Development of Side Channel Attacks

Jose Gallegos
Department of Computer Science
University of California, Santa Barbara, CA 93106
E-mail: {jlgallegos}@cs.ucsb.edu

Abstract— **In this paper I will present five types of attacks that have been introduced in several papers. These attacks use different channels, side channels, to try to break cryptographic algorithms like RSA. The attacks range from simple timing attacks to attacks that recover information left from processor components, i.e. caches and branch predictors.**

I. Introduction

Side channel attacks allow an attacker to determine secrets by viewing information gathered from channels other than direct communications. These other channels can be the amount of time a process takes, the caches that it uses, or the branches that it take. As will be seen later, the security of the RSA algorithm is based off of the fact that some information can be kept a secret. Once this secret is discovered by an attacker, the attacker can impersonate the owner of the secret, signatures, or read encrypted data only meant to be read by the owner of the secret, decryption. Throughout the rest of this paper I will present the main ideas, in attacks that can obtain this secret information.

In sections 3-7 I will briefly describe side channel attacks presented in several papers. In section 3 we will see timing attacks presented in [1]. Section 4 will be about practical timing attacks in [2]. Section 5 will cover cache attacks presented in [3]. Section 6 will contain branch prediction attacks in [4]. The last attack, simple branch prediction analysis [5], will be contained in section 7. Before getting into the attacks I will briefly describe RSA, the cryptographic algorithm that is under attack in most of these papers. In this paper I will assume that the reader knows how caches and branches work, so I will not be discussing those details in this paper.

II. Brief Overview of RSA

Developed by Rivest, Shamir, and Adleman, RSA is a public-key system that also allows for digital signatures [6]. RSA solved the problem of trying to figure out how to securely communicate without having to exchange keys before hand. RSA uses two keys for each person, one public key (e) used for encryption and verifying a signature and a private key (d) used for decryption and creating signatures and another public value (N) for modulus where $N = p * q$ (p,q are both large prime numbers). We use p, q to figure out d, e: $d * e = 1 \pmod{(p-1)*(q-1)}$.

$$\text{Decryption: } M = C^d \bmod N$$
$$\text{Encryption: } C = M^e \bmod N$$

D is the only part that needs to be kept private, because with it anyone else would able to decrypt messages that are meant for the owner of d or be able create signatures that belong to the owner of d. Which means that RSA is only secure if d can be kept secret. It has been shown that since N is the product of two large prime numbers, it would be too difficult to factorize N to compute d from e [6]. Since directly trying to figure out d is too difficult, attackers had to turn to these side channel attacks to figure out d.

The following are two methods used to implement RSA. Both algorithms have timing differences depending on the private key, which could be taken advantage of by side channel attacks. The attacks that I will present in the rest of this paper will show what can be taken advantage of.

Square and Multiply Method

1: $S = M$
2: for i from 1 to $n-1$ do
3: $\quad S = S * S \pmod{N}$
4: $\quad$ if $d_i = 1$ then
5: $\quad\quad S = S * M \pmod{N}$
6: return S

Montgomery Exponentiation

1: Compute n' using Euclid's algorithm
2: $\bar{M} = M * r \bmod n$
3: $\bar{C} = 1 * r \bmod n$
4: for $i = k-1$ to 0 do
5: $\quad \bar{C} = MonPro(\bar{C}, \bar{C})$
6: $\quad$ if $e_i = 1$ then $\bar{C} = MonPro(\bar{M}, \bar{C})$
7: $C = MonPro(\bar{C}, 1)$
8: return C

Montgomery Product

1: $t = \bar{a} * \bar{b}$
2: $m = t * n' \bmod r$
3: $u = (t + m * n)/r$
4: if $u \geq n$ then $u = u - n$
5: return u

III. Introduction To Timing Attacks

In the 1996, the idea of using information collected from other channels to break RSA came from a paper by Kocher[1], which introduced the dangers of timing attacks. In this paper, Kocher brought attention to the information that could be extracted from data dependent execution paths. Kocher mentioned how changes could be caused by optimizations in branches and cache hits. The attack he presented in his paper was applied to a smartcard and was able to determine the private key with 2450 measurements.

IV. Network Attacks

In 2003 Brumley and Boneh showed that not only smart cards were in danger of becoming victims of timing attacks [2]. Brumley and Boneh showed that RSA implemented OpenSSL could also be broken. This is the attack that they described: Assuming we have $q_0, ..., q_{i-1}$ of the secret key we try to figure out the q_i th key.

