TMS Cryptography Pack

Contenu et démo

17 mai 2017

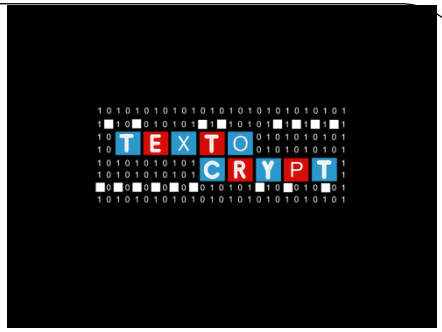
Marion CANDAU

marion.candau@cyberens.fr
05 56 58 62 45
Tour 6
74, rue Georges Bonnac
33000 Bordeaux

Plan de la session

- Présentation de Cyberens
- Introduction à la crypto
- Présentation de TMS Cryptography Pack
- Démonstration de plusieurs fonctions en mode client/serveur

Cyberens



- 2 personnes : Bernard Roussely, gérant et Marion Candau, ingénieure
- Depuis 2010
  - Conseil en cybersécurité

- Depuis 2015
  - Applications à base de cryptographie
    - Confidentialité, Intégrité, Traçabilité / preuve (signature)
  - Bibliothèque spécifique pour les objets connectés
    - Algorithmes cryptographiques
    - Gestion de certificats de sécurité
  - Chiffrement de SMS (TextoCrypt™ sur Google Play)
  - Chiffrement de messages/répertoires/fichiers (en cours)

Marion Candau

- Docteure en mathématiques appliquées à la cryptologie
- Travaille depuis 2 ans et demi chez Cyberens
- Chargée de la R&D
- Code initialement en C++
- Depuis 1 an, code en Delphi pour TMS Cryptography Pack
- Twitter : @marioncandau

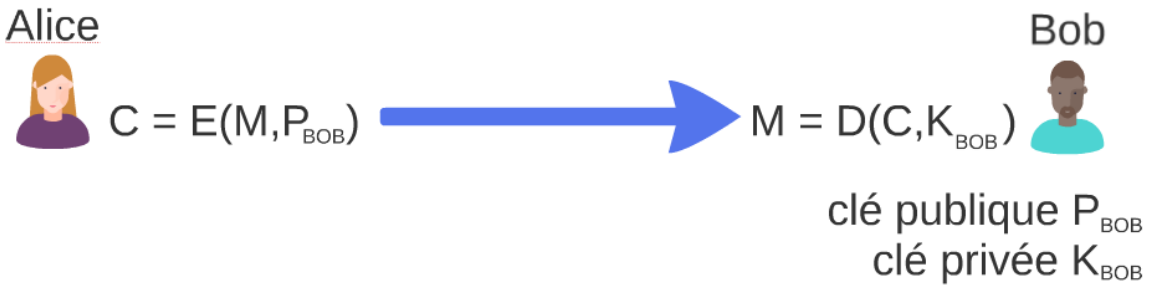
Crypto ?

- Cryptologie = science du secret
- Plusieurs objectifs :
  - Rendre confidentiel
  - Signer des documents
  - Comparer des secrets sans les sauvegarder
  - Générer une clé à partir d'un secret

Rendre confidentiel

- Deux types d'algorithmes : à clé symétrique ou asymétrique
  - À clé symétrique :

