Fiches techniques
Génération d'une signature

Sami Babigeon

19 novembre 2021

1 Problématique

Dans le cadre du projet, le client souhaite pouvoir générer une licence pour protéger un ou plusieurs de ses logiciels. Pour se faire nous allons devoir mettre en place un système de signature permettant de vérifier l'authenticité d'une licence.

2 Méthode de réalisation

Le schéma classique d'une signature est le suivant :

1. Génération d'un couple de clefs (public, privé) via un algorithme probabiliste

   $(vk, sk) \leftarrow KG(\lambda)$, λ étant un paramètre modulable de l'algorithme de génération

2. Création de la signature pour un message m

   $$s \leftarrow Sign(sk, m)$$

3. Vérification de la signature

   $$r \leftarrow Verify(vk, m, s') \text{ avec } r = (s == s')$$

3 Exemple avec OpenSSL

Le script suivant permet de comprendre le fonctionnement d'une signature avec la librairie OpenSSL.

Script `sign.sh` :

```
#!/bin/bash

echo -e "\n===== Generating (Private key, Public key) ====="
openssl genrsa -out private.pem 1024 # generate private key file
# openssl rsa -in private.pem -text # view info in the private key file
# extract public key to file
openssl rsa -in private.pem -pubout -out public.pem
# view info in the public key file
# openssl rsa -in public.pem -pubin -text
cat /proc/sys/kernel/random/uuid > msg.txt
echo -e "\n===== File to sign ====="
cat msg.txt

openssl dgst -sha256 -sign private.pem -out sign.sha256 msg.txt
base64 sign.sha256 > sign.txt

echo -e "\n===== Signature (Encoded in B64) ====="
cat sign.txt

base64 -d sign.txt > sign.sha256
echo -e "\n===== Verification ====="
openssl dgst -sha256 -verify public.pem -signature sign.sha256 msg.txt

echo -e "\n===== Changing Content of Msg.txt ====="
cat /proc/sys/kernel/random/uuid > msg.txt
cat msg.txt

echo -e "\n===== Verification ====="
openssl dgst -sha256 -verify public.pem -signature sign.sha256 msg.txt

echo -e "\n===== Cleaning ====="
rm *.pem *.txt *.sha256
echo -e "done"
```