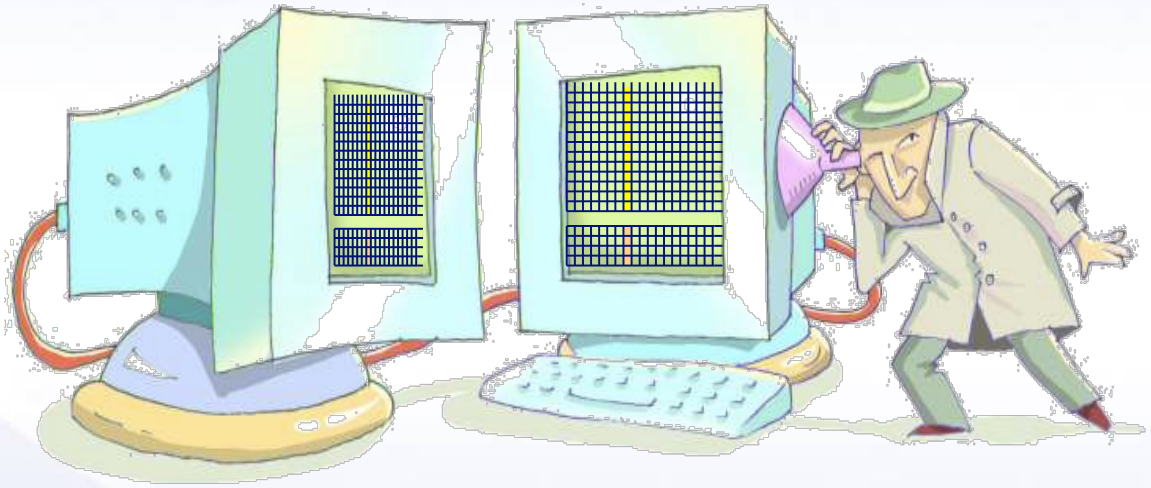
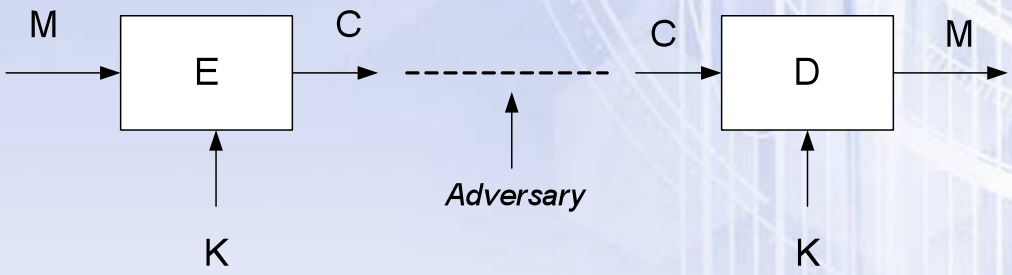
Predicting Secret Keys via Branch Prediction

How to Spy on Other People's Keys
on
Commodity PC Platforms

Çetin Kaya Koç



Classical Model of Cryptography



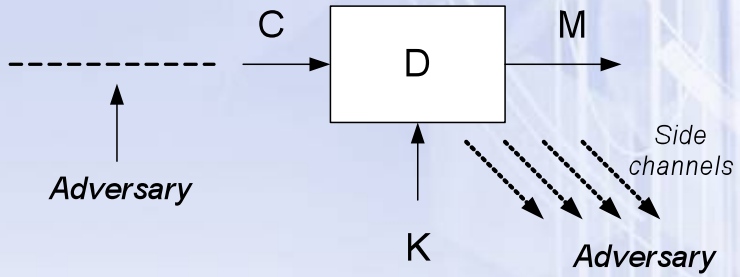
♪

Basic Assumptions:

- Adversary knows the algorithm
- Adversary can intercept the ciphertext

- Further Scenarios:
  - Known ciphertext attack: C_1, C_2, C_3, …
  - Known plaintext attack: (M_1, C_1), (M_2, C_2), …
  - Chosen plaintext attack: (M'_1, C_1), (M'_2, C_2), …
  - Chosen ciphertext attack: (M_1, C'_1), (M_2, C'_2), …
  - …

Side-Channel Model of Cryptography



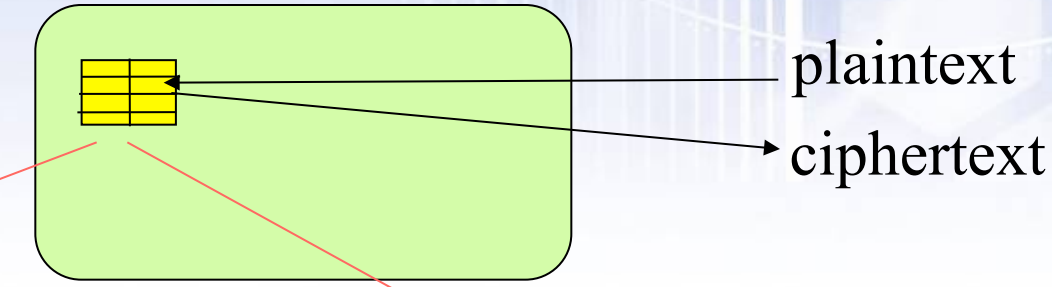
Cryptographic algorithms must run on a real device

- Devices have physical properties
- Devices will emanate information regarding cryptographic algorithm, key, and message
- Adversary having access to these side channels will extract information
  - Timing
  - Power
  - Electromagnetic

} Side channels

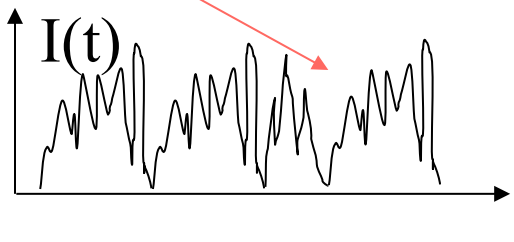
Side-Channel Cryptanalysis

- A new area of applied cryptography
- The study of breaking cryptosystems using side-channel information



t_i (running time)

