Segurança Informática e nas Organizações - Resumos 2

José Mendes 107188

2023/2024

1 Criptografia Assimétrica

1.1 Criptografia Assimétrica (de blocos)

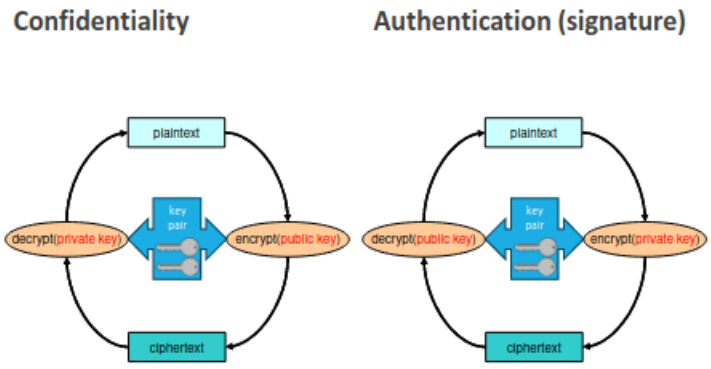
Usa um par de chaves:

- **Chave privada:** pessoal, não transmissível;
- **Chave pública:** disponível a todos;

Permite:

- Confidencialidade sem qualquer exchange of secrets prévia;
- Autenticação
  - De conteúdos (integridade dos dados);
  - De origem (atenticação da source, ou assinatura digital);

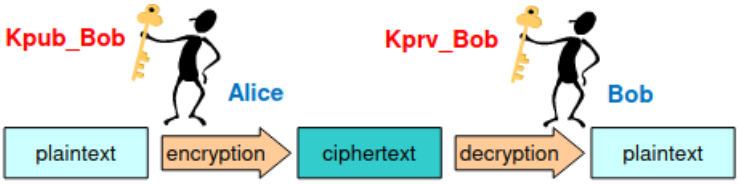
1.2 Operaçõees de uma Cifra Assimétrica



1.3 Use Cases: Comunicação Segura

Comunicação segura com um target (Bob)

- A Alice encrípta o plaintext **P** com a chave pública do Bob, **Kpub_Bob**
  - **Alice:** $C = \{P\}_{kpub\_bob}$
- O Bob decifra o ciphertext **C** com a sua chave privada, **Kpriv_Bob**
  - **Bob:** $P' = \{C\}_{kpriv\_bob}$
- P' deve ser igual a **P** (é necessário verificar)
- **Kpub_Bob** precisa de ser conhecida pela Alice



1.4 Cifras Assimétricas

Vantagens:

- São um mecânismo de autenticação fundamental;
- Permitem explorar caracteristicas que não são possíveis com cifras simétricas;

Desvantagens:

- Performance;
- Normalmente não são muito eficientes e consomem muita memória;

Problemas:

- Distribuição confiável de chaves públicas;
- O lifetime do par de chaves é limitado;

Abordagens: problemas matemáticos complexos

- Logaritmos discretos de números grandes;
- Factorização inteira de números grandes;

Algoritmos mais comuns:

- RSA;
- ElGamal;
- Eliptic Curves (ECC);

Outras tecnicas com pares assimétricos de chaves:

- Diffie-Hellman (key agreement);

1.5 RSA (Rivest, Shamir, Adelman, 1978)

Chaves:

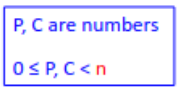
- **Privada:** (d, n)
- **Pública:** (e, n)

Encriptação da chave pública (confidencialidade)

- $C = P^e \mod n$
- $P = C^d \mod n$

Encriptação da chave privada (assinatura)

- $C = P^d \mod n$
- $P = C^e \mod n$



Complexidade Computacional

- Logaritmo discreto;
- Factorização inteira;

Seleção de Chaves

- **n** grande (centenas ou milhares de bits);
- $n = p \times q$ com **p** e **q** sendo números primos grandes (secretos);
- Escolher um **e** co-primo de $(p-1) \times (q-1)$;
- Computar **d** tal que $e \times d \equiv 1 \pmod{(p-1) \times (q-1)}$;
- Discartar **p** e **q**;
- O valor de **d** não pode ser facilmente computado a partir de **e** e **n** (apenas de **p** e **q**);

