SEGURANÇA – RESUMO GERAL

TIPOS ATIVIDADES ILEGAIS – Acesso a informação alheia, modificação de informação alheia, sobrecarga de recursos alheios, denial of service alheio, vandalismo (pura interferência em sistemas alheios sem beneficios para quem ataca).

PRIMEIRA ABORDAGEM – Computadores processam muita informação em rápidos intervalos de tempo, o que favorece fraquezas e brechas. Quando conetados à internet é possivel a alheios intervirem no sistema e influência-lo. Isto tudo acontece porque os utilizadores não conhecem os perigos e não estão preparados para eles. A segurança é cara, e nunca é 100% garantida. Ficam então sujeitos vulnerabilidades que podem originar riscos como destruição de informação que levou tempo e pode valer dinheiro, acesso a informação confidencial e privada, sobrecarga do sistema, ou fazerem-se passar por nós. Estes riscos advêm de possiveis aplicações com bugs, pessoas e falta de conhecimento, comunicações inseguras. É por isso importante definirmos dominios de segurança (o que é aceite ou não, quem pode ou não, quem são), estratégias de segurança e meios de combate a estas vulnerabilidades. Estas estratégias espalham-se por vários setores, como a autenticação, controlo de acesso, privilégios de execução, algoritmos e protocolos cifrados, etc. Devemos definir politicas de segurança, que definem o que deve ser feito, e implementar mecanismos, que irão aplicar essas politicas estabelecidas.

MEDIDAS DE SEGURANÇA – Existem várias medidas de segurança: descorajamento (através de firewalls, autenticações, comunicações seguras, ou fazendo uso da lei), deteção (sistemas de deteção de intrusos, ouvir, análises), decepção (armadilhas a invasores), prevenção (scans de vulnerabilidade, princípio do menos priviligiado) e recuperação (backups, sistemas redundantes). As três primeiras aplicam-se para a maior parte de problemas conhecidos (port scanning, buffer overflows...). Prevenção aplica-se para problemas bem conhecidos ou mesmo desconhecidos (reação a mensagens mal formadas, ataques surpresa, software bugs)

ATAQUE DO DIA ZERO/HORA ZERO – É um ataque que explora vulnerabilidades desconhecidas para os vendedores do software. Ou seja, mal o software é lançado, ainda não há registo de bugs, mas eles existem e são imediatamente explorados pelos atacantes. A unica forma de resistir a este ataque é apostar na diversidade de sistemas operativos para o qual o software pode ser lançado.

CVE (Common Vulnerabilities and Exposures) – Dicionário com todas as vulnerabilidades e exposições de segurança conhecidas. Isto permite melhor gestão das vulnerabilidades, e alerta no caso de uma. Permite a troca de informação entre produtos de segurança e fornece um ponto de referência para avaliar o indice de cobertura de serviços e ferramentas. Uma vulnerabilidade é um erro num software, que pode ser aproveitado por um atacante para acesso a informação, mudança de identidade, denial of service, etc,... Uma vulnerabilidade pode ser aproveitada por um atacante, uma exposição serve como um trampolim para o atacante poder aproveitar vulnerabilidades. Uma exposição pode ser uma falha de configuração ou um erro se não comprometer diretamente, mas pode vir a ser uma componente para um ataque bem sucedido. A existência do CVE permite que estes problemas sejam referenciados sempre usando uma mesma linguagem, permite a partilha de dados entre ferramentes e base de dados de vulnerabilidades ou equipas de pesquisa. Além disso, o CVE permite que as próximas ferramentas de segurança sejam melhores, mais compreensivas e melhor preparadas. Não permite a defesa contra ataques do dia zero. Uma entrada CVE tem sempre o seguinte formato: numero identificador, estado (candidato ou entrada), descrição, informação extra.

CWE (Common Weakness Enumeration) – Linguagem usada para discutir, encontrar e tratar de vulnerabilidades em softwares. É possivel consultar uma lista de CWE no site da MITRE, lista organizada hierarquicamente, pode haver CWEs filhas associadas.

CERT (Computer Emergency Readiness Team) – Organização responsável por assegurar que sistemas e tecnologias trabalhem de forma a resistir a ataques em redes, e consigam resistir a falhas criticas nos seus serviços em caso de ataques bem concedidos, acidentes ou falhas.

CSIRT (Computer Security Incident Response Team) – Organização responsável por receber, rever e responder sobre incidentes de segurança em computadores. Existem organizações responsáveis por partilhar rapidamente novas informações sobre vulnerabilidades, por forma a se agir rapidamente (US-Cert, SANS, Microsoft Security Response Center, Cisco Security Center, etc).

STACK – Funciona como uma caixa com várias prateleiras de 4 bytes cada. É usada para passar parâmetros a funções ou guardar variáveis locais. Cresce do topo para o fundo, e os valores são colocados na stack por push, e retirados por pop. Para inserir valores na stack é necessário alocar espaço. No fim esse espaço é esvaziado. É gerida por vários registos, EBP (indica o inicio da stack), ESP (indica o fim da stack), EIP (indica a instrução atual), ESI (um endereço na stack), EDI (um endereço no segmento de dados).

BUFFER OVERFLOW – Quando escrevemos para além da capacidade da stack. Isto vai fazer com que começemos a escrever por cima de outras variáveis referentes à função em questão, como variáveis locais, ou endereços de retorno. Dessa forma, é possivel controlar o comportamento do programa, injectando código especifico na stack até ela rebentar e escrever por cima de outras variáveis. Podemos fazer com que a função salte para endereços desejados, ou que certas variáveis assumam valores à nossa escolha. Os buffer overflows podem acontecer em C mas não em Java, uma vez que em Java, o compilador verifica sempre se o tamanho dos dados introduzidos viola o tamanho previamente estabelecido, dando assim erro, e impedindo o buffer overflow. Em C, não há esse controlo. Uma maneira de detetar o buffer overflow é usando canários. Na stack, entre o buffer e as variáveis de controlo (EBP e EIP) é colocado um número aleatório, numa posição definida da memória. Este valor é o canário. Ao sairmos da stack vamos comparar este valor com o original, e caso seja diferente, é sinal que o valor foi alterado e dessa forma ocorreu overflow. Como o número é aleatório é praticamente impossivel de adivinhar, daí ser eficaz este mecanismo. Outra forma de evitar que variáveis de controlo sejam afetadas é guarda-las numa outra stack. Temos assim stack para dados, e stack para controlo.

SQL INJECTION – Este tipo de ataques é comum em páginas web que façam uso de servidores SQL. O atacante faz uso de comandos SQL em zonas de validação que usem comandos SQL, alterando assim o objetivo pré-definido. Isto é possivel através de comandos básicos como 'password is X or '1' = 1. Desta forma como 1 é sempre igual a 1, a password introduzido é completamente inutil, e é possivel aceder às contas do utilizador em questão. É possivel alterar passwords, nome de utilizadores ou mesmo apagar todos os utilizadores com simples comandos de DROP.

ARP SPOOFING – É possivel usar o protocolo ARP para ataques na rede. Os endereços MAC podem ser alterados. Se usarmos um endereço MAC já existente na rede, isto vai provocar que pacotes com destino a esse endereço MAC venham parar a nós em vez do host com o MAC real. Desta forma é possivel envenenar as caches de outros hosts com falsos endereços MAC. Dessa forma podemos fazer com que pacotes com destino a um determinado endereço IP com endereço MAC x, seja desviado para outro host com o mesmo MAC x. É assim possivel fazer com que o host atacado esteja a comunicar sozinho na rede, e nem receba os seus pacotes a não ser o atacante, ou então desviar os pacotes destinados a outro host, ou mesmo impedir o acesso à internet por parte do host atacado. Para evitar este tipo de ataques, alguns switches limitam o numero de endereços MAC por porta, ou então podemos usar entradas estáticas, ou usar software para deteção de alteração de endereços MAC.

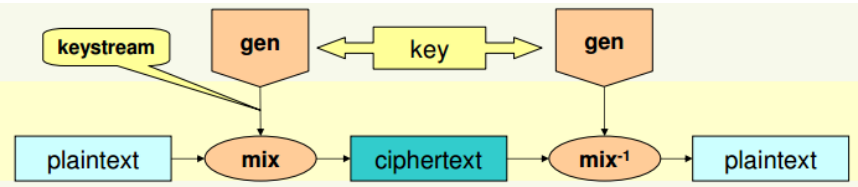
CRIPTOGRAFIA – Arte de escrever texto escondido. Para isso, o texto vai ser cifrado usando um algoritmo especifico, e é depois decifrado. É utilizado um parâmetro chave no algoritmo que vai permitir assim a correta decifragem consoante a cifragem feita. A arte de cifrar e decifrar designa-se por criptoanálise. Existem vários tipos de cifragem:

- Transposição – texto original é misturado;
- Substituição – cada caractér é alterado para outro. Dividido em vários métodos
    - Mono-alfabético – usa apenas um alfabeto de substituição. No entanto, pode ser decifrado usando estatisticas;
    - Poli-alfabético – usa vários alfabetos de substituição, com um dado periodo. No entanto, sabendo o periodo, é possivel decifrar o conteúdo (descobrir o periodo por estatistica, medir distâncias entre caracteres iguais...);
    - Homofónico – usa um alfabeto para gerar vários;
    - Poligramático – usa vários alfabetos para gerar vários;

MÁQUINAS COM ROTOR – Implementam cifras poli-alfabéticas complexas. Cada máquina é constituida por vários rotores, onde cada um, após carregarmos numa tecla, vai rodar e gerar uma respetiva letra cifrada. Essa letra é indicada através de luzes. Para chave de cifragem pode ser considerados o conjunto de rotores usado, a sua posição ou a ordem deles. A decifragem é conseguida usando rotores simétricos, half-rotors que fazem uso de um disco de reflexão.

ASPETOS IMPORTANTES SOBRE CRIPTOGRAFIA – É importante termos a noção que o mau uso de cifragens pode levar a criação de vulnerabilidades. Temos também de ter em conta o nivel de segurança computacional usado, tendo em conta o custo da criptoanálise, a disponibilidade de estruturas para criptoanálise e o tempo de vida de um texto cifrado. Para isso devemos seguir os 5 critérios de Shannon: o tamanho e complexidade da chave usada, a simplicidade da implementação, a propagação de erros e a dimensão do texto cifrado. Uma relação complexa entre a chave, o texto a cifrar e o texto cifrado pode originar confusões. Pode também ocorrer difusão, conhecido pelo efeito de avalanche, ou seja, se um bit do texto a cifrar é alterado, então o texto cifrado vai mudar completamente, vários bits vão mudar também! O segredo da cifragem está apenas na chave, o texto cifrado não é secreto! Como um criptoanalista possui vários exemplos de texto cifrado usando um mesmo algoritmo e chave, mesmo não conhecendo a chave, ele pode ter uma ideia do texto original baseado nos exemplos que tem. Isto pode originar ataques! Devemos sempre assumir esta possibilidade.

