Sujets connexes	Informations sur la signature avant la version 10 de LoGeAs Web Signature des enregistrements dans les tables

Partie cryptologie de LoGeAs (chaine aléatoire, cryptage, signature...)

En vue de la certification NF 552, et suite à l'audit de robustesse de celle-ci, la logique des signature dans LoGeAs Web à été revu avec comme objectifs :

- utilisé une bibliothèque unique, suivi et reconnu
- utilisé un format de signature recommandé par l'ansii

La bibliothèque sur laquelle se base LoGeAs Web

LoGeAs se base pour toute la partie crypto sur deux dll externes

Fichier	Description du fichier	Version du ficher	Version du produit	Copyright	Fin de support par OpenSSL
libcrypto-1_1-x64.dll	OpenSSL Librairy	1.1.1.12	1.1.1u	Copyright @1998-2021 The OpenSSL Authors	2023
libssl-1_1-x64.dll	OpenSSL Librairy	1.1.1.12	1.1.1l	Copyright @1998-2021 The OpenSSL Authors	2023

Remarque : Il ne s'agit pas la dernière version d'OpenSSL, mais il s'agit d'une version stable encore maintenu.
La version 3 d'Opensll n'est à ce jour pas encore correctement supportée par nos outils et semble poser des problèmes en mode win64

Quelques ressources

- https://www.openssl.org/
- https://wiki.openssl.org/index.php/Binaries
- https://en.delphipraxis.net/

Choix technologique

Chaine aléatoire

A plusieurs reprise lors des processus de signature des chaines aléatoire sont nécessaire, ils sont générés via la dll OpenSLL

```
RAND_bytes(@Res[0], NbCaractere)
```

Les tailles prise en compte sont :

Objet	Taille en caractère
Mot de passe des bases de données	70
Sel pour le cryptage des mots de passe utilisateurs	10
Mot de passe des certificats; A noter que celui-ci est en plus “bidouillé”	70

Le certificat

Le certificat généré pour signer dans l'application est de type X509 auto-signé avec un RSA de 2048. La clef privé étant encapsulé en AES256. Il est stocké dans une base de donnée externe à l'application qui est déployé (en release) sur un serveur indépendant non accessible depuis Internet.

Stockage des mots de passe utilisateurs

Les mots de passe des utilisateurs (on parle ici des utilisateurs du logiciel LoGeAs Web) sont stockés dans la base de donnée sous forme d'un hachage. Celui est fait de deux parties le mot de passe de l'utilisateur + une partie aléatoire, appelé sel (différente pour chaque utilisateur).
Le mot de passe en clair de l'utilisateur n'est jamais stocké.
Le hachage est stocké dans la base principale de “Monespace”, ainsi que dans les bases sur lesquels il a des droits d'utilisations.
La partie aléatoire (sel) est stocké dans une base de donnée externe à l'application qui est déployé (en release) sur un serveur indépendant non accessible depuis Internet.

Les signatures

La signature mise en place ne concerne pas l'intégralité des données chiffrées, mais seulement leur empreinte. On trouvera dans ce document le détail des éléments pris en compte. Signer un document entier serait la solution la plus simple, cependant ce n'est absolument pas optimisé, car une signature n'étant rien d'autre que des données encryptées, la taille de la signature serait égale à la taille des données encryptées. Insignifiant pour un fichier texte brut, mais inenvisageable pour des dossiers.
Le fait de rendre accessible directement les données encryptées pourrait également rendre l'intégrité de la signature vulnérable à une attaque cryptographique visant à forger une signature valide pour des données, à partir de signatures et de données existantes.
Pour contrer ces deux problèmes majeurs, une solution existe : le hachage. C'est une fonction réductrice qui nous permettra de calculer un « hash » qui est une empreinte des données qu'on lui a envoyé. En effet ce n'est qu'une empreinte, bien que la fonction soit déterministe, elle est impossible à inverser en raison d'une perte d'information induite volontairement.

