PKCS#1 v1.5

Meziane HAMOUDI
Marc-Henri KASSES

Octobre 2017

1 Introduction

PKCS (Public Key Cryptography Standards) ou standard de cryptographie à clef publique, a été conçu par la société RSA security qui a développé et promus les PKCS qui permettent l'implémentation des techniques de cryptographie à clef publique, après l'expiration de la licence RSA le 21 septembre 2000. Il existe 15 sections de PKCS qui définissent chacune un standard. Nous nous intéresserons plus à la première section de PKCS, plus précisément la version 1.5. Le PCKS 1 définit le standard de cryptographie RSA. Nous essayerons de présenter ce standard (génération de clé, chiffrement, déchiffrement, et signature), avant d'en voir les limites.

Aperçu général:
Chaque entité doit générer une paire de clés: une clé publique et une clé privée. Le processus de chiffrement doit être effectué avec l'une des clés (la clé publique) et le processus de déchiffrement doit être effectué avec l'autre clé (la clé privée). Les deux processus transforment une chaîne d'octets en une autre chaîne d'octets, ils sont inverses l' un de l'autre. Si un processus utilise la clé publique d'une entité, l'autre processus utilise la clé privée de la même entité.

2 Génération des clefs:

Chaque entité doit sélectionner un entier positif e comme exposant public. et de manière privée et aléatoire les nombres premiers p et q tels que $(p-1)$ et $(q-1)$ n'ont pas de diviseur en commun avec e.

Le module public n sera le produit p et q $n = pq$.

L'exposant privé doit être un entier positif d tel sorte que $de-1$ soit divisible par $p-1$ et $q-1$.

La longueur du module n en octets est l'entier k satisfaisant:

$$2^{8(k-1)} \leq n < 2^{8k}$$

.

remarque 1

1. *L'exposant public peut être standardisé dans des applications spécifiques.*
2. *Quelques conditions supplémentaires sur le choix des nombres premiers pourraient bien être prises en compte pour dissuader la factorisation du module. Ces conditions de sécurité sortent du cadre de notre exposé. La borne inférieure sur la longueur k sert à accueillir les formats de bloc, pas pour la Sécurité.*

3 Processus de chiffrement:

Le processus de chiffrement comprend quatre étapes: formatage du bloc de chiffrement, conversion chaîne d'octets-entier, calcul RSA et conversion entier-chaîne d'octets. L'entrée du processus de chiffrement doit être une chaîne d'octets D, les données, un entier n, le module et un entier c, l'exposant. Pour une opération à clé publique, l'entier c doit être l'exposant public d'une entité e; pour une opération à clé privée, il doit être l'exposant privé d'une entité d. Le résultat du processus de chiffrement doit être une chaîne d'octets ED, les données chiffrées.

3.1 Formatage du bloc de chiffrement:

le bloc de chiffrement EB est composé un bloc de type BT, une chaîne de remplissage PS et les données D.

$$EB = 00||BT||PS||00||D$$

BT doit être un seul octet indiquant la structure du bloc de chiffrement. Pour une opération à clé privée,BT doit être 00 ou 01 Pour une opération à clé publique, il doit être 02. PS doit être constituée de $k-3-||D||$ octets. Pour le bloc de type 00, les octets doivent avoir la valeur 00; pour le type de bloc 01, ils doit avoir la valeur FF; et pour BT 02, ils doivent être pseudo-aléatoire et non nulle. Cela fait que la longueur du bloc de chiffrement EB égale à k.

remarque 2

1. *L'octet principal 00 s'assure que le bloc de chiffrement soit converti en nombre entier inférieur au module.*

2. *Pour le bloc de type 00, les données D doivent commencer par un octet non nul ou avoir une longueur connue de sorte que le bloc de chiffrement peut être analysé sans ambiguïté. Pour les types de bloc 01 et 02, le bloc de chiffrement peut être analysé sans ambiguïté puisque la chaîne de remplissage PS ne contient pas d'octets avec la valeur 00 et la chaîne de remplissage est séparée des données D par un octet avec la valeur 00.*