Step 1: There guess, g, would be equal to the first i bits of q and would set the rest to 0. Then they would get a second guess, g_{hi}, equal to g except for the i_{th} which is set to 1. This would leave two cases: if q_i is 1 $g < g_{hi} < q$ else $g < q < g_{hi}$
Step 2: Compute $u_g = g * R^{-1}$ and$u_{g_{hi}} = g_{hi} * R^{-1}$
Step 3: Get the time to decrypt $u_g(t_1)$ and $u_{g_{hi}}(t_2)$
Step 4: Compute $T = t_1 - t_2$. If T is small then q_i is 1, else T is large then q_i is 0.

In order to increase the chances that q_i would be calculated correctly, neighborhood size and sample size were introduced. For each guess, g, the attacker would test $g + 1, \ldots, g + n$, giving a neighborhood size of n. Each of these $g + i$ would be tested s times, where s would be the sample number. Brumley and Boneh' s attack was used to discover the secret key in processes that ran on the same machine and processes separated by a network. It took over 1 million queries for this attack to work.

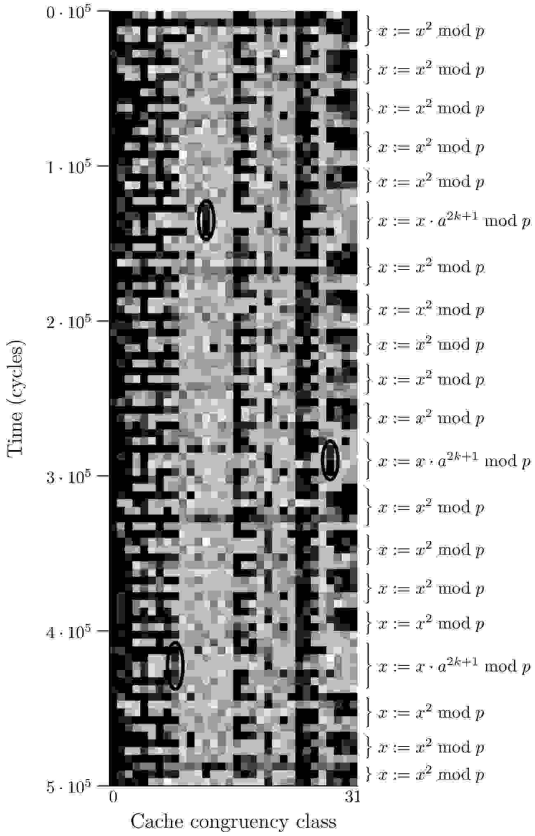

Figure 1: The results of a Percival Cache Attack on RSA

V. Cache Attacks

In 2005, Percival came out with an attack on a cache system [3]. Like the rest of attacks presented in this paper, Percival's attack attempted to discover the secret key of RSA. Percival attack needed two processes, a crypto process and a spy process, which had access to the same cache. The spy process continuously read the cache blocks and measured the time it took to read. Whenever the crypto process used a cache block, the spy process would have seen a change in the read time. When the crypto process was done executing, the attacker would have a record of the squaring and multiplication operations used. This would mean the attacker could attempt to start guessing the secret key used for RSA. The results of Percival's attack are shown in Figure 1 which is from [3]. With one query this attack was able to recover 310 bits out of 512 bits of the private key.

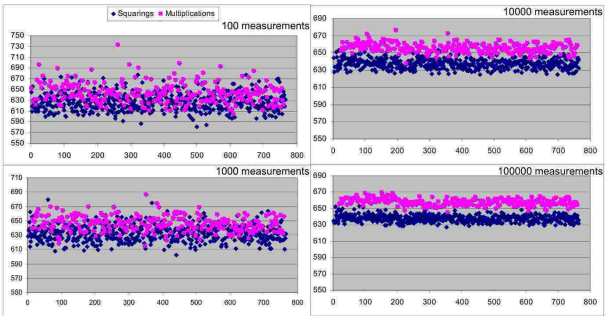

Figure 2: The results of a trace-driven attack on RSA

VI. Branch Prediction Attacks

In 2007, attacks described in Acıiçmez, Koç and Seifert's paper [4] allowed the attacker to go beneath security mechanisms, i.e. memory protection, sandboxing, and virtualization, and use information from a key architecture unit, the branch predictor, to help extract the secret key.