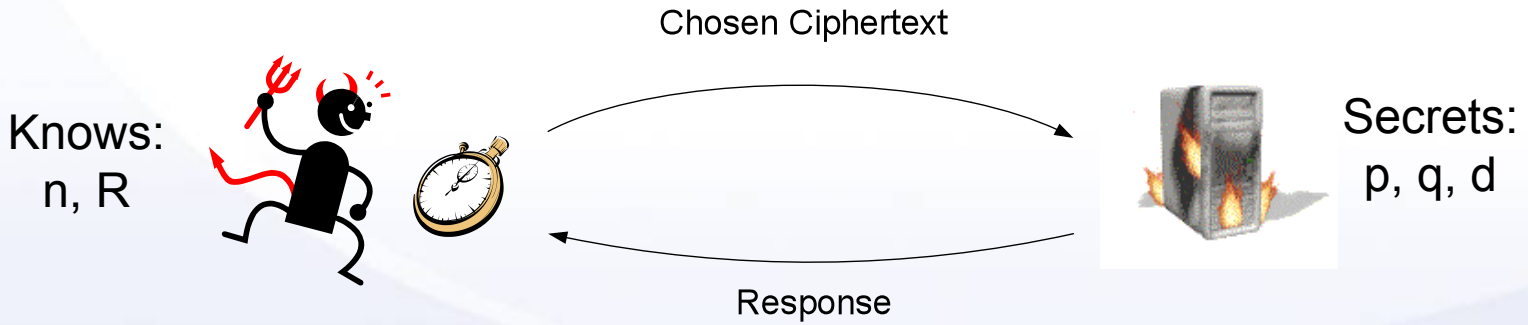
Timing attacks exploit time differences occurring for various input values



Power attacks exploit the instantaneous power consumption

Side-Channel Attacks on *Computers*

- Historical targets of side-channel attacks: smart cards
- The vulnerability of computer systems (e.g., remote servers) was not known until Brumley and Boneh Attack (2003)
- This remote timing attack on RSA (OpenSSL implementation on a web server) shows the practicality of such attacks
- The attack revealed 1024-bit RSA key of a server over a LAN
- Remote RSA attack was improved by Acıiçmez, Schindler, and Koç, now requiring < 100k queries (while the original attack needed 1.4 million queries)



Side-Channel Attacks on *Computers*

- Recently, the side-channel attacks hit the PC as a new victim platform:
  - Especially interesting since maturing Trusted Computing efforts promise a “trusted environment” with isolated execution for applications, etc.
- These new side-channel attacks are different from embedded platforms
  - The PC platform environment is quite different from the embedded security platforms.
  - Only pure “unprivileged” software-based attacks are really interesting.
- Timing variances exploited in previous timing attacks are caused by:
  - Different data-dependent execution paths
  - Different (number of) instructions executed in those paths
- Micro-architectural attacks exploit timing and access variations caused by the components of the CPU (even if the same sequence instructions are always executed)

Micro-Architectural Analysis

• Micro-Architectural Side-Channel attacks are a special new class of attacks that exploit the microarchitectural and throughput-oriented internal functionality of modern processor components.

• Micro-Architectural attacks exploit the execution time / power consumption variations caused by CPU components

• Currently there are 4 types of Micro-Architectural Attacks:
  - Cache Analysis
  - Branch Prediction Analysis
  - Instruction Cache
  - Shared Functional Units

• These attacks capitalize on the situations where several applications share the same processor resources, and the shared usage between spy and crypto process allows a spy process
  - **running in parallel to the victim process**

to extract critical information like secret keys.

Micro-Architectural Analysis

- On powerful PC-platforms many applications can run in parallel:
  - Either quasi-parallel enabled by OS scheduling, or
  - More or less explicitly parallel depending on the degree of additional hardware:
    - **Dual Processors, Dual Cores, Simultaneous Multi Threading, …**

- Thus, several applications share the same processor and its resources, and also at more or less the same time.

- Therefore, when a highly critical crypto algorithm is executed, there is the potential threat that a malicious or so called spy process is executed in parallel with the crypto process which might try to extract critical or secret information by “spying” on the crypto process during its execution.

Related and Previous Work

…

[Hu 1992] – Covert channels by caches

[Trostle 1998] – Cache attack against trusted keyboard input

…

[Page 2002] – Theoretical cache attacks via power trace

[Tsunoo Tsujihara Minematsu Miyuachi 2002],
[Tsunoo Saito Suzaki Shigeri Miyauchi 2003] – Timing attacks via internal cache collisions

…

[Bernstein 2004] – Pure timing attack on AES

[Percival 2005] – Cache attack on RSA

[Tromer Shamir Osvik 2005/2006] – First work which fully demonstrated an efficient cache attack on AES in a real-life setting

[Neve Seifert 2006] – Improvements of Tromer Shamir Osvik AES cache attack

[Acıiçmez Koç Schindler 2007] – Remote cache attack on AES

[Acıiçmez Koç Seifert 2006/2007] – Branch Prediction Attacks

[Acıiçmez 2007] – Instruction Cache Attack

[Acıiçmez Schindler 2007] – A recent RSA vulnerability in OpenSSL due to MA

Basics of Cache Attacks

• Cache is a small and fast storage area used by the CPU to reduce the average time to access main memory.

• It stores copies of the most frequently used data.

• When a CPU needs to read a location in main memory, it first checks to see if the data is already in the cache.

- Cache Hit: data is already in the cache. CPU immediately uses this data in cache.
- Cache Miss: data is not in the cache. CPU reads it from the memory and stores a copy in the cache.

• Cache Line (Block): The minimum amount of data that can be read from the main memory into the cache at once.

  – Each cache miss causes a cache block to be retrieved from a higher level memory.

