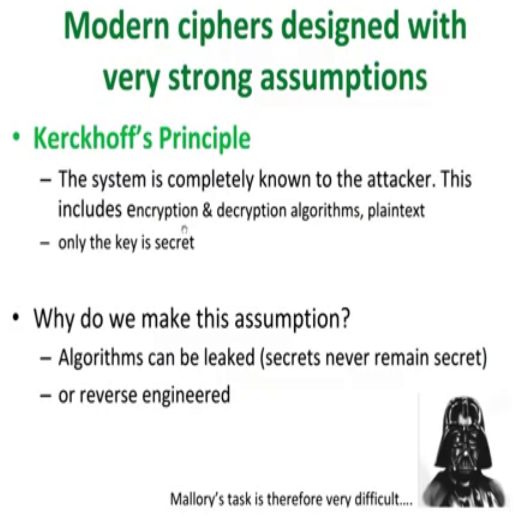
Information Security - 5 - Secure Systems Engineering
Professor Chester Rebeiro
Indian Institute of Technology, Madras
Side Channel Analysis

Hello and welcome to this video lecture in the course for Secure Systems Engineering, in this particular video lecture we will be looking at a brief introduction to Side Channel Analysis. So this video lecture assumes that you have decent background about the AES block cipher and you also know a little bit about the RSA public key cryptosystem. So what we will see in this lecture is how we could use a few tricks to break these extremely strong algorithms.

(Refer Slide Time: 00:47)



Modern ciphers designed with very strong assumptions

- Kerckhoff's Principle
  - The system is completely known to the attacker. This includes encryption & decryption algorithms, plaintext
  - only the key is secret
- Why do we make this assumption?
  - Algorithms can be leaked (secrets never remain secret)
  - or reverse engineered

Mallory's task is therefore very difficult....

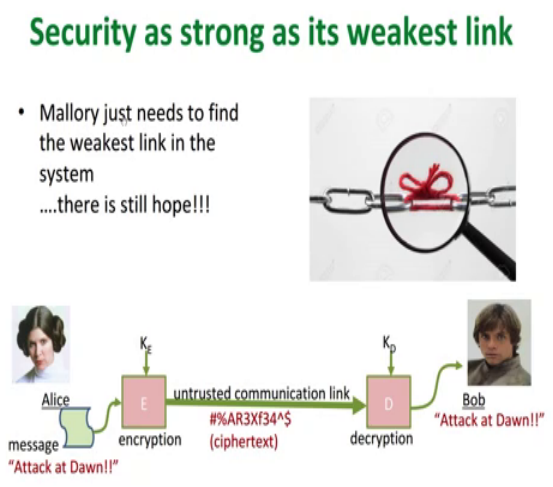
To actually start off with the modern block ciphers are designed with very strong assumptions so they are infact design with the assumption that an attacker could know everything about the algorithm except the secret key. For instance attacker would know the entire encryption algorithm the entire decryption algorithm and we also assume at the design time the attacker would be able to determine what the plain text is and the corresponding cipher text. Stronger assumptions are also done where we make the assumption that the attacker also can choose the plain text that he wants to encrypt or in other circumstances choose the cipher text that he wants to decrypt. The only thing that is unknown in all of this is the secret key.

Now the goal of the attacker over here is to be able to manipulate the plain text, cipher text choose different plain text choose different cipher text with the objective of identifying what the secret key is. The whole idea is that if the secret key is determined then subsequence cipher text based on that algorithm and the that specific secret key can be easily decrypted why exactly do we make such an assumption? So we make this assumptions because it is very difficult to keep things a secret so we want to actually minimize the amount of information that is kept secret.

For example if we use an encryption or a decryption algorithm and we assume that the attacker does not know about this algorithm and we base our entire security on this particular assumption it may be possible that attackers actually learn about this algorithm and how it actually functions or is able to actually reverse engineer the specific algorithm. An interesting case study is about the two stream ciphers A 5/1 and A 5/2 which pair initially kept secret. Now so various researchers have figured out how A 5/1 and A 5/2 there is a combination of leaked information about this algorithm plus reverse engineering and eventually after a few years the attackers were able to identify how exactly this algorithms function and they would be able to create crypt analytic attacks which can break these algorithms.