Bien que ce hash ne nous permette pas de récupérer les informations initiales, il va nous permettre de les identifier avec certitude : en effet les risques de collision (que deux données différentes donnent le même hash) sont faibles, et les mêmes données donneront toujours le même hash. Afin de réduire la taille de notre signature, nous allons donc hasher les données avant de signer le hash final, ceci nous permettra effectivement de confirmer l'intégralité du document sans nécessairement conserver l'intégralité des données du document original. Logeas utilise comme fonction de **hachage**

l'algorithme SHA-384. Celle-ci fait partie de la famille SHA-2 (Secure Hash Algorithm) qui ont été conçues par la National Security Agency des États-Unis (NSA), sur le modèle des fonctions SHA-1 et SHA-0, elles-mêmes fortement inspirées de la fonction MD4 de Ron Rivest (qui a donné parallèlement MD5). Telle que décrite par le National Institute of Standards and Technology (NIST), elle comporte les fonctions, SHA-256 et SHA-512 dont les algorithmes sont similaires mais opèrent sur des tailles de mot différentes (32 bits pour SHA-256 et 64 bits pour SHA-512), SHA-224 et SHA-384 qui sont essentiellement des versions des précédentes dont la sortie est tronquée, et plus récemment SHA-512/256 et SHA-512/224 qui sont des versions tronquées de SHA-512. Le dernier suffixe indique le nombre de bits du haché (Voir l'article de Wikipédia).

Extrait du code

Appel de la procédure avec les paramètres :

Data	Qui contient la chaine à signer
Pivatekey	La partie privé du certificat
HasDigest	La méthode de hachage, **on utilise SHA 384**

```
var
    Signature: array [0..8000] of AnsiChar;   // binary result, not hex or
base64, V8.67 larger
    Etype: PEVP_MD;
    PkeyCtx: PEVP_PKEY_CTX;
    DigestCtx: PEVP_MD_CTX;
    SigLen: size_t;           { V8.64 }
    Ret, keytype: integer;
begin
    Result := '';
    if not Assigned(PrivateKey) then
              Raise EDigestException.Create('Private key required');
    keytype := EVP_PKEY_base_id(PrivateKey);
    if (keytype <> EVP_PKEY_RSA) and (keytype <> EVP_PKEY_EC) and
        (keytype <> EVP_PKEY_ED25519) and (keytype <> EVP_PKEY_RSA_PSS) then
              Raise EDigestException.Create('Unsupported private key type');

    DigestCtx := EVP_MD_CTX_new;
    PkeyCtx := Nil;
    SigLen := SizeOf(Signature);
    try
        Etype := IcsSslGetEVPDigest(HashDigest);
        if keytype = EVP_PKEY_ED25519 then
            EType := Nil // Needs 1.1.1
        else if NOT Assigned(Etype) then
            Raise EDigestException.Create('Unsupported hash digest ' +
GetEnumName(TypeInfo(TEvpDigest), Ord(HashDigest)));
        Ret := EVP_DigestSignInit(DigestCtx, @PkeyCtx, Etype, Nil,
PrivateKey);
        if (Ret <= 0) then RaiseLastOpenSslError(EDigestException, FALSE,
```