1.5.1 RSA - Exemplo

p = 5 q = 11 **(prime numbers)**

- n = p x q = 55
- (p-1) x (q-1) = 40

e = 3 **(public key = e, n)**

- Coprime of 40

d = 27 **(private key = d, n)**

- e x d ≡ 1 (mod 40) → d x e mod 40 = 1, (27 x 3) mod 40 = 1

For P = 26 **(notice that P, C∈ [0, n-1])**

- $C = P^e \bmod n = 26^3 \bmod 55 = 31$
- $P = C^d \bmod n = 31^{27} \bmod 55 = 26$

1.6 Encriptação Hibrida

Mistura criptografia simétrica com assimétrica

- Usa o melhor dos dois mundos, evitando os problemas;
- Cifra assimétrica: usa chaves públicas (mas é lenta);
- Cifra simétrica: Rápida (mas com métodos fracos de troca de chaves);

Método

- Obtém K_{pub} do destinatário;
- Gera uma chave simétrica aleatória K_{sym};
- Calcula $C1 = E_{sym}(K_{sym}, P)$;
- Calcula $C2 = E_{asym}(K_{pub}, K_{sym})$;
- Envia $C1 + C2$;
  - $C1$ é o texto encriptado com a chave simétrica;
  - $C2$ é a chave simétrica encriptada com a chave pública do destinatário (pode também conter um IV);

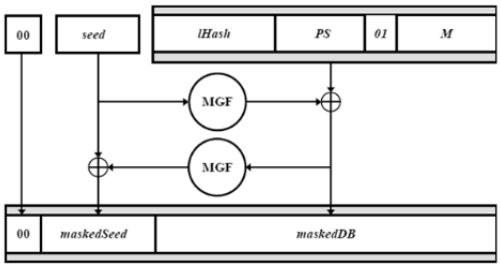
1.7 Randomização de encriptações assimétricas

Resultado de encriptações assimétricas não deterministico (não é previsível)

- **N** encriptações do mesmo valor, com a mesma chave, deve produzir **N** resultados diferentes;
- **Objetivo:** Previnir a descoberta de valores encriptados através de tentativa e erro;

Abordagens: Concatenação de um valor a encriptar com dois valores, um fixo (para controlo de integridade) e outro aleatório (para randomização);

1.7.1 OAEP (Optimal Asymmetric Encryption Padding)



lHash: digest over **Label**

seed: random

PS: zeros

M: plaintext

MGF: Mask Generation Function

- Similar to Hash, but with variable size

1.8 Diffie-Hellman Key Agreement (1976)

q (large prime)
α (primitive root mod q)

a = random

$Y_a = \alpha^a \bmod q$

$K_{ab} = Y_b^a \bmod q$

Y_a

Y_b

b = random

$Y_b = \alpha^b \bmod q$

$K_{ba} = Y_a^b \bmod q$

$K_{ab} = K_{ba}$

1.8.1 DH Key Agreement: MitM Attack

a = random

$Y_a = \alpha^a \bmod q$

$K_{ac} = Y_c^a \bmod q$

Y_a

Y_c

c = random

$Y_c = \alpha^c \bmod q$

$K_{ca} = Y_a^c \bmod q$

$K_{cb} = Y_b^c \bmod q$

Y_b

Y_c

b = random

$Y_b = \alpha^b \bmod q$

$K_{bc} = Y_c^b \bmod q$

1.9 Eliptic Curve Cryptography (ECC)

Curvas elipticas são funções específicas

- Têm um gerador **G**;
- Uma chave privada K_{priv}, é um inteiro com um máximo de bits permitidos pela curva;
- Uma chave pública K_{pub}, é um ponto $(x, y) = K_{priv} \times G$
- Dada K_{pub}, deve ser computacionalmente dificil determinar K_{priv};

Curves

- NIST curves (15)
  - P-192, P-224, P-256, P-384, P-521
  - B-163, B-233, B-283, B-409, B-571
  - K-163, K-233, K-283, K-409, K-571

