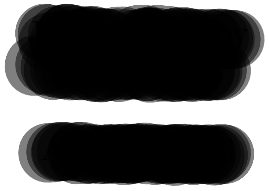
Introduction to Modern Cryptography

Paper Review



1 Introduction

We started this project wanting to look at side-channel attacks. The reasoning behind this is that we mostly trust the hardness assumptions made to ensure a lot of security properties in Modern Crypto. These include the random oracle model, the hardness of the discrete log problem for DDH, the hardness of factoring large numbers for RSA, etc. In fact, breaking a few of these assumptions would require large leaps of understanding in the current computational complexity landscape (a lot of results in the P vs. BPP area as well and by extension the P and NP area rely on the hardness of PRFs). Then, if we trust these hardness assumptions, attacking most encryption algorithms based on their mathematical properties alone is a hard problem indeed.

For these reasons, we have been interested in alternative attacks that don't necessarily attack an algorithm directly, but the hardware it runs on or its specific implementation. This is an area collectively known as side-channel attacks. The paper we are mainly looking at is from Standaert, Malkin, and Yung from 2009 titled *A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks* [4]. We chose this paper because it seems to be one of the first to try to develop a general framework for many different kinds of side channel attacks.

2 Main Review

The main goal of the paper is to formalize the notion of side-channel security. Previous attempts at analyzing side-channel security were mostly ad-hoc and depended heavily on specific implementations and attack combinations. In the authors' words, analyses were of the form "Implementation X is better than implementation Y against adversary A", which leaves something to be desired as this can be due to the effectiveness of the implementation *or* the ineffectiveness of the adversary. Like all security notions, we would like to say something about a scheme being secure against *arbitrary* adversaries (possibly with some underlying assumption about the algorithm itself, à la discrete log).

Specifically, this paper deals only with *key recovery* attacks that utilize side channels to read *leaked* pieces of information from the algorithm to narrow down the space of possible keys to a small class of candidates. The notion of a side-channel key-recovery attack is formalized in Figure 1 (taken straight from their paper). Basically, the idea is as follows (again, terms are taken from their paper). A *device* is a physical realization of a cryptographic primitive (e.g. AES on some specific

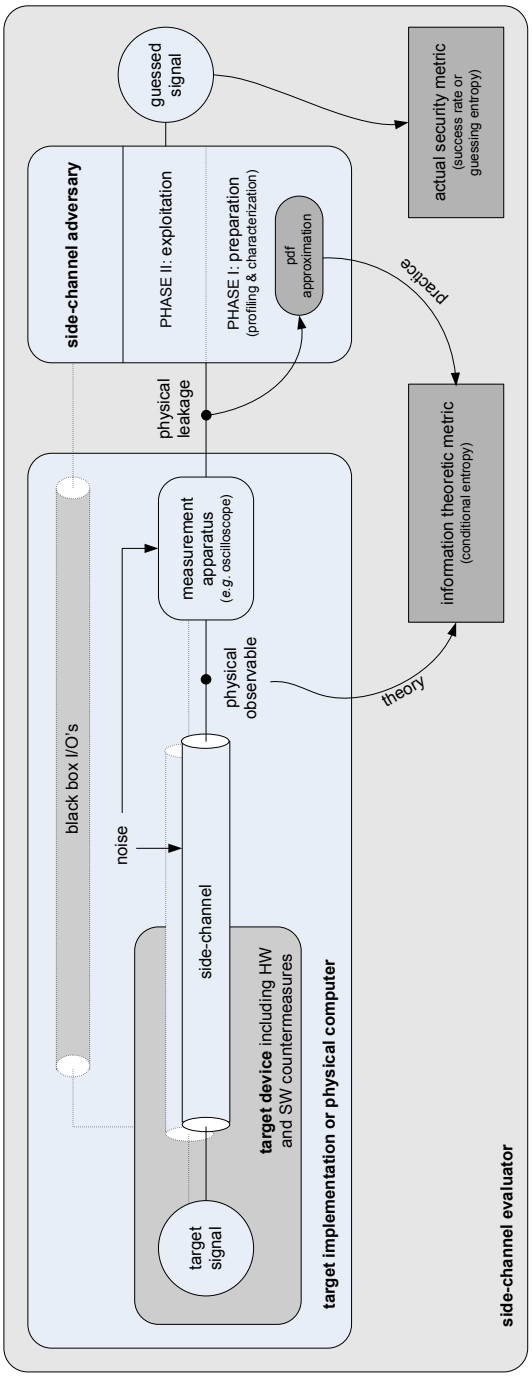


Figure 1: Intuitive description of side-channel key recovery attack

$\underline{\mathsf{Exp}^{\text{sc-kr-o}}_{E_K,L}(A)\text{:}}$
$k \xleftarrow{\$} \mathcal{K}$
$s \leftarrow \gamma(k)$
$[g_1, g_2, ..., g_{|S|}] \xleftarrow{\$} A(E_k, L)$
Ret $s \in [g_1, ..., g_o]$