So this was in the late 90's and early 2000's where both A 5/1 and A 5/2 were actually broken and it eventually resulted in a new cipher known as the A 5/3 that was designed. Now all ciphers that we use today in practice are designed with this assumption so ciphers such as the AES, RSA, A 5/3 and so on are designed with the assumption that the attacker knows the complete algorithm for encryption, decryption and can choose plain text and cipher text and the only secret is the secret key. Inspite of all these assumptions the ciphers are designed in such a way that it any attack would be extremely inefficient to develop.

(Refer Slide Time: 04:10)



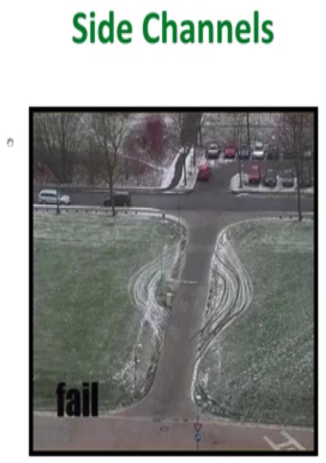
Security as strong as its weakest link

- Mallory just needs to find the weakest link in the system
  ....there is still hope!!!

Alice
message
"Attack at Dawn!!"
K_E
E
encryption
untrusted communication link
#%AR3Xf34^$
(ciphertext)
K_D
D
decryption
Bob
"Attack at Dawn!!"

However what we will see today is that security is as strong as it is weakest link so we see that it may not be always required for attackers to break the algorithms but just find one weak link in the entire system and utilize that particular weak link to break the cipher.

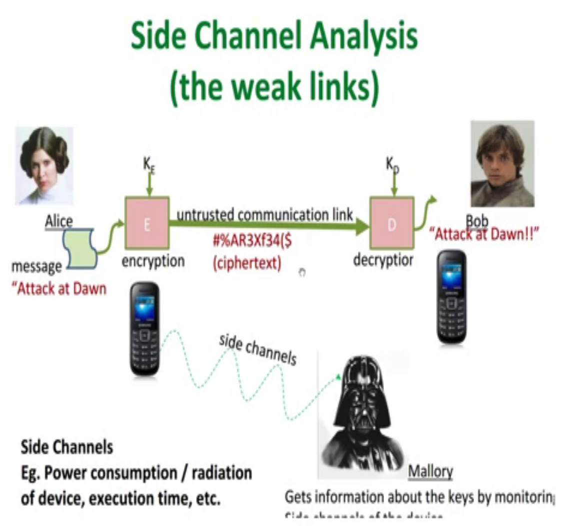
(Refer Slide Time: 04:32)



Side Channels

So the weak link that we actually look is known as side channels. As we see in this very common figure is that if an attacker cannot go through the right way then what he could possibly use is a side channel and use some indirect information to actually break the specific cipher.

(Refer Slide Time: 04:51)



Side Channel Analysis (the weak links)

Side Channels
Eg. Power consumption / radiation of device, execution time, etc.

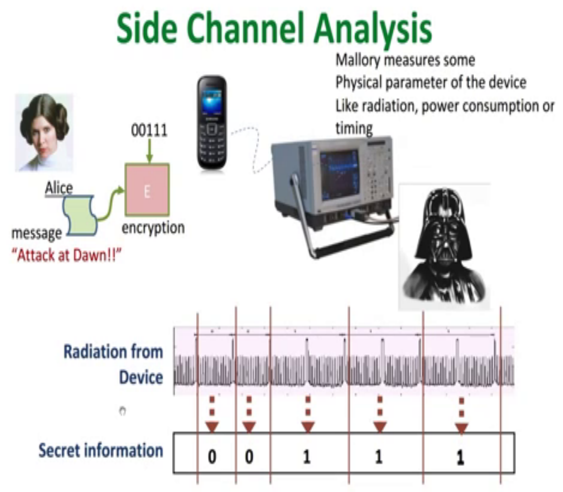
Gets information about the keys by monitorin

So will take a small example of what a side channel attack is in a very abstract way, a side channel not only targets the algorithm that is used for encryption or decryption but also targets the implementation of that particular algorithm. So for example if we consider to people Alice and Bob and they are using a specific encryption algorithm to communicate with each other what a side channel attacker would use is the implementation of that particular algorithm this is due to the fact that this is for example let us say that Alice is using a mobile phone like this to speak to bob so a side channel attacker would keep track of the side channel information that is leaking from this particular device.

