

# INTRODUCTION À LA SÉCURITÉ INFORMATIQUE

## SÉANCE 11

*Marc-Olivier Killijian*  
*UQAM, CNRS*  
*INF4471*

# PROGRAMME DU JOUR

- D2 : correction en laboratoire
- Mots de passe
- Logarithme discret
- Échange de clé Diffie-Helman
- Signatures numériques et certificats

# MOTS DE PASSE

*Marc-Olivier Killijian*

*UQAM, CNRS*

*INF4471*

« La meilleure façon d'expliquer comment choisir un bon mot de passe, c'est d'expliquer comment on les casse »

*-Bruce Schneier*

# VULNÉRABILITÉS

- Les mots de passe dont un moyen très **pratique** d'authentifier quelqu'un
- Mais pas de garantie forte de sécurité
- Diverses attaques
  - **Rejeu** : un password observé peut être rejoué
  - **Faiblesse** : un mot de passe faible peut être deviné

# OBSERVATION

- **Keyloggers** très efficaces pour capturer des MDP
- **Logiciel** (e.g. Actual Keylogger, Beelogger dans Kali, etc.)
- **Matériel** : entre le clavier et l'ordinateur, un dispositif avec micro-controlleur, nvram, WiFi, BT, etc.



AirDrive Keylogger - Logger de Keylogger USB matériel avec Wi-Fi

Marque : AirDrive

★★★★★ 53 évaluations

Ancien prix: 57,98 CDN\$

Prix : 49,98 CDN\$

Économisez : 8,00 CDN\$ (14 %)

Recevez instantanément une carte-cadeau Amazon.ca de 15 \$ et jusqu'à 5 % de remise pendant six mois sur approbation pour la carte Mastercard Récompenses Amazon.ca.  
Payez 49,98 \$ 34,98 \$ pour cette commande sur approbation.

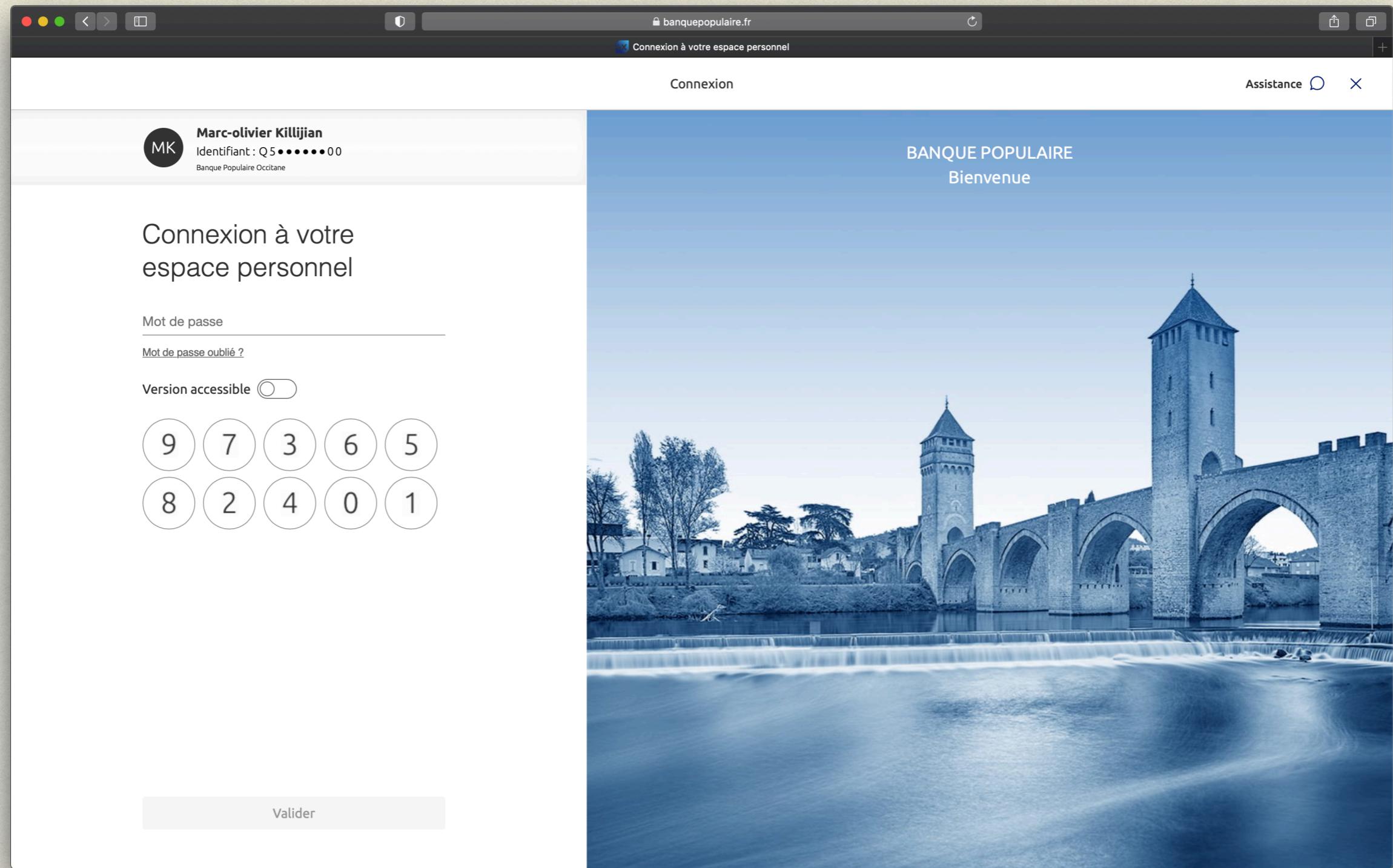
Nouveau (3) à partir de 49,98 CDN\$ + Livraison GRATUITE

- Le plus petit enregistreur de clé sur le marché, seulement 21 mm de long.
- Fonctionne comme un hotspot Wi-Fi, se connecte à partir de n'importe quel ordinateur, smartphone ou tablette.
- Accès aux données du clavier depuis le navigateur Web, aucun logiciel ou application nécessaire.
- Récupère les données à distance sans toucher l'appareil.
- Prend en charge plus de 40 configurations de clavier nationales

Avez-vous besoin de plus d'informations sur les produits en français?  
Contactez-nous

Signaler des informations incorrectes sur les produits

# CONTRE-MESURE KEYLOGGERS



# STOCKAGE DE MDP

- Gestionnaires de MDP (surtout ceux des butineurs) peuvent révéler les MDP (e.g. ordinateur public ou partagé)
- MDP envoyés par courriel, restent sur le serveur
- MDP tapés dans le champs utilisateur par erreur (historique accessible)

# AUTRES VULNÉRABILITÉS

- Partage de MDP (51% util. - Ponemon inst. 2020)
- Par dessus l'épaule : MDP complet ou partiel
- Post-it (surtout si changements réguliers)
- Ecoutes réseau (Wireshark)
- Ecrans tactiles



