

9 – Signatures digitales

SIGNATURES DIGITALES

- ▶ la notion de **signature électronique** a été introduite par **Diffie** et **Hellman** en 1976
- ▶ la **signature électronique** a été reconnue en France par la loi n° 2000-30 du 13 mars 2000 portant adaptation du droit de la preuve aux technologies de l'information et relative à la signature
 - « Art. 1316-3. -L'écrit sur support électronique a la même force probante que l'écrit sur support papier »

- ▶ elle vise à garantir l'identité de l'auteur d'un message par **authentification** et aussi l'**intégrité** du message signé
- ▶ on veut empêcher la **falsification** de la signature et la **non-reconnaissance** par son expéditeur

Pour la mise-en-œuvre, on a recours aux moyens classiques :

- ▶ système à **clef secrète** : basé sur une fonction de hachage utilisée en mode **MAC** (Message Authentication Code, cf. cours 8)
- ▶ système à **clef publique** : inspiré des techniques de chiffrement à clef publique (**signatures RSA, El Gamal, DSA**)

CAHIER DES CHARGES

Une **signature électronique** doit dépendre du document à signer et être :

- ▶ facile à calculer
- ▶ vérifiable par le destinataire
- ▶ vérifiable par un tiers
- ▶ infalsifiable, inaltérable
- ▶ inimitable, non-réutilisable
- ▶ non-répudiable

Les signatures garantissent donc simultanément l'**authentification**, l'**intégrité** et la **non-répudiation**.

SYSTÈME DE SIGNATURE

- ▶ $S_{K_{pr}}$ est l'algorithme de **signature privée** de paramètre la **clef** K_{pr} :

$$S = S_{K_{pr}}(M)$$

- ▶ il va de pair avec un algorithme **public de vérification** $V_{K_{pub}}$:

$$V_{K_{pub}}(M, S) = \begin{cases} True & \text{si } S = S_{K_{pr}}(M) \\ False & \text{si } S \neq S_{K_{pr}}(M) \end{cases}$$

- ▶ à noter que le plus souvent, c'est une **empreinte** $h(M)$ du message qui est signée, et non le message lui-même

SIGNATURE RSA

- ▶ Alice et Bob conviennent d'une fonction de hachage h
- ▶ Alice veut envoyer un message M signé à Bob
- ▶ Alice dispose d'une clef privée d et d'une clef publique (e, n)
le module n , l'entier d et son inverse e modulo $\varphi(n)$ sont les paramètres RSA habituels
- ▶ Alice calcule l'empreinte $h(M)$ de son message M
- ▶ Alice signe l'empreinte $h(M)$ et envoie à Bob le couple (M, S) avec :

$$S = S_d(h(M)) = h(M)^d \pmod n$$

- ▶ Bob qui reçoit le message M et la signature S de son empreinte $h(M)$ vérifie la signature d'Alice :

$$V_e(M, S) = \text{True} \iff h(M) \equiv S^e \pmod n$$

SIGNATURE RSA : EXEMPLE

- ▶ Alice et Bob conviennent d'une fonction de hachage h
- ▶ Alice veut envoyer un message M signé à Bob
- ▶ elle choisit $p = 13$ et $q = 19$ soit $n = 247$
- ▶ elle calcule $\varphi = 216$ et choisit $e = 61$
- ▶ par Bezout, elle obtient sa clef privée $d = 85$
- ▶ Alice publie sa clef publique $(e, n) = (61, 247)$
- ▶ Alice calcule l'empreinte de son message M , disons que $h(M) = 165$
- ▶ Alice signe l'empreinte $h(M)$ et envoie à Bob le couple (M, S) avec :

$$S = S_d(h(M)) = h(M)^d \pmod n = 165^{85} \pmod{247} = 243$$

- ▶ Bob ou un tiers reçoit le couple (M, S) et vérifie la signature d'Alice en recalculant $h(M)$:

$$V_e(M, S) = \text{True} \iff h(M) \equiv S^e \pmod n$$

$$V_e(M, S) = \text{True} \iff 165 \equiv 243^{61} \pmod{247}$$

CHIFFREMENT + SIGNATURE RSA

- ▶ Alice veut envoyer un message M secret et signé à Bob
- ▶ Alice dispose d'une clef privée d_A et d'une clef publique (e_A, n_A)
- ▶ Bob dispose d'une clef privée d_B et d'une clef publique (e_B, n_B)
- ▶ Alice chiffre son message M ($M < \min(n_A, n_B)$) avec la clef publique de Bob puis signe le chiffré et l'envoie à Bob :

$$C = E_{e_B}(M) = M^{e_B} \pmod{n_B}$$

$$Z = S_{d_A}(C) = C^{d_A} \pmod{n_A}$$

- ▶ Bob qui reçoit le couple chiffré, signature du chiffré (C, Z) vérifie la signature d'Alice puis le déchiffre :

$$V_{e_A}(C, Z) = \text{True} \iff C \equiv Z^{e_A} \pmod{n_A}$$

$$M = D_{d_B}(C) \equiv C^{d_B} \pmod{n_B}$$

- ▶ en pratique, cette technique du fait de sa lenteur se limite à l'échange de clefs

