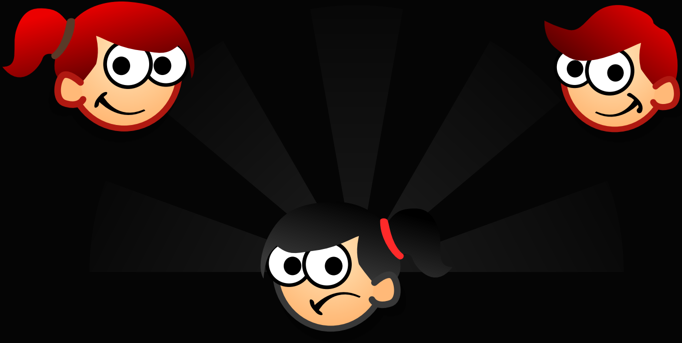
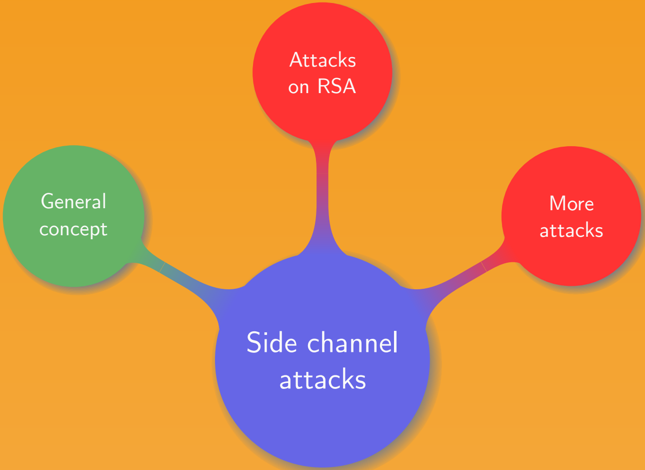