Basics of Cache Attacks

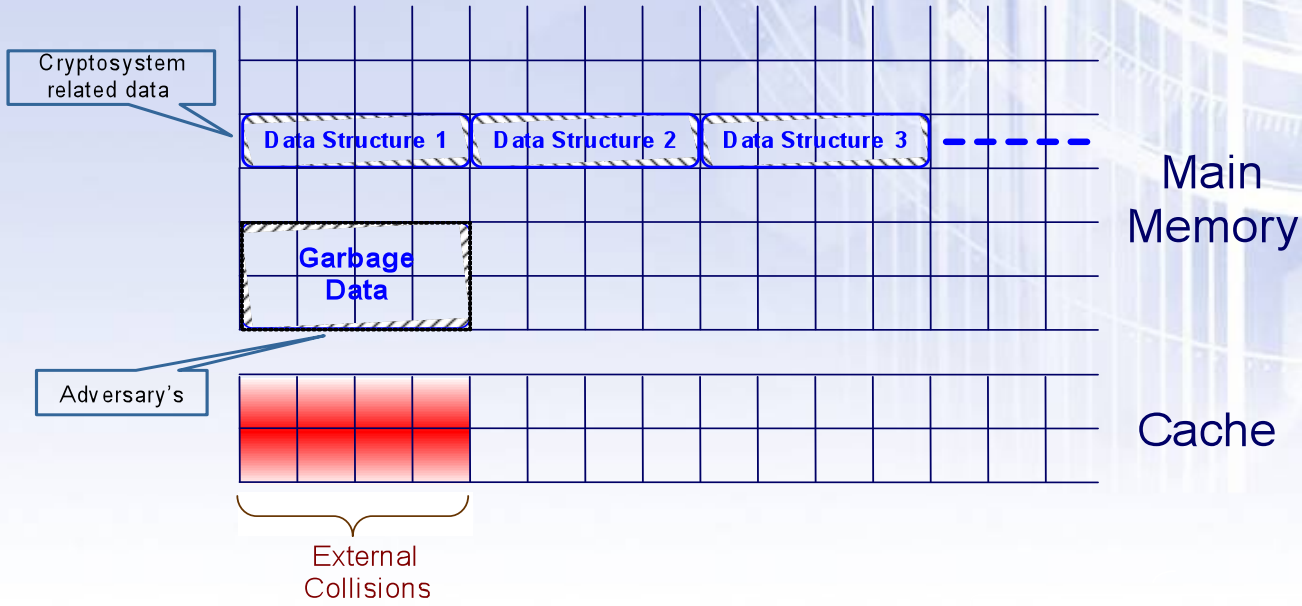
- Cache attacks exploit the cache behavior (i.e. cache hit/miss statistics) of cryptosystems

- Cache architecture leaks information about memory access patterns

- The sources of information leakage:
  - Execution Time: cache misses take more time to execute than a cache hit
  - Power Consumption: cache misses require more power than a cache hit

- Cryptosystems have data dependent memory access patterns. Once the access patterns are extracted, an adversary can recover the secret key

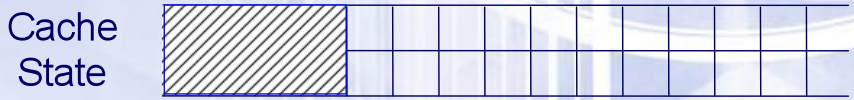
Basics of Cache Attacks



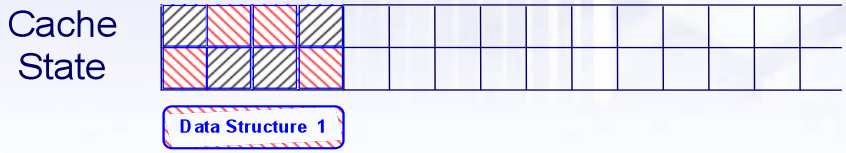
- An access to “Data Structure 1” may evict adversary’s data and vice versa
- An adversary can understand if/when “Data Structure 1” is accessed during the encryption

Basics of Cache Attacks

Adversary reads the garbage data via the "S*py Process*"



Case 1: "Data Structure 1" is accessed



Case 2: It is not accessed

Spy process reads the garbage data again.

- Case 1: takes more time to read it
- Case 2: takes less time to read it

Branch Prediction Attacks

- A very new software side-channel enabled by the branch prediction capability common to all modern CPUs.

- The penalty paid (extra clock cycles) for a mispredicted branch can be used for cryptanalysis of cryptographic primitives that employ a data-dependent program flow.

- BPA allows an unprivileged process to attack other processes running in parallel on the same processor

- Works despite of sophisticated partitioning methods such as memory protection, sandboxing or even virtualization

- Public-key ciphers like RSA and ECC are susceptible to BP attacks

Branch Prediction Attacks

- Superscalar processors have to execute instructions speculatively to overcome control hazards.

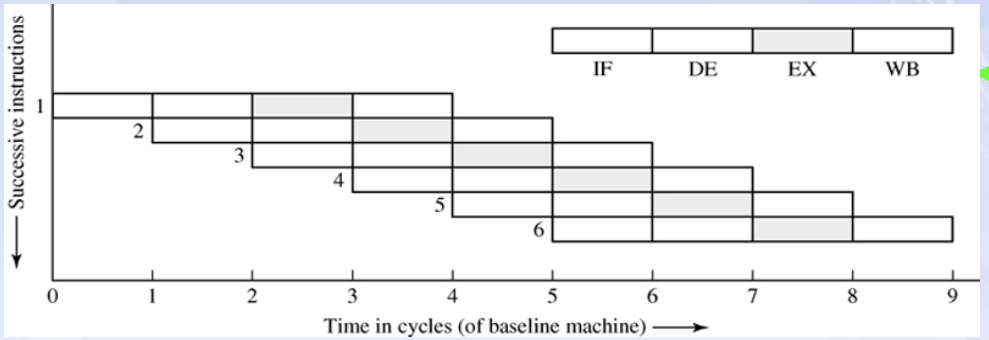
- Branch prediction units try to predict the most likely execution path after a branch.

- A *branch instruction* is a point in the instruction stream of a program where the next instruction is not necessarily the next sequential one.