CIFRAS CONTINUAS – Tipo de cifra poli-alfabética simétrica onde cada caracter de um texto a cifrar vai ser combinado com um caractér de uma chave continua (como uma cadeia) originando um caracter cifrado. Acontece assim uma mistura da chave (pseudo-aleatória) com o texto a cifrar. A chave continua é originada através de um valor alterado por shift registers (LFSR, não sendo dessa forma completamente aleatória). É facilmente invertida através de um XOR. A cadeia da chave continua é originada a partir de outra chave, que vai ser usada para a decifragem. A cadeia apenas deve ser usada uma vez, senão a soma de duas decifragens usando a mesma key vai resultar na soma dos dois textos a cifrar. O tamanho do texto a cifrar deve ser inferior ao tamanho da cadeia, senão o caracter gerado para decifragem vai ser o caracter da cadeia. É também importante que não haja ciclos na cadeia, porque isso pode facilitar o trabalho de criptoanalistas para descobrirem a chave e poderem decifrar textos. Em comparação com as cifras em bloco, as contínuas são bem mais rápidas e menos complexas em termos de hardware. As cifras continuas podem ser sincronas (a chave continua muda independentemente do texto a cifrar e do texto cifrado) e auto-sincronas (atualizam a chave continua conforme os bits do texto cifrado).



CIFRA DE VERNAN – Cada bit do texto a cifrar é cifrado por uma adição modular (XOR, por exemplo) com um bit de uma chave aleatória completamente secreta (pad) do mesmo tamanho, originando texto cifrado. Uma adição modular consiste num sistema onde os numeros "voltam para trás" quando atingem certo valor, chamado módulo. No entanto apresenta problemas práticos, como garantir que o pad é perfeitamente aleatório, obrigatoriedade do tamanho da pad ser do tamanho do texto a cifrar, e garantir que cada pad é usado apenas uma vez.

CIFRAS MODERNAS – Divide-se consoante o modo de operação, e consoante a chave utilizada: Cifra por bloco com chave simétrica, cifra por bloco com chave assimétrica e cifra continua com chave simétrica. As chaves simétricas são partilhadas por 2 ou mais pares, permitindo mais privacidade entre eles, uma vez que apenas eles conhecem as chaves. Têm como grande vantagem o desempenho, e como desvantagem o facto de quantos mais pares forem precisos, mais chaves têm que ser distribuidas (o quadruplo), e mais relações secretas existem, porque a atribuição de chaves é sempre um problema, então entre muitos hosts pior ainda.

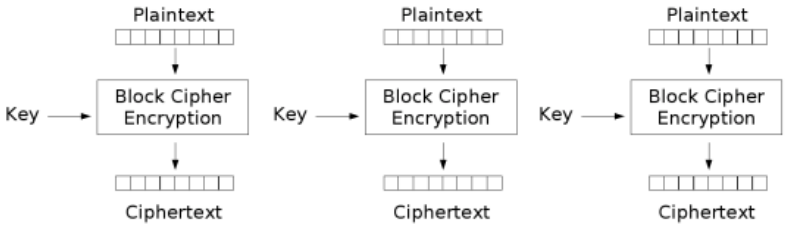
$$E_K(P) := E(K,P) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n,$$

$$E_K^{-1}(C) := D_K(C) = D(K,C) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n,$$

$$\forall K : D_K(E_K(P)) = P.$$

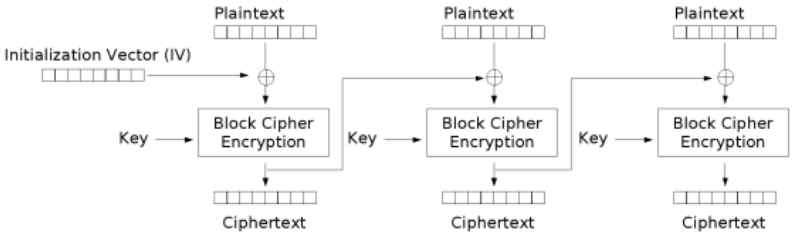
- **Cifra por bloco com chave simétrica** – Em vez de cifragem bit a bit, aqui cifra-se um conjunto de bits como se de um bloco se tratasse. Consideram-se blocos grandes (64, 128, 256 bits). Os bits destes blocos vão ser combinados com os da chave (substituição), gerando confusão. De seguida são permutados gerando difusão. Usam o método de Feistel, em que a cifragem e decifragem é praticamente igual, apenas invertendo a forma como a chave está construida. Usa como principais algoritmos:
    - **DES (Data Encryption Standard) -** Para blocos de 64 bits e chaves de 56 bits. Primeiramente os bits estão sujeitos a permutações e iterações, passando de seguida por uma rede de Feistel (bloco de texto a cifrar, é partido em 2, e um deles sofre passa fela função de arredondamento juntamente com uma sub-chave, e o resultado é juntado com a outra parte com um XOR) e finalmente mais substituições, permutações, expansões e compressões. O DES dá a possibilidade de escolhermos a chave a usar, ou 4 fracas, ou 12 semi fracas, ou 48 potencialmente fracas. É possivel o ataque a este algoritmo, através da pesquisa intensiva por espaços na chave. Dessa forma considera-se que 56 bits são poucos, e tornam a pesquisa exaustiva passivel de se usar. Por isso podemos adoptar por dupla ou tripla encriptação com 2 ou 3 chaves;
    - **IDEA (International Data Encryption Algorithm) -** Para blocos de 64 bits e chaves de 128 bits;
    - **AES (Advanced Encryption Standard)** – Blocos variáveis, chaves de 128, 192 ou 256 bits;

  Este tipo de cifra sobre a mesma chave origina sempre blocos cifrados iguais. Dessa forma é necessário ter vários modos de operação, que baralhem a informação por forma a não repetir o texto cifrado. Temos como principais modos de operação:
    - **ECB (Electronic Code Book) –** Mensagem dividida em blocos, cifrados separadamente. Blocos iguais geram blocos cifrados iguais, podendo criar padrões.

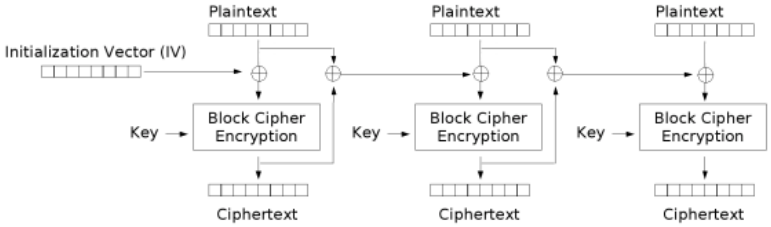


    - **CBC (Cypher Block Chaining) –** Mensagem dividida em blocos, e a cada bloco é aplicado um XOR com o bloco previamente cifrado, criando uma dependência entre todos os blocos cifrados. O

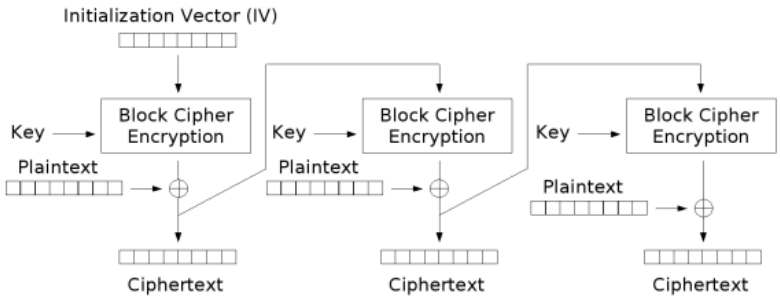
primeiro bloco é cifrado com um vector de inicialização aleatório. A fórmula de cifragem é $C_i = E_k (P_i$ XOR $C_{i-1})$, C_0 = IV e decifragem é $P_i = D_k (C_i$ XOR $C_{i-1})$, C_0 = IV. A mensagem no fim tem de se juntar a um padding para igualar o tamanho do bloco inicial.



- **PCBC (Propagation Cypher Block Chaining) –** Semelhante ao CBC, mas o XOR do texto a cifrar é feito com um XOR do texto a cifrar e texto cifrado do ultimo bloco; Se dois textos cifrados adjacentes mudam isto não vai afectar a decifragem dos restantes blocos. A fórmula de cifragem é $C_i = E_k (P_i$ XOR P_{i-1} XOR $C_{i-1})$, P_0 XOR C_0 = IV e decifragem é $P_i = D_k (C_i$ XOR P_{i-1} XOR $C_{i-1})$, P_0 XOR C_0 = IV;

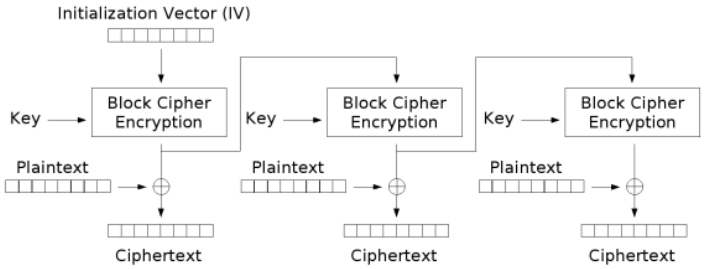


- **CFB (Cypher Feedback Mode) -** Tranforma blocos de cifra em cifradores continuos auto-sincronizados. Processo semelhante ao CBC, mas o vector de inicialização aleatório é cifrado. Neste modo é possivel cifrar bit a bit, como se fosse uma cifragem continua. Para isto, faz-se uso de um conjunto de shift registers do tamanho do bloco, que vão receber o vector de inicialização e só depois vai ser enviada a informação para cifragem. Após isto, os x bits mais significativos vão ser combinados por XOR com x bits do texto a cifrar, originando x bits de texto cifrado. Estes bits vão ser colocados no shift register, e o processo repete-se para os próximos x bits do texto a cifrar. O processo de decifragem é similar, mas invertido. O processo de cifragem é traduzido pela fórmula: C_i = head($E_k (S_{i-1})$, x) XOR P_i, onde head (a, x) são os x bits mais significativos de *a,* e *S* é o estado do shift register. A decifragem é traduzida por P_i = head($E_k (S_{i-1})$, x) XOR C_i, o estado do shift register é dado por: $S_i = ((S_{i-1} << x) + C_i) \bmod 2^n$, onde *n* é o numero de bits do vetor de inicialização, e S_0 = IV. É capaz de recuperar de erros, graças à conversão bit a bit e à constante sincronização.

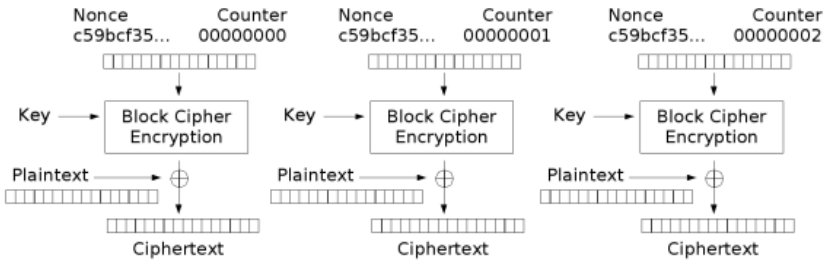


- **OFB (Output Feedback Mode) –** Tranforma blocos de cifra em cifradores continuos sincronizados, que sofrem um XOR com o texto a cifrar originando o texto cifrado. Para o primeiro bloco é usado um vector de inicialização aleatório cifrado, e vai sendo repetidamente cifrado originando uma chave continua. Há realimentação. Cada saída do bloco de cifragem vai depender de todos os blocos anteriores. No entanto, o texto a cifrar e o texto cifrado são usados apenas num XOR posterior, por

isso a cifragem no bloco pode ser feita com avanço, permitindo que os XORS possam ser feitos em paralelo. É a grande diferença para o CFB.