Using the side channel information the attacker would be able to identify the secret key that was used in the communication between Alice and Bob. So over the last 20 to 25 years there have been a large number of different side channel that have been used to break cryptographic ciphers so most common ones of them are power consumptions that is an attacker if he has position of this particular device or is able to gather the power consumed by that device then the attacker would be able to gain information about the computations that the device performs.

The basic fact over here is that the power consumed by this particular device is correlated with the operations that the device does. So for example some operations say a load instruction or a store instruction would take considerably more power compared to other instructions such as a simple move or an addition and so on. Other side channels which are very popular are radiation based side channels as well as execution time. So we had seen such a side channel in the in a previous video where we seen how cryptographic cipher can be broken by monitoring the execution time for that particular cipher. So in today's video lecture we will be looking at another form of side channel attack known as a fault attack.

(Refer Slide Time: 07:22)

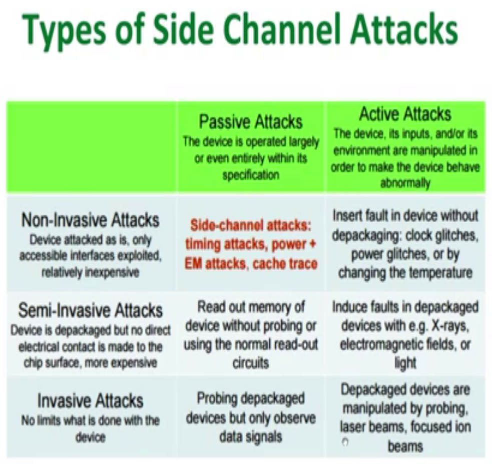


So here is a very simple example of how a fault attack would look like so we have Alice who wants to encrypt this message attack at dawn so she is using a particular encryption algorithm who's secret key is 0 0 1 1 1, now this encryption algorithm is executing on a device like this and this device would essentially use the message which Alice wants to encrypt and the corresponding secret key and pass it to an implementation of this encryption algorithm. Now as this implementation is executing within this device we have an attacker who is monitoring the device side channel.

So in this particular case the attacker we assume has used an antenna or has been able to tap into the power consumed by this device and monitor dynamically the radiation or other side channels such as the power consumed by this device. So for example over here this particular part of this figure shows dynamically the radiation from this device. So what we see over here is that the radiation differs depending on what bit of the secret key is being executed. So if you see look at this part of this waveform and compare it with this part what we see is that the glitches are very thin over here while over here we have longer glitches.

By using this fact the attacker could monitor or distinguish between a zero that has being used in the secret key and a one that is being used thus by just by monitoring this particular device radiation an attacker would be able to identify whether a key bit was 0 or a key bit is 1 this is known as a simple power attack and was actually demonstrated in the mid 90's and since then there have been a lot of different counter measures for this specific attack and also popular devices like the ones we see today can easily prevent such an attack. However there have been many more, more sophisticated attacks which are far more difficult to actually prevent.

(Refer Slide Time: 09:41)

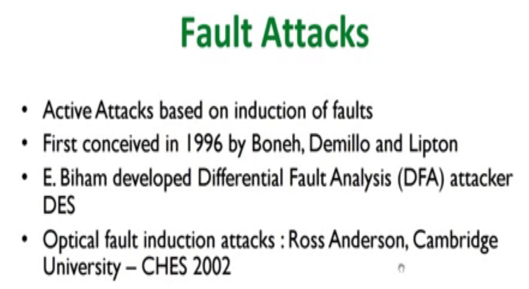


Types of Side Channel Attacks

	Passive Attacks The device is operated largely or even entirely within its specification	Active Attacks The device, its inputs, and/or its environment are manipulated in order to make the device behave abnormally
Non-Invasive Attacks Device attacked as is, only accessible interfaces exploited, relatively inexpensive	**Side-channel attacks: timing attacks, power + EM attacks, cache trace**	Insert fault in device without depackaging: clock glitches, power glitches, or by changing the temperature
Semi-Invasive Attacks Device is depackaged but no direct electrical contact is made to the chip surface, more expensive	Read out memory of device without probing or using the normal read-out circuits	Induce faults in depackaged devices with e.g. X-rays, electromagnetic fields, or light
Invasive Attacks No limits what is done with the device	Probing depackaged devices but only observe data signals	Depackaged devices are manipulated by probing, laser beams, focused ion beams

There are different types of side channel attacks some are non-invasive like the power electromagnetic radiation and the timing attacks are all examples of non-invasive passive attacks. So these attacks are passive because the attacker is passively monitoring the side channels that are emitted from the device. On the other hand the attacks that we would actually look at today are known as non-invasive active attacks. So these attacks popularly known as fault attacks would modify the device functionality in order to glean secrets secret information. So these attacks are what we are going to look at in this video lecture so let us look at what fault injection attacks are.

