Universidad de Buenos Aires

Facultad de Ciencias exactas y Naturales

Licenciatura en Ciencias de la computación

# Apunte de Seguridad de la Información

*Autores:*
Julián Sackmann
Juan Pablo Darago

2 de Noviembre de 2013




**Facultad de Ciencias Exactas y Naturales**
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://exactas.uba.ar

# Índice

# 1 Unidad 1

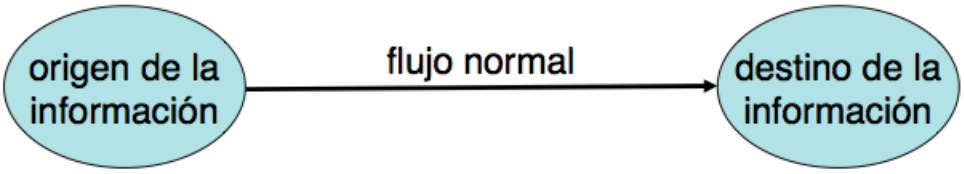
## 1.1 Definiciones

- **Información**: se refiere a toda comunicación o representación de conocimiento.
- **Seguridad de la información**: se entiende como la preservación de: **Confidencialidad**, **Integridad** y **Disponibilidad**.
- **Confidencialidad**: se garantiza que la información sea accesible sólo por aquellas personas autorizadas.
- **Integridad**: se salvaguarda la exactitud y totalidad de la información y los métodos de procesamiento. La integridad incluye: la integridad de los datos (el contenido) y el origen de los mismos.
  - Integridad de datos: nadie altere el contenido.
  - Integridad de origen: el origen de los datos sea cierto.
- **Disponibilidad**: se garantiza que los usuarios autorizados tengan acceso a la información y a los recursos relacionados con la misma, toda vez que lo requieran.
- **Vulnerabilidad**: una debilidad en un activo.
- **Amenaza**: una violación potencial de la seguridad. No es necesario que la violación ocurra para que la amenaza exista. Las amenazas "explotan" vulnerabilidades.
- **Ataque**: una acción que puede causar una violación.
- **Atacante (o intruso)**: persona o entidad que ejecuta un ataque.

## 1.2 Tipos de amenazas

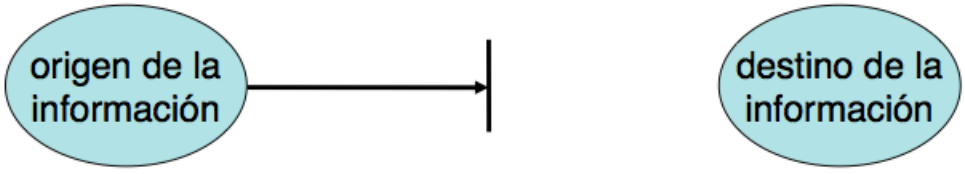
### 1.2.1 Flujo

Representamos el flujo normal de la información como un grafo donde los nódos son los entes que intercambian información y las aristas dirigidas la información que se transmite (mediante algún canal) entre un nodo y otro:



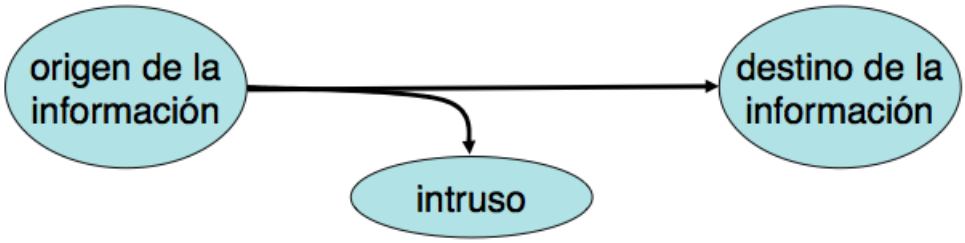

Las amenazas se pueden categorizar en las siguientes formas:

**Interrupción de flujo**



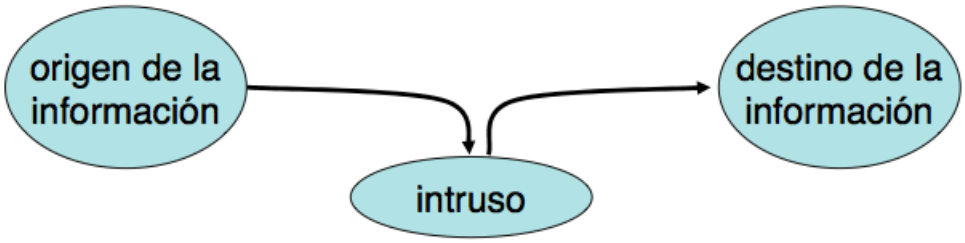
El flujo de datos entre un nodo y otro es completamente interrumppido. Amenaza la **disponibilidad** de la información. Puede darse por *bloqueo*, *saturación* o *destrucción* del recurso. Un ejemplo de esta amenaza es un *denial of service*.

**Intercepción de flujo**



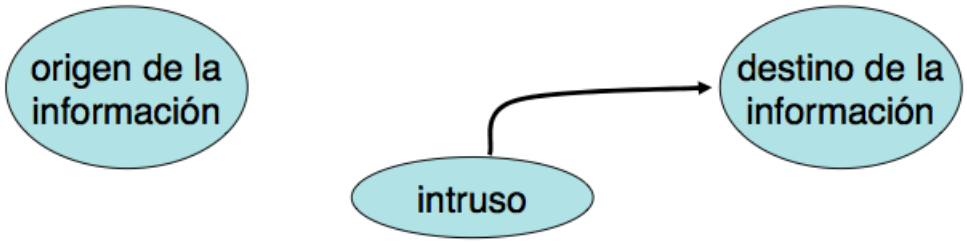

El flujo de datos entre un nodo y otro es interceptado por un atacante, quién accede a la información, pero no la modifica. Amenaza la **confidencialidad** de la información. Puede darse por un *acceso no autorizado* al recurso, *monitoreo* o *ingeniería social*. Un ejemplo de esta amenaza es un *keylogger*.

**Alteración de flujo**




El flujo de datos entre un nodo y otro es interceptado por un atacante que lee la información y la modifica. Amenaza la **integridad** de la información. Un ejemplo de alteración de flujo son los ataques *man in the middle*.

**Fabricación de flujo**




Un atacante se hace pasar el emisor y le envía información al receptor. Amenaza la **autenticidad** (se considera parte **integridad de origen**) de la información. Un ejemplo de alteración de flujo es el *phishing*.

### 1.2.2 Pasividad del atacante

A su vez, las amenazas se pueden categorizarse en:

**Amenazas pasivas**

Son las que el atacante no interfiere con el sistema. Este sigue funcionando con normalidad. Suelen ser amenazas más difíciles de detectar y dependen fuertemente del medio físico de transmisión.

Ejemplos:

- *Sniffing*
- *Side Channel Attack* (se basa en obtener información de la implementación física de un criptosistema. Ej: "usuario o clave inválidos" vs "clave inválida").

**Amenazas activas**

Son las que el atacante interactúa con el sistema, no se limita a observar. Realiza acciones que afectan al sistema con diversos objetivos (obtener más información, asegurar que pueda volver a realizar la amenaza, modificar información, etc.)

Algunos ejemplos son:

- *Keyloggers*
- Suplantación de identidad.
- Retransmisión de mensajes.
- Falsificación de datos.
- Escaneo de puertos.

### 1.2.3 Otras

Las amenazas también se pueden categorizar en:

- Intencionales vs accidentales.
- Interas vs externas.

### 1.2.4 Resumiendo

En resumen, las amenazas se categorizan en:

- Flujo
  - Interrupción
  - Intercepción
  - Alteración
  - Fabricación
- Activo vs Pasivo
- Intencionales vs accidentales.
- Interas vs externas.

## 1.3 Políticas y Mecanismos

Una **política de seguridad** es una declaración de lo que está permitido y lo que no. *Es el qué.*

Las políticas de seguridad son afectadas tanto por cuestiones tecnológicas como administrativas, de gerencia, etc.

Un **mecanismo de seguridad** es un método, herramienta o procedimiento que intenta hacer cumplir una (o parte de una) política de seguridad. *Es el cómo.*

Los mecanismos de seguridad no son necesariamente automáticos y/o técnicos. Por ejemplo: pedir DNI para entrar a tal habitación.

## 1.4 Objetivos

El objetivo ideal de las políticas de seguridad es prevenir la ocurrencia de todo ataque. Como esto es imposible, se considera que la seguridad tiene tres objetivos:

- **Prevención**: si un mecanismo logra este objetivo, entonces el ataque va a fallar. Un ejemplo de esto es un ataque a través de internet a una computadora que no está conectada a ninguna red.
- **Detección**: si un mecanismo logra este objetivo, entonces el ataque será detectado. Puede ser usado tanto cuando el ataque no puede ser prevenido (por ejemplo para minimizar su impacto, o reaccionar de alguna forma) como cuando si (por ejemplo, para medir la efectividad del mecanismo de prevención). Los mecanismos de detección dan por hecho que un ataque va a ocurrir e intentan reportarlos lo antes posible.
- **Recuperación**: si un mecanismo de seguridad logra este objetivo, significa que se pueden deshacer las consecuencias de un ataque. Luego de un ataque, se deben analizar y reparar daños y volver el sistema a operación normal en el menor tiempo posible. Por ejemplo, si un archivo fue borrado, puede ser recuperado de un backup.

## 1.5 Relación costo-beneficio

Es importante notar que a la hora de implementar cualquier combinación de mecanismo/política de seguridad es necesario tener en consideración la relación costo/beneficio entre:

- El costo de implementar la política y el mecanismo de seguridad.
- El valor de la información que quiero proteger.

El costo de implementación no sólo hace referencia al costo monetario de poner en funcionamiento el mecanismo, sino también costos adicionales tales como:

- Tiempo de implementación.
- Costo computacional.
- Complicar la interacción con el usuario.

## 1.6 Evaluación de riesgos

Se entiende por **evaluación de riesgos** a la evaluación de las amenazas y vulnerabilidades relativas a la información y a las instalaciones de procesamiento de la misma, la probabilidad de que ocurran y su potencial impacto en la operatoria de la organización.

Informalmente, consiste en saber cuáles son lo puntos más riesgosos riesgosos de la seguridad de una organización, qué impacto tendría su potencial violación y cómo se manejan.

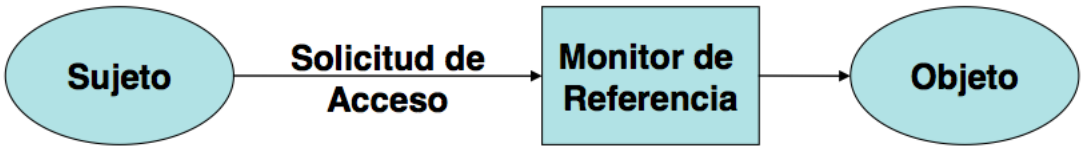
# 2 Unidad 2

## 2.1 Definiciones

- **Estado de un sistema**: es el conjunto de los valores actuales de todas las posiciones de memoria, almacenamiento secundario, registros, y otros componentes del sistema.
- **Estado de protección de un sistema**: es el subconjunto de estados del sistema relacionado con la seguridad y protección del mismo.
- **Sujetos**: entidades (usuarios, grupos, roles y procesos) que modifican los objetos y, en consecuencia, cambian el estado del sistema.
- **Objetos**: entidades que son relevantes para el estado de protección del sistema, que deben ser protegidas. Por ejemplo: memoria, archivos, datos, directorios, programas, usuarios, etc.
- **Permisos**: acciones autorizadas que un sujeto puede realizar sobre un objeto.
- **Solicitud de acceso**: es una acción en la que un sujeto le pide al sistema acceso a un objeto.

### 2.1.1 Monitor de referencia

Es un mecanismo encargado de mediar cuando los sujetos intentan realizar operaciones sobre los objetos en cualquier política de acceso.



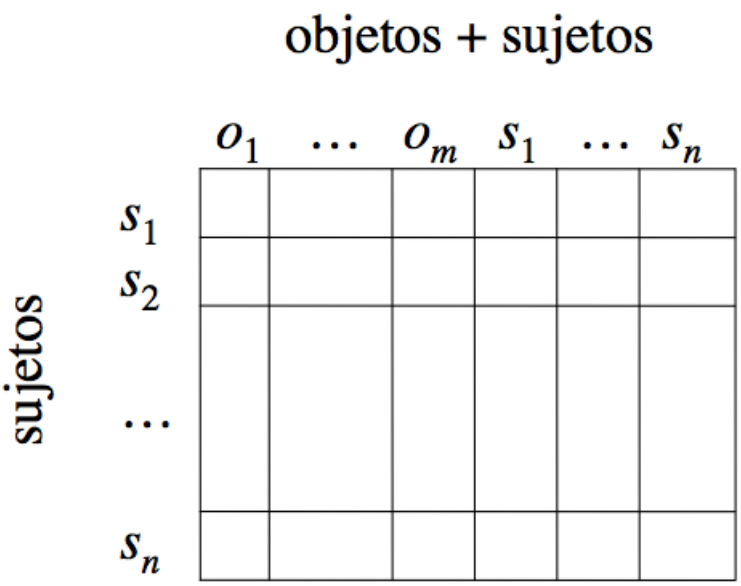

Los monitores de referencia cumplen las siguientes propiedades:

- Intermediación obligatoria: el MR debe intervenir en **todos** los accesos.
- Aislamiento: tanto el MR como sus datos deben ser incorruptibles y seguros.
- Verificabilidades: debe ser demostrable que su funcionamiento es correcto.

### 2.1.2 Matriz de control de acceso

Es un modelo conceptual que describe, de manera precisa, el estado de protección del sistema.

Suele concebirse como una matriz que representa los permisos que tienen los sujetos (usuarios o procesos) sobre los distintos objetos. En las filas se ubican los sujetos, mientras que en las columnas se ubican los objetos+sujetos. En la celda $i, j$ se ubican los permisos que el sujeto $i$ tiene sobre el objeto o sujeto $j$. Definimos **permisos** como "derechos de acceso a un recurso que se asignan a usuarios individuales o a grupos de usuarios".




