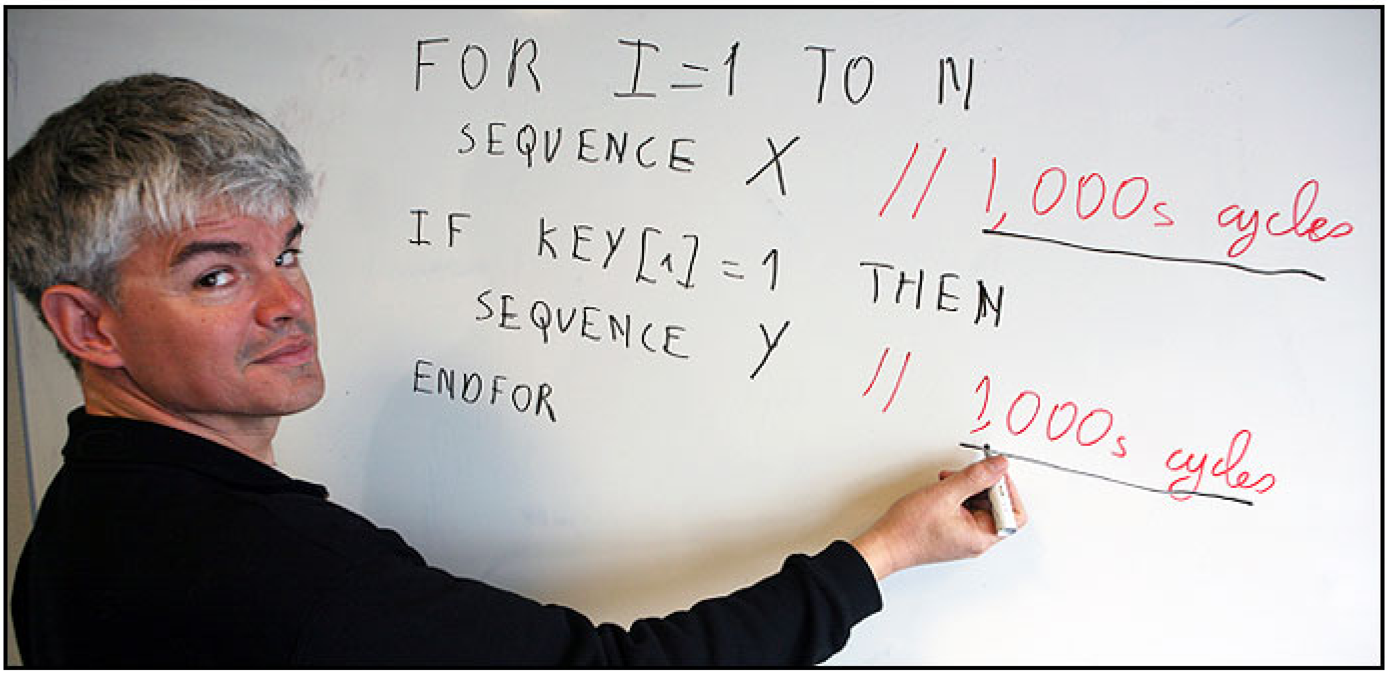
Branch Prediction under Scrutiny for Possible Security Flaw

Branch predictors allow processors to execute the next instructions without waiting for the previous ones to be resolved. These predictors are crucial for achieving high performance. But a recently published research suggests that this CPU feature might as well encompass some security weakness. A branch prediction specialist at Irisa/INRIA research center, André Seznec explains what it is all about.



When the media broke the news, last November, the computer industry went scared stiff for a few breathless seconds. According to punch lines, transactions over the Internet were not secure anymore due to a newly discovered and nothing less than fundamental security flaw in superscalar (1) microprocessor technology.

A team made of cryptology experts Onur Acıiçmez , Çetin Kaya Koç and Jean-Pierre Seifert (2), has evidenced that, in some conditions, some processors may leak information to an attacker because of hardware features initially introduced for enabling faster execution. The breach in processor conception might let top notch hackers squeeze through and indirectly decipher asymmetric keys that are routinely used to protect all sort of on-line transactions. Acıiçmez and his colleagues managed to grab 508 bits out of a 512-bit key on RSA encryption , at first shot, in just a few thousandths of a second. Quite a feat when compared to the endless three months and the line-up of 80-some 2.2 GHz CPU computers that the German Federal Office for Information Security (BSI) once poured in to crack a SSL 640-bit key (3).