# CRAQUAGE DE MDP

- Craquer un MDP = essayer de le deviner
  - ~brute-forcer sauf si on a des indices
- Difficulté dépend de la taille, du format, de l'approche et des contre-mesures
  - NIP de carte bancaire (4 digits):  $10^4 = 10000$  mais limités à 3 essais
  - MDP unix {A-Z;a-z;0-9;&é »'!#@}  $10 \approx 95^{10}$  mais potentiellement plus d'essais possible

# CRAQUAGE DE MDP EN LIGNE

- Attaque en boite noire, essai de MDP, attente réponse (lent)
- Contre-mesures nombreuses
  - Blocage du compte après X essais ratés
  - **DoS** et couts de service
- CAPTCHAs
  - Protection faible

Origin	Samples	Success rate	Comments
Captchbservice.org [22]		92%	Weaknesses: constant pixel count of the same character, non-textured background, constant colors, no perturbation, only capital characters used
Clubic [3]		100%	Weaknesses: constant font, no rotation, no deformation, aligned glyph, constant background, weak color variation, weak perturbation
Slashdot [3]		89%	Weaknesses: constant font, no deformation, constant colors, weak perturbation, weak color variation
Microsoft [23]		60%	Weaknesses: easy to tell random arcs from valid characters by examining characteristics, easy to locate connected characters by 'even cut'
Google [8]		62%	Weaknesses: non-textured background, constant colors, no perturbation
NuCaptcha [21]		36.3%	Weaknesses: weak overlapping, constant color of 'codewords', short length of 'codewords', constant font
Megaupload [9]		78.3%	Weaknesses: shared components are white, non-textured background, constant font, short and fixed length, no perturbation

Gao et al. The Robustness of Hollow CAPTCHAs

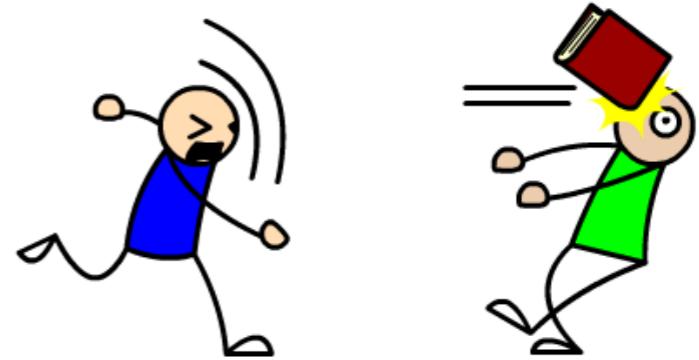
# STOCKAGE DES MDP

- Serveur doit pouvoir vérifier qu'un MDP est valide
  - ~~Stockage en clair des MDP ?~~
  - Stockage d'un tuple  $\langle \text{user}, h(\text{MDP}) \rangle$  avec un fonction de hachage cryptographique ?

# CRAQUAGE DE MDP EN LIGNE

- L'attaquant a obtenu le fichier de MDP
  - Fonction de hachage connue
  - Rechercher des MDP et tester leur haché
    - Recherche exhaustive :  $\approx 95^{\text{longueur}}$  (ou  $25^{\text{longueur}}$ ) au rythme de 200000 à 3 000 000 000 haché/s
    - Dictionnaire : 150000-200000 mots (noms personnes, animaux, mots courants) dont les hachés peuvent être pré-calculés !
    - Mémoriser les user/MDP trouvés pour d'autres sites

## DICTIONARY ATTACK!



Src

	$25^8$	$25^{10}$	$95^8$	$95^{10}$
	2E+11	1E+14	7E+15	6E+19
J à 1Mds hash/s	0,00	1,10	77	692983

# DICTIONNAIRES ET HEURISTIQUES

- Application de règles de transformation des mots du dictionnaire en fonction des pratiques
  - minuscules, majuscules, inverses (Fred->derF), duplications (FredFred), reflet (FreddieF), rotationD (dFre), rotationG (redF), ajout d'un caractère en fin, en début, etc.
- Automatisation, e.g. John the Ripper

# CONTRE-MESURES

- Choix de MDP difficiles (longs, complexes,  $\neq$ mots)
  - Mais pas trop difficile à retenir
- Salage des hachés : du bruit est ajouté au MDP avant hachage -> « impossible » de pré-calculer les hachés
- Renforcement de clé : fonction de hachage très lente avec salage répétée X fois ( $X > 1000$ )
  - ex: PBKDF2 (WPA, WPA2) - BitWarden  $X=200.000$
- Test de la difficulté des MDP

# TESTER LA DIFFICULTÉ D’UN MDP

- HaveIBeenPwned ?

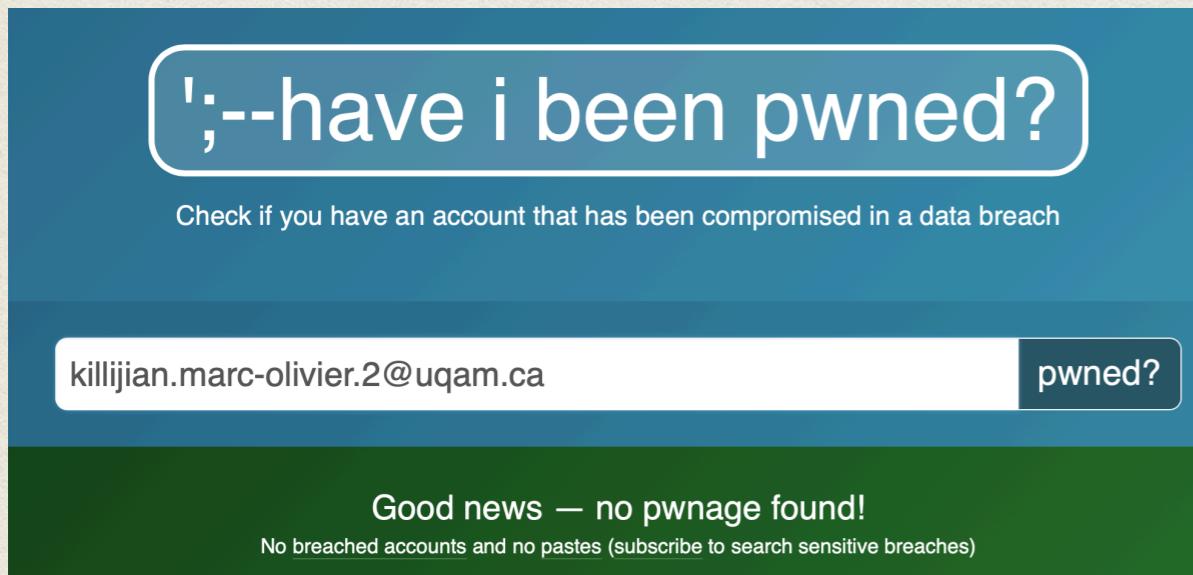
**';--have i been pwned?**

Check if you have an account that has been compromised in a data breach

killijian.marc-olivier.2@uqam.ca pwned?