CHIFFREMENT + SIGNATURE RSA : EXEMPLE

- ▶ Alice veut envoyer un message $M = 13$ secret et signé à Bob
- ▶ Alice a sa clef privée $d_A = 29$ et sa clef publique $(e_A, n_A) = (5, 65)$
- ▶ Bob a sa clef privée $d_B = 43$ et sa clef publique $(e_B, n_B) = (7, 77)$
- ▶ Alice chiffre son message M avec la clef publique de Bob puis signe le chiffré et l'envoie à Bob :

$$C = E_{e_B}(M) = M^{e_B} \pmod{n_B} = 13^7 \pmod{77} = 62$$

$$Z = S_{d_A}(C) = C^{d_A} \pmod{n_A} = 62^{29} \pmod{65} = 17$$

- ▶ Bob qui reçoit le couple $(C, Z) = (62, 17)$ vérifie la signature d'Alice puis le déchiffre :

$$V_{e_A}(C, Z) = \text{True} \iff C \equiv Z^{e_A} \pmod{n_A} = 17^5 \pmod{65} = 62$$

$$M = D_{d_B}(C) \equiv C^{d_B} \pmod{n_B} = 62^{43} \pmod{77} = 13$$

SIGNATURE EL GAMAL

- ▶ cette **signature** électronique a été proposée en 1984 par **El Gamal**
- ▶ au contraire de **RSA**, la **signature El Gamal** a été d'emblée conçue pour **signer**
- ▶ elle repose sur le **problème du logarithme discret** réputé difficile
- ▶ toutefois, elle diffère de l'algorithme de **chiffrement El Gamal**
- ▶ la signature El Gamal permet bien à un **tiers** de vérifier l'**authenticité** d'un message
- ▶ là encore, c'est le plus souvent une **empreinte h** qui est signée à la place d'un message
- ▶ le standard actuel **DSA** (*Digital Signature Algorithm*) en est une variante

SIGNATURE EL GAMAL (MISE EN ŒUVRE)

- ▶ préparation des clefs :
 1. le signataire **Bob** choisit un grand entier premier p et un générateur α du groupe multiplicatif \mathbb{Z}_p^*
 2. il choisit pour **clef privée** un entier k vérifiant $1 < k < p$
 3. il calcule $\beta = \alpha^k \bmod p$ et publie sa **clef publique** (p, α, β)

- ▶ signature du message M :
pour M ($0 < M < p$) **Bob** choisit un entier aléatoire r premier avec $p - 1$ et calcule

$$\gamma = \alpha^r \bmod p$$

$$\delta \text{ tel que } k \cdot \gamma + r \cdot \delta \equiv M \pmod{p-1}$$

et envoie à **Alice** la signature $S = (\gamma, \delta)$

- ▶ vérification de S :
Alice ou un **tiers** vérifie que la signature $S = (\gamma, \delta)$ provient de **Bob** :

$$0 < \gamma < p \text{ et } 0 < \delta < (p-1)$$

$$V_{K_{pub}}(M, \gamma, \delta) = True \Leftrightarrow (\beta^\gamma \gamma^\delta) \bmod p = \alpha^M \bmod p$$

SIGNATURE EL GAMAL (JUSTIFICATION)

- ▶ clefs :
 - ▶ **clef privée** de **Bob** : un entier k vérifiant $1 < k < p$
 - ▶ **clef publique** de **Bob** : (p, α, β) avec $\beta = \alpha^k \bmod p$
- ▶ signature du message M :
 - ▶ **Bob** choisit un entier aléatoire r premier avec $p - 1$
 - ▶ **Bob** signe avec $S = (\gamma, \delta)$ avec :

$$\gamma = \alpha^r \bmod p$$

$$\delta \text{ tel que } k \cdot \gamma + r \cdot \delta \equiv M \pmod{p-1}$$

- ▶ vérification de S :

$$(\beta^\gamma \gamma^\delta) \bmod p = \alpha^M \bmod p ?$$

- ▶ justification :

$$(\beta^\gamma) \bmod p \equiv \alpha^{k\gamma} \bmod p$$

$$(\gamma^\delta) \bmod p \equiv \alpha^{r(M-k\gamma)r^{-1}} \bmod p$$

$$(\gamma^\delta) \bmod p \equiv \alpha^{M-k\gamma} \bmod p$$

$$(\beta^\gamma \gamma^\delta) \bmod p \equiv \alpha^{k\gamma + M - k\gamma} \bmod p$$

$$(\beta^\gamma \gamma^\delta) \bmod p \equiv \alpha^M \bmod p$$

SIGNATURE EL GAMAL (EXEMPLE)