Other curves

- Curve25519 (256 bits)
- Curve448 (448 bits)

1.10 ECDH: DH com ECC

ECC curve → G

a = random

$Y_a = a\ G$

$K_{ab} = a\ Y_b$

Y_a

Y_b

b = random

$Y_b = b\ G$

$K_{ba} = b\ Y_a$

$K_{ab} = K_{ba}$

1.11 Encriptação de chave pública com ECC

Mistura encriptação hibrida com EDHC

Método

- Obtém K_{pub_recv} do destinatário;
- Gera um random K_{priv_send} com um correspondente K_{pub_send};
- Calcula $K_{sym} = K_{priv\_send} \times K_{pub\_recv}$;
- $C = E(P, K_{sym})$;
- Envia $C + K_{pub\_send}$;
- Destinatário calcula $K_{sym} = K_{pub\_send} \times K_{priv\_recv}$;
- $P = D(C, K_{sym})$;

2 Assinaturas digitais

2.1 Cifras Assimétricas (de blocos)

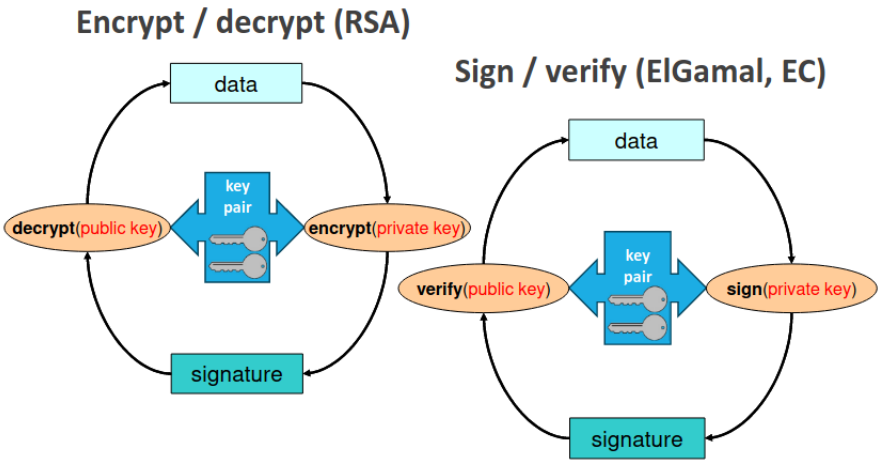
Usa pares de chaves:

- Uma **chave privada** (pessoal, não transmissível);
- Uma **chave pública** (disponível a todos);

Permite:

- Confidencialidade sem qualquer exchange of secrets prévia;
- Autenticação
  - De conteúdos (integridade dos dados);
  - De origem (atenticação da source, ou assinatura digital);

2.2 Assinaturas Digitais



Autenticação de conteúdos de um documento - Garante a sua integridade (não se alterou);

Autenticação do autor - Garante que a identidade do criador/origem;

Previnir repudiação de assinaturas

- Non-repudiation (o autor não pode negar a autoria);
- Autores genuínos não podem negar a autoria (apenas a identidade do autor pode gerar uma dada assinatura);

Aboradgens

- Encriptação/Decifração assimétrica ou assinatura/verificação;
- Funções digest (apenas para performance);

$$\text{Signing: } A_x(\text{doc}) = \text{info} + E(K_x^{-1}, \text{digest}(\text{doc} + \text{info}))$$

$$A_x(\text{doc}) = \text{info} + S(K_x^{-1}, \text{digest}(\text{doc} + \text{info}))$$