(Refer Slide Time: 10:36)



Fault Attacks

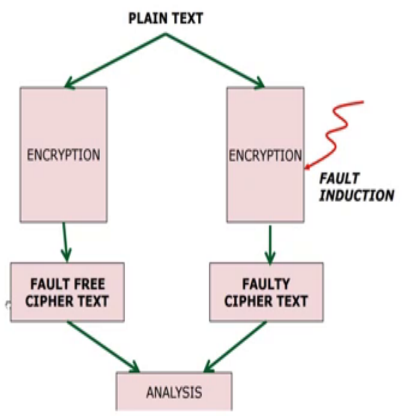
- Active Attacks based on induction of faults
- First conceived in 1996 by Boneh, Demillo and Lipton
- E. Biham developed Differential Fault Analysis (DFA) attacker DES
- Optical fault induction attacks : Ross Anderson, Cambridge University – CHES 2002

So just to give a brief background fault injections attacks are active attacks where faults are induced in a device while it is actually computing a cryptographic function for example while an encryption is being executed by the device an attacker would induce a fault into the device and manipulate the device operation what would happen when the fault is injected is that a certain paths during the execution would get modified or skipped and as a result the output from the encryption or the decryption would be incorrect.

Now the attacker would then use this incorrect output to gain secret information about the secret key. So this attack was first conceived in 1996 by Boneh, Demillo and Lipton in a very popular attack on RSA later on Biham actually developed differential fault analysis attack on the block cipher desk and also there were other different types of fault attack like the optical fault induction attacks by Anderson which was published in CHES-2002.

(Refer Slide Time: 11:57)

Illustration of a Fault Attack



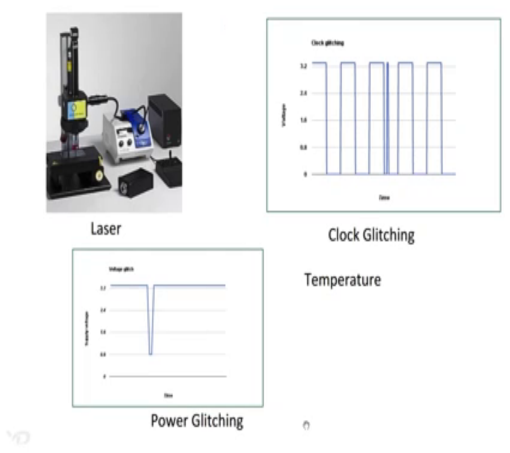
So the basic idea of a fault attack is something like this, the assumption is that the attacker has in position a device now this device for the attacker is like a black box internally it has an encryption algorithm which is stored inside the device. Now the attacker is able to sent messages which we know call as plain text trigger this encryption algorithm to execute and also collect the cipher text the objective of the attacker is to somehow extract the secret key which is stored inside the device. So what the attacker would do is that first of all he would fabricate some particular message and pass it to the device and force the device to create an encryption so the encryption would the message will pass through the encryption algorithm which is implemented in the device and would give you cipher text we cal this particular cipher text as a fault free cipher text.

Next what the attacker does he is that he sends the same message the same plain text through the same encryption algorithm in the device but this time as the device is encrypting that particular message the attacker would induce a fault in the execution what the fault would do, is that it would toggle a few bits during the computation or it would skip a few instructions or it would manipulate what operation is being executed during the execution. As a result of this error or this fault that is induced the cipher text that is obtained would be incorrect thus the attacker would have a faulty cipher text which looks different from the original fault free cipher text.

So we have two cipher text a fault free cipher text and a faulty cipher text and both the cipher text are essentially different. Now the attacker would analyze the fault free cipher text and the faulty cipher text and from this derive the secret key so there are essentially two things to look over here first is how would the attacker induce the fault during an encryption? And secondly how would the attacker identify from the faulty cipher text and the fault free cipher text what the secret key is?