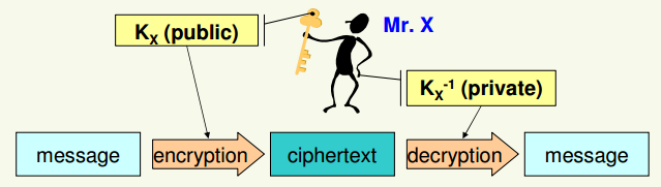


- **CTR (Counter Mode) -** Tranforma blocos de cifra em cifradores continuos sincronizados. A próxima chave continua é gerada através da cifragem sucessiva de valores de um vector de inicialização contador. Há realimentação.

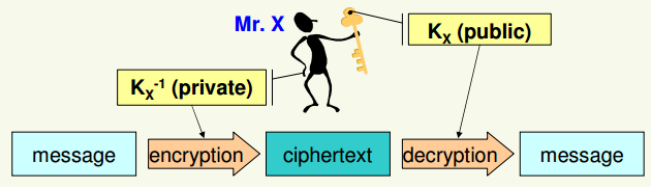


	ECB	CBC	OFB	CFB	CTR
Pattern exposure		✓	✓	✓	✓
Confusion on the cipher input		✓		✓	Secret counter
Same key for different messages	✓	✓	other IV	other IV	other IV
Tampering difficulty	✓	✓ (...)		✓	
Pre-processing			✓	...	✓
Parallel processing Uniform random access	✓	Decryption Only	w/ pre-processing	Decryption only	✓
Error propagation	Same block	Same block Next block		Some bits afterwards	
Capacity to recover from losses	Block Losses	Block Losses		✓	

- **Cifra por bloco com chave assimétrica –** Este algoritmo usa um par de chaves (uma privada, pessoal e não transmitida, e uma pública). Isto vai permitir privacidade sem que ocorra troca de informação secreta, e autenticação de conteudo (integridade de dados) e da origem (autenticação da fonte ou assinatura digital). Tem como vantagem o facto de se houver N pares a requererem troca de informação, apenas serão precisas N pares de chaves e em como principal desvantagem o desempenho, e o elevado consumo de memória e pode apresentar problemas na distribuição de chaves públicas e devido ao tempo de vida das chaves. O processo de cifra é simples. A mensagem é cifrada com a chave pública do destinatário, e este quando a receber desencripta com a sua chave privada. Dessa forma para enviarmos algo confidencial para um host, apenas temos de saber a chave pública dele.



Neste modo, não há autenticação na fonte, não haveria forma de saber quem enviou a mensagem. Para isso, a mensagem teria de ser cifrada com a chave privada do emissor, mas neste caso qualquer um com a chave publica do emissor poderia decifrar a mensagem.



Tem como principais algoritmos:

- **RSA (Rivest, Shamir, Adelman) –** Algoritmo cuja segurança baseia-se na complexidade computacional de factorização (decomposição de um número por pequenos divisores) e cálculo de logaritmos modulares de grandes numeros (loga(b) => $a^x = b$). A cifragem segue os passos seguintes:
  - Existência de dois números primos aleatórios, *p* e *q*, e do mesmo tamanho;
  - Multiplar *p*q* = n, onde n é usado como o tamanho da chave em bits;
  - Computar @(n) = (*p*-1)(*q*-1);
  - Escolher um inteiro *e* de tal forma que 1 < *e* < @(n), em que *e* e @(n) seja co-primos. *e* vai ser o expoente da chave publica;
  - Calcular *d* de tal forma que *e*d* = mod(@(n)). *d* vai ser o expoente da chave privada;
  - A chave pública é tal que K = (e, módulo(n)), e a privada é tal que K^{-1} = (d, módulo(n));
  - A cifragem vai consistir em $C = P^e$ mod(n);
  - A decifragem vai consistir em $P = C^d$ mod(n);
  - *d, p, q* e @ devem ser secretos;
  - O valor do expoente *e* deve ser grande, poruqe caso P seja pequeno, o resultado de P^e é ligeiramente mais pequeno que o mod(n). Dessa forma usando a *e*th raiz do texto cifrado, é possivel decifrar a mensagem;

- **ElGamal –** Semelhante ao RSA, sem a complexidade factorial, possui uma variante para DSA (Digital Signature Algorithm)

  $\beta = \alpha^x \bmod p \qquad K = (\beta, \alpha, p) \qquad K^{-1} = (x, \alpha, p)$
  k random, $k \cdot k^{-1} \equiv 1 \bmod (p-1)$
  Signature of M: $(\gamma,\delta) \quad \gamma = \alpha^k \bmod p \quad \delta = k^{-1}(M - x\gamma) \bmod (p-1)$
  Validation of signature over M: $\beta^\gamma \gamma^\delta \equiv \alpha^M \pmod p$

- **Outras técnicas – Diffy-Hellman Key Agreement**

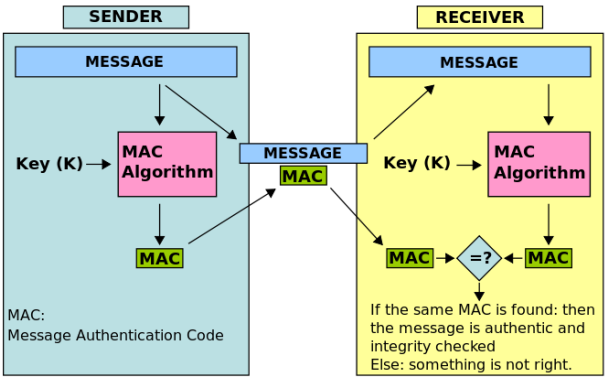
- **Cifra continua com chave simétrica –** Cifra continua que origina numeros aleatórios através do uso de shift registers lineares com realimentação (LFSR) (os valores de saída dos shift registers são considerados numa função booleana, aumentando assim a segurança. Para isso ser posivel, os LFSR têm relógios irregulares, controlados pela saída de outro LFSR. A saída deste é que escolhe qual valor dos outros usar, por exemplo se o valor é 0, o LFSR1 recebe o relógio, se o valor é 1, o LFSR2 recebe o relógio. A saída pode ser um OR entre os ultimos bits do LFSR de escolha e do escolhido. O estado inicial de todos os LFSR é a chave. Uma boa função de realimentação permite criar sequências praticamente aleatórias, e cujo período é grande). O novo bit é assim uma função linear do estado anterior. A função linear mais usada é o XOR. O LFSR vai originar 2^n -1 sequências não nulas. Se uma das sequências tiver um período de 2^n -1, então todas têm.
  Tem como principais algoritmos o A5(/1 ou /2 (GSM), usa 3 LFSR de comprimento diferente), RC4 e SEAL.

REFORÇO DE SEGURANÇA – Podemos reforçar a segurança através de multiplas cifragens, usando em cada iteração uma chave nova. Temos a dupla cifragem que é facilmente quebrada, e temos a tripla cifragem, feita com uma

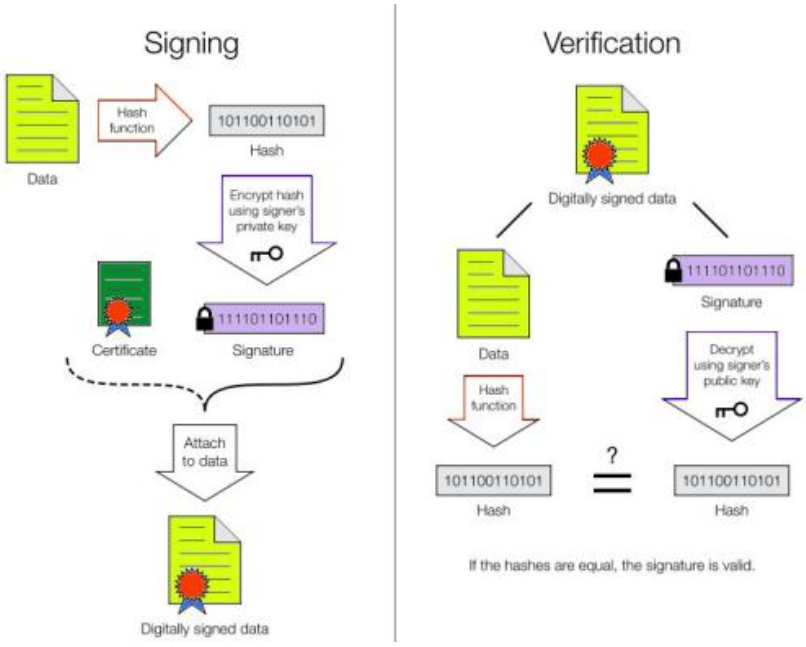
cifragem, seguida de uma decifragem e novamente uma cifragem (EDE). Podemos também usar a técnica do branqueamento, que faz uso de duas chaves, mas gera confusão.

FUNÇÕES DE SÍNTESE – Usadas para gerar e validar assinaturas digitais, MACs, e verificar a integridade de uma mensagem. Produz um valor de tamanho fixo a partir de um texto de comprimento variável. Usam o método de Merkle-Damgard, que aplica um padding criando um bloco de tamanho fixo. De seguida uma função hash parte esse bloco em vários de tamanho definido, e processa cada um individualmente com uma função de compressão (blocos de tamanho igual à entrada dão blocos de tamanho igual à saída), juntando de cada vez o bloco originado no último processamento, com o atual. Para garantir a segurança, cada mensagem é juntada a um padding que codifica o tamanho original da mensagem. As funções de sintese apresentam três propriedades fundamentais: 1) dado uma hash, é inviável que se encontre a sua mensagem original; 2) dado uma mensagem e a sua hash, é inviável que se encontre outra mensagem com a mesma hash; 3) é inviável que se encontre duas mensagens com a mesma hash (paradoxo do aniversário – dadas várias pessoas, é inviável que duas delas tenham a mesma data de aniversário). Tem como principais algoritmos o MD5 de 128 bits, não seguro uma vez que é facil detetar colisões, e o SHA-1 (Secure Hash Algorithm) de 160 bits, ainda sem colisões detetadas.

MAC (Message Authentication Codes) – Pedaço de informação usado para autenticar uma mensagem por forma a garantir integridade (alteração da informação) e autenticação (validade do emissor). O algoritmo pode ser uma hash que recebe como entrada uma chave secreta e uma mensagem para ser autenticada, originando um MAC (HMAC, por exemplo, com MD5 de 128 bits ou SHA-1 de 160 bits). É assim garantido que apenas os possuidores da chave possam validar a mensagem, implicando que haja uma troca de chaves numa primeira fase (chaves simétricas). É aqui que reside a principal diferença em relação a assinaturas digitais. Pode ser feita com resumos também. A cifragem de um resumo pode ser feita, por exemplo, pelo método de cifra por bloco simétrico (CBC-MAC). Pode-se também usar um método de cifragem com realimentação e propagação de erro (DES-MAC). Como são usadas chaves simétricas, qualquer pessoa pode ao verificar o MAC gerar outros MACs para outras mensagens. Dessa forma não permite a não repudiação.



ASSINATURAS DIGITAIS – Têm como principal objetivo autenticar o conteudo de um documento, garantindo a sua integridade e a identidade do autor (não é possivel ao autor original negar o documento). Faz uso de cifragem assimétrica e funções de sintese. Possui algoritmos para assinatura e verificação da assinatura. A mensagem a enviar vai passar por uma hash gerando um numero. Esse número vai ser cifrado com a chave privada do emissor gerando uma assinatura. Esta assinatura vai ser anexada juntamente com um certificado de chave pública de autenticidade à mensagem a enviar. Na fase de verificação no receptor, este vai extrair a assinatura e decifra-la com a chave publica do certificado do emissor. Vair passar a mensagem pelo hash gerando um número que vai confirmar com o numero gerado pela decifragem para confirmar se é válida a assinatura. Normalmente, é preferivel assinar hashs de funções de resumo, do que o documento original por questões de: eficiência (a assinatura é mais pequena, poupando tempo), compatibilidade (normalmente as mensagens são strings, mas as assinaturas podem ter vários formatos, e graças à função de sintese podemos escolher o formato a converter) e integridade (sem a função de sintese, o texto original corre o risco de ser repartido em vários blocos para efeitos de assinatura e o receptor ao receber os vários blocos pode não saber a ordem deles ou se estão todos).