Introduction to Cryptography

10. Side channel attacks

Manuel – Summer 2019

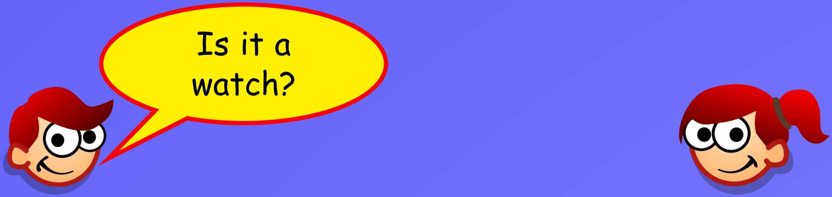
Chapter organisation



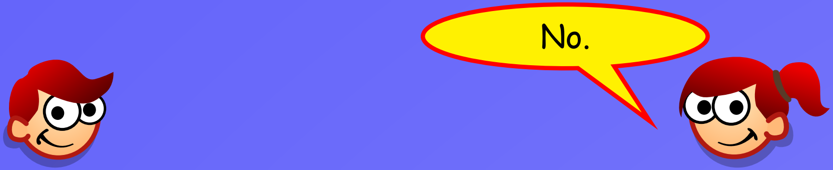
Basic idea

Curious Bob want to know what is his present

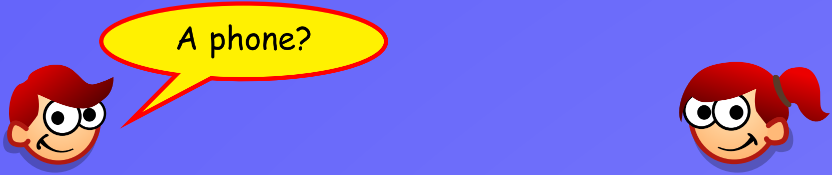
Basic idea



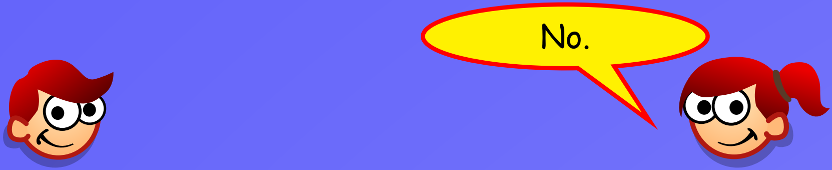
Basic idea



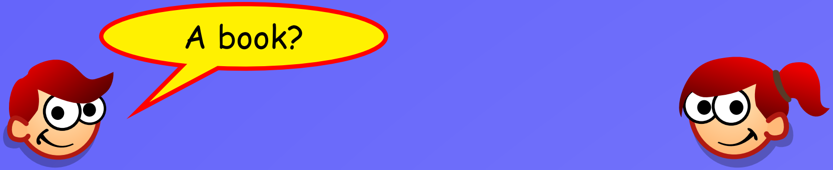
Basic idea



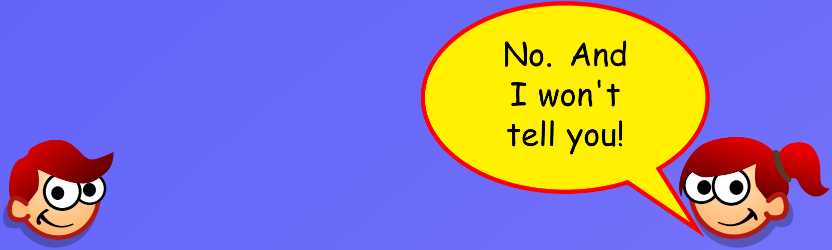
Basic idea



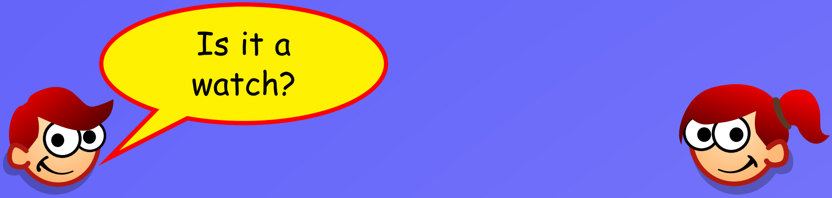
Basic idea



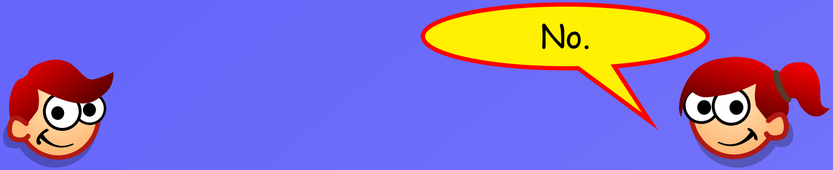
Basic idea



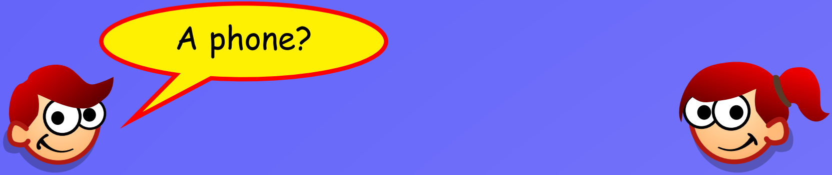
Replay



Replay

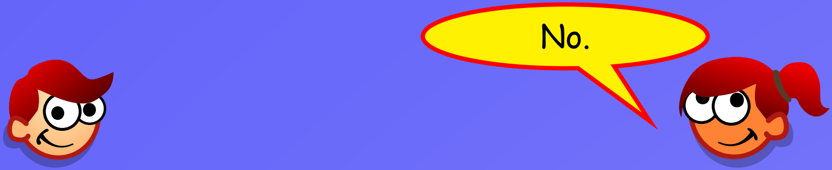


Replay

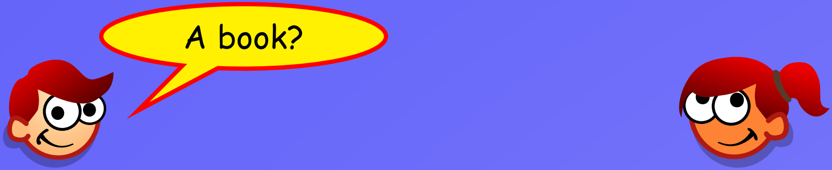


5

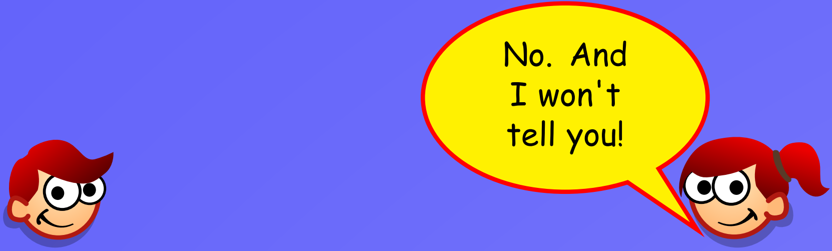
Replay



Replay



Replay



Side channel attacks

Computers leak information when performing an operation:

- Amount of power used
- Time spent
- Area of the memory used
- Electromagnetic radiations
- Sounds (hard disk, beep...)
- Frequency of sending packets on the network

General remarks

A few important notes:

- Side channel attacks apply to any protocol
- A secure cipher with no attack on the protocol is not immune to side channel attacks
- Not many way to get protected

General remarks

