Criptografia Assimétrica com RSA

Dados obtidos através
do estudo da história das ciências da criptografia e
criptoanálise e sua implementação através da linguagem C#.

Resumo

Procuramos através deste trabalho descrever a história da criptografia e do seu desenvolvimento, impulsionado pela criptoanálise.

Fizemos referência a estenografia, que não era uma cifra propriamente dita mas, foi através dela que se tudo começou.

Os tipos de criptografia são divididos em 2 grupos: o grupo da criptografia simétrica e o grupo da criptografia assimétrica.

Durante mais de 2.000 anos o mundo só tinha conhecimento da criptografia simétrica, onde o emissor e o receptor devem possuir a mesma chave. A primeira criptografia simétrica foi uma cifra de César, que era uma simples transposição de letras do alfabeto. Conforme a criptoanálise ia quebrando as cifras existentes, os criptógrafos foram obrigados a desenvolver cifras mais seguras. Nesta história, passamos pela evolução das cifras monoalfabéticas até as polialfabéticas, onde desta, a mais famosa ficou conhecida como cifra de Vigènere. Esta cifra foi a base para criação da máquina Enigma, a mais terrível arma da criptografia utilizada a serviço da guerra.

Nos anos 60 começou-se a desenvolver a ideia da criptografia assimétrica. Não foi uma descoberta de uma só pessoa ou de um só momento das histórias. Levou-se alguns anos para desenvolver a prova teórica desta até sua aplicação prática, conhecida como chave RSA, praticamente impossível de ser quebrada através da tecnologia dos computadores atuais.

Ainda no século 20, desenvolveu-se a ideia do computador quântico, e, tamanha a capacidade deste teórico computador que, poderia quebrar uma informação em criptografia assimétrica em poucos minutos.

Na carona da ideia do computador quântico, ainda teórico, surgiu uma ideia mais teórica ainda, que é a criptografia quântica, esta sim, baseada em elementos quânticos como armazenamento de fótons, afim de transcrever uma mensagem.

Por fim, apresentamos um trabalho prático que é baseado na implementação da criptografia RSA, através da linguagem C#.

ABSTRACT

We intend, through this research, to tell something about encryption history and its development, based on cryptanalysis.

We've referenced stenography, which was not a cipher itself, but it has started the encryption concept.

Encryption types are divided in 2 groups: symmetric-key encryption and public-key encryption.

For over 2000 years, world only knows the symmetric-key type, where the sender and the receiver must have the same key.

The first symmetric encryption was a Caesar's cipher, which consists in a simple transposition of the alphabet letters.

As long as cryptanalysis unveiled ciphers, cryptographers had to develop safer ciphers.

We've passed through the evolution of monoalphabetic to the polyalphabetic ciphers, whose the most well-known cipher is Vigènere. This cipher was the base to the conception and creation of Enigma machine, the most dreadful weapon of encryption used in World War.

In the 60's, the idea of public-key encryption began to emerge. It was not a discovery of a unique person or a unique moment in the History.

It took a few years to develop the theoretical proof to its practical application, known as RSA key, practically impossible to be broken by the current computer technology.

In the 20th century, the quantum computer's idea was developed. Its enormous capability could, theoretically, break a public-key encryption in a few minutes.

Along with this idea, another concept, even more theoretical, emerged: quantum encryption. Based on quantum elements, like photons storage, its objective was transcribing messages.

in all, we presented a practical work based on the RSA encryption implementation, using C# language.

Sumario

1 Objetivo do trabalho..... 4
2 Introdução..... 5
3 Criptografia (conceitos gerais)..... 8
4 Técnicas criptográficas mais utilizadas..... 10
5 Dissertação..... 14
6 Estrutura da criptografia assimétrica RSA..... 18
6.1 Angelita tem um software de criptografia assimétrica..... 18
6.2 Multiplicar 2 números..... 18
6.3 Repositório de chaves públicas..... 18
6.4 Cifrar uma letra..... 19
6.5 Aplicada a cada caractere..... 19
6.6 Número relativo primo..... 19
6.7 Cifrar este número M..... 19
6.8 Fórmula numa calculadora convencional..... 19
6.9 Módulo da matemática..... 20
6.10 Dois números primos *p* e *q*..... 20
6.11 Conseguir decifrar a mensagem..... 20
6.12 Software de Angelita..... 20
6.13 Exemplificação prática..... 21
7 Estrutura da criptografia simétrica DES..... 22
7.1 Mensagem..... 22
7.2 Fileira..... 22
7.3 Bloco..... 22
7.4 64 dígitos..... 22
7.5 Função mutiladora..... 22
7.6 Original..... 22
7.7 Conjunto de operação..... 23
7.8 Processo é repetido..... 23
8 Estrutura da criptografia simétrica AES..... 24
9 Relatório com as linhas de código..... 25
10 Blibiografias..... 46