ASSINATURAS CEGAS – Método de autenticação onde quem vai assinar não conhece o conteúdo da mensagem que vai assinar. É importante para garantir o anonimato do dono da mensagem, fornecendo ainda funcionalidades extras.

GESTÃO DE CHAVES ASSIMÉTRICAS – Tem como principais objetivos:

- **Geração de chaves assimétricas** – Quando e como devem ser geradas. Devem seguir alguns principios como uma boa geração aleatórioa (usar o gerador de Bernoulli, por exemplo), ter chaves eficientes mas sem comprometer segurança (chaves de poucos bits, que permitam operações rápidas) ou auto-geração de chaves privadas (por forma a maximizar privacidade);
- **Exploração de chaves privadas** – Como mantê-las privadas. Uma chave privada deve sempre representar um sujeito, e o seu compremetimento deve ser sempre minimo! É possivel criar cópias de chaves em alguns casos. O caminho de acesso à chave deve ser controlado, por exemplo, com password ou PIN. Podemos também garantir segurança à chave privada mantendo-a num ambiente já por si seguro (criar um token cifrado, onde apenas ele pode criar e trabalhar as chaves pública e privada);
- **Distribuição de chaves públicas** – Como distribui-las corretamente. Se pretendermos enviar informação para alguém, devemos distribuir as chaves manualmente, ou usando uma forma secreta partilhada ou uma rede ad-hoc com certificados digitais. Se pretendermos enviar assinaturas digitais, então as chaves devem ser distribuidas por rede ad-hoc. Os meios de partilha de chaves devem ser sempre de confiança para um dos lados envolvido.
- **Tempo de vida de uma chave** – Até quando podem ser usadas. Todos os pares de chaves devem ter um tempo limite de vida, uma vez que as chaves privadas podem ser descobertas ou podem-se perder, e isso pode trazer problemas. No entanto, os certificados podem ser livremente copiados e distribuidos, e dessa foram é impossivel saber quem tem os certificados, por forma a serem eliminados. A solução é então atribuir um periodo de validade a cada certificado. Após a validade expirar, os certificados passam a estar numa lista de certificados anulados (CRL).

CERTIFICADOS DIGITAIS DE CHAVES PÚBLICAS – São documentos públicos seguros, assinados por Autoridades de Certificação (CA) que ligam uma entidade a uma chave pública. Podem ser usados para distribuição de chaves públicas de forma segura. Por exemplo, se eu receber um certificado, em que X usa a chave Y, e este certificado está assinado por uma chave pública de uma CA, se essa chave pública for confiável, e a assinatura por parte do certificado estiver correta, então é confiável dizer que a chave Y pertence à entidade X. Usam como estruturas principais o X.509v3 e o PKCS (Public Key Cryptography Standard).

AUTORIDADES DE CERTIFICAÇÃO – Organizações responsáveis por gerir certificados de chaves públicas. Definem politicas e mecanismos que envolvem a emissão, revogação e distribuição de certificados bem como emissão e distribuição das chaves privadas correspondentes. Uma CA é de confiança se possuir uma chave pública de confiança ou for dado como de confiança por outra CA (através de um certificado), criando assim hierarquias de certificação. Estes CA's de confiança normalmente produzem os seus próprios certificados e distribuem-nos manualmente (por browser).

HIERARQUIA DE CERTIFICAÇÃO – Apresenta dois modelos especificos:

- **PEM (Privacy Enhanced Mail) –** Distribuição de certificados para segurança de email através de cifragens com chaves públicas. A hierarquia tinha apenas um root (IPRA), seguido de vários PCA (Policy Certificate Authorities) e vários CA's. Este modelo nunca foi implementado, porque verificou-se que havia muitas CA's que não tinham como root um PCA, e que havia hierarquias em que as CA roots eram independentes.
- **PGP (Pretty Good Privacy) –** Programa que se apresenta como uma rede de confiança, sem autoridades de confiança que se destaquem. Isto é, cada indivíduo pode certificar uma chave pública (um certificado) e publicar a assinatura para outros. Aqui a confiança nestes modos pode ser feita de duas maneiras: confiar nas chaves por nós conhecidas (usar meios necessários para saber se são de confiança), ou confiar no comportamento de cada certificado (ao usar o certificado, sabemos qual é o objetivo dele, e dessa forma podemos verificar se está a fazer o devido ou não). Por exemplo, se A confia em B, e B certifica C, então A pode confiar em C.

CERTIFICADO RAÍZ – Certificado que foi auto-assinado pelo próprio emissor do certificado. Quando temos CA's que assinam certificados de outras CA's, vai haver sempre uma CA raíz, cujo certificado não pode ser assinado por ninguém superior. Dessa forma, essa CA assina o próprio certificado (self-signed certificate). Numa cadeia de confiança, como não há uma CA central, os certificados são auto-assinados pelo próprio emissor.

CRL (CERTIFICATE REVOCATION LISTS) – Listas que contêm todos os certificados anulados, e a razão pela qual foram anulados. Devem ser consultadas regularmente por identidades que tenham certificados. A anulação de um certificado pode ter como base questões de segurança, como por exemplo, indicar que um CA usou indevidamente um certificado, ou caso uma chave privada tenha sido comprometida, ou quando uma pessoa tenta usar o seu certificado para documentos que na verdade são falsos. Em termos de criação e distribuição das CRL's, temos o método institucional e o método ad-hoc. No institucional, cada Autoridade de Certificação mantém e distribui a sua própria CRL. Numa hierarquia, as CA's trocam CRL's entre si para facilitar o anulamento de certificados. No método ad-hoc, cada possuidor de uma chave anulada deve criar e distribuir um certificado de anulamento assinado pela chave privada anulada.

OCSP (ONLINE CERTIFICATE STATUS PROTOCOL) – Protocolo usado para obter estados de anulamento de certificados X.509. É uma alternativa às CRL, em especifico devido aos problemas de usar CRL em PKI's. Em comparação com as CRL's, como as OCSP têm menos informação, então são mais rápidas a responder sobre o estado de anulamento de um certificado. Numa CRL uma pessoa veria o seu estado tornado público, isso no OCSP não acontece, e o OCSP pode informar um host que determinado host usou um determinado certificado numa determinada altura.

PKI (Public Key Infrastructure) – Infraestrutura (hardware, software, pessoas, politicas, planos) cujo objetivo é fazer um bom uso de chaves assimétricas e certificados de chaves públicas. Para esse bom uso é necessário:

- Criação de um par de chaves assimétricas para cada entidade envolvida (definição de politicas de criação e troca de chave);
- Criação e distribuição de certificados de chave pública (definição dos atributos do certificado e politicas envolvidas);
- Definição e uso de cadeias de certificação (hierarquia de certificação, certificados de outras CA's);

- Atualização publicação e consulta de CRLs (politicas para anular certificados);
- Uso de estruturas de informação e protocolos que permitam a cooperação entre componentes, serviços e pessoas.

Uma PKI define relações de confiança de duas formas diferentes: emitindo certificados para chaves públicas de outras CA's que estejam abaixo de si na hierarquia ou não relacionados entre si na hierarquia, ou requerindo o certificado da chave pública so seu root, caso esteja acima na hierarquia, ou não relacionados entre si na hierarquia. Uma relação de confiança consegue-se então de forma hierarquica, de forma cruzada (A certifica B, e vice-versa), ou de forma ad-hoc (criando uma malha de confiança). O X.509 é um exemplo de uma PKI.

PKCS #11 (Public Key Cryptography Standard) – Define uma plataforma API chamada "Cryptoki", para tokens de criptografia como o HSM (Hardware Security Modules) e Smart Cards. Esta API define os objetos mais usados em criptografia como chaves RSA, certificados X.509, Triplo DES, bem como todas as funções necessárias para gerar, modificar ou apagar esses objetos.

AUTENTICAÇÃO – Provar que uma entidade é quem ela afirma ser. É usada para confirmar e autorizar uma entidade, e dessa forma garantir-lhe acesso a politicas e mecansimos. A autenticação é guiada por cinco principios:

- Confiabilidade – Quão bom é a provar a sua entidade, e o quão dificil é rejeita-la;
- Secretismo – Não divulgação de credenciais secretas usadas por utilizadores legitimos;
- Robustez – Prevenir ataques à troca de informação de autenticação;
- Simplicidade – A autenticação deve ser o mais simples possivel evitando que entidades usem atalhos;
- Lidar com vulnerabilidades feitas por pessoas – Tendência natural para simplificar ou usar atalhos.

Existem duas formas principais de autenticação:

- **Direta** - Providenciar credenciais e esperar pelo veredicto. Pode ser feita de várias formas:
    - **Password memorizada** – A password é validade de acordo com um valor guardado pela pessoa. Esse valor guardado pode ser concebido de várias formas (transformado por uma função unidireccional, em UNIX uma hash DES com salt, em Windows com uma função de sintese, em Linux com uma função de sintese MD5). Este método tem como grande vantagem a simplicidade, tem como grande problemas o uso de passwords fracas (permite ataques de dicionário), e a transmissão destas por canais inseguros (facilmente alguém consegue interceptar a password).
    - **Biométrica** – A autenticação é feita com partes do corpo de uma pessoa (impressões digitais, iris do olho, formas da cara, timbre da voz, escrita manual, etc) que são depois comparadas com registos gravados pela mesma pessoa. Tem como grandes vantagens o não uso de passwords e o facto de as pessoas não terem de se relembrar de nada. No entanto tem como grandes problemas o facto de ainda estar em desenvolvimento este método, e podem ser facilmente ultrapassadas em alguns casos, pode ser arriscado para algumas pessoas, não permite partilha entre pessoas e não é facil o desenvolvimento remoto deste método.
    - **One-time password –** Passwords que apenas podem ser usadas uma vez. Tem como grande vantagem caso a password seja descoberta, não poderá ser novamente usada. Tem como grande problema, a constante sincronização entre quem haja no processo, por forma a estar constantemente a informar das novas passwords, e como estão sempre a ser renovadas, a pessoa pode não se recordar de todas imediatamente, o que exije que a anote em algum lado. Um exemplo que aplica esta método é o RSA SecurID, que tem um token que gera um numero aleatório por minuto e combina-o com o ID da pessoa gerando a password.
- **Desafio-resposta** – O autenticador providencia um desafio, e quem vai ser autenticado gera uma resposta com base no desafio recebido e nas suas credenciais de autenticação, e envia-a para o autenticador, espera-se depois pelo veredicto (o autenticador vai gerar um resultado de forma semelhante, e verifica se é semelhante à resposta do autenticado). A resposta do autenticado é normalmente um valor computado em

resposta a um valor proposto por um desafio, ou pode simplesmente ser uma password. Tem como grande vantagem o facto de não expor as credenciais de autenticação. Tem como principais problemas a necessidade de haver hardware e software suficiente para computar uma resposta, o facto de o autenticador ter de usar provavelmente informação secreta, e pode ocorrer ataques de dicionário contra registos desafio-resposta guardados. Este método pode ser incorporado de várias formas:

