Fériel BOUAKKAZ

PROJET CRYPTOGRAPHIE ATTAQUES SUR UN CRYPTO SYSTEME SYMETRIQUE : BLOW FISH

Joël ASSOGO KOUNGOU

Table des matières

1. Introduction 3
1.1. Les crypto systèmes symétriques 3
1.2. Cryptanalyses 3
2. Le système Blow Fish 4
2.1. Qu'est-ce que c'est ? 4
2.2. Fonctionnement de l'algorithme 4
2.3. Algorithme similaire 6
2.4. Cryptanalyse 7
2.5. Cas d'utilisation courante 7
3. Exemple précis d'attaque sur système Blow Fish 8
3.1. L'attaque Sweet32 8
3.1.1. TLS 8
3.1.2. OpenVPN 9
3.2. L'attaque de Kara et Manap 10
4. Annexes 11
4.1. Définitions 11
4.2. Sources 12

1. Introduction

1.1. Les crypto systèmes symétriques

Le cryptage symétrique, également connu sous le nom de cryptage à clé secrète, est la plus ancienne forme de cryptage. Cette méthode de cryptage est différente du cryptage asymétrique. Dans le cryptage asymétrique, une paire de clés (une clé publique et une clé privée) est utilisée pour crypter et décrypter les messages. Les entités communiquant par cryptage symétrique doivent échanger des clés (clés publiques) afin de pouvoir être utilisées dans le processus de décryptage.

Les algorithmes de chiffrement symétrique permettent de convertir les données en une forme que personne ne peut comprendre ou déchiffrer sans clé secrète. Une fois que le destinataire possédant la clé a reçu le message, l'algorithme annule son opération pour renvoyer le message sous une forme originale et facile à comprendre. La clé secrète utilisée à la fois par l'expéditeur et le destinataire peut être un mot de passe / code spécifique, ou une chaîne aléatoire de lettres ou de chiffres générée par un générateur de nombres aléatoires sécurisé.

1.2. Cryptanalyses

La cryptanalyse est l'étude des textes chiffrés, des nombres et des systèmes cryptographiques. Le but est de décrypter les textes, les principes de fonctionnement des nombres et des systèmes cryptographiques, et de trouver et d'améliorer les techniques pour les vaincre ou les affaiblir. Par exemple, un cryptanalyste tente de déchiffrer un mot de passe sans connaître la source du texte en clair, la clé de chiffrement ou l'algorithme utilisé pour le chiffrer. Le cryptanalyste cible également le hachage sécurisé, les signatures numériques et d'autres algorithmes cryptographiques.

Bien que le but de la cryptanalyse soit de découvrir les faiblesses des algorithmes cryptographiques ou de les rendre invalides, les résultats des recherches des cryptographes sont utilisés par les cryptographes pour améliorer et renforcer ou remplacer les algorithmes présentant des défauts. La cryptanalyse, est la partie déchiffrage des données chiffrées, et la cryptographie, est la partie création et l'amélioration des codes de chiffrement et autres algorithmes. Ils sont toutes deux des aspects de la cryptologie, l'étude mathématique des codes, des chiffres et des algorithmes connexes.

La cryptanalyse est la technique qui permet d'extraire un texte en clair à partir d'un texte chiffré sans la clé de chiffrement. Le processus consistant à essayer de comprendre un message particulier est appelé une attaque. L'attaque est généralement caractérisée en fonction des données requises pour l'attaque :

- Attaque sur texte chiffré seul (ciphertext-only) : le cryptanalyste est en possession d'exemple chiffré de message, lui permettant de des hypothèses sur les messages originaux dont il n'est pas en possession. Le manque d'informations disponibles rend l'analyse cryptographique plus difficile.
- Attaque à texte clair connu (known-plaintext attack) : le cryptanalyste connait une partie et/ou les messages en clair et chiffrés. Exemple : La cryptanalyse linéaire
- Attaque à texte clair choisi (chosen-plaintext attack) : Le cryptanalyste a des messages clairs et il peut utiliser des algorithmes pour créer des versions chiffrées de ces messages. Exemple : La cryptanalyse différentielle
- Attaque à texte chiffré choisi (chosen-ciphertext attack) : Le cryptanalyste possède des messages chiffrés et demande des versions non chiffrées de certains d'entre eux pour lancer une attaque.