Good news — no pwnage found!

No breached accounts and no pastes (subscribe to search sensitive breaches)



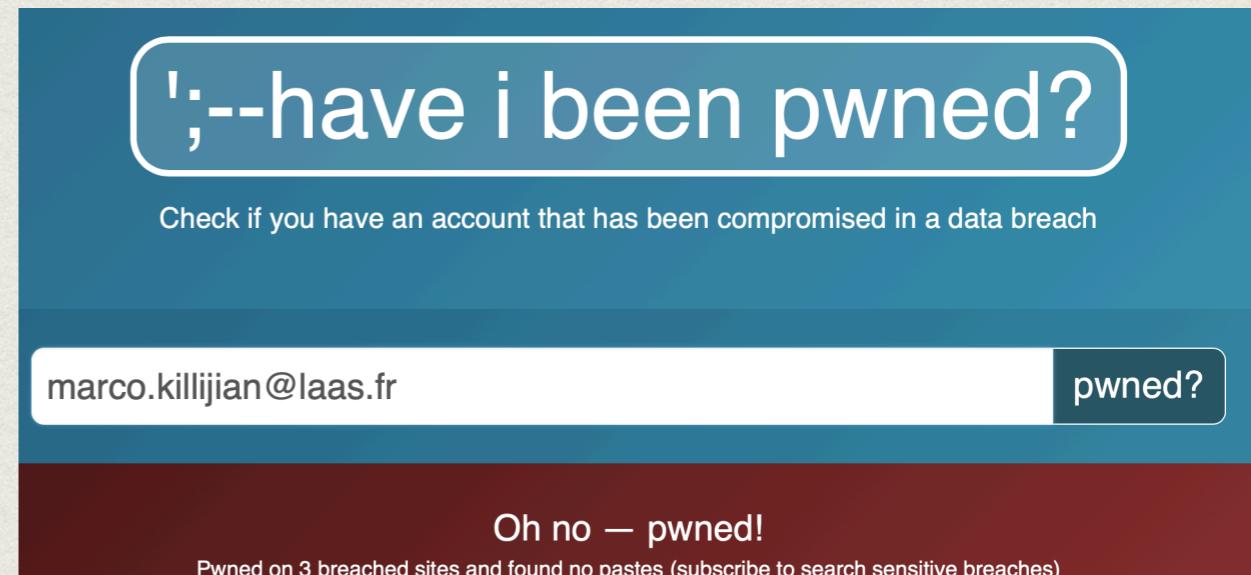
**';--have i been pwned?**

Check if you have an account that has been compromised in a data breach

marco.killijian@laas.fr pwned?

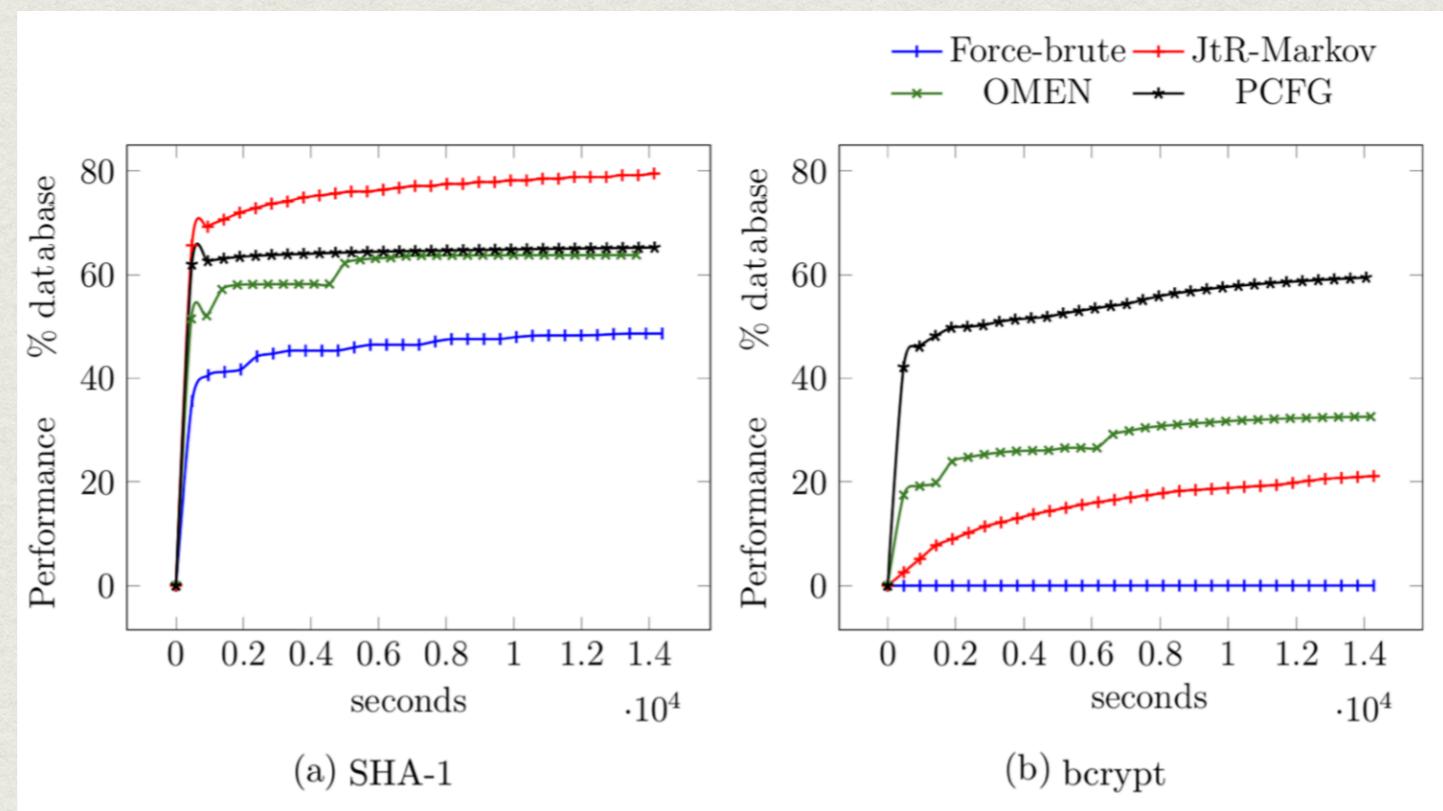
Oh no — pwned!