The new attack though, is sharply different from the brute force approach that has been in use since WW II and the Bletchley Park-versus-Enigma spooky old days. It does not require billions of computation measurements under the same key. In fact, no upfront number crunching is even involved. The Acıiçmez-Koç-Seifert attack analyzes the CPU's Branch Predictor (4) states through a spy process running in parallel with the cipher. They call this attack the **Simple Branch Prediction Analysis Attack** (SBPA). According to the three scientists, this successful SBPA bears a "critical security implication." In the context of simple side-channel attacks (5), it is widely believed that equally balancing the operations in the two paths of a branch is a secure countermeasure against such simple attacks.

"Not true, Acıiçmez *et al.* study warns. Even such balanced branch implementations can be completely broken by SBPA attacks." Moreover, "despite sophisticated hardware-assisted partitioning methods" such as memory protection, sandboxing or even virtualization, SBPAs "empower an unprivileged process to successfully attack other processes running in parallel on the same processor." Conclusion? SBPAs prove "much more dangerous than previously anticipated".

A branch prediction specialist at Irisa/Inria research center, André Seznec (6) has embarked into reproducing the experiment as he wanted to see for himself. *"I've tried to validate the principle. It works! Beautiful case study by the way!"* An essential prerequisite to the experiment, the cracker must be able to access the machine where some spyware has to be installed. *"But this access can be legal. Might just be your office colleague running a process on your PC!"* The goal, then, is to *"swipe the legitimate user's private key."* For RSA, the challenge that seems out of reach by standard algorithmic means if very long keys in the 1,000s bits range are used. *"RSA crypting is considered as highly robust. Our current state of knowledge simply doesn't allow us to envisage cracking RSA for very long keys by algorithmic mean or brute force in a foreseeable future. It is an algorithm that everybody trusts."*

Unable to crack RSA upfront, SBPA will ingeniously opt for side-channel attack. As Seznec explains, *"on a given hardware, it is possible to externally measure various pieces of information regarding its functioning. Things as simple as the electric flow for instance. It yields an indirect indication. So, even if one doesn't have direct access to the data, one can grab indirect information on this data."* Till recently side-channel attacks were performed through hardware mean. *"However, the introduction of SMT* [simultaneous multithreaded] *processors has allowed to implement pure software side-channel attacks."*

Until not so long ago, processors were executing threads in a time shared mode: *T0* was executing during a time slice, then *T1* was executing during the next time slice, then *T0* again, …*"Each of these time slices lasts far longer than the processor execution cycle. Say a thread lasts around 10 milliseconds, representing about 20 to 30 million processor cycles. As long as a spy thread and a cryptographic thread are not executed simultaneously, there is no* *way the former can grab very precise information on the latter."* The impervious architecture keeps threads peep proof. But things have changed with the arrival of Pentium 4 HT processor generation (7), a SMT processor in PCs and servers. These CPUs run two threads at the same time: on the very same cycle, instructions from the two threads are executed on the CPU. Why? *"Mainly to squeeze performance from the processor, Seznec answers. The processor can execute several instructions per cycle, but generally a significant part of the resource is lost if a single thread executes. When two threads execute at the same time, the hardware is significantly better utilized."* Unfortunately, running two threads in parallel on the same hardware CPU can lead to some information leakage. *"One can manage to grab an indirect view on a thread execution from a spying thread that is executed simultaneously. This indirect information about its execution can allow to recover critical information such an encryption key."*

Now, how to spy a cipher? *"By trying to get indications on (microscopic) execution times,* Seznec details. At the heart of some versions of the RSA algorithms, is a loop that handles two long sequences of operations (say X and Y): X is always executed, and Y is executed only if the key bit is 1. Each of these requires several thousands processor cycles." A rather large time span. *"Now, if someone is able to say, for each of these iterations, this iteratrion is rather X-type, or XY-type, if someone is able to measure how long they last, then he just… got the key."* Q.E.D.