(Refer Slide Time: 14:26)

How to achieve fault injection

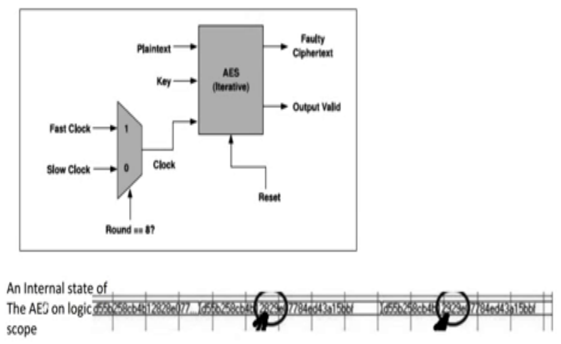


In order to actually inject the fault there have been several ways proposed in the literature some of them are extremely cheap and you could actually be able to do it with less than a 1000 dollars, while other techniques were fault injection are much more expensive. So the expensive ways would look something like this where we would have a laser so we would place the device which the attacker is interested in the laser and fire laser at this specific device. Now this laser when it is pointed to a specific to the device would toggle some bits during the computation of the cipher and thus induce the fault. Similarly a fault can also be induced by injecting glitches in the power consumption or by having a glitch in the clock source for that specific device. So what happens when we have a glitch in the power consumption is that this glitch or this glitch can again toggle the device state resulting in a wrong or faulty computation.

Similarly a glitch in the clock source for the device can create a setup or a hold violation injecting a fault into the operation of that device. There are other techniques which also have been experimented with such as the use of temperature increasing the temperature of a device as we know would cost induce faults in the device however as the experiment show injecting faults using temperature is more difficult to achieve compare to the other techniques of power glitching or clock glitching.

(Refer Slide Time: 16:14)

Fault Injection Using Clock Glitches

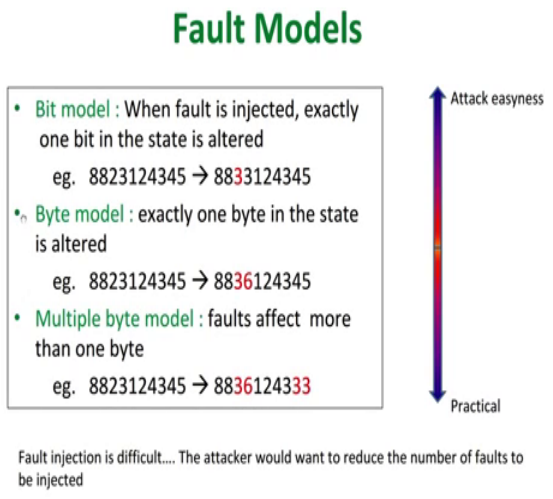


So this is an example of fault injection using clock glitching so what we have here is an AES implementation. Now this AES implementation takes a plain text and the secret key and gives you a cipher text. Now this AES implementation also takes as input a clock source as well as a reset signal. Now in order to induce a fault at a specific time during the execution of this AES what we do is we have a multiplexer which multiplexes between a slow clock and a fast clock. So whenever an attacker wants to induce a fault he would change the value of this select line for this multiplexer from 0 to 1 and thereby inducing a fault. So what would typically happen in the nominal operation of this device is that we would have a slow clock which is pass through so this means that the select line for this multiplexer is zero and therefore you would have a slow clock (())(17:21) to this and AES is operational on a slow clock.

Now when the attackers which is (())(17:29) so momentarily we would have a fast clock that is pushed into the AES clock input and momentarily the AES is functioning at a very high rate using the fast clock. So this fast clock (())(17:45) setup and hold times of various gates in this AES circuit resulting in a faulty output. Now this faulty output hen propagates to the output of this AES hardware resulting in a faulty cipher text. So this particular figure over here shows the effect of a fault so as we if you highlight on this particular area we see that it has a value of 2829e so this is the result when there is no fault which is when present now if we just inject a fault due to things like a clock glitching what we see is that this value of 282 gets modified to 292 so what we observe here is that one bit has actually toggled.

So instead of 8 over here it has become a 9 indicating that the LSB bit of this particular nibble has actually been modified and this modification has occurred due to the fault that has been induced.