Pwned on 3 breached sites and found no pastes (subscribe to search sensitive breaches)



# TESTER LA DIFFICULTÉ D’UN MDP

- Outils de craquage de MDP
  - John The Ripper
  - PCFG : apprentissage machine pour adapter la stratégie à la base de données
  - OMEN : basé sur des modèles de Markov



# CONCLUSION

- Utiliser des mots de passe complexes et longs
- Ne pas utiliser un MDP sur différents services
- Tester son MDP
- Utiliser un gestionnaire de MDP  
**local et open source** : Keepass, Bitwarden  
**ou reconnu** : Apple Keychain (iCloud ...?)

# INTRODUCTION À LA SÉCURITÉ INFORMATIQUE LOG DISCRET ET DIFFIE-HELLMAN

*Marc-Olivier Killijian*

*UQAM, CNRS*

*INF4471*

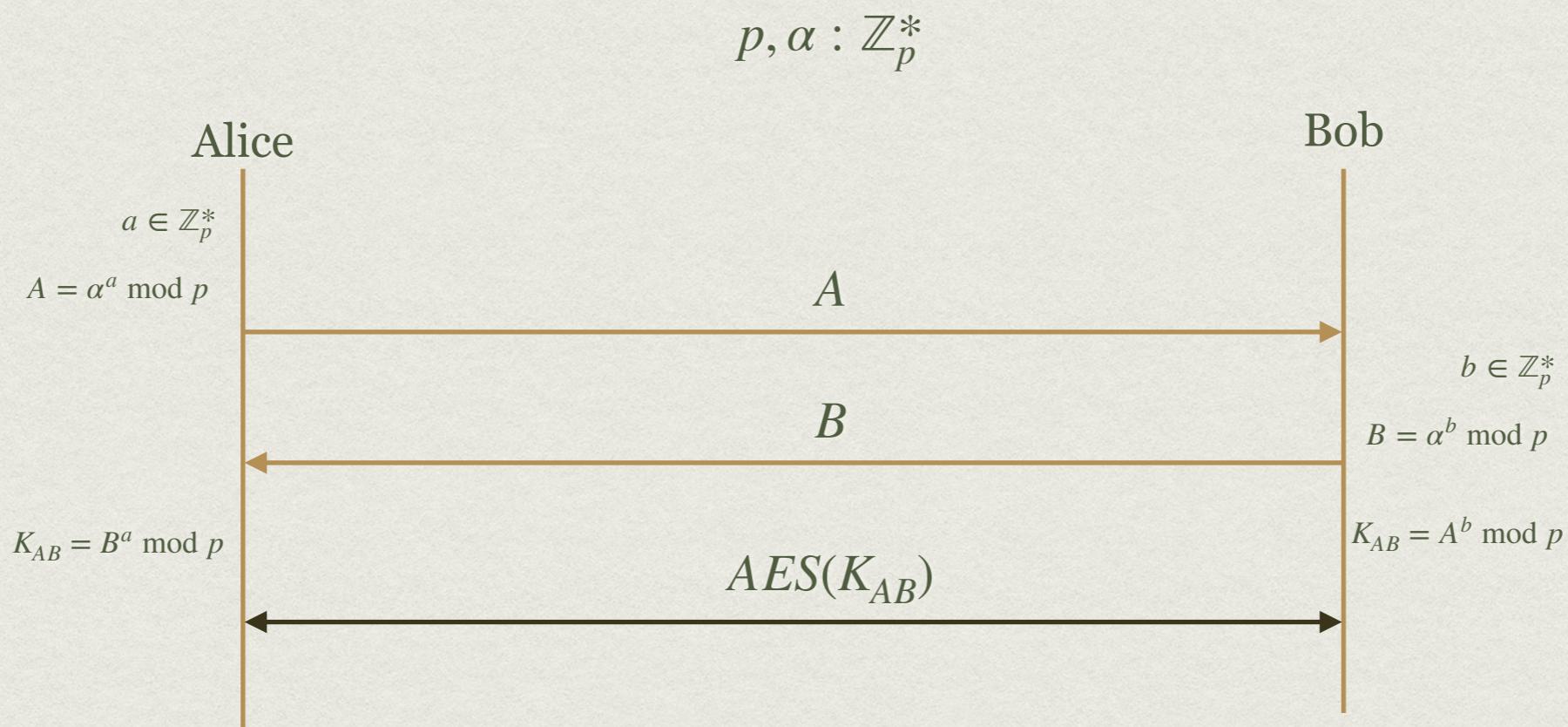
# LOG DISCRET

- Soit un groupe multiplicatif cyclique  $\mathbb{Z}_p^* = \{1, \dots, p-1\}$  l'ensemble des entiers mod  $p$  avec  $p$  un premier qui possède un générateur  $g$  tel que  $\mathbb{Z}_p^* = \{g^0, g^1, \dots, g^{p-2}\}$
- Facile de trouver  $p$  (de taille donnée) et  $g$  pour  $\mathbb{Z}_p^*$   
*par exemple pour  $p=13$ ,  $g \in \{2, 6, 7, 11\}$*   
$$g^{0..12} = \{1, 2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7\}$$
- **LDP:** soit  $y = g^x \pmod{p}$  il est difficile de trouver  $x$  à partir de  $(y, p, g)$
- Calculer le logarithme (discret) dans  $\mathbb{Z}_p^*$  est difficile