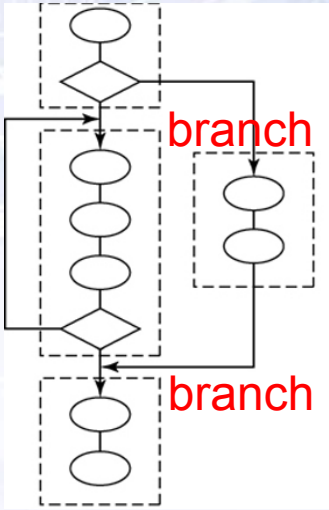
- For conditional branches, the decision to take the branch or not to take depends on some condition that must be evaluated in order to make the correct decision.

- During this evaluation period, the processor speculatively executes instructions from one of the possible execution paths instead of stalling and awaiting for the decision to come through.

Branch Prediction Attacks



Branch could stall pipeline!



Branch Prediction Unit

Basic Pentium III Processor Misprediction Pipeline

1	2	3	4	5	6	7	8	9	10
Fetch	Fetch	Decode	Decode	Decode	Rename	ROB Rd	Rdy/Sch	Dispatch	Exec

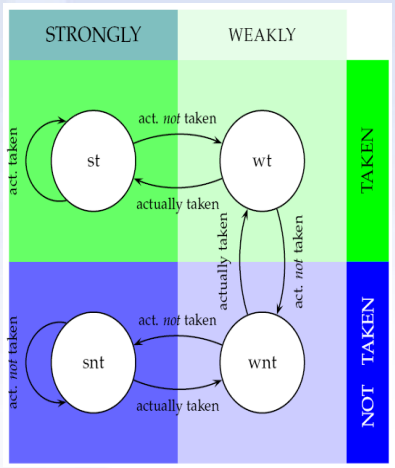
Basic Pentium 4 Processor Misprediction Pipeline

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Single misprediction in branch outcome: ≤ 150 cycles

Branch Prediction Attacks

• A **branch predictor** is that part of a processor that determines whether a conditional branch in the instruction flow of a program is likely to be taken or not.



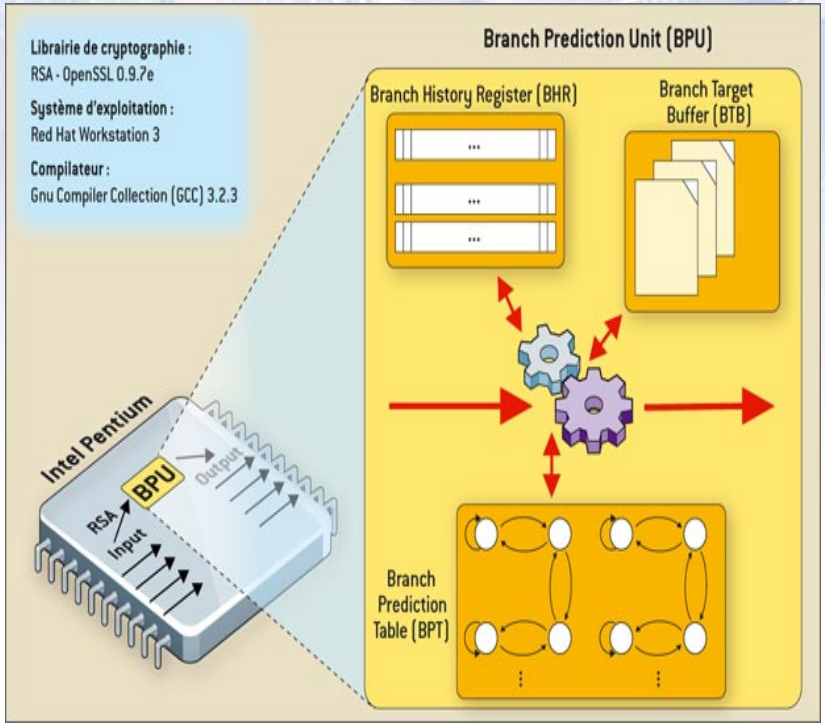
- This is called branch prediction.
- Branch predictors are crucial in today's modern, superscalar processors for achieving high performance.
- They allow processors to fetch and execute instructions without waiting for a branch to be resolved.
- Almost all pipelined processors do branch prediction of some form, because they must guess the address of the next instruction to fetch before the current instruction has been executed.

• Branch prediction is not the same as branch target prediction.

• **Branch target prediction** attempts to guess the target of the branch or unconditional jump before it is computed by parsing the instruction itself.

Branch Prediction Attacks

- BPU consists of mainly two “logical” parts: the branch target buffer (BTB) and the predictor.
- BTB is the buffer where the CPU stores the target addresses of the previously executed branches.
- BTB is limited in size, the CPU can store only a number of such target addresses, and previously stored addresses are evicted from the BTB if a new address needs to be stored instead.
- The predictor is that part of the BPU that makes the prediction on the outcome of the branch♪



Branch Prediction Attacks

- A very new group of software side-channel attacks developed by Acıiçmez, Seifert, and Koç in 2006

- They use the branch misprediction delays to break cryptographic primitives that employ a data-dependent program flow (e.g. RSA and ECC)

- BPA also allows an unprivileged process to attack other processes running in parallel on the same processor even in the presence of sophisticated partitioning methods such as memory protection, sandboxing or even virtualization

- Several BP attacks are proposed:
  - Attack 1 - Exploiting the Predictor directly (Direct Timing Attack)
  - Attack 2 - Forcing the BPU to the Same Prediction (Asynchronous Attack)
  - Attack 3 - Forcing the BPU to the Same Prediction (Synchronous Attack)
  - Attack 4 - Trace-driven Attack against the BTB (a.k.a. Simple Branch Prediction Attack)

♪

Branch Prediction Attacks

- Relies on the fact that the prediction algorithms are deterministic (i.e. predictable)

- Assume that an adversary attacks an RSA cipher with a private exponent *d* and knows the first *i* bits of *d* and is trying to reveal d_i.