2. Le système Blow Fish

2.1. Qu'est-ce que c'est ?

Blowfish est un algorithme de cryptage symétrique conçu par Bruce Schneier en 1993 et constituant une alternative aux algorithmes de chiffrement existants.

Il utilise une taille de bloc de 64 bits et la clé de longueur variable peut aller de 32 à 448 bits. L'algorithme de Blowfish est basé sur l'idée qu'en utilisant de très grandes clés pseudo-aléatoires on obtient une bonne sécurité contre les attaques de cryptanalyse.

Même si Blowfish date de 1993, il reste encore assez résistant aux attaques sur les versions avec moins de tours. De nos jours la version complète de Blowfish avec 16 tours est extrêmement fiable et le seul moyen de l'attaquer est la recherche exhaustive.

Blowfish n'est pas protégé par des brevets et peut être utilisé gratuitement sans autorisation. Toutes les utilisations sont rendues publiques par son créateur. Cela explique en partie son succès car il a été l'un des premiers algorithmes de chiffrement à être utilisé gratuitement. Il est utilisé dans de nombreux logiciels propriétaires et libres.

2.2. Fonctionnement de l'algorithme

Blowfish est un chiffrement par blocs, ce qui signifie qu'il divise le message en blocs de longueur fixe pendant le processus de chiffrement et de déchiffrement. La longueur de bloc de Blowfish est de 64 bits ; les messages dont la taille n'est pas un multiple de 8 octets doivent être remplis.

Blowfish a besoin d'environ 5 Ko de mémoire. Une mise en œuvre minutieuse sur un processeur 32 bits peut crypter ou décrypter un message 64 bits en environ 12 cycles d'horloge. Des messages plus longs augmentent linéairement le temps de calcul. Par exemple, un message de 128 bits nécessite environ (2 x 12) horloges. Blowfish utilise une clé d'une longueur maximale de 448 bits.

Pour procéder au déroulement de l'algorithme Blowfish, nous avons besoins de

- Un message en clair de 64 bits
- Un tableau P (P-array) :P est un tableau de dix-huit entiers 32 bits.
- Un tableau S (S-box) :S est un tableau bidimensionnel d'entiers 32 bits de dimension 4 × 256.

Les deux tableaux sont initialisés avec des constantes, qui se trouvent être les chiffres hexadécimaux de π (une source de nombres aléatoires assez décente).

D'abord le message en clair de 64 bits est divisé en 2 blocs de 32 bits. On appellera les 32 bits de « gauche » par L et les 32 bits de « droite » par R. Ensuite on applique un XOR entre le message L et le premier élément du P-array pour créer une valeur qu'on appellera L '.

$\rightarrow L \oplus P1 = L'$

Par la suite on exécutera une fonction de transformation appelée F sur le message obtenu précédemment « L' », le résultat de cette fonction sera appeler F'. Puis on appliquera un XOR entre le F' et avec le message R, ainsi on obtient un une valeur qu'on appellera R'.

→ *F(L') = F'*

→ *F' ⊕ R = R'*

R ' remplace alors la moitié « gauche » (L) du message et L' remplace la moitié « droite ». Ce processus représente une boucle et est répété 15 fois de plus avec les membres successifs du P-array.

On applique ensuite un XOR entre les R' L' résultants et P17 P18, les deux dernières entrées du P-array respectivement, et concaténer pour produire le texte chiffré à 64 bits.