- **Smartcards** – O smartcard é usado para autenticar a pessoa. Para isso, faz-se uso da chave privada existente no cartão e do PIN para se aceder a essa chave. Quem vai autenticar a pessoa, deverá ter conhecimento da chave pública correspondente. Aplicando o método de desafio-resposta, o autenticador vai criar um desafio aleatório, e o detentor do smartcard vai cifrar esse desafio com a sua chave privada, envia a resposta, e o autenticador decifra a resposta com a chave pública correspondente, e caso o resultado seja igual ao do desafio, a autenticação é bem sucedida.
- **Password** memorizada – Para autenticar, a password é conhecida pela pessoa, e o autenticador vai conhecer uma transformação dessa password. Ele vai gerar um desafio aleatório, a pessoa ao receber vai computar uma transformação unidirecional do desafio e da password (Result = h(challenge, password)), e o autenticador vai fazer exactamente o mesmo. Caso o resultado seja igual à resposta, a autenticação é bem sucedida. Exemplos deste método:
  - **CHAP (Challenge-response Authentication Protocolo) –** Usado em point-to point, com autenticação unidirectional (autenticador não é autenticado), e garante melhor segurança que o PAP. O autenticador manda um desafio para a pessoa (com base no ID dela). A pessoa ao receber o desafio vai gerar uma resposta com base num valor calculado por uma hash (MD5, por exemplo) sobre o desafio e a password. O autenticador vai receber a resposta, e com base no seu cálculo da hash vai verificar se os valores são iguais. Caso sejam, então a autenticação é bem sucedida. O autenticador pode pedir uma reautenticação a qualquer altura.
  - **MS-CHAP (Microsoft CHAP) –** Neste protocolo realiza-se a autenticação tanto do autenticador como da pessoa (confirmando ao utilizador que está a lidar com um servidor que conhece a sua password, confirmando também que ele é fiável), e permite a atualização de passwords. O cliente ao receber o desafio vai gerar a resposta com base numa hash. Falta o da versão 2, não sei o que faz.
  - **S/Key –** Este método usa a one-time password (OTP). O processo vai começar com uma password, gerada ou pelo cliente ou pelo autenticador, e um valor base aleatório, que vai ser comunicado ao cliente. O cliente vai aplicar uma função hash x vezes à password e ao valor base, produzindo uma cadeia de x passwords (x é então um indice). O autenticador vai conhecer então os valores da primeira password gerados, com base no valor base e no primeiro indice. A primeira password nunca vai ser usada pelo cliente para autenticação, mas o autenticador vai usa-la para verificação. O cliente vai então enviar para o servidor a segunda password (indice x-1) e este vai computar a password. Caso dê o valor da primeira password gravada, então a autenticação é validada, e vai guardar a OPT de indice x-1 como referência. O processo repete-se, diminuindo o indice da OPT no cliente e no servidor.
  - **Shared-Key –** Em vez de uma password, usa uma chave partilhada, o que torna este método mais robusto contra ataques de dicionário, mas requer um token para guardar a chave e um canal partilhado seguro para partilhar a chave. Exemplo deste método é o GSM.

PAP (PPP Authentication Protocol) - Método simples, onde apenas se usa uma password. Este método é inseguro, uma vez que a password é transmitida por código ASCII não encriptado.

GSM (Global System for Mobile Communications) – Controla três tipos de serviços: serviços de suporte (para comunicar com ISDN e PSDN), serviços de telefone (transmissões de voz de alta qualidade, mensagens até 160 caractéres e serviços de fax. Estes serviços foram mais tarde reforçados com a WAP (Wireless Application Protocol)

e o GPRS (General Packet Radio Service), permitindo enviar pacotes ainda maiores. A rede GSM é repartida em três componentes principais:

- **BST (Base Station Subsystem) –** Composto por vários BTC's (Base Transceiver Stations), que tratam de toda a recepção e comunicação para os nossos telemóveis. Estes BTC's suportam vários alcances e capacidades, e juntos formam uma estrutura que cobrem áreas grandes e espalhadas em zonas rurais, pequenas e apertadas em áreas urbanas. Os BTC's vão comunicar com uma BSC, que vai controlar a possibilidade de nos deslocarmos entre várias áreas sem que a chamada venha abaixo (handover), possibilitar o envio de sinais para telefones especificos responderem (paging), bem como a comunicação com o MSC.
- **MSC (Mobile Switching Centre) –** É a espinha dorsal do GSM. Permite controlar a handover entre vários BSC's e implementa e controla também serviços adicionais sobre os componentes ditos acima. Nesta componente temos ainda o HLR (Home Location Register) e o VLR (Visitor Location Register), que funcionam em conjunto como uma base de dados de informação dos utilizadores na rede e imediações. O HLR guarda os registos permanentemente, e o VLR guarda-os dinamicamente, permitindo poupança de tempo no acesso ao HLR.
- **Operation Subsystem –** Contem um centro de autenticação (AUC) e equipamento de identificação de registos (EIR), usados para segurança.

É importante a segurança para garantir o anonimato e privacidade na rede de um utilizador quando este efetua uma chamada, e para garantir que a cobrança é feita ao utilizador correto e para garantir que não haja interferência de chamadas entre utilizadores. Para realizar a segurança, é necessário autenticar o utilizador. A rede vai enviar um desafio de 128 bits aleatório para o telefone do utilizador. De seguida o cartão SIM do telefone vai usar o algoritmo A3 o Ki (numero de identificação do SIM) e o valor aleatório do desafio para computar uma SRES (Signed Response) e envia-o para a estação base. Se o valor for igual ao computado na estação, então está tudo válido. De seguida o cartão SIM vai usar o algoritmo A8, o Ki e o valor original do desafio para computar uma chave de sessão (Kc) e envia-a para a estação. Esta chave vai ser usada com o algoritmo A5 para cifrar toda a comunicação feita.

AUTENTICAÇÃO DO HOST – Pode ser feita por nome ou endereço (nome DNS, endereço IP, endereço MAC ou outro), mas é uma forma bastante fraca de autenticação, uma vez que não usa cifragem (NFS e TCP wrappers usam este método), ou pode ser feita com chaves assimétricas, chaves partilhadas entre hosts mais precisamente.

AUTENTICAÇÃO DO SERVIDOR – A autenticação do servidor pode ser feita indiretamente pelo host, ou pode ser feita através de credenciais (chaves assimétricas partilhadas).

SSH (SECURE SHELL) – Procolo de rede para comunicação segura ou execução de comandos entre computadores de duas redes (cliente-servidor), ligadas por um canal seguro sobre uma rede insegura. Faz uso de chaves assimétricas para autenticar o servidor, e este distribui chaves públicas pelos hosts que façam conexão com quem possua a chave privada. Dessa forma o SSH só tem de verificar se o utilizador possui a chave pública corresponde à sua privada, e dessa forma faz a conexão. A autenticação do cliente é feita por username e password, ou username e chave privada dele que devem ser enviadas para o servidor primeiro que a chave pública.

ATAQUE POR DICIONÁRIO – Técnica usada para descobrir a chave de decifragem ou password através do teste com várias possibilidades. Temos uma lista com palavras, como se de um dicionário se tratasse, e essa lista vai ser corrida e testada exaustivamente com o objetivo de acertar na palavra que decifra/abre o conteúdo em questão. Normalmente este método tem sempre sucesso, porque uma pessoa tende a colocar como passwords palavras conhecidas e pequenas que constem em dicionários. É então uma questão de tempo até a pesquisa encontrar essa palavra.

PAM (Pluggable Authentication Modules) – Biblioteca que fornece uma API que permite a integração de multiplos esquemas de autenticação num único host ou em aplicações, sem necessitar de as compilar, dando a possibilidade ao utilizador de escolher qual usar, evitando assim mecanismos de autenticação complexos e garantindo mais segurança (uma vez que o desenho da password está ao critério do utilizador). Permite alterar politicas de autenticação simplesmente editando ficheiros configuráveis e fornece interfaces que permitem interação com linha de comandos, smartcards, leitores biométricos, etc. No geral, a API do PAM vai dividir a autenticação em quatro pequenos grupos:

- **Autenticação –** Verificação da identidade;
- **Gestão de Contas –** Aplicação de politicas de acesso com base nas propriedades da conta (como um role);
- **Gestão de Passwords –** Controlo do processo de modificação de uma password;
- **Gestão de Sessões –** Verificação e aplicação de parâmetros de uma sessão (máximo de memória, etc).

O PAM fornece uma API tradicional com várias funções disponiveis e códigos variáveis (PAM_SUCCESS, PAM_AUTH_ERR, etc) e a sua biblioteca é carregada dinamicamente (/lib/security/pam_*.so). Temos também um ficheiro de configuração (um por cliente) que especifica como as acções devem ser aplicadas (que mecanismos e parâmetros usar). Estas configurações têm sempre como base os recursos disponiveis em cada módulo.

AUTENTICAÇÃO LINUX – A autenticação no Linux faz-se por meio de um username e de uma password. Após introduzidos, o username vai ser usado para procurar o respetivo UID no ficheiro /etc/passwd e o respetivo GID no ficheiro /etc/group. O username presente no ficheiro está associado com uma password cifrada, presente no ficheiro /etc/shadow. Essa password cifrada vai ser comparada com a password fornecida no login (transformada usando uma hash MD5), e caso sejam iguais, o login é validado e é criado um processo com base nos parâmetros do utilizador loggado. Os ficheiro /etc/passwd e /etc/group podem ser lidos por qualquer pessoa, mas o ficheiro /etc/shadow só pode ser lido por root (proteção contra ataques de dicionário).

CONTROLO DE ACESSO – Politicas e mecanismos que controlam o acesso de um sujeito (processos, computadores, redes) a um objecto (informação guardada, tempos de CPU, memória, redes). Este controlo de acesso normalmente requer autenticação, autorização e responsabilidade. Essas politicas e mecanismos de controlo vão atribuir privilégios (autorização para executar algo) ao sujeito conforme o tipo de tarefa que eles vão fazer. Esses privilégios devem ser minimos, os suficientes para se puder executar o acesso desejado (se o sujeito tivesse mais privilégios que os requeridos, isso podia levar a vulnerabilidades, erros, circulação de informação não necessária). Existem vários modelos de controlo de acesso:

- **Matriz –** Temos uma matriz com os sujeitos (linhas) e os objectos (colunas) e o devido acesso;
- **ACL's (Access Control List) –** É uma coluna da matriz, que contém os acessos (positivos ou negativos) para objectos especificos.
- **Mecanismos baseados na capacidade –** É uma linha da matriz, que contem os acessos para os sujeitos especificos.

	O1	O2	..	Om-1	Om
S1		Access rights			
S2					
..					
Sn-1					
Sn					

Temos também vários tipos de controlo de acesso:

- **MAC (Mandatory Access Control) –** Politicas implementadas estáticamente. Cada sujeito e objecto têm atributos/regras de segurança. Sempre que um sujeito tenta aceder a um objecto, o kernel do sistema operativo examina as regras de segurança e decide se o acesso pode ocorrer;

- **DAC (Discretionary Access Control) –** Neste tipo de acesso, existe restrição de acesso, ou seja, apenas sujeitos com permissões adequadas podem aceder aos objectos (normalmente apenas os donos dos objectos ou administradores do sistema);
- **RBAC (Role Based Access Control) –** Método de control de acesso que atribui roles aos sujeitos de forma dinâmica (o acesso é dado conforme o role). Pode implementar tanto o MAC como o DAC. Apresenta três regras fundamentais:
    - **Designação de roles –** Para poder realizar uma operação, um sujeito tem de ter um role;
    - **Autorização de roles –** Para poder realizar uma operação, o controlo de acesso tem de validar o role do sujeito;
    - **Autorização de operações –** Para poder realizar uma operação, o controlo de acesso tem de validar essa operação para o role em questão, e o sujeito tem de passar em todas as restrições que existam.

