Authentification mutuelle

- Définition
  - L'authentification mutuelle ou bidirectionnelle désigne le fait que deux parties s'authentifient en même temps

Authentification mutuelle

- Exemple simple d'authentification mutuelle
  - Double poignée de main « défi – réponse »
  - A et B possèdent un secret partagé S
  - Le client A envoie une valeur de défi unique *sc* au serveur B
  - B envoie une valeur de défi unique *cc* à A
  - A calcule *cr* = hachage (sc + secrète) et envoie à B
  - B calcule *sr* = hachage (cc + secrète) et envoie à A
  - B calcule la valeur attendue de *cr* et vérifie que A a répondu correctement
  - A calcule la valeur attendue de *sr* et vérifie que B a répondu correctement

Authentification mutuelle

- KERBEROS
  - Système de cryptographie à base de clés symétriques
  - Kerberos repose sur un tiers de confiance, appelé Key Distribution Center (KDC)
  - Un KDC se compose d’un serveur d’authentification (AS), qui authentifie un utilisateur, et d’un serveur d’octroi de tickets (TGS)

Authentification mutuelle

- KERBEROS (suite)
  - Kerberos partage avec chaque client du réseau une clé secrète faisant office de preuve d'identité
  - Les clés secrètes partagées permettent l’authentification mutuelle des clients
  - L'authentification proposée par le serveur Kerberos a une durée limitée dans le temps,
    - Pour éviter les attaques par rejeu

Authentification mutuelle

- L'authentification mutuelle est par défaut dans certains protocoles
  - IPSEC via IKE (Internet Key Exchange)
  - SSH

- Facultatif dans d'autres
  - Exemple : SSL-TLS

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

INF4420: Éléments de Sécurité Informatique
Identification, Authentification

Nora Cuppens

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

INF4420a: Sécurité Informatique

Cryptographie III

Aperçu – Crypto III

- Cryptographie à clé publique
  - Algorithme RSA
  - Algorithme de El-Gamal
    - Chiffrement à courbe elliptique (ECC)
  - Algorithmes post-quantiques
- Autres primitives cryptographiques
  - Hachage cryptographique
  - Signature numérique et infrastructures à clé publique (PKI)
  - Échange de clés
    - Diffie-Hellman
    - Cryptographie quantique
- Méthodes cryptanalytiques (suite)
  - Type d'attaques
  - Attaques quantiques
  - Complexité
- Principes d'utilisation de la cryptographie
  - Gestion de clés (privés et publiques)
  - Risques résiduels liés à l'utilisation de la crypto

RSA : préliminaires

- Deux clefs:
  - clef publique : une paire (e, N)
  - clef privée : une paire (d, N)
- Première étape: choisir N
  - Choisir n, la taille de N en fonction de sécurité requise
    - Selon le niveau de sécurité désiré
    - Tailles typiques n = (1024), 2048, 3072, 4096, 8192
  - Trouver deux nombres premiers p et q de taille $n/2$
  - Calculer $N = p*q$
- Choisir un entier e de taille n
  - relativement premier à $\varphi(N) = (p-1)*(q-1)$ i.e. pgcd (e, $\varphi(N)$) = 1
- Choisir d tel que
  - $e * d \equiv 1 \mod ((p - 1)*(q - 1))$
- À la fin, on n'a plus besoin de conserver p et q

RSA : Chiffrement et déchiffrement

- Alphabet de codage
  - $T = Z_N^*$
    - généralement représentés par des chaînes de n bits
- Fonction de chiffrement
  - $x \in Z_N^* \rightarrow y = E_e(x) = x^e \bmod N$
- Fonction de déchiffrement
  - $y \in Z_N^* \rightarrow x' = D_d(y) = y^d \bmod N$
- Est-ce que $x' = x$?
  - $D_d(E_e(x)) = (x^e)^d \bmod N = x^{ed} \bmod N = x \bmod N$

    (parce que $e*d = 1 \bmod \varphi(N)$ et théorème de Euler)

