Cryptologie: Compléments

robert.erra@epita.fr

2017

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

Secrets partagés

1. Algorithme de partage de secret à seuil (T, T) de Shamir
2. Algorithme de partage de secret à seuil (T, W) $(T < W)$ de Shamir (*Shamir's Secret Sharing Scheme*)

Système à seuil (T, W) ([Sti94]

Soient T, W deux entiers (> 0) tels que $t \leq W$. Un système à seuil de partage de secret est une méthode de partage d'une clé K entre W participants tels que :

- Tout groupe de T participants ($\neq$) peut calculer la clé K
- Mais aucune association de $(T-1)$ ou participants moins ne peut calculer K.

Sysème à seuil : *Threshod scheme*. $\mathcal{P}$: ensemble des participants.

Système à seuil (T, T) ([Sti94]

C'est un cas particilier du système à seuil (T, W) où $T = W$.

- D : initiateur, choisit p (pas forcément premier ici).
- D tire au hasard $(T-1)$ valeurs $\{y_i\}_{i=1}^{T-1} \in \mathbb{Z}_p$.
- Chaque valeur y_i est donnée au participant P_i.
- D calcule $y_T = \sum_{i=1}^{T-1} y_i \in \mathbb{Z}_p$.
- La valeur y_T est donnée au participant P_T.
- Ainsi chaque association de $T-1$ participant $\mathcal{P}P_i$ qui voudrait se liguer contre le dernier doit trouver la valeur qui « manque », soit $K - y_i$.
- Ils n'ont aucune information et doivent donc faire une recherche exhaustive.

Système à seuil (T, W) ([Sti94]

On revient au cas général : système à seuil (T, W) où $T < W$.

- D choisit un nombre premier p « assez » grand.
- D : choisit W valeurs $x_i \in \mathbb{Z}_p$ et attribue x_i à P_i.
- Les valeurs $x_1, \cdots x_W$ peuvent être publiques.
- D tire au hasard $(T-1)$ valeurs $\{a_i\}_{i=1}^{T-1}$ (indépendantes)
- On définit le polynôme $Q(x) = K + \sum_{j=1}^{T-1} a_j x^j \bmod p$.
- Pour chaque valeur x_i , D calcule $y_i = Q(x_i)$ qui est donnée au participant P_i.

Calcul de la clé

- Soient T participants qu désirent calculer la clé K.
- On peut calculer le polynôme $Q(x)$ (simple interpolation à faire, par exemple avec les formules de Lagrange).
- On en déduit $K = Q(0)$.
- Soit on calcule la valeur de $Q(0)$ directement par exemple avec l'algorithme de Neville-Aitken ([WIK]) adapté à l'arithmétique dans $\mathbb{Z}_p$.

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

Identification

- Alice désire s'identifier auprès de Bernard
- Elle désire qu'aucune des informations qu'elle donnera à Bernard ne pourra servir à une usurpation d'identité à un adversaire
- Mais elle désire aussi ne pas permettre à Bernard d'usurper son identité
- Idée : protocole utilisable dans une carte à puce [ne pas donner son PIN au DAB].

1er procotole à base de cryptographie symétrique (DES par exemple)

Algorithme 1 : Protocole Question-Reponse

Acteurs : Bernard et Alice ;

Outils : une cle K connue de Bernard et Alice
et un algorithme symetrique E_K sur n bits ;

Objectif : Obtenir une preuve de l'identite d'Alice (pour Bernard)

Bernard choisit une valeur x aleatoire sur n bits

Bernard envoie x a Alice

Alice calcule $y' = E_K(x)$ et l'envoit a Bernard

Bernard compare y' et $E_K(x)$: $y' \stackrel{?}{=} E_K(x)$

On peut iterer le procede ce qui diminue la probabibite d'erreur.

Fin.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Procédé d'Identification de Schnoor

- Le protocole précédent nécessite une clé connue de Alice et Bernard
- Le procédé d'Identification de Schnoor suppose
  - TA= Tiers de Confiance (autorité supposée honnête)
  - p : premier assez grand ($> 2^{512}$) tel que DLP soit difficile dans $\mathbb{Z}_p^*$.
  - q :grand facteur premier de $p-1$.
  - $\alpha \in \mathbb{Z}_p^*$ d'ordre q.
  - un paramètre t tel que $q > 2^t$ avec $t > 40$ voire $t > 80$.
  - un algorithme de signature $sig_{K_{TA}}$ et son procédé de vérification $ver_{K_{TA}}$
  - Une fonction de hachage sûre $H()$.

Certificat d'Alice

- TA calcule une « identité numérique » : chaîne ID(A) composée d'informations personnelles
- Alice choisit un exposant aléatoire a tel que $0 < a < q$ (q grand)
- Alice calcule $v = \alpha^{-a} \bmod p$ et le donne à TA
- TA calcule la signature $s = sig_{K_{TA}}(ID(A), v)$
- Le certificat $C(Alice) = (ID(A), v, s)$ est transmis à alice.

Procédé d'Identification de Schnoor

- Alice reçoit un certificat de TA (voir avant)
- Bernard veut vérifier l'identité d'Alice
- Il utilise par exemple le protocole appellé *procédé de Schnorr*.

2d procotole à base de cryptographie asymétrique

Algorithme 2 : Procede de Schnorr

Acteurs : Bernard et Alice ;

Outils : $C(Alice) = (ID(A), v, s)$;

Objectif : Obtenir une preuve de l'identite d'Alice (pour Bernard)

- Alice choisit k au hasard : $0 < k < q - 1$ et calcule $\gamma = \alpha^k \bmod p$;
- Alice envoie $C(Alice)$ et γ A Bernard
- Bernard verifie la signature $ver_{K_{TA}}(ID(A), v, s)$
- Bernard choisit un r aleatoire : $0 < r < 2^t$ et l'envoie a Alice
- Alice calcule $y = k + ar \bmod q$ et envoie y a Bernard
- Bernard verifie que $\gamma = \alpha^y v^r \bmod p$
- On peut iterer le procede ce qui diminue la probabilite d'erreur.

Fin.