Figure 2: The side-channel key recovery of order o experiment. Typically, if we require the adversary to guess correctly, we set $o = 1$.

hardware). A *side-channel* is an unintended communication channel that leaks some information from a device through some physically observable media (e.g. power consumption or radiation). This can be abstracted into a notion of a *leakage function*. An *implementation* is a cryptographic primitive together with a suitable leakage function. Then, a *side-channel adversary* is a normal adversary attacking that primitive now equipped with access to the leakage function as an oracle as well. We note that these terms were first introduced in a paper by Micali and Reyzen in 2004 [3], but the results there were too general and lacked this paper's focus on key-recovery.

In general, side-channel attacks are based around how well an attacker can exploit the information given by the leakage function L. There are typically two phases attached to this, which make the leakage function seem almost hash-like. First, the adversary can be given some offline time to "profile" the leakage function. This is basically saying that the adversary knows what the leakages mean and how to read them. In a real world situation, we can assume the adversary has a copy of an implementation offline to study the leakages.

The second phase, perhaps with more cryptographic merit, is the exploitation phase which is an attack mounted from an adversary against an implementation with the goal of key recovery. The authors make an important note that the profiling stage may or may not be present. In other words, sometimes the adversary does not have any offline time to profile an implementation and so must do the profiling online.

The actual game is defined as follow. Let $E_K = \{E_k(.)\}_{k \in \mathcal{K}}$ be a family of cryptographic abstract computers indexed by a variable key K. Let (E_K, L) be the physical computers with associated leakage function L. The adversaries goal is to map the space of available keys $\mathcal{K}$ to a much smaller space of so-called key-equivalence classes S such that $|S| << |\mathcal{K}|$. In other words, we can assume their is some oracle like function $\gamma : \mathcal{K} \to S$ for which the adversary is trying to match. The adversary is given access to E_k and L and outputs a guess vector $\{g_1, ..., g_{|S|}\}$ which basically represents its guesses to which key class k belongs. This is all summarized in Figure 2.

The advantage is as we would expect:

$$\mathbf{Adv}^{\text{sc-kr-o}}_{E_K,L}(A) = \Pr[\mathsf{Exp}^{\text{sc-kr-o}}_{E_K,L}(A) = 1] \tag{1}$$

We note the extra parameter o represents sort of a measure of confidence in the adversary. A adversary playing the game with $o = 1$ is basically required to output the correct single class the key belongs in. This seems to us to be an abstraction of a random adversary. Basically, we can take a randomized $o = 1$ adversary A and use it to make an $o = o'$ adversary by running the (randomized) A many times and ranking its outputs. This is not addressed in this paper, though, and we talk a bit more about S and the guess vector in Section 3.

The resources tracked in the experiment are the time complexity τ and the memory complexity m, which are typically limited by the power of the adversary. We also note that m will limit the

effective size of S, as the adversary cannot form a guess vector longer than its memory complexity. We also keep track of the number of queries q to L and E_k. Finally, we note that if an adversary succeeds on an attack of order o, then it still has to check a maximum of o candidates after its key recovery attack. The paper also defines another game which measures the average number of key candidates left after an attack (i.e., order o for which an adversary has high advantage), but we do not include it here.

This is the main contribution in terms of security notions the paper gives. This setup can be applied to a variety of implementation and adversaries. As the authors say, one of the nice things about this formulation that hasn't been done before this is that the notion of leakage is independent of the type of attacker. Before this, most of the analysis on side channel attacks was done with strong assumptions of the attacker. Abstracting this out allows for more discussion on the general security of your implementation from *any* attackers, something that is obviously desirable.