- Recall Montgomery Multiplication (MM) and the extra reduction step:

$$
\begin{aligned}
&S = A * B \\
&S = (S - (S * N^{-1} \bmod R) * N)/R \\
&\textbf{if } S > N \textbf{ then } S = S - N \\
&\textbf{return } S
\end{aligned}
$$

Branch Prediction Attacks

- For any message *m*, the adversary can simulate the first *i* steps of the operation and obtain the intermediate result that will be the input of the $(i+1)^{th}$ squaring.

- Then, the attacker creates 4 different sets M_1, M_2, M_3, and M_4, where

$$M_1 = \{m \mid m \text{ causes a misprediction during MM of } (i+1)^{\text{th}} \text{ squaring if } d_i = 1\}$$
$$M_2 = \{m \mid m \text{ does not cause a misprediction during MM of } (i+1)^{\text{th}} \text{ squaring if } d_i = 1\}$$
$$M_3 = \{m \mid m \text{ causes a misprediction during MM of } (i+1)^{\text{th}} \text{ squaring if } d_i = 0\}$$
$$M_4 = \{m \mid m \text{ does not cause a misprediction during MM of } (i+1)^{\text{th}} \text{ squaring if } d_i = 0\}$$

- If the difference between timing characteristics, e.g., average execution time, of M_1 and M_2 is more significant than that of M_3 and M_4, then he guesses that *di* = 1. Otherwise *di* is guessed to be 0

Branch Prediction Attacks

- We assume that the cipher runs on an SMT machine and the adversary can run a dummy process simultaneously with the cipher process.

- Adversary can clear the BTB via the operations of the dummy process and causes a BTB miss during the execution of the target branch.

- BPU automatically predicts the branch not to be taken if it misses the target address in the BTB.

- There will be a misprediction whenever the actual outcome of the target branch is `taken‘.

- In Attack-1, adversary has to know details of prediction algorithm. We remove this necessity on SMT systems.

Branch Prediction Attacks

- If the adversary finds a way to establish a synchronization:
  - he can determine for (e.g.) the *i* th step of the exponentiation and can clear the BTB just before the *i* th step
  - he can introduce misprediction delays at certain points during the computation

- Assume that the RSA implementation employs S&M exponentiation and the if statement in S&M exponentiation (c.f. Line 4) is used as the target branch.

```
S = M
for   i from 1 to n − 1 do
      S = S ∗ S (mod N)
      if   d_i = 1 then
           S = S ∗ M (mod N)
return S
```

Branch Prediction Attacks

- The adversary runs RSA for a *known* plaintext and measures the execution time.

- Then he runs it again for the same input but this time he clears the single BTB set *during* the encryption just before the i^{th} execution of the conditional branch of Line 4

- This conditional branch is taken or not taken depending only on the value of d^i.

- If it turns out to be taken, the second encryption will take longer time than the first execution because of the misprediction delay.

- Therefore, the adversary can easily determine the value of this bit by successively analyzing the execution time.

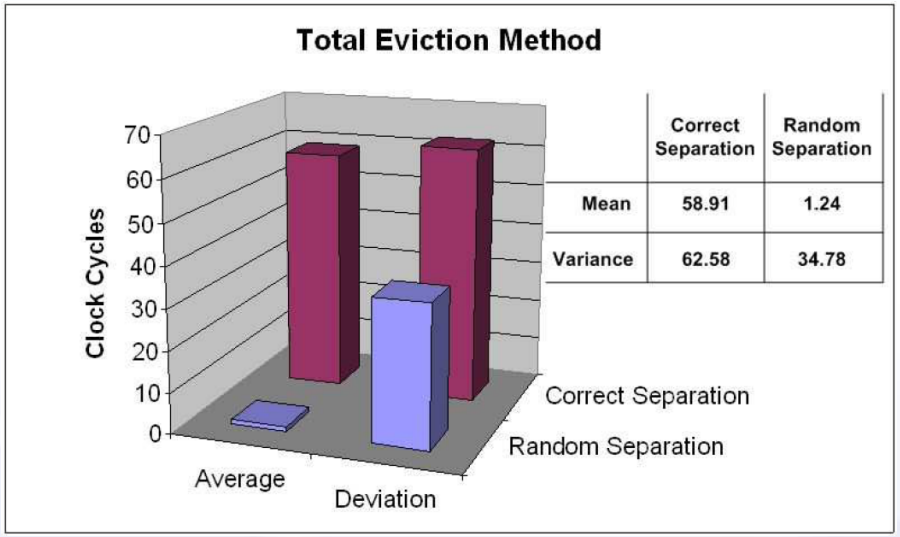
Branch Prediction Attacks

- In the previous three attacks, we have considered analyzing the execution time of the cipher. In this attack, we will follow a different approach : analyzing spy process' execution time
- Spy process continuously executes unconditional branches and all of these branches map to the same BTB set with the conditional branch under attack
- The adversary starts the spy process before the cipher, so when the cipher starts encryption/signing, the CPU *cannot* find the target address of the target branch in BTB and the prediction *must* be *not-taken*
- If the branch turns out to be taken, then a misprediction will occur and the target address of the branch needs to be stored in BTB. When the spy-process re-executes its branches, it will encounter mispredictions
- Adversary can simply determine the complete execution flow of the cipher process by continuously performing the same operations

Branch Prediction Attacks

- The figure shows us the statistics of the execution time in two different cases:
  - The separation under the correct assumption (Correct S.)
  - The separation under the wrong assumption (Random S.)