Partage de secrets

Procédés d'Identification

The TLS/SSL Protocol

TLS/SSL Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Preuve

- $\alpha^y v^r \equiv \alpha^{k+ar} v^r \bmod p$
- $\alpha^y v^r \equiv \alpha^{k+ar} v^r \equiv \alpha^{k+ar} \alpha^{-ar} \bmod p$
- $\alpha^y v^r \equiv \alpha^{k+ar} v^{-ar} \equiv \alpha^k \equiv \gamma \bmod p$
- Soit : $\alpha^y v^r \equiv \gamma \bmod p$
- Cette propriété est appellée consistance [*completeness*)
-

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

TLS (cf RFC 5246)

1. SSL : Secure Socket Layer
2. SSL : originally developed by Netscape.
3. Netscape's patent buyed in 2001 by l'IETF
4. Now called TLS : *Transport Layer Security*.
5. TLS is a client-server model.

Transport Secure Layer : objectives

1. Authentification : Who is on the other side ?
   - Server Authentication — required [*via* PKC and possibly with a X.509 certificate]
   - Client authentication — optional
2. Confidentiality of all data exhanged : *via* symmetric cryptography
3. Integrity : messages integrity *via* hash functions
4. Spontaneity : any Client contacts Server (possibly for the first time) without problem
5. Transparency : You do not want to know about security, so you do not have to modify http, you just use it.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

SSL (cf RFC 6101) : The SSL protocol provides connection ...

... security that has three basic properties :

1. The connection is **private**. Encryption is used after an initial handshake to define a secret key. Symmetric cryptography is used for data encryption (e.g., DES [obsolete], 3DES , AES, FORTEZZA, RC4).
2. The peer's identity can be **authenticated** using asymmetric, or public key, cryptography (e.g., RSA , DSS).
3. The connection is **reliable**. Message transport includes a message integrity check using a keyed Message Authentication Code (MAC) [RFC2104]. Secure hash functions (e.g., SHA, MD5) are used for MAC computations.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

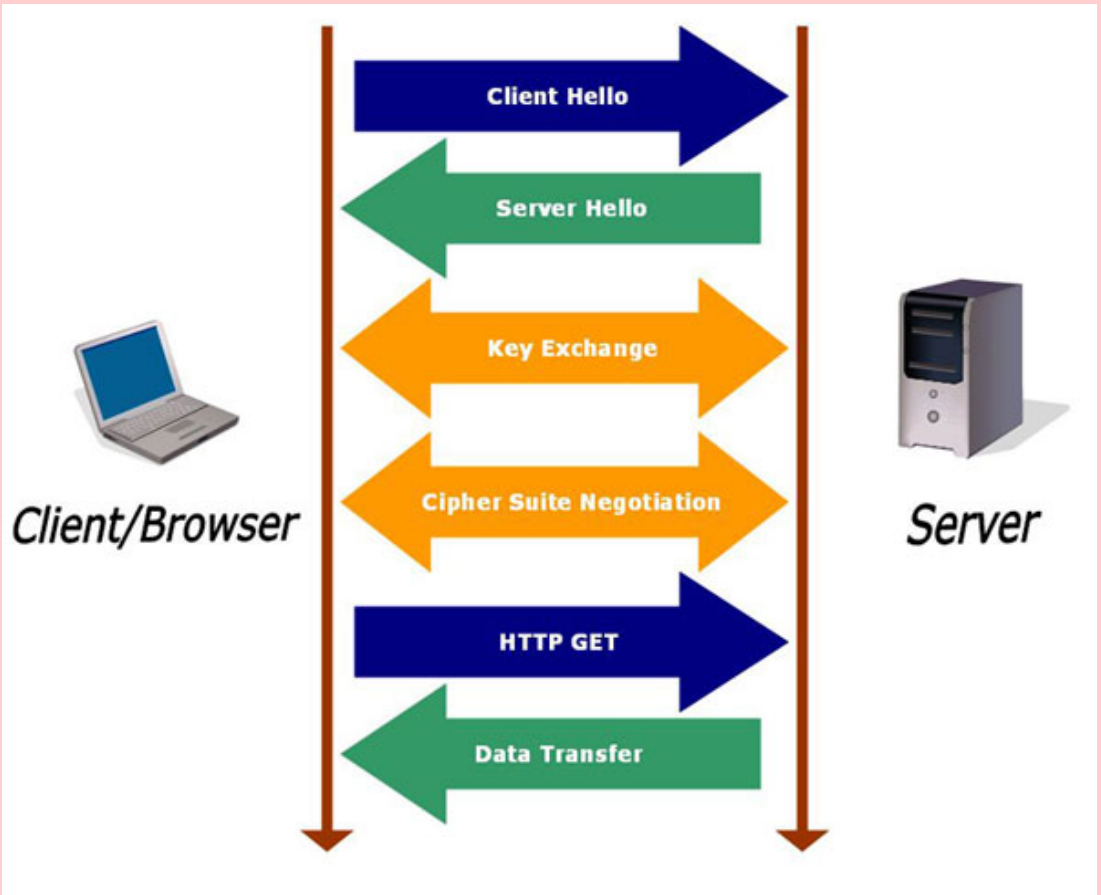
Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Image ! (cf ssl.trustwave.com)



EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

CipherSuites

1. Suites used to choose algorithms :
   - Key Exchange
   - Authentication
   - Encryption
   - MAC : Message Authentication Code [MAC is a special digest, which incorporate a key into the computation of the digest. The MAC value is dependent on both the message and the key.]
2. Construction : Kx-Auth-Enc-MAC
3. Examples :
   - DHE-RSA-AES128-SHA
   - RSA-RC4-MD5 : RSA is used for Key Exchange and for Server Authentication.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Key Exchange Algorithms

- RSA : The client chooses the secret and encrypts it with the Server RSA PK. THis PK has to be authenticated in a certificate (PKCS#1 v1.5 standard).
- DH_DSS & DH_RSA : fixed Diffie-Hellman with long-term parameters.
- DHE_DSS & DHE_RSA : *Ephemeral* Diffie-Hellman with random parameters (Client & Server).
- DH_Anon : DH_Anon_EXPORT (Ex : TLS_DH_anon_WITH_AES_128_CBC_SHA)/ Particular case where parameters are not authenticated. Of course, vulnerable to MIM.
- ECDHE : Elliptic Curve Diffie-Hellman Exchange.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

TLS/SSL Session in Details : Client_Hello

Handshaking - Ciphersuit Negotiation [A d, Vau06]

Client sends a plaintext Client_Hello message and suggests some cryptographic parameters (collectively called ciphersuit) to be used for their communication session. The Client_Hello message also contains a 32-byte random number denoted as client_random. For example,
Client_Hello :

1. Protocol Version : TLSv1 if you can, else SSLv3.
2. Key Exchange : RSA if you can, else Diffe-Hellman.
3. Secret Key Cipher Method : 3DES if you can, else DES.
4. Message Digest : SHA-1 if you can, else MD5.
5. Data Compression Method : PKZip if you can, else gzip.
6. Client Random Number : 32 bytes.