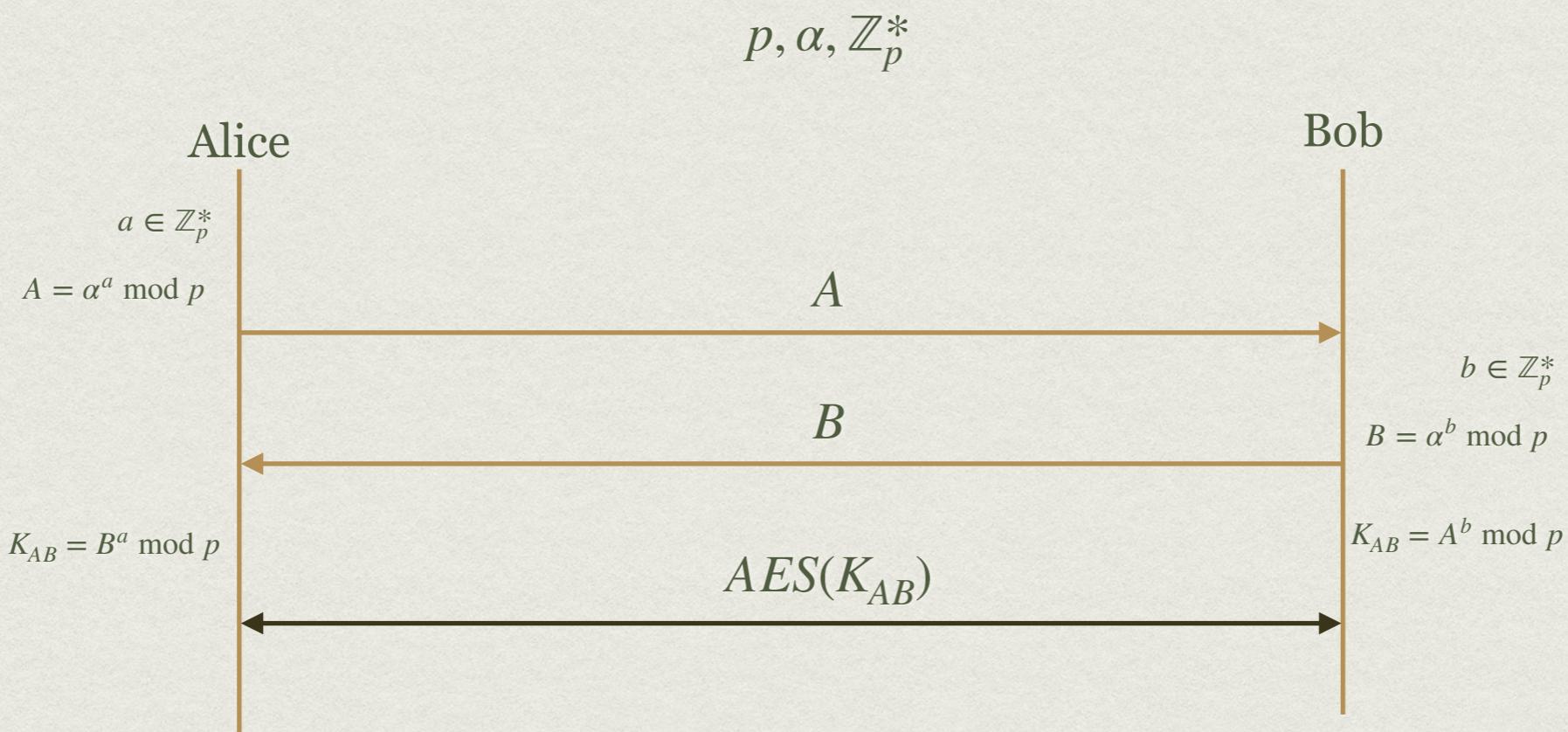
# DLP: PROBLEME DU LOG DISCRET

- Log discret  $\approx$  inverse de exponentiation modulaire
- A partir de  $(y,a,n)$  trouver  $x$  tel que  $a^x = y \pmod{n}$ 
  - $a$  : la base du logarithme
  - Log base 2 de 3 (mod 13) :  $2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 3$   
donc  $x=4$
  - Log base 3 de 4 (mod 13) :  $3^1 = 3, 3^2 = 9, 3^3 = 1, 3^4 = 1, \dots$  aucun
  - ... nombreuses multiplications
  - **DLP** : à partir de  $y = a^x \pmod{p}$  retrouver  $x = \log_a y \pmod{p}$

# ECHANGE DE CLÉ DIFFIE-HELLMAN

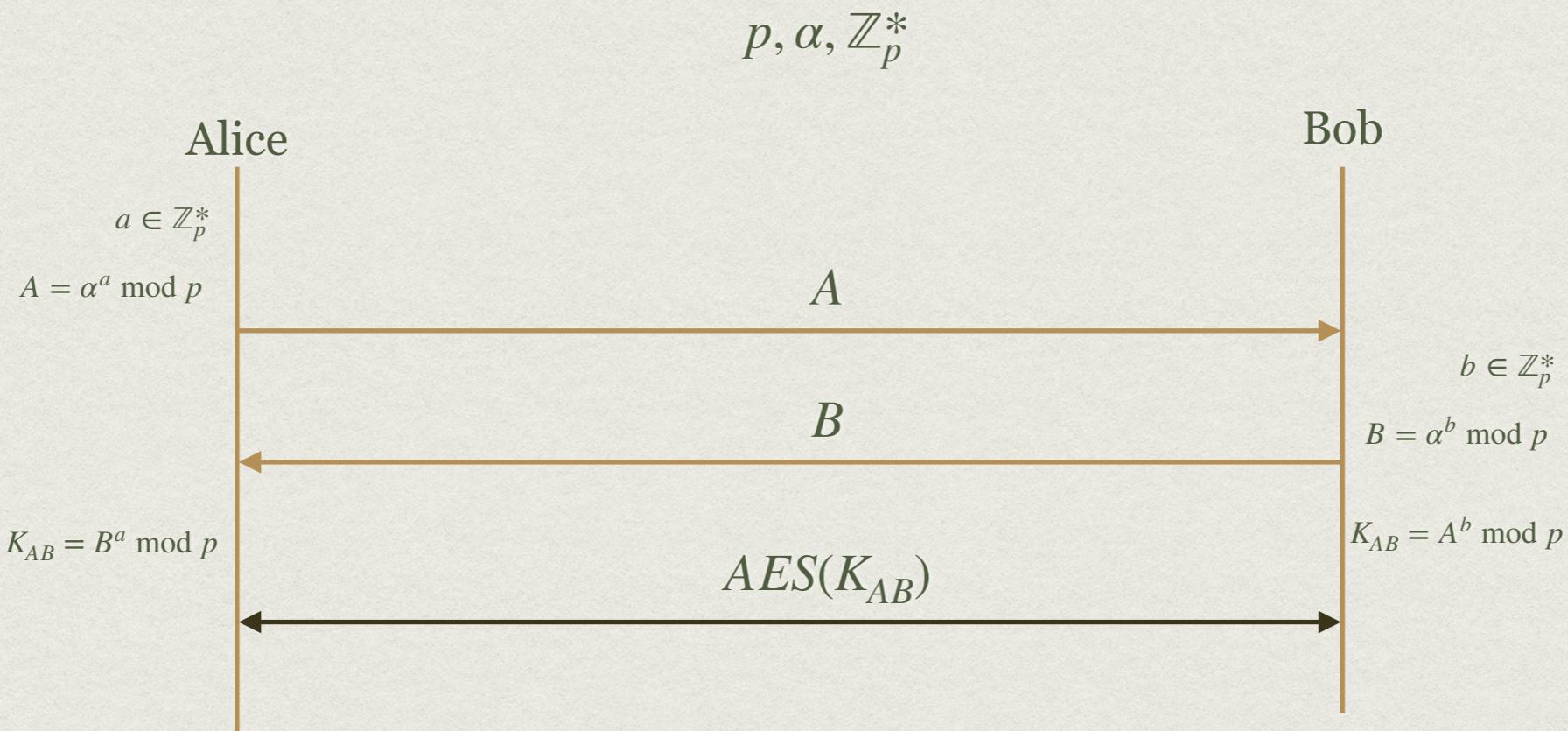


# ECHANGE DE CLÉ DIFFIE-HELLMAN



Preuve:  $K_{AB} = B^a \bmod p = (\alpha^b)^a \bmod p = (\alpha^a)^b \bmod p = A^b \bmod p = K_{AB}$