3. *Pour le bloc de type 02, la chaîne de remplissage est d'au moins huit octets de longueur, ce qui est une condition de sécurité pour les opérations à clé publique elle empêche un attaquant de récupérer des données en essayant tous les blocs de chiffrement possibles. Pour simplicité, la longueur minimale est la même pour le type de bloc 01.*

3.2 Conversion chaîne d'octets-entier:

Le bloc de chiffrement EB doit être converti en un entier X, l'entier bloc de chiffrement. Soit $EB_1, \ldots, EB_k$ les octets de EB à partir du premier au dernier.

Alors l'entier x doit satisfaire:

$$X = \sum_{i=1}^{k} 2^{8(k-i)} EB_i$$

En d'autres termes, le premier octet de EB a le plus d'importance dans l'entier et le dernier octet de EB la plus petite.

3.3 Calcul RSA:

Le bloc de chiffrement entier X doit être élevé à la puissance c modulo n pour donner un entier Y, les données chiffrées entières.

$$Y = X^c \mod (n)$$

tel que: $0 \leq Y < n$
C'est le calcul RSA classique.

3.4 Conversion entier-chaîne d'octets:

L'entier données chiffrées Y doit être convertis vers une chaîne d'octets ED, les données chiffrées de longueur k. ED doit satisfaire:

$$X = \sum_{i=1}^{k} 2^{8(k-i)} ED_i$$

où $ED_1, \ldots, ED_k$ sont les octets de ED du premier au dernier.

En d'autres termes, le premier octet de ED a le plus d'importance dans l'entier et le dernier octet de ED en a le moins.

4 Processus de déchiffrement:

Le processus de déchiffrement se compose de quatre étapes: conversion chaîne d'octets-entier, calcul RSA, conversion entier-chaîne d'octets , et analyse de bloc de chiffrement. L'entrée dans le processus de déchiffrement doit être une chaîne d'octets ED, les données chiffrées; un nombre entier n, le module; et entier c, l'exposant. Pour une opération à clé publique , l'entier c doit être l'exposant public d'une entité e; pour une opération de clé privée, elle doit être l'exposant privé d'une entité d. La sortie du processus de décryptage doit être une chaîne d'octets D, les données. C'est une erreur si la longueur des données chiffrées ED n'est pas k.

4.1 Conversion entier-chaîne d'octets:

Les données chiffrées ED doivent être converties en un entier Y, l'entier données chiffrées. C'est une erreur si les données cryptées entières y ne satisfont pas $0 \leq y < n$.

4.2 Calcul RSA:

Les données chiffrées entières Y doivent être élevé à la puissance c modulo n pour donner un entier X, le bloc de chiffrement entier.

$$X = Y^c \mod (n).$$

$0 \leq X < n$.
C'est le calcul RSA classique.

4.3 Conversion entier-chaîne d'octets:

l'entier bloc de chiffrement x doit être converti en une chaîne d'octets EB de longueur k, le bloc de chiffrement

5 Analyse de blocs de chiffrement:

Le bloc de chiffrement EB doit être parsé en un BT,une PS, et les données D selon l'équation de la section (3.1). C'est une erreur si l'une des conditions suivantes se produit:

- Le bloc de chiffrement EB ne peut pas être analysé sans ambiguïté (voir la remarque 2.2).
- La chaîne de remplissage PS comprend moins de huit octets, ou est incompatible avec le type de bloc BT.
- Le processus de déchiffrement est une opération à clé publique et le type de bloc BT n'est pas 00 ou 01, ou une opération de clé privée et le type de bloc est pas 02.

