Introdução à Criptografia: Algoritmos de Chaves Assimétricas

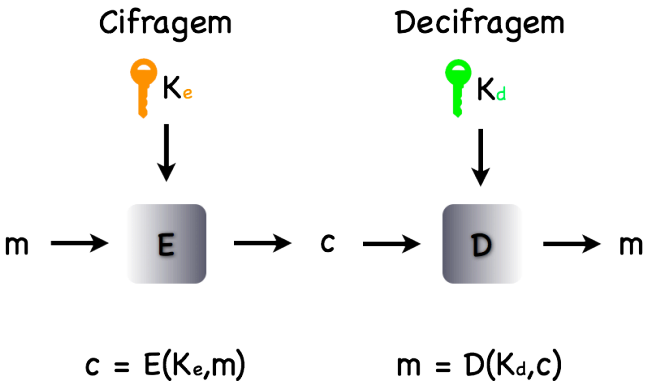
Roteiros e tópicos para estudo por

Vinícius da Silveira Serafim
professor@serafim.eti.br
http://professor.serafim.eti.br

Palavras-chave: algoritmo, chaves assimétricas, assinatura digital

Algoritmos de Chaves Assimétricas: $K_e \neq K_d$

Conforme já estudamos, nos algoritmos de chaves assimétricas, as chave de cifragem e de decifragem são diferentes. É claro que, para que isso seja possível, as chaves estão relacionadas de algum modo. Vamos primeiro estudar essa relação.



- *As chaves são sempre geradas aos pares*: são geradas duas chaves K_a e K_b utilizando-se um algoritmo gerador de chaves específico para o algoritmo de chaves assimétricas em uso. (FERGUSON, 2003) Note que diferenciei as chaves em chave *a* e chave *b* e não mais em chave *e* e *d*. Fiz isso pois tanto K_a quanto K_b pode ser utilizada como chave de cifragem ou de decifragem. (continue lendo...)

- *Não é possível obter uma chave do par a partir da outra*: isso significa que, uma vez gerado o par de chaves, alguém que possua a chave K_b não é capaz de calcular a chave K_a e *vice-versa.* (FERGUSON, 2003)

- *O que for cifrado por uma das chaves do par somente pode ser decifrado com a chave correspondente*: numa forma mais simples, o que for cifrado com a chave K_a somente pode ser decifrado com a chave K_b correspondente e vice-versa. Nenhuma outra chave que não faça parte do par pode ser utilizada para essa decifragem. Sendo assim:

  - $c = E(\mathbf{K_a}, m)$ somente pode ser revertido por $m = D(\mathbf{K_b}, c)$; e

  - $c = E(\mathbf{K_b}, m)$ somente pode ser revertido por $m = D(\mathbf{K_a}, c)$.

Chave pública e Chave privada

Imagine agora que o nosso personagem Bob gere um par de chaves: K_a e K_b. Uma dessas chaves, K_a, Bob guarda somente para si, somente ele possuirá essa chave. A outra, K_b, Bob distribui publicamente (ex.: divulga a chave no seu site).

A chave K_a passa a ser denominada chave **privada (ou secreta)** pois somente Bob a possui (ou conhece).

A chave K_b, é então denominada de chave **pública** pois qualquer um que desejar pode obtê-la.

É devidoa esta última que os algoritmos de chaves assimétricas também são chamados de **algoritmos de chaves públicas**.

A partir deste momento, em nossas aulas, vamos representar a chave privada (secreta) pela letra *S* e a chave pública pela letra *P*. Assim, por exemplo, se Bob gerou um par de chaves, vamos fazer referência às chaves de Bob do seguinte modo:

- S_{bob}: representa a chave privada (secreta) pertencente à Bob; e
- P_{bob}: representa a chave pública pertencente à Bob.

Revendo a relação entre as chaves

Usando as chaves de Bob como exemplo, vamos rever as relações entre as chaves assimétricas. Ao ler os itens abaixo, tenha sempre em mente que *S* representa a chave privada (ou secreta) e que *P* representa a chave pública do par de chaves assimétricas.

- Bob gera um par de chaves (*S*,*P*) usando um algoritmo de geração de chaves, específico para o algoritmo de chaves assimétricas a ser utilizado;
- A partir da chave P_{bob}, que é pública e pode ser obtida por qualquer um, não é possível descobrir a chave S_{bob}, que só Bob deve possuir.
- Tudo o que for cifrado com a chave *P* pertencente à Bob, somente pode ser decifrado pela chave *S* que também pertence à Bob:
  - $c = E(P_{bob},m)$ somente pode ser revertido por $m = D(S_{bob},c)$.
- Tudo o que for cifrado com a chave *S* pertencente à Bob, somente pode ser decifrado pela chave *P* que também pertence à Bob:
  - $c = E(S_{bob},m)$ somente pode ser revertido por $m = D(P_{bob},c)$.