1. OBJETIVO DO TRABALHO

Este trabalho tem como objetivo apresentar uma introdução a história da ciência da criptografia e da criptoanálise. E, por fim, apresentar uma implementação em linguagem C#. Nosso programa, conforme será demonstrado posteriormente, utiliza a criptografia de chave assimétrica com tecnologia RSA.

2. INTRODUÇÃO

Sun Tzu (544 – 456 a,C.), através de sua obra A Arte da Guerra, defendia que a informação era uma das armas mais importantes em uma guerra. Ensinava a importância de obter informação dos inimigos, bem como, a protegê-la para que eles não a obtivessem.

Há mais de 2500 anos, militares e governantes precisam transmitir e guardar informações de forma segura, afim de que não caiam em mãos inimigas.

Com a posse de determinada informação, pode-se antever manobras militares, evitando assim o indesejado efeito surpresa. Estadistas poderiam perder a sua governabilidade, caso suas informações fossem interceptadas por terceiros.

A primeira forma de esconder informação, que não era um código cifrado propriamente dito, era a estenografia. Simplesmente é definida como a arte de se esconder determinada informação. Na Esparta antiga, utilizava-se a estenografia, determinadas informações eram transportadas em cintos e, quando um soldado chegava ao seu destino, desenrolava-o e lia o que estava escrito atrás do acessório.

Existem muitas outras formas de estenografia. Durante a época dos governos absolutistas, conspiradores transportavam informações em tampas de barris de vinho e outros lugares inusitados, como em madeira de carroças.

Mesmo sendo antiga, a estenografia ainda é utilizada. Espiões da 1ª e 2ª guerra utilizaram tinta invisível. O receptor da mensagem conseguiria ler o conteúdo apenas se colocasse em contato com outro produto ou contra um determinado tipo de luz, revelando seu conteúdo.

Através da estenografia, a forma mais rudimentar de criptografia, desenvolveu-se outra ciência, a criptoanálise. Esta é a ciência que se define como a arte de quebrar códigos e, assim, descobrir informações escondidas. É a contrapartida da criptografia.

Fizemos esta definição de estenografia para salientar a importância desta no início da “guerra” entre criptografia e criptoanálise. “ **E, mesmo sendo antiga, o conceito da estenografia tem sua importância ainda hoje, com o objetivo de reforçar a segurança de determinado código cifrado, pois, escondendo-o, aumenta-se a dificuldade de descobri-lo**”.

“ Segundo SINGH Simon. O livro dos códigos. Rio de Janeiro: Record, 2003, p.124.”

Quando a estenografia perdeu sua eficiência, os criptógrafos desenvolveram outra técnica de criptografia, essa sim, utilizando cifras ou alfabetos cifrados.

" **Uma cifra historicamente famosa é a de César. Este líder na Roma Antiga utilizava uma técnica chamada cifra de transposição de letras. Onde colocava um alfabeto correspondente a outro, mas neste 2º alfabeto, havia um deslocamento de 3 casas para a direita, assim a letra A tinha o seu correspondente com a letra D, a letra B com a letra E, etc."**

" Segundo SINGH Simon. O livro dos códigos. Rio de Janeiro: Record, 2003, p.30."

Apesar de ser uma cifra extremamente rudimentar comparado a criptografia atual, teve sua eficiência comprovada no passado pelo fato de que eram poucas as pessoas que tinham o conhecimento da escrita e, mesmo através deste conhecimento, precisava-se de certa erudição sobre o alfabeto, a fim de se deduzir que se tratava apenas de uma transposição. Da mesma forma que os criptoanalistas "quebraram" a estenografia séculos antes, esta cifra de substituição perdeu sua eficiência. Com isso, os criptógrafos desenvolveram um novo tipo de criptografia, a chamada, criptografia de substituição monoalfabética.

