Protocolo de Acordo de Chaves de Diffie-Hellman

Gustavo Zambonin

Segurança em Computação (UFSC-INE5429)

- Originalmente publicado em 1975, o *protocolo de acordo de chaves Diffie-Hellman* (DH) [1] habilita duas entidades a estabelecerem uma *chave secreta* mesmo que estas não tenham se comunicado previamente, com o benefício desta troca de dados poder ser monitorada sem que tal chave seja descoberta. Esta pode ser utilizada, possivelmente, em um algoritmo de cifragem simétrico, de modo a habilitar a troca de mensagens criptografadas entre as entidades. Uma distinção importante é que *nenhuma* informação é trocada além das inerentes ao processo de criação de chave — assim sendo, o protocolo DH *não* é classificado como um algoritmo de criptografia assimétrica.

  Uma descrição matemática pode ser resumida da seguinte maneira: sejam Alice e Bob as entidades em questão; um número primo g e uma de suas raízes primitivas p, ambos suficientemente grandes (e geralmente publicamente padronizados [2]) são escolhidos por Alice e Bob. Cada uma das entidades gera um número secreto para si, chamados de X_A e X_B; Alice calcula $A = g^{X_A} \pmod p$ e envia para Bob; do mesmo modo, $B = g^{X_B} \pmod p$ é enviado para Alice por Bob. Assim, as únicas trocas propostas são efetuadas e as entidades já compartilham uma chave secreta.

  Esta reside no cálculo de $B^{X_A} \pmod p$ por Alice e $A^{X_B} \pmod p$ por Bob. Expandindo os números recebidos, temos $(g^{X_A} \pmod p)^{X_B} \pmod p$ e $(g^{X_B} \pmod p)^{X_A} \pmod p$ respectivamente; é possível reduzir estes números, através de aritmética modular, para $g^{X_A X_B} \pmod p$ e $g^{X_B X_A} \pmod p$; nota-se a igualdade dos termos, em virtude da comutatividade da operação de multiplicação sob os números reais. Assim, Alice nunca soube o número secreto de Bob e vice-versa, e a chave secreta nunca foi transmitida, porém é conhecida por ambos.

  $$\begin{gathered} Alice \xrightarrow{A} Bob \\ Alice \xleftarrow{B} Bob \end{gathered}$$

  Utilizando um exemplo numérico, supõe-se que Alice e Bob concordam em usar o número primo $p = 1949$ e sua raiz primitiva $g = 1475$. Seus números secretos são, respectivamente, $X_A = 128$ e $X_B = 64$. O segredo compartilhado é chamado de s.

  $$\begin{aligned} A &= g^{X_A} \pmod p = 1475^{128} \pmod{1979} = 448 \\ B &= g^{X_B} \pmod p = 1475^{64} \pmod{1979} = 872 \\ s &= B^{X_A} \pmod p = 872^{128} \pmod{1979} = A^{X_B} \pmod p = 448^{64} \pmod{1979} = 560 \end{aligned}$$

  Estas computações podem ser realizadas rapidamente com um interpretador Python, pois a linguagem implementa exponenciação modular[1].

```
>>> pow(1475, 128, 1979)
448
>>> pow(1475, 64, 1979)
872
>>> pow(448, 64, 1979)
560
>>> pow(872, 128, 1979) == pow(448, 64, 1979)
True
```

- O programa que implementa DH está localizado em `diffie_hellman.py`, e o acordo de chaves automatizado, em `key_exchange.py`. Este programa apresentará falha (e saída) *apenas* se a chave secreta não for igual para ambas as entidades, consequência de algum processo matemático anômalo. É importante notar que os números e suas raízes primitivas foram computados utilizando algoritmos implementados pelo autor em trabalhos passados, porém os segredos de cada entidade foram computados utilizando o gerador de números aleatórios da linguagem, similar ao implementado.

gustavo.zambonin@grad.ufsc.br src/

[1] operação do tipo $d = a^b \pmod c$ onde a exponenciação, um passo extremamente custoso se a e b forem números relativamente grandes, não precisa ser calculada. O resultado d é a inversa multiplicativa de $a \pmod c$.

O tamanho de ≈ 80 bits obtido para os números primos aleatórios acontece pois a extração de raízes primitivas é um processo custoso; depois de otimizações no cálculo da função totiente de Euler e da utilização de uma implementação alternativa da linguagem Python[2], percebeu-se que uma abordagem diferente seria necessária para que tal processo fosse acelerado, como uma mudança de linguagem de programação, e assim todo o ecossistema já construído para o trabalho precisaria ser refeito. Portanto, a prova de conceito é demonstrada com números de tamanho reduzido.

- Um ataque do tipo *man-in-the-middle* existe quando uma entidade monitora secretamente a comunicação entre duas outras entidades, podendo alterar as mensagens entre elas, simulando uma impersonificação dupla simultânea; como o protocolo DH original não contém qualquer tipo de autenticação dos usuários, ele torna-se vulnerável a este tipo de ataque. Caso o atacante, chamado de Eve, consiga obter as chaves parciais A e B de Alice e Bob, então é possível que ele crie uma chave secreta com cada um deles e simule a troca de mensagens direta caso o canal não seja seguro o suficiente. Sejam as entidades Alice, Bob e Eve, seus números secretos $X_A = 128$, $X_B = 64$ e $X_E = 32$, $p = 1949$ e $g = 1475$. O segredo compartilhado entre Alice e Eve é chamado de s_{ae}, e entre Eve e Bob, s_{eb}.

$$
\begin{aligned}
A &= g^{X_A} \pmod p = 1475^{128} \pmod{1979} = 448 \\
E &= g^{X_E} \pmod p = 1475^{32} \pmod{1979} = 542 \\
B &= g^{X_B} \pmod p = 1475^{64} \pmod{1979} = 872 \\
s_{ae} &= E^{X_A} \pmod p = 542^{128} \pmod{1979} = A^{X_E} \pmod p = 448^{32} \pmod{1979} = 560 \\
s_{eb} &= E^{X_B} \pmod p = 542^{64} \pmod{1979} = B^{X_E} \pmod p = 872^{32} \pmod{1979} = 322
\end{aligned}
$$

Um diagrama segue abaixo, mostrando uma mensagem M e uma mensagem alterada por Eve M' a partir do ataque descrito (decodificada com s_{ae} e codificada com s_{eb}).

$$
\begin{gathered}
Alice \xrightarrow{A} Eve \xrightarrow{A} Bob \\
Alice \xleftarrow{E} Eve \xleftarrow{B} Bob \\
Alice \xrightarrow{M} Eve \xrightarrow{M'} Bob
\end{gathered}
$$

- Se g não for uma raiz primitiva módulo p, então g gerará apenas um subgrupo do grupo multiplicativo $\mathbb{Z}/p\mathbb{Z}$, e assim a segurança do protocolo será proporcional à *ordem* de g em $\mathbb{Z}/p\mathbb{Z}$, onde o ideal seria a ordem total do grupo. Para que esse aspecto seja contornado, um número suficientemente grande é escolhido para que a ordem de seu subgrupo seja consequentemente afetada, e assim, um nível de segurança plausível seja obtido.

Referências

[1] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 2006.

[2] M. Lepinski and S. Kent. Additional Diffie-Hellman Groups for Use with IETF Standards. RFC 5114 (Informational), January 2008.

[2] PyPy — o principal recurso apresentado é a utilização de um compilador JIT (*just-in-time*).