- There is a clear distinction between these two cases !!!
- Total Eviction Method: The spy clears the entire BTB



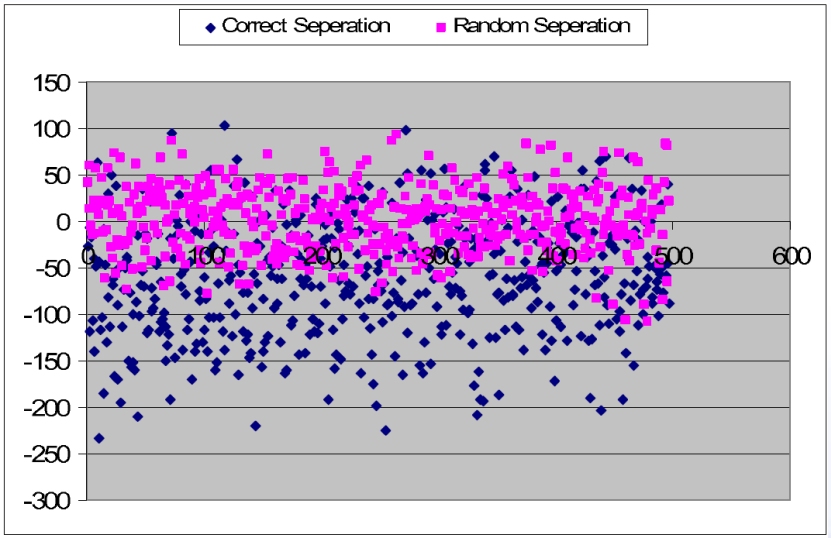
	Correct Separation	Random Separation
Mean	58.91	1.24
Variance	62.58	34.78

Branch Prediction Attacks

- The timing differences under the correct and random separations

- The results are the measured differences under a random key and a sample of plaintext.

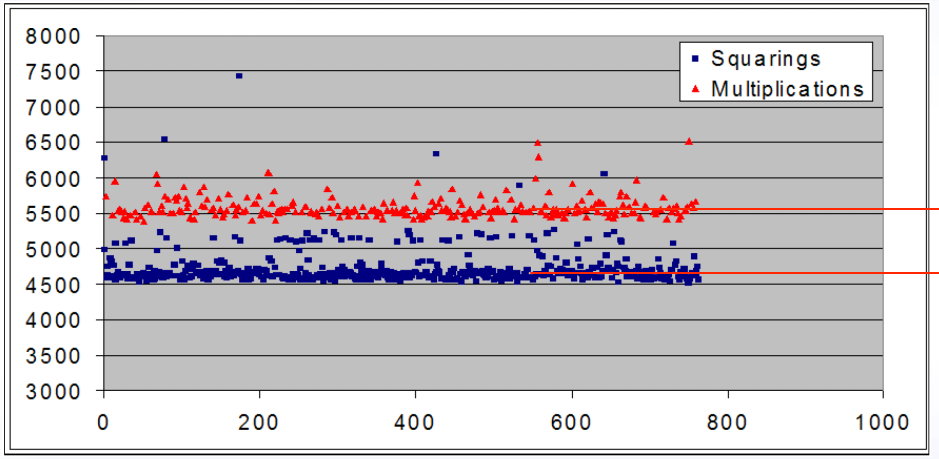
- The x-axis represent the location of the bit under consideration, i.e., *i* where *i* Є {0, 1, …, 511}



Branch Prediction Attacks

- The following result shows that a carefully designed machine-specific spy process is able to infer almost all bits of the secret key during a ***single*** RSA exponentiation.

- The results clearly visualize the difference between squaring and multiplication.

