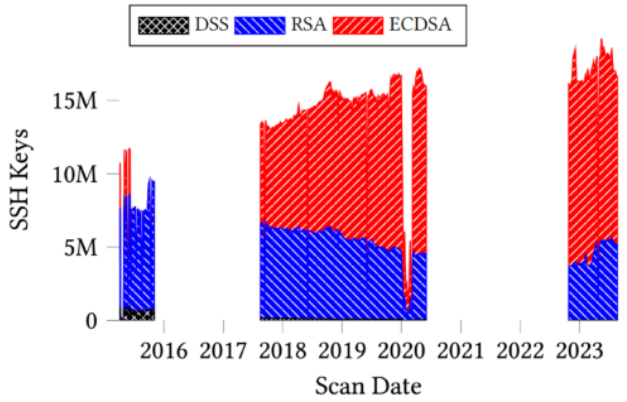
SSE - Passive SSH Key Compromise via Lattices - Ferrara Justin

Les signatures RSA PKCS#1 v1.5 sont considérées comme anciennes et obsolètes (Duc Alexandre, 2024). Cependant, elles restent largement utilisées dans des protocoles tels que SSH et IPsec (Keegan Ryan, Kaiwen He, George Arnold Sullivan, Nadia Heninger, 2023). Leur utilisation persiste notamment parce qu'elles signent un message qui n'est pas transmis directement sur le canal, ce qui rend impossible une attaque directe par calcul du PGCD en cas d'erreur dans la signature.

Ce document, largement inspiré des travaux de Keegan Ryan, Kaiwen He, George Arnold Sullivan et Nadia Heninger (2023), présente une attaque exploitant une erreur de calcul survenant lors de l'utilisation du théorème des restes chinois (CRT) pour la signature RSA PKCS#1 v1.5. Cette faille permet de récupérer la clé privée, même si le message signé reste inconnu de l'adversaire, ce qui est le cas dans des protocoles comme SSH ou IPsec. L'attaque repose sur un modèle d'adversaire passif capable d'observer une seule signature erronée.

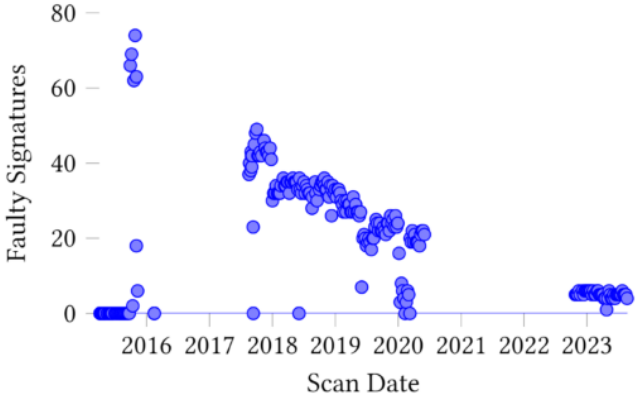
Comme nous pouvons le voir sur le graphique ci-dessous, les signatures RSA représentent encore un tiers environ des signatures SSH collectées par Keegan Ryan, Kaiwen He, George Arnold Sullivan et Nadia Heninger (2023).



Erreurs dans les signatures digitales

Les erreurs pouvant arriver lors du calcul d'une signature peuvent découler de plusieurs facteurs, par exemple d'une mauvaise implémentation, d'une erreur de calcul provoquée par une attaque par canal auxiliaire, etc. Selon l'article, des routeurs ou pare-feu des marques comme Zyxel, SSHD, Mocana ou Cisco peuvent générer de fausses signatures (Keegan Ryan; Kaiwen He; George Arnold Sullivan; Nadia Heninger, 2023). Cependant, la plupart des erreurs ont été corrigées après mise à jour du logiciel.

Comme nous pouvons le voir sur le graphique ci-dessous, les erreurs dans les signatures RSA représentent encore environ 5% des signatures SSH collectées par Keegan Ryan, Kaiwen He, George Arnold Sullivan et Nadia Heninger (2023).



