Descoberta da Chave Privada do Algoritmo DES por Força Bruta com Processamento Paralelo e Distribuído.[1]

Isaac Roque Sartori Junior

Universidade do Vale do Rio dos Sinos – UNISINOS
São Leopoldo – RS - Brasil
isaac@redesul.com.br

Introdução

Ainda hoje, é utilizado o DES (Data Encryption Standard – Padrão de encriptação de dados) como algoritmos de criptografia simétrica tanto na Internet quanto em troca de informações. Por ser um algoritmo de criptografia simétrica, o DES utiliza a mesma chave para cifrar e decifrar um dado. Por isso se faz necessário manter esta chave secreta, isto é, somente o emissor e o destinatário devem conhecer esta chave.

Este artigo descreve uma maneira de modelar uma solução que tente descobrir, por força bruta, a chave privada a partir de uma mensagem e um conjunto de possíveis chaves para verificar a eficácia do algoritmo DES. O modelo de implementação desta solução é Cliente/Servidor, onde existe um servidor que mantém os dados necessários para fazer a decifragem e os gerencia, e vários clientes que fazem o trabalho pesado das decifragens.

Na seção de "Caracterização do Problema" é discutida a garantia de confidencialidade deste algoritmo, visto que ele possui chaves de apenas 56 bits e estas podem não ser muito grandes com o poder computacional existente hoje. Nesta seção, também é mostrado que este problema é trivialmente paralelo, pois os dados são independentes. Na seção "Modelo da Solução" é sugerida uma forma de implementar um aplicativo paralelo e distribuído que tenta decifrar, por força bruta, uma mensagem cifrada com várias chaves privadas. Em "Tecnologia Utilizada" cita-se o framework dotNET da Microsoft como ferramenta utilizada para esta solução. Além disto, é justificado o seu uso nesta implementação. A seção Resultado, explica que resultados serão obtidos com este experimento, bem como os seus significados.

Caracterização do Problema

O DES utiliza uma chave de 56 bits. Por esta chave não ser grande, torna-se possível pensar em utilizar algum algoritmo de força bruta para tentar descobrir esta chave utilizando um conjunto de chaves, e com isso decifrar o dado. Utilizando processamento concorrente o tempo para se descobrir uma chave pode ser reduzido, pois o problema é trivialmente paralelo e, por esta razão, algumas possibilidades de chaves podem ser testadas ao mesmo tempo. A figura 1 ilustra muito bem este paralelismo.

O problema está em criar um aplicativo capaz de gerenciar vários agentes (clientes) tentando descobrir o segredo que é a chave privada com processamento distribuído. Este estudo se faz válido, até mesmo, para verificarmos se esta forma de criptografia é segura ou não com os recursos computacionais existentes hoje.

[1] Trabalho Prático para a disciplina "Programação Paralela e Distribuída" orientada pelo professor Gerson Cavalheiro.

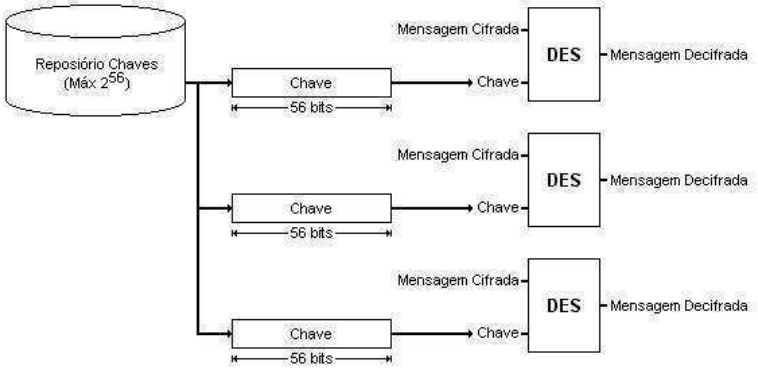

Figura 1: Problema trivialmente paralelo

Modelo da Solução

A modelagem da aplicação para descobrir uma chave privada por força bruta, prevê a utilização da programação Cliente/Servidor.