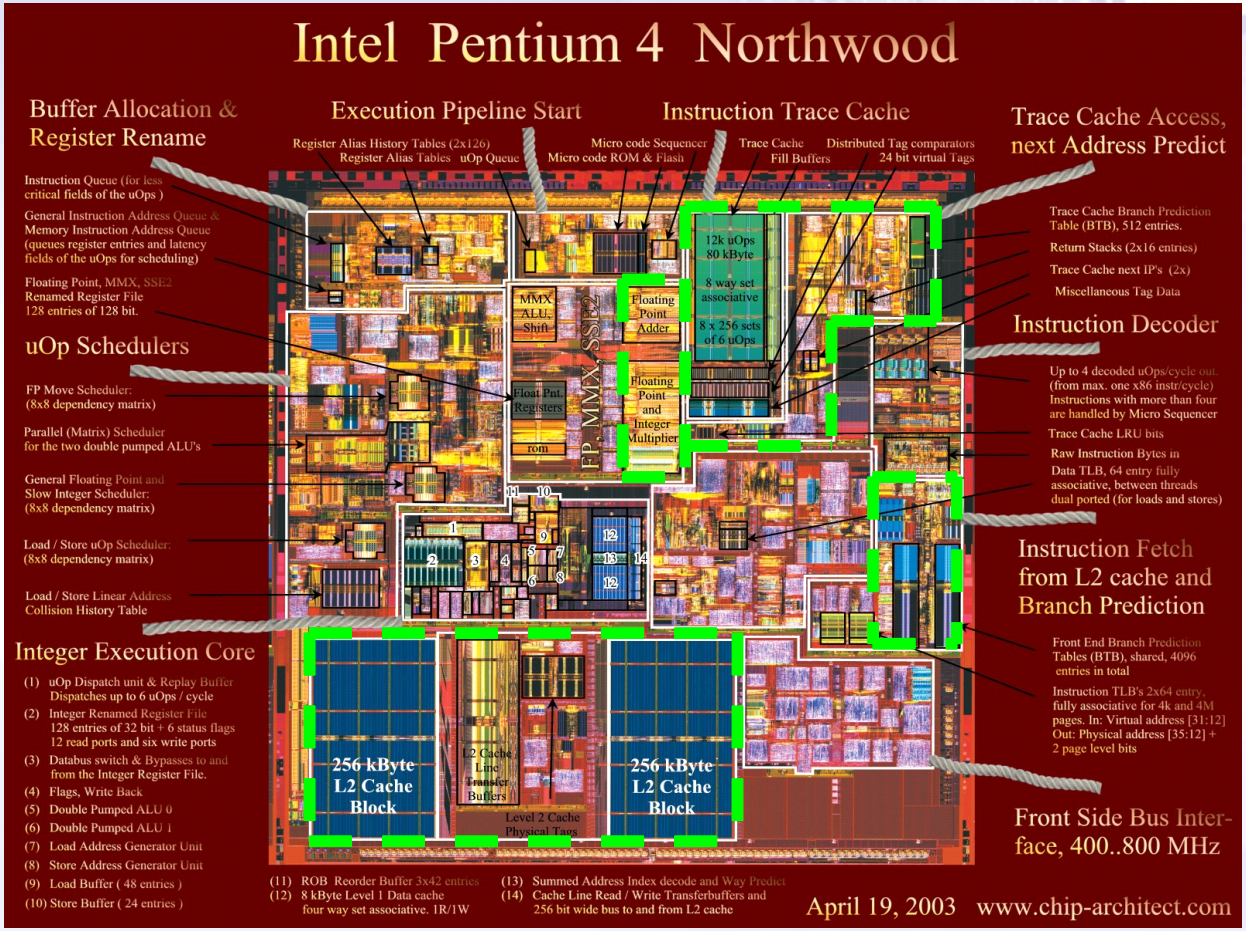


Implications of Micro-Architectural Attacks

- Works despite of sophisticated partitioning/protection methods such as memory protection, sandboxing, virtualization, etc.
- Impact:
  - Multiuser systems
  - VPNs
  - Virtual machines
  - Trusted computing
  - Sandboxes (JVM, JavaScript)
  - Remote attacks
  - …
- Easy to deploy – pure software
- Hard to detect
- Hard to protect efficiently

Future Directions

- Current known attacks exploit Data Cache, Branch Prediction Unit, Instruction Cache, and Shared Functional Units
- What are the other sources of Micro-architectural side-channel leakage?



Future Directions

- We have to develop effective countermeasures against micro-architectural attacks
  - Hardware countermeasures for microprocessor architects
  - Software countermeasures for software developers

- OpenSSL had gone into several revisions and implemented many software countermeasures

- Yet, new micro-architectural vulnerabilities of OpenSSL are being discovered

- Are software countermeasures sufficient?
  - Solving the problem in software means solving it one by one
  - Software solutions are algorithm- and attack-specific
  - They incur high performance overhead

Future Directions♪

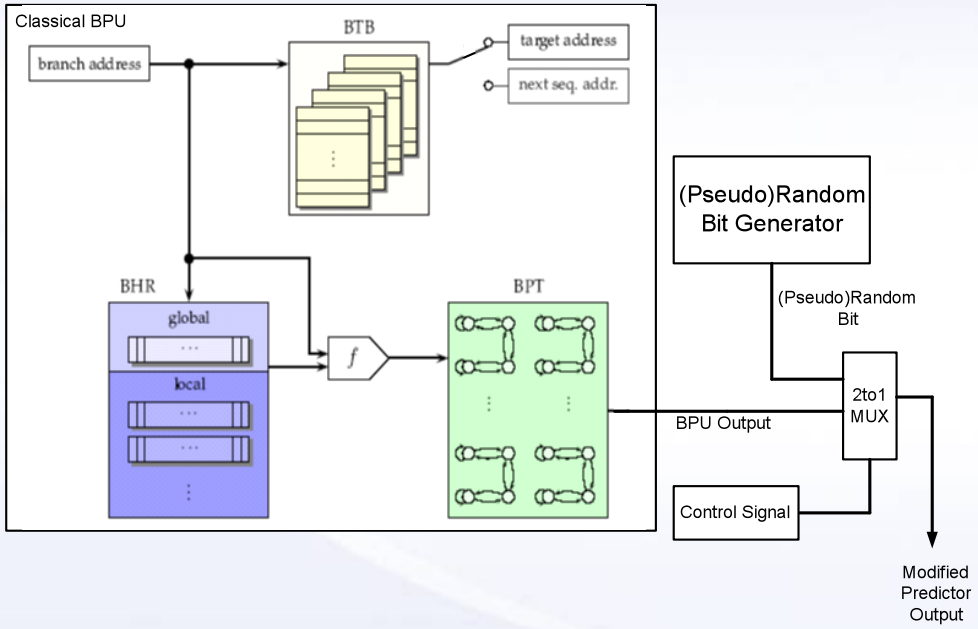
- Hardware countermeasures:
  - Solving it in hardware may mean solving it for all
  - They may not be algorithm-specific
  - They may have much less overhead

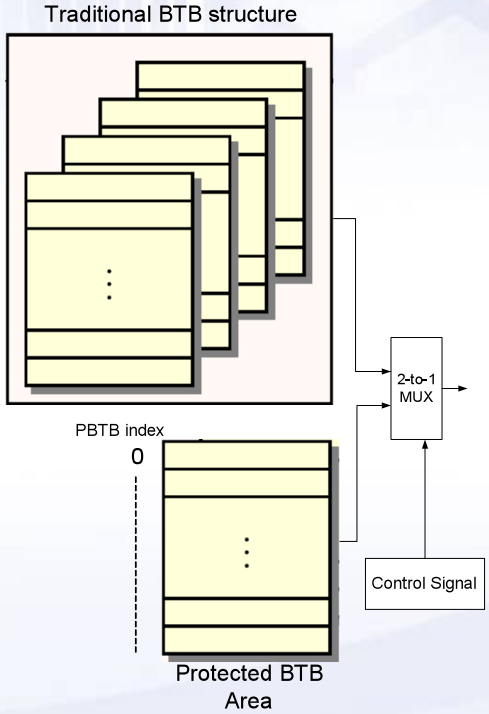
- Possible branch prediction countermeasures:
  - Randomizable branch prediction
  - Partitioned Branch Target Buffers
  - Locking mechanism for BTB
  - Protected BTB area
  - Flushing mechanism for BTB
  - Dynamically disabling branch predictions
  - …

♪

Future Directions♪

- Branch prediction hardware countermeasure examples:♪
  - Randomizable Prediction:
    - Modify the prediction output
    - BPU can output random predictions for critical branches
  - Protected Branch Target Buffer:
    - Allow critical code to benefit from using a protected buffer
    - The entries in PBTB can be handled in a more secure way





Future Directions

- Investigate interplay between system software and micro-architectural attacks

- What is the actual impact of these attacks on trusted computing?