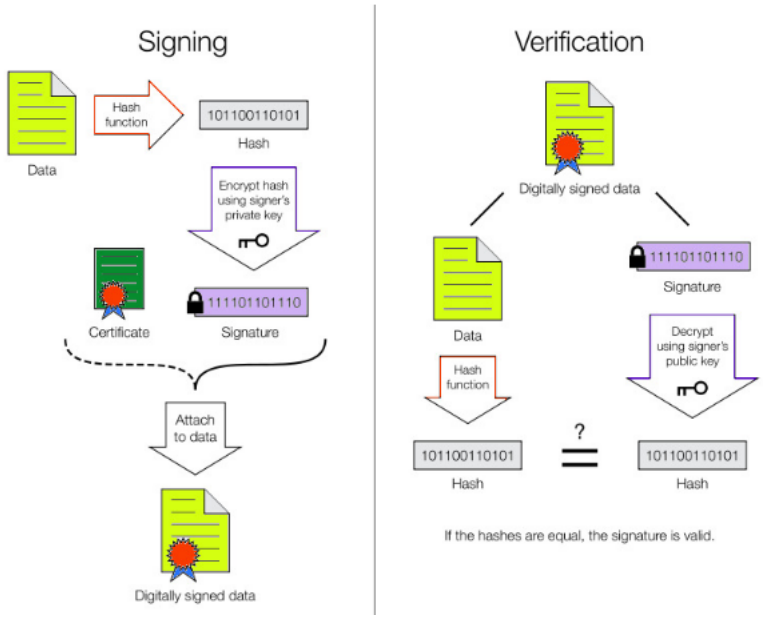
$$\text{info} = \text{signing context, signer identity, } K_x$$

$$\text{Verification:}$$

$$D(K_x, A_x(\text{doc})) \equiv \text{digest}(\text{doc} + \text{info})$$

$$V(K_x, A_x(\text{doc}), \text{doc}, \text{info}) \rightarrow \text{True / False}$$

2.2.1 Encriptação/Decifração signatures



2.2.2 Assinatura digital num email: Multipart content, signature w/ certificate

```
From - Fri Oct 02 15:37:14 2009
[…]
Date: Fri, 02 Oct 2009 15:35:55 +0100
From: =?ISO-8859-1?Q?Andr=E9_Z=FAquete?= <andre.zuquete@ua.pt>
Reply-To: andre.zuquete@ua.pt
Organization: IEETA / UA
MIME-Version: 1.0
To: =?ISO-8859-1?Q?Andr=E9_Z=FAquete?= <andre.zuquete@ua.pt>
Subject: Teste
Content-Type: multipart/signed; protocol="application/x-pkcs7-signature"; micalg=sha1; boundary="------------ms050405070101010502050101"

This is a cryptographically signed message in MIME format.

--------------ms050405070101010502050101
Content-Type: multipart/mixed;
 boundary="------------060802050708070409030504"

This is a multi-part message in MIME format.
--------------060802050708070409030504
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable

Corpo do mail

--------------060802050708070409030504-
--------------ms050405070101010502050101
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: S/MIME Cryptographic Signature

MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAoIIamTCC
BUkwggSyoAMCAQICBAcnIaEwDQYJKoZIhvcNAQEFBQAwdTELMAkGA1UEBhMCVVMxGDAWBgNV
[…]
KoZIhvcNAQEBBQAEgYCofks852BV77NVuww53vSxO1XtI2JhC1CDlu+tcTPoMD1wq5dc5v40
Tgsaw0N8dqgVLk8aC/CdGMbRBu+J1LKrcVZa+khnjjtB66HhDRLrjmEGDNttrEjbqvpd2QO2
vxB3iPTlU+vCGXo47e6GyRydqTpbq0r49Zqmx+IJ6Z7iigAAAAAAAA==
--------------ms050405070101010502050101--
```

3 Derivação de chaves

Algotitmos de cifras requerem chaves de tamanho fixo - 56, 128, 256,... bits;

Podemos derivar chaves de múltiplas origens- shared secrets, passwords geradas por humanos, PIN codes e secrets de tamanho pequeno;

Origem original pode ter baixa entropia - reduz a dificuldade de ataques de força bruta, no entanto, devemos ter uma relação forte para uma chave útil;

Por vezes percisamos de múltiplas chaves do mesmo material - enquanto não permite encontrar o material (a password, outra chave) de uma chave nova;

3.1 Prepósitos de derivação de chaves

Refroço de chaves: aumenta a segurança de uma password

- Nomralmente definido por humanos;
- Tornando ataques de dicionário nada práticos;