Este tipo de cifra foi um marco na história da criptografia, onde foi introduzido o conceito de chave. Isso nada mais era do que uma palavra ou uma pequena frase, sem espaços, inserida no início de um alfabeto. Chegando ao tamanho da chave, aí, o alfabeto continuava na mesma sequência que antes. Esta técnica teve sua eficiência por mais de 1000 anos e, somente com os criptoanalistas árabes, através da utilização da ferramenta matemática de análise de frequência, obtiveram êxito em quebrar cifras que utilizavam esta criptografia.

A evolução da criptografia de substituição monoalfabética deu-se através da cifra de substituição polialfabética. **" Foi criada por *Blaise de Vigenère* e sua dificuldade de quebra a fez levar a alcunha de *Le Chiffre Indéchiffrable*. A análise de frequência não era suficiente para quebrá-la e, depois de alguns séculos, somente com Charles Babbage, houve progresso para decifrá-la, através da técnica de ciclos de repetição de letras, desmembrando-a em cifras monoalfabéticas e, depois disso, aplicando a análise de frequência".**

" Segundo SINGH Simon. O livro dos códigos. Rio de Janeiro: Record, 2003, p.81."

Até o início 2ª Guerra Mundial, não havia mecanização da criptografia e, era impraticável construir textos cifrados maiores, utilizando uma técnica eficiente como a de *Vigenère* (a polialfabética) pois, levava-se um tempo enorme para o emissor criar e depois para o receptor decifrar, além da enorme possibilidade de erros entre criptografia e descriptografia.

Entre a 1ª e a 2ª Guerra Mundial, houve o desenvolvimento da mecanização da criptografia. Os alemães criaram a máquina Enigma. Esta máquina utilizava os conceitos da criptografia polialfabética. Através de ajustes iniciais feitos em base de livros códigos pelos seus operadores, eram criados códigos extremamente difíceis de serem quebrados. Houve grandes esforços de países como Polônia e Inglaterra, que envolveram espionagem e recrutamento de grandes matemáticos pelos governos, para tentar quebrar esta terrível cifra. Esta era transmitida através de código Morse pelos operadores de rádio alemães. Se não fosse o esforço dos criptoanalistas aliados, possivelmente os nazistas teriam ganhado a guerra.

Apesar de toda esta evolução por dois milênios, a criptografia ainda tinha suas fraquezas. Além de transportar a informação, havia a necessidade de se enviar a chave para decifrar a mensagem. Os próprios alemães eram obrigados a imprimir e transportar extensos livros de códigos no campo de guerra para que os operadores regulassem a máquina Enigma. Assim, havia uma simetria entre a chave do emissor e a chave do receptor da mensagem.

Entre o final da década de 1960 e metade dos anos 1970 criou-se e desenvolveu-se a aplicação prática de um conceito que foi a maior descoberta da criptografia em toda a sua história. Três pesquisadores criaram o conceito da chave assimétrica e, alguns anos depois, outros três pesquisadores, criaram sua implementação prática, que é a criptografia assimétrica RSA. Diferente de tudo o que existia até então, a chave do receptor e do emissor da mensagem não era a mesma, ou seja, eram assimétricas. Para conseguir tal feito, conforme veremos mais a frente, foram utilizados conceitos matemáticos como fatoração e as propriedades dos números primos.

3. CRIPTOGRAFIA (CONCEITOS GERAIS)

A criptografia é a ciência que estuda técnicas que visam proteger informações de terceiros.

Até o início da Internet, informações criptografadas eram utilizadas apenas por militares, governantes e grandes corporações que queriam proteger seus segredos como fórmulas de determinados produtos.