→ *(L' ⊕ p18) + (P17 ⊕ R') = texte chiffré*

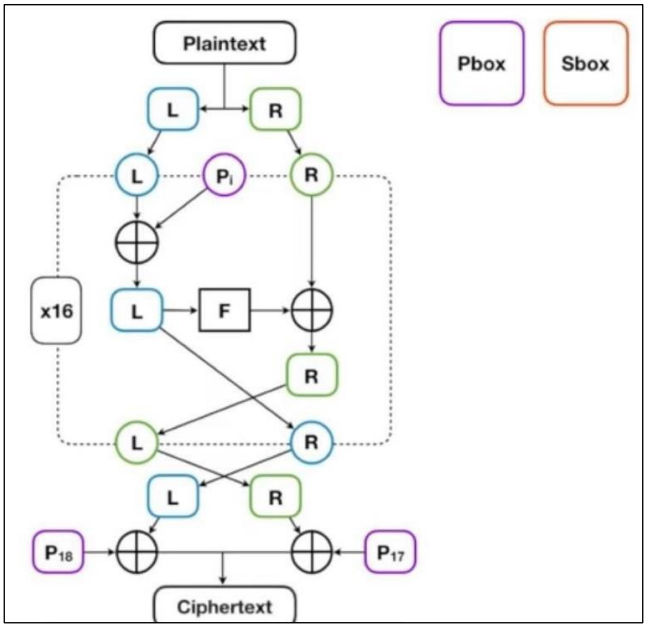
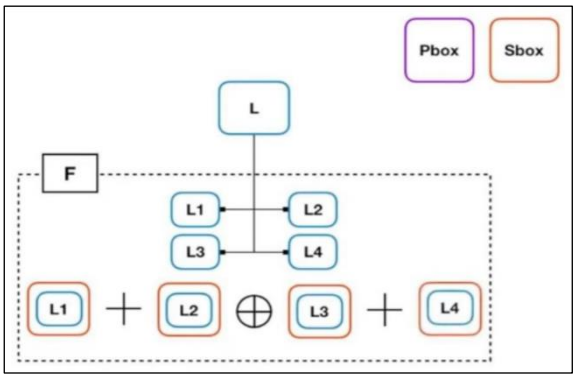


Schéma de l'algorithme de Blowfish

Représentation graphique de F

La fonction divise une entrée 32 bits en quatre octets et les utilise comme indices dans un S-array, après avoir obtenue le message L' on divise ce dernier en 4. On appellera les parties des messages respectivement L1, L2, L3, L4 , ces messages seront respectivement mis dans des s-box L1 sera additionné a L2 ensuite le résultat sera ⊕ L3 et le résultat de ces dernières sera additionné à L4 pour nous donner le message F.



→ *L1 + L2 = L12*

Schéma de la fonction F

→ *L12 ⊕ L3 = L123*

→ *L123 + L4 = F'*

Parce que Blowfish est un algorithme symétrique, la même procédure est utilisée pour le décryptage ainsi que le cryptage. La seule différence est que l'entrée du chiffrement est en texte brut; pour le déchiffrement, l'entrée est le texte chiffré.

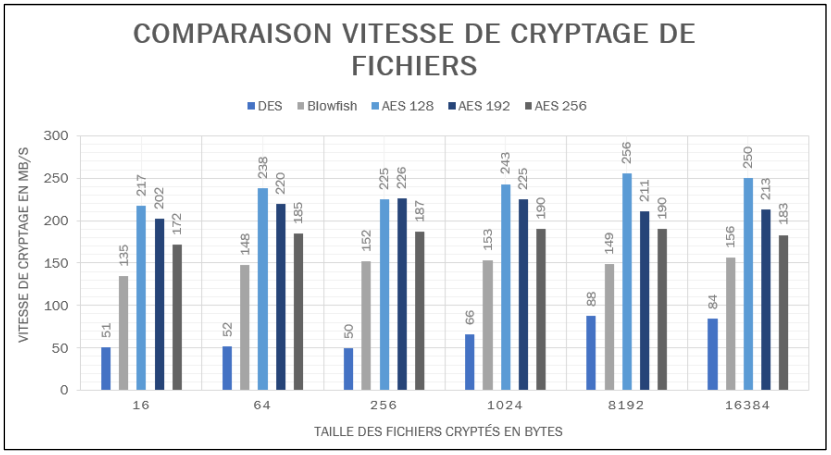
Les valeurs P-array et S-array utilisées par Blowfish sont précalculées en fonction de la clé de l'utilisateur. En effet, la clé de l'utilisateur est transformée en P-array et S-array; la clé elle-même peut être supprimée après la transformation. Le P-array et le S-array n'ont pas besoin d'être recalculés (tant que la clé ne change pas), mais doivent rester secrets.