Expansão de chaves: aumenta o tamanho de uma chave

- Expande o tamanho que serve o algoritmo;
- Eventualmente deriva outras chaves relacionadas para outros algoritmos (ex: MAC);

3.2 Derivação de chaves

Derivação de chaves requer a existência de:

- Um **salt** que trona a derivação única;
- Um problema difícil;
- Um nível de complexidade escolhido;

Dificuldade de Computação

- A transformação requer recursos computacionais relevantes;

Dificuldade de Memória

- A transformação requer recursos de armazenamento relevantes;
- Limita os ataques usando aceleração de hardware;

3.3 Derivação de chaves: PKBDF2

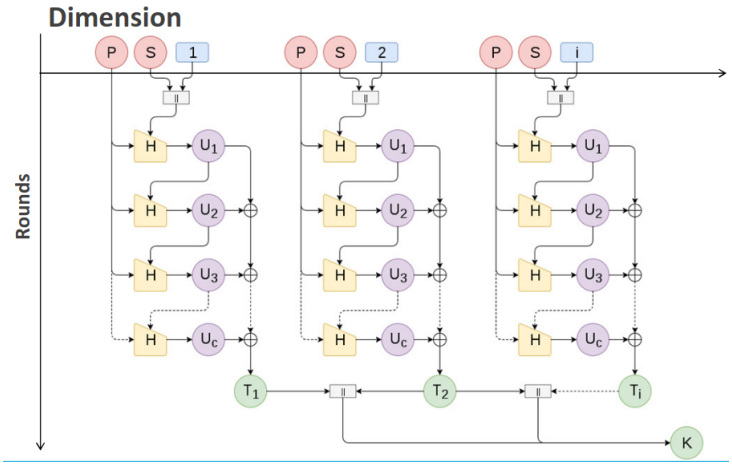
Password Based Key Derivation Function 2

Produz uma chave a partir de uma password, com uma dificuldade escolhida

$$K = PBKDF2(PRF, Salt, rounds, dim, password)$$

- **PRF** - Pseudo-Random-Function: função digest;
- **Salt** - Valor aleatório;
- **Rounds** - O custo computacional (dezenas ou centenas de milhares);
- **Dim** - Tamanho do resultado pretendido;

Operação: calcula operações ROUNDS x DIM a partir do PRF utilizando o SALT e a PASSWORD - um tamanho maior de rounds aumenta a custo;

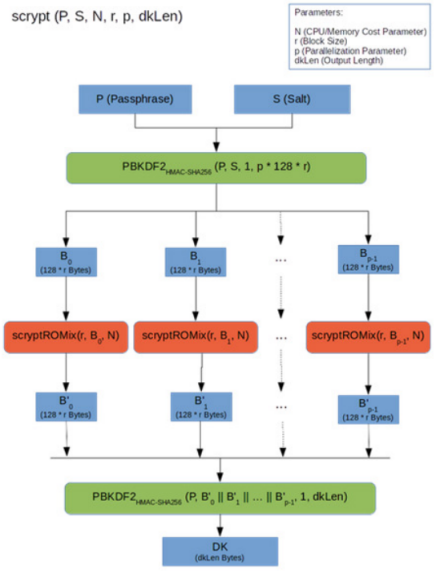


3.4 Derivação de chaves: Scrypt

Produz uma chave com um custo de armazenamento escolhido

$$K = scrypt(password, salt, n, p, dim, r, hLen, Mflen)$$

- **Password** - Um segredo;
- **Salt** - Valor aleatório;
- **n** - Parâmetro de custo;
- **p** - Parâmetro de paralelismo $p \leq (2^{32} - 1) \times hLen/Mflen$;
- **dim** - Tamanho do resultado pretendido;
- **r** - Tamanho do bloco a usar (default: 8);
- **hLen** - Tamanho do da função digest (32 para SHA256);
- **Mflen** - Bytes na internal mix (default: $8 \times r$);