A criptografia se desenvolveu pois existia outra ciência que trabalha exclusivamente com o objetivo de decifrar seus códigos, a criptoanálise. Se ninguém conseguisse quebrar as cifras mais rudimentares, possivelmente a criptografia não se desenvolveria ao nível que chegou hoje. Todos os tipos de criptografias desenvolvidos até o final dos anos 1960 são de chaves simétricas. Praticamente era um axioma o fato de que tanto o emissor e o receptor da informação precisavam possuir a mesma chave a fim de poder descriptografar a informação transmitida. A necessidade do receptor ter que conhecer a chave, é um grande problema pois, pode ser mais uma pista para um criptoanalista conseguir quebrar a cifra através da interceptação desta chave.

Este problema foi resolvido apenas com a descoberta da chave de criptografia assimétrica. Esta utiliza o conceito de chave de mão única. Utiliza-se de conceitos matemáticos como fatoração e a de números primos, onde um número é divisível apenas por 1 (um) ou por ele mesmo.

Devido a complexidade desta chave, só é possível sua utilização em computadores. Mesmo os primeiros computadores tinham enormes esforços para decifrar esta chave.

“ Alguns softwares como o PGP utilizam em simbiose a chave assimétrica e a simétrica para transmitir a informação. Devido ao custo de processamento necessário em decifrar uma mensagem utilizando uma criptografia assimétrica pura, este software criptografa com este tipo de criptografia somente a chave simétrica e, ao chegar a mensagem ao receptor, é descriptografada pela chave assimétrica e, depois disso, é aplicada sobre a mensagem. Este processo é interno e transparente ao usuário do software.”

“ Segundo SINGH Simon. O livro dos códigos. Rio de Janeiro: Record, 2003, p. 319.”

Conceituando o que é criptografia de chave simétrica, resume-se que a chave do emissor tem que ser idêntica a do receptor. E, independente do tipo de cifra

utilizada, sempre haverá uma falha conceitual de segurança pois além do cuidado de transmitir a informação, a chave, de alguma forma, também tem que ser repassada ao destinatário. O ápice de utilização deste tipo de chave se deu na 2ª Guerra. Esta guerra quase foi vencida pela Alemanha pois este país criou uma aparelho mecânico chamado Enigma que cifrava e decifrava mensagens a uma grande velocidade cifras do tipo polialfabética.

Houve enorme esforço dos países aliados na interceptação e tentativa de decifrar mensagens nazistas. O primeiro país a conseguir o feito de quebrar a cifra polialfabética da Enigma foi a Polônia, graças a *Marian Rejewski*. Este foi o primeiro criptógrafo a conseguir quebrar o código da máquina enigma, em suas primeiras versões. Quando seu país foi invadido, as autoridades passaram o conhecimento descoberto por *Rejewski* a Inglaterra.

Os aliados britânicos, com vastos recursos financeiros, puderam continuar o trabalho do matemático polonês. A Inglaterra recrutou os melhores profissionais de diversas áreas de seu país e reuniu-os num setor militar chamado *Bletchley Park*. O principal personagem da luta da Inglaterra contra a criptografia nazista foi *Alan Turing*, também considerado o pai da Informática. Ele criou recursos mecanizados conhecidos como *bombas de turing*, que ajudaram a quebrar as cifras criadas através das novas versões da Enigma. Não existia somente um tipo desta máquina, para os submarinos era um tipo, para o primeiro escalão nazista, existia outro.
Muitas vidas foram poupadas, muitos submarinos inimigos foram localizados e interceptados no Atlântico e, hoje com certeza, o mundo é um lugar melhor graças a esta equipe secreta de *Bletchley Park*. Somente nos anos de 1970, é que foram descobertos seus feitos e esforços sobre-humanos para ajudar o mundo a vencer os nazistas.

Ainda nos anos de 1970, a história da criptografia começou a escrever um novo capítulo em sua história. Desenvolveu-se a prova teórica e a implementação prática de uma nova técnica de criptografia, a criptografia assimétrica. Foi o maior avanço da criptografia desde quando o ser humano inventou o primeiro meio de esconder informações de terceiros.

4. Técnicas criptográficas mais utilizadas

Nos anos de 1960, a criptografia não era somente utilizada pelos governos e militares. As grandes corporações, conforme adquiriam computadores, começaram a utilizar a criptografia também.