- ▶ préparation des clefs :
 1. le signataire **Bob** choisit un grand entier premier $p = 467$ et un générateur $\alpha = 2$ de \mathbb{Z}_p^*
 2. il choisit pour **clef privée** un entier $k = 127$ vérifiant $1 < k < p$
 3. il calcule $\beta = 132$ et publie sa **clef publique** $(p, \alpha, \beta) = (467, 2, 132)$
- ▶ signature du message M :
pour signer $M = 100$, **Bob** choisit un entier aléatoire $r = 213$ premier avec $p - 1$

$$\gamma = \alpha^r \bmod p = 2^{213} \bmod 467 = 29$$

$$\text{bezout}(r, p-1) = (1, -35, 16) \Rightarrow r^{-1} = -35 + 466 = 431$$

$$\delta = (M - k \cdot \gamma) \cdot r^{-1} \bmod (p-1) = (100 - 127 \cdot 29) \cdot 431 \bmod 466 = 51$$

$$S = (\gamma, \delta) = (29, 51)$$

- ▶ vérification de S :
Alice ou un **tiers** vérifie $S = (29, 51)$ à l'aide de M en calculant :

$$V_{K_{pub}}(M, \gamma, \delta) = True \Leftrightarrow (132^{29} \cdot 29^{51}) \bmod 467 = 189 = 2^{100} \bmod 467$$

DIGITAL SIGNATURE ALGORITHM (DSA)

- ▶ proposé en 1991 au NIST pour leur **DSS (Digital Signature Standard)**
- ▶ c'est clairement une variante de la **signature El Gamal**
- ▶ elle repose donc sur le **problème du logarithme discret**
- ▶ ce problème est difficile pour de grandes valeurs du **module p** de **512 bits** voire 1024
- ▶ or la signature El Gamal double la taille de son module
- ▶ pour les **cartes à puce** par exemple, c'était trop et la variante **DSA** consiste notamment à réduire la taille de la signature
- ▶ **DSA** offre une signature de 320 bits sur un message de 160, mais reste lent comparée à la **signature RSA**
- ▶ **Ecdsa (Elliptic Curve Digital Signature Algorithm)** est une variante de **DSA** utilisé par exemple par PLAYSTATION 3

SIGNATURE DSA (MISE EN ŒUVRE)

- ▶ préparation des clefs :
 0. choisir une **fonction de hachage h** sûre
à l'origine SHA-1, actuellement de la famille SHA-2
 1. le signataire **Bob** choisit un grand entier q premier de 160 bits
 2. puis un module p premier de 512 ou 1024 bits tel que $p - 1$ multiple de q
 3. il trouve α dont l'ordre dans \mathbb{Z}_p^* est q
 4. il choisit pour **clef privée k** un entier aléatoire vérifiant $0 < k < q$
 5. il calcule $\beta = \alpha^k \bmod p$ et publie sa **clef publique (p, q, α, β)**
- ▶ signature du message M ou de son empreinte :
pour M tel que $0 \leq M < p$, **Bob** choisit un entier aléatoire éphémère r tel que $0 < r < q$ et calcule :
$$\gamma = (\alpha^r \bmod p) \bmod q$$
$$\delta = (M + k \cdot \gamma) \cdot r^{-1} \bmod q$$
et envoie à **Alice** le message M accompagné de sa **signature $S = (\gamma, \delta)$**

SIGNATURE DSA (EXEMPLE)

- ▶ préparation des clefs
 - 1.-2. **Bob** choisit un entier premier $q = 29$ puis le module premier $p = 59$ tel que $p - 1 = 58$ est un multiple de q
 - 3.-4. il choisit $\alpha = 3$ et prend pour **clef privée $k = 7$**
On a bien les 30 premières puissances de $\alpha = 3$ modulo $p = 59$ qui sont :
1 3 9 27 22 7 21 4 12 36 49 29 28 25 16 48 26 19 57 53 41 5 15 45 17 51 35 46 20 1
 5. il calcule $\beta = \alpha^k \bmod p = 3^7 \bmod 59 = 4$ et publie sa **clef publique $(p, q, \alpha, \beta) = (59, 29, 3, 4)$**
- ▶ signature du message M :
pour $M = 26$, **Bob** choisit aléatoirement $r = 10$ et calcule :
$$\gamma = (\alpha^r \bmod p) \bmod q = (3^{10} \bmod 59) \bmod 29 = 49 \bmod 29 = 20$$
$$\text{bezout}(r, q) = \text{bezout}(10, 29) = (1, 3, -1)$$
$$\delta = (M + k \cdot \gamma) \cdot r^{-1} \bmod q = (26 + 7 \cdot 20) \cdot 3 \bmod 29 = 5$$
et envoie à **Alice** le message $M = 26$ et sa **signature $S = (\gamma, \delta) = (20, 5)$**