Prérequis pour que l'attaque réussisse

1. Posséder une signature invalide d'un message (posséder le message n'est pas nécessaire).
2. Posséder la clé publique correspondante.
3. L'algorithme étudié ici concerne les signatures RSA PKCS#1 V1.5.
4. Les signatures sont réalisées en utilisant le théorème des restes chinois.
5. L'erreur dans la signature provient d'une des opérations faites dans le monde des tuples (théorème des restes chinois).

SSH

SSH est un protocole de communication sécurisé permettant d'établir une connexion chiffrée entre un client et un serveur. Il est principalement utilisé pour l'administration système, le transfert de fichiers et l'exécution de commandes sur des machines distantes.

Authentification du serveur

Les serveurs SSH sont identifiés par leurs clés publiques. L'échange commence par une négociation des différents algorithmes utilisés par la suite suivie d'un échange de clés Diffie-Hellman. Le serveur s'authentifie en signant le session identifier avec sa clé privée, vérifié ensuite par le client. Le

session identifier contient le secret partagé du D-H, tous les messages échangés (algorithmes, etc.), l'ID du client et l'ID du serveur.

Authentification du client

Elle se déroule après l'établissement du canal chiffré. Il y a deux méthodes possibles, par mot de passe ou par clé publique. Pour l'authentification par mot de passe, il est envoyé tel quel à travers le canal chiffré. Pour l'authentification par clé publique, le client signe l'identifiant de session avec sa clé privée.

Algorithmes cryptographiques

Pour l'échange des clés, différents algorithmes sont à choix notamment Diffie-Hellman, Elliptic Curve Diffie-Hellman (ECDH), RSA.
Pour les signatures, différents algorithmes sont à choix notamment DSA, RSA, ECDSA, Ed25519.

Conséquences d'une compromission de clés de signatures

Une compromission des clés de signatures ne permet pas de déchiffrer les connexions de manière passive mais peut permettre une attaque active (Man-in-the-middle). Elle peut permettre également à un attaquant d'usurper l'identité du serveur et établir une connexion chiffrée avec le client pour obtenir son mot de passe par exemple.

Exploitation des méthodes d'authentification

Lors d'une authentification par mot de passe, un attaquant peut effectuer un Man-in-the-middle en usurpant l'identité du serveur pour intercepter et relayer les identifiants et mot de passe du client. Lors d'une authentification par clé publique, cela n'est pas possible. Pour faire un Man-in-the-middle complet il faudrait également compromettre la clé de signature du client. L'attaquant peut cependant connaître les commandes que le client aurait envoyé au serveur.

IPsec

IPsec est un ensemble de protocoles (RFC 2408, 2409, 7296) visant à garantir la confidentialité, l'intégrité des données et l'authentification des sources des paquets IP. Il est largement utilisé par les VPN et repose sur le protocole Internet Key Exchange (IKE) pour négocier les algorithmes utilisés, définir comment dériver les clés, etc.

IKE : Versions et fonctionnement

IKE existe en deux versions : IKEv1 et IKEv2. IKE permet l'établissement d'une Security Association (SA) et l'authentification mutuelle. Une SA permet le choix des algorithmes et l'échange initial de clés Diffie-Hellman. L'authentification mutuelle est effectuée entre l'initiateur et le répondeur via différentes méthodes (signatures, clés pré-partagées, etc.).

IKEv1

IKEv1 prend en charge trois modes d'authentification, les signatures numériques, par clés publiques ou des clés pré-partagée (PSK). Il y a également deux modes de communication, soit le Main Mode qui est le plus sécurisé, toutes les communications sont chiffrées après l'échange initial soit le mode agressif, où l'échange initial est réduit mais moins sécurisé (les signatures sont envoyées en clair).

Sécurité et attaques possibles

Avec le Main Mode, une attaque passive seule n'est pas possible car elle ne permet pas de collecter des signatures du fait qu'elles sont transmises de manière chiffrée. Pour recevoir des signatures, un adversaire doit se connecter de manière active au serveur.

Avec le Mode Agressif, un attaquant peut capturer les signatures envoyées en clair par le client et le serveur en écoutant passivement la communication.