- Trusted Computing Group's approach:
  - Hardware as the trust anchor
  - The TPM itself is perhaps not vulnerable to micro-architectural attacks
  - But the systems built on higher layers are susceptible
  - These attacks undermines isolation principle

- Microsoft's approach: Next Generation Secure Computing Base
  - To be studied …

Future Directions♪

- So far, we had focused on cryptosystem vulnerabilities. What else can be done with these attacks?
- Can applications snoop into one another's data?
- How about virtual machines?
  - Spy process can run in one virtual machine and look into the others
  - How can we achieve high degree of isolation between different VMs on a shared physical device?
  - Can hypervisors and virtual machine monitors solve the problem in software efficiently? For example, they can flush all caches before switching between virtual machines at the expense of performance degradation.

♪

REFERENCES

• O. Acıiçmez, Ç. K. Koç, and J.-P. Seifert. Micro-Architectural Cryptanalysis. IEEE Security and Privacy, volume 5 issue 4, pages 62-64, 2007

• O. Acıiçmez, Ç. K. Koç, and J.-P. Seifert. On the Power of Simple Branch Prediction Analysis. 2007 ACM Symposium on Information, Computer and Communications Security (ASIACCS'07), pages 312-320, 2007.♪

• O. Acıiçmez, Ç. K. Koç, and J.-P. Seifert. Predicting Secret Keys via Branch Prediction. RSA Conference'07, Cryptographers' Track (CT-RSA), pages 225-242, LNCS 4377, 2007.

• O. Acıiçmez, W. Schindler, and Ç. K. Koç. Cache Based Remote Timing Attack on the AES. RSA Conference '07, Cryptographers' Track (CT-RSA), pages 271-286, LNCS 4377, 2007.

• O. Acıiçmez and Ç. K. Koç. Trace-Driven Cache Attacks on AES. 8th International Conference on Information and Communications Security (ICICS'06), pages 112-121, LNCS 4307, 2006.

• O. Acıiçmez, W. Schindler, and Ç. K. Koç. Improving Brumley and Boneh Timing Attack on Unprotected SSL Implementations. Proceedings of the 12th ACM Conference on Computer and Communications Security (ACM CCS'05), pages 139-146, 2005.

• O. Acıiçmez, S. Gueron, and J.-P. Seifert. New Branch Prediction Vulnerabilities in OpenSSL and Necessary Software Countermeasures. IMA International Conference on Cryptography and Coding, pages 185-203, 2007

REFERENCES

• O. Acıiçmez and J.-P. Seifert. Cheap Hardware Parallelism Implies Cheap Security. Fault Diagnosis and Tolerance in Cryptography (FDTC'07), pages 80-91, IEEE Computer Society, 2007.

• O. Acıiçmez. Yet Another MicroArchitectural Attack: Exploiting I-Cache. Computer Security Architecture Workshop, pages 11-18. 2007.

• O. Acıiçmez and W. Schindler. A Major Vulnerability in RSA Implementations due to MicroArchitectural Analysis Threat. RSA Conference'08, Cryptographers' Track (CT-RSA), to appear

• D. J. Bernstein. Cache-timing attacks on AES. April, 2005.

• J. Bonneau and I. Mironov. Cache-Collision Timing Attacks against AES. Cryptographic Hardware and Embedded Systems - CHES 2006, to appear.

• G. Bertoni, V. Zaccaria, L. Breveglieri, M. Monchiero, and G. Palermo. AES Power Attack Based on Induced Cache Miss and Countermeasure. International Symposium on Information Technology: Coding and Computing - ITCC 2005, Volume 1, 4-6 April 2005.

• E. Brickell, G. Graunke, M. Neve, and J.-P. Seifert. Software mitigations to hedge AES against cache-based software side channel vulnerabilities. Cryptology ePrint Archive, Report 2006/052, February 2006.

REFERENCES

- D. Brumley and D. Boneh. Remote timing attacks are practical. Computer Networks, 48 (5):701–716, 2005.
- D. Brumley and D. Boneh. Remote Timing Attacks are Practical. Proceedings of the 12th Usenix Security Symposium, pages 1-14, 2003.
- C. Lauradoux. Collision attacks on processors with cache and countermeasures. Western European Workshop on Research in Cryptology - WEWoRC, pages 76-85, 2005.
- M. Neve and J.-P. Seifert. Advances on Access-driven Cache Attacks on AES. Selected Areas of Cryptography - SAC, 2006.
- M. Neve. Cache Vulnerabilities and SPAM analysis. PhD thesis, July 2006.
- M. Neve, J.-P. Seifert, and Z. Wang. A refined look at Bernstein's AES side-channel analysis. Proceedings of ACM Symposium on Information, Computer and Communications Security - ASIACCS'06, 2006.
- D. A. Osvik, A. Shamir, and E. Tromer. Cache Attacks and Countermeasures: The Case of AES. Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, pages 1-20, 2006.
- D. Page. Partitioned Cache Architecture as a Side Channel Defence Mechanism. Cryptography ePrint Archive, Report 2005/280, August 2005.
- D. Page. Defending Against Cache Based Side-Channel Attacks. Technical Report. Department of Computer Science, University of Bristol, 2003.

REFERENCES

- D. Page. Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel. Technical Report CSTR-02-003, Department of Computer Science, University of Bristol, June 2002.
- C. Percival. Cache missing for fun and profit. BSDCan 2005, Ottawa, 2005.
- Y. Tsunoo, T. Saito, T. Suzaki, M. Shigeri, and H. Miyauchi. Cryptanalysis of DES Implemented on Computers with Cache. Cryptographic Hardware and Embedded Systems - CHES 2003, pages 62-76, 2003.
- Y. Tsunoo, E. Tsujihara, K. Minematsu, and H. Miyauchi. Cryptanalysis of Block Ciphers Implemented on Computers with Cache. ISITA 2002, 2002.
- Y. Tsunoo, E. Tsujihara, M. Shigeri, H. Kubo, and K. Minematsu. Improving cache attacks by considering cipher structure. International Journal of Information Security, February 2006.