Cryptographie à clé publique

Principes de la cryptographie asymétrique

Afin de pallier ces inconvénients, Diffie, Hellman et Merkle ont eu une proposition révolutionnaire basée sur l'idée suivante : Il n'est pas nécessaire que la clé possédée par la personne qui chiffre le message (c'est Alice dans notre exemple) soit secrète. La partie cruciale est que Bob, le récepteur, ne peut déchiffrer qu'à l'aide d'une clé secrète. Pour réaliser un tel système, Bob publie une clé de chiffrement publique connue de tous. Bob a également une clé secrète correspondante, qui est utilisée pour le déchiffrement. Ainsi,

La clé k de Bob se compose de deux parties, une partie publique, kpub, et une partie privée, kpr.

Une analogie simple d'un tel système est illustrée à la Fig. 6.3. Ce système fonctionne de manière assez similaire à la bonne vieille boîte aux lettres au coin d'une rue : tout le monde peut mettre une lettre dans la boîte, c'est-à-dire chiffrer, mais seule une personne disposant d'une clé privée (secrète) peut récupérer des lettres, c'est-à-dire déchiffrer. Si nous supposons que nous avons des cryptosystèmes avec une telle fonctionnalité,

un protocole de base pour le chiffrement à clé publique ressemble à celui illustré à la Fig. 6.4

En examinant ce protocole, vous pourriez dire que même si nous pouvons chiffrer un message sans canal secret pour l'établissement de la clé, nous ne pouvons toujours pas échanger une clé si nous voulons chiffrer avec, par exemple, AES. Cependant, le protocole peut facilement être modifié pour cette utilisation. Ce que nous devons faire est de chiffrer une clé symétrique, par exemple une clé AES, en utilisant l'algorithme de clé publique. Une fois que la clé symétrique a été déchiffrée par Bob, les deux parties peuvent l'utiliser pour chiffrer et déchiffrer les messages à l'aide de chiffrements symétriques. La figure 6.5 montre un protocole de transport de clé de base où nous utilisons AES comme chiffrement symétrique à des fins d'illustration (bien sûr, on peut utiliser n'importe quel autre algorithme symétrique dans un tel protocole). Le principal avantage du protocole de la Fig. 6.5 par rapport au protocole de la Fig. 6.4 est que la charge utile est chiffrée avec un chiffrement symétrique,

qui a tendance à être beaucoup plus rapide qu'un algorithme asymétrique.

D'après la discussion jusqu'à présent, il semble que la cryptographie asymétrique soit un outil souhaitable pour les applications de sécurité. La question reste de savoir comment construire des algorithmes à clé publique. Dans Chap. 7, 8 et 9, nous introduisons la plupart des schémas asymétriques d'intérêt pratique. Ils sont tous construits à partir d'un principe commun, la fonction à sens unique. La définition informelle en est la suivante :

Fonction unidirectionnelle

Une fonction f() est une fonction à sens unique si :

1. y = f(x) est calculatoirement facile, et

2. x = f −1(y) est calculatoirement irréalisable.

Aspects pratiques de la cryptographie à clé publique

Principaux mécanismes de sécurité des algorithmes à clé publique :

Établissement de clés Il existe des protocoles pour établir des clés secrètes sur un canal non sécurisé. Des exemples de tels protocoles incluent les protocoles d'échange de clés Diffie Hellman (DHKE) ou de transport de clés RSA.

Non-répudiation La non-répudiation et l'intégrité des messages peuvent être réalisées avec des algorithmes de signature numérique, par exemple, RSA, DSA ou ECDSA.

Identification Nous pouvons identifier des entités à l'aide de protocoles de défi et de réponse associés à des signatures numériques, par exemple dans des applications telles que les cartes à puce pour les banques ou les téléphones mobiles.

Chiffrement Nous pouvons chiffrer les messages à l'aide d'algorithmes tels que RSA ou Elgamal.

Nous notons que l'identification et le chiffrement peuvent également être réalisés avec des chiffrements symétriques, mais ils nécessitent généralement beaucoup plus d'efforts avec la gestion des clés. Il semble que les schémas à clé publique puissent fournir toutes les fonctions requises par les protocoles de sécurité modernes. Même si cela est vrai, l'inconvénient majeur dans la pratique est que le chiffrement des données est très gourmand en calculs - ou plus familièrement : extrêmement lent - avec des algorithmes à clé publique. De nombreux chiffrements de blocs et de flux peuvent chiffrer environ cent à mille fois plus rapidement que les algorithmes à clé publique. Ainsi, quelque peu ironiquement, la cryptographie à clé publique est rarement utilisée pour le chiffrement réel des données.

D'autre part, les algorithmes symétriques sont médiocres pour fournir des fonctionnalités de non-répudiation et d'établissement de clés. Afin d'utiliser le meilleur des deux mondes, la plupart des protocoles pratiques sont des protocoles hybrides qui intègrent à la fois des algorithmes symétriques et à clé publique. Les exemples incluent le protocole SSL/TLS qui est couramment utilisé pour les connexions Web sécurisées, ou IPsec, la partie sécurité du protocole de communication Internet.