Handshaking - Ciphersuit Negotiation

The stronger method (in terms of security) shall precede the weaker one, e.g. RSA (1024-bit) precedes DH, 3DES precedes DES, SHA-1 (160-bit) precedes MD5 (128-bit). Server responds with a plaintext Server_Hello to state the ciphersuit of choice (server decides on the ciphersuit). The message also contains a 32-bytes random number denoted as server_random. For example, Server_Hello : :

1. Protocol Version : TLSv1.
2. Key Exchange : RSA.
3. Secret Key Cipher Method : DES.
4. Message Digest : SHA-1.
5. Data Compression Method : PKZip.
6. Server Random Number : 32 bytes.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING
R. ERRA
Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses

Handshaking - Key Exchange

The server sends its digital certificate to the client, which is supposedly signed by a root CA. The client uses the root CA's public key to verify the server's certificate (trusted root-CAs' public key are pre-installed inside the browser). It then retrieves the server's public key from the server's certificate. (If the server's certificate is signed by a sub-CA, the client has to build a digital certificate chain, leading to a trusted root CA, to verify the server's certificate.)
The server can optionally request for the client's certificate to authenticate the client. In practice, server usually does not authenticate the client. This is because :

1. Server authenticates client by checking the credit card in an e-commerce transaction.
2. Most clients do not have a digital certificate.
3. Authentication via digital certificate takes time and the server may lose an impatient client.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING
R. ERRA
Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses

Handshaking - Key Exchange :

HKE : Next step is to establish the Session Key

1. The client generates a 48-byte (384-bit) random number called pre_master_secret, encrypts it using the verified server's public key and sends it to the server.
2. Server decrypts the pre_master_secret using its own private key. Eavesdroppers cannot decrypt the pre_master_secret, as they do not possess the server's private key.
3. Client and server then independently and simultaneously create the session key, based on the pre_master_secret, client_random and server_random. Notice that both the server and client contribute to the session key, through the inclusion of the random number exchange in the hello messages. Eavesdroppers can intercept client_random and server_random as they are sent in plaintext, but cannot decrypt the pre_master_secret.
4. In a SSL/TLS session, the session key consists of 6 secret keys (to thwart crypto-analysis). 3 secret keys are used for client-to-server messages, and the other 3 secret keys are used for server-to-client messages.

Handshaking - Key Exchange : . . .

4 . . . Among the 3 secret keys, one is used for encryption (e.g., DES secret key), one is used for message integrity (e.g., HMAC) and one is used for cipher initialization. (Cipher initialization uses a random plaintext called Initial Vector (IV) to prime the cipher pump.)

5 Client and server use the pre_master_secret (48-byte random number created by the client and exchange securely), client_random, server_random, and a pseudo-random function (PRF) to generate a master_secret. They can use the master_secret, client_random, server_random, and the pseudo-random function (PRF) to generate all the 6 shared secret keys. Once the secret keys are generated, the pre_master_secret is no longer needed and should be deleted.

6 From this point onwards, all the exchanges are encrypted using the session key.

7 The client sends Finished handshake message using their newly created session key. Server responds with a Finished handshake message.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Message Exchange

Client and server can use the agreed-upon session key (consists of 6 secret keys) for secure exchange of messages.

Sending messages :

1. The sender compresses the message using the agreed-upon compression method (e.g., PKZip, gzip).
2. The sender hashes the compressed data and the secret HMAC key to make an HMAC, to assure message integrity.
3. The sender encrypts the compressed data and HMAC using encryption/decryption secret key, to assure message confidentiality.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Message Exchange

Retrieve messages :

1. The receiver decrypts the ciphertext using the encryption/decryption secret key to retrieve the compressed data and HMAC.
2. The receiver hashes the compressed data to independently produce the HMAC. It then verifies the generated HMAC with the HMAC contained in the message to assure message integrity.
3. The receiver un-compresses the data using the agreed-upon compression method to recover the plaintext.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