# ECHANGE DE CLÉ DIFFIE-HELLMAN



**DHP:** Adversaire passif qui connaît  $p, \alpha, \mathbb{Z}_p^*, A, B$  peut-il retrouver  $\alpha^{ab}$  ?

On ne sait le faire qu'en résolvant **DLP** : à partir de  $A = \alpha^a \bmod p$  retrouver  $a = \log_\alpha A \bmod p$

# ECHANGE DE CLÉ DIFFIE-HELLMAN

- Deux participants peuvent créer une clé symétrique partagée
  - Sur un canal non sécurisé
  - Sans relation de confiance préalable
- Utilisé dans TLS, IPSec VPN, SSH
- Clés de 512, 768, 1024, **2048** bits ...
  - La NSA est sûrement capable de casser 1024 bits en précalculant d'énormes tables en fonction de p

# INTRODUCTION À LA SÉCURITÉ INFORMATIQUE SIGNATURES, CERTIFICATS, PKI

*Marc-Olivier Killijian*

*UQAM, CNRS*

*INF4471*

# SIGNATURES

- Intégrité et authenticité d'un message (CAM)
- Authenticité de l'émetteur (CAM+Clé symétrique)
- N'importe qui peut vérifier l'origine du message avec une signature (clé asym.)



# SIGNATURE NUMÉRIQUE

- Chaque entité possède une paire  $\langle K_p, K_s \rangle$
- $\text{Sign}(K_s, m)$  génère une signature pour  $m$
- $\text{Verif}(K_p, m, s) \Leftrightarrow (s = \text{Sign}(K_s, m))$
- N'importe quelle entité connaissant  $K_p$  peut vérifier la signature

# PROPRIÉTÉS $\Leftrightarrow$ CRYPTO ASYM

- Difficile de dériver  $K_s$  à partir de  $K_p$  (même après avoir observé plusieurs messages signés)
- Fouille exhaustive possible et attaques existantes plus rapide → clés plus longues qu'en crypto symétrique
- Vérifier une signature → être certain de la clé publique

# SIGNATURE RSA\*

- Schéma de chiffrement RSA peut facilement être transformé en schéma de signature
- Bob a  $\langle K_p, K_s \rangle = \langle (n, e), d \rangle$
- Les vérificateurs ont  $\langle K_p = (n, e) \rangle$
- $\text{Sign}(K_s, m) = m^d \pmod{n} \equiv s$
- $\text{Verif}(K_p, m, s) \Leftrightarrow (s^e \pmod{n} \equiv m)$

# SÉCURITÉ RSA\*

- Eve veut faire croire que son message  $m^*$  vient de Bob
- Evaluer  $(m^*)^d \pmod{n}$  avec  $d = e^{-1} \pmod{\phi(n)}$
- Équivalent à trouver  $K_s = d$  : difficile, hypothèse RSA (factoriser premiers)

# INSÉCURITÉ RSA \*

- Attaque sans message : Eve peut trouver un message  $m^*$  qui pourrait venir de Bob (contrefaçon de signature)
  - Choisir  $s \in \mathbb{Z}_n^*$
  - Calculer  $m^* \equiv s^e \pmod{n}$
  - $(m^*, s)$  est un message valide car  $s^e \equiv m^* \pmod{n}$
- Pas besoin d'avoir observé de signature valide, seul  $K_p$  suffit
- $m^*$  n'a pas de sens mais à éviter !

# INSÉCURITÉ RSA\* (2)

- Attaque à message choisi : Eve peut trouver une signature « de Bob » pour un message  $m^*$  choisi
  - Choisir  $w \in \mathbb{Z}_n^*$
  - Faire signer Bob :  $w$  et  $w^{-1}m^*$  et obtenir resp.  $s_1$  et  $s_2$
  - Eve a alors  $s = s_1 \cdot s_2 \pmod{n}$  signature valide de  $m^*$   
$$s^e = (s_1 \cdot s_2)^e = s_1^e \cdot s_2^e = w \cdot m^* \cdot w^{-1} = m^*$$
- Il faut convaincre Bob de signer des messages mais à éviter !
- NB: utilise l'homomorphie multiplicative de RSA

# HACHE-ET-SIGNE RSA\*

- Si la signature RSA\* est faite sur un haché de message, alors les attaques sans-message et à-message-choisi sont plus difficile
- + efficace de signer des messages de taille fixe (sortie de H)

# DIGITAL SIGNATURE ALGORITHM DSA

- Algorithme de signature numérique
- Standardisé par le NIST en 1991
  - basé sur ElGamal (log discret)
  - clés de 3072 bits (équiv. 128bits symétrique)

# GÉNÉRATION DE CLÉS DSA

- Choisir des longueurs L et N - NIST 800-57 : L=3072 et N=256 -> sécurité eq. 128 bits
- H : fonction de hachage cryptographique (e.g. SHA256) de  $n$  bits
- $q \leftarrow$  nb premier aléatoire de n bits
- $p \leftarrow$  nb premier aléatoire de l bits où  $p - 1$  est multiple de  $q$  :  $p-1=qz$
- $g$  : générateur d'ordre multiplicatif  $q \pmod p$   
choisir  $1 < h < p - 1$  :  $g = h^z \pmod p$
- $x \in [0,q] \leftarrow$  nb aléatoire
- $y = g^x \pmod p$  (LDP)
- $K_p = (p, q, g, y)$  et  $K_s = x$

# SIGNATURE DSA

- $k \in [1, q - 1] \leftarrow$  nb aléatoire
- $r = (g^k \bmod p) \bmod q$
- $s = \frac{(H(m) + x \cdot r) \bmod q}{k}$
- $Sign(m, K_s) = (r, s)$

# VÉRIFICATION DSA

- si  $0 > r > q$  ou  $0 > s > q$  rejeter la signature
- $w = s^{-1} \pmod{q}$
- $u_1 = H(m) \cdot w \pmod{q}$
- $u_2 = r \cdot w \pmod{q}$
- $v = (g^{u_1} \cdot y^{u_2} \pmod{p}) \pmod{q}$
- accepterssi  $r==v$  rejeter sinon

# PREUVE DSA

- $r = v \Leftrightarrow (g^k \pmod p) \pmod q = (g^{u_1} \cdot y^{u_2} \pmod p) \pmod q$   
or  $(g^{u_1} \cdot y^{u_2} \pmod p) \pmod q =$   
 $(g^{H(m).w} \pmod q \cdot y^{r.w} \pmod q \pmod p) \pmod q =$   
 $(g^{H(m).w} \pmod q \cdot g^{x.(r.w) \pmod q} \pmod p) \pmod q =$   
 $(g^{w(H(m)+x.r)} \pmod q \pmod p) \pmod q =$   
 $(g^{(H(m)+x.r)/s} \pmod q \pmod p) \pmod q =$   
 $(g^{(H(m)+x.r)/(H(m)+x.r)} \pmod q \pmod p) \pmod q =$   
 $(g^k \pmod p) \pmod q$