4 Gestão de chaves assimétricas

4.1 Problemas a resolver

Garante um uso correto do par de chaves assimétricas

- **Privacidade das chaves privadas**
  - Garante a autenticidade;
  - Previne a repudiação de assinaturas digitais;
- **Distribuição correta de chaves públicas**
  - Garante confidencialidade;
  - Garante a correta validação de assinaturas digitais;

Evolução temporal da entidade $\longleftrightarrow$ mapeamento de pares chave

- **Para combater ocurrência catastróficas** (ex: perda de chaves privadas)
- **Para combater os requisitos de exploitations normais** (ex: refresh do par de chaves para reduzir riscos personificação)

Garante a correta geração de pares de chaves

- **Geração aleatória de valores secretos**, de forma a poderem ser facilmente previstos;
- **Aumentar a eficiência sem reduzir a segurança**
  - Tornar mecânismos de segurança mais eficientes;
  - Aumentar a performance;

4.2 Objetivos

- **Geração do par de chaves** - quando e como gerar;
- **Lidar com a chave privada** - como manter a chave privada;
- **Distribuição da chave pública** - como ditribuir corretamente as chaves públicas world-wide;
- **Tempo de vida do par de chaves** - quando vão expirar, até quando as usar e como verificar se esse par de chaves está obsuleto;

4.3 Geração de pares de chaves: Principais Designs

Usar bons geradores de números aleatórios para produzir segredos

O resultado é indestinguível de noise, isto é, todos os valores possíveis são igualmente prováveis e, não existem padrões resultantes do número da iteração ou de valores prévios;

Exemplo: Bernoulli 1/2 Generator

- Gerador sem memória;
- $P(b = 1) = P(b = 0) = 1/2$;
- Coin toss (atirar uma moeda ao ar);

Facilidade sem comprometer a segurança

Chaves públicas eficientes

- Algumas são 1 bits, tipicamente $2k + 1$ valores (3, 17, 65537);
- Acelerar o processo com chaves públicas (o custo é proporcional ao número de bits 1);
- Sem security issues;

Self-generation de chaves privadas

Maximiza a privacidade, uma vez que outros nunca vão conseguir usar a dada chave privada. Apenas o dono tem a chave, melhor ainda, o dono não tem a chave, mas pode usá-la;

Principio pode ser relaxado quando não involve a geração de assinaturas. Quando não existem issues relacionados com non-repudiation.

4.4 Lidar com chaves privadas

- **Correctness**
  - **A chave privada representa o sujeito** (i.e., um cidadão, um servidor, etc.). O seu compromise deve ser minimizado. Cópias físicas seguras (backups) podem existir em alguns casos;
  - **O caminho de acesso à chave privada deve ser controlado**. Proteção de acesso com password ou PIN code. Correctness das aplicações que usam;
- **Confinement**
  - **Proteção da chave privada dentro de um domínio seguro (reduzido) (ex: cryptographic token)**. O Token gera pares de chaves, exporta a chave pública mas nunca a privada, e, este Token encripta/decifra internamente com a chave privada.
  - **Exemplo: SmartCards**, podemos pedir ao cartão para cifrar/decifrar algo. A chave privada nunca sai do SmartCard.

4.5 Distribuição de chaves públicas

Distribuição a todos os senders de dados confidenciais. Processo manual, usando um shared secret. Ad-hoc usando certificados digitais;

Distribuição a todos os receivers de assinaturas digitais. Processo manual. Ad-hoc usando certificados digitais;

4.5.1 Problema

Como garantir a Correctness de uma chave pública?

Disseminação confiável de chaves públicas - Paths/Graphs confiáveis. Se **A confia em** K_X^+ e **B confia em A**, então **B confia em** K_X^+.

Hierarquias de certificação/grafos com as relações de confiança expressas entre entidades. Certificação é unidirecional!