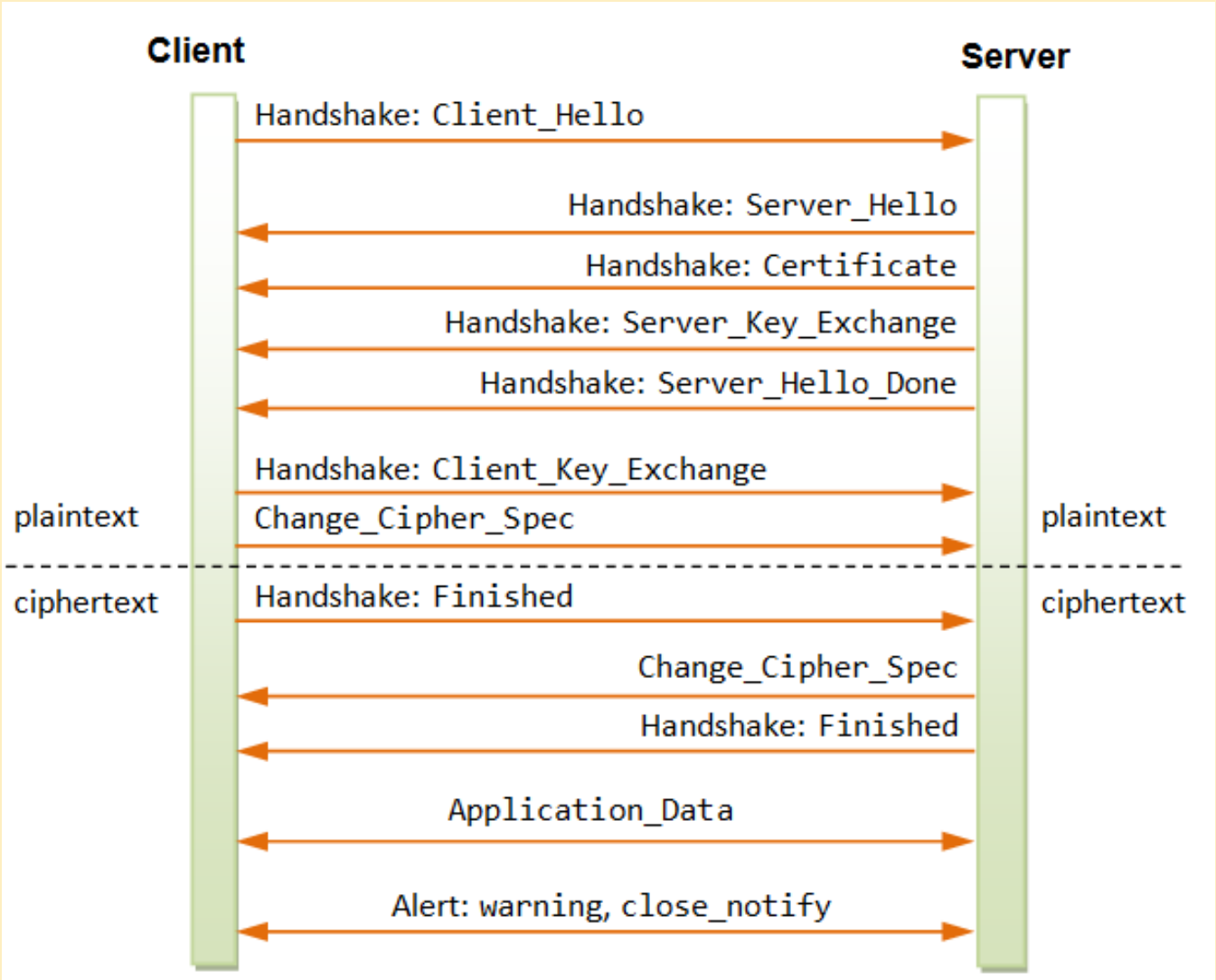
SCRAM

The future : SHA-3 ?

Hash functions : Uses

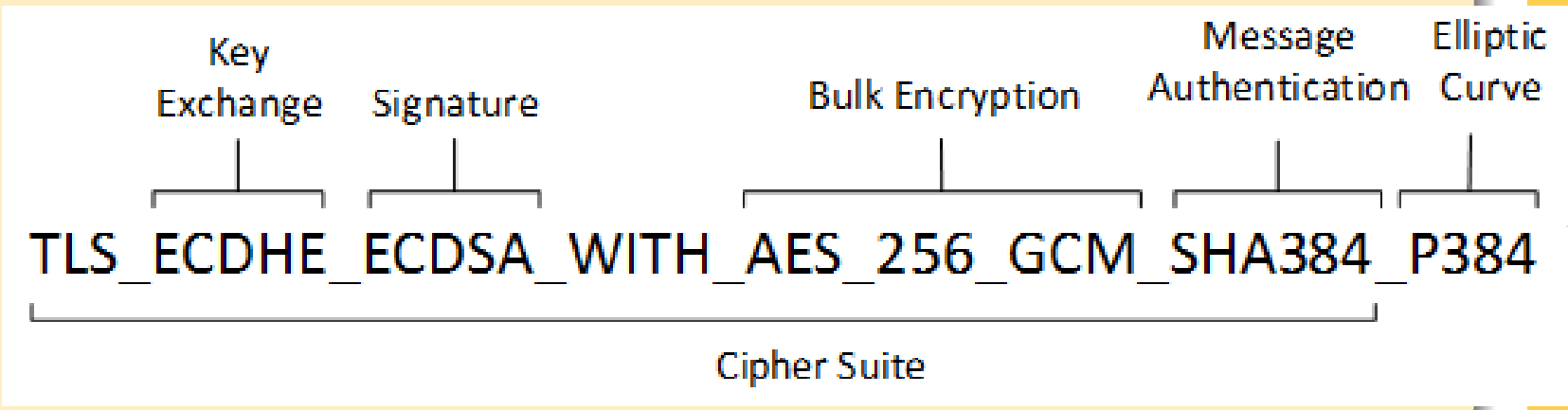
TLS/SSL : a picture

The following diagram shows the sequence of the SSL messages for a typical client/server session.



TLS/SSL : a picture

A picture from Microsoft [?]



