Instituto Tecnológico de Aeronáutica
Curso de Engenharia da Computação
Disciplina de CTC-21

Sistema criptográfico em RSA

Daniel Prince Carneiro
Dylan N. Sugimoto
Gabriel Adriano de Melo
Laurival Siqueira Calçada Neto
Thiago Filipe de Medeiros

Docente: Professor Carlos Ribeiro

São José dos Campos
11/07/2017

Designação de responsabilidades

Daniel:

Dylan: Pesquisa e compreensão do método RSA. Pesquisa e compreensão da correlação entre o método RSA e a matemática discreta. Redação da motivação e da metologia do relatório que propôs o tema desse projeto. Redação do esqueleto do relatório. Redação da capa, motivação, metologia e recursos utilizados.

Gabriel:

Laurival:

Thiago: Implementação do algoritmo RSA na linguagem Python. Comentários do código explicando e exemplificando o funcionamento do mesmo. Testes e análise do código implementado em relação ao espaço utilizado e tempo consumido durante execução.

_____
Daniel Prince Carneiro

_____
Dylan Nakandakari Sugimoto

_____
Laurival Siqueira Calçada Neto

_____
Gabriel Adriano de Melo

_____
Thiago Filipe de Medeiros

Motivação

Nesta Era da Informação, a humanidade está cada vez mais dependente da internet, pois a facilidade que essa tecnologia trouxe para o compartilhamento de informação mudou a forma como os indivíduos se relacionam consigo mesmo e com os outros ao seu redor. Contudo, um problema desse compartilhamento de informação é a segurança dos dados compartilhados, no sentido de evitar que agentes invasores sequestrem esses dados ou informações compartilhadas. Afinal, como praticamente todos os setores da economia e da política utilizam computadores para armazenar e compartilhar informações, torna-se evidente que haja informações sensíveis ou confidenciais armazenas na forma de bits; logo, esses computadores podem sofrer ataques cibernéticos, como ocorreu recentemente, no mês de Maio e Junho de 2017, em que "*hackers*" sequestraram informações de computadores de empresas e pediram um valor de resgate dessas informações para essas empresas.

Assim, nesse contexto da segurança da informação, destaca-se a criptografia RSA por possibilitar a comunicação segura de informações e, pelo menos até agora, ter resistido a todas as tentativas de quebra. Além disso, é o principal método de criptografia utilizado na segurança de dados compartilhados via internet, e por tudo isso foi escolhido como tema para este trabalho aplicar a criptografia RSA em textos em português, bem como pelo fato desse método de criptografia utilizar conhecimento embasado na Teoria dos Números, que está inserida no contexto da matemática discreta, ou seja, este tema está correlaciona com o conteúdo ministrado na disciplina CTC-21. De forma mais específica, foi utilizado conceitos da aritmética módulo n para aplicação do método RSA[1][3], porém é sabido que a criptografia faz uso de teorias mais avançadas da matemática discreta, como por exemplo, um problema da computacional da criptografia é encontrar vetores curtos em reticulados modulares, em que nesse caso faz uso de uma teoria de reticulados.[4][5]

Metologia

Este projeto foi executado em três etapas:

1. Busca por informações no sentido de compreender o funcionamento do algoritmo do método de criptografia RSA e a sua correlação com a disciplina de CTC-21.
2. Implementação do método em código utilizando as linguagens recomendadas, em especial, a linguagem python. E documentação do código.
3. Análise do código e redação do relatório.

Na primeira etapa, percebeu-se que há uma grande quantidade de materiais sobre a metologia RSA, e de diferentes formatos, ou seja, há diferentes fontes sobre essa metologia disponíveis na internet que variam desde web enciclopédias e outros sites da área de criptografia e matemática discreta até monografias e artigos científicos, e videoaulas sobre a metologia RSA. Diante dessa grande variedade de fontes, deu-se prioridade para as fontes com referências confiáveis, ou para as documentações advindas de instituições de educação e de pesquisa de conceito e de alto prestígio. Assim, consultando essas fontes compreendeu-se o algoritmo de geração, de encriptação e de decriptamento, e que tal método fundamenta-se na matemática módulo n.[1][3] E com uma pesquisa mais aprofundada, no sentido, de investigar mais a fundo a correlação entre a matemática discreta e a criptografia, descobriu-se a aplicação da teoria de reticulados em criptografia, em específico, na resolução do problema do vetor mais curto.[4][5]

De forma resumida, o método RSA é composto por geração das chaves, encriptação da mensagem e decriptamento da mensagem. Na geração das chaves, primeiramente escolhe-se dois números primos e realiza-se o produto deles, obtendo-se o primeiro número da chave pública e da chave privada. O segundo número da chave pública é escolhido de forma que seja primo com a função toliente do primeiro número, e seja menor que a função toliente do primeiro número. O segundo número da chave privada é tal que o produto entre o segundo número das duas chaves deixa resto unitário na divisão pela função toliente do primeiro número da chave pública. Na encriptação, cada caractere é transformado para um número de dois dígitos, e o texto é quebrado em blocos, chamados de mensagem. O número criptografado na mensagem é tal que deixa o mesmo resto que a mensagem elevada à potência do segundo número da chave pública na divisão pelo primeiro número da chave pública. No decriptamento, recupera-se a mensagem ao calcular o número que deixa o mesmo que a mensagem criptografa elevada à potência do segundo número da chave privada na divisão pelo primeiro número da chave privada. Essa lógica matemática foi implementada computacionalmente na etapa seguinte.