4.6 Public key (digital) certificates

É um **documento digital issued por uma autoridade de certificação (CA)**

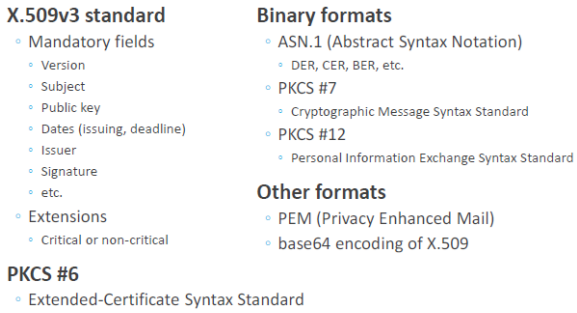
- **Liga a chave pública a uma entidade** (ex: pessoa, servidor ou serviço);
- **São documentos públicos**, não contêm informação privada, apenas pública. Pode ter informação adicional (ex: URL, nome, email, etc.);
- **São seguros criptograficamente**, digitalmente assinados pelo issuer, não podem ser alterados;

Pode ser usado para ditribuir chaves públicas de uma forma confiável

O certificate receiver pode ser validade de várias formas

- Com a chave pública do CA;
- Pode também validar a identificação;
- Validar a validade;
- Validar se a chave está a ser usada para o propósito correto;

O certificate receiver confia no comportamento do CA, pelo que confia os documentos que este assina. Quando o CA associa um certificado a A, se o receicer confiar no CA, então confia que a associação de A é correta.



4.7 Key pair usage

O certificado publico conecta o par de chaves a um perfil de utilização. As chaves privadas raramente são multifuncionais.

Perfil de utilização tipico

- Autenticação/distribuição de chaves (Digital signature, Key encipherment, Data encipherment, Key agreement)
- Assinar documentos (Assinatura digital não repudiável)
- Certificate issuing (exclusivo de CAs). Assinar certificados, assinar CRLs (Certificate Revocation Lists)
- Timestamping (exclusivo de Time Stamping Authorities TSAs)

Certificados de chaves públicas têm uma extensão para isto, key usage (critical) que indica o perfil de utilização da chave pública.

4.8 Assinatura de Certificados (CA)

Organizações que gerem certificados de chaves públicas. Companhias, não por lucro, governamentais, etc.;

Define politicas e mecanismos para:

- Issuing de certificados;
- Revoking de certificados;
- Distribuição de certificados;
- Issuing e distribuição das correspondentes chaves privadas;

Gerir a lista de certificados revogados (CRLs), interfaces programaticas para verificar o estado atual de um certificado;

4.9 Trusted Certification Authorities

CAs intermediários - CAs certificados por outras CAs confiaveis. Usando um certificado, permitindo a criação de uma hierarquia de certificação;

Anchor confiavel (ou root CA) - Um tem uma chave pública confiavel, normalmente implementada por certificados self-certified, ou seja, issuer e subject são o mesmo. Distribuição manual (ex: dentro do código do browser, OS, distribuição, etc.).

Ver Exemplo de certificado nos slides 19-23.

4.10 Refreshing of asymmetric key pairs

O par de chaves deve ter um tempo de vida limitado - uma vez que, as chaves privadas podem ser perdidas ou descobertas, e para implementar uma politica de update;

Problema - Os certificados podem ser copiados e distribuidos. O universo de donos de certificados é desconhecido, pelo que, não podemos contactá-los para eliminar certificas específicos;

Solução - Certificados com um periodo de validade (nem antes, nem depois). Listas de certificados revogados, para revogar certificados antes da validade expirar;

4.11 Certificate Revocation Lists (CRLs)

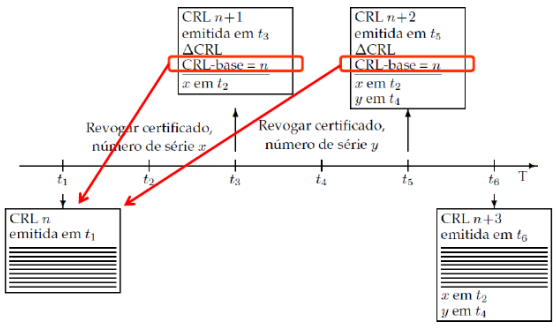
Base ou delta - Completa / diferenças

Listas de certificados assinados (identifiers) prematuramente invalidados

- Devem ser regularmente visitado por donos de certificados
- Protocolo OCSP para verificar a validade de um certificado (RFC 2560)
- Pode dizer a razão da revogação (slide 31)

Publicação e distribuição de CRLs - Cada CA mantém a sua CRL e permite o acesso publico da mesma.