Partage de secrets

Procédés d'Identification

e TLS/SSL otocol

LS/SSL : egociation

euve nteractive) sans port information

Mots de passe sous Unix/Linux

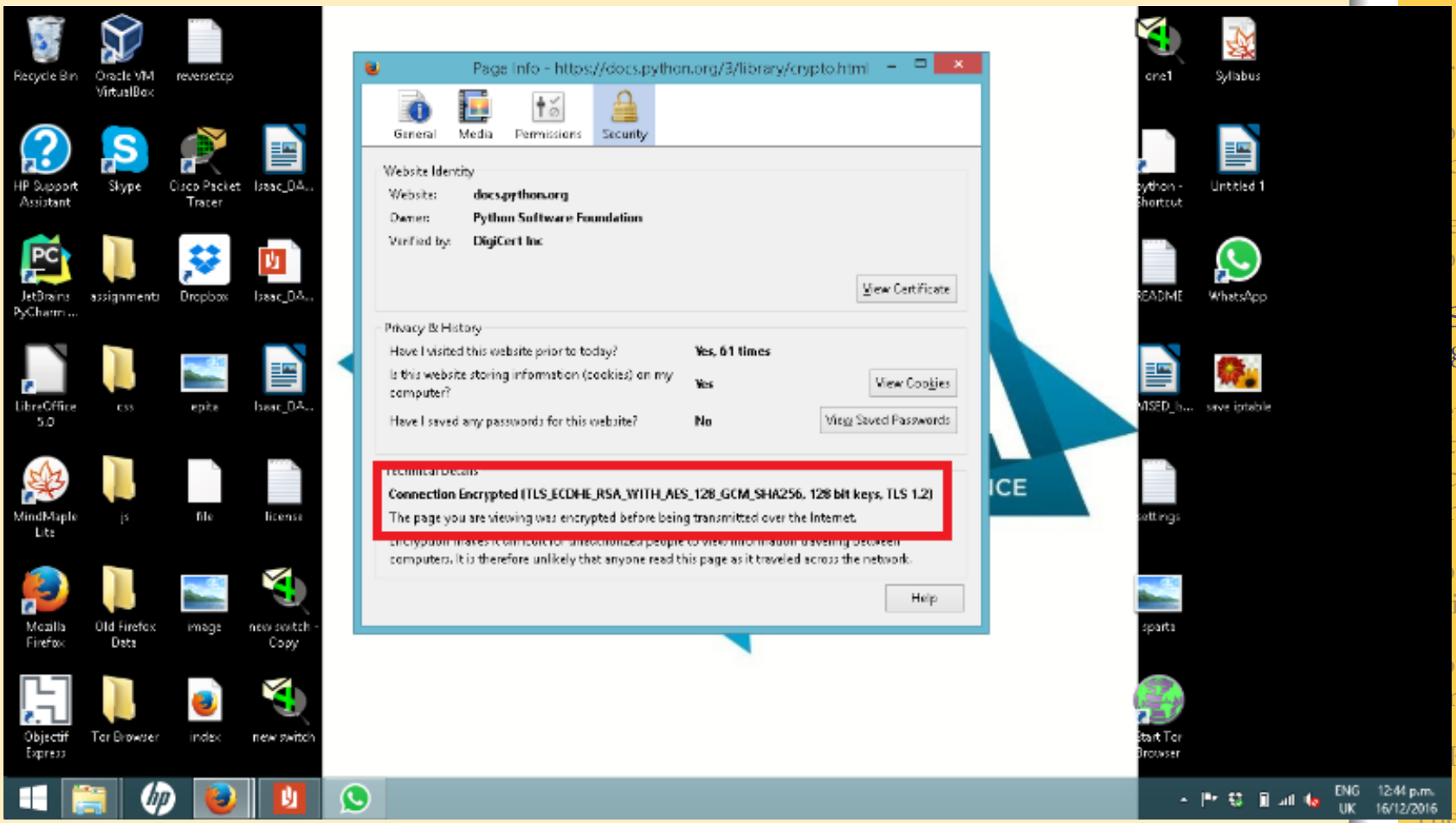
Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

A picture from a student



Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

Preuve (interactive) sans apport d'information

- Preuves dites *zéro-knowledge*
- Peggy est le prouveur
- Vic est le vérifieur
- Outils : envoie de messages, calculs privés, réception de messages
- C'est aussi un protocole de question-réponse
- Vic pose une question, Peggy répond, Vic accepte ou non suivant la réponse reçue
- Consistance (déjá vue) : si x est une instance positive de Π (problème de décision associé au protocole) Vic accepte lapreuve de Peggy [on peut itérer]
- Significativité (*soundness*) : si x est une instance négative de Π Vic accepte avec une très faible probabilité.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Preuve interactive de résiduosité quadratique

Algorithme 3 : Preuve interactive de residuosite quadratique
Donnees : $n = pq$ $x \in QR(n) = \{y \; y = x^2 \bmod n$ a une solution$\}$;
Sortie : V=*Vic accepte* ou F=*Vic rejette* la preuve de Peggy
Debut :
Repeter $\lfloor \log(n) \rfloor$ fois ;
Peggy choisit au hasard $v \in \mathbb{Z}_n^*$;
Peggy calcule $y = v^2 \bmod n$ et l'envoie a Vic ;
Vic choisit au hasard 1 ou 2 et l'envoie a Peggy ;
Peggy calcule $z = u^i v \bmod n$ avec u racine carree de y et l'envoie a Vic ;
Vic verifie si $z^2 \equiv x^i y \bmod n$;
Retourner V si les verifications sont positives pour les $\lfloor \log(n) \rfloor$ challenges, sinon **Retourner** F.
Fin.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING
R. ERRA
Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

Mots-de-passe sous Unix/linux

1. crypt(1) : fonction d'Unix basée sur l'algorithme ENIGMA (sic !). Très peu sûr.
2. crypt(3) : (avec variantes) utilisée actuellement (sauf sur de vieux Unix à cause de la loi US *International Traffic in Arms Regulations* (ITAR) contre l'exportation de cryptographie).

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

crypt(3) : exemple avec DES

1. On utilise comme « message » la chaîne composée de 8 caractères blancs.
2. Le MDP est tronqué à ses 8 premiers caractères, paddé avec des 0 si moins de 8 caractères
3. On chiffre 25 fois avec DES : la clé c'est le mot-de-passe !
4. DES est légèrement modifié (bits de poids faible) de manière à empêcher d'utiliser des hardwares DES rapides pour casser les MDP.
5. Pour rendre + difficile l'attaque par dictionnaire on ajoute du « sel » (12 bits soit 4096 valeurs différentes) qui modifie la fontion d'expansion E de DES.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

crypt(3) : exemple avec DES

1. Variantes [1] : Quelquefois on utilise MD5 ou BLOWFISH ou SHA-1 (Red Hat) à la place de DES , ou ce que vous voulez ...
2. Variantes [2] : sur BSDi : on utilise 24 bits de sel.
3. Variantes [3] : Et on voir apparaître SHA-256 . . .

EPITA ING2/ING3 Majeure SRS/TECOMS/APPING
R. ERRA
Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses

crypt(3) : Linux Programmer's Manual

```
#define _XOPEN_SOURCE /* See feature_test_macros(7) */
#include <unistd.h>

 char *crypt(const char *key, const char *salt);

#define _GNU_SOURCE /* See feature_test_macros(7) */
#include <crypt.h>

 char *crypt_r(const char *key, const char *salt,
                    struct crypt_data *data);

-----       Link with -lcrypt.
```

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

crypt(3) : Linux Programmer's Manual

key is a user's typed password.

salt is a two-character string chosen from the set [a-zA-Z0-9./].This string is used to perturb the algorithm in one of 4096 different ways.