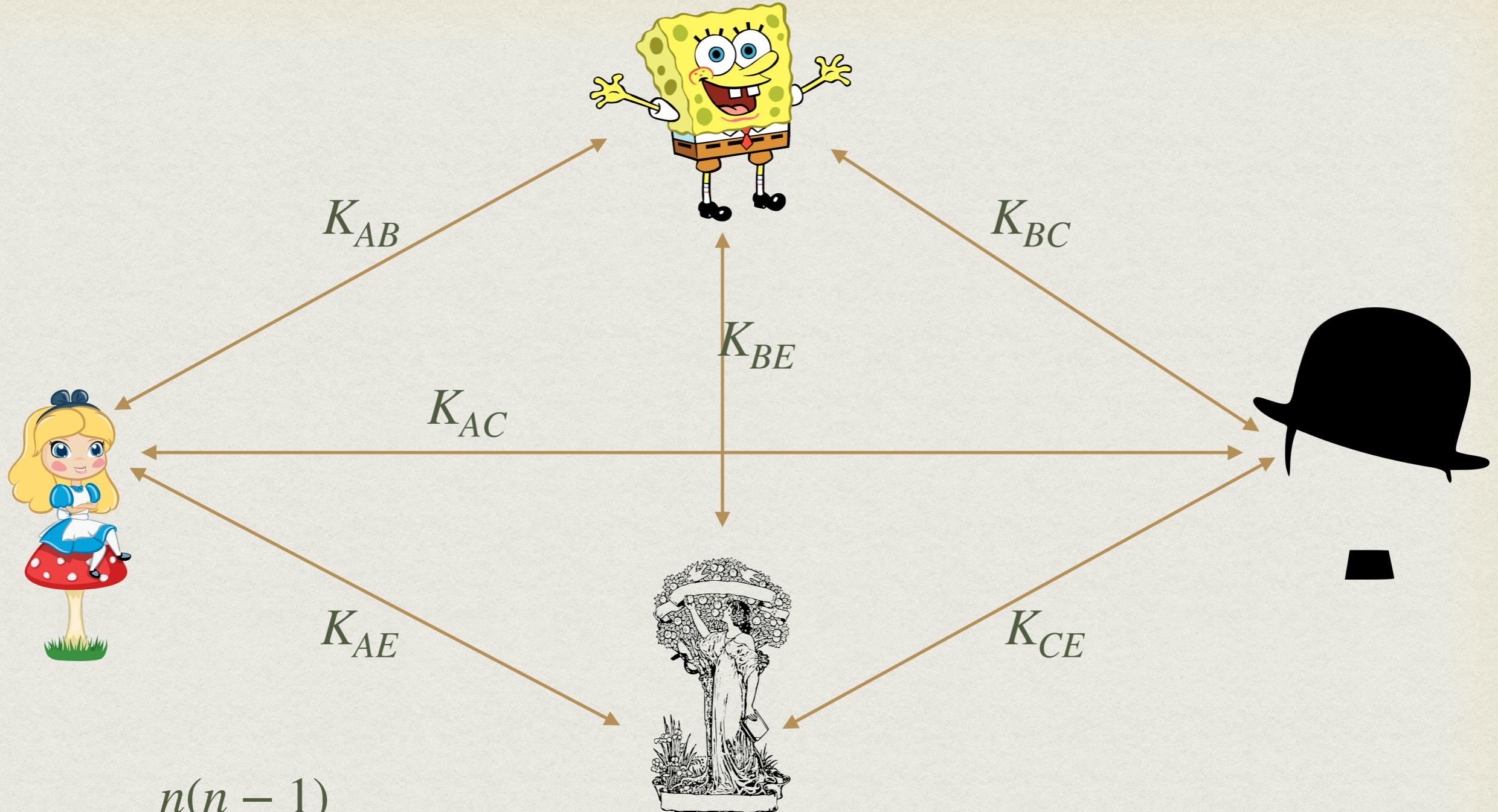
# NON-RÉPUDIATION

- Souvent signature numérique  $\Leftrightarrow$ signature manuscrite
  - Clés suffisamment longues et conservées en sécurité
  - On doit pouvoir signaler une clé compromise
    - ce qui doit invalider les signatures de cette ancienne clé
  - Risque : un signataire malveillant peut signaler une compromission pour invalider une signature préalable

# NON-RÉPUDIATION EN PRATIQUE

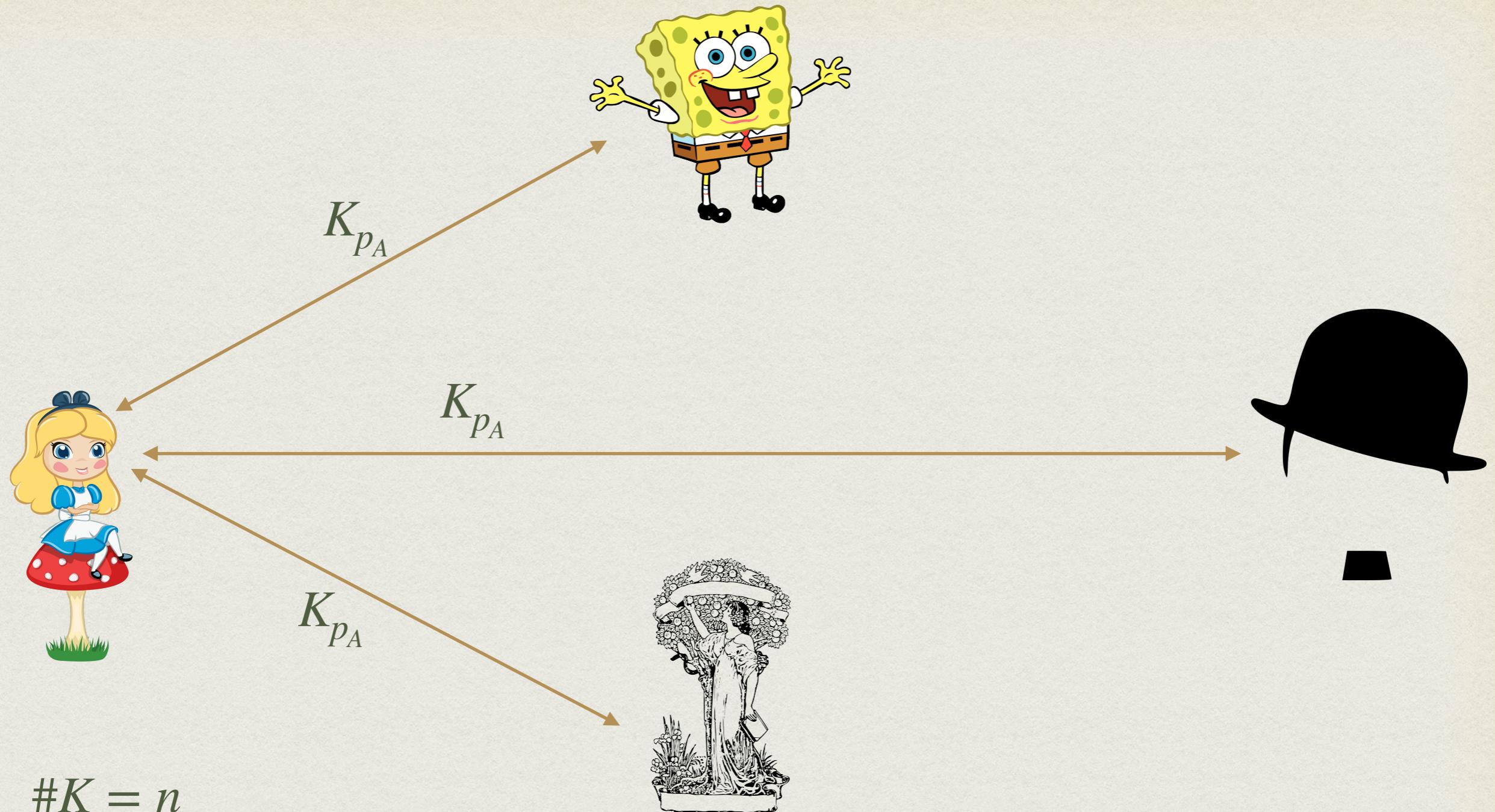
- Service d'horodatage sécurisé qui contresigne les documents
- Et/ou faire signer (papier) un code de bonne conduite