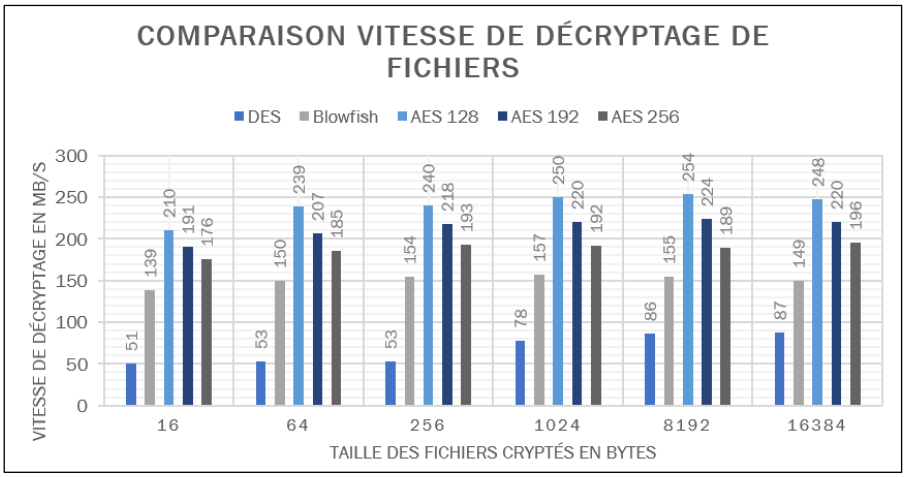
2.3. Algorithme similaire

Blowfish a un certain nombre d'algorithme concurrent plus ou moins courant. Les principaux algorithmes de chiffrement à bloc s'opposant à Blowfish sont : AES · DES · Serpent · Triple DES · Twofish. Twofish a un cryptage basé sur cet algorithme et adopte en partie le concept proposé par le même auteur dans le populaire Blowfish.

Afin de pouvoir tester l'efficacité de BlowFish, nous l'avons comparé à d'autres algorithmes symétriques. Sur un ordinateur Linux Ubuntu avec 4Go de Ram nous avons tester la vitesses de cryptage et de décryptage de plusieurs algorithmes. Ce test a été réaliser sur Openssl grâce à la commande Openssl speed. L'option speed nous permet de tester la fonction du système en utilisant différents algorithmes de cryptage pour crypter les données dans une période de temps donnée.

Les test ont été réaliser avec les algorithmes DES, BlowFish, AES 128, 192 et 256; en chiffrement par bloc avec le mode d'opération d'enchainement des blocs. L'algorithme de Twofish n'a pas pu être tester car il n'était pas pris en charge par l'outil Openssl. Les nombres présents dans les tableaux correspondent au nombre de fichier qu'un algorithme peut crypter ou décrypter par seconde. On calcul donc le temps de cryptage et le temps de décryptage des algorithmes. Par exemple dans le tableau un on peut voir que Blowfish est capable de crypter 152 fichiers de taille 256bytes par seconde.





D'après ces analyses, on s'aperçoit que l'algorithme AES est plus rapide que Blowfish, dans le cryptage et le décryptage. L'algorithme de Blowfish est tout de même plus rapide que celui de DES.

2.4.Cryptanalyse

Il n'existe actuellement aucune cryptanalyse efficace de Blowfish mettant en évidence la force de l'algorithme.

En 1996, Serge Vaudenay a conçu une méthode d'attaque de type " texte clair connu" permettant de connaître $2^{8r} + 1$ texte brut avec la même clé, afin que le texte puisse être forcé à être chiffré, où r indique le nombre de cycles. De plus, il connaissait le texte en clair $2^{4r} + 1$, il a donc trouvé une classe de clés faibles qui pourraient être identifiées et détruites par la même attaque. Cette attaque ne peut pas être utilisée pour les Blowfish ordinaires qui se déplacent tous les 16 cycles ; elle suppose la connaissance des S-box dépendantes des clés. Vaudenay utilise une variante basée sur moins de cycles de cryptage. Vincent Rijmen, dans sa thèse de doctorat, a introduit une attaque différentielle de second ordre qui peut briser quatre coups et pas plus. Il n'existe toujours pas de moyen connu de briser les 16 coups, à l'exception d'une recherche par la force brute.

En 2005 Dieter Schmidt a étudié la liste des clés de Blowfish et a noté que les sous-clés du cycle troisième et quatrième sont indépendantes des 64 premiers bits de la clé de l'utilisateur.