(Refer Slide Time: 18:48)



Fault Models

- Bit model : When fault is injected, exactly one bit in the state is altered

  eg. 8823124345 → 8833124345
- Byte model : exactly one byte in the state is altered

  eg. 8823124345 → 8836124345
- Multiple byte model : faults affect more than one byte

  eg. 8823124345 → 8836124333

Attack easyness
Practical

Fault injection is difficult.... The attacker would want to reduce the number of faults to be injected

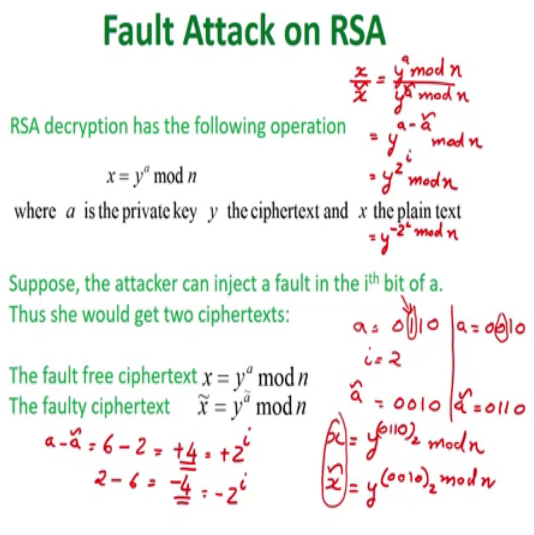
So there are various different types of fault models the most common one is the bit fault model so we have seen a bit fault model over here where exactly one single bit has been modified a bit fault model as an example is seen over here you have a state in the cipher which has a value 8823124345 in the bit fault model we assume that the attacker is very precisely be able to inject a fault rather one bit over here as just been modified by a single bit.

So for instance over here 23 has now become 33 that is 1 bit that has modify the state of the cipher so we call this as a bit fault model. Another fault model is known as the byte fault model here what we assume is that 1 byte in this entire state has been modified so we have like 8823124345 which has changed to 8836124345 essentially this 23 has become 36 so the assumption here is that within this specific byte of 23 it could change to any of the other 255 different values but the other bytes like 88 12 43 and 45 are not affected by the fault.

Third fault model which is the most practical of all is the multiple byte fault model. Here we see that the fault is not restricted to a single byte but rather it could actually spread to more than 1 byte so in this particular example what we see is that 23 has become 36 similarly 45 has changed to 33 due to this multiple faults. These fault models actually influence the complexity and power of the attacker now if an attacker is able to inject a bit fault that is precisely change exactly 1 bit in the fault then the attack becomes extremely easy.

On the other hand actually injecting such a bit fault is extremely difficult in practice, on the other hand injecting multiple byte faults is much far more easier and the attacker would be able to injct such multiple byte faults far more easily however extracting secret information using a multiple byte fault is more difficult. What we see over here is that as we move from the multi byte fault to a but fault model the attack becomes much more easy. On the other hand the practicality of the attack starts to reduce. So attacks which use the multiple byte model is more practical compare to the attacks which use bit fault model, this is due to the fact that it is easier to inject faults in multiple bytes then to inject faults in precisely one single bit.

(Refer Slide Time: 21:58)



Fault Attack on RSA

RSA decryption has the following operation

$$x = y^a \bmod n$$

where a is the private key y the ciphertext and x the plain text

Suppose, the attacker can inject a fault in the i^th^ bit of a.
Thus she would get two ciphertexts:

The fault free ciphertext $x = y^a \bmod n$
The faulty ciphertext $\tilde{x} = y^{\tilde{a}} \bmod n$

$$\frac{x}{\tilde{x}} = \frac{y^a \bmod n}{y^{\tilde{a}} \bmod n} = y^{a-\tilde{a}} \bmod n = y^{2^i} \bmod n = y^{-2^i} \bmod n$$