By taking the lowest 7 bits of each of the first eight characters of the key, a 56-bit key is obtained. This 56-bit key is used to encrypt repeatedly a constant string (usually a string consisting of all zeros). The returned value points to the encrypted password, a series of 13 printable ASCII characters (the first two characters represent the salt itself). The return value points to static data whose content is overwritten by each call.

crypt(3) : Linux Programmer's Manual

```
If salt is a character string starting with the
characters "$id$" followed by a string terminated
 by "$":
               $id$salt$encrypted

then instead of using the DES machine, id
identifies the encryption method used and
this then determines how the rest of the
password string is interpreted
[...]
So $5$salt$encrypted is an SHA-256 encoded
password and $6$salt$encrypted is an
SHA-512 encoded one.
```

EPITA ING2/ING3 Majeure SRS/TECOMS/APPING
R. ERRA
Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses

crypt(3) : Linux Programmer's Manual

```
The following values of id are supported:
ID   | Method
-------------------------------------------------------------
1    | MD5
2    | Blowfish (not in mainline glibc; added in
     | some Linux distributions)
2a   | $2a=eksblowfish Algorithm
5    | SHA-256 (since glibc 2.7)
6    | SHA-512 (since glibc 2.7)
```

EPITA ING2/ING3 Majeure SRS/TECOMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

crypt(3) : Linux Programmer's Manual

```
"salt" stands for the up to 16 characters following
"$id$" in the salt.  The encrypted part of the
password string is the actual computed password.
The size of this string is fixed:
       MD5     | 22 characters
       SHA-256 | 43 characters
       SHA-512 | 86 characters
The characters in "salt" and "encrypted" are drawn
from the set [a-zA-Z0-9./].
In the MD5 and SHA implementations the entire key
is significant (instead of only the first 8 bytes
 in DES).
```

EPITA ING2/ING3 Majeure SRS/TECOMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

http ://cvsweb.openbsd.org/cgi-bin/cvsweb/src/etc/etc.amd64/login.conf ?rev=1.6&conte

OpenBSD : fichier login.conf

Blowfish sur 8 caractères par défaut. (Plus de DES/MD5 etc.).

```
default:\
:path=/usr/bin /bin /usr/sbin /sbin /usr/X11R6/bin /usr/local
:umask=022:\
:datasize-max=512M:\
:datasize-cur=512M:\
:maxproc-max=256:\
:maxproc-cur=128:\
:openfiles-cur=512:\
:stacksize-cur=4M:\
:localcipher=blowfish,8:\
:tc=auth-defaults:\
:tc=auth-ftp-defaults:
```


Partage de secrets

Procédés d'Identification

IPSec SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

Mots-de-passe locaux sous Windows

Pour des raisons de compatibilité, deux algorithmes :

1. LANMAN (ou LAN Hash) (le plus ancien) : peu sécurisé
2. NTLM : a priori dans (presque) tous les Windows XX

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

LANMAN (d'après C. GRENIER)

1. Le MdP est mis en majuscules
2. Il est complété par des zéros s'il a moins de quatorze caractères.
3. La chaîne est séparée en deux moitiés (de chacune 7 caractères)
4. Ensuite, la chaîne ASCII CONST=**0x4B,0x47,0x53,0x21,0x40,0x23,0x24,0x25** (ou **"'KGS !@#$%"'**) est cryptée avec les sept premiers caractères du mot de passe par du DES pour former la 1ère partie du mot de passe crypté **C1** (64 bits). (On ajoute un bit=0 tous les 8 bits pour arriver à 64 bits, mais toujours 56 bits "'utiles"' de clé).

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

LANMAN (d'après C. GRENIER)

(Suite)

1. Cette même chaîne est chiffrée avec les sept caractères suivants (amenés aussi à 64 bits) pour former la 2d partie du mot de passe **C2** (64 bits).
2. On concatène **C1** et **C2** ce qui donne **C1|C2** sur 128 bits= 16 octets : le *LM hash*.
3. Note : ce *n'est pas* techniquement un algorithme de hash mais un *One Way Encryption* (*OWE*).

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Quelques défauts de LANMAN (d'après C. GRENIER et WIKIPEDIA)

1. Le MdP est mis en majuscules : cela réduit l'alphabet !
2. **C1** et **C2** sont **indépendants** donc attaquables *en parallèle*
3. AFB[a] : nombre d'essais (**128-26=92**)
   - Avec 14 caractères imprimables : $95^{14} \approx 2^{92} = 4,95\ 10^{27}$.
   - Avec 7 caractères imprimables : $95^{7} \approx 2^{47} = 7,03\ 10^{13}$.
   - Avec 7 caractères MAJ imprimables : $2^{43} = 8,7\ 10^{12}$.
4. (C. GRENIER) : *Une attaque brute permet 2,7 millions tentatives par seconde avec John The Ripper sur un Athlon 1,4 Ghz et à peine 4000 pour le codage MD5 utilisé sur la majorité des Linux. Microsoft fait vraiment de drôles de choix...*

---

a. Attaque par force brute

EPITA ING2/ING3 Majeure SRS/TECOMS/APPING
R. ERRA
Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses

Choisir un bon mot de passe ?

1. Le MdP contient des minuscules et des majuscules : cela augmente l'alphabet mais pas sous Windows !
2. Le MdP contient des nombres : *NUMB3R5* cela augmente l'alphabet sous Windows et sous Linux !
3. Choisir réellement 14 caractères sous Windows -(avec une phrase : exemple :
   - Partir de *Il fait beau aujourd'hui* : cela fait 24 caractères
   - Partir de *Il Fait Beau Aujourd'hui* : cela fait 24 caractères
   - Partir de *Il1Fait2Beau3Aujourd'hui* : cela fait 24 caractères
   - Partir de *Il1Fait2Bea3Ajord'hi* : cela fait 24 caractères
   - etc.

)

EPITA ING2/ING3 Majeure SRS/TECOMS/APPING
R. ERRA
Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses

https://www.microsoft.com/security/sir/strategy/default_word_hashes

LM is an obsolete hash scheme originally developed for Microsoft LAN manager, a network operating system product that predated the Windows Server line of products. To provide backward compatibility with legacy resources, some older versions of Windows, including Windows XP and Windows Server 2003, were designed to store password hashes in both LM and NT forms by default. The LM hash is very weak, and Microsoft has long recommended that it be disabled.
(See support.microsoft.com/kb/299656 for instructions for preventing Windows from storing LM hashes).