2.5.Cas d'utilisation courante

OpenBSD est un système d'exploitation gratuit de type Unix fondée en 1994 par Theo de Raadt. Le projet OpenBSD est connu pour sa transparence de la liberté des logiciels et du code source, la qualité de sa documentation et l'accent mis sur la sécurité et la cryptographie embarquée.

OpenBSD propose deux algorithmes de coût paramétrable à utiliser avec les mots de passe.

- Eksblowfish : un chiffrement par bloc permet de stocker en toute sécurité des données sur le disque. Provos et Mazières ont conçu un nouvel algorithme d'établissement de clé appelé Eksblowfish (utilisé pour la planification coûteuse des clés Blowfish). Dans cet algorithme, la première étape consiste à créer une sous-clé à l'aide de la clé et du sel. Ensuite, utilisez une alternance de sel et de clés pour transformer de nombreux algorithmes de Blowfish standard. Chaque tour commence à partir de l'état de sous-clé du tour précédent. Cela ne rend pas l'algorithme plus puissant que la version standard de Blowfish, mais il est possible de choisir le nombre d'itérations pour le ralentir arbitrairement et aider à prévenir les tables arc-en-ciel et les attaques par force brute. Le nombre d'itérations doit être une puissance de 2, qui est un paramètre de l'algorithme, et le nombre d'itérations est encodé dans le résultat final.

- Bcrypt a été utilisé dans le cadre d'OpenBSD pour l'authentification par mot de passe des systèmes d'exploitation. bcrypt est une fonction de hachage créée par Niels Provos et David Mazières. Il est basé sur l'algorithme de cryptage Blowfish et a été exposé sur USENIX en 1999. Bcrypt est un algorithme lent, il réduit donc le nombre de mot de passe par seconde qu'un attaquant peut hacher lors de la création d'une attaque par dictionnaire, il nécessite également un sel dans le processus de hachage. Le hachage combiné avec des sels nous protège contre les attaques de table arc-en-ciel.

3. Exemple précis d'attaque sur système Blow Fish

3.1. L'attaque Sweet32

3.1.1. TLS

Karthikeyan Bhargavan et Gaëtan Leurent sont deux chercheurs de l'INRIA, l'institut national français de recherche en informatique. Ils ont créé une attaque sur des chiffrements 64 bits qui peut permettre à l'attaquant de récupérer les données en clair de l'utilisateur, sans connaître la clé de chiffrement. Cette attaque a été baptisé Sweet32.

L'attaque Sweet32, nécessite des conditions particulières, comme des serveurs utilisant les chiffrements 3DES et Blowfish en mode CBC (Cipher Block Chaining), et l'attaquant parvenant à obtenir une position pour analyser le trafic entre les deux parties.

De plus, le serveur auquel le client se connecte doit prendre en charge une session TLS persistante et la clé de déchiffrement n'est pas obligatoire lors de l'attaque Sweet32.

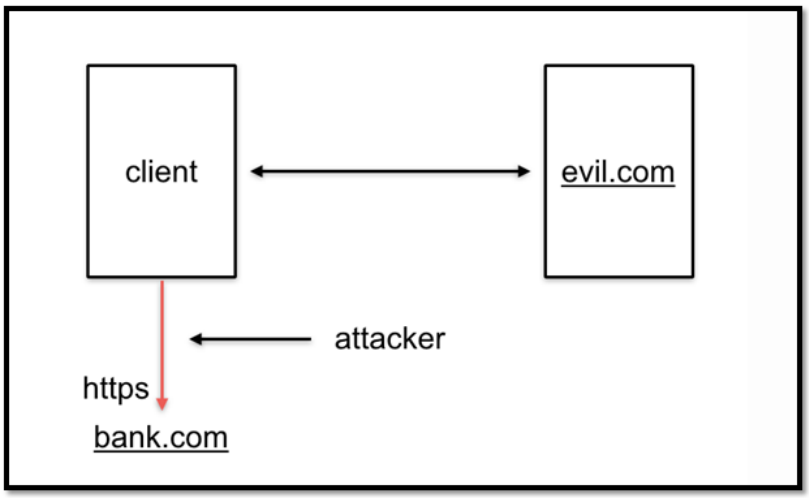
Si l'attaquant est capable de surveiller les connexions TLS et HTTPS à longue durée de vie négociées via les suites de chiffrement vulnérables, il peut utiliser des fichiers JavaScript malveillants insérés côté client (via des publicités ou des logiciels malveillants) pour envoyer un ping au serveur avec un nombre élevé de requêtes.