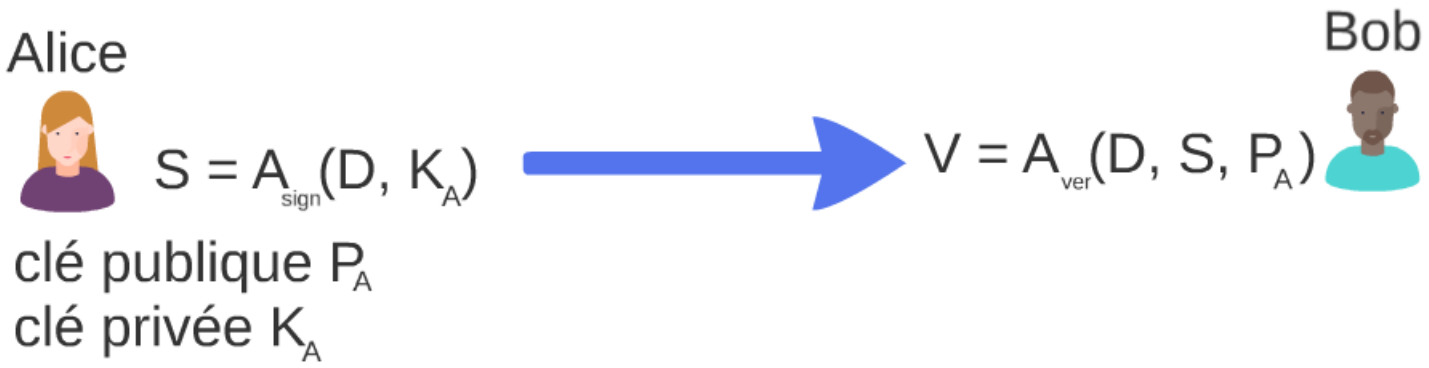
  - À clés asymétriques :



- Usage : on chiffre la clé symétrique avec les clés asymétriques et on utilise la clé symétrique ensuite pour chiffrer les messages

Signer des documents

- Garantit l'intégrité du document et authentifie l'auteur
- Utilisation des clés asymétriques :



Hacher et générer des clés

- fonctions de hachages : pour comparer des mots de passe sans les sauvegarder
- Facile de calculer « mot de passe -> condensat » mais difficile voir impossible de faire l'inverse « condensat -> mot de passe »
- Possibilité pour 1 attaquant d'utiliser un dictionnaire de mot de passe

- fonctions de dérivations de clés : pour générer des clés symétriques à partir d'un mot de passe.
- Plus difficile d'utiliser un dictionnaire de mot de passe, car un mot de passe = plusieurs clés possibles

TMS Cryptography Pack (1/3)

- Bibliothèque cryptographique complète
  - Delphi / C++Builder (le code est en C)
  - Développée par CYBERENS
  - Distribuée par TMS Software
  - Multi-plateforme

TMS Cryptography Pack (2/3)

- Algorithmes symétriques
  - AES
  - SPECK (IoT)
  - SALSA 20
  - ~~DES, 3DES~~
- Algorithmes asymétriques
  - RSA 2048 à 4096
  - ECC (2 courbes)
  - ~~RSA 512, RSA 1024~~
- Fonctions de hachage
  - SHA 2
  - SHA 3
  - BLAKE 2b
  - RIPEMD-160
  - ~~RC4, MD5, SHA1~~
- Fonctions de dérivation de clé
  - PBKDF 2
  - Argon 2
- Fonctions de conversion
  - Base 64, Base64url, Base32
  - Etc.

TMS Cryptography Pack (3/3)

Méthode	Date	Symétrique	Factorisation Module	Logarithme discret Clef	Logarithme discret Groupe	Courbe elliptique	Hash
[1] Lenstra / Verheul	2017	83	1717 1344	147	1717	157	166
[2] Lenstra mise à jour	2017	80	1300 1435	159	1300	159	159
[3] ECRYPT II	2016 - 2020	96	1776	192	1776	192	192
[4] NIST	2016 - 2030	112	2048	224	2048	224	224
[5] ANSSI	2014 - 2020	100	2048	200	2048	200	200
[6] IAD-NSA	-	256	3072	-	-	384	384
[7] RFC3766	-	-	-	-	-	-	-
[8] BSI	2017 - 2022	128	2000	250	2000	250	256

TMS CP 128 à 256 2048 à 4096 256 et 512 + de 224

Exemple (1/5)