Here is the basic outline for this attack. There are two processors: a spy process and the crypto process. The crypto process computes the cipher text from a plain text. The spy process continuously executes branches in order to fill up the BTB and measures the time it takes to run. The idea would be that if the crypto process never branched the spy process would always run quickly, because all of its branches would be in the BTB. From the RSA algorithm we know that this isn't the case, branches are taken during the crypto process. Since the BTB is filled with spy branches, whenever a crypto branch is taken it would have to evict a spy branch to make room. This eviction will be noticed since the spy process is continuously executing its branches, a missing branch in the BTB will result in a longer execution time for the spy process. This process will be able to determine the path of execution the crypto process took, from this the attacker would be able to extract the private key. Figure 2, taken from [4], shows the results of this attack. With 10000 measurements, this attack was able to extract the private key.

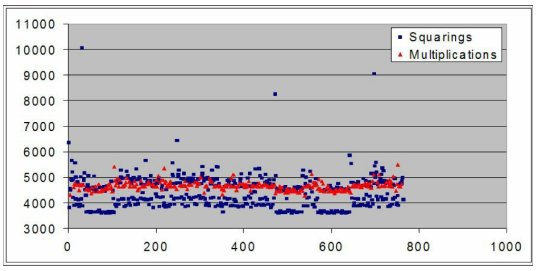

Figure 3: Results of the improved trace driven attack

VII. Simple Branch Prediction Analysis Attacks

Later on in 2007, Acıiçmez, Koç and Seifert came out with another paper [5] that improved on a trace driven attack presented in their previous paper [4]. Just as before this attack tried to extract the private key by using a spy process that would continuously execute branches. In this improved attack, Acıiçmez, Koç and Seifert increased the number of branches executed by the spy process, t_{opt}, to account for another Branch Traget Buffers (BTB) provided by a architecture. With the original attack, whenever a branch was evicted it could be temporally stored in other BTB, which could have thrown off the spy process's goal of determining when the crpyto process executed a branch. With t_{opt}, the attacker could now be sure that all of the BTBs only contained spy branches so there were no branches temporally stored anywhere else. Now whenever the crypto process took a branch it would evict a spy branch. When the spy process ran it would hit a misprediction and evict some other spy branch to make room for this spy branch, then when the spy process reached that spy branch it would hit another misprediction causing other evictions and mispredictions. By the time the spy process reached the end of that round of executions it would have suffered a timing penalty of a ripple of mispredictions. This reippling timing penalty would give a better indication of when a branch was taken in the crypto process. Figure 3 is from [5] and shows the results of the improved trace driven attack. With t_{opt}, the results of this branch attack was able to improve to recovering 508 out of 512 bits of the private key in one measurement.

VIII. Conclusion

In this paper I have shown that because RSA is a secure algorithm, attackers can not directly break it. Instead attackers try to recover information that would help in extracting the private key. Through the years the methods used to get information has evovled. First we saw that timing attacks could lead to extracting the private key, but were limited to smart cards. Later on it was shown that RSA could be attacked over a network, but required a lot of measurements to recover the private key. In order to find a way to reduce number of measurements needed, attacks moved away from directly using timing. Newer attacks came out that took advantage of information that could be determined from architecture components that were used to help processor performance. Data left behind in a cache system, resulted in a trail of information that could lead to determining half the number of bits in a private key. Attacks on a branch predictor unit also emerged that were similar to the attacks on caches. Both branch attacks that I showed here, used the BTB to track when a branch was taken during a process involving RSA. By tracking the branches taken, an attacker could determine the private key. These were the five kinds of attacks that I presented in this paper, which showed how the power of side-channel attacks grew over time.

References

[1] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," 1996, pp. 104–113, Springer-Verlag.

[2] D. Brumley and D. Boneh, "Remote Timing Attacks are Practical," in *Proceedings of the 12th USENIX Security Symposium*, 2003, pp. 1–14.

[3] C. Percival, "Cache Missing for Fun and Profit," in *Proceedings of BSDCan 2005*, 2005.

[4] O. Acıiçmez, Ç. K. Koç, and J-P Seifert, "Predicting Secret Keys via Branch Prediction," in *Cryptology CT-RSA 2007, The Cryptographers Track at the RSA Conference 2007*. 2006, pp. 225–242, Springer-Verlag.

[5] O. Acıiçmez, Ç. K. Koç, and J-P Seifert, "On the Power of Simple Branch Predication Analysis," in *2007 ACM Symposium on Information, Computer and Communications Security (ASIACCS07*. 2007, pp. 312–320, ACM Press.

[6] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.