Ces demandes sont généralement accompagnées de fichiers de cookies HTTP, intégrés dans le flux de données HTTPS, utilisés pour authentifier le client.

Les chercheurs disent qu'après avoir envoyé des requêtes de serveur constantes pendant 30 à 38 heures et collecté environ 785 Go de trafic, ils seront en mesure de repérer une attaque par collision, ce qui leur permettra de récupérer le contenu du fichier cookie, contenant l'authentification et détails de la session utilisateur.

Ces détails peuvent ensuite être utilisés pour se connecter au compte de l'utilisateur. Certains des sites que les chercheurs ont trouvés vulnérables aux attaques Sweet32 incluent eBay, Walmart, NASDAQ et une série de banques du monde entier.

Dans le cas de TLS, l'attaque peut être effectuée activement au lieu d'observer passivement de grandes quantités de communication. Les attaquants peuvent utiliser les mêmes types de techniques basées sur JavaScript que celles utilisées dans BEAST pour exploiter Sweet32.



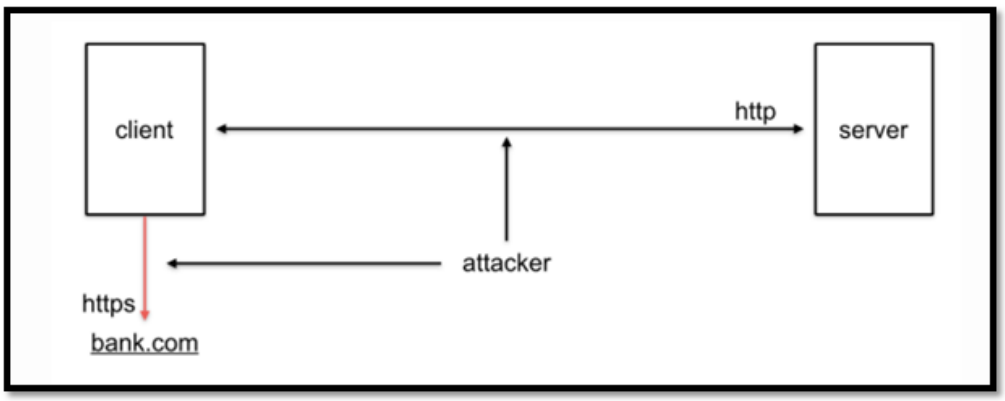
3.1.2. OpenVPN

Les chercheurs ont également démontré une attaque similaire contre les VPN, plus précisément sur OpenVPN, où des connexions Blowfish à longue durée de vie sont établies par défaut. OpenVPN est une solution VPN à source ouverte populaire, écrite à l'origine par James Yonan. Le cryptage par défaut du protocole de transport d'OpenVPN est Blowfish - un chiffrement de 64 bits - avec le mode CBC. OpenVPN supporte deux modes différents pour générer des clés de session afin de chiffrer les messages :

- En mode pré-partage de clés, des clés statiques sont utilisées pour tout le trafic. En particulier, il n'y a pas de limite à la durée de vie de ces clés.
- En mode TLS, les clés de session sont générées avec une poignée de main TLS, en utilisant des certificats pour authentifier les pairs. Les clés de session sont mises à jour périodiquement, avec des limites sur le nombre de paquets, le nombre d'octets, ou une durée de session. La configuration par défaut permet de relancer le tunnel toutes les heures.

Pour démontrer l'attaque contre OpenVPN, les chercheurs ont utilisés un tunnel à clés pré-partagées entre deux machines physiques fonctionnant sous Linux, avec Firefox Developer Edition 47.0a2 d'un côté, et un serveur nginx de l'autre. L'accès au serveur est protégé par BasicAuth et l'utilisateur a saisi ses informations d'identification.