Los permisos de la celda pueden ser:

- r = read.
- w = write.
- x = execute.
- a = append.
- o = own.

Ejemplo:

| | arch1 | arch2 | proc1 | proc2 |
|---|---|---|---|---|
| proc1 | *rwo* | *r* | *rwxo* | *w* |
| proc2 | *a* | *ro* | *r* | *rwxo* |

En la realidad, este modelo no es implementable en forma directa como matriz, porque el tamaño de esa matriz sería inmanejable. Se implementa con listas, en dos formas:

- Lista de control de acceso.
- Lista de capacidades.

**Lista de control de acceso**

Para cada objeto, tengo una lista de sujetos que indica los permisos que posee cada sujeto sobre el objeto. Es como almacenar la matriz por columnas

Ej: se usa normalmente para *filesystems*, puesto que dado un objeto, es muy fácil ver qué usuarios tienen pemisos sobre él. Además, es muy fácil revocar los permisos del objeto, reemplazándolo por una lista vacía. Sin embargo, no es trivial ver todos los objetos a los que puede acceder un sujeto.

**Lista de capacidades**

Para cada sujeto, tengo una lista de objetos que indica los permisos que posee ese sujeto para cada objeto. Es como almacenar la matriz por filas.

Si bien es más fácil chequear todos los objetos a los que puede acceder un sujeto dado (así como revocar sus permisos), es más complicado, dado un obeto, responder quiénes tienen permisos sobre él.

## 2.2 Control de acceso

El objetivo de los controles de acceso es poder controlar quién y cómo accede al sistema y sus recursos. Esto separa las estrategias de control de acceso en dos ramas:

- Control de acceso al sistema: su objetivo es permitir el acceso al sistema sólo a aquellos usuarios que están autorizados.
- Control de acceso a los recursos: establece qué usuarios pueden acceder a qué recursos y cómo. Los recursos sólo deben poder ser accedidos por usuarios autorizados con permisos explícitos y las acciones que estos realizan deben ser las permitidas.

En general las estrategias para control de acceso se clasifican en:

- Control de acceso discrecional (`DAC`).
- Control de acceso mandatorio (`MAC`).
- Control de acceso basado en roles (`RBAC`).

### 2.2.1 Control de acceso discrecional

Es una política en la cual el sujeto que define los permisos sobre un recurso es el dueño del recurso. Se dice que es *discrecional*, porque queda a discreción del dueño los permisos de un objeto.

Para este control de acceso se definen conceptos claves tales como:

- Propiedad: cada objeto en el sistema debe tener un dueño. Lo normal es que el dueño sea quien crea el recurso.
- Permisos: derechos de acceso que el dueño de un recurso asigna a usuarios individuales o a grupos de usuarios.

**Ventajas y Desventajas**

Ventajas:

- Es muy flexible y versátil.
- Fácilmente puedo descentralizar el sistema de permisos.

Desvenajas:

- Es peligroso: por error o impericia un usuario puede estar otorgando permisos incorrectos a otro sin siquiera saberlo.
- Es obligatorio el concepto de "dueño" o "propietario" de un recurso.
- También hay discrecionalidad para la revocación de derechos.

En general existe un usuario privilegiado (*root*) con la capacidad especial de adueñarse de cualquier recurso.

Además existen atributos extendidos, como *chattr* que permite modificar atributos extendidos tales como *append only*, *immutable*, etc.

### 2.2.2 Control de acceso mandatorio

Es una política en la cual el acceso a un recurso es determinado por el sistema y no por el dueño del recurso. Busca evitar que el usuario no pueda modificar los permisos que el sistema le otorgó. Para eso, se basa en clasificar a los sujetos y objetos en base a **niveles de seguridad**. Se definen restricciones muy fuertes sólo como moverse de un nivel a otro.

Windows implementa políticas de control de acceso discrecional con el `Integrity Level` y Linux lo hace con `SELinux`.

Los sistemas multiniveles manejan múltiples niveles de clasificación entre sujetos y objetos. Se usa para sistemas gubernamentales y/o militares.

### 2.2.3 Control de acceso basado en roles

Es una política de control de acceso que surge con mucha fuerza en la década del '90. Combina aspectos de control mandatorio y discrecional, pero intentando mantener una estructura jerárquica organizacional de una empresa u organización.

Básicamente consiste en la creación de **roles** para los trabajos o funciones que se realizan en la organización. Se define un rol como *el conjunto de acciones y responsabilidades asociadas con una actividad en particular*. A las personas se les asignan roles que les permiten obtener los permisos para ejecutar funciones del sistema. Los permisos se asignan a los roles, no a los usuarios.

Los sujetos acceden a los objetos en base a las actividades que (los sujetos) llevan a cabo en el sistema. Es decir, considerando los roles que ocupan en el sistema.

Para implementar `RBAC` se necesitan mecanismos que permitan:

- Identificar los roles en el sistema y asignar los sujetos a roles.
- Establecer los permisos de acceso a los objetos para cada rol.
- Establecer permisos a los sujetos para que puedan adoptar roles.

El concepto de **separación de responsabilidades** se basa en el principio de que ningún usuario tenga suficientes privilegios para usar el sistema en su propio beneficio. Se debe poder definir dentro del modelo restricciones relacionadas con prevenir que un usuario legítimo en forma maliciosa pueda hacer abuso del sistema. La separación de reponsabilidades se puede implementar estática o dinámicamente. En su implementación estática se definen los roles *excluyentes* para un mismo usuario. Los roles excluyentes aseguran que un usuario no pueda realizar una acción maliciosa obteniendo permisos de dos roles al mismo tiempo (por más que tenga permisos para obtener ambos). En su versión dinámica se realiza el control al momento de acceso.

El concepto de **menor privilegio** si una tarea no va a ser ejecutada por un usuario, entonces su rol no debe tener los permisos para hacerla. De esta manera se minimizan riesgos de seguridad

Algunas aplicaciones que implementan `RBAC`:

- Sun Solaris
- Microsoft Active Directory
- Base de datos Oracle
- Base de Datos MS Sql Server

**Ventajas y Desventajas**

Ventajas:

- Administración de autorizaciones.
- Facilidad de cambio de permisos cuando cambia un rol.
- Jerarquía de roles (meter roles adentro de otros y herencia de permisos).
- Menor privilegio.
- Separación de responsabilidades.

Desventajas:

- Es difícil la tarea de establecer los roles y definir sus alcances.

## 2.3 Políticas de seguridad

Formalmente, podríamos considerar un sistema como un autómata finito con funciones que permiten cambiar de un estado a otro. En este contexto, una **política de seguridad** es una declaración que particiona el conjunto de estados en:

- Autorizados / seguros: estados en los que el sistema puede entrar.
- No autorizados / no seguros: estados en los que, si el sistema entra hay una violación de seguridad.

Un sistema se considera **seguro** si comienza en un estado autorizado y que, si el sistema está en un estado seguro, toda transición lo llevará a otro estado seguro.

Existen distintos tipos de políticas de seguridad. En general se las categorizan en:

- Políticas de confidencialidad.
- Políticas de integridad.

### 2.3.1 Política de confidencialidad: Modelo Bell-Lapadula

Es una política de confidencialidad cuyo objetivo es prevenir el acceso no autorizado a la información. No se considera un objetivo primario el evitar las modificaciones no autorizadas. Fue concebida en los '70-'80, principalmente con fines militares. Controla el flujo de la información combinando técnicas de control de acceso mandatorio y discrecional.

Este modelo define niveles de clasificación de seguridad ordenados:

- Alto secreto.

- Secreto.
- Confidencial.
- No clasificada.

Para establecer los permisos de un sujeto sobre un objeto, se definen **habilitaciones** de seguridad para los **sujetos** ($L(s)$) y **clasificaciones** para los **objetos** ($L(o)$).

Ante un requerimiento, el sistema lo admite o rechaza considerando las habilitaciones y clasificaciones involucradas. Para eso se aplican dos principios:

**Read down**

El sujeto $s$ puede leer el objeto $o$ sii $L(o) \leq L(s)$. Coloquialmente, **un sujeto puede leer todo objeto que esté en su nivel o menos**.

**Write up**

El sujeto $s$ puede escribir el objeto $o$ sii $L(o) \geq L(s)$. Coloquialmente, **un sujeto puede escribir todo objeto que esté en su nivel o más**.

La idea detrás de este principio es evitar que un usuario le filtre información confidencial a un sujeto de menos prioridad escribiéndo esa información con menos prioridad.

Ambos principios son aplicables **si además el sujeto $s$ tiene permiso discrecional para escribir el objeto $o$.**

Además se expanden estos conceptos agregando **categorías**, que agrupan información relacionada entre si: representan distintas áreas de información dentro de un mismo nivel y no responden a un esquema jerárquico. Con esto se define el **nivel de seguridad** como la tupla *(habilitación, conjunto de categorías)*.

Se define la **dominancia** de la siguiente forma: $(A, C) dom (A', C') sii A' \leq A$ y $C' \subseteq C$.

Luego, podemos definir los principios de **read down** y **write up** como

- El sujeto $s$ puede leer el objeto $o$ sii $L(s) dom L(o)$ y $s$ tiene permiso para leer $o$.
- El sujeto $s$ puede escribir el objeto $o$ sii $L(o) dom L(s)$ y $s$ tiene permiso para escribir $o$.

## 2.4 Política de integridad

Las políticas de integridad tienen como principal objetivo preservar los datos y su integridad. Es importante identificar las maneras autorizadas en las cuales la información puede ser alterada y cuales son las entidades autorizadas para alterarlas.

Sus principios de operación consisten en:

- Separación de tareas.
- Separación de funciones.
- Auditabilidad.

### 2.4.1 Modelo BIBA

Es un modelo que implementa política de integridad. Se lo puede pensar como el inverso de *Bell-Lapadula*. Cuanto más alto el nivel de integridad, más confianza en que:

- Un programa ejecutará correctamente.
- La información es correcta y/o confiable.

Se plantean dos principios de **read up** y **write down**:

- Un sujeto $s$ puede leer un objeto $o$ sii $i(s) \leq i(o)$.
- Un sujeto $s$ puede leer un objeto $o$ sii $i(s) \geq i(o)$.

### 2.4.2 Modelo Clark-Wilson

Es un modelo que implementa la política de integridad y utiliza la noción de transacción: el sistema comienza en un estado inicial consistente y se realizan una serie de acciones (llamadas *transacciones*) que verifican que:

- No pueden ser interrumpidas.
- Si se completan, el sistema queda en un estado consistente.
- Si no se completan, el sistema vuelve al estado anterior.

## 2.5 Políticas Híbridas: Pared China

Organiza las entidades en clases de **conflictos de interés**. Se debe controlar el acceso de los sujetos a cada clase. Se controla la escritura a todas las clases para asegurarse que la información no es pasada de una a otra violando las reglas. La pared china permite que todos vean la información *esterilizada* (información pública).

Se definen los siguientes conceptos:

- Objetos: elementos de información relacionados con una compañía.
- Company dataset ($CD$): contiene objetos relacionados con una compañía.
- Clase de Conflicto de Interés ($COI$): contiene datasets de compañías que compiten entre si. Se asume que cada objeto pertenece a una sola clase de conflicto de interés.

**Condición de seguridad simple**

El sujeto $s$ puede leer el objeto $o$ sii alguna de las siguientes condiciones se cumple:

- Existe un objeto $o'$ tan que $s$ ha leido $o'$ y $\text{CD}(o') = \text{CD}(o)$.

  (Es decir, $s$ leyó previamente algún dato en el dataset de la compañía).
- Para todo $o' \in O$ tal que $s$ ya leyó $o'$, entonces $COI(o') \neq COI(o)$.

  (Es decir, $s$ no leyó ningún objeto de algún dataset de la compañía en la misma clase de conflicto de interés).

### 2.5.1 Comparación con Bell-LaPadula

- Bell-LaPadula no puede llevar un histórico de cambios en el tiempo.
- Armando se enferma, Nancy debe reemplazarlo: Pared China permite determinar si Nancy puede hacerlo, mientras que Bell-LaPadula no puede decir nada al respecto.
- Las restricciones de acceso cambian por el tiempo: en Pared china, inicialmente todos los sujetos pueden leer cualquier objeto y va cambiando, mientras que Bell-LaPadula es estático.
- En pared china no se puede borrar a todos los sujetos de todas las categorías porque viola la condición de seguridad simple.

## 2.6 ORCON

Es un método que intenta resolver el problema de controlar la diseminación de los documentos generados dentro de una organización. Se quiere poder distribuir la información a quien quiera pero garantizar que esa persona no va a poder distribuirla.

## 2.7 Windows Mandatory Integrity Control

Es un control de acceso mandatorio implementado a partir de Windows Vista, basado en el modelo Biba de control de integridad. Define 4 niveles de integridad

- Low.
- Medium.
- High.
- System.

Todos los archivos, carpetas, usuarios y procesos tienen niveles de integridad. El nivel medio es el defecto. Un sujeto no puede darle a un objeto un nivel de integridad más alto que el suyo.

## 2.8 Covert Channel

Un **covert channel** (o canal secreto) es un mecanismo de comunicación que no fue diseñado para ser utilizado con ese fin. Un ejemplo de esto es utilizar una carpeta (*/tmp*) donde cualquiera puede escribir. Se podría definir un protocolo mediante el cual cada cierto tiempo creo o no un archivo. No puedo ver el contenido del archivo, pero crear y borrar un archivo puede encodear un mensaje binario.

## 2.9 Side Channel

Un **side channel** es un ataque basado en información obtenida de la implementación del algoritmo criptográfico y no basada en una debilidad del algoritmo en si.

Hay numerosos tipos de ataques side channel:

- Tiempo: cuánto tardan ciertos cómputos.
- Consumo eléctrico: basados en diferencias de consumo del hardware dependiendo de la operación realizada.
- Electromagnéticos: basados en información fugada como radiación.
- Acústico: basados en sonidos emitidos durante el cómputo.