Na segunda etapa, implementou-se o método RSA em linguagem Python começando pela geração das chaves, depois, a encriptação de texto, e finalmente, o decriptamento, sendo que a documentação do código foi feita à medida que parte era completada. Após o término de cada parte do código (geração, encriptação e decriptamento), testes foram realizados para assegurar a funcionalidade do código. E ao fim das três partes estarem completas, iniciou-se os testes finais para assegurar o funcionamento global do código sendo seguidos pelos testes de análise de desempenho. Os resultados desses teste são descritos na secção "Resultados e Análise" deste relatório.

Na terceira etapa, foi feito uma análise do código, e melhorias foram sugestionadas e implementadas. E o projeto foi documentado na forma do presente relatório.

Implementação:

Deve-se utilizar dois primos maiores com mais de 100 dígitos decimais para que o código possa funcionar, apesar de que também foi condi ficada uma função que gera primos aleatórios, mas que não é utilizada em toda a execução por ser lenta. Gerou-se primos com o programa OpenSSL

O código foi dividido em três arquivos de forma a organizá-lo melhor. No arquivo "RSA.py", mostra-se um exemplo de criptografia e descriptografia de uma mensagem de texto presento no arquivo "mensagem.txt". No arquivo "classes.py" utilizou-se orientação a objetos para encapsular o código que definem o mensageiro e o destinatário da mensagem. No arquivo "suporte.py" definiram-se funções matemáticas básicas e computacionais auxiliares, como o máximo divisor comum entre dois números ou ainda o inverso modular.

No arquivo "RSA.py", as chaves públicas e privadas são geradas a partir dos números primos, isto é, primeiro faz-se a multiplicação de ambos os primos para obter a chave pública e o seu expoente é obtido como o coprimo do produto dos dois primos menos um. O expoente da chave privada é obtido como o inverso modular do expoente da chave pública. Em seguida, a mensagem é lida de um arquivo e criptografada utilizando-se a chave pública, sendo a mensagem criptografada impressa na tela. Por fim, a mensagem é descriptografada utilizando-se a chave privada, sendo então impressa na tela.

É importante notar que como a criptografia ocorre em texto, é necessário que o mesmo seja convertido para uma forma numérica. Para melhorar a segurança da implementação, cada grupo de 78 caracteres são convertidos para um único número, este número pode ser aumentado caso se escolham números primos ainda maiores. É importante, contudo, que a codificação da mensagem original esteja no formato UTF-8, pois cada caractere poderá ser codificado com até 1000 códigos

diferentes, na implementação utilizada em python, o que cobre todos os acentos e caracteres especiais utilizados em português. Esse passo de converte o texto em número é realizado pela função “conversor” presente na classe “mensageiro”.

Após ter a mensagem de texto expressa como um vetor de números, cada número é então criptografado utilizando-se a chave pública, isto é, é realizada uma exponenciação modular desse número na base da chave pública.

Assim, a mensagem criptografada, que corresponde a esse vetor de números criptografados pode ser transmitida com segurança para o destinatário, que ao recebê-la, utilizará a sua chave privada para descriptografar cada número desse vetor de números. Assim, com um vetor de números descriptografados, basta converter cada número em uma sequência de 78 caracteres, que poderá ser lido no formato UTF-8.

Recursos e Avaliação

Neste projeto, foi utilizado as seguintes ferramentas de pesquisa e codificação:

- Pesquisa na internet;
- Consulta a materiais acadêmicos relacionados ao assunto (artigos científicos, monografia, livros, etc);
- Codificação em Python;
- Código redigido utilizando o programa “Sublime Text”.
- Redação do relatório no programa “LibreOffice Writer”;

Resultados e Análise

Conclusão

Referência Bibliográfica

[1] de Oliveira, P. E. R.; de Andrade, P. T. E.; D'Oliveira, R. L. G. *Rsa*. Disponível em: http://www.ime.unicamp.br/~ftorres/ENSINO/MONOGRAFIAS/oliv_RSA.pdf, acessado em: 11 julho. 2017.
[2] Disponível em: https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_totiente_de_Euler, acessado em: 11 julho. 2017.
[3] Disponível em: https://pt.wikipedia.org/wiki/RSA, acessado em: 20 maio. 2017.
[4] Zanon, G. H. M.. *Criptografia Homomórfica*. São Paulo: Novembro de 2016. Disponível em: http://bcc.ime.usp.br/tccs/2016/gustavozanon/monografia.pdf, acessado em: 20 maio. 2017.
[5] Bollauf, M. F. *Criptografia baseada em Reticulados*. Campinas: 21 de Novembro de 2014. Disponível em: http://www.ime.unicamp.br/~campello/geometria/maiara.pdf, acessado em: 20 maio. 2017.

Apêndice