6 Signature et vérification

Le processus de signature par une entité doit se faire avec sa clé privée et la vérification avec la clé publique de l'entité. Le processus de signature transforme une chaîne d'octets (le message) en une chaîne de bits (la signature). La vérification, elle, détermine si une chaîne de bits (la signature) correspond à la bonne chaîne d'octets (le message). Le processus de signature se compose de quatre étapes:

- hachage du message
- codage
- chiffrement RSA
- conversion chaîne d'octets-chaîne de bits.

L'entrée dans le processus de signature est: une chaîne d'octets D (le message) et la clé privée du signataire. Le hachage se fait donc à l'aide d'une fonction de hachage (MD2, MD4, MD5, qui ne sont plus utilisées aujourd'hui). Cette fonction changera donc les données D en haché DH. Le haché obtenu, sera ensuite encodé en une donnée M. C'est sur cette donnée M que l'on appliquera le calcul RSA classique, mais avec la clé privée, pour obtenir un chiffré MC ($MC = M^d \mod n$). Enfin, MC subira une transformation pour devenir une simple chaîne de bits S (la signature).

Pour la vérification, les procédés inverses seront réalisés sur S jusqu'à ré-obtenir DH; on appliquera alors la fonction de hachage sur le message D' reçu pour obtenir DH': si $DH = DH'$, alors l'authentification est réussie.

7 Limite de PKCS#1 v1.5

Malgré son fonctionnement, la version 1.5 de PKCS#1 admet une faiblesse, à cause de laquelle cette version n'est plus utilisée (la version actuelle étant la 2.2).
L'attaque qui à provoqué l'arrêt de cette version est l'attaque du million de messages (MMA, *Million Message Attack*). Cette attaque a été découverte par l'allemand Daniel Bleichenbacher en 1998.
Cette attaque s'appuie sur le fait que les chiffrés envoyés ont un format particulier (voir la section *formatage du bloc de chiffrement*); si le format n'est pas bon, une erreur est renvoyée. L'attaquant va donc utiliser le receveur comme un oracle pour récupérer le message en envoyant des versions transformées du texte chiffré et en observant la réponse du receveur.
Soit le texte chiffré C. L'attaquant va générer une série d'entiers t et calculer $C' = C \times t^e \mod n$. Au déchiffrement, C' produit un prétendu message M'. La plupart des valeurs de M' vont passer à la trappe (à cause du mauvais formattage) mais certaines valeurs de M' auront les deux premiers octets corrects et vont donc paraître correctement formatées en PKCS-1. L'attaque se poursuit en cherchant une séquence de valeurs t telles que le M' résultant soit formaté correctement en PKCS-1. Ces informations peuvent être utilisées pour découvrir M. Pour son fonctionnement, cette attaque exige normalement environ 2^{20} messages et réponses ($2^{20} \simeq 10^6$ d'où le nom de l'attaque).
Comme la MMA exige une telle quantité de messages, elle doit être montée contre une victime qui accepte de traiter un grand nombre de messages. En pratique, aucun humain ne veut lire autant de messages et donc la MMA ne peut être montée que contre une victime automatisée. La MMA exige également que l'attaquant soit capable de distinguer le type d'erreur qu'il provoque.

[1] [2] [3] [4] [5]

References

[1] B. Kaliski (RSA Laboratories East). Pkcs #1: Rsa encryption version 1.5, Mars 1998. https://tools.ietf.org/html/rfc2313.

[2] Wikipedia. Public key cryptographic standards, Septembre 2017. https://fr.wikipedia.org/wiki/Public$_K ey_C ryptographic_S tandards$.

[3] Wikipedia. Pkcs 1, Mars 2017. https://en.wikipedia.org/wiki/PKCS$_1$.

[4] RSA Laboratories. Pkcs#1 v2.2 : Rsa cryptography standard, Octobre 2012. https://www.emc.com/collateral/white-papers/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp.pdf.

[5] Inc E. Rescorla, RTFM. Empêcher l'attaque du million de messages sur la syntaxe de message cryptographique, Janvier 2002. http://abcdrfc.free.fr/rfc-vf/rfc3218.html$_T oc$227517292.