# 3 Unidad 3: Criptografía

Definiciones:

- **Criptografía**: es una rama de la matemática que busca cifrar y descrifrar información utilizando métodos y técnicas que permitan el intercambio de mensajes de manera que sólo puedan ser leídos por las personas a quienes van dirigidos. Su objetivo es mantener la información cifrada secreta.
- **Criptoanálisis**: es el estudio de los métodos que se se utilizan para quebrar textos cifrados con objeto de recuperar la información original en ausencia de la clave.
- **Criptología**: es la ciencia que estudia las técnicas criptográficas y de criptoanálisis.
- **Cifra**: método o técnica que protege a un mensaje al aplicar un algoritmo criptográfico.
- **Esteganofrafía**: es la comunicación secreta lograda mediante ocultación de *la existencia* de un mensaje.
- **Atacante**: un sujeto cuya meta es quberar un criptosistema.

## 3.1 Criptosistema

Se puede definir un **criptosistema** como una tupla $(E, D, M, K, C)$ donde:

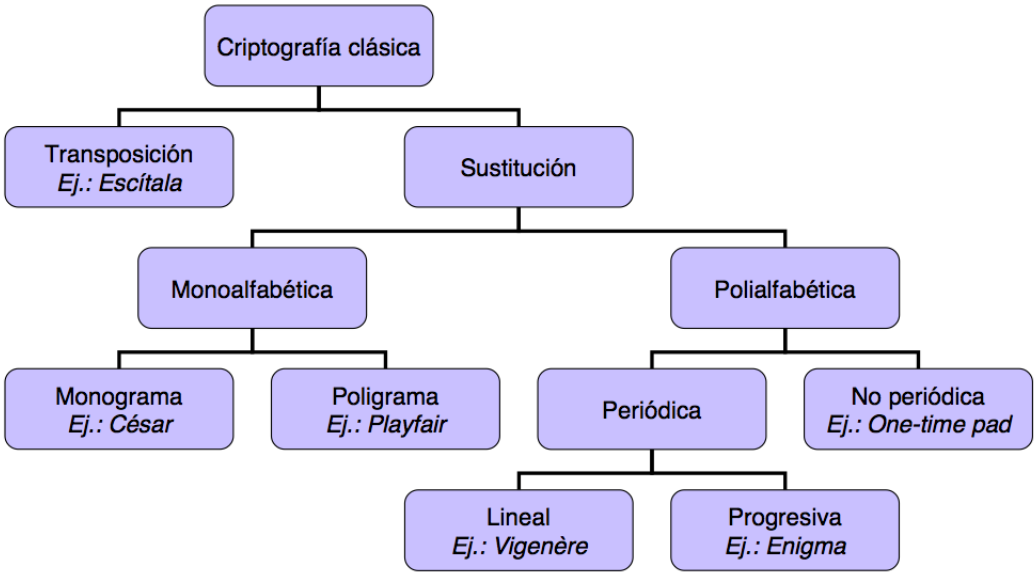
- $M$ es el conjunto de textos en claro.
- $K$ es el conjunto de claves.
- $C$ es el conjunto de textos cifrados.
- $E$ es el conjunto de funciones de cifrado ($e : M \times K \to C$).
- $D$ es el conjunto de funciones de descifrado ($d : C \times K \to M$).

## 3.2 Tipos de ataque

- Fuerza bruta.
- Sólo texto cifrado: el atacante sólo ve el texto cifrado.
- Texto en claro conocido: el atacante conoce el texto cifrado y el claro, pero no lo puede elegir.
- Texto en claro elegido: el atacante conoce el texto cifrado y el claro, y además lo puede elegir.
- Ataques matemáticos: basadaos en análisis matemáticos de los algoritmos.
- Ataques a la implementación: en muchos casos hay algoritmos que si bien son "correctos", su implementación.
- Ataques estadísticos: se basan en hacer suposiciones sobre la disposición de las letras (*monogramas*), pares de letras (*digramas*), etc. Examinar el texto cifrado y relacionar sus propiedades con las suposiciones realizadas.

## 3.3 Criptografía clásica

En la criptografía clase el emisor y receptor comparten una clave.
Hay tres tipos de criptografía clásica: **sustitución**, **transposición** y **combinados**.
Categorización:




**Cifra por sustitución**
Consiste en cambiar caracteres del texto en claro para producir el texto cifrado.
La cifra por sustitución puede ser:

- Monoalfabética: a una misma letra del mensaje en claro le corresponde siempre la misma letra del mensaje cifrado.
- Polialfabética: a una misma letra del mensaje en claro le pueden corresponder distintas letras en el mensaje cifrado.

Ejemplo: cifra de césar: es una sustitución monoalfabética monograma. Consiste en reemplazar cada letra del mensaje original por la letra que se encuentra tres lugares adelante en el alfabeto.

**Cifra por transposición**
Consiste en reorganizar los caracteres del texto en claro para producir el texto cifrado. En muchos casos en sigue una pauta simétrica: escribir por filas y leer por columnas o cosas así.
Ejemplo: Cifra de escítala.

### 3.3.1 Análisis de frecuencia

Es un tipo de análisis estadísico que se basa en el análisis de la frecuencia de aparición de los símbolos del texto cifrado y su intento de correlación con los símbolos del lenguaje en el cual está escrito el mensaje. Se buscan los caracteres más frecuentes en el criptograma y se los asocia a las letras de mayor aparición en el idioma original. En general se prueban distintas alternativas hasta alcanzar un texto coherente.

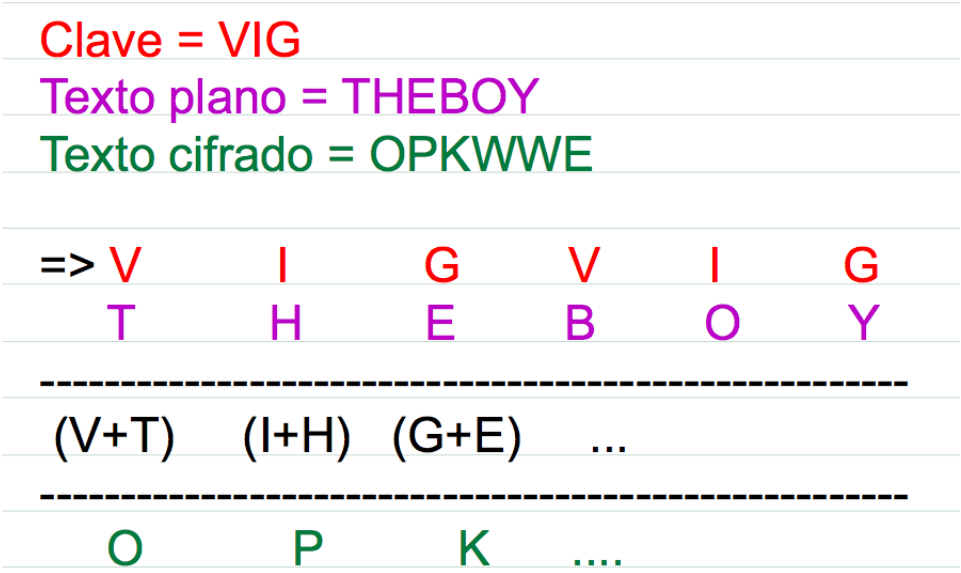
### 3.3.2 Cifra de Vigenère

Es una sustitución polialfabética periódica lineal en la que se usa una frase para establecer el corrimiento. El período es igual a la longitud de la clave. Durante años se consideró "indescifrable", pero en 1863 se inventó el **Ataque de Kasiski**. Consiste en:

- Buscar cadenas repetidas.
- Buscar el período de la clave obteniendo el $MCD$ (máximo común divisor) entre las posiciones de todas las cadenas repetidas.
- Descomponer el problema en $N$ sistemas monoalfabíticos (donde $N$ es el tamaño de la clave).
- Abordar cada sistema monoalfabético por medio del análisis de frecuencias.

Se define el **índice de coincidencias** como la probabilidad de que dos letras de un texto cifrado elegidas al azar sean la misma.

Ejemplo:




### 3.3.3 One-time pad

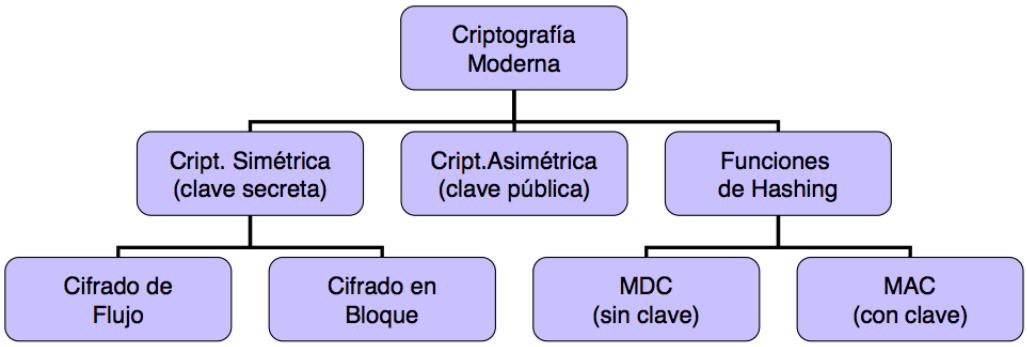
Es una clave de Vigenère con una clave aleatoria tan larga como el mensaje en claro. Es un sistema criptográfico perfectamente seguro porque dado un texto cifrado, todos los textos en claro son equiprobables.

Ejemplo: El texto cifrado `DXQR`, puede corresponder al texto en claro `DOIT` (cifrado con la clave `AJIY`) y al texto en claro `DONT` (cifrado con la clave `AJDY`) y a cualquier otra combinación de 4 letras.

Sin embargo, si tiene riesgos que pueden redundar en ataques:

- Las claves deben ser aleatorias, de no serlo se puede atacar tratando de regenerar la clave.
- Las claves se deben usar una sola vez.

## 3.4 Criptografía moderna




Se definen los principios de Kerckhoffs (1883):

- El sistema debe ser en la práctica imposible de cripoanalizar.
- La seguridad de un sistema criptográfico debe depender sólo de que la clave sea secreta y no de que el algoritmo de cifrado sea secreto.
- Método de elección de claves fácil de recordar.
- Transmisión del texto cifrado por telégrafo.
- La máquina de cifrar debe ser portable.
- No debe existir una larga lista de reglas de uso.

Shannon (1948) define **información** como el conjunto de datos o mensajes inteligibles creados con un lenguaje de representación. Ante varios mensajes posibles, aquel que tenga una menor probabilidad de aparición será el que contenga una mayor cantidad de información.

Un sistema criptográfico es **perfectamente seguro** si el texto cifrado no da ninguna información adicional sobre el texto en claro. Es decir, dado un texto cifrado $C$, cualquier posible texto plano es igualmente probable con respecto a $C$. Una propiedad de los sistemas perfectamente seguros es que la longitud de las claves es mayor o igual que la de los mensajes.

Un sistema es **incondicionalmente seguro** cuando es seguro frente a ataques con capacidad de cálculo ilimitada.

Un sistema es **computacionalmente seguro** cuando es seguro frente a ataques con capacidad de cálculo limitada.

Para mejorar las operaciones de cifra, Shannon propone dos técnicas:

- **Difusión**: es la transformación del texto claro con el objeto de dispersar las propiedades estadísticas del lenguaje sobre el criptograma. Se logra con *transposiciones*.
- **Confusión**: es la transformación del texto claro con el objeto de mezclar los elementos de éste, aumentando la complejidad de la dependencia funcional entre la clave y el criptograma. Se obtiene mediante *sustituciones*.

## 3.5 Criptografía simétrica

También conocidos como criptografía de clave secreta o clave privada. La clave utilizada en la operación de cifrado es la misma que se utilizada para el descifrado.

Existen dos mecanismos básicos: **de flujo** y **de bloque**.

Sea $E$ una función de cifrado.

- $E_k(b)$ cifrado del mensaje $b$ con la clave $k$.
- $m = b_1 b_2, \ldots$, donde cada $b_i$ es de longitud fija.

### 3.5.1 Cifrado de flujo

Se genera una clave de la misma longitud que el mensaje y se va cifrando cada parte del mensaje con esa clave. Si $k = k_1, k_2, \ldots$, entonces

$$E_k(m) = E_{k_1}(b_1), E_{k_2}(b_2), \ldots$$

Si la clave se repite (tiene un período) $k_1, k_2, \ldots$, el cifrador se dice **periódico** y la longitud de su periodo es su ciclo.

Utiliza los siguientes conceptos:

- El espacio de claves es mayor o igual que el espacio de los mensjes.
- Las claves son aleatorias.
- La secuencia de clave se usa sólo una vez.

La idea es que se usan generadores pseudoaleatorios con un algoritmo determinístico a partir de una semilla de $n$ bits, pudiendo generar secuencias con período de hasta $2^n$. Como el generador es determinístico, alcanza con transmitir la semilla.

La secuencia cifrante debe cumplir:

- Tener un período muy alto.
- Tener propiedades psuedoaleatorios.

Existen dos mecanismos de operación de las secuencias cifrantes:

- Sincrónicos: el emisor y el receptor deben sincornizarse previamente a la transmisión. La pérdida de 1 bit en el flujo de datos puede inutilizar el resto de la transmisión.
- Auto-sincrónicos: se utiliza parte de la información del texto cifrado para renovar la clase de la secuencia cifrante.

Problema de los algoritmos de cifrado de flujo: si un atacante obtiene el texto plano y el cifrado, es muy fácil obtener la clave.

### 3.5.2 Cifrado en bloque

El mensaje se divide en bloques de longitud fija y se aplica el algoritmo de cifrado a cada bloque en forma independiente con la misma clave.

$$E_k(m) = E_k(b_1)E_k(b_2)$$

Existen distintos modos de operación que dependen de cómo se mezcla la clave con el texto en claro:

- `EBC` (*Electronic CodeBook*): el texto se divide en bloques y cada bloque es cifrado en forma independiente utilizando la clave. Tiene el problema de que el mensaje puede quedar entendible (por ejemplo, en el caso de una imagen) porque mantiene mucha relación con el texto original.