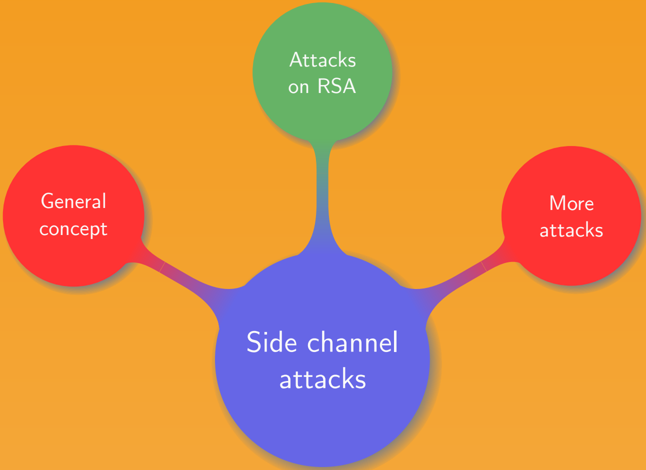
A few important notes:

- Side channel attacks apply to any protocol
- A secure cipher with no attack on the protocol is not immune to side channel attacks
- Not many way to get protected

Example.
SSH is a secure way to connect to a remote computer. If a user authenticate using a password, the time between each keystroke, or packet sent on the network, can be analysed and the password recovered.

An audio recording of someone typing on a keyboard is enough to know what he is writing.

Chapter organisation



RSA implementations

To break RSA the goal is to find the secret key:

- When a user decrypts a message
- When a user signs a message

RSA implementations

To break RSA the goal is to find the secret key:

- When a user decrypts a message
- When a user signs a message

The attacker can access to the host device to:

- Run some malicious code
- Perform measurements

RSA implementations

To break RSA the goal is to find the secret key:

- When a user decrypts a message
- When a user signs a message

The attacker can access to the host device to:

- Run some malicious code
- Perform measurements

As running RSA leaks information the attacker only needs to read it and then perform some analysis in order to interpret it.