  Roles vs Grupos – Um role é um conjunto de permissões, permissões essas que são aplicadas aos sujeitos. Um sujeito só pode ter um role de cada vez. Um grupo é um conjunto de sujeitos, e podemos indicar permissões e roles tanto a grupos de sujeitos como a apenas um sujeito. Um sujeito pode pertencer a vários grupos ao mesmo tempo.

- **CBAC (Context-Based Access Control) –** Tipo de acesso baseado no historial de acesso de um sujeito (exemplos disso o objecto locking e o filtro de pacotes numa firewall (o tráfego é filtrado, e o acesso é garantido conforme o tráfego feito)).

SEPARAÇÃO DE DEVERES – Técnica que consiste em termos vários sujeitos a realizar uma única tarefa, permitindo evitar fraudes e erros (implementada normalmente com RBAC). Podemos dizer que usa o velho ditado de "quantos mais, melhor", os erros que uma simples pessoa podia obter, são evitados colocando várias pessoas a tratar do mesmo assunto.

MODELOS DE FLUXO DA INFORMAÇÃO – A autorização de um determinado sujeito pode ser feita de duas formas:

- **Por fluxo de informação –** Considera-se a fonte e destino da informação, evitando-se informação perigosa;
- **Com base nos atributos da fonte e destino –** Só há fluxo de dados caso a fonte e destino apresentem atributos determinada segurança;

SEGURANÇA MULTINIVEL – Existem sempre vários niveis de segurança, organizados tanto hierarquicamente como entrelaçados. Um role costuma estar sempre associado a um nivel de segurança, e um sujeito com esse role só pode atuar nesse nivel de segurança. Se o sujeito pretender aceder a outro nivel de segurança, esse acesso tem sempre de ser controlado. Os niveis de segurança estão divididos conforme as categorias existentes (autónomo, militar, civil). Os niveis de segurança na categoria civil estão divididos em quatro: publico sensivel, proprietário e restrito. A nivel militar e de organizações internacionais estão tipicamente divididos em cinco: inclassificavél, restrito, confidencial, secreto e muito secreto. Podemos ter um objecto que pertença a várias categorias, e dessa forma pode ter vários niveis de segurança conforme a categoria. Normalmente o nivel de segurança de um objecto constrói-se usando uma etiqueta (nivel de segurança, categoria).

MODELO BELL-LA PADULA – Modelo baseado numa máquina de estados que força o uso de controlo de acesso com base em objectos com etiquetas e sujeitos com determinadas autorizações. Foi primeiramente inventando para estabeler controlo de acesso em bases militares, abordando a confidencialidade de dados e o acesso a informação confidencial. Usado o modelo da máquina de estados, em cada estado são considerados o sujeito, o objecto a aceder, uma matriz de acessos e a informação de acesso actual. Existe um conjunto de politicas de segurança, e esse estado só é considerado seguro caso os modos de acesso permitidos entre o sujeito (roles) e o objecto (etiquetas) estejam de acordo com essas politicas (por exemplo, role de tenente tem acesso a objectos com etiquetas do nivel secreto). Este modelo define três regras fundamentais, duas MAC e uma DAC cujas propriedades são:

- **Propriedade da segurança simples (no read up, MAC) –** Um sujeito S com um nivel de segurança L pode ler um objecto O cujo nivel de segurança seja inferior ou igual a L, ou seja, L(S) >= L(O), mas não pode ler um objecto cujo nivel de segurança seja acima de L, ou seja, L(O) > L(S).
- **Propriedade estrela (no write down, Confinamento, MAC) –** Um sujeito S com um nível de segurança L não deve escrever num objecto O com um nível de segurança inferior a L, ou seja, L(S) > L(O);
- **Propriedade de Segurança Discricionária (DAC) –** Usa uma matriz de acesso para especificar o acesso;

MODELO DE INTEGRIDADE DE BIBA – Modelo que vem complementar o modelo Bell-La Padula, visto que apenas visa a confidencialidade dos dados. Com o modelo de Biba é também garantida a integridade dos dados. Em vez de vários níveis de segurança, temos vários níveis de integridade, que visam impedir a modificação de dados por pessoas não autorizadas, impedir a modificação de dados não autorizada por parte de pessoas autorizadas, e manter a consistência externa (dados que reflitam o mundo real). Este modelo assenta em duas regras fundamentais, inversas às do modelo Bell-La Padula:

- **Axioma da integridade simples (no read down) –** Um sujeito S com um dado nivel de integridade I, não deve ler um objecto O com um nivel de integridade inferior ao seu, ou seja, I(S) > I(O);
- **Axioma da integridade estrela (no write up) –** Um sujeito S com um dado nivel de integridade I não deve escrever um objecto O com um nível de segurança acima do seu, ou seja, I(S) <= (O);

MODELO DE INTEGRIDADE DE CLARK-WILSON – Modelo que visa o controlo da integridade da informação. Para isso define uma politica de integridade constituida por regras de aplicação (E) e regras de certificação (C). Este modelo assenta nas seguintes definições:

- **Transação –** Ideia centra do modelo, uma transação é um conjunto de operações;
- **CDI (Constrained Data Item) –** Considerados os itens chave deste modelo;
- **UDI (Unconstrained Data Item) –** Dados introduzidos por utilizadores;
- **IVP (Integrity Verification Procedure) –** Verifica que todas as CDI's do sistema são válidas;
- **TP (Transformation Procedure) –** É uma transação que recebe como entrada uma CDI ou UDI e produz uma CDI segura e com integridade. Apenas a TP pode alterar as CDI's.

Com base nas regras de aplicação (E) e nas regras de certificação (C), o modelo de Clark-Wilson é constituido por 9 regras fundamentais que asseguram a integridade:

- **Regras Básicas**
  - **C1 –** Sempre que um IVP é executado, ele tem de garantir que todos os CDI's são válidos;
  - **C2 –** Para um conjunto de CDI's, uma TP deve transformar essas CDI's de um estado válido para outro estado válido;
  - **E1 –** O sistema deve ter uma lista de relações certificadas e garantir que apenas as TP's certificadas para correr um CDI mudam esse CDI;
- **Separação de Deveres**
  - **E2 -** O sistema deve associar um utilizador com cada TP e um conjunto de CDI's;
  - **C3 –** Um tripleto (utilizador, TP, CDI's) deve cumprir os requisitos da separação de deveres;
- **Recolha de Informação**
  - **E3 –** O sistema deve autenticar todos os utilizadores que tentem usar TP's;
- **Auditoria**
  - **C4 –** Todas as TP's devem registar as suas operações num log para possiveis recuperações;
- **Processamento UDI**
  - **C5 –** Uma TP que receba uma UDI só pode efectuar transformações sobre valores válidos da UDI. Dessa forma a TP pode negar-se a converter a UDI;
- **Restrições de certificação**
  - **E4 –** Apenas o certificador da TP pode alterar a lista de entidades associada a esa TP;

SEGURANÇA EM SISTEMAS OPERATIVOS – Podemos dividir um sistema operativo (SO) em duas camadas, uma onde actuam os utilizadores, ao nível das aplicações, sem acessos privilegiados, e outra ao nível do kernel, onde actuam os supervisores, com acessos privilegiados. O kernel (núcleo) é o ponto central de um SO, fazendo uma ponte entre as aplicações e o processamento de dados feito ao nível do hardware. É ao nível do kernel que são aplicadas as politicas de segurança e são implementados os mecanismos de segurança de um SO. O kernel vai gerir um conjunto de entidades que vão formar o modelo computacional: identificadores de utilizadores, processos, memória virtual, ficheiros e sistemas de ficheiros, canais de comunicação e dispositivos físicos (armazenamento, interfaces de rede, interfaces humano-computador, interfaces serie ou I/O).

- **Identificadores de utilizadores –** Para o kernel, um utilizador é um número estabelecido durante o login (UID). Todas as atividades executadas num computador têm em função o UID, ou seja é ele que define as atividades que são permitidas ou não. Em Linux, todo o utilizador administrativo tem um UID 0, e em Windows, vários utilizadores podem ter privilégios de administrador (normalmente através de grupos de administradores), não havendo um UID definido. Um utilizador pode pertencer a um ou vários grupos, identificados por um GID. Dessa forma os privilégios que um utilizador tem são os seus, e os do seu grupo. Em Linux, todas as actividades são executadas em função de um conjunto de grupos.
- **Processos –** Define o contexto de uma actividade, tanto para questões de segurança como para outros aspectos. Para o contexto da segurança, deve-se ter sempre em conta a entidade (UID e GID), e os recursos em uso (ficheiros abertos, memória virtual reservada, tempo de cpu usado).

O Kernel é por si só um monitor de controlo de acesso, controlando todas as interações entre o modelo computacional e o hardware. Ele gere o controlo de acesso com base em mecanismos já estudados:

- **MAC (Mandatory Access Controls) –** O Kernel está cheio de politicas MAC, como por exemplo o facto de o Kernel correr em modo privilegiado, e as aplicações correrem em modo não privilegiado, separação de áreas de memória virtutal, limites de reserva de recurso, sinalizações entre processos, etc;
- **Protection with ACLs –** Pode ser MAC (não pode ser anulada) ou DAC (pode ser adaptada). Cada objecto tem uma ACL, que indica o que o sujeito pode fazer ou não, e essa ACL é verificada sempre que uma actividade tenta aceder a um objecto por ordem de um sujeito. Caso o acesso não for autorizado pela ACL, o acesso é negado. O Kernel é o responsável por aplicar proteção à base de ACL's. Em Linux, cada objecto tem uma ACL com 3 tipos de permissão (escrita, leitura e execução) e apenas o dono pode alterar a ACL (estrutura DAC). Em Windows, as ACL's aplicam-se a ficheiros NTFS, onde cada objecto tem uma ACL e um dono. A ACL garante 14 tipos de permissão (Execute, Read, Create, Delete, etc) a uma lista de sujeitos (UID ou GID).
- **Protection with Capabilities –** Pouco comum em Kernels. Exemplo disso são os "open file descriptors" (indicadores para aceder a ficheiros).

ELEVAÇÃO DE PRIVILÉGIOS – Acto de configuração que permite aumentar os privilégios de um dado utilizador, fazendo com que consiga aceder a recursos que normalmente não conseguia aceder. Este aumento de privilégios é feito graças ao mecanismo set-UID que permite que um simples utilizador aceda a ficheiros com o UID do dono ou do grupo em que está (que detêm mais privilégios, os necessários para aceder ao ficheiro). O conhecido sudo em Linux, é um set-UID com UID = 0, permitindo agir no sistema como root, com privilégios elevados.

REDUÇÃO DE PRIVILÉGIOS – Consiste em reduzir a visibilidade de um ficheiro de sistema. Para este efeito é usado o comando chroot, que aplicado a um directório, vai isolar todo o conteúdo desse directório e filhos, impedindo o acesso a conteúdo fora desse directório. Isto pode ter várias vantagens a nivel de testes e desenvolvimento (efectuamos testes no directório isolado, impedindo que os testes influenciem o sistema), instalação de software (todo o software é construido no directório isolado, impedindo a linkagem com o exterior e a sua alteração).