Não siga adiante nesse roteiro até que você tenha entendido claramente o que foi explicado até aqui. Se precisar, procure ajuda!

O que oferecem?

Distribuição de Chaves e Confidencialidade

Você lembra do problema de distribuição de chaves dos algoritmos de chaves simétricas? Se não lembra volte ao roteiro anterior antes de prosseguir.

Com os algoritmos de chaves assimétricas esse problema, **em parte**, está "resolvido". Logo veremos, ao falarmos de gerenciamento de chaves, porque destaquei o "em parte".

Retomemos novamente o exemplo em que Alice deseja enviar para Bob uma determinada mensagem com a garantia de confidencialidade (apenas Bob deve ler a mensagem).

Para que essa comunicação seja possível, com algoritmos de chaves assimétricas, primeiro Bob, que irá receber a mensagem de Alice, deve gerar seu par de chaves e enviar a sua chave pública à Alice.

$$Bob: P_{bob} > Alice$$

Note que, como essa chave pública foi enviada pelo canal inseguro, o qual Eve tem a capacidade de interceptar, Eve também obtém uma cópia desta chave:

$$Eve < P_{bob}$$

Porém não temos problema algum (lembre do "em parte") com esta captura, pois a chave capturada é a chave **pública** de Bob e não a sua chave privada. Alice então cifra a mensagem que deseja enviar à Bob usando a chave pública dele e envia o resultado para Bob:

$$Alice: c = E(\mathbf{P_{bob}}, m)$$

$$Alice: c > Bob$$

Bob recebe a mensagem cifrada enviada pela Alice e Eve, que captura tudo o que é transmitido no canal inseguro, consegue uma cópia. E aqui podemos perceber mais claramente a vantagem dos algoritmos de chaves assimétricas no que diz respeito à distribuição de chaves: como a mensagem foi cifrada com a chave pública de Bob, somente quem possui a chave privada correspondente pode decifrar, ou seja, somente Bob:

$$Bob: m = D(\mathbf{S_{bob}}, c)$$

Eve tem somente a mensagem cifrada e a chave pública de Bob, portanto não consegue decifrar a mensagem por não possuir também a chave privada pertencente ao Bob:

$$Eve: !\ D(\mathbf{S_{bob}}?, c)$$

Alice obteve a garantia de confidencialidade desejada. Já autenticidade e irrefutabilidade não são garantidas pois qualquer um que possua a chave pública de Bob pode enviar a ele uma mensagem cifrada, mesmo Eve pode fazer isso. Bob não tem qualquer garantia sobre a identidade de quem enviou a mensagem.

A garantia de integridade da mensagem é bastante frágil. Modificações acidentais ocorridas na mensagem cifrada serão facilmente detectáveis pois Bob, ao tentar decifrá-las, receberá um erro. No entanto, modificações maliciosas são possíveis e não poderão ser detectadas facilmente. Por exemplo: Eve intercepta a mensagem de Alice, evitando que Bob a receba; Eve cria então uma nova mensagem e a cifra com a chave pública de Bob e então envia essa nova mensagem cifrada à Bob fazendo-se passar pela Alice.

A não ser que a mensagem criada pela Eve não faça sentido algum para Bob, Bob não será capaz de detectar a alteração (substituição) da mensagem da Alice.

Autenticidade, Irrefutabilidade e Integridade

Alice, como no exemplo anterior, deseja enviar uma mensagem à Bob. No entanto, neste novo exemplo, Alice não deseja garantir a confidencialidade da mensagem, mas a autenticidade, irrefutabilidade e integridade da mesma.

Nesta vez, Alice gera seu par de chaves e envia sua chave pública à Bob e, pelas razões que já conhecemos, Eve consegue obter uma cópia desta chave. Assim:

$$\text{Alice: } P_{alice} > \text{Bob}$$

$$\text{Eve} < P_{alice}$$

De novo, essa cópia obtida pela Eve não é um problema, pois é uma cópia de uma chave pública. Agora Alice cifra a mensagem com a sua chave privada:

$$\text{Alice: } s = E(\mathbf{S_{alice}}, m)$$

Esse processo é o que chamamos de assinatura. Quando alguém, neste caso Alice, cifra uma mensagem com uma chave privada (S), dizemos que esse alguém assinou a mensagem. Por essa razão, ao invés da letra *c* que representa a mensagem cifrada, usaremos a letra *s* para representar a mensagem assinada sem garantia alguma de confidencialidade. Vamos adiante e isso deverá ficar mais claro.