  Problema: si se cuánto mide el bloque, me da mucha información, porque el mismo cypher es el mismo plain. Ventaja: si tengo un error en un bloque, el error no se propaga.

- `CBC` (*Cipher Block Chaining*): el texto se divide en bloques y cada bloque es mezclado con la cifra del bloque previo, luego es cifrado utilizando la clave.

  Ventaja: Por más que el plaintext sea el mismo, el cyphertext va a ser distinto.

- `CFB` (*Cipher Feedback*): opera como un cifrador de flujo auto-sincrónico, generando la clave de cifrado sobre la base de la clave y el bloque cifrado previo, y luego cifrando el mensaje con una operación `XOR`.

- `OFB` (*Output Feedback*): opera en forma similar al CFB, pero sin incluir el texto claro en el siguiente paso de realimentación. Es similar a un cifrador de flujo sincrónico.

## 3.6 Padding

Al dividir el texto original en bloques de longitud fija, algunos modos de cifrado requieren que se rellene el último bloque antes de realizar la operación. Este texto de relleno, llamado **padding**, debe ser quitado durante la operación de descifrado.

Para estandarizar el padding, se define estándar `PKCS#5`: el último bloque se completa con $N$ bytes con valor $N$, si es múltiplo se completa con un bloque completo de padding.

| Primer Bloque | | | | | | | | Segundo Bloque | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| A | A | A | A | A | A | A | A | B | B | B | B | B | B | B | 0x01 |
| A | A | A | A | A | A | A | A | B | B | B | B | B | B | 0x02 | 0x02 |
| ... | | | | | | | | ... | | | | | | | |
| A | A | A | A | A | A | A | A | B | 0x07 | 0x07 | 0x07 | 0x07 | 0x07 | 0x07 | 0x07 |
| A | A | A | A | A | A | A | A | 0x08 | 0x08 | 0x08 | 0x08 | 0x08 | 0x08 | 0x08 | 0x08 |

El problema del `PKCS#5` es que se sabe cómo es el padding. Ergo, se tiene una porción de texto plano y una porción de texto cifrado, lo que da información. Esto generó un ataque conocido como **Padding Oracle Attack**. No sacaba la clave pero lograba inyectar texto "coherente" encriptado.

## 3.7 Números aleatorios

La generación de números aleatorios es muy importante para la criptografía. Sin embargo, es muy difícil utilizar fuentes aleatorias verdaderas (como ruido físico). Es por esto que se suelen utilizar fuentes pseudo-aleatorias (algoritmos determinísticos que estadísticamente parecen aleatorios), inicializadas por una semilla.

### 3.7.1 LFSR: Linear Feedback Shift Register

Registros de desplazamiento con retroalimentación lineal. Un $n$-stage LFSR consiste en:

- Un registro de $n$ bits $r = r_0, \ldots, r_{n-1}$.
- Una secuencia de $n$ bits $t = t_0, \ldots, t_{n-1}$.

Ventajas:

- Muy rápido.
- Se conoce cómo construirlo para que el período sea máximo ($2^{n+1}$).

Desvenajas:

- La transformación es lineal. Si obtengo el mensaje en claro y el cifrado es fácil obtener la clave. Para disimular esta linealidad se puede tomar como input como parte del mensaje en claro.

### 3.7.2 NLFSR: Non Linear Feedback Shift Register

Es similar al `LFSR`, pero utiliza una función de reemplazo de bits más general.
En general no se los suele utilizar, sino que se usa *output feedback mode*.

Ventajas:

- Es más difícil adivinar la clave aún teniendo mensaje en claro y cifrado.

Desvenajas:

- No se conocen formas de garantizar que el período es máximo (y es difícil lograrlo).

## 3.8 RC4

Es un cifrador de flujo utilizado en `TLS/SSL` y `WEP`. Opera de modo sincrónico, a nivel de byte (no de bit). Es muy utilizado por su notable eficiencia en implementaciones de software.

`RC4` genera un flujo pseudoaleatorio de bytes (un *keystream*) que, para cifrarlo, se combina con el texto plano utilizando la función `XOR`. La fase de descrifrar se realiza del mismo modo.

Para generar el *keystream*, el algoritmo utiliza un estado interno secreto y dos funciones:

- Algoritmo de programación de claves (`KSA`).
- Algoritmo de generación pseudoaleatoria (`PRGA`).

## 3.9 DES: Data Encryption Standard

Es un cifrador por bloques que tiene las siguientes características:

- Utiliza bloques de 64 bits (8 bytes).
- La clave es de 7 bytes + 1 de paridad.
- Los bloques cifrados son de 64 bits (8 bytes).
- Su unidad básica es el bit.
- Utiliza *sustituciones* y *permutaciones* sobre los bits.

El algoritmo consiste en 16 iteraciones; en cada una utiliza una clave generada a partir de la clave suministrada originalmente. En cada iteración:

1. Se divide el bloque en dos mitades de 32.
2. Se pasa una de las mitades por una función $F$ de *Feistel*. Esta función consta de 4 pasos:
   - Expansión.
   - Mezcla.
   - Sustitución.
   - Permutaión.
3. Se mezclan ambas mitades con `XOR`.
4. Se rotan ambas mitades.

### 3.9.1 Ataques

DES tiene muchas propiedades indeseadas:

- Tiene 4 claves débiles (que son su propia inversa).
- Tiene 6 pares de claves semi-débiles
- Propiedad complementaria: $DES_k(m) = c \Rightarrow DES_{k'}(m') = c'$.
- Las S-Box (parte de la generación de claves) tienen propiedades irregulares (distribución de números no aleatoria y dependencias no deseadas).

Estas propieadedes permiten muchos ataques:

- Fuerza bruta ($2^{56}$ intentos).
- Criptoanálisis diferencial ($2^{47}$ intentos): usar pares de texto en claro elegidos y analizar cómo evolucionan a medida que se ejecutan las rondas. Se usa cuando no puedo elegir el texto.
- Criptoanálisis lineal ($2^{43}$ intentos): se usa cuando puedo elegir el texto.

### 3.9.2 Ventajas y Desventajas

Ventajas:

- Rápido.

Desvenajas:

- Débil.

### 3.9.3 Modos de operación

`DES` tiene distintos modos de operación:

- `ECB` (*Electronic Code Block*): se cifra cada bloque de forma independiente.
- `CBC` (*Cypher Block Chain*): `XOR` de cada bloque con el cifrado anterior.
- `EDE` (*Encrypt-Decrypt-Encrypt*): se usan dos claves ($k$ y $k'$), de tamaño 112 bits y se realiza la operación:

$$c = DES_k(DES_{k'}^{-1}(DES_k(m)))$$

- TripleDES (*Encrypt-Encrypt-Encrypt*): se usan tres claves ($k$, $k'$ y $k''$) de tamaño 168 y se realiza la operación:

$$c = DES_k(DES_{k'}(DES_{k''}(m)))$$

(problema: es lento)

## 3.10 AES: Advanced Encryption Standard

Es un cifrador de bloque y por producto. Opera con bloques y claves de longiud variable, que pueden ser especificadas independientemente a 128, 192 o 256 bits (las 9 combinaciones son posibles), siendo fácilmente extendible a múltiplos de 32 bits.

### 3.10.1 Ventajas y Desventajas

Ventajas:

- Implementación eficiente en software y hardware.
- Extensible.
- Seguro

Desvenajas:

- Distribución de clave.
- Complejidad en la gestión de la clave.

## 3.11 Criptografía asimétrica

Se le atribuye a Diffie-Hellman. Conceptualmente, la criptografía asimétrica toma como principio el tener dos claves: una **pública** (disponible para todos) y una **privada** (sólo para el individuo). La clave de cifrado y la de descifrado no son la misma.

El uso de criptografía asimétrica permite:

- **Confidencialidad**: cifrar usando la clave pública del destinatario, que lo descifra con su clave privada.
- **Integridad/Autenticación**: se cifra usando la clave privada del emisor, con lo que sólo se descifra con su clave pública.

Los sitemas de criptografía asimétrica deben cumplir las siguientes propiedades:

- Dada la clave apropiada debe ser computacionalmente fácil cifrar y descifrar un mensaje.
- Debe ser computacionalmente imposible derivar la clave privada a partir de la clave pública.
- Debe ser computacionalmente imposible determinar la clave privada a partir de un ataque de texto en claro elegido.

### 3.11.1 Diffie-Hellman

El algoritmo de Diffie-Hellman permite a dos sujetos acordar una clave de sesión en un medio inseguro, sin que otros sujetos que estén oyendo en el medio puedan saber cuál es. Se basa en operaciones matemáticas de exponenciación y en el problema de obtener el logaritmo discreto.

Su principal problema es que no provee autenticación: es vulnerable a un ataque *man in the middle*.

El algoritmo es el siguiente:

1. Alice y Bob eligen dos números $p$ y $g$ tales que:
   - $p$ es primo.
   - $2 \leq g \leq g - 1$
2. Alice elige un entero $a$ y le envía a Bob el resultado de $g^a \ mod \ p$.
3. Bob elige un entero $b$ y le envía a Bob el resultado de $g^b \ mod \ p$.
4. Alice calcula $k = (g^b \ mod \ p)^a \ mod \ p$.
5. Bob calcula $k' = (g^a \ mod \ p)^b \ mod \ p$.

Al final resulta que

$$k = k' = g^{ab} \ mod \ p$$

### 3.11.2 RSA

Es un algoritmo creado en 1977 por Ron **R**ivest, Adi **S**hamir y Len **A**dleman. Puede ser usado para cifrar y firmar mensajes. Se basa en el problema de la factorización de números muy grandes.

**Algoritmo**

Matemáticamente, involucra la función $\phi(n)$ que se define como el número de enteros positivos menores que $n$ que son coprimos[1] con $n$. Si $n = p \times q$, se puede demostrar que $\phi(n) = (p-1)(q-1)$. Usando esta propiedad, el algoritmo para obtener la clave pública y privada consiste en:

1. Elegir dos primos grandes distintos $p$ y $q$ ($p \neq q$). Consideremos $n = p \times q$.
2. Elegir $e < n$ tal que $e$ es coprimo con $\phi(n)$.
3. Calcular $d$ la solución de la ecuación: $(e \times d)\ mod\ \phi(n) = 1$.

Al final se obtienen:

**Clave pública**: $(e, n)$
**Clave privada**: $d$

Una vez obtenidas las claves, para cifrar y descifrar un mensaje $m$ se lo divide en bloques de longitud menor a $n$ y se calcula:

**Cifrar**:

$$c_i = m_i^e\ mod\ n$$

**Descifrar**:

$$m_i = c_i^e\ mod\ n$$

**Por un tema de costos, normalmente no se usa `RSA` para encriptar mensajes, sino que se encriptan claves para algoritmos simétricos**.

**Ataques**

La seguridad de `RSA` se basa en la complejidad de factorizar números muy grandes. Hoy en día si bien existen tests de primariedad polinomial (Rabin-Miller), no se conocen algoritmos polinomiales de factorización.

Los ataques principales a `RSA` se basan en:

- Texto cifrado elegido (firma de texto aleatorios).
- Los números tienen un módulo común (exponentes diferentes coprimos).
- Exponente de cifrado bajo.
- Exponente de descifrado bajo.

**Ventajas y Desventajas**

Ventajas:

- Provee autenticación y confidencialidad.
- Muy seguro.
- Claves grandes (2048 o 4096 bits).

Desvenajas:

- Muy lento.

[1] Un número es coprimo con $n$ si no tiene factores en común con $n$.

## 3.12 Checksum

Las funciones de **checksum** utilizaban para detectar errores no intencionales (por ejemplo, errores de transmisión por el medio). Algunas técnicas que se usan para checksums:

- Bit de paridad: es un bit que indica si la cantidad de bits en los 7 bits precedentes es par o impar.
- CRC: se basa en la idea de dividir polinomios y usar el resto.

## 3.13 Hash

Las funciones de **hash** transforman un mensaje de longitud variable en una cadena de longitud fija. Estas funciones deben cumplir:

1. No son reversibles.
2. Pueden tener colisiones.
3. Son rápidas de calcular.
4. Resistencia a preimágenes: dado un hash $z$, no es factible encontrar un documento de entrada $x$ tal que $f(x) = z$.
5. Resistencia a segundas preimágenes: dado un documento $x$ no es factible encontrar un $x'$ tal que $f(x) = f(x')$.
6. Resistencia a colisiones: no es factible encontrar dos documentos $x$ y $x'$ tal que $f(x) = f(x')$

Si se una función cumple (4) y (5) se la llama **de una vía**. Si cumple (5) y (6) se dice que es **resistente a colisiones**.

Estas funciones se construyen mediante Merkle-Damgard: `SHA-1`, `SHA-2`, `MD5`, etc.

Una de las cosas que dificulta en análisis de las funciones de `hash` es el **efecto avalancha**: un cambio de 1 bit en un mensaje cambia al rededor del 50% de los bits.

### 3.13.1 Buenos y malos usos

Lo correcto:

- Utilizar un hash cuando se puede distribuir de manera segura $H(x)$ y se desea verificar que un valor $x'$, recibido de manera insegura, es de hecho igual a $x$.

Lo malo:

- Distribuir $H(x)$ y $x$ por el mismo medio: si un atacante modifica $x$ también puede modificar $H(x)$.
- Utilizar $H(x)$ como una firma: cualquiera puede calcular $H(x)$.

### 3.13.2 Length Extension Attack

El **length extension attack** es un ataque que permite a un atacante inyectar texto arbitrario a un hash con secreto aún sin conocer el secreto. Se basa en la propiedad de que las funciones vulerables (las que utilizan la construcción de Merkle-Damgard), cumplen la propiedad de:

$$H(x + a) = H(x) \text{ unido con } H(a)$$

Luego, si queremos transmitir un mensaje con hash con secreto y hacemos $H(secreto+msj)$, un atacante que intercepte el tráfico puede inyectar texto en el documento y obtener la firma aún sin conocer el secreto directamente. Para evitar esto, se puede:

- Hacer hash doble: $H(H(secreto + msj)$.
- Dejar el secreto al final: $H(msj + secreto)$ *(Obs: esto tiene otro tipo de ataques)*
- Usar `HMAC`.