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

https://www.microsoft.com/security/sir/strategy/default_word_hashes

The much stronger NT hash scheme is used by all currently supported versions of Windows. Windows Vista, Windows Server 2008, and later versions of Windows support the use of AES encryption for password keys in conjunction with the Kerberos authentication protocol. Windows 7, Windows 8, Windows Server 2008 R2, and Windows Server 2012 attempt to use Kerberos and AES for authentication by default. Digest hashes are stored in Active Directory only if the appropriate option (Store passwords using reversible encryption) is enabled, and can be used for Internet Information Service (IIS) 6.0 (and earlier) digest authentication.

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

SCRAM : A Protocol for Password Authentication [WIK]

SCRAM (Salted Challenge Response Authentication Mechanism)

In cryptography, the Salted Challenge Response Authentication Mechanism (SCRAM) is a family of modern, password-based challenge-response authentication mechanisms providing authentication of a user to a server.

As it is specified for Simple Authentication and Security Layer (SASL), it can be used for password-based logins to services like SMTP and IMAP (e-mail), or XMPP (chat).

For XMPP, supporting it is mandatory.

EPITA ING2/ING3 Majeure SRS/TE-COMS/APPING

R. ERRA

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

https ://en.wikipedia.org/wiki/SCRAM

SCRAM : A Protocol for Password Authentication

Although all clients and servers have to support the SHA-1 hashing algorithm, SCRAM is, unlike CRAM-MD5 or DIGEST-MD5, independent from the underlying hash function. All hash functions defined by the IANA can be used instead. As mentioned in the Motivation section, SCRAM uses the PBKDF2 mechanism, which increases the strength against brute-force attacks, when a data leak has happened on the server.
Let H be the selected hash function, given by the name of the algorithm advertised by the server and chosen by the client.
'SCRAM-SHA1' for instance, uses SHA1 as hash function.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

SCRAM : A Protocol for Password Authentication

The salted password $\mathbf{password}_s$ is calculated as follows :

$$\mathbf{password}_s = H_i(\mathbf{password}, \mathbf{salt}, \mathbf{it}) \ ,$$

where $H_i(p, s, i)$ defined as :

PBKDF2(HMAC, p, s, i, output length of H).

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

https ://en.wikipedia.org/wiki/SCRAM

PBKDF2 ?

PBKDF2 (Password-Based Key Derivation Function 2) is a key derivation function that is part of RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 v2.0, also published as Internet Engineering Task Force's RFC 2898. It replaces an earlier standard, PBKDF1, which could only produce derived keys up to 160 bits long.[1]

In 2013, a Password Hashing Competition was held to develop a more resistant approach. On 20 July 2015 Argon2 was selected as the final PHC winner, with special recognition given to four other password hashing schemes : Catena, Lyra2, yescrypt and Makwa.

Partage de secrets

Protocoles d'identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

https ://en.wikipedia.org/wiki/SCRAM

DK = PBKDF2(PRF, Password, Salt, c, dkLen)

- PRF is a pseudorandom function of two parameters with output length hLen (e.g. a keyed HMAC)
- Password is the master password from which a derived key is generated
- Salt is a sequence of bits, known as a cryptographic salt
- c is the number of iterations desired
- dkLen is the desired length of the derived key
- DK is the generated derived key

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

The future : SHA-3 ?

SHA-3 ? see [?] on Documentation Center

The National Institute of Standards and Technology (NIST) is in the process of selecting a new cryptographic hash algorithm through a public competition. The new hash algorithm will be referred to as 'SHA-3' and will complement the SHA-2 hash algorithms currently specified in Federal Information Processing Standard (FIPS) 180-3, Secure Hash Standard [1]. The selected algorithm is intended to be suitable for use by the U.S. government, as well as the private sector and, at the completion of the competition, to be available royalty-free worldwide. The competition will be referred to as the SHA-3 competition hereafter in this document. The competition is NIST's response to recent advances in the cryptanalysis of hash algorithms, including the government standard SHA-1 hash algorithm [1, 2].

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

The future : SHA-3 ?

SHA-3 see [?]

An attack by Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu [3], and extended by many others, has seriously called into question the security of SHA-1's use in digital signatures and other applications that require collision resistance. While the SHA-2 family [1] of hash algorithms provides an immediate alternative, NIST expects the selected SHA-3 to offer security that is at least as good as the SHA-2 algorithms with significantly improved efficiency or additional features. In preparation for the SHA-3 competition, NIST held workshops on October 31-November 1, 2005 [4] and August 24-25, 2006 [5] to discuss the status of hash algorithms and develop a path forward for developing a new hash algorithm standard. As a result, NIST instituted a public competition, similar to that used to select the Advanced Encryption Standard (AES) [6, 7]. [. . .]

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

The future : SHA-3 ?

SHA-3 see [?]

- NIST received 64 candidate algorithm by the October 31, 2008 entry deadline for the SHA-3 competition.
- Of these, NIST accepted 51 first-round candidates as meeting the minimum acceptance criteria for being 'complete and proper submissions', as defined in FRN-Nov07. These criteria included provisions for reference and optimized C code implementations, known-answer tests, a written specification, and required intellectual property statements.
- In addition, the algorithms were required to be implementable in a wide range of hardware and software platforms, support message digest sizes of 224, 256, 384, 512 bits, and support a maximum message length of at least $2^{64}1$ bits.
- NIST selected 51 entries for the Round 1 and 14 of them advanced to Round 2.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

The future : SHA-3 ?

Proposal for SHA-3 see [WIK, ?] :Accepted for Round Two

The following hash function submissions have been accepted for Round Two :

- BLAKE
- Blue Midnight Wish
- CubeHash (Bernstein)
- ECHO (France Telecom)
- Fugue (IBM)
- Grøstl (Knudsen et al.)
- Hamsi
- JH
- Keccak (Keccak team, Daemen et al.)
- Luffa
- Shabal
- SHAvite-3
- SIMD
- Skein (Schneier et al.)

The future : SHA-3 ?

Proposal for SHA-3 see [WIK, ?] : : Conceded entrants

The following Round One entrants have been officially retracted from the competition by their submitters ; they are considered broken according to the NIST official Round One Candidates web site. As such, they are withdrawn from the competition.

- Abacus
- Boole
- DCH
- Khichidi-1
- MeshHash
- SHAMATA
- StreamHash
- Tangle
- WaMM
- Waterfall

The future : SHA-3 ?

Proposal for SHA-3 see [WIK, ?] : Rejected entrants

Several submissions received by NIST were not accepted as First Round Candidates, following an internal review by NIST. In general, NIST gave no details as to why each was rejected. NIST also has not given a comprehensive list of rejected algorithms ; there are known to be 13, **but only the following are public**.