Bom, uma vez que a mensagem foi cifrada ou, melhor dizendo, assinada usando a chave privada da Alice, qualquer um de posse da chave pública da Alice pode realizar a decifragem ou, mais corretamente, a verificação da assinatura. Assim, Bob, já de posse da chave pública da Alice, ao receber a mensagem pode realizar a verificação:

$$\text{Bob: } m = D(\mathbf{P_{alice}}, s)$$

Se Bob foi capaz de decifrar corretamente a mensagem *s* com a chave pública da Alice, isso significa que foi a Alice, com a sua chave privada, quem fez a assinatura. Note que Eve pode realizar a mesma operação.

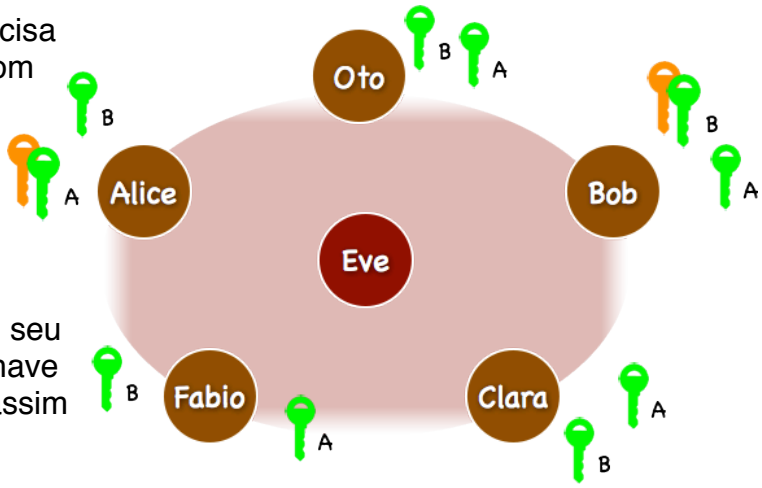
Problemas

Gerenciamento das chaves

É importante que você leia a seção homônima do roteiro sobre algoritmos de chaves simétricas e então volte neste ponto.

O gerenciamento de chaves, no caso dos algoritmos de chaves assimétricas, é facilitado, porém não totalmente resolvido.

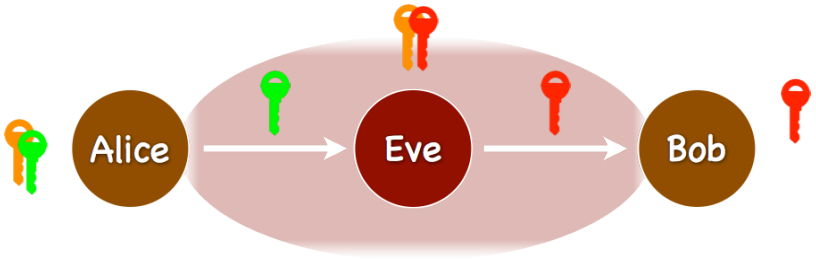
Facilitado pois, agora, Alice não precisa mais combinar uma chave secreta com cada participante, basta que ela gere o seu par de chaves e distribua sua chave pública (chave verde com a letra A) para todos os seus colegas.



O mesmo vale para Bob: ele gera seu par de chaves e distribui sua chave pública para todos os colegas e assim por diante.

No entanto, ainda temos um problema bem sério a resolver. Em todos os nossos exemplos anteriores, nos quais estudamos a garantia de confidencialidade e a assinatura digital, Eve poderia interceptar o envio da chave pública da Alice para Bob e do Bob para Alice e colocar no lugar uma chave pública do par que ela, Eve, possui.

Vejamos um exemplo na figura abaixo:



Alice possui um par de chaves e ela envia sua chave pública ao Bob. Porém Eve intercepta esse envio e, no lugar da chave pública da Alice, envia para Bob a sua própria chave pública. O objetivo é que Bob acredite estar com a chave pública da Alice quando, na verdade, está com a chave pública da Eve. Se isso acontecer, quando Bob cifrar algo com a chave pública que ele acredita ser da Alice, Eve poderá decifrar utilizando sua chave privada e Eve, por sua vez, pode repassar a mensagem à Alice. Ou seja:

$$\text{Bob: } c = E(P_{alice}, m)$$

$$\text{Bob: } c > \text{Alice}$$

Então Eve intercepta a mensagem e a decifra, violando a confidencialidade:

Eve: m = D(S_{alice},c)

Para que Alice e Bob não percebam a interferência da Eve, ela cifra novamente a mensagem, mas agora com a chave pública verdadeira da Alice e envia para Alice passando-se pelo Bob:

Eve: c = E(P_{alice},m)

Eve: (Bob) c > Alice

E Alice decifra normalmente, fazendo: m = D(S_{alice},c).