## 3.14 HMAC

Las funciones más comunes (`SHA-1`, `MD5`, etc), no fueron diseñadas para la autenticación, puesto que carecen de clave secreta (o, si se utiliza incorrectamente, pueden ser fácilmente atacados).

La diferencia principal entre una función de hash y un `MAC` es que conocer $MAC_k(x)$ no permite computer $MAC_k(y)$ para algún otro $y$.

Los `HMAC` se calculan como:

$$HMAC_k(m) = h((k \oplus opad)||h((k \oplus ipad)||m))$$

donde:

- $opad = 0x5c5c5c\ldots$
- $ipad = 0x363636\ldots$

## 3.15 Certificado

Un **certificado** es una estructura de datos que contiene:

- La identidad del poseedor de la clave pública.
- La clave pública.
- Emisor del certificado.
- La fecha en que se emitió.
- Información adicional (ej. identidad del emisor, uso que se le puede dar a ese certificado).

Supongamos que *Alice* y *Bob* quieren compartir información. Para verificar su origen, ambos tienen sus pares de claves pública/privada. El problema que surge es: ¿cómo se pasan sus respectivas claves públicas? Una opción es utilizar un tercero confiable (*Cathy*). Sin embargo simplemente pateamos el problema para adelante. ¿Cómo obtienen la clave pública de *Cathy*?

Para solucionar este problema se plantean dos soluciones de cadenas de firmas:

- X.509
- PGP

### 3.15.1 PGP

La gestión de claves en `PGP` se basa en la confianza mutua y es adecuada solamente para entornos privados o intranet.

Usa el mecanismo de clave pública y privada, pero resuelve el mecanismo de confianza en base a confianza mutua: no tengo una autoridad centralizada que certifica, sino que a medida que uno se pone en contacto con más gente que usa PGP, más "certifica" la clave.

Los datos asociados a las claves `PGP` son:

- Versión de PGP.
- Clave pública junto con el algoritmo (`RSA`, `DSA`, `DH`).

- Información sobre la identidad del titular.
- Firma digital del titular del certificado (auto-firma).
- Período de validez.
- Algoritmo simétrico de cifrado preferido.
- Conjunto de firmas de terceros: (opcional)
  - Definen nivel de confianza.
  - Definen nivel de validez.

### 3.15.2 X.509

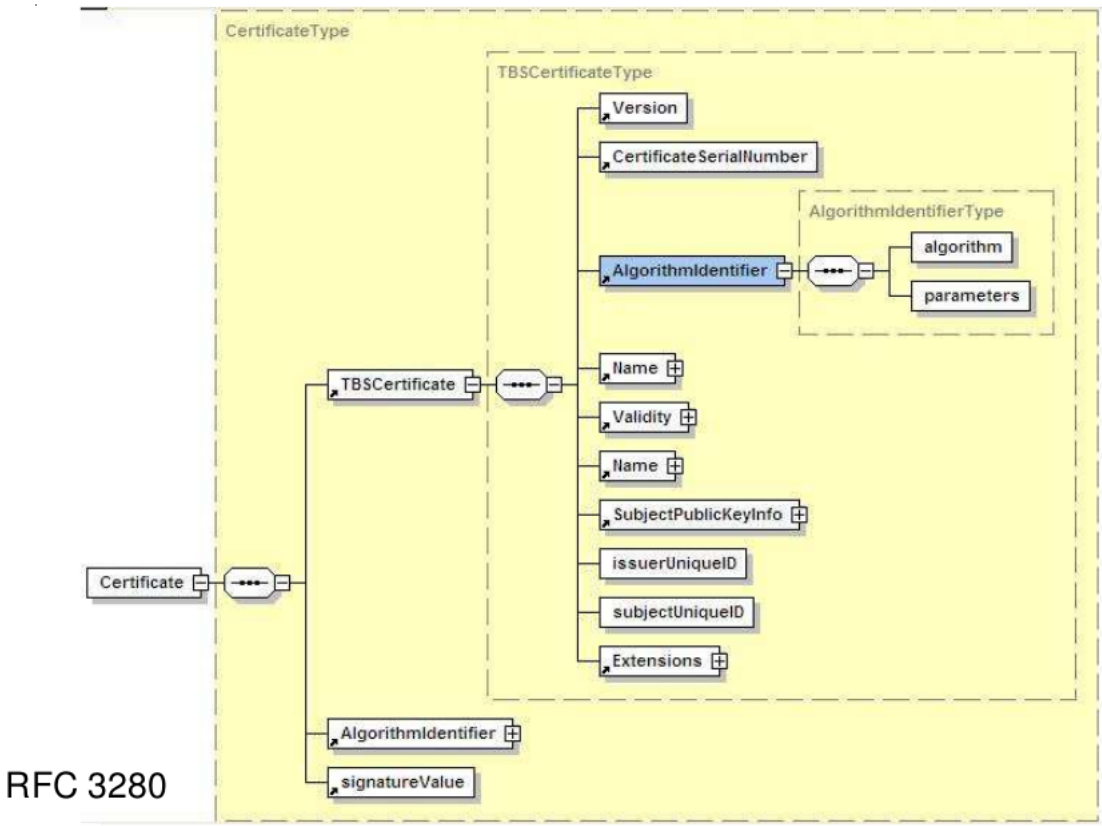
Los certificados **X.509** tienen el problema de centralización del certificado. Para solucionarlo, se usa el concepto de **autoridades certificantes**: terceras partes confiables que dan fé de la verdadi de la información inluida en los certificados que emiten. Sin embargo, tener entidades certificantes centralizadas tiene muchos problemas:

- Muchos puntos de ataque.
- Una entidad certificante de Malasia puede emitir certificados para Argentina. O para Google.

Los datos asociados son:

- Versión de X.509 (v3).
- Número de serie.
- Algoritmo de firma (`SHA-1` with `RSA`).
- Nombre del emisor.
- Período de validez.
- Nombre del titular / suscriptor.
- Clave pública del titular.
- Firma: hash del certificado cifrado.
- Extensiones (opcional).

Cada certificado puede identificarse unívocamente con su número de seríe + su emisor.
Las extensiones pueden usarse para cosas como manejar la herencia de certificación, restringir el uso del certificado o aportar mayores preciosones al uso del certificado.




**Validación**

El proceso de validación de un certificado `X.509` consiste en:

1. Obtener la clave pública del emisor.
2. Descifrar la firma para obtener el hash del certificado.
3. Recalcular el hash del certificado y compararlo con el obtenido.
4. Chequear el período de validez del certificado.
5. Chequear `CRL`s (*Certificate Revocate Lists*).

### 3.15.3 FIPS140

Es un estándar para evaluar números criptográficos. Define criterios, 4 niveles y 11 categorías con requerimientos específicos:

**Nivel 1**

- Verifica que los algoritmos de cifrado están aprobados por una oficna de EEUU.
- Verifica que los algoritmos de cifrado estén bien implementados.
- No especifica seguridad física.
- Permite que los componentes de software o firmware se ejecuten en un sistema de propósito general utilizando un sistema operativo no evaluado.

**Nivel 2**

- Especifica seguridad física: Sellos o revestimientos "tamper-proof" en las cubiertas removibles del módulo.
- Requiere autenticación basada en roles.
- Los componentes de software y firmware deben ejecutarse en un sistema operativo que haya sido evaluado en *Common Criteria EAL2* o superior.

**Nivel 3**

- Más restricciones físicas: suficiente para prevenir que los intrusos accedan a los parámetros críticos de seguridad del módulo criptográfico.
- Autenticación basada en identidad.
- Fuertes requerimientos para leer y alterar los parámetros críticos de seguridad.
- Los componentes de software y firmware deben ejecutarse en un sistema operativo que haya sido evaluado en *EAL3*.

**Nivel 4**

- Más restricciones físicas: si se abre un módulo crítico, debe destruirse la información que contiene. Cambios en las condiciones de temperatura / presión / movimiento debe destruir la información.
- Los componentes de software y firmware deben cumplir los requerimientos funcionales del nivel de seguridad 3 y deben ejecutarse en un sistema operativo que haya sido evaluado en *EAL4*.

### 3.15.4 Revocación de certificados

Por diversos motivos, un certificado puede ser revocado antes de su fecha de expiración:

- Se comprometió su clave.
- Cambio de la situación del titular.

La revocación de certificados plantea muchos problemas:

- La entidad que revoca el certificado debe estar autorizada a hacerlo.
- La información de revocación debe estar disponible rápidamente.

**CRL**

Para implementar la revocación de certificados se utilizan las `CRL`: (*Certificate Revocation List*). Es una lista de los certificados que se encuentran revocados. Son el equivalente a las listas de tarjetas de crédito robadas. Para evitar fraudes, sólo el emisor del certificado puede revocar el mismo. Las autoridades certificantes están obligadas a publicar permanentemente la `CRL`, que tiene un período de validez.

Las `CRL`s consisten de los siguientes campos:

- Version.
- Algoritmo de firma.
- Nombre del emisor.
- Fecha de emisión.
- Fecha de próxima emisión.

- Lista de certificados revocados:
  - Nro de serie del certificado revocado.
  - Fecha de revocación.
  - Extensiones de revocación (motivo de revocación)
- Extensiones de la `CRL`.

Los principales problemas de la `CRL` es que no contienen el estado actual: es muy posible que estén desactualizadas. Además, la responsabilidad de verificarlos recae en el usuario. Además, tienen muchos problemas de volúmen y distribución.

Las soluciones planteadas involucran:

- Dividir el alcance (*scope*) de una `CRL` para reducir la cantidad de certificados incluidos.
- Emitir *delta* `CRL` sólo con los nuevos certificados revocados.
- `CRL`s indirectas.

Además, se implementan técnicas como **OSCP**: un servicio que permite saber si un certificado es válido o no. Hace una conslulta a la autoridad certificante, que responde con un mensaje firmado que puede ser: *good*, *revoked* o *unknown*.

`OSCP` tiene varios problemas:

- ¿Cómo sé que una autoridad certificante es confiable?
- X.509 usa los strings como Pascal (`ASN1`): longitud fija, con la longitud adelante de todo. Pero la mayoría de las aplicaciones que verifican están hechas en `C`. Luego, pudeo pedir un certificado para algo como "facebook.com
  0midominio.com". Si está mal implementado, la autoridad lo va a emitir para eso, pero el navegador puede leerlo como para "facebook.com".
- Cuando una autoridad certificante responde por `OSCP` con *unknown*, esa respuesta no está firmada. Si alguien está haciendo *man in the middle*, puede mandar constantemente eso para evitar la validación de un certificado.

Las claves `PGP` también pueden ser revocadas mediante un menasje con un flag especial. Puede ser revocada por su firmante o, si lo permitió el dueño, que la revoque un tercero.

### 3.15.5 Tiempo de vida de los certificados

Según su tiempo de vida, los certificados pueden ser clasificados en dos cosas:

**Corto plazo**

- Se generan de manera automática.
- Se utilizan para un mensaje o una sesión y luego se descartan.

**Largo plazo**

- Son generadas por el usuario de manera explícita.
- Se utilizan para autenticación y confidencialidad.

### 3.15.6 A tener en cuenta en manejo de certificados

- Como se generan las claves.
- Como se asocia una clave a la identidad de su poseedor.
- Como se distribuyen las claves.
- Como dos partes establecen una clave común.
- Como se almacenan las claves de manera segura.
- Que ocurre cuando se compromete una clave.
- Como se destruyen las claves.

### 3.15.7 PKCS: Public Key Cryptography Standards

Son un conjunto de estándares y especificaciones técnicas cuyo objeto es uniformizar las técnicas y protocolos de criptografía pública.

Algunas cosas de las que predica `PKCS` son:

- Cómo se manejan estándares cifrados.
- Cómo se hace un backup de claves.
- Cómo se interactúa con un token criptográfico externo.

### 3.15.8 PKI: Public Key Interface

Es una combinación de hardware y software, políticas y procedimientos que permiten asegurar la identidad de los participantes en un intercambio de datos usando criptografía de clave pública. No sólo hacen referencia a la parte teórica sino también a:

- **Autoridades certificantes**: tercero confiable que da fé de la veracidad de la información incluida en los certificados que emiten. Emiten certificados digitales según su política de certificación (`CP`) (reglas que indican la aplicabilidad de un certificado digital a una comunidad y/o a una clase de aplicaciones con requerimientos de seguridad en común).

  Las autoridades certificantes tienen un manual de procedimientos de certificación (`CPS`), que consiste en una declaración de las prácticas empleadas para emitir, administrar y revocar certificados.

- **Autoridades de regirstro**: verifica la propiedad del alguna manera y le dice a la autoridad certificante que puede (o no) emitir el certificado. Por ejemplo, para emitir un certificado para una `URL` podría decirse "tal dia a tal hora poné tal contenido en la página".

  Además, pueden iniciar revocaciones de certificados, autorizarlos o negar su creación y revocación, etc.

- **Tercer usuario**: el receptor de un certificado que actúa basados en el mismo y/o en cualquier firma digital que se verifique con ese certificado.

- **Suscriptores**: sujeto que solicita la emisión de un certificado.

- **Repositorios**: estructuras encargadas de almacenar la información relativa a la `PKI`. Las más importantes son:

**Emisión de certificados**

Los pasos para la emisión de un certificado son:

1. El suscriptor genera un par de claves. Firma la clave pública y la información que lo identifica con su clave privada. Luego envía todo a la autoridad certificante. *Esto prueba que posee la privada correspodiente y protege la información.*
2. La autoridad certificante verifica la firma del suscriptor en los datos recibidos. Opcionalmente se puede verificar la información por otros medios, tales como presencia física, correo electrónico, etc. En este paso interviene la autoridad de registro.
3. La autoridad certificante firma la clave pública y parte de la información que el suscriptor envió con su clave privada y crea el certificado. *De esta manera se asocia el suscriptor con su clave pública y sus datos.*
4. El suscriptor recibe el certificado y verifica la firma de la autoridad certificante (mediante su clave pública) y los datos del certificado. *De esta manera se asegura que la autoridad certificante no cambió sus datos y se protege la información del certificado.*
5. La autoridad certificante publica el certificado.