# CLÉS SYMÉTRIQUES



$$\#K = \frac{n(n - 1)}{2}$$

# CLÉS ASYMÉTRIQUES



# GESTION DE CLÉ

- Problème de la distribution de toutes ces clés
- Administration
- Révocation : gestion des clés compromises
- Mise-à-jour périodique pour mitiger la perte de sécurité des clés utilisées longtemps

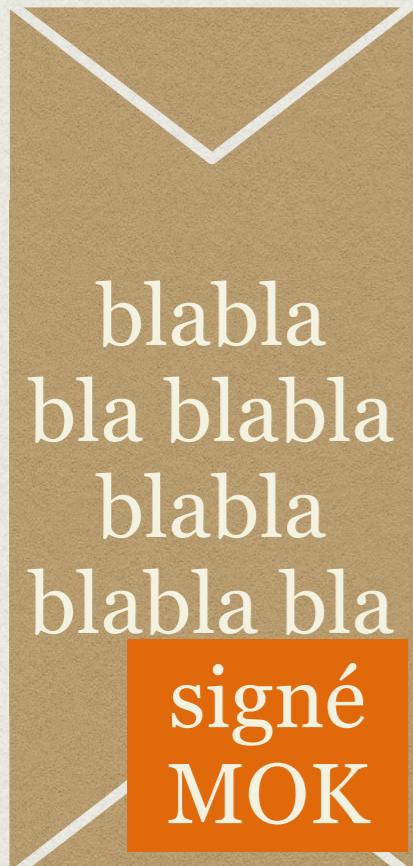
# GESTION DE CLÉ

- Centre de distribution de clés symétriques
- Un tiers de confiance a une paire de clé avec chaque utilisateur
  - génère à la demande des clés pour que les utilisateurs communiquent entre eux deux-à-deux
  - ou est utilisé dans un protocole d'authentification lors duquel une clé de session est générée

# CERTIFICAT

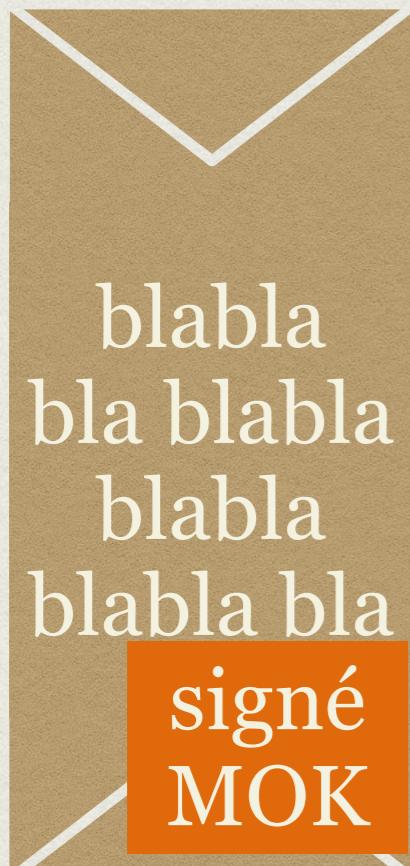
- Objectif : lier une **clé publique** à son **propriétaire**
- La paire <clé publique, propriétaire> est signée par une **partie de confiance**
- Qu'on appelle **Autorité de Certification** (CA)
- Vérifier la signature ← clé publique de la CA !!
- La paire  $\langle k_{p_{CA}}, \text{CA} \rangle$  est auto-signée : **certificat racine**
- L'authenticité du certificat racine est **essentielle**
  - e.g. les butineurs viennent avec un trousseau de certificats racines au **standard X.509**

# CERTIFICAT



Message signé

# CERTIFICAT

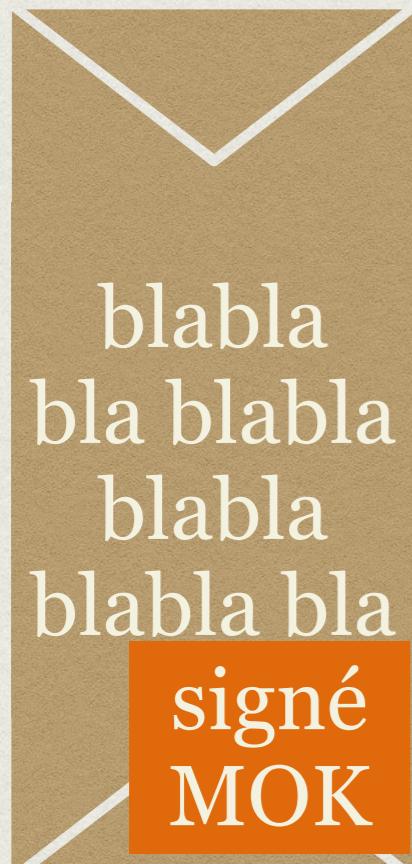


Message signé



Certificat

# CERTIFICAT



Message signé



49



Certificat racine

# CONCLUSION

- Grâce à une signature numérique, n'importe qui peut vérifier la signature (pas le cas avec un CAM)
- Clé publique (asym) moins efficace que clé secrète (sym)
- Important de **vérifier l'authenticité de la clé**
  - Et celle de **l'autorité de certification ne doit pas être compromise**
  - Téléchargez vos **navigateurs Web de sources sûres**, vérifiez les hashes ...