- Chiffrement symétrique avec AES:

```
aes := TAESEncryption.Create;
aes.AType := atcbc;
aes.KeyLength := kl128;
aes.OutputFormat := base64;
aes.key := 'hjdks56HDfUf5JAE';
aes.IVMode := TIVMode.rand;
aes.paddingMode := TPaddingMode.PKCS7;
aes.Unicode := yesUni;
s:= aes.Encrypt('test ');
aes.Free;
```

- Alternative: SPECK, Salsa 20

Exemple (2/5)

- Chiffrement asymétrique avec la courbe elliptique d'Edwards 255-19:

```
ecc := TECCEncSign.Create;
ecc.ECCType := cc25519;
ecc.OutputFormat := hexa;
ecc.PublicKey := BobPublicKey; //en hexa
ecc.Unicode := yesUni;
ecc.NaCl := naclNo;
s := ecc.Encrypt('test');
ecc.Free;
```

- Alternative: RSA
- Les clés sont aussi utilisées pour signer et vérifier des signatures.

Exemple (3/5)

- Fonction de hachage SHA2: génère un condensat

```
sha2 :=TSHA2Hash.Create;
sha2.HashSizeBits := 256;
sha2.OutputFormat := base64url;
sha2.Unicode := yesUni;
s := sha2.Hash('test');
sha2.Free;
```

- Alternative: SHA3, Blake2B, RIPEMD-160

Exemple (4/5)

- Fonction de dérivation de clé, Argon2: on génère une clé à partir d'un mot de passe

```
argon2 := TArgon2KeyDerivation.Create;
argon2.OutputFormat := base32;
argon2.OutputSizeBytes := 16;
argon2.Counter := 10;
argon2.Memory := 16;
argon2.StringSalt := '0123456789012345';
argon2.Unicode := yesUni;
k := argon2.GenerateKey('password123:');
argon2.Free;
```

- Alternative: PBKDF2

Exemple (5/5)

- D'un mot de passe à un échange de clés

```
argon := TArgon2KeyDerivation.Create(16, '0123456789012345', 10, raw, 16, yesUni);
aesKey := argon.GenerateKey('passWord123:');
ecc := TECCEncSign.Create(cc25519, 'D0192BF57EF4625FA2B1DF3564C494BB31C5B7483243F21933A7B65708581A71', NaClno, hexa, yesUni);
eccCipher := ecc.Encrypt(aesKey);
aes := TAESEncryption.Create(kl128, aesKey, atcbc, TPaddingMode.PKCS7, hexa, yesUni);
aesCipher := aes.Encrypt('message');
argon.Free;
ecc.Free;
aes.Free;
```