Timing attack

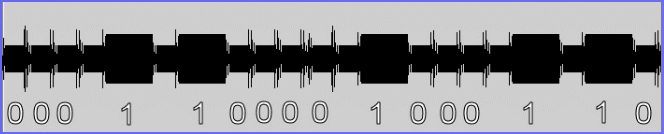
General approach:

- RSA decryption/signature uses the square and multiply algorithm (3.38)
- On a 0 only a squaring occurs
- On a 1 a squaring and a multiplication occur

The mean not being precise enough the variance is used to analyse a large number of decryption requests. The attacker then uses the fact that the variance of the sum of two independent random processes is the sum of their respective variance. He gains little information on the secret key but he reuses it to perform more accurate measurements and in the end he is able to totally recover the key.

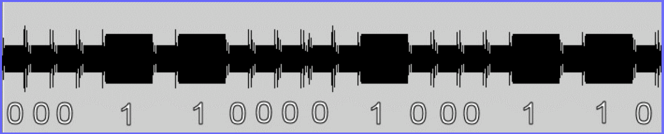
Power analysis attack

In this attack the key idea is to observe the power consumption of the computer when decrypting or signing a message.



Power analysis attack

In this attack the key idea is to observe the power consumption of the computer when decrypting or signing a message.



On a 1 both a square and a multiply are carried out. When only a squaring occurs the power consumption is much lower. This attack is clearly much more powerful and efficient than the previous one. In fact no more than one decryption is necessary to recover the secret key.

We know present an algorithm that performs modular exponentiation without leaking much information.

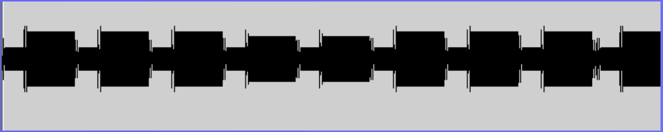
Montgomery's ladder technique

Algorithm. (*Modular exponentiation*)

```
Input  : $m$ an integer, $d = (d_{k-1} \dots d_0)_2$ and $n$ two positive integers
Output: $x = m^d \bmod n$
power1 ← m; power2 ← m^2;
for $i \leftarrow k-2$ to 0 do
    if $d_i = 0$ then
        $power2 \leftarrow (power1 \cdot power2) \bmod n$;
        $power1 \leftarrow power1^2 \bmod n$;
    else
        $power1 \leftarrow (power1 \cdot power2) \bmod n$;
        $power2 \leftarrow power2^2 \bmod n$;
    end if
end for
return $power1$
```

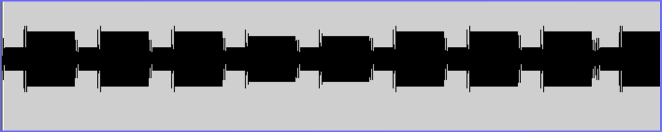
Preventing attacks on RSA

The previous algorithm has the advantage of performing both multiplication and squaring whatever the bit considered. Therefore monitoring the power consumption would not bring any information on the secret key as it would look as on the following picture.



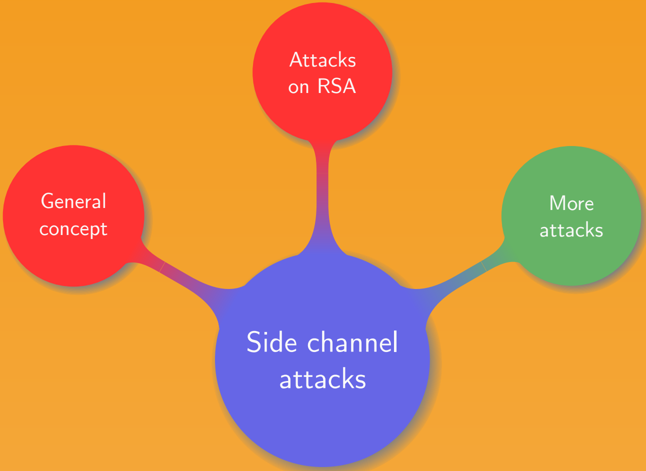
Preventing attacks on RSA

The previous algorithm has the advantage of performing both multiplication and squaring whatever the bit considered. Therefore monitoring the power consumption would not bring any information on the secret key as it would look as on the following picture.