### 3.15.9 Modelos de confianza

El modelo de autoridad certificante plantea el dilema de la confianza: ¿cómo se determina en qué certificados se puede confiar? ¿cómo se establece la confianza? ¿bajo qué circunstancias fluctúa esa confianza?.

Se plantean muchos modelos de confianza:

- Jerárquico: hay una única autoridad certificante que emite certificados para autoridades certificantes intermedias.
- Modelo Web: tengo una lista interminable de autoridades certificantes en las que confío. Cualquiera puede emitir para cualquiera.
- Bridge CA: hay una autoridad certificante que hace de puente para otras. otras AC.
- Certificación cruzada.
- Reconocimiento cruzado.
- CTL (lista de certificados confiables).

### 3.15.10 Tipos de certificados

Existen muchos tipos de certificados distintos:

- SSL.
- S/MIME (mail).
- S/MIME (personales).
- Firma de código.
- Autoridad certificante.
- WPA-PSK.
- VPN.

## 3.16 Firma digital

La **firma digital** es un conjunto de datos expresados en formato digital que se utiliza para **identificar a un firmante**, **verificar la integridad del contenido de un documento digital** y **garantizar el no repudio del firmante**.

La información debe reunir las siguientes condiciones:

- Autoría.
- Integridad.
- Confidencialidad.
- Disponibilidad.

## 3.17 Codificaciones

### 3.17.1 Base64

**Base64** es un mecanismo de codificación que utiliza un conjunto de 64 caracteres para codificar cualquier valor posible de un byte. Toma 3 bytes y los convierte en 4. Usa letras mayúsculas y minúsculas, números, + y /. Para el padding usa =.

### 3.17.2 MIME

*Multipurpose Internet Mail Extensions* (`MIME`) es un estándar de internet que extiende el format ode los mails para soportar texto que no sea `US-ASCII`, binarios anexados, etc.

Soporta distintos tipos de contenido (definido en el *content-type*), tales como texto plano o richtext, imágenes, video, PostScript, multipart, etc.

### 3.17.3 SMIME

*Secure MIME* (`SMIME`) es un estándar para cifrado de clave pública y firma de mails. Provee servicios de:

- Autoría.
- Integridad de mensaje.
- No repudio.
- Confidencialidad de los datos.

### 3.17.4 ASN.1

`ANS.1` es una norma para representar datos que establece una sintaxis abstracta para la definición de estructuras independientemente de la arquitectura de hardware o lenguaje de implementación. Es utilizado en la definición de estructuras de datos para intercambio de aplicaciones.

### 3.17.5 OID

Un **object identifier** (`OID`) es un código de identificación único de un objeto o estructura que forma parte de una estructura jerárquica. Existe un registro internacional de `OID`s. Se utilizan para la identificación de

- Atributos.
- Extensiones.

- Algoritmos.
- Políticas de certificación.
- Estructuras de datos.

## 3.18 OpenSSL

**OpenSSl** es una implementación open source de diversos algoritmos y estándares criptográficos. Se puede utilizar para:

- Crear y ver certificados.
- Generar números aleatorios.
- Cifrados de varios tipos.
- Firmar mails.

# 4 Unidad 4

## 4.1 Introducción

- Definiciones:
  - Autenticación: Asociación entre una identidad y un objeto.
  - Maneras de establecer identidad:
    * Que es lo que la entidad sabe.
    * Que es lo que la entidad tiene.
    * Que es lo que la entidad es físicamente (e.g., huellas digitales).
    * Donde esta la entidad.
  - Sistema de Autenticación: Tupla $(A, C, F, L, S)$ con:
    * $A$ es la información que prueba la identidad.
    * $C$ información almacenada digitalmente para validar la identidad.
    * $F$ función que transforma $A$ en $C$.
    * $L$ función que prueba la identidad.
    * $S$ un conjunto de funciones que permiten crear o alterar información en $A$ o $C$.
  - Ejemplo para sistema de almacenamiento de claves en texto plano:
    * $A$: conjunto de strings que forman claves.
    * $C$: $A$
    * $F$: función identidad.
    * $L$: función de comparación de strings.
    * $S$: funciones para editar el archivo con las claves.
  - Claves:
    * Secuencias de caracteres o palabras.
    * Algoritmos como one-time passwords o challenge response.
  - Almacenamiento de claves
    * Texto transparente: Inseguro, si se compromete el archivo se comprometen todas las claves.
    * Archivo cifrado: Necesita claves de cifrado y descifrado, terminamos en problema anterior.
    * One way hash de la clave: Si se compromete el archivo, el atacante debe invertir la función de one way hash, o adivinar las claves.
  - Salt: comprende bits aleatorios que son usados como una de las entradas en una función derivadora de claves. La otra entrada es habitualmente una contraseña. La salida de la función derivadora de claves se almacena como la versión cifrada de la contraseña. La sal puede también ser usada como parte de una clave en un cifrado u otro algoritmo criptográfico. La función de derivación de claves generalmente usa una función hash.

## 4.2 Sistemas de autentificación

### 4.2.1 UNIX Passwords

- Usa hash de la clave, usando una función de hash dentro de 4096 para pasarlo a un string de 11 caracteres.
- Sistema:
  - $A$: Strings de 8 o menos caracteres.

  - $C$: 2 letras de id de hash concatenadas a 11 letras del hash en si.
  - $F$: 4096 versiones de DES modificadas.
  - $L$: login, su
  - $S$: passwd, nispasswd, passwd+

- Algoritmo
  - Cifra un bloque de 64 bits en 0 con DES y el password como clave.
  - Repite esto 25 veces usando como clave en cada paso el resultado de la anterior.
  - Se selecciona la variante de DES usando un SALT con la hora del día. Esta se almacena sin cifrar junto con el password.

#### 4.2.2 Ataques

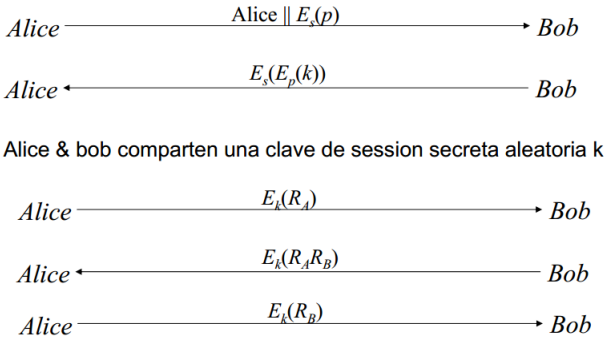
- Ataque: Tiene como objetivo encontrar $f \in F$, $f(a) = c \in C$ y que $c$ este asociado a una entidad.
  - Se puede lograr directamente o indirectamente (mediante un $l(a)$ tal que $f(l(a)) = c$).
- Prevencion:
  - Ocultar la mayor cantidad de variables posible.
  - Bloquear acceso a $l \in L$ o al resultado de $l(a)$, por ejemplo anulando login desde la red.
- Ataques posibles
  - Diccionario: Consiste en adivinar probando todas las palabras de una lista. Se puede hacer offline si conocemos $f$ y $c$ y probamos multiples $a$ hasta que coincida. Sino, hacemos intentos de login (mas lento).
  - Fuerza bruta: Probar todas las combinaciones posibles (menos eficiente que diccionario, necesito conocer el espacio de busqueda).
  - Choques: Si no se utiliza SALT, a cada clave le corresponde un unico hash. Es fácil encontrar que dos usuarios usan la misma clave si tengo la base de datos y el espacio de búsqueda se reduce significativamente. Puedo además precomputar hashes y buscarlos en la base de datos de passwords $c$.
  - Formula de Anderson:

$$P \geq \frac{T \cdot G}{N}$$

    * $P$ la probabilidad de obtener una clave en un período de tiempo.
    * $G$ el número de intentos posibles por unidad de tiempo.
    * $T$ la cantidad de unidades de tiempo.
    * $N$ el número de claves posibles.
  - Logins repetidos: Intentar darle al sistema desde afuera. Se puede enlentecer cortando despues de una cierta cantidad de logins, deshabilitando la cuenta o creando un ambiente *sandboxeado* para el agresor.

## 4.3 Claves

- Maneras de elegirlas:
  - Aleatorias: Las elige el sistema. Tienen casos borde (claves muy cortas, caracteres repetidos) si no se tiene cuidado. Son difíciles de recordar y su calidad depende de la implementación del generador de números aleatorio.
  - Pronunciables: Basadas en generación de fonemas, lo que permite que sean fáciles de pronunciar y por ello de recordar. Son pocas.
  - Elegidas por el usuario: Suelen ser "faciles", en el sentido que hay patrones establecidos. Se puede hacer chequeo activo de claves para evitar que el usuario utilice claves muy sencillas.
    - `passwd+` provee un lenguaje de scripting para definir estas restricciones.
    - Se puede forzar el usuario a cambiar a un password nuevo cada cierto tiempo. Avisar con anticipación para que el usuario pueda pensar.
  - Challenge response: El usuario y el sistema comparten una función secreta $f$, que puede ser una criptográfica basada en alguna clave compartida.
    - El usuario pide ser autenticado.
    - El sistema le contesta un valor $r$ aleatorio que es el "desafío".
    - El usuario contesta $f(r)$, la "respuesta".
    - Se puede atacar de manera similar a las claves, si dispongo de $f$,$r$ y $f(r)$.
    - Se puede evitar cifrando el challenge $r$.
  - Protocolo EKE: Asume que Alice y Bob tienen una clave $s$ que comparten. Generan una clave $k$ de sesión.




  - One Time Passwords: Se invalidan una vez que son usados. Se envía un número y la respuesta es la clave asociada a ese numero. Se dificulta sincronizar las claves, es dificil generar buenas claves aleatorias y distribuir las claves entre los clientes y servidores como para poder hacer esto andar.
  - One Time Passwords con Hardware:
    - Tokens: Utilizado para calcular la respuesta a un challenge. Puede requerir de un PIN de usuario.
    - Tiempo: Cada lapso de tiempo muestra un número. El usuario usa ese número junto con una clave fija adicional.
  - Biometría: Medición automática de características biológicas o de comportamiento de un individuo.
    - Huellas digitales, mapeado a un grafo y comparado de manera aproximada con una base de datos.

    - Voces: Verificación y reconocimiento.
    - Ojos: Patrón único del iris, muy muy intrusivo.
    - Cara: Detección de patrones como largo del labio, ángulo de cejas, etc.
    - Secuencias de tipeo (presión sobre las teclas, cadencia, etc.)
  - Localización: Sabiendo donde esta el usuario, validar si esta ahi (usa GPS).

#### 4.3.1 PAM

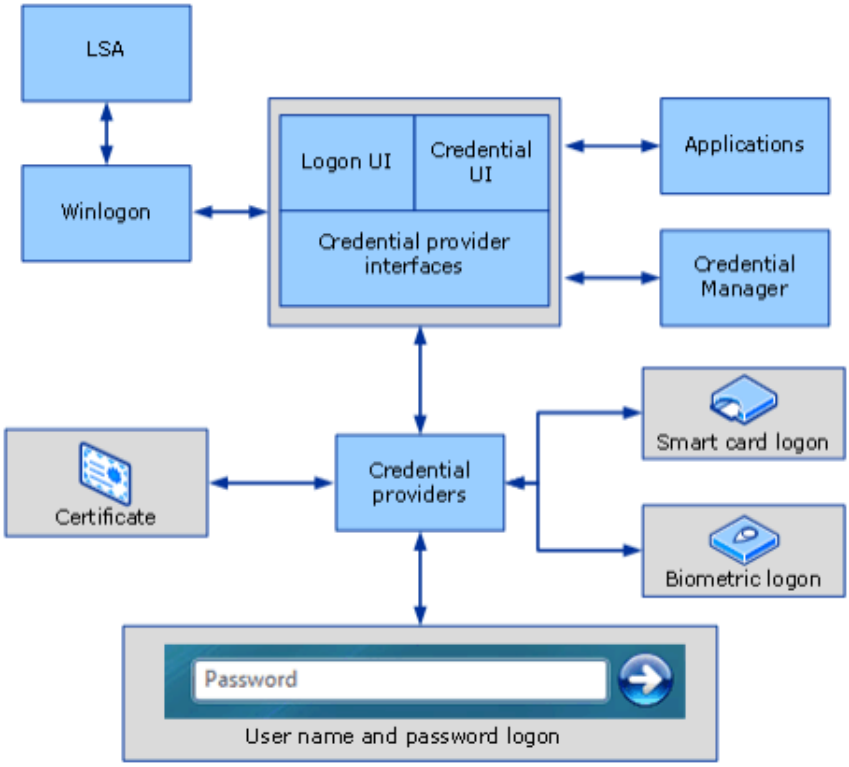
- *Pluggable Authentication Modules*: Mecanismo flexible para la autenticación de usuarios. PAM permite el desarrollo de programas independientes del mecanismo de autenticación a usar. Así es posible que se pueda usar desde autenticación por password básica hasta mecanismos físicos o de localización.
- Permite distintas políticas de autenticación para cada servicio.
- Es configurable, utiliza un repositorio de métodos a usar con un archivo del mismo nombre que el servicio a usar en `/etc/pam.d/`.
- Módulos independientes efectuan el chequeo:
  - Sufficient: Acepta si el módulo acepta.
  - Required: Falla si el módulo falla, pero ejecuta todos los requisitos antes de reportar falla.
  - Requisite: Igual a required pero no falla en el resto.
  - Optional: Solo es invocado si todos fallan.

#### 4.3.2 GINA

- Graphical Identification and Authentication Library
- Componente de Windows que se carga en el contexto de WinLogon.
- Atiende Ctrl+Alt+Del e interactua con el usuario. Luego arranca el proceso inicial de usuario (shell).
- Puede ser reemplazado, hay implementaciones con biometría o tokens en hardware.
- Servía para Windows XP/2003

#### 4.3.3 MS Credentials Providers

- Nuevo mecanismo a partir de Windows Vista.