$a = 0\boxed{1}10 \quad | \quad a = 0\boxed{0}10$

$i = 2$

$\hat{a} = 0010 \quad | \quad \tilde{a} = 0110$

$\boxed{x} = y^{(0110)_2} \bmod n$

$\boxed{\tilde{x}} = y^{(0010)_2} \bmod n$

$a - \tilde{a} = 6 - 2 = \underline{\underline{+4}} = +2^i$

$2 - 6 = \underline{\underline{-4}} = -2^i$

So let us take a look at a popular fault attack on the RSA (())(22:02) key cipher. Now the RSA decryption works like this so we have y which is the cipher text and we reset to a secret key known as a so y to the a modulus n would give you the plain text x typically the value of a would be quite large it will be around 1024 bit decryption which is done with an exponential algorithm would then give you a value of x. Now let us look at a particular and very simple fault attack where an attacker could determine 1 bit of a that is 1 bit of the private key a. Now what the attacker would do is he would first consider a bit fault model where during the execution of this RSA decryption h forces 1 bit of a to go from 0 to 1 right.

So that means if the bit of a is 0 then he would inject a bit fault and change that particular bit to 1. On the other hand if the value of a in that ith bit was 1 the attackers fault would force the value of a to go to 0. Let us say that the value of a is equal to say 0 1 1 0 and the ith bit we will take i to be having a value of 2 so we considering the bit fault model and therefore the attacker targets this specific bit for example he would focus his laser beam on this bit and change this bit from a value of 1 to a faulty value of 0. So this is the faulty value of a. Note that he attacker has does not know the value of a because a is actually embedded into the device in the device but rather he somehow has figured out how to inject a fault that could potentially change this bit.

So without knowing the bit what the value of the bit is the attacker has somehow injected a fault to toggle the ith value of this bit. So we have taken this particular example where a which is secret has a value of 0 1 1 0 and the attacker has somehow due to the fault injection change the value of a to 0 0 1 0. Now the decryption goes on as normal now when the fault is not injected x would have the value y to power of 0 1 1 0 just the base 2 mod n while in the case when the fault is injected he would get a wrong value which we denote as this which is y to power of 0 0 1 0to base 2 mod n.

So what we identify is that the value of x and x tilda is going to be different. Now what the attacker has in his position these two values x and x tilda form this he needs to identify that the original value for this ith bit should be 1.in order to do so the first thing we observe is that if we subtract a and a tilda a minus a tilda we would get 6 minus 2 which is essentially plus 4.on the other hand let us say that a had a value of 2 injecting a fault so this was one condition and we have now assuming that a has a value of 2 and injecting a fault in the (ith) equal to two location the attacker has somehow be able to change a to 6 like this.

Now in this particular case he would have obtained a minus a tilda to be 2 minus 6 to be equal to minus 4 right. So what you see here is that a minus a tilde (tilda) is either equal to plus 2 to the power of I where I is 2 in this case or minus total power of I so you see that depending on whether the original value was 1 or 0o would get either a positive value of 2 to the power of I or a negative value of 2 to the power of I. However what the attacker only has is x and the x tilda. So how does hew obtain this difference 2 of I or minus 2 of I. So in ordered to do this what he does is he takes x and he devise it by x tilda as follows.

So we have x and divide it by x tilda so essentially you would get y to the power of a mod n divided by y a tilda mod n this is nothing but y a minus a tilda mod n right so this is essentially either y 2 to the power of I mod n if the ith bit over here had a value of 1 or he would get the value of y minus 2 to the power of i mod n. So you see that dividing x and x tilda would either give him a value of y 2 to the power of i mod n y power minus 2 i a power I mod n. So since the attacker also knows the value of y that is the cipher text he can pre-compute what is the value of y 2 to power of i mod n as well as y minus 2 power of i mod n and given x and x tilda he can determine whether it was either this case or this case and this infer whether the ith bit was 1 or a 0.

With this way the attacker would obtain 1 bit of the secret key. Similarly by injecting false and every other bit the attacker get slowly be able to recover every bit of the secret key. So what we see over here is that this attack is extremely easy provided that the attacker is precisely able to inject 1 bit fault in the secret key of the RSA. So looking at the practicality of this attack it is not going to be that easy because injecting bit false is first of all difficult and in this particular case the attacker would need to inject a large number of faults so if the key side is say 1024 bit he would need to inject 1024 bit false in order to fully recover the secret key.

So there have been many further attacks on RSA so many further fault attacks on RSA which have been much more efficient and which were able to more efficiently get the secret key extracted from the RSA device but we will not go into those details and I would leave it to the interested readers to actually look at the papers that follow this and look at the various attacks, thank you.