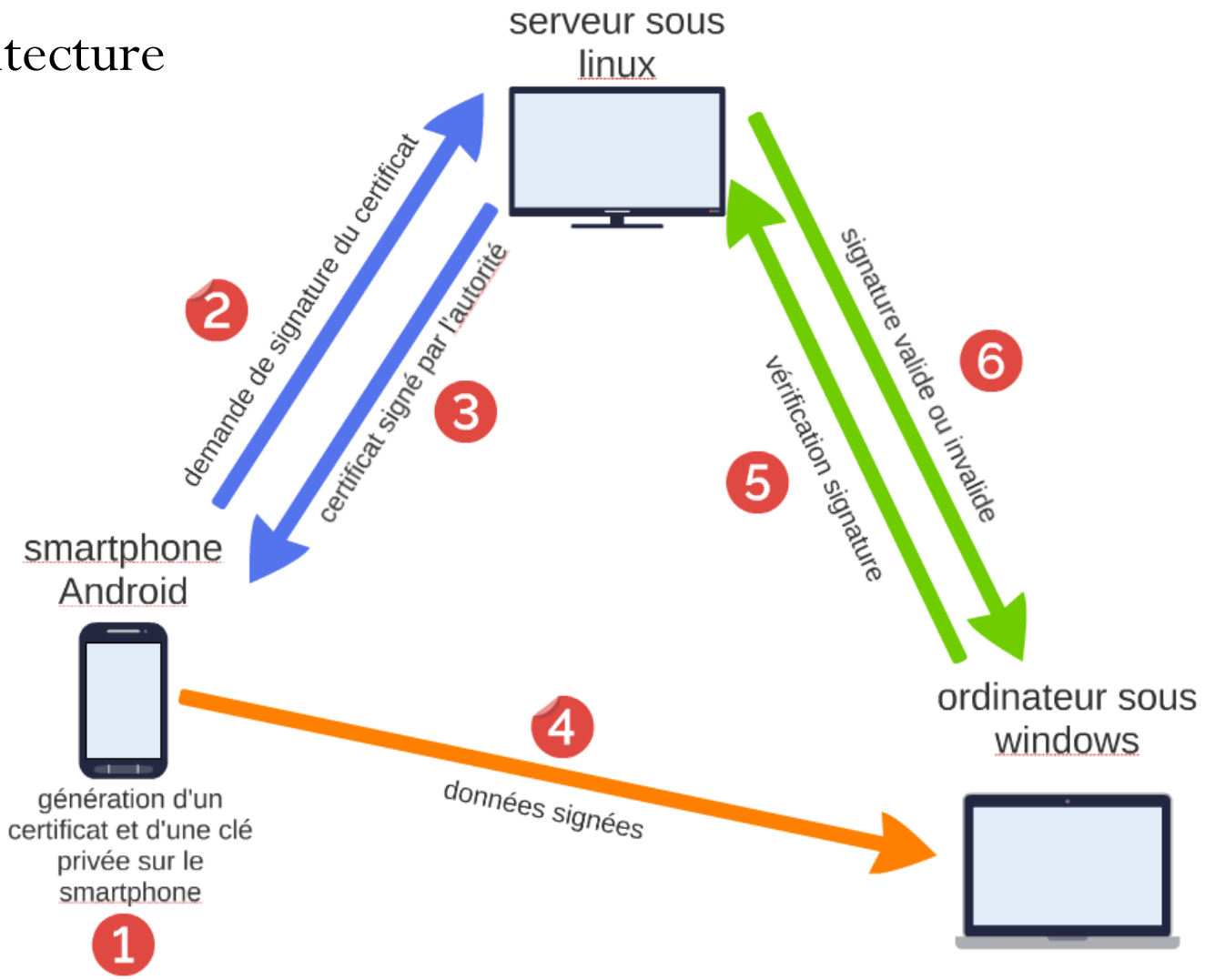
On envoie eccCipher et aesCipher à Bob. On utilise AES pour chiffrer d'autres messages.

Présentation de la démo (1/3)

- Objectifs
  - Authentifier les données provenant d'un smartphone
  - Sans diffuser le certificat du smartphone à toutes les parties
  - Utiliser une autorité de certification, qui signe le certificat du smartphone (garante de son identité)

Présentation de la démo (2/3)

- Architecture



Présentation de la démo (3/3)

- Vérification de la signature en pratique:
  - Envoi de l'adresse mail du signataire + texte signé + signature
  - Sur le serveur : extraction des données
    - Avec l'adresse mail, on identifie le certificat qui a signé
    - On vérifie que ce certificat est valide, non expiré et non révoqué.
    - On utilise SHA2 pour générer un condensat du texte signé
    - On utilise la fonction de vérification de signature du certificat pour vérifier la signature du condensat.

Conclusion

- TMS CP est une bibliothèque ne contenant que des algorithmes réputés forts
- 9 MAJ depuis sa sortie le 20 juin 2016
  - 2 bogues d'algorithme (dont un trouvé par Cyberens) sur les 20+ répertoriés
  - Support technique efficace
  - Traitement de 95% des demandes
- Roadmap 2017
  - Amélioration du traitement de l'unicode
  - Ajout de fonctions d'interopérabilité