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

L'algorithme RSA : exemple

- On utilise des valeurs petites pour illustrer
  - Soit p = 11 et q = 13, alors
    - $N = p * q = 143$ et $\phi(N) = (p-1) * (q-1) = 120$
  - e doit être relativement premier à $(p-1) * (q-1)$:
    - Exemple 1 : soit e = 11
      - l'inverse de 11 mod 120 est aussi 11: 11 * 11 = 121 = 1 * 120 + 1 (pas très astucieux ! Cas atypique …)
      - Pour chiffrer un mot de code $x = 7 \rightarrow y = 7^{11} \bmod 143 = 106$
      - Pour déchiffrer $y = 106 \rightarrow x = 106^{11} \bmod 143 = 7$
      - La clef publique est N = 143, e = 11
    - Exemple 2 : soit e = 17
      - Alors d = 113, parce que 17 * 113 = 1921 = 16 * 120 + 1
      - Pour chiffrer un mot de code $x = 7 \rightarrow y = 7^{17} \bmod 143 = 50$
      - Pour déchiffrer $y = 50 \rightarrow x = 50^{113} \bmod 143 = 7$
      - La clef publique est N = 143, e = 17

L'algorithme RSA : exemple

- Avec des petites valeurs, on peut retrouver *d*
  1. En factorisant *N*
     - 143 ne peut pas être autre chose que 11 * 13
     - on peut le retrouver en 11 essais
  2. En calculant $\phi(N)$
  3. En retrouvant la clé privé d
     - en résolvant équation $e * d = 1 \; mod \; \phi(N)$, avec l'algorithme d'Euclide

Rappels arithmétiques (en nombres entiers)

- Plus grand commun diviseur (pgcd)
  - soit deux nombres a et b; le plus grand entier qui divise a et b sans reste est le pgcd de ces deux nombres
- Algorithme d'Euclides
  - Calcul du pgcd
    - Pour pgcd(3615807, 2763323)

3,615,807 = 1 * 2,763,323 + 852,484 a = m * b + r

a <= b; b <= r

2,763,323 = 3 * 852,484 + 205,871

852,484 = 4 * 205,871 + 29,000

205,871 = 7 * 29,000 + 2,871

29,000 = 10* 2871 + 290

2,871 = 9 * 290 + 261

290 = 1 * 261 + 29

261 = 9 * 29 + 0

    - pgcd(3615807, 2763323) = 29
  - Permet également de calculer les inverses multiplicatifs dans Z_N^*
  - Complexité = $O(n^3)$

- Exponentiation rapide

Permet de calculer $x^e \bmod N$ en temps $O(n^3)$

Comment trouver un nombre premier très grand

- L'algorithme standard - Crible d'Ératosthène
  - On examine tous les nombres en éliminant ceux qui sont un produit de deux facteurs différents de 1
- Test de primalité probabiliste
  - Tout nombre premier doit satisfaire deux conditions:
    - pgcd $(p, r) = 1$ et $J(r, p) \equiv r^{(p-1)/2} \bmod p$
    - r est un nombre plus petit que p
    - J est la fonction de Jacobi

$$J(r,p) = \begin{cases} 1 \text{ si } r = 1 \\ J(r/2)*(-1)^{(p*p-1)/8} & \text{si } r \text{ est pair} \\ J(p \bmod r, r)\,(-1)^{(r-1)*(p-1)/4} & \text{si } r \text{ est impair et} \neq 1 \end{cases}$$

  - si c'est satisfait, la probabilité que p soit premier est 50%
  - si on répète le test k fois avec succès, la probabilité est $1\text{-}2^{-k}$

Problème du log discret

- Propriétés mathématiques de Z_p
  - Tous les éléments de Z_p ont des inverses multiplicatifs, sauf 0
    - Donc, $Z_p^* = Z_p - \{0\}$
  - Il existe des éléments g dit *générateur* ou *racine primitive* tel que : $Z_p^* = \{g^0, g^1, \ldots , g^{p-1}\}$
    - Notes :
      - Il est possible de vérifier en temps polynomial si un élément g est un générateur.
      - Il existe un très grand nombre de générateurs dans Z_p^*
  - Définition : Le logarithme discret en base g de $a \in Z_p$ est l'entier x tel que $a = g^x \bmod p$
  - Hypothèse calculatoire : Il n'est pas possible de calculer le log discret en temps polynomial sans connaître la factorisation de p-1.

