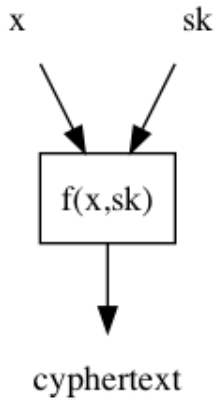
Side-channel attacks

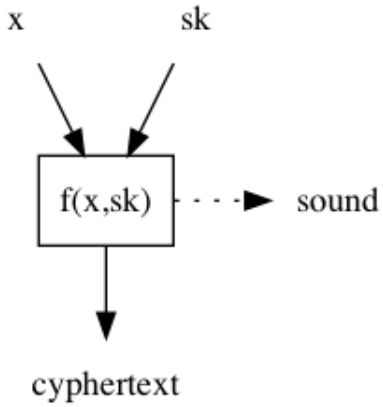
Scribe: Michael Dennis

1 Attack Model

Consider any security setting, for example a cryptosystem:



In many cryptographic settings we try to show that cyphertext does not leak anything about sk or x, assuming that there are no other sources of information about sk or x. However, in a side channel attack, we break this abstraction by looking at other ways of observing the computation of f that may leak more information even when the output of f mathematically leaks nothing. In this case a more realistic model is something like:



This is a very practical attack scenario, especially in cases like smart cards(like VideoGuard), a system used for streaming encrypted movies to clients. In this case an attacker can buy a smart card and have full

access to their power, and performance characteristics. Having this much access to the physical system allows the attacker to measure many aspects of the computation that the could leak otherwise secure information.

1.1 Power Consumption Side Channel Attack on RSA

As a simple example we will start by looking at how to attack RSA through a side channel on power consumption. RSA needs something like the following subroutine:

```
def modexp(z,d,q):
    y = z

    for i in range(log(d),0):
            y = y**2 % q
            if d & (1 << i) == 0:
                    y= zy % q

    return y
```

You can see in the above code that there is a branch-dependency on the bits of d. This means that if you can reliably detect which branch was taken, you can recover the bits of d. It turns out that the amount of computation and the power usage in the two branches are distinct enough that you can distinguish which branch was taken, and in the extreme case, can even see it visually by looking at the power readings during the computation.

1.2 Examples of Side Channel Attacks in the Wild

There are a large variety of side channel attacks that have been discovered, using a wide range of different channels to leak information, below we will cover just a small selection to give a sense of how diverse the attack surface can be.

1.3 Exploiting Size Leakage

In most instances of encryption, the encrypted text is roughly the same size as the input text. This means that even though you don't know what is being sent you know how much is being sent. In some settings this gives you quite a bit of information. For instance, when you visit a website the size of your GET request is some constant plus the size of the URL, most of which is determined by which server you are talking to. Also, by looking at the response you can determine the size of the webpage. Even worse, you can can get information from all of the new requests that are issued by the browser for resources it needs to render the page, and you can get their sizes. All of this means an attacker can often infer what page you are on even when you are encrypting all of your traffic.

Another version of this attack occurs when you type things into Google. The webpage asks the server for auto-complete suggestions as you type and these suggestions can be of different lengths. An attacker can try all next characters and view the size of the response. It knows that the response that matches the size of your response corresponds to the character that you typed and thus the attacker can see what you are typing as you type it. In a similar vein you can infer what movie youre watching on Netflix based on how well the movie compresses at different points, this usually gives a unique signature for the movie.

All of this information seems to be secure upon first glance, but can be inferred by knowing nothing more than the size and the behavior of the server you are interacting with. This means these sorts of attacks are very prevalent and are often overlooked.

1.4 Timing attacks

Another often-ignored side channel is timing. Simply by recording the amount of time between different packets it is sometimes possible to infer how long it took the user to respond or how long it took some program to run. For instance in ssh each keystroke is sent encrypted as the user types. This means that even though an attacker does not know what you are typing, it does know how long you spend between keystrokes. This can allow an attacker to see the length of the commands that you type, since you type a command, wait for a response, and then type a new command. If you are typing longer text in an editor this can allow an attacker to get a good guess as to what you are typing. For the average typist, there is a large difference between the typing speed when two consecutive keys are typed by separate hands or not. By recording these speeds and training a program to predict what was typed you can recover a surprising amount of information.

Any extra information can make this attack much, much stronger so full English paragraphs can be correctly reconstructed. For instance, if you are on a Skype call, and someone is typing you can train a classifier to reconstruct the text from the sound of the keystrokes base on both the timing of the the key presses as the tone of the keys (since different keys sound different).