## 4.4 Almacenamiento de claves

- Lan Manager Hash:
  - Se convierte todo a mayúsculas antes de crear el hash.
  - Se usa un set de characteres reducido dentro de ASCII.
  - El hash se divide en dos bloques de 7 caractéres. Si la clave tiene menos de 14 caractéres se paddea con null.
  - Utiliza DES con el password "KGS!@#$%".
  - No usa SALT y el hash es un valor de 16 bytes.
  - Muy inseguro, se puede romper fácilmente.
- NT Hash:
  - Distingue entre mayúsculas y minúsculas.
  - Usa MD4 (inseguro).
  - Longitud variable (no se paddea) hasta 128 caractéres.
  - No se usa salting.
  - NT4 lo soporta desde Service Pack 4.
- Linux - MD5:
  - Contraseña se almacena en un archivo solo accesible por root (`/etc/shadow`) y la clave se cifra usando tres campos separados por $:

    - El algoritmo de *one way hashing* usado (1 es MD5).
    - El SALT usado.
    - El Hash propiamente dicho.
  - Se pueden usar otros algoritmos como Blowfish o SHA-256 o SHA-512.
  - Se puede configurar el número de repeticiones.

- OpenBSD: Implementa un mecanismo por el cual el tiempo computacional necesario para cifrar una password con una función de una vía puede ir variando en el tiempo, a medida que avanza la velocidad del hardware. Las claves de usuarios con mayores privilegios pueden ser configuradas para ser más difíciles de calcular.

### 4.4.1 Cracking - Herramientas

- John The Ripper: Permite crackear claves MD5, Crypt, Blowfish, LM, etc. Usando diccionarios, reglas o fuerza bruta.
- Rainbow Tables: Precomputar los pares (hash, texto en claro) y usarlos para buscar de manera rápida hashes.

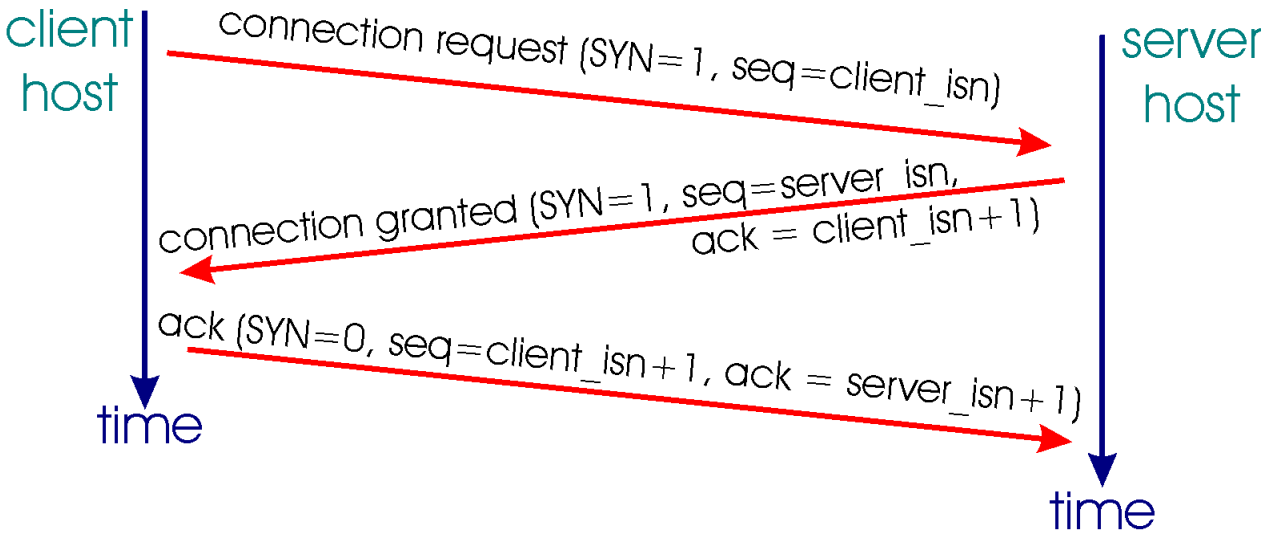
# 5 Unidad 5: Seguridad en Redes

## 5.1 Introducción

Definiciones:

- **TCP**: *Transmission Control Protocol*, protocolo orientado a la conexión. Provee control de flujo, recuperación de errores y confiabilidad.
- **UDP**: *User Datagram Protocol*, protocolo orientado a la conexión. Muy sencillo, no provee garantías. La recuperación de errores es responsabilidad de la aplicación.
- **ICMP**: *Internet Control Message Protocol*. Es usado para mensajes de control, mensajes de error, etc. El Ping utiliza ICMP. Algunos tipos de ICMP:
  - Echo Request
  - Echo Reply
  - Destination Unreachable
  - Time Exceeded
  - Timestamp
  - Timestamp Reply
  - Redirect Message

### 5.1.1 Three Way Handshake




### 5.1.2 Sniffers

- Un *Sniffer* es un programa de software o hardware que puede "ver" y registrar el tráfico que pasa sobre una red digital. Mientras el flujo de datos viaja por la red, el sniffer captura cada paquete y opcionalmente lo decodifica en base a reglas, estándares y especificaciones.
- Se puede sniffear todo el tráfico que pasa por una red o solo una parte.
- Opera en "modo promiscuo" porque escucha todo lo que pasa en el medio, incluso lo que no va para él.

- Usos:
  - Analizar problemas de red.
  - Detectar intentos de intrusión a través de la red.
  - Obtener información para luego hacer una intrusión.
  - Monitorear el uso de la red.
  - Reportar estadísticas de red.
  - Espiar a otros usuarios de la red.
  - Hacer ingeniería reversa de protocolos.
- Ejemplo: TCPDump, Wireshark, etc.

### 5.1.3 Monitoreo

- La mayoría de las redes *switcheadas* puede definir un puerto de monitoreo, mecanismo conocido como *port mirroring*. En este puerto se copia todo el tráfico. Sino se puede utilizar un *network tap* insertado en el segmento de red para que transmita todo el tráfico.
- Se puede utilizar una estación de trabajo con dos placas de red, para armar un *bridge* transparente. Para administrar este equipo que esta en el medio del tráfico, se puede usar una tercera placa con IP para accederla por afuera.

### 5.1.4 ARP spoofing

- Definición: ARP (*Address Resolution Protocol*) es el protocolo responsable de encontrar la dirección de hardware (MAC) que corresponde a una determinada IP.
- Funcionamiento: Se envía un paquete del tipo ARP Request a la dirección de Broadcast conteniendo la dirección IP a la que se quiere contactar. Se espera un ARP Reply con la dirección MAC correspondiente.
- ARP Table: Cada equipo tiene una tabla temporal como *cache* de resultados obtenidos.
- ARP Spoofing: Consiste en, con el propósito de hacerse pasar por otra una maquina con otra IP (por ejemplo el *default gateway*), enviar paquetes ARP Reply *spoofeados* a la red local para que quede asociado a esa IP.

### 5.1.5 IP spoofing

- IP spoofing consiste en crear paquetes IP con una IP de origen distinta a la IP del que envía.
- El receptor confía en que la IP del paquete es la del emisor.
- Sirve diversos propósitos:
  - Esconder el origen de un ataque.
  - Secuestrar una sesión abierta.
  - Aprovecharse de aplicaciones que autentifican con la IP de origen.

### 5.1.6 Ataques de Denegación de Servicio

- Definición: Un ataque de Denegación de Servicio (DoS) consiste en evitar que un sistema pueda ser usado por usuarios legitimos, generalmente mediante la sobrecarga de pedidos tal que no hay recursos suficientes para los pedidos legitimos.
- Ataque de SYN Flooding:
  - Cada paquete con el *flag* SYN prendido crea una nueva conexión *half-open*.
  - Estas conexiones tardan minutos en *timeout*ear y desaparecer.
  - Las tablas de conexiones son finitas, con lo cual se pueden sobrecargar.
  - Se puede *spoofear* la IP.
  - Se pueden utilizar las SYN Cookies para evitar esto.
    - Las SYN Cookies son para establecer el número inicial de secuencia TCP de manera de poder saber que MSS esta usando la conexión y poder reconstruir esa cola SYN en su momento.
    - Es decir, se guarda el estado en el número de secuencia y no en el servidor (disminuye la carga).
    - Se calcula en base de las IPS, puertos, etc.
- Reset de Conexiones:
  - Un host envia un paquete RST con una IP *spoofeada* a cualquier extremo de la conexión.
  - Necesito saber las IPs y puertos de los extremos de la conexión.
  - Necesito tener un numero de secuencia dentro de la ventana que se esta enviando. Lo puedo obtener escuchando el tráfico y adivinando con los bits.
- Ataque de ICMP:
  - Fabricar un paquete ICMP que indique un error "hard"
    - Protocol Unreachable
    - Port unreachable
    - Fragmentation Needed y DF set.
  - Segun el RFC 1122, TCP debería abortar la conexión.
  - No se recomienda hacer chequeo de los errores por lo tanto no hay que adivinar números de secuencia.

### 5.1.7 Comandos varios

- Los Protocolos r (*rcp*, *rlogin*, *rsh*, *rwho*).
- Permitir el acceso a terminales remotos sin tener que loguearse con usuario y clave.
- Los archivos `/etc/hosts.equiv` y `.rhosts` proveen el mecanismo de autenticación.
  - Especifican que equipos remotos son confiables.
  - Los usuarios confiables pueden acceder sin password.
  - `/etc/hosts.equiv` vale para todo el sistema, cada usuario tiene su `.rhosts`
- Ident: Protocolo para identificar al usuario remoto de una conexión TCP determinada.
  - Permite especificar dos puertos (uno X local a la maquina A y otro remoto Y en la maquina B) para que B devuelva el usuario que esta conectado desde el puerto Y de B al X de A.

- Telnet
  - Login remoto.
  - Tráfico en claro.
  - User y password para loguearse.
  - Suceptible a *man in the middle*, *session hijacking*, etc.
- SSH
  - Login remoto, transferencias de archivos, y *tunneling* de conexiones.
  - Cifra todo el tráfico para eliminar las "escuchas".
  - Mecanismo de user/pass o mecanismo de pares de claves
  - Protege MitM, Session Hijacking, Spoofing, Sniffing, Tampering, etc.
- TFTP:
  - Protocolo sin autenticación para transferencia de archivos.
  - Se puede usar para bootear discos a través de una red.
- HTTP (80):
  - Protocolo de aplicación para sistemas de información hipermediales, distribuidos y colaborativos.
  - Protocolo simple, *stateless*, basado en texto.
  - Cliente envía *requests* al servidor, con un método (GET,POST,etc.), URI, headers y un cuerpo opcional.
  -
- HTTP Authentication
  - Cliente hace un *request*
  - Servidor responde con código 401 y pide autenticación.
  - Cliente envía la request, y además sus credenciales encodeadas en base 64 (no provee confidencialidad)
  - Servidor valida las credenciales y contesta con el pedido.
- HTTPS (443)
  - Comunicación encriptada mediante SSL.
  - Permite identificar fehacientemente al servidor y da confidencialidad.
  -
- SMTP, POP3
  - Se pueden usar con SSL
  - Problema de *Open Relay*: Un servidor configurado para enviar mail de cualquier dirección y a cualquier dirección. Solía ser la configuración por *default*. Es usado por *spammers* y usualmente esta blacklisteada.
- DNS
  - *Domain Name System* como esquema jerárquico de resolución de nombres.
  - Hay servidores primarios y secundarios, que resuelven de a pedazos.

  - Las consultas se responden preguntando quien es el servidor responsable de cada zona.

- SMTP
  - *Simple Network Management Protocol* versión v3 sirve para administrar máquinas con ambiente no asegurado.

## 5.2 Firewalls

- **Firewall:** Es un separador entre dos partes de una red para controlar lo que pasa entre ellas.
- **Propositos:**
  - Proteger mis datos, a nivel de confidencialidad, integridad y disponibilidad.
  - Proteger mis recursos de uso por agentes externos (ejemplo *botnets*, *zombies*, etc.).
  - Proteger mi reputación: Evitar que mis recursos se usen para fines maliciosos generando desconfianza en mí.
- **Tipos:**
  - Filtrado de paquetes: Cada paquete que entra o sale de la red es verificado y permitido o denegado de acuerdo a un conjunto de reglas definidas por el usuario.
    * Se basa en las direcciones (IP + Puerto) de origen y destino, además del protocolo.
    * Suelen usarse en los routers.
    * Son eficientes y fáciles de implementar.
    * Puede producir reglas complicadas y potencialmente inseguras. Por ejemplo:
    * FTP Activo, reglas para el servidor y cliente:
      · S: Permitir cualquier puerto $p \in [1024, 65535]$ al 21 del server FTP
      · S: Permitir al 21 del server FTP cualquier $p \in [1024, 65535]$
      · S: Permitir al 20 del server FTP cualquier $p \in [1024, 65535]$
      · C: Permitir cualquiera en mi red de cualquier puerto alto a un puerto 21
      · C: Permitir cualquier 21 a un puerto alto.
      · C: Permitir cualquier 20 a un puerto alto.
      · C: Permitir cualquiera en mi red de cualquier puerto alto a un puerto 20.
  - Stateful Inspection: Es como filtrado de paquetes normal, pero además puede tener en cuenta el estado de la sesión de la conexión, y utilizarlas para aplicar reglas que varían según el momento de la conexión.
    * Puede analizar el protocolo superior que se esta usando.
    * Mayor precisión en el filtrado, reglas más cortas y más estrictas.
    * Mayor necesidad de procesamiento para cada paquete y cada conexión (potencialmente caro).
  - Gateways de circuito: *proxy* no inteligente, simplemente renvían la conexión. Están en la capa de sesión del modelo OSIŚon independientes del protocolo pero el cliente debe conocerlo. Se usan con políticas estrictas de filtrado.
  - Gateways de aplicación: *proxy* inteligente. Conoce del protocolo con el que se esta haciendo la comunicación. El cliente no solo debe conocerlo sino que se debe estar usando un protocolo que permita proxys. Permite un mejor uso de autenticación, permite un mejor control de uso de servicios y facilita la generación de registros de auditoría y el uso de caches.
  - Ingress/Egress Filtering: Sirve para evitar IP Spoofing. Ingress es que no permito pasar paquetes a mi red que vienen de afuera pero cuya IP de origen es una IP interna. Egress es que paquetes con IP *spoofeada* no pasan por el Firewall (para evitar que se dañe mi reputación).