O mesmo pode ocorrer com relação à assinatura digital. A melhor solução que temos para esse problema, ao menos até o momento, são os certificados digitais. Certificados digitais serão estudados logo adiante em nossas aulas.

Além dessa possibilidade, lembre-se de que Eve pode invadir o computador da Alice e lá trocar as chaves públicas de seus colegas por chaves públicas falsas. Eve pode ainda roubar a chave privada da Alice se esta não estiver adequadamente protegida.

Velocidade

Algoritmos de chaves assimétricas são geralmente 10.000 vezes mais lentos que os algoritmos de chaves simétricas. (PFLEEGER, 2006) Isso torna muito caro, em termos de tempo de processamento, a cifragem e decifragem de mensagens muito grandes (ex.: 10MB).

A solução, no processo de garantia da confidencialidade, é unirmos algoritmos de chaves simétricas e algoritmos de chaves assimétricas. (Vamos ver em seguida)

Já para o processo de assinatura digital - onde há garantia de autenticidade, irrefutabilidade e integridade - temos que unir aos algoritmos de chaves assimétricas às *funções criptográficas de hash*. (Próximo roteiro)

Velocidade no processo de garantia da confidencialidade

Os algoritmos de chaves simétricas são 10.000 vezes mais rápidos que os de chaves assimétricas, porém há o problema de envio da chave K através do meio inseguro.

Já os algoritmos de chaves assimétricas “resolvem” o problema de troca de chaves com a ideia de chaves públicas e privadas, porém são 10.000 vezes mais lentos.

O que ocorre é que é possível unirmos os dois e ficarmos com o melhor de cada um. O processo sem uso de chaves assimétricas seria assim (já vimos isso no roteiro sobre chaves simétricas):

Alice: c = E(K,m)

Alice: c > Bob

$$Bob: m = D(K,c)$$

A cifragem é rápida, porém como Alice envia para Bob a chave K? Vamos resolver isso com as chaves assimétricas. Bob gera um par de chaves e o processo fica:

$$Bob: P_{bob} > Alice$$

Alice continua cifrando a mensagem com uma chave simétrica K por ela escolhida (de preferência aleatoriamente):

$$Alice: c_m = E(K,m)$$

Só que agora Alice tem um meio seguro de enviar a chave K ao Bob, ela cifra a chave K com a chave pública que pertence ao Bob:

$$Alice: c_k = E(P_{bob},K)$$

E envia ao Bob, pelo canal inseguro, tanto a mensagem cifrada c_m quanto a chave cifrada c_k:

$$Alice: c_m, c_k > Bob$$

Ao receber a mensagem cifrada e a chave cifrada, a primeira coisa que Bob deve fazer é recuperar a chave simétrica e, para isso, ele decifra a chave recebida com sua chave privada:

$$Bob: K = D(S_{bob}, c_k)$$

Uma vez decifrada a chave K, ele pode decifrar a mensagem cifrada:

$$Bob: m = D(K, c_m)$$

Perceba que a mensagem *m* é cifrada e decifrada sempre utilizando algoritmos de chaves simétricas, que são muito mais rápidos. Já a chave simétrica K, que é um conjunto sempre pequeno de bits (cerca de 128 a 512), é cifrada com algoritmos de chaves assimétricas e isso minimiza o impacto da baixa velocidade desses algoritmos. Lembre-se de que enquanto uma mensagem pode ter desde alguns kilobytes até alguns gigabytes, a chave simétrica será sempre de alguns poucos bits.

Velocidade na assinatura digital

Antes de ler esta seção, leia o próximo roteiro sobre *funções criptográficas de hash*, depois volte aqui.

Assinatura digital: mais uma vez o problema da velocidade dos algoritmos de chaves assimétricas se apresenta. Pois, como vimos até aqui, para assinar uma mesagem, Alice cifra a mensagem com sua chave privada *S*. E lembremos de que a mensagem pode ser pequena (alguns kilobytes) como pode ser muito grande (alguns gigabytes).

Agora, com as funções criptográficas de hash, temos uma forma de evitar a cifragem da mensagem com a chave privada visando autenticidade, irrefutabilidade e integridade. Vejamos como fica o processo com as funções de hash sendo utilizadas em conjunto com um algoritmo de chaves assimétricas visando assinatura digital.