À noter qu'un format non standard de signatures RSA est utilisé. RFC 2409 impose une variation du format PKCS#1 v1.5, supprimant l'OID du hachage car il est inclus dans la SA, ce qui diminue la partie connue de la signature et augmente légèrement la complexité de l'attaque.

En cas de compromission des clés de signatures, un attaquant pourrait usurper l'identité du client ou serveur. Pour pouvoir faire une attaque Man-in-the-middle, il faut compromettre la clé privée de signature des deux parties à cause du D-H qui est signé.

IKEv2

IKEv2 n'est pas compatible avec IKEv1 et introduit certaines modifications notamment le fait que toutes les signatures sont chiffrées et transmises dans le payload AUTH après l'établissement de la SA. L'extensible Authentication Protocol (EAP) permet d'obtenir de manière active une signature de l'hôte distant sans authentifier l'initiateur de la connection. L’authentification par clés cryptographiques peut concerner qu'un participant en fonction des modes utilisés.

Sécurité et attaques possibles

En cas d'une compromission d'une clé de signature, un attaquant peut usurper l'identité du détenteur de la signature. Pour pouvoir faire une attaque Man-in-the-middle, il faut compromettre les clé privées des deux parties.

Pour compromettre la clé de la deuxième partie nous pouvons utiliser d'autres attaques notamment en cas d'utilisation de clés pré-partagées (PSK) pour l’authentification, un attaquant peut, si la PSK est faible (mot de passe faible), mener une attaque par dictionnaire hors ligne. Ou, par exemple, avec certains modes EAP (comme EAP-MS-CHAPv2) qui utilisent des protocoles dont la sécurité est compromise par d'autres attaques.

PKCS#1 V1.5 padding pour signature RSA

> Note : toutes les formules mathématiques suivantes sont dérivées de l'article Keegan Ryan, Kaiwen He, George Arnold Sullivan, Nadia Heninger, 2023.

Le padding PKCS#1 V1.5 utilisé par les signatures est donné par :

$$00||01||FF...FF||ASN.1||Hash(m)$$

En sachant que :

- `ASN.1` est un identifiant pour définir la fonction de hachage utilisée
- `Hash(m)` est le hash du message `m`

Expression du padding

Pour le padding d'un message `m` inconnu, le padding PKCS#1 V1.5 peut être exprimé de la manière suivante :

$$a + x$$
$$x < 2^{hash\_len}$$

Avec `a` la partie connue du padding et `x` représentant la partie inconnue qui est le hash du message, borné par la taille de sortie de la fonction de hachage. En posant cette équation, il est possible de retrouver la clé privée de signature si `hash_len <= log(N) / 4`.

RSA Signatures avec padding PKCS#1 V1.5

La signature `s` RSA d'un message `m` sans padding est donnée par :

$$s = f(m)^d \mod N$$

La vérification de cette signature est donnée par :

$$f(m') = s^e \mod N$$
$$f(m) =? f(m')$$

Avec `f(m)` la fonction de padding du message.

> À noter que dans ce cas la signature doit être faite en utilisant le théorème des restes chinois.

PACD (Partial Approximate Common Divisors)

PACD est une généralisation du PGDC :

$$N_i = p_i \cdot q_i + r_i$$

Pour pouvoir résoudre le problème PACD, il faut que les diviseurs communs approximatifs soient suffisamment proches pour que l'algorithme LLL puisse les trouver. Nous pouvons donc essayer de poser les équations suivantes :

$$N_0 = p_0 \cdot q_0 + r_0$$
$$N_1 = p_1 \cdot q_1 + r_1$$

Que nous pouvons réécrire ainsi:

$$N = p \cdot q_0$$
$$N_1 = p_1 \cdot q_1 + r_1$$

Avec `r_1` une erreur de calcul et `h` l'espace de la fonction de hachage (par exemple pour SHA-256 alors h = 2^256).

$$|r_1| < 2^{log_2(h)}$$

Comme nous n'avons pas `N1`, nous pouvons essayer de le trouver en utilisant les équations qui définissent la padding PKCS#1 V1.5.