Les chercheurs ont utilisés un code Javascript pour envoyer un grand nombre de requêtes au serveur via le tunnel. Ils ont constaté que l'augmentation de la taille de la requête à 4~kB ne réduit pas de manière significative le taux de requête, mais réduit le nombre de requêtes requises pour l'attaque. Dans leurs configuration, le navigateur génère environ 2900 requêtes par seconde, en utilisant plusieurs web Worker fonctionnant en parallèle. Ils prévoient la première collision après environ 232,3 blocs (40 Go), soit une heure. En pratique, ils ont détecté la première collision tôt, après seulement 30 minutes (231,3 blocs) ; comme prévu, la différence xor prévue était correcte. L'attaque complète nécessite environ 236,6 blocs (785 Go) pour récupérer un secret de deux blocs sur des messages de 4 kB ; cela devrait prendre environ 19 heures dans ce cadre. Dans cette démo, cela a pris 18,6 heures et 705 Go, et ils ont réussi à récupérer le jeton d'authentification de 16 octets.



3.2. L'attaque de Kara et Manap

Schneier n'a pas affirmé que Blowfish n'avait pas de clés faibles, mais il a dit que s'il y avait des clés faibles, la probabilité qu'une clé soit choisi au hasard devrait être faible. Kara et Manap ont montré qu'il existe des touches faibles pour l'algorithme Blowfish . Dans son document original sur Blowfish, Schneier a déclaré que toutes les clés faibles devraient être décrites et classées afin qu'ils puissent être exclus au cours du processus de génération de clés. Le fait que l'algorithme soit encore largement utilisé rend l'étude de ses clés faibles importante.

Dans leur étude des clés faibles, Kara et Manap ont reformulé l'algorithme standard de Blowfish. L'auteur n'a pas examiné tous les détails concernant l'équivalence des deux versions de Blowfish. L'opération XOR est commutative, et Kara et Manap affirment que cette propriété de XOR est ce qui permet la modification à l'algorithme original. La fonction F n'étant pas commutative, la modification ne peut être effectuée

Étant donné que l'algorithme BlowFish équivalent mais reformulé se compose principalement des deux types de blocs, les résultats concernant le nombre de points fixes peuvent être étendus à une bonne partie de l'algorithme. En examinant l'algorithme reformulé, si la toute première étape et la dernière étape sont laissées de côté, il y a certaines clés qui feront que l'algorithme aura 2^{32} points fixes. Reformulé, cela signifie qu'il y a 2^{32} plaignants (paires ordonnées (L; R) où L et R sont des chaînes de 32 bits) qui resteront inchangées par ces tours. Toute clé qui permettra que cela se produise est appelé une clé faible réfléchissante.

L'attaque de Kara et Manap comporte deux parties. La première partie concerne l'identification d'une clé comme une clé réfléchie faible. Une fois par réflexion la clé faible est identifiée, la deuxième partie du l'attaque détermine le tableau P. Une fois le tableau P récupéré, la clé peut être reconstruite.

Supposons qu'une clé réfléchissante faible est utilisée dans l'algorithme Blowfish. Cela signifie que si le premier et le dernier tour de la reformulation L'algorithme Blowfish est laissé de côté, il y aura 2^{32} points fixes. Si un attaquant veut tester pour voir si une clé réfléchissante faible a été utilisé, l'attaquant doit connaître de nombreuses paires de texte en clair. Soit (x, y) le texte brut et soit (x ′, y ′) désignent le correspondant texte chiffré, et supposons que (x, y) est l'un des points fixes de l'algorithme. Revenant à la description reformulée de Blowfish, les étapes ressemblerait à ceci:

$$(x1, y1) = (P1 \oplus x, P2 \oplus y)$$
$$(x2, y2) = \ldots$$
Point fixe ⇒ Rien ne se passe vraiment ici !
$$\ldots$$
$$(x17, y17) = (P1 \oplus x, P2 \oplus y)$$
$$(X', y') = (P18 \oplus P1 \oplus x, P17 \oplus P2 \oplus y)$$
Il s'ensuit que $x' \oplus x = P1 \oplus P18$ et que $y' \oplus y = P2 \oplus P17$.