Alice deseja enviar um documento para Bob apenas assinado digitalmente, sem garantia de confidencialidade. Assim, Alice gera um par de chaves e envia sua chave pública à Bob:

$$\text{Alice: } P_{alice} > \text{Bob}$$

Alice então calcula o hash da mensagem:

$$\text{Alice: } h = H(m)$$

Pelas características que já conhecemos das funções de hash, podemos afirmar que o valor *h* representa de forma única e inequívoca a mensagem m. O que Alice faz então é assinar o hash *h* da mensagem, que representa a mensagem e é pequeno.

$$\text{Alice: } s = E(S_{alice},h)$$

Agora Alice envia para Bob a mensagem *m* e o hash assinado *s*.

$$\text{Alice: } m,s > \text{Bob}$$

Bob, ao receber a mensagem e o hash assinado, primeiro recupera o hash gerado pela Alice utilizando, é claro, a chave pública da Alice:

$$\text{Bob: } h = D(P_{alice},s)$$

Se a decifragem ocorreu corretamente com a chave pública da Alice, significa que foi a Alice que, com sua chave privada, assinou o valor de hash resultante. Aqui temos a autenticidade e irrefutabilidade: foi Alice quem enviou o hash *h*.

Em seguida, Bob calcula novamente o hash da mensagem recebida e compara com o hash enviado pela Alice:

$$\text{Bob: } h' = H(m)$$

$$\text{Bob: } i == (h == h')$$

Se os hashes forem iguais a integridade não foi violada e a mensagem recebida é a mensagem que foi assinada pela Alice. Bob sabe que foi Alice quem enviou a mensagem, sabe que a mensagem não foi alterada e Alice não pode negar a autoria da mensagem.

Notem que em nossos passos a mensagem não mais é cifrada ou decifrada utilizando-se um algoritmo de chaves assimétricas. Apenas o hash sobre essas operações e, como ele é sempre pequeno, o impacto da baixa velocidade dos algoritmos de chaves assimétricas é minimizado.

O processo completo

Para entender o que está aqui descrito você tem que ter compreendido muito bem o que foi estudado anteriormente. Havendo dúvidas, volte no roteiro. Persistindo as dúvidas, me procure!

Alice deseja enviar uma mensagem para Bob garantindo:

- Confidencialidade;
- Autenticidade;
- Irrefutabilidade; e
- Integridade.

1. Troca de chaves: Alice envia sua chave pública para Bob e vice-versa:

$$\text{Alice: } P_{alice} > \text{Bob}$$
$$\text{Bob: } P_{bob} > \text{Alice}$$

2. Alice assina: sempre primeiro a assinatura e depois a cifragem para garantia da confidencialidade. Lembre-se: ninguém deve assinar algo que não pode ler.

$$\text{Alice: } h = H(m)$$

$$\text{Alice: } s = E(S_{alice},h)$$

3. Alice cifra e envia ao Bob a mensagem cifrada, a chave simétrica cifrada e o hash assinado:

$$\text{Alice: } c_m = E(K,m)$$

$$\text{Alice: } c_k = E(P_{bob},K)$$

$$\text{Alice: } c_m, c_k, s > \text{Bob}$$

4. Bob recebe os dados e realiza a decifragem:

$$\text{Bob: } K = D(S_{bob},c_k)$$

$$\text{Bob: } m = D(K,c_m)$$

5. Agora Bob pode verificar a assinatura:

$$\text{Bob: } h' = H(m)$$

$$\text{Bob: } i == (h == h')$$

Algoritmos atualmente em uso

O algoritmo de chaves assimétricas mais popular atualmente é o RSA (Rivest-Shamir-Adelman) de 1978. (PFLEEGER, 2006)

Existem alternativas como o El Gamal e mesmo os algoritmos de curvas elípticas.

Considerações sobre o tamanho da chave

Enquanto o tamanho seguro para os algoritmos de chaves simétricas é a partir de 128bits, sendo 256bits o recomendado, para os algoritmos de chaves assimétricas o tamanho mínimo recomendado é 1024bits, alguns autores já indicam 2048bits como sendo o tamanho mínimo recomendado. (FERGUSON, 2003)

Bibliografia

FERGUSON, N. F.; SCHNEIER, B. Practical Criptography. 1a edição. Wiley, 17 de abril de 2003. 432 pág.

PFLEEGER, C. P. Security in Computing. 4a edição. Prentice Hall, 23 de outubro de 2006. 880 pág.

SCHNEIER, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2a edição. Wiley, 18 de outubro de 1996. 758 pág.