Superscalar processors rely on branch prediction hardware to achieve performance. In particular, the branch is stored in a hardware table called the Branch Target Buffers, BTB. In the loop, as long as *i* doesn't equal *n*, the BTB is searched for the branching instruction on each iteration. If the branching instruction is absent from BTB then the hardware brings it back.

In described SBPA, the spyware is designed to occupy the same entries in the BTB as this branching instruction. *"As long as this branching instruction is absent from the BTB, the spy has a regular behavior,* Seznec describes. *But once the branching instruction of the loop is executed, a branch from the spy looses 20-some cycles. What the cracker wants to do from there, is to detect when the loop comes back: since X and Y last for several thousands cycles, this detection doesn't have to be cycle accurate, but may be even +/- a few hundreds cycles."*

This task is made possible by the fact that processors are equipped with a cycle counter: this yardstick of a sort can be read by any process. *"Since the spy process has filched branch entrance, there is an abnormality in the counting that we can pinpoint with this counter. From this counting, we can figure if the operation within the loop is rather of X or XY type. We get enough precision for discriminating. Key length doesn't make much different at this stage. The code is going to be broken anyway. More clever implementations of RSA will be more difficult to crack, but may also be attacked."* So, doom times ahead for chip builders? *"No,* says Seznec. *All this doesn't imply a rethinking of the way processors are built. But it should* *bolster cipher application developers to take into account the OS and the hardware side of the job. Might be a good idea to deactivate the multithread mode when running crypto, for a start."*

Footnotes

(1) **Superscalar architecture** implements a form of parallelism within a single processor, allowing thereby faster CPU throughput than would otherwise be possible at the same clock rate. It executes more than one instruction during a single pipeline stage by pre-fetching multiple instructions and simultaneously dispatching them to redundant functional units on the processor.
(2) **Çetin Kaya Koç** is professor on leave Oregon State University. **Onur Acıiçmez** is PhD Student at Oregon State University. **Jean-Pierre Seifert** is professor at University of Innsbruck, Austria. The trio is also known under acronym ASK.
For more on their research, see:
- Onur Acıiçmez, Çetin Kaya Koç, and Jean-Pierre Seifert. On The Power of Simple Branch Prediction Analysis. Cryptology ePrint Archive, Report 2006/351, October 2006.
- Onur Acıiçmez, Jean-Pierre Seifert, and Çetin Kaya Koç. Predicting Secret Keys via Branch Prediction. Cryptology ePrint Archive, Report 2006/288, August 2006.
- Onur Acıiçmez, Çetin Kaya Koç, and Jean-Pierre Seifert. Predicting Secret Keys via Branch Prediction. Topics in Cryptology --- CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, M. Abe, editor, pages 225-242, Springer-Verlag, Lecture Notes in Computer Science series 4377, 2007.
(3) Secure Sockets Layer. **SSL** is a communications protocol. Initially developed by Californian company Netscape, SSL has been endorsed by many financial institutions for commerce over the Internet. Many versions of SSL rely on RSA cryptography. Developed in 1977, by Rivest, Shamir and Adleman, **RSA** is the most widely used public key cryptosystem. This algorithm for public-key encryption is widely used in electronic commerce protocols. RSA keys are typically 1024–2048 bits long.
(4) A **branch instruction** is a point in the instruction stream of a program where the next instruction is not necessarily the next sequential one. They are of two types: unconditional (jump, goto instructions...) or conditional (if-then-else clauses …). For conditional branches, the decision to take or not to take the branch depends on some condition that must be evaluated. During this evaluation period, the processor speculatively executes instructions from one of the possible execution paths instead of awaiting idle for the decision to come through. If its prediction happens to be true, the execution continues without any delays. If it is wrong, the misprediction results in a loss of clock cycles.
(5) **Side channel attacks** are based on exploiting information gained from the observation of a system rather than from exploiting the weaknesses in the algorithms. For instance, information about power consumption or electromagnetic leaks can help break some systems.
(6) A research director at Irisa/INRIA-Rennes, **André Seznec** heads CAPS project team (Compilation, Architectures Processeurs Superscalacaires et Spécialisés). He has been working on processor architecture for 20 years.
(7) In 2002, the Intel **Pentium 4 HT** was the first desktop processor to implement simultaneous multithreading. This two-thread SMT engine provides a 30% speed improvement compared against a non-SMT version of the same chip.