Le problème restant : l'authenticité des clés publiques

De la discussion jusqu'à présent, nous avons vu qu'un avantage majeur des schémas asymétriques est que nous pouvons distribuer librement des clés publiques, comme indiqué dans les protocoles des Fig. 6.4 et 6.5. Cependant, en pratique, les choses sont un peu plus délicates car il faut encore s'assurer de l'authenticité des clés publiques. En d'autres termes : savons-nous vraiment qu'une certaine clé publique appartient à une certaine personne ? En pratique, ce problème est souvent résolu avec ce qu'on appelle des certificats. En gros, les certificats lient une clé publique à une certaine identité. Il s'agit d'un problème majeur dans de nombreuses applications de sécurité, par exemple,

lors de transactions de commerce électronique sur Internet. Nous abordons ce sujet plus en détail dans la Sect. 13.3.2.

Un autre problème, moins fondamental, est que les algorithmes à clé publique nécessitent des clés très longues, ce qui entraîne des temps d'exécution lents. La question de la longueur des clés et de la sécurité est abordée ci-dessous

Algorithmes de clé publique importants

Dans les chapitres précédents, nous avons découvert certains chiffrements par blocs, DES et AES. Cependant, il existe de nombreux autres algorithmes symétriques. Plusieurs centaines d'algorithmes ont été proposés au fil des ans et même si beaucoup se sont avérés non sécurisés,

il en existe de nombreux cryptographiquement forts comme discuté dans la Sect. 3.7. La situation est assez différente pour les algorithmes asymétriques. Il n'y a que trois grandes familles d'algorithmes à clé publique qui ont une pertinence pratique. Ils peuvent être classés en fonction de leur problème de calcul sous-jacent.

Familles d'algorithmes à clé publique de pertinence pratique Schémas de factorisation d'entiers Plusieurs schémas à clé publique sont basés sur le fait qu'il est difficile de factoriser de grands nombres entiers. Le représentant le plus important de cette famille d'algorithmes est RSA.

Schémas de logarithme discret Il existe plusieurs algorithmes basés sur ce que l'on appelle le problème du logarithme discret dans les corps finis.

Les exemples les plus marquants incluent l'échange de clés DiffieHellman,

Cryptage Elgamal ou algorithme de signature numérique (DSA).

Schémas de courbe elliptique (EC) Une généralisation de l'algorithme de logarithme discret sont les schémas de clé publique de courbe elliptique. Les exemples les plus populaires incluent l'échange de clés DiffieHellman à courbe elliptique (ECDH) et l'algorithme de signature numérique à courbe elliptique (ECDSA).

Longueurs de clé et niveaux de sécurité

Les trois familles d'algorithmes à clé publique établies sont basées sur des fonctions de la théorie des nombres. Une caractéristique distinctive d'entre eux est qu'ils nécessitent une arithmétique avec des opérandes et des clés très longs. Sans surprise, plus les opérandes et les clés sont longs, plus les algorithmes deviennent sécurisés. Pour comparer différents algorithmes, on considère souvent le niveau de sécurité. Analgorithms dit avoir un « niveau de sécurité de n bit » si la meilleure attaque connue nécessite 2n étapes. C'est une définition assez naturelle car les algorithmes symétriques avec un niveau de sécurité de n ont une clé de longueur n bit. Cependant, la relation entre la force cryptographique et la sécurité n'est pas aussi simple dans le cas asymétrique. Le tableau 6.1 indique les longueurs de bits recommandées pour les algorithmes à clé publique pour les quatre niveaux de sécurité 80, 128, 192 et 256 bits. Nous voyons dans le tableau que les schémas de type RSA et les schémas à logarithme discret nécessitent des opérandes et des clés très longs. La longueur de clé des schémas de courbes elliptiques est nettement plus petite, mais toujours deux fois plus longue que les chiffrements symétriques avec la même force cryptographique.

\*\*\*\*\*

Vous voudrez peut-être comparer ce tableau avec celui donné dans la Sect. 1.3.2, qui fournit des informations sur les estimations de sécurité des algorithmes à clé symétrique. Afin d'assurer une sécurité à long terme, c'est-à-dire une sécurité sur plusieurs décennies,

un niveau de sécurité de 128 bits doit être choisi, ce qui nécessite des clés assez longues pour les trois familles d'algorithmes.

Une conséquence indésirable des opérandes longs est que les schémas de clé publique sont extrêmement intensifs du point de vue arithmétique. Comme mentionné précédemment, il n'est pas rare qu'une opération publique, disons une signature numérique, soit de 23 ordres de grandeur plus lente que le cryptage d'un bloc à l'aide d'AES ou de 3DES. De plus, le calcul la complexité des trois familles d'algorithmes augmente approximativement avec la longueur en bits du cube.