- HASH 2X
- Maraca
- NKS 2D
- Ponic
- ZK-Crypt

The future : SHA-3 ?

Proposal for SHA-3 see [WIK, ?] : The winner

Keccak (Keccak team, Daemen et al.)

- SHA3 : a subset of the cryptographic primitive family Keccak
- Designed by Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche,
- Building upon RadioGatun.
- SHA-3 is a member of the Secure Hash Algorithm family.
- The SHA-3 standard was released by NIST on August 5, 2015.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

Hash functions : Uses

- Hash functions are used to get a digest of a message Must take variable size input, produce fixed size pseudorandom output, be efficient to compute
- Cryptographic hash functions should be preimage resistant, 2nd preimage resistant, and collision resistant (*i.e. reminder : computationally difficult*)
- Cryptographic hashes are used for **message authentication, digital signatures, password storage**
- SHA-1 produces 160 bit output, SHA-224, SHA-256, SHA- 384, and SHA-512 produce 224, 256, 384, and 512 bit outputs. All consist of 80 rounds.

Partage de secrets

Procédés d'Identification

The TLS/SSL protocol

TLS/SSL : Negociation

Preuve (interactive) sans apport d'information

Mots de passe sous Unix/Linux

Mot de passe sous Windows

SCRAM

The future : SHA-3 ?

Hash functions : Uses

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

The past : SHA0/SHA1

12 August 2004 : Antoine Joux and his team have found a collision on SHA-0

a766a602 b65cffe7 73bcf258 26b322b3̄ d01b1a9̄7 2684ef53̄ 3̄e3b4b7f 5̄3fe3762
24c08e47 e959b2bc 3b519880 b9286568 247d110f 70f5c5e2 b4590ca3 f55f52fe
effd4c8f e68de835 329e603c c51e7f02 545410d1 671d108d f5a4000d cf20a439
4949d72c d14fbb03 45cf3a29 5dcda89f 998f8755 2c9a58b1 bdc38483 5e477185
f96e68be bb0025d2 d2b69edf 21724198 f688b41d eb9b4913 fbe696b5 457ab399
21e1d759 1f89de84 57e8613c 6c9e3b24 2879d4d8 783b2d9c a9935ea5 26a729c0
6edfc501 37e69330 be976012 cc5dfe1c 14c4c68b d1db3ecb 24438a59 a09b5db4
35563e0d 8bdf572f 77b53065 cef31f32 dc9dbaa0 4146261e 9994bd5c d0758e3d
a766a602 b65cffe7 73bcf258 26b322b1̄ d01b1a*d̄*7 2684ef51̄ *b̄*e3b4b7f *d̄*3fe3762
a4c08e45 e959b2fc 3b519880 39286528 a47d110d 70f5c5e0 34590ce3 755f52fc
6ffd4c8d 668de875 329e603e 451e7f02 d45410d1 e71d108d f5a4000d cf20a439
4949d72c d14fbb01 45cf3a69 5dcda89d 198f8755 ac9a58b1 3dc38481 5e4771c5
796e68fe bb0025d0 52b69edd a17241d8 7688b41f 6b9b4911 7be696f5 c57ab399
a1e1d719 9f89de86 57e8613c ec9e3b26 a879d498 783b2d9e 29935ea7 a6a72980
6edfc503 37e69330 3e976010 4c5dfe5c 14c4c689 51db3ecb a4438a59 209b5db4
35563e0d 8bdf572f 77b53065 cef31f30 dc9dbae0 4146261c 1994bd5c 50758e3d

Rappel du Plan

1. Partage de secrets
2. Procédés d'Identification
3. The TLS/SSL protocol
4. TLS/SSL : Negociation
5. Preuve (interactive) sans apport d'information
6. Mots de passe sous Unix/Linux
7. Mot de passe sous Windows
8. SCRAM
9. The future : SHA-3 ?
10. Hash functions : Uses
11. Attacks of SHA-0
12. Exponentiation rapide : pour RSA/DH/etc.

Exponentiation rapide : version classique

Algorithm 1 : **Version « Droite-Gauche »**

Donnees : $N \in \mathbb{N}^*, m \in \mathbb{Z}_N, e \in \mathbb{Z}_N, ;$
Sortie : Puiss-Mod$(m, e, N) = m^e \bmod N$;
Debut :
Calculer $e = (b_{k-1}, b_{k-2} \cdots b_1, b_0)_2$;
$R_0 = 1$; $R_1 = m$;
Pour $i = 0$ **jusque** $i = k - 1$ **Faire**
  Si $(b_i == 1)$ **alors** $R_0 = R_0 R_1 \bmod N$ **FinSi** ;
  $R_1 = R_1^2 \bmod N$;
FinPour
Retourner R_1
Fin.

Exponentiation rapide : pour RSA/DH/etc.

Version « moins » classique mais Ãl'quivalente

Algorithm 2 : Version « Gauche-Droite »

Donnees : $g \in \mathbb{Z}_n, e \in \mathbb{Z}_n, n \in \mathbb{N}^*$;
Sortie : Puiss-Mod$(g, e, n) = g^e \bmod n$;
Debut :
Calculer $e = (d_{k-1}, d_{k-2} \cdots d_1, d_0)_2$;
$R_0 = 1$; $R_1 = g$;
Pour $i = k - 1$ **jusque** $i = 0$ **Faire**
  $R_0 = R_0^2 \bmod n$;
  Si $(d_i == 1)$ **alors** $R_0 = R_0 R_1 \bmod n$
  FinSi ;
FinPour
Fin.

Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses

A description of SSL/TLS.
https ://www3.ntu.edu.sg/home/ehchua/programming/webprogramm

D. Stinson.
Cryptographie, Théorie et Pratique.
Vuibert, 1994.

S. Vaudenay.
A classical introduction to cryptography.
Springer, 2006.

WIKIPEDIA.
http ://www.wikipedia.org.

Partage de secrets
Procédés d'Identification
The TLS/SSL protocol
TLS/SSL : Negociation
Preuve (interactive) sans apport d'information
Mots de passe sous Unix/Linux
Mot de passe sous Windows
SCRAM
The future : SHA-3 ?
Hash functions : Uses