Com isso, o governo americano percebeu que poderia perder o controle em questões que envolvessem problemas com a segurança nacional quando se deparassem com eventual informação criptografada. Através de uma cifra criada nos laboratórios da IBM, chamada Lucifer e, de pesquisadores da NBS (*National Bureau of Standards*), criou-se um padrão de criptografia simétrica, que é utilizada até hoje, a chamada DES (*Data Encription Standard*).

Ela é a forma de criptografia simétrica mais segura já criada até então. Ela tem várias versões, a de uso comercial é a de 64 bits, já suficiente para proteger determinada informação com alto grau de segurança. Existem outras versões de criptografia DES, com outros níveis de segurança, como 192 bits, a chamada 3DES.

Este tipo de criptografia encorajou as empresas a utilizarem mais a criptografia para suas comunicações. Para cifrar e decifrar seu conteúdo, como qualquer criptografia simétrica, é necessário conhecer a chave, porém não é possível fazer isso de forma plausível apenas manualmente, necessitando o uso de computadores. Embora seja complexo o processo para decifrar da DES, existe um padrão que a agência nacional de segurança dos Estados Unidos, ainda consegue decifrar, mesmo que, para isso, precise de supercomputadores.

A DES utiliza cifras em blocos de 64 bits, algo em torno de 100.000.000.000.000.000 possibilidades de chave. A NSA acreditava que nos anos de 1970, seria suficiente para manter segura a comunicação entre as empresas.

Existem outros padrões de criptografia simétrica criados desde os anos de 1970, como o AES (*Advanced Encryption Standard*), que é o padrão hoje utilizado pelos Estados Unidos.

Apesar da criptografia simétrica ter adquirido um alto grau de segurança com a DES, 3DES, AES, e RC4, ainda existia um problema que se perdurava por mais de 2000 anos, que é o da distribuição de chaves. Não importa o quão complexo é a forma de cifra simétrica a ser utilizada, o emissor e o receptor da mensagem ainda precisariam conhecer, além da mensagem, a chave para decifrá-la.

Nos anos 1970, três pesquisadores, através de grande esforço intelectual, conseguiram criar o conceito de chave assimétrica. Estes pesquisadores são Hellman, Markle e Diffie. O grande nível de segurança que temos hoje nas comunicações,foi conquistado, com a ajuda destes três pesquisadores.

Era dado como um axioma matemático o fato de que tanto o emissor e receptor tinham que conhecer a chave para quebrar determinada cifra. Estes três pesquisadores conseguiram criar a prova teórica de que poderia existir uma chave assimétrica. Embora terem criado a teoria que tornava possível existir uma criptografia com chave assimétrica, porém não obtiveram sucesso em tornar isso possível praticamente.

Foram outros três pesquisadores que conseguiram descobrir algumas regras matemáticas que tornava isso possível. Esses pesquisadores são Rivest, Shamir e Adleman. Estes criaram a primeira implementação de criptografia assimétrica e a mais utilizada até hoje, chamada criptografia RSA.

Hoje, quase toda comunicação segura via Internet, utiliza-se de criptografia assimétrica baseada em RSA. Devido a lentidão que ainda perdura para quebrar esta cifra, ela é utilizada sempre em conjunto com outras cifras simétricas, como DES, AES, etc.

O primeiro software que foi utilizado em grande escala com o objetivo de utilizar a criptografia DES/RSA em conjunto foi o PGP. Seu criador, Phil Zimmermann, sofreu inclusive investigação e processo do FBI, pois este tipo de criptografia poderia ser usada por inimigos dos Estados Unidos. Depois de anos de discussão e de mobilização da opinião pública, Zimmermann foi absolvido do processo.

Generalizou-se o uso da criptografia desde os anos 1980 e, aumentou-se ainda mais o ritmo de uso com a utilização da Internet no mundo. Protegemos hoje nossos e-mails e computadores com simples senhas. Estas, geralmente, não utilizam a criptografia propriamente dita. Porém, quando entramos em determinados sites, como os de comércio eletrônico ou de bancos, utilizamos avançada criptografia assimétrica e simétrica como AES/RSA ou DES/RSA de forma transparente.

A comunicação é toda criptografada, afim de proteger tanto o internauta como também o banco de dados da empresa que controla o site. Todo cracker que tenta interceptar a comunicação acaba por se tornar um criptoanalista em potencial,

porém, um quebrador de código certamente não terá sucesso, mesmo utilizando todo recurso que a tecnologia atual possa oferecer.