  - Personal: Es un software instalado en una computadora generalmente personal y que controla la comunicación entre el equipo y el mundo exterior. Permiten por ejemplo filtrar accesos por aplicación, protocolo, etcẏ consultar al usuario cuando una aplicación quiere hacer una conexión.

### 5.2.1 NAT

- Definición: *Network Address Translation*. Consiste en rescribir las IPs de origen y destino *on the fly* para permitir modificar las políticas de envio. Se usa por ejemplo para tener una red privada con un punto de acceso en la internet pública (ejemplo: un router *wifi* que usa NAT para tener una sola IP pública que le da el *ISP* pero para llegar a todos los dispositivos de la casa, renviandoles a sus IPS privadas de manera acorde).
- Tipos:
  - DNAT: Consiste en modificar la IP de destino a los paquetes que llegan a una IP privada, y modificar la IP de origen cuando ocurre lo inverso. También conocido como *port forwarding*. Se usa para publicar un servicio local en la Internet, o poner un servidor en la *DMZ*.
  - SNAT: Lo mismo que DNAT pero usando la IP de origen. El significado varía segun *vendor*
  - IP Masquerading: Consiste en que el Router enmascara la IP de una computadora local con la propia, de manera que el pedido parece venir de la IP indicada en la máscara. Permite ocultar la topología de la Red. Es una forma de SNAT.

### 5.2.2 Esquemas de redes

- Screening Router: Filtra paquetes a la red interna. Generalmente usan filtrado estático y reglas complejas.
- Bastion Host: Se utiliza un servidor en la DMZ que es el que esta más "expuesto" a ataques. Generalmente corre un proxy server y nada más, y se tunea especialmente (se lo audita seguido, corre software parcheado y que no tenga exploits conocidos, etc.) para ser lo más resistente posible a ataques. Suele tener reglas simples para evitar agujeros que permitan ataques. Protege entonces el resto de la red.
- Screened Host: Se utiliza un Bastion Host que efectivamente SEPARA las dos redes. Usa proxies y es muy seguro. Como el Bastion Host es el que hace las requests entre zonas, no necesita NAT.

### 5.2.3 Tipos de políticas de Firewall

- Default Permit: Permite todo salvo algunos protocolos.
- Default Deny: Niega todo excepto lo explicitamente permitido.
- Default Permit es MUY MUY INSEGURO, porque cualquier servicio habilitado sin conocimiento y que posee una vulnerabilidad es una puerta a un ataque.

### 5.2.4 Implementación

- Diseño de la red (topología)
- Definición de políticas entre las redes (usando herramientas gráficas como FWBuilder o shorewall).
- Mantenimiento: Mantener las reglas al día, mantener el software y el SO parcheados, y revisar los logs de firewall.

## 5.3 Monitoreo de redes

### 5.3.1 Recolección de trafico

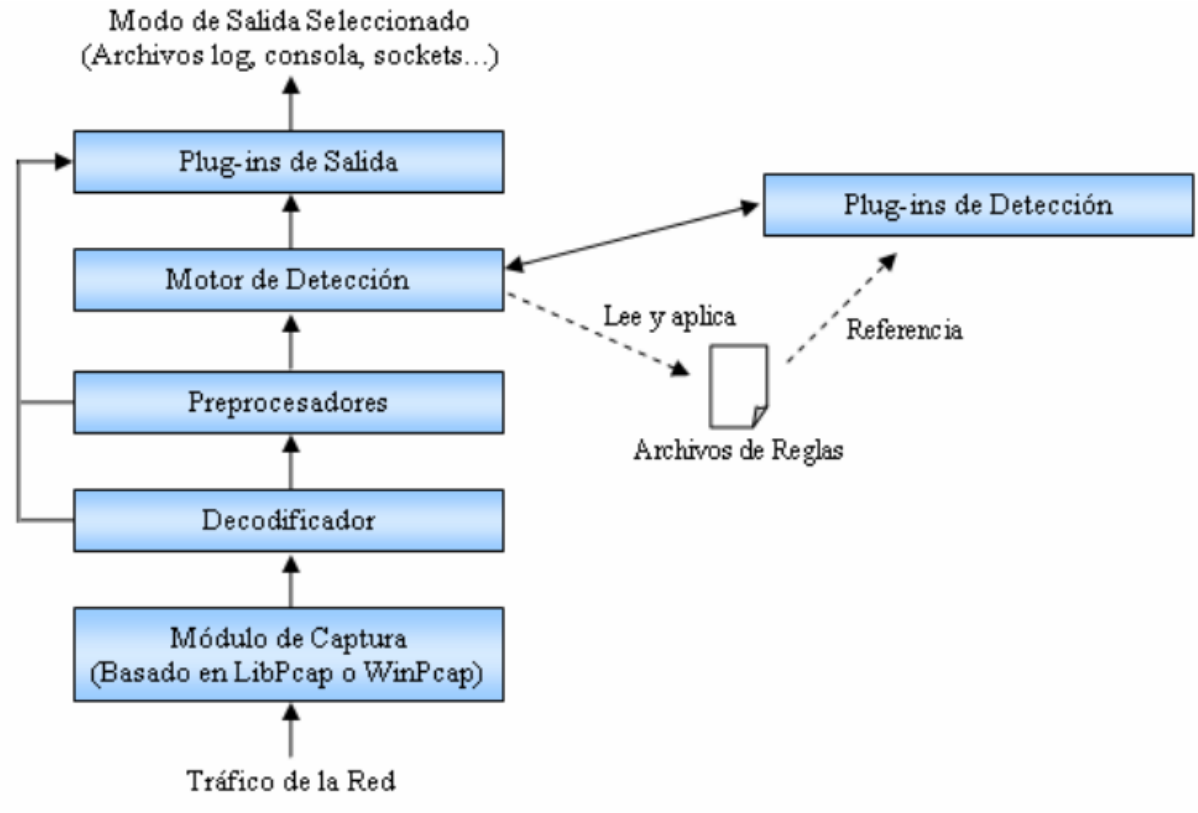
- **tcpdump**: Consigue todos los paquetes, con un alto costo de almacenamiento y procesamiento pero permite mejor granularidad, con lo cual se pueden utilizar todas las herramientas que queramos para analizarlos y se puede hacer un análisis forense más completo, incluso en tráfico cifrado.
- **sesiones (argus,...)**: Resumen de conversaciones entre sistemas. Es compacto e independiente de los datos que se esten enviando. No tiene en cuenta datos de la aplicación así que no es suceptible por encriptación, utiliza los paquetes de los protocolos para armar tablas de conversaciones.
- **estadisticas (tcpdstat,...)**: Visión sumarizada de eventos, permite ver el tráfico a alto nivel. Se puede usar por ejemplo en equipos que generen muchisimo tráfico.
- **alertas (snort,...)**: Alerta por IDS tradicionales, por regla o anomalía.

### 5.3.2 IDS

- Definición: Se denomina IDS (*Intrusion detection system*) al proceso de monitorear los eventos que ocurren en un sistema o red de computadoras, buscando señales que indiquen que haya habido o esta habiendo una intrusión.
- Problemas comunes:
  - **Falso positivo**: Ocurre cuando una herramienta clasifica una acción como una posible intrusión pero la misma era un uso legítimo del sistema, y no constituye una violación de seguridad. Se pueden disminuir haciendo más específicos los patrones a detectar.
  - **Falso negativo**: Ocurre cuando una intrusión no es detectada por el IDS.
  - **Falsa alarma**: Cuando se dispara una alerta por un ataque por un patrón de comportamiento que es parte de uno, pero donde el mismo no representa un peligro. Por ejemplo, el uso de un exploit en un software distinto al vulnerable, o que fue parcheado, o que no funciono.
- **Objetivos**:
  - Detectar una amplia variedad de intrusiones, con la correspondiente necesidad de adaptarse a nuevos ataques o patrones de comportamiento y aprenderlos, por parte de la herramienta.
  - Detectar las intrusiones en un tiempo razonable, incluso a veces en tiempo real, sin afectar los tiempos de respuesta del sistema. Se puede por ejemplo detectar intrusiones en las ultimas horas o minutos.
  - Presentar la información de manera fácil de entender: A veces se monitorean muchos sistemas, y el analista necesita además información sobre el ataque para poder determinar si fue un ataque y que respuesta dar. Lo ideal sería un indicador binario (OK - Alerta) pero no alcanza.
  - Ser tan preciso como se pueda.
- Clasificacion:
  - IDS de Host (HIDS, ejemplo Swatch) vs IDS de Red (NIDS, ejemplo SNORT).
    - NIDS: Reconoce patrones en el tráfico de Red.
    - HIDS: Monitorea logs e integridad de archivos, comportamiento extraño de usuarios, parametros de sistema excedidos.
    - Ataques a NIDS: Insertion (Meter un paquete que el IDS acepta pero el sistema no, de manera que ambos ven cosas distintas) y Evasion (Meter un paquete que el IDS rechaza pero que el sistema aceptaría, de manera de esconder un ataque en por ejemplo la manera en la que se rearman los paquetes).
  - Basado en heuristicas/estadisticas vs Basado en patrones

### 5.3.3 SNORT

Arquitectura:




Componentes:

- Sniffer: Captura los paquetes de la red usando pcap.
- Decodificador: Identifica protocolos y construye las estructuras de datos necesarias para examinar los paquetes.
- Preprocesadores: Prepara los datos para detección. Se dedica a por ejemplo detectar escaneos, ensamblar los datos contenidos en paquetes de una misma sesión o reensamblar paquetes fragmentados.
- Motor de detección: Analiza los paquetes en base a reglas definidas para detectar ataques.
- Salida o postprocesadores: Permiten definir que, como y donde se guardan las alertas y los paquetes de red que las generaron.

Diferencias y cosas a remarcar

- Existen herramientas como Snorby o Sguil que permiten analizar los resultados de Snort.
- Existen otras herramientas como Bro NSM que también pueden describir restricciones de actividad, analizar contexto, relacionar eventos, etc.

### 5.3.4 HIDS

- Swatch:
  - Empezo como un simple watchdog para monitorear actividades del log que producía syslog en UNIX.
  - Permite definir entradas a resaltar o ignorar.
- Logcheck:
  - Similar a Swatch, con reglas adaptadas para Debian. Se ejecuta usualmente cada hora, revisa los logs por data anormal y genera un reporte que se puede mandar por email.
- Tripwire,Aide:
  - Monitorea el agregado, borrado y modificación de archivos. Genera una base de datos con la información de cada archivo en el sistema. Periodicamente vuelve a generar la información y la compara.
  - Checkea archivos nuevos, eliminados, contenido y atributos (fecha, firmas, permisos, tamaño, etc.)
- OSSEC
  - HIDS Open Source, multiplataforma, realiza analisis de logs, checkeo de integridad, monitoreo de registro, etc.

### 5.3.5 IPS

- Definición: Un IPS de Red es un sistema de prevención de intrusiones, es decir que además de detectar intrusiones puede hacer algo al respecto. Pueden ser de red o de host.
- Red: Funciona como dispositivo inline en modo bridge, puede decidir filtrar el tráfico para que no llegue a destino si detecta una intrusión. Puede también modificar el contenido del tráfico.
- ModSecurity: Motor de detección y prevención de intrusiones para aplicaciones web, es un modulo de Apache. Es un IPS de Host.
  - Intercepta pedidos HTTP antes de que sean procesados por el host.
  - Intercepta el cuerpo de los pedidos.
  - Intercepta almacena y valida los archivos subidos.
  - Realiza acciones antievasivas en forma automática.
  - Procesa mediante un conjunto de reglas configurables.
  - Las respuestas al cliente son también interceptadas y se analizan en base a reglas.

### 5.3.6 Honeypots

- Honeypot: Es una trampa cuyo proposito es detectar, esquivar o contraatacar al mal uso de un sistema informático. Se suele implementar como una computadora que parece estar conectada a la red y que tiene recursos de interés para un atacante, pero que en realidad esta fuertemente monitoreada y aislada de la red.
- Tipos:
  - Baja interaccion: Emula servicios, aplicaciones, SO, etc. Bajo riesgo, faciles de mantener, pero proveen información limitada.

  - Alta interacción: Servicios, aplicaciones y SO reales. Consumen mucho tiempo de mantener, alto riesgo pero dan mucha información.

- Ejemplos en orden creciente de interacción: Honeyd, Nepenthes, Honeynets
- Honeyd:
  - Crea equipos virtuales en una red
  - Los equipos puede ser configurados para ejecutar servicios arbitrarios.
  - Simula distintos sistemas operativos.
  - Puede reportar resultados a Prelude.
- Nepenthes:
  - Baja interacción, emula vulnerabilidades conocidas, especialmente las que usan los worms para diseminarse y capturar worms.
- Honeynets:
  - Se agregan equipos enteros, no un producto o software. De alta interacción. Toda la red es altamente monitoreada y los paquetes son capturados.
  - Control de flujo de datos: Es necesario controlar los datos por el riesgo de que la honeynet sea usada como punto de ataque.
  - Captura de datos: Se toma la actividad de red, de sistema y de aplicaciones.
  - Análisis de datos.
  - Sebek: Herramienta de captura de datos que corre como módulo de kernel. Envia la información de la red sobre el atacante a un servidor de manera que no se pueda sniffear, utilizando para eso SSH entre los dos. De esta manera se juntan datos sin que el atacante lo sepa.

# 6 Bibliografía


- Diapositivas de clase de Rodolfo Baader.
- Apuntes de Julián Sackmann.
- Wikipedia.