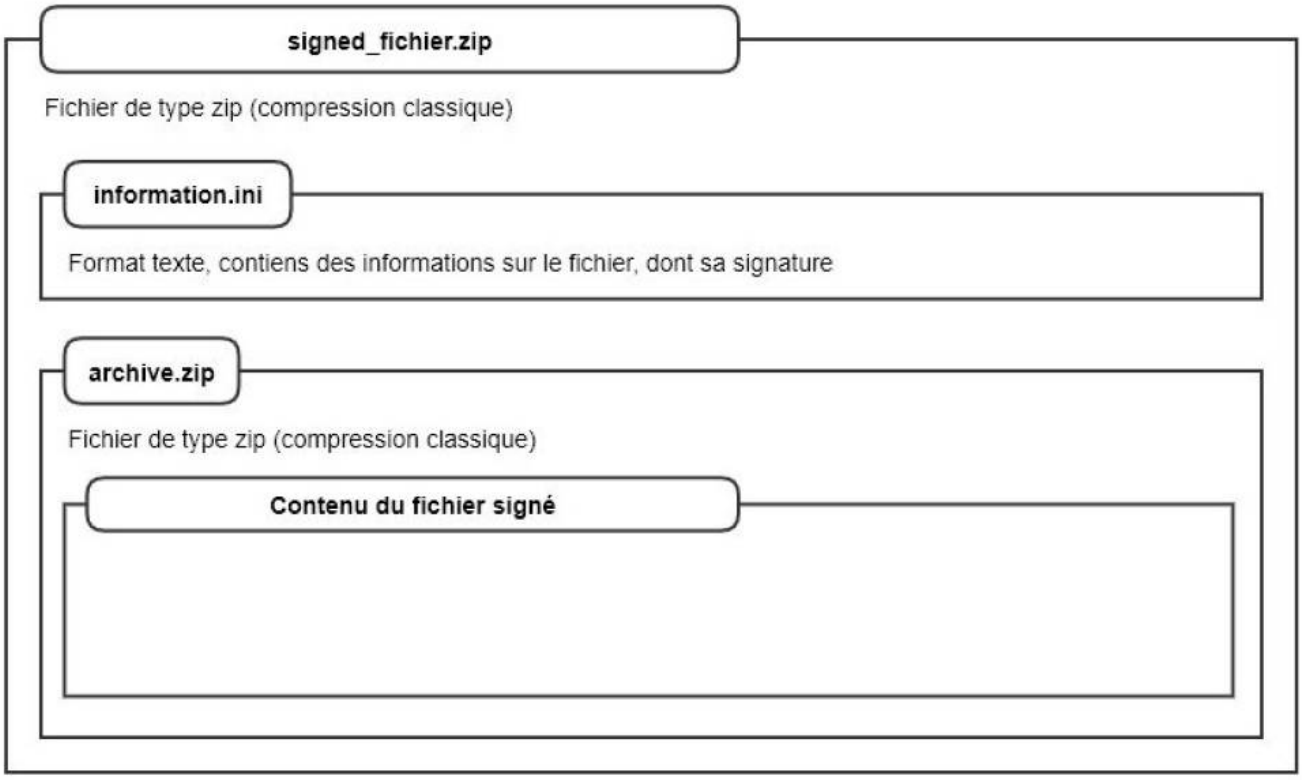
```
'Failed to initialise signing digest');

        ret := EVP_DigestSign(DigestCtx, @Signature, @SigLen,
PAnsiChar(Data), Length(Data));  // Needs 1.1.1  }
        if (Ret <= 0) then RaiseLastOpenSslError(EDigestException, FALSE,
Failed to finalise signing digest');
        if SigLen > 0 then begin
            SetLength(Result, SigLen);
            Move(Signature[0], Result[1], SigLen);
        end;
    finally
        EVP_MD_CTX_free(DigestCtx);
    end;
end;
```

Extrait de bibliothèque ICS de Angus Robertson, Magenta Systems Ltd
Pour les explications des fonctions d'OpenSSL (et pour les spécialiste seulement ...) on pourra se référer à https://www.openssl.org/

Architecture des fichiers signés par LoGeAs Signature

Un fichier signé par LoGeAs Signature est un conteneur au format zip dont la structure est.



(le format source de cette image est dans le dossier de certification)

Le fichier "Information.ini"

D'un format facilement lisible (format descendant du format de Windows) il comprends :

Section	Identification	Contenu
Editeur	Nom	Logeas Informatique
Editeur	Adresse	22 rue Saint Genest - 31800 LABARTHE INARD
Logiciel	Nom	Nom du logiciel qui a réalisé la signature
Serveur	Version	Version du logiciel
Serveur	Nom	Nom du serveur hébergeant la solution
Signature	Version	Version interne de la signature
Signature	Type	Type de la signature utilisée
Signature	Certificat	Type de certificat
Signature	Dll-nom	Nom de l'outil de cryptographie utilisé
Signature	Dll-organisme	Organisme source
'Signature	Dll-version	Version
Signature	Data	Signature du fichier “archive.zip”
Signature	Information	https://logeas.wiki.logeas.fr/doku.php?id=certif:dat
Signature	Date	Date de la signature

Vérifier que le fichier est conforme à sa signature

Utilisez le menu “Tester la signature” de “Administration\Tests” dans Logeas-Web.fr

Extraire la partie "utile"

LoGeAs Web dispose d'un menu permettant d'extraire facilement la partie utile d'une archive signée “Administration\Exporter l'archive d'un fichier signé par LoGeAs-Signature”

From:
https://wiki-logeas.fr/certif/ - **dokuwiki-certif**

Permanent link:
https://wiki-logeas.fr/certif/doku.php?id=certif:signature

Last update: **2026/10/13 08:18**