Basicamente um modelo de solução válida se compõe de um servidor, um ou mais clientes, um repositório de possíveis chaves para serem testadas e uma mensagem cifrada a ser decifrada. A figura 2 ilustra esta estrutura.

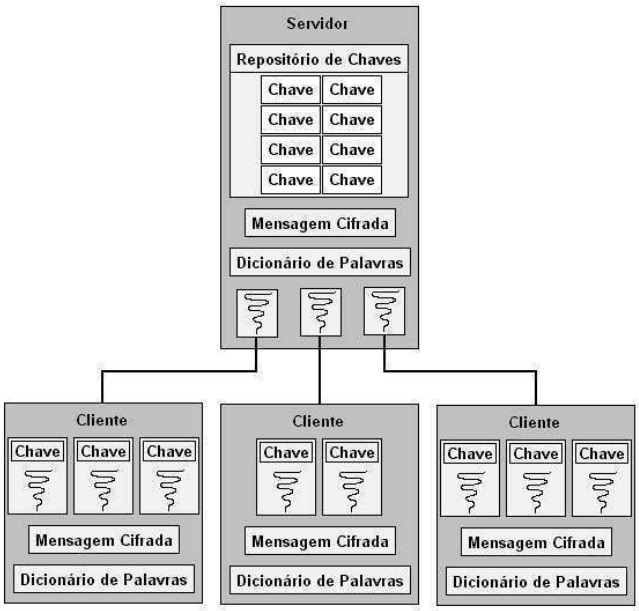

Figura 2: Estrutura do aplicativo Cliente/Servidor

No servidor, um fluxo de execução independente será responsável por gerenciar o repositório de chaves a serem testadas. O servidor deverá criar também, um fluxo de execução para cada cliente que solicitar conexão. Para cada fluxo de execução do servidor para um cliente, outro fluxo de execução responsável pelo gerenciamento das chaves deve separar um conjunto de possíveis chaves que cada cliente deverá utilizar para decifrar a mensagem cifrada. Então, quando um cliente se conectar ao servidor, este deve mandar, quando solicitado, uma mensagem com um conjunto de chaves, a mensagem a ser decifrada, e um dicionário de possíveis palavras da mensagem.

No lado do cliente deve-se solicitar ao servidor um conjunto de chaves, a mensagem cifrada para tentar decifrá-la, e um dicionário de possíveis palavras da mensagem. Neste momento o servidor solicita

ao gerenciador de chaves um conjunto de chaves e as manda para o cliente juntamente com a mensagem e o dicionário. Então o cliente, de posse a este conjunto de chaves cria um novo fluxo de execução para cada chave, implementando assim o paralelismo. Cada fluxo de execução tenta decifrar a mensagem com uma chave e uma cópia da mensagem. Decifrada a mensagem, este fluxo de execução verifica se a mensagem contém alguma palavra do dicionário, afirmando se esta chave é valida ou não. Assim que uma chave for considerada válida, o cliente manda uma mensagem para o servidor informando a chave válida. Neste caso, o servidor não permitirá que ninguém mais se conecte solicitando novos conjuntos de chaves. Caso o cliente teste todas as chaves, e não encontrar nenhuma válida, este solicita para o servidor um novo conjunto de chaves para mais uma rodada de tentativas. Se o servidor já possuir a chave válida, este manda uma mensagem para o cliente notificando que não precisa mais dos serviços dos clientes.

Nota-se que para esta implementação, será utilizado o conceito de memória compartilhada no servidor e nos clientes para o acesso aos conjuntos de chaves (controle de sessões críticas em processamento paralelo), e o conceito de processamento distribuído, pois deverá existir troca de mensagens entre o servidor e os clientes para transferência de dados e estados dos mesmos.