Note that the Montgomery's ladder technique is still vulnerable to cache timing attacks. Indeed an attacker could measure the time necessary to access the memory, and as this depends on which variable is used he could recover some information on the bits composing the secret key.

Chapter organisation



Reverse engineering

Extract information on the design and process of a program:

- Get the binary file
- Disassemble it
- Understand the generated assembly code
- Extract some important information

Reverse engineering

Extract information on the design and process of a program:

- Get the binary file
- Disassemble it
- Understand the generated assembly code
- Extract some important information

Never store a secret key in a binary, “everybody” can retrieve it

Background on memory

Page sharing:

- Share parts of the memory between processes
- Avoid replicated copies of identical content
- Pages are read-only
- On a write request, copy the page onto a new writable location

Cache structure in modern processors:

- Each core has two levels of cache L1 and L2
- A third level L3 is shared among all the cores
- Removing data from L3 also flushes it from both L1 and L2
- The closer from the CPU the faster to retrieve data

A cache timing attack measures how long it takes the CPU to fetch the data. This leaks information on what operation is performed.

L3 cache side channel attack

High level idea applied by the attacker:

- Use `mmap` to map the victim's executable file into the attacker's address space
- Mark up memory lines related to specific operations
- Flush the memory line from the L3 cache and wait
- Call the memory line and measure how long it takes it load it
- Slow loading: the line was not called by the victim
- Fast loading: the line is in the cache, meaning it was used

It is impossible to prevent this attack on a multi-user system as it is inherent to the implementation of the X86 architecture. Only a hardware fix could solve this weakness.

Preventing side channel attacks

Simple conclusion:

- It is impossible to prevent all the kinds of side channel attack
- A system can never be 100% secure

Preventing side channel attacks

Simple conclusion:

- It is impossible to prevent all the kinds of side channel attack
- A system can never be 100% secure

What can be done:

- Chose secure protocols
- Select secure libraries
- GMP implements function intended for a cryptographic use
- Such functions are slower but more resistant to side channel attacks

A final example

VMWare View is a popular remote desktop protocol. VMWare recommends to switch from AES-128 to SALSA20-256 for the "best user experience".

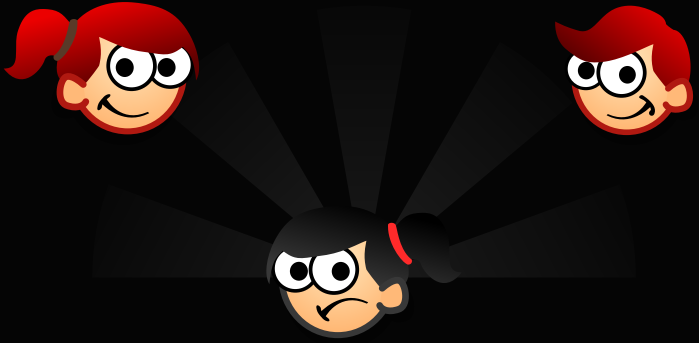
A closer look at the specs shows that AES-128 refers to AES-128-GCM, which includes AES and message authentication. On the other hand SALSA20-256 refers to SALSA20-256-Round12, which does not feature any message authentication.

Although SALSA20 has speed and security advantages over AES an attacker can easily forge packets if it is not used in conjunction with with message authentication.

Is it worth sacrificing security for the sake of speed?

Key points

- Explain what are side channel attacks
- List two examples of side channel attacks
- How to prevent side channel attacks?
- Can a system be made fully secure?



Thank you!

References I

10.11 G. Hollestelle, W. Burgers, J. den Hartog, *Power analysis on smartcard algorithms using simulation*

10.13 G. Hollestelle, W. Burgers, J. den Hartog, *Power analysis on smartcard algorithms using simulation*