3 Security Bounds

It is clear that one of the main goals of this paper was to develop this notion of side-channel key recovery for a wide variety of side-channel leakages and attacks. To add to this though, the authors also give a more concrete interpretation of the notion of leakages in the form of information theory. They abstract the adversary in the following manner. Let S be the set of target class variables . Then, assuming an adversary makes q queries to the leakage oracle, let $\mathbf{l}_q = [l_1, ..., l_q]$ be the results of such queries (i.e., $\mathbf{l}_q$ is the leaked information). Then, we assume the adversary seeks to use the conditional probability $\Pr[s|\mathbf{l}_q]$ to estimate the likelihood the key is of class $s \in S$. Using this, a conditional entropy can be generated, which we will not go into the details of here.

The author's note that one main problem in actually implementing this type of model is that it is not clear how an adversary gets a good estimate of the leakage distribution $\Pr[\mathbf{L}_q|S]$, needed to calculate the entropy. This is where the offline part of the attack comes into play, in the sense that an adversary would have to have time to estimate $\Pr[\mathbf{L}_q|S]$ beforehand. They go on to give conditions under which the adversary can approximate the leakage distribution. However, there results only work for adversaries with bounded time to do this.

We are not sure this information theory based approach is entirely reasonable. Specifically, the problem of estimating the leakage distribution being completely probabilistic and bounded in time. We think that this bound can be circumvented by an adversary with enough access to the inner workings of whatever implementation is being used. With enough access to such inner things (i.e., assume the adversary has a copy of the hardware and algorithm being run), it is not clear why the leakage distribution should be hard to predict. For side-channels such as power consumption or radiation, this makes some sense as it is hard to turn these into readable bits.

There are other side-channel attacks such as the cache attack seen in [2] that do not seem to fit this mold though. Specifically, the idea here is that in all of these IaaS (Infrastructure as a service) providers like AWS (Amazon Web Services), programs that share hardware can read certain parts of the cache left by others on that hardware. If the attacker reading the cache can determine what was being run, or even better, it can send requests to the victim to do what it wants to, say set up a key exchange), then it can read data directly from the cache that the victim was using to do its computations. For instance, a victim has to load bits of its private key into the cache to do key exchange. If it is pre-empted before it finishes, the attacker can then read the data in the clear. In this case, the data is either there or not there, it is not clear what the leakage distribution would be like. We could be comparing apples to oranges here, but we would like to see where this sort of attack fits into this framework, or what sort of framework it would fall in.

Another hole in this paper specifically is that the set of classes S is never really talked about. I.e., how does an adversary take the key space $\mathcal{K}$ and turn it into a smaller representative set S? We note though, that there has been some research into this very problem, see for example [1]. This addresses the question we had before about how the algorithm generates this guess vector.

4 Open Problems and Future Research

This paper was mostly developed to lay the groundwork to study side-channel attacks in the abstract. As such, many of the open problems raised are in the line of real world realizations of these theoretical concepts. Specifically, how to build implementations that are secure in this notion. According to the introduction of [1], much of the side-channel analysis is still done on a case by case basis. Most of the actual security measures developed throughout the years are ad-hoc and rely on empirical evidence of security. These include masking and shuffling. This is similar to what the authors of [2] ultimately suggested to do. (What they actually suggest is a way to detect when you are under attack, and when that occurs, go into a bunch of random computations until the attacker decides to leave.) These are certainly effective from a security standpoint, but oftentimes at a large cost of efficiency. Basically, you are doing a lot of extra work to hide the "real" work. It definitely seems a worthy goal to bring this from the empirical world to a nice neat theoretical space like described in this paper.

Another topic to address is that in this paper, they focused solely on key recovery attacks, which we think comes directly from the fact that this is what most real-world side-channel attacks exploit. This falls short of our normal notions of PRFs, IND-CPA, etc. It would be nice to see a notion of indistinguishability from random in terms of data measured from a side-channel. This seems to be what the shuffling and masking discussed above are trying to emulate, but we would like to do some more research to make sure.

References

[1] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. *Fast Software Encryption: 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, chapter Simpler and More Efficient Rank Estimation for Side-Channel Security Assessment, pages 117–129. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[2] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE Symposium on Security and Privacy*, pages 605–622, May 2015.

[3] Silvio Micali and Leonid Reyzin. *Theory of Cryptography: First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004. Proceedings*, chapter Physically Observable Cryptography, pages 278–296. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[4] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. *Advances in Cryptology - EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, chapter A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks, pages 443–461. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.