$$s'^e = m' = a + x \mod N$$

Nous pouvons ensuite poser le padding moyen :

$$mean\_pad = 00||01||FF...FF||ASN.1||100...$$

Nous pouvons ensuite soustraire ce padding moyen à notre signature fausse :

$$(s'^e - mean\_pad \mod N) = p \cdot k + r_1 \mod N$$

Nous pouvons donc déduire les équations suivantes :

$$N_0 = p \cdot q_0$$
$$N_1 = (s'^e - mean\_pad \mod N) = p \cdot k + r_1$$

Lattice

Nous pouvons ensuite remplir la matrice suivante pour construire la lattice :

$$B = \begin{bmatrix} -2^{2(\log(h)-1)} & 2^{\log(h)-1}N_1 & 0 \\ 0 & -2^{\log(h)-1} & N_1 \\ 0 & 0 & N_0 \end{bmatrix}$$

Avec `r` l'espace de la fonction de hachage.

LLL

Pour réduire la matrice précédente, nous pouvons utiliser l'algorithme LLL pour essayer de trouver des vecteurs avec une base plus courte et plus proche de l'orthogonalité que la base d'entrée:

- Appliquer la réduction de Gram-Schmidt afin de décomposer chaque vecteur.
- Modifier légèrement les vecteurs en les ajustant, en remplaçant certains vecteurs par des combinaisons linéaires des autres vecteurs.
- Vérifier la condition de Lovász pour chaque vecteur.

À la fin de l'algorithme, on obtient une base plus courte et plus proche de l'orthogonalité que la base d'entrée.

$$reduced\_matrix = LLL(B)$$

$$reduced\_matrix = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$$

Si la réduction a fonctionné, nous pouvons retrouver un vecteur `v` qui peut être interprété comme les coefficients du polynôme suivant :

$$\vec{v}$$

$$g(2^{\log(h-1)}x)$$

$$g(x) = a_i * \frac{x^2}{2^{2*log(h-1)}} + b_i * \frac{x}{2^{log(h-1)}} + c_i$$

Si les coefficients sont assez petits, on peut poser les équations suivantes :

$$g(y) = 0$$

$$y = r_1$$

Avec `r_1` une petite racine mod p de `g(x)` .

Comme nous connaissons `r_1` , nous pouvons donc déduire `p` :

$$p = \gcd(N, N_1 - r_1)$$

> Note : LLL trouvera une solution seulement si `log(h) <= log(N) / 4` .

Digression de l'attaque

L'attaque est possible si dans le message représentant le message original, il y a une partie connue et une partie inconnue avec une erreur dans la signature. Je peux donc déjà exclure les algorithmes suivants:

- Tous les algorithmes de signatures basés sur les courbes elliptiques
- RSA PSS

Mesures de protections

- Mettre à jour les équipements pour éviter les implémentations générant des signatures erronées.
- Valider les signatures effectuées pour vérifier qu'elles ne contiennent pas d'erreur avant de les envoyer et, dans le cas contraire, en refaire de nouvelles.
- Ne pas utiliser un padding déterministe mais plutôt utiliser un algorithme de signature sûr, par exemple RSA-PSS.
- Conception du protocole :
  - Chiffrer la communication le plus tôt possible dès que les clés cryptographiques sont disponibles pour protéger les meta datas et les signatures.
  - Se baser sur TLS 1.3 pour plus de détails.

Bibliography

- **Denis Bider**. *RFC 8332: Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol*. RFC Editor, USA, 2018. https://doi.org/10.17487/RFC8332
- **Duc, Alexandre, 2024**. *Asymmetric Cryptography Standards* [PDF]. Support de cours : Cryptographie avancée appliquée, HEIG-VD, 2024.
- **Keegan Ryan, Kaiwen He, George Arnold Sullivan, Nadia Heninger**. *Passive SSH Key Compromise via Lattices*. Cryptology ePrint Archive, Paper 2023/1711, 2023. DOI: 10.1145/3576915.3616629, URL: https://eprint.iacr.org/2023/1711.
- **Tatu Ylonen**. *SSH - Secure Login Connections over the Internet*. Proceedings of the 6th USENIX Security Symposium, pp. 37-42, USENIX, 1996.
- *ssh_config(5) - OpenBSD manual pages*. (s. d.). https://man.openbsd.org/ssh_config.5