Si un attaquant en a assez les paires en clair-texte chiffré, alors l'attaquant peut tester pour voir si une clé réfléchissante faible a été utilisé. Pour toutes les paires de texte clair-chiffré, calculez $(x' \oplus x, y' \oplus y)$.

Si une clé réfléchissante faible a été utilisé, puis la paire $(P1 \oplus P18, P2 \oplus P17)$ se produira plus fréquemment car ce sera le résultat pour chacun des 2^{32} points fixes.

Certes, cela prend beaucoup de paires de texte en clair. Kara et Manap déclarent que 2^{32} de telles paires sont nécessaires. L'auteur a essayé pour déterminer exactement pourquoi 2^{34} de ces paires sont nécessaire, mais n'a aucun résultat à afficher pour le moment investi.

Sachant qu'une clé faible et réfléchie est utilisé donne automatiquement des informations concernant le tableau P. Rappelez-vous que les petits blocs qui ont des points fixes ont des parties du P tableau qui sont identiques. Cela donne la moitié du 18eme éléments du tableau P.

Kara et Manap ont déclaré que « en devinant r/2 + 1 sous-clés que nous pouvons déterminer restant r/2 + 1 sous-clés et obtenez

L'ensemble du tableau P », mais ils ne donnent aucune information sur une méthode pour deviner le reste sous-clés. Une fois tous les éléments du tableau P sont déterminés, un attaquant n'aurait besoin qu'effectuer 9 chiffrements pour récupérer la clé.

Kara et Manap affirment que les propriétés de Blowfish qui permettent cette attaque sont la similarité des fonctions du cycle. En fait, dans la norme description de l'algorithme Blowfish, tous les tours sont exactement les mêmes. Une approche peut être de modifier l'action de chaque cycle en fonction sur le tableau P. Par exemple, utilisez des de multiples éléments du tableau P à chaque tour, peut-être choisis au hasard. Un inconvénient de cette approche est que pour décrypter, le nombre et l'ordre des éléments du P utilisé à chaque tour devrait être.

4. Conclusion

Blowfish est un algorithme de chiffrement symétrique assez robuste, le principe de son algorithme est de découper le texte en claire et le mélanger 16 fois afin de crypté le message. Blowfish est particulièrement solide contre les attaques en raison de la complexité du processus de génération de sous-clés. Aucune cryptanalyse efficace n'a pu être effectué sur Blowfish, soulignant son efficacité. Quelques chercheurs ont essayer d'attaquer des algorithme similaire à Blowfish mais l'algorithme de Blowfish n'a jamais subi de réelle attaque contre lui.

5. Annexes

5.1. Définitions

Temps de cryptage : il correspond au temps pris par un algorithme de cryptage pour convertir des données en texte clair en texte chiffré se réfère au temps de cryptage. C'est un indicateur de l'efficacité de l'algorithme. Dans l'analyse suivante, le temps de cryptage est mesuré en millisecondes et est considéré comme un facteur déterminant la vitesse de cryptage dans les réseaux sans fil.

Temps de décryptage : il correspond au temps pris par un algorithme de cryptage pour convertir des données en texte chiffré en données en clair fait référence au temps de décryptage. Moins la vitesse de décryptage est élevée, plus l'algorithme est efficace. Dans l'analyse suivante, le temps de décryptage est mesuré en millisecondes et il est également utilisé pour déterminer la vitesse du réseau sans fil.

Sel : Le salage, est une méthode permettant de renforcer la sécurité des informations qui sont destinées à être hachées (par exemple des mots de passe) en y ajoutant une donnée supplémentaire afin d'empêcher que deux informations identiques conduisent à la même empreinte (la résultante d'une fonction de hachage).

5.2. Sources

- https://fr.wikipedia.org/wiki/Blowfish
- https://fre.small-business-tracker.com/new-collision-attacks-against-triple-des-blowfish-break-https-sessions-984325
- https://fr.wikipedia.org/wiki/OpenBSD
- https://fr.wikipedia.org/wiki/Cryptographie_sym%C3%A9trique
- https://fr.wikipedia.org/wiki/Cryptanalyse#:~:text=La%20cryptanalyse%20est%20la%20technique,particulier%20est%20appel%C3%A9%20une%20attaque.
- http://karbalus.free.fr/sat/docsat/PaperGonzalezTom.pdf