Tecnologia Utilizada

A tecnologia utilizada neste experimento, é o Framework dotNET da Microsoft, pois além de ser gratuita, ela oferece recursos suficientes para este experimento. Os principais recursos implementados que serão utilizados neste experimento são os sockets sobre TCP/IP (para programação distribuída), as Threads (para a programação paralela) bem como Monitores (componente que controlam o acesso a sessões críticas entre várias threads), o próprio algoritmo DES para a decifragem da mensagem, entre outros recursos simples, como coleções de objetos, forms para criação de janelas, etc.. Uma característica interessante, que pode ser explorada, é a portabilidade do framework, pois ele está sendo implementado para rodar em vários sistemas operacionais. Além da portabilidade, esta tecnologia possui alto grau de conectividade, pois é possível ter um servidor rodando em um PC, e ter vários clientes em PCs, Computadores, dispositivos de bolso, e outros dispositivos que suportem o framework dotNET conversando entre si. Outro fato importante, é a vasta documentação que acompanha o framework e o fato da Microsoft estar apostando muito nesta tecnologia.

Resultados

No servidor, o número de chaves testadas sobre o número de chaves do Repositório de Chaves demonstra a dificuldade de decifrar a mensagem, quando isto ocorrer. Neste caso, este número pertence ao intervalo entre 0 e 1, sendo que quanto maior este valor, maior é a dificuldade de conseguir descobrir a chave privada.

Um cronômetro será disparado no servidor e nos clientes quando estes iniciarem seus processamentos. Este cronômetro servirá para uma análise do Somatório dos tempos dos clientes sobre o tempo de execução do servidor. Esta razão mostra o ganho que se obtém com o processamento paralelo e distribuído. Quanto maior este valor, mais concorrente foi o processamento desta solução, pois o tempo total da aplicação (medida no servidor) foi menor.

Além dos dados citados acima, outros dados serão obtidos, como por exemplo, o número de conexões que foram criadas e a própria chave privada.

Conclusão

A programação paralela e distribuída deve ser utilizada na descoberta de uma chave privada do algoritmo de criptografia DES, pois este problema é trivialmente paralelo. Por ter esta característica, o uso deste modelo de programação (paralela e distribuída) torna viável esta tentativa de processamento por força bruta, pois se as tentativas forem executadas de forma seqüencial, o tempo necessário de processamento seria extremamente grande (como a chave contém 56 bits, então existem 2^{56} chaves diferentes para serem testadas).

O sucesso ou não de se encontrar a chave privada, depende do conjunto de chaves que deverão ser testadas, e das palavras contidas no dicionário. Quanto maior for o número de chaves que deverão ser testadas, provavelmente maior será o tempo necessário de processamento, e maior será a probabilidade de se obter sucesso.

Com os dados que serão obtidos neste trabalho, será possível verificarmos, o quão difícil é descobrir uma chave privada, e o ganho de se utilizar o processamento concorrente em relação ao seqüencial. A dificuldade de se descobrir a chave primária se consegue com o número de chaves que foram testadas. E o ganho é obtido pela razão entre o tempo no qual o servidor está funcionando com o somatório de todos os tempos de cada cliente. Dependendo do ganho e da dificuldade encontrada, é possível verificar se o DES continua sendo uma boa solução para a cifragem de mensagens com os recursos computacionais disponíveis hoje.

Bibliografia

[NET 02] **Net Framework SDK Documentation** (ajuda do frameword .net)

[DEI 03] DEITEL, H. M.;DEITEL, P. J. **C# Como Programar**. São Paulo: Editora Makron Books, 2003. 1ª Edição. (Cap. 21 – ASP.NET e Serviços da Web; Cap. 22 – Rede: Soquetes e Datagramas Baseados em Fluxos).

[WAT 02] WATSON, KARLI. **Beginning C# Programando**. São Paulo: Editora Makron Books, 2002. 1ª Edição. (Cap. 25 – Web Services).

[SMI 97] SMITH, RICHARD E. **Internet Cryptography**. Berkeley. Addison Wesley, 1997.