1.5 Sound

In fact, more generally sound can be an interesting side channel in its own right.

As a historical example, the British planted a microphone near enough to a Russian computer that it could hear the sounds of the computation (which were much more obvious for older computers). They were able to get the secret keys to decrypt their communications from the sounds of this computation. Even now you can hear computation on desktop computers with a sensitive enough microphones, as capacitors discharge. This gives you a way to estimate the amount of computation and allows channels to attacks like the one we performed on RSA.

2 Putting Side Channels into Practice

One obvious concern about actually implementing any of the above mentioned attacks is the amount of noise in the system. Here we will look at how to use some simple tools from statistics to conduct effective attacks despite a large amount of noise and show that, in general, introducing more noise into a side channel does not protect it from leaking information.

As an example, imagine that you have an eel and you know that it is either a 3 volt eel or a 5 volt eel, but you don't know which. You want to determine which of these is the case, but your measuring device is not perfect. To make this more concrete we have V_{obs} as the measured voltage, V_{eel} as the actual voltage and $V_{noise} \sim N(0, 1)$ as the noise in our measurement (which for simplicity we will assume is a Gaussian) then:

$$V_{obs} = V_{eel} + V_{noise}$$

If you have only can take one sample and the two species are equally likely the best decision rule is to guess 5 volts if the measurement is above 4 volts and guess 3 volts if the measurement is below 4 volts. This is about 80%.

Now lets assume that 80% is not good enough, and we decide to take more samples. We will model the noise of each of these samples as being independent.

$$V_{obs,i} = V_{eel} + V_{noise,i}$$

Now it seems reasonable to take the average of all of our measurements and use that to make our prediction. In fact, this simple approach gives us a dramatic improvement in accuracy. Lets see how this effects the standard deviation of the noise:

$$\text{Let: } w = \frac{1}{n}\sum_{i=1}^{n} V_{obs,i}$$

Then:

$$\begin{aligned} var[w] &= 1/n^2(\sum_{i=1}^{n} v_{noise,i}) \\ &= 1/n^2(\sum_{i=1}^{n} 1) \\ &= 1/n \\ sd[w] &= 1/\sqrt{n} \end{aligned}$$

This means that the standard deviation of the measurement error goes to 0 very quickly as the number of measurements increase. In our eel example, using 100 samples results in an probability of misclassification so small that every calculator I could find rounds it down to 0.

3 Applying Statistics to the RSA attack

Going back to the RSA example, suppose we tried to add random noise to our computation to make it take longer or use more power randomly. Then we can determine if $x < q$ or $x > q$ based on multiple samples and using statistics. Let $T_{obs,i}$ be the observed time for sample i, let $T_{noise,i}$ be the noise for sample i and let T_{act} be the actual time for a to run without the added noise. We saw earlier how knowing T_{act} can reveal the secret key. This gives us:

$$T_{obs,i} = T_{act} + T_{noise,i}$$

Now we simply use the same method as with the eels, running multiple tests to get the desired accuracy. We saw that the noise was quickly removed as the number of samples increased, so with even a moderate amount of samples we would again be able to successfully attack RSA.

Possible Defenses

One easy thing you could think to do is to pad out the amount of time you take to be the maximum time that you will ever take. However, in practice this is prohibitively slow, as you will end up spending more than half the time waiting. Much more efficient is to pad out to the amount of time that you will take in 99.9999% of cases. This will work in the majority of cases, but every so often it will fail. At this point you could throw out the key and get a new one.

4 Student Talk: Side channels on Amazon Cloud

Amazon cloud is a service that hosts your virtual machines on their servers. There are usually multiple virtual machines per server, allowing a unique opportunity for cross-VM side channel attacks. This is the setting that "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds" by Ristenpart et al. successfully attacked.

4.1 The Attack

The attack is in three parts:

- Find the victim
- Co-locate with the victim
- Attack the victim

The paper being reviewed attacked all three. They started by mapping the infrastructure of EC2. Finding that VMs get internal/external IP and that asking the internal DNS for the IP associated with the external domain name gives the internal IP. Then they built a way to detect co-residence, by cross referencing IP address with the request latency. Finally, they launched exploratory probes to co locate with a victim, abusing idiosyncrasies of Amazon's VM assignment strategy. Then an attacker can use any number of side channel attacks like cache covert channel attacks, keystroke timing attacks, and estimate traffic rates.