SIGNATURE DSA (VÉRIFICATION)

- ▶ rejeter la signature si on n'a pas à la fois : $0 < \gamma < q$ et $0 < \delta < q$
- ▶ calculer $w = \delta^{-1} \bmod q$
- ▶ calculer $u_1 = (M \cdot w) \bmod q$
- ▶ calculer $u_2 = (\gamma \cdot w) \bmod q$
- ▶ calculer $v = ((\alpha^{u_1} \cdot \beta^{u_2}) \bmod p) \bmod q$
- ▶ **la signature est valide si $v = \gamma$!**

Retour à l'exemple :

Alice ou un **tiers** vérifie la validité de la **signature $S = (\gamma, \delta)$** de **Bob** en calculant :

- ▶ sachant que $\text{bezout}(5, 29) = (1, 6, -1)$, **Alice** calcule

$$w = \delta^{-1} \bmod q = 6 \bmod 29 = 6$$

- ▶ $u_1 = (M \cdot w) \bmod q = (26 \cdot 6) \bmod 29 = 11$
- ▶ $u_2 = (\gamma \cdot w) \bmod q = (20 \cdot 6) \bmod 29 = 4$
- ▶ $v = ((\alpha^{u_1} \cdot \beta^{u_2}) \bmod p) \bmod q$
 $v = ((3^{11} \cdot 4^4) \bmod 59) \bmod 29 = 49 \bmod 29 = 20$
- ▶ **la signature est valide car $v = \gamma = 20$!!!**

SIGNATURE DSA (JUSTIFICATION)

On montre que $v = \gamma$:

- ▶ on sait que $\alpha^q \equiv 1 \pmod p$ (pour plus de clarté, on omet le $\pmod q$ en exposant)
- ▶ $w \equiv \delta^{-1} \pmod q$
- ▶ $u_1 \equiv (M \cdot w) \pmod q$
- ▶ $u_2 \equiv (\gamma \cdot w) \pmod q$
- ▶ calcul de v :

$$((\alpha^{u_1} \cdot \beta^{u_2}) \pmod p) \pmod q$$

$$((\alpha^{Mw} \cdot \beta^{\gamma w}) \pmod p) \pmod q$$

$$((\alpha^{M\delta^{-1}} \cdot \alpha^{k\gamma\delta^{-1}}) \pmod p) \pmod q$$

$$(\alpha^{\delta^{-1}(M+k\gamma)} \pmod p) \pmod q$$

$$(\alpha^{\delta^{-1}(\delta r)} \pmod p) \pmod q$$

$$(\alpha^r \pmod p) \pmod q$$

γ

GOST

- ▶ la Fédération de Russie a développé ses propres **Russian cryptographic standard algorithms** nommés **GOST algorithms**
- ▶ parallèlement à **Dss** développé aux États-Unis, **GOST** propose un standard pour la signature numérique
- ▶ la préparation des clés est à peu près similaire :
 1. la signataire **Alice** choisit un grand entier q premier de 254 à 256 bits
 2. puis un module p premier entre 509 et 512 bits (ou entre 1020 et 1024) tel que $p - 1$ multiple de q
 3. elle trouve α dont l'ordre dans \mathbb{Z}_p^* est q
 4. elle choisit pour **clef privée** k , entier aléatoire vérifiant $0 < k < q$
 5. elle calcule $\beta = \alpha^k \pmod p$ et publie sa **clef publique** (p, q, α, β)
- ▶ la signature du message M (ou de son empreinte) diffère sur δ : pour son message M tel que $0 \leq M < p$, **Alice** choisit un entier aléatoire éphémère r tel que $0 < r < q$ et calcule :

$$\gamma = (\alpha^r \pmod p) \pmod q$$

$$\delta = (r \cdot M + k \cdot \gamma) \pmod q$$

et envoie à **Alice** la signature $S = (\gamma, \delta)$ et le message M

- ▶ ce qui modifie un peu la vérification (et la justification aussi bien-sûr!)

A suivre ...