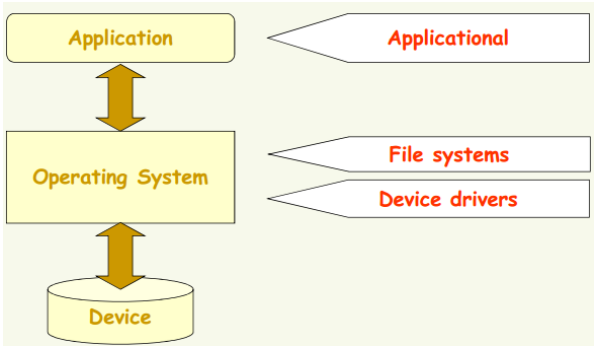
ANEIS DE EXECUÇÃO – Mecanismo de proteção hierárquico, que visa proteger dados e funcionalidades de falhas e comportamentos indevidos. Cada anel traduz um nivel de privilégio, sendo o anel mais interior o de maior privilégio. Isto vai impedir que aplicações ou processos que corrarm num dado anel (nivel de privilégio) interfiram ou mexam em recursos localizados noutros anéis. Para os anéis poderem comunicar entre si, usam-se portões (gates) especiais entre eles, que permitem comunicação controlada entre os anéis. Por exemplo, um browser que corre num nivel de privilégio mais exterior no anel, vai precisar de aceder a recursos de maior privilégio, ou seja, aneis mais interiores, e dessa forma, um gate é configurado para permitir o acesso.

EXECUÇÃO DE MÁQUINAS VIRTUAIS – É criado um anel de nível -1, abaixo do nivel 0, com privilégios ainda mais acrescidos (hipervisor). Uma máquina virtual é capaz de virtualizar hardware de vários kernel de nível 0.

ARMAZENAMENTO SEGURO DE FICHEIROS – A segurança no clássico sistema de ficheiros é muito limitada (tanto a nível fisico como lógico) e por vezes é mesmo inútil. Além disso o acesso distribuido vai criar muitos problemas de segurança. Uma solução para este problema é então cifrar o conteúdo dos ficheiros, possibilitando que sejam transmitidos mesmo em redes perigosas e guardados em dispositivos de armazenamento inseguros. No entanto, este método requer que os utilizadores que partilhem dados ou tentem aceder ao seu conteúdo possuam chaves de cifragem e decifragem e não as percam! Se tivessemos uma arquitetura ideal era seria teria os seguintes pontos:

- **Transparência na cifragem e decifragem –** ao nível da aplicação e da cache do sistema operativo;
- **Visibilidade de dados protegidos –** deve ser possível ver o que está protegido ou não;
- **Fácil partilha de dados cifrados -** por grupos de utilizadores;
- **Capacidade de decifragem sob circunstâncias especiais por pessoas autorizadas –** forçagem legal, proteção contra a perda de chaves privadas;

Em vez disso temos uma arquitetura da seguinte forma:



- **Nivel aplicacional –** A informação é cifrada/decifrada por aplicações autónomas (quase sem integração com outras aplicações, e é possivel ver o que é seguro ou não graças às extensões dos ficheiros). Enquanto a informação está em texto original, há vulnerabilidades. A cifragem pode ser feita por vários algoritmos e chaves aumentando a segurança, mas complicando a recuperação. A partilha de ficheiros envolve partilha de chaves. Exemplos disto o PGP e o AxCrypt.
- **Nivel Device Drivers –** A cifra/decifra neste nível garante transparência total para aplicação e para o OS, no entanto não existe visibilidade de dados protegidos. Não consegue distinguir os acesso de diferentes utilizadores. Exemplos disto são o PGPdisk e Secure Digital Cards;
- **Nível de Sistemas de ficheiros –** A integridade de um sistema de ficheiros deve ser preservada, isto é, alguns ficheiros não podem ser escondidos porque têm funções administrativas (tipos de objectos, datas, dimensão, etc...) outros podem ser escondidos (nomes de directorios, conteúdo de ficheiros). Temos então dois tipos de sistemas de ficheiros:
    - **CFS (Cryptographic File System) –** Usado em Linux, acenta sobre o sistema de ficheiros NFS e permite guardar os nossos ficheiros numa forma cifrada num directório normal. Tanto o nome como o conteúdo do ficheiro são cifrados usando duas chaves derivadas de uma password:

- A cifragem do nome é feita com ECB, onde vai ser concatenado com um valor de controlo de integridade;
- A cifragem do conteúdo é feita com o XOR de uma cifra contínua OFB (recebe uma das chaves)e uma cifra de blocos ECB (recebe a segunda chave). O valor do vector de inicialização usado vai ser alterado aleatóriamente por cada ficheiro usado e vai ser guardado num nó GID.

Fazendo uso de uma chave, podemos decifrar o conteudo do ficheiro para a janela onde estamos. Este processo é realizado graças a uma montagem loopback NFS, referida como "attach". O ficheiro vai ficar em cache, e todos os utilizadores com devidas permissões de acesso podem lê-lo. Caso modifiquemos algo no texto decifrado, as alterações vão-se reflectir no ficheiro cifrado. Ao terminarmos a sessão CFS estamos a fazer um "detach", fazendo desaparecer os ficheiros decfirados até à proxima vez que fizermos um attach a ele. Todas as operações de cifragem e decifragem são feitas por um servidor CFS, e geram-se chaves por cada montagem feita. Os pacotes circulam na rede sempre cifrados.

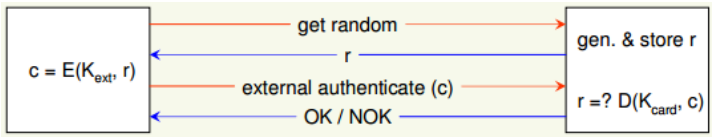
- **EFS (Encrypted File System) –** Usado em Windows, acenta sobre o sistema de ficheiros NTFS, e cifra o sistema de ficheiros na sua totalidade, mas de forma transparente, isto é, para o utilizador normal, este vê o seu sistema de ficheiros na sua totalidade, sem alterações. Um atacante apenas vai conseguir ver o disco físico, ou seja, não vai ver nada. Numa primeira parte, um utilizador vai receber do servidor EFS um conjunto de chaves assimétricas, baseadas na sua password. Cada ficheiro vai ser convertido usando uma chave simétrica chamada FEK (File Encryption Key, usando o algoritmo DESX). Para cada ficheiro, o EFS vai cifrar a FEK com a chave pública do utilizador (usando RSA), e vai guardar esses registos numa espécie de corrente associados ao ficheiro. Os ficheiros NTFS são formados por conjuntos de correntes. Este tipo de ficheiros contém alguns problemas, como o facto de as chaves assimétricas serem guardadas no disco e o facto de que os ficheiros são desencriptados por servidores, não havendo proteção na rede para ficheiros guardados remotamente.

SMARTCARD – Cartão com circuitos imbutidos, que fornecem identificação, autenticação, armazenamento de dados e processamento de aplicações. Tem como principais componentes um CPU (8 ou 16 bits), uma ROM (comunicação e algoritmos de cifragem), uma EEPROM (sistema de ficheiros com programas, aplicações, chaves e passwords), uma RAM (para transição de informação), contactos mecânicos (reset, power, clock, I/O) e segurança física (inviolável). A comunicação entre o cartão e o leitor de cartões é feita pelo APDU (Application Protocol Data Unit), que existe na forma de comando (enviado pelo leitor para o cartão) que tem um cabeçalho de 4 bytes (CLAsse da Instrução, INStrução, Parametro1, Parametro2) e até 255 bytes de informação, e resposta (enviado pelo cartão para o leitor) que contém uma word de 2 bytes e até 256 bytes de informação. A troca de informação pode ser feita byte a byte (mais lento) ou por blocos de bytes (mais rápido). O sistema de ficheiros de um smartcard contém:

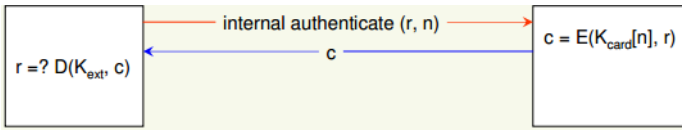
- **Identificação do ficheiro –** Nome ou número;
- **Tipo de ficheiro –** Master File (root), Elementary File (um ficheiro normal), Dedicated File (um directorio que contem EF's ou DF's);
- **Tipo de sistema de ficheiro –** Transparente (blocos de dados identificados por offset + length), registos fixos (indexados), registos variáveis (indexados), ciclico (ponteiro para ler e escrever, incremento em circulos);
- **Controlo de acesso –** Pode ser sem restrições, protegido (o ficheiro que acede ao APDU deve conter um MAC computado com uma chave partilhada entre o cartão e o leitor), ou autenticado externamente (o ficheiro que acede ao APDU só é permitido se o cartão já verificou a existência de uma chave partilhada com o leitor).

Em termos de cifragem, temos vários pontos que vão ser necessários proteger, cifrando-os:

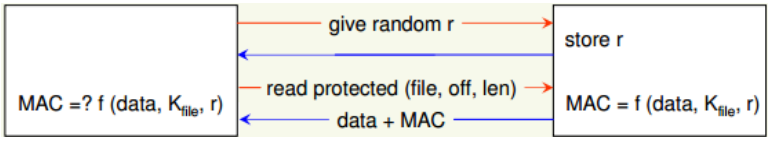
- **Autenticação externa –** Smartcard autentica o leitor com o protocolo desafio-resposta com um número aleatório. O processo é iniciado pelo leitor;



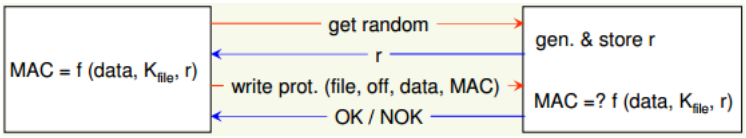
- **Autenticação interna –** O leitor autentica o cartão com o protocolo desafio-resposta com um número aleatório e o número de uma chave. Processo iniciado pelo leitor:



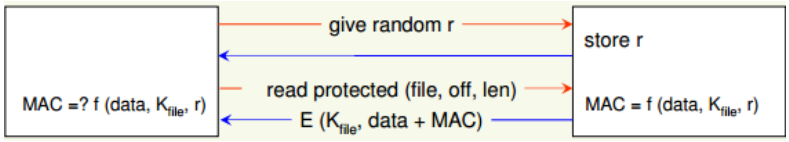
- **Troca de mensagens segura –** Proteger informação lida e escrita no cartão. Pode-se proteger por MAC ou MAC e cifragem de informação;
- **Leitura autenticada –** Leitor e cartão trocam um número aleatório, a informação a ler é requerida, e depois enviada com um MAC;



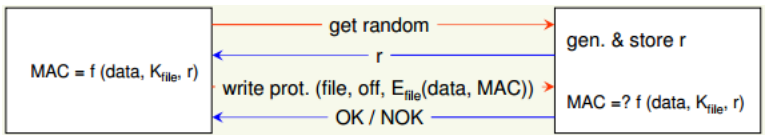
- **Escrita autenticada –** Leitor e cartão trocam um número aleatório, a informação a escrever é enviada com um MAC, e a confirmação é feita por um OK ou NOK;



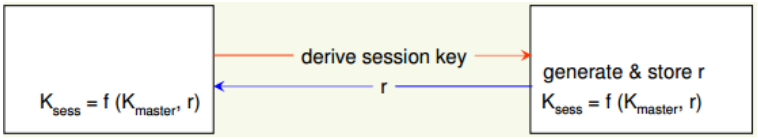
- **Leitura autenticada e confidencial –** Leitor e cartão trocam número aleatório, a informação a ler é requerida, e depois enviada cifrada com a chave e o MAC;



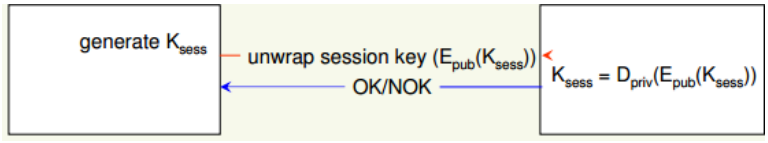
- **Escrita autenticada e confidencial –** Leitor e cartão trocam número aleatório, a informação a escrever é enviada cifrada com o MAC;



- **Derivação da chave de sessão –** A chave é requerida, e é enviado um numero aleatório que permite derivar a chave;



- **Upload da chave de sessão –**



O uso de chaves assimétricas num smartcard vai permitir a individualidade da autenticação, porque a um cartão vai estar atribuida uma unica chave privada, e o utilizador que usar o cartão vai ter uma identificação unica. Isto permite a devida identificação do utilizador perante o servidor quando for necessário enviar informação cifrada.