4.12 CRL e Delta CRL



4.13 Online Certificate Status Protocol (OCSP)

Protocolo baseado em HTTP para dar assert ao estado de um certificado

- **Request** - Inclui o serial number do certificado;
- **Response** - Inclui se o certificado está revoked, é enviado pela CA e não possui validade;
- Uma verificação por certificado;

Requer menor largura de banda para clientes - uma verificação por certificado em vez que fazer download de uma lista de certificados revogados (CRL);

Involve maior largura de bannda para CAs - uma verificação por certificado, problemas de privacidade uma vez que um CA saberá que um certificado está a ser usado;

OOCSP stapling - Inclui um timestamp recentemente assinado na resposta do servidor para dar assert à validade. Reduz a demora da verificação e carregamento no CA. Previne problemas de privacidade.

4.14 Distribuição de certificados de chaves públicas

Transparente (integrado com sistemas ou aplicações)

- Direcory systems, larga escala (ex: X.500 através de LDAP), organizacional (ex: Windows 2000 Active Directory), etc.;
- On-line: sem protocolos que usam certificados para autenticação peer (ex: protocolos de comunicação segura (TLS, IPSec, etc.), assinaturas digitais, dentro de MIME mail messages ou dentro de documentos);

Explicito (voluntáriamente ativado pelos users)

- User faz request de um serviço para obter um certificado necessário (ex: pedido mandado por email, acesso a uma página HTTP pessoal).

4.15 PKI (Public Key Infrastructure)

Infrastrutura para permitir um uso correto de chaves assimétricas e de certificados de chave pública.

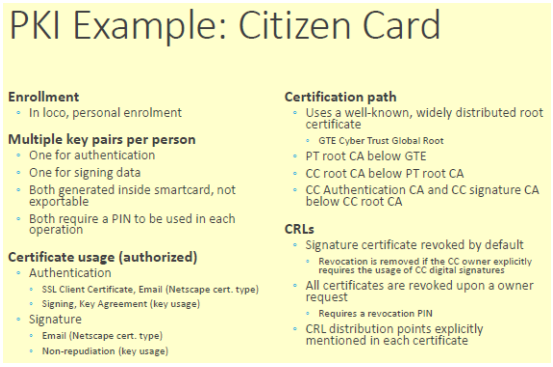
Criação do par de chaves assimétricas para cada entidade que participa. Politicas de participação, ploiticas de geração do par de chaves.

Criação e distribuição de certificados de chaves públicas. Politicas de participação, definição de atributos de certificados.

Definição e uso de certification chains (ou paths). Inserção numa hierarquia de certificação, certificação de outros CAs.

Atualização, publicação e consulta de CRLs. Politicas de revogação de certificados, serviços de distribuição de CRL, serviços OCSP.

Usa estruturas de dados e protocolos permitindo inter-operabilidade entre componentes/serviços/pessoas.



4.16 Certificate Pinning

Se um atacante tem acesso a uma Root confiavel, pode fingir ser qualquer entidade. Manipula um CA confiavel em dar issue ao certificado (pouco provável). Injetar um CA costumizado na base de dados da vitima (mais provável).

Certificate Pinning: adiciona a fingerprint do PubK (public key) ao código fonte. A fingerprint é uma hash (ex: SHA256).

Validação do processo: O certificado deve ser válido a regras locais, deve possuir uma chave pública com a dada fingerprint.

4.17 Certificate Transparency (RFC 6962)

Problemas

- CAs podem ser comprometidos (ex: DigiNotar) por atacantes, governo, etc.
- Comprometer é dificil de detetar. Resulta na mudança de suposições associadas com o comportamento do CA. O proprietário saberá sozinho.

Definição: um sistema global que regista todas as informações sobre certificados públicos criados

- Garante que apenas um certificado tem as roots corretas;
- Guarda a chain de certificados inteira para cada certificado;
- Apresenta esta informação para auditoria (organizações ou ad-hoc por end users);