A titre d'exemple, augmenter la longueur en bits de 1024 à 3076 dans un logiciel de génération de signature RSA donne une exécution 33 = 27 fois plus lente ! Sur les PC modernes, des temps d'exécution de l'ordre de quelques 10 msec à quelques 100 msec sont courants,

ce qui ne pose pas de problème pour de nombreuses applications. Cependant, les performances de la clé publique peuvent constituer un goulot d'étranglement plus sérieux dans les appareils contraints où les petits processeurs sont répandus, par exemple, les téléphones mobiles ou les cartes à puce, ou sur les serveurs réseau qui doivent calculer de nombreuses opérations de clé publique par seconde. Chaps. 7, 8 et 9 introduisent plusieurs techniques pour implémenter des algorithmes à clé publique de manière raisonnablement efficace.

Théorie essentielle des nombres pour les algorithmes à clé publique

Nous allons maintenant étudier quelques techniques de la théorie des nombres indispensables à la cryptographie à clé publique. Nous introduisons l'algorithme euclidien, la fonction phi d'Euler ainsi que le petit théorème de Fermat et le théorème d'Euler. Tous sont importants pour les algorithmes asymétriques, en particulier pour comprendre le schéma de chiffrement RSA.

Algorithme euclidien

Nous commençons par le problème du calcul du plus grand commun diviseur (pgcd). Le pgcd de deux entiers positifs r0 et r1 est noté pgcd(r0,r1) et est le plus grand nombre positif qui divise à la fois r0 et r1. Par exemple pgcd(21,9)= 3. Pour les petits nombres, le pgcd est facile à calculer en factorisant les deux nombres et en trouvant le facteur commun le plus élevé

Pour les grands nombres qui se produisent dans les schémas à clé publique, cependant, la factorisation n'est souvent pas possible, et un algorithme plus efficace est utilisé pour les calculs pgcd, l'algorithme euclidien. L'algorithme, également appelé algorithme d'Euclide,

repose sur le simple constat que pgcd(r0,r1)= pgcd(r0−r1,r1),

où nous supposons que r0 > r1, et que les deux nombres sont des entiers positifs. Cette propriété se démontre facilement : Soit pgcd(r0,r1)= g. Puisque g divise à la fois r0 et r1,

on peut écrire r0 = g·x et r1 = g·y, où x > y, et x et y sont des entiers premiers entre eux,

c'est-à-dire qu'ils n'ont pas de facteurs communs. De plus, il est facile de montrer que (x−y) et y sont aussi premiers entre eux. Il découle de là que : pgcd(r0−r1,r1)= pgcd(g·(x−y),g·y)= g.

Nous devrions maintenant avoir une idée de l'algorithme d'Euclide, et nous pouvons donner une description plus formelle de l'algorithme.

Euclidean Algorithm

Input: positive integers r0 and r1 with r0 > r1 Output: gcd(r0,r1)

Initialization: i = 1

Algorithm:

1 DO

1.1 i = i+1

1.2 ri = ri−2 mod ri−1

WHILE ri = 0 2 RETURN gcd(r0,r1)= ri−1

Notez que l'algorithme se termine si un reste de valeur ri = 0 est calculé. Le reste calculé à l'itération précédente, noté rl−1, est le pgcd du problème original.

L'algorithme euclidien est très efficace, même avec les nombres très longs généralement utilisés dans la cryptographie à clé publique. Le nombre d'itérations est proche du nombre de chiffres des opérandes d'entrée. Cela signifie, par exemple, que le nombre d'itérations d'un pgcd impliquant des nombres de 1024 bits est 1024 fois une constante. Bien sûr,

des algorithmes de quelques milliers d'itérations peuvent facilement être exécutés sur les PC d'aujourd'hui,

rendant les algorithmes très efficaces dans la pratique.

Algorithme euclidien étendu

Fonction Phi d'Euler

Fonction Phi d'Euler

Le nombre d'entiers dans Zm relativement premiers à m est noté Φ(m).

Nous examinons d'abord quelques exemples et calculons la fonction phi d'Euler en comptant tous les entiers de Zm qui sont relativement premiers.

Exemple 6.8. Soit m = 6. L'ensemble associé est Z6 = {0,1,2,3,4,5}.

pgcd(0,6)= 6 pgcd(1,6)= 1 pgcd(2,6)= 2 pgcd(3,6)= 3 pgcd(4,6)= 2 pgcd(5,6)= 1 Puisqu'il y a deux nombres dans l'ensemble qui sont relativement premiers à 6, à savoir 1 et 5, la fonction phi prend la valeur 2, c'est-à-dire, Φ(6)= 2.

Conclusion

Les algorithmes à clé publique ont des capacités que les chiffrements symétriques n'ont pas, en particulier les fonctions de signature numérique et d'établissement de clé.

Les algorithmes à clé publique sont gourmands en calculs (une belle façon de dire qu'ils sont lents) et sont donc mal adaptés au chiffrement de données en masse.

Seules trois familles de schémas à clé publique sont largement utilisées. C'est considérablement moins que dans le cas des algorithmes symétriques.

L'algorithme euclidien étendu nous permet de calculer rapidement les inverses modulaires,

ce qui est important pour presque tous les schémas à clé publique.

La fonction phi d'Euler nous donne le nombre d'éléments plus petits qu'un entier n qui sont relativement premiers à n. Il s'agit d'une fonction importante pour le schéma de chiffrement RSA.