Algorithme de El-Gamal

- Génération de clé
  1. Trouver un grand entier premier p tel que p-1 a au moins un grand facteur premier (donc difficile à trouver)
  2. Choisir au hasard un générateur g de Z_p^* et un entier d
  3. Calculer la valeur $e = g^d \bmod p$
  4. Clé publique = (p, g, e), Clé privé = d

- Chiffrement/Déchiffrement − pour un message $x \in Z_p^*$
  - Choisir un entier $k \in [0..p\text{-}1]$ <u>au hasard</u>
  - $E(k,x) = (y_1, y_2) = (g^k \bmod p, xe^k \bmod p)$
  - $D(y_1, y_2) = y_2 / y_1^d \bmod p$

Algorithme de El-Gamal

- Intuition
  - Le message x est "masqué" dans y_2 en le multipliant par e^k par
  - La partie y_1

    fournit à qui connaît *d*, l'information nécessaire pour reconstruire *x* en "divisant" par y_1^d (en réalité, calculer son inverse et multiplier)

Algorithme de El-Gamal – Notes importantes

- Méthode de chiffrement dite "probabiliste"
  - il n'existe pas de chiffrement unique pour un même x.
- Choix de k
  - Ce n'est pas une clé *strictu sensu*
    - Il n'est pas nécessaire que Bob connaisse k en avance pour déchiffrer x
  - C'est plutôt un *masque*
    - Qui ne doit pas être dévoilée (!)
  - Très important de choisir un k différent à chaque fois
    - Un mauvais choix de k (toujours pareil, basse entropie) rend possible des attaques à texte clair choisi (dangereuses si entropie après codage basse)

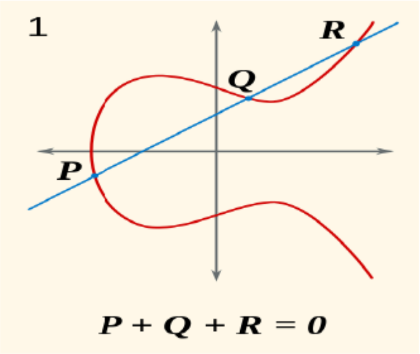
Notion de groupe - rappel

- Notion de groupe (G, *)
  - Un ensemble abstrait G sur lequel on a défini une opération abstraite "*" avec certaines propriétés :
    - Il existe un élément identité 1 : $a * 1 = a$
    - Associativité : $a * (b * c) = (a * b) * c$
    - Tout éléments à un inverse : $a * a^{-1} = 1$
    - (Commutatif): $a * b = b * a$, on dit que le groupe est "abélien" ou "commutatif"
  - Définition (exponentiation):
    - $a^n = a * a * \ldots * a$, n fois où n est un entier et (G,*) est un groupe abélien
  - <u>Note</u>: On peut définir le problème de log discret sur n'importe quel groupe
- Exemples
  - Z_p*, Corps fini (corps de Galois), Courbe elliptique

El-Gamal sur Corps de Galois

- GF(2^n)
  - Corps de Galois (corps fini) de taille 2^n
  - Se base sur l'arithmétique modulaire avec des polynômes de degré n
  - Toutes les coefficients des polynômes sont binaires
    - toute l'arithmétique est binaire
- El-Gamal sur GF(2^n)
  - Chiffrement et déchiffrement très efficaces
  - Surtout utilisée sur des plateformes matériel
  - Peu utilisé aujourd'hui

Cryptographie à courbes elliptiques (ECC)



- Courbe elliptique
  - Se base sur l'opération de "somme" de points sur une courbe elliptique
  - Combiné avec des opérations sur des corps de Galois
- ECC
  - Permet un niveau équivalent de sécurité avec des tailles de clés entre 6-12 fois plus petites (p.ex. 256 bit EC $\cong$ 3072 bit RSA)

  ➔ meilleure performance de chiffrement et déchiffrement

  ➔ taille réduite des signatures numériques pour applications avec peu de bande passante