O uso do Cartão de Cidadão apresenta uma mais valia para validação de assinaturas digitais, uma vez que este possui uma chave privada que identifica inequivocamente o dono do cartão e dessa forma, o autor da assinatura.

JAVACARD – Smartcards que correm Java Applets através de JCRE (Java Card Runtime Environment), criando uma máquina virtual Java que permite usar funções da biblioteca Java Card Framework.

SEGURANÇA EM BASE DE DADOS – Uma BD apresenta inúmeras vantagens óbvias como o acesso partilhado, redundância, consistência e integridade de dados e acesso controlado. É necessário por isso garantir alguns requisitos de segurança como integridade fisica (falhas de energia, reconstrução por falhas), lógica e de elementos, auditabilidade (saber quem mexeu em algo), controlo de acesso, autenticação e disposição (possibilidade de aceder à BD e a todos os dados conforme os roles obviamente). É possivel que ocorram erros durante updates, e isso pode deixar a BD num estado incoerente. Para evitar isto, existem as transações, ou update de duas fases. Numa primeira fase A BD Management System (DBMS) prepara os recursos necessários, não alterando dados, e marca uma flag de commit em determinada posição. Numa segunda fase, são efectuadas as alterações à BD conforme o especificado na primeira fase, e o commit é feito com a BD num estado coerente. Podemos garantir consistência interna e redundância através da deteção de erros (bits de paridade, código de Hamming), ou usando os campos sombra que duplicam campos já existentes (necessita de espaço de armazenamento).

Um monitor é uma unidade DBMS responsável pela integridade da BD, verificando valores inseridos para assegurarem a validade deles conforme o tipo de campo, usando registos ou definindo restrições à BD. Temos três tipos de monitores: comparadores de alcance (verifica validade do alcance do campo), restridores de estado (impõe regras de integridade), restridores de transição (descrevem condições necessárias antes das aplicações serem aplicadas à BD).

Numa BD podemos usar o conceito de segurança multiniveis, colocando etiquetas nas queries com niveis de segurança (nivel de segurança da entidade responsável pela query). Isto vai impedir que as queries verifiquem valores de campos com niveis de segurança acima dos seus. É possivel termos campos que pertençam a vários niveis de segurança (poli-instanciação), contudo isto vai reduzir o nivel de precisão da BD uma vez que é validade da informação dependa da entidade que faz a query, e do seu nível de segurança. Existem várias maneiras de separar os niveis de segurança:

- **Particionar –** Diferentes niveis de segurança, BD's diferentes, e as queries são direccionadas para a BD certa. É facil de implementar, contundo pode haver redundância e cria-se problemas no acesso a campos de niveis de segurança diferentes;
- **Encriptação –** Campos são cifrados com chaves de segurança. Isto proporciona apenas uma BD com a mesma estrutura. No entanto é necessário fazer a decifra em cada query com o nivel de segurança adequado, tem de se garantir que a cifragem é feita por forma a não originar criptogramas iguais (passivel de ataques graças a estatisticas ou valores decifrados conhecidos), algo que se pode resolver com chaves ou vetores de inicialização diferentes para cada registo. Outro problema reside no facto de um valor cifrado não poder ser actualizado com outro cifrado, ou seja é necessário decifra-los primeiro.
- **Fecho de integridade –** Cada campo de dados é formado por três partes (dados, etiqueta de sensibilidade, checksum), e a etiqueta deve ser inalterável, unica e escondida. Este método permite o uso de um DBMS normal, e pode ser implementado graças a stored procedures de confiança. A unica desvantagem prende-se com o armazenamento do tripleto dados-etiqueta-checksum.

INFORMAÇÃO SENSIVEL – Informação que requer uma proteção extra, tanto para perda, mau uso, modificação ou acesso não autorizado. A informação é sensivel porque pode implicar muitos riscos como afectar a privacidade e bem-estar de individuos, pode afectar actividades de negócio ou relativas a segurança. Numa BD, a informação sensivel não deve estar como pública. A sensividade da informação de uma BD vai depender do objectivo da BD e na informação existente na BD, podendo ser alguns registos, todos os registos ou mesmo toda a BD (como exemplo, os registos médicos de todas as doenças detectadas é informação sensivel). Em termos de complexidade podemos considerar casos simples (nada é sensivel ou tudo é sensivel) ou casos complexos (alguns, mas não todos os elementos da BD são sensiveis). Existem vários factores que podem contribuir para a sensividade dos dados:

- **Inerentemente sensivel –** Só por si o valor pode revelar-se sensivel;
- **Fonte sensivel –** O valor pode revelar a sua origem;
- **Declarado sensivel –** O valor foi declarado sensivel;
- **Registo sensivel –** O registo foi declarado sensivel;
- **Sensivel em relação a outra informação –** Por si só, o valor nem é sensivel, mas comparado a outra informação, é sensivel.

A informação sensivel pode dar origem a vários tipos de informações:

- **Informação exacta –** O mais problemático é fornecido o exacto valor da informação sensivel;
- **Limites –** Com base na informação sensivel podemos estabelecer limites para outros dados;
- **Resultados negativos –** Obtendo um resultado negativo de um valor sensivel numa query, isto pode dar informação extra sobre alguns valores da query;
- **Existência –** A existência de informação sensivel pode ser por si só, informação sensivel;
- **Valor provável –** Sabendo o valor sensivel, é possivel cruzar queries e obter resultados prováveis.

Uma inferência é uma maneira de derivar informação sensivel a partir de informação não sensivel. Estes ataques podem ser de três tipos:

- **Directos –** Queries com uma mistura de regras que usem dados sensiveis e dados não sensiveis, tentando enganar o DBMS com os dados não sensiveis que não estão destinados a usar alguns registos particulares;
- **Indirectos –** Inferência de alguns valores com base em valores estatisticos computador por vários registos (contagens, somas, médias);
- **Rastreador –** A BD pode esconder alguns dados quando alguns registos formam a grande porção de dados revelados. Um rastreador pode enganar o DBMS usando queries diferentes que revelem dados, e combinando os resultados, o atacante pode obter a informação desejada.

SANDBOX (CAIXA DE AREIA) – Cria uma barreira à volta de um ambiente de execução Java, e as aplicaçõe a decorrer são executadas apenas nesses limites da caixa e impedidas de usar recursos fora da caixa. Tem de ser capaz de fornecer recursos de proteção remotos (pelo sistema remoto) e locais (pelo Gestor de Segurança Local).

SEGURANÇA EM MÁQUINA VIRTUAL JAVA – A Java Virtual Machine (JVM) é uma "caixa de areia" segura para executar programas Java implementados por um conjunto de classes do Java. Em termos de segurança, as suas politicas são fáceis de configurar, tal como o controlo de acesso, fornecendo ainda extensões de segurança para os programas em Java. Todo o código e de dados da JVM são protegidos com verificações estáticas e dinâmicas.

SEGURANÇA NO JRE (JAVA RUNTIME ENVIRONMENT) - Aplica os seguintes métodos:

- **Carrega as classes necessárias –** Usando invocação de métodos;
- **Verifica a validade das classes carregadas –** Verifica a integridade e consistência;
- **Compila bytecodes –** Para métodos invocados, mantendo os bytecodes originais para validações run-time;
- **Correcta gestão de memória –** Alocação automática de memória e recolha de lixo automático;
- **Verifica a correcta execução do código das classes –** Com verificações de integridade (limites de arrays, casts, pointeiros nulos) e segurança (controlo de acessos (public, protected, package, private), isolação de proteção de dominios) em run-time;

Ele mantém e tem instalado politicas de segurança especificadas em ficheiros de configuração, que determinam o conjunto de autorizações permitidas ou negadas. O JRE mantém sempre uma politica por definição, sendo possivel instalar outras (requer permissão getPolicy) ou escrever por cima das que existem (requer permissão setPolicy).

Um JRE pode ter um gestor de segurança cujo objectivo é ajudar a implementar politicas de segurança em aplicações, ajudando a verificar se uma ação é permitida ou não antes de a executar. O gestor por predefinição do

JRE é o SecurityManager do java.lang, que pode ser redefinido mas apenas uma vez por cada execução do programa (impedindo que classes corrompam o SM antes definido). Este SM tem muitos métodos de verificação através da classe AccessController e do método checkPermissions. Esta classe é usada para:

- **Decidir se o acesso a um recurso critico do sistema é permitido ou negado;**
- **Marcar código como privilegiado –** Afectando consequentes acessos;
- **Obter um snapshot do actual contexto –** Por forma a tomar decisões de controlo de acesso com base nele.

DOMINIO DE PROTEÇÃO – Conjunto de classes cujas instâncias têm o mesmo conjunto de permissões. O JRE mantem um mapeamento entre código (classes e instâncias) e os seus dominios de proteção. Uma permissão define o que é permitido ou não e pode ser de vários tipos como BasicPermission, FilePermission, SocketPermission.

CARREGAMENTO DE CLASSES DINÂMICO – Define politicas a impor ao carregamento de classes:

- **Não é permitido carregar classes de pacotes java.* da rede –** Para evitar a substituição das classes básicas do Java;
- **Separar espaços de nomes –** Para evitar problemas de duplicação de nomes de classes de fontes diferentes;
- **Classes carregadas de diferentes servidores de rede não interagem –** Têm dominios diferentes, não há interferência entre programas com origens diferentes.

Uma classe deve ser carregada segundo os seguintes passos:

- **Localizar a classe binária requerida –** O ficheiro .class;
- **Traduzir para estrutura de dados interna para emulação;**
- **Respeitar convenção de nomes –** Dominio, package, class, campos/métodos, niveis de acesso;
- **Verificar validade da classe binária –** Consistência e integridade;
- **Executar traduções de código e metadados –** Preparar o método para executar;
- **Iniciar memória e passar o controlo ao motor de emulação;**

E quem vai carregar a classe é um carregador de classes. Um carregador de classes é um componente critico numa VM, e vai prevenir o spoofing de nomes de classes da biblioteca java.*. Um utilizador pode definir um carregador de classe, ajudando a aplicação a localizar e descarregar conteudo de classes. Ao descarregar uma classe, esta vai ser etiquetada com o loader que a descarregou, e classes desse dominio não podem comunicar com classes de outro dominio, mas é possivel ter várias versões de uma mesma classe, que normalmente estão associadas a códigos com origens diferentes.