“Qualquer informação que é criptografada com a técnica de criptografia simétrica de blocos (DES, AES, etc) em conjunto com criptografia assimétrica (RSA), acaba se tornando quase impossível de ser decifrada. Qualquer computador do mundo, mesmo que trabalhasse durante bilhões de anos, não conseguiria quebrar esta cifra. “

“ Segundo SINGH Simon. O livro dos códigos. Rio de Janeiro: Record, 2003, p. 200.”

Hoje em dia a única esperança da ciência da criptoanálise de quebrar uma cifra que utiliza a criptografia assimétrica RSA e a criptografia simétrica DES ou AES, é utilizar um computador quântico.

“ Toda física quântica é baseada no conceito teórico do paradoxo de uma experiência imaginária chamada gato de Schrödinger. Este conceito é uma ideia simples que indica que, um determinado cientista no papel de observador não consegue determinar o estado de vida ou morte de uma cobaia, no caso, um gato, onde este está dentro de uma caixa fechada, junto a um veneno. Analogicamente tomamos a situação de determinada partícula subatômica, onde não podemos determinar seu estado, ora onda, ora matéria”.

Segundo http://www.bbc.co.uk/news/science-environment-19879890

Não se tem notícia que este computador exista. Há rumores que a NSA possui algum computador quântico mas, são só apenas rumores.

Se existisse um computador quântico, a ciência daria um novo passo de desenvolvimento de cifras, que é a criptografia quântica. Esta criptografia, ainda está totalmente no conceito teórico e, para existir, depende de tecnologia que ainda não existe como armazenamento de fótons. É um campo de estudo totalmente novo.

Alguns cientistas achavam impossível mensurar particulas subatômicas sem destruí-las. Mas, em 2012, os cientistas *David Wineland* e *Serge Haroche* ganharam o prêmio nobel de física por descobrir uma forma de medi-las. É o maior passo da história da humanidade em busca da criação de um computador quântico.

Quando este computador quântico chegar, provavelmente a criptografia assimétrica RSA será facilmente decifrada. Em contrapartida, nova forma de cifra virá, que é a criptografia quântica.

O conceito desta criptografia veio através de uma ideia de um cientista chamado *Stephen Wiesner*, que criou um conceito novo para moeda em papel, chamado dinheiro quântico, que seria impossível falsificar.

Baseia-se em fótons armazenados no papel moeda e, só conseguiria ser decifrado através de um SPCM (*Single Photon Counting Module*), que indica o número de fótons presentes em um pulso de luz. Apesar de absurda, esta ideia foi o primeiro passo para o desenvolvimento teórico da criptografia quântica, que é a sucessora direta da criptografia assimétrica.

5) DISSERTAÇÃO

Conforme apresentado neste texto, foram desenvolvidos vários tipos de criptografias durante a história. O programa que desenvolvemos poderia trabalhar com o mais rudimentar tipo de cifra, com a cifra de César, bem como temos opção de trabalhar com o que há de mais avançado, que é a criptografia assimétrica RSA.

Estando num curso de ciência da computação, exploramos o máximo possível o potencial de nossos estudos com a linguagem de computação a ser aplicada, bem como a mais eficiente tecnologia de criptografia existente, que é assimétrica.

Optamos em desenvolver um pequeno software que abrange estes estudos. Desenvolvemos um programa em GUI (*Graphic User Interface*) com C Sharp. Utilizamos recursos da própria linguagem que trabalham com este tipo de chave.

Antes da explicação das funções sobre o software criado e sobre o procedimento para cifragem e decifragem de texto, será descrito o processo da criptografia assimétrica.

A característica da criptografia simétrica define que tanto o emissor, quanto o receptor da informação deve ter a mesma chave. Isso não acontece com a criptografia assimétrica. Esta, como o próprio nome diz, a chave do emissor não é a mesma do receptor da mensagem.

Entra aí o conceito de chave pública e chave privada. Cada pessoa que deseja receber informações criptografadas deve possuir uma chave pública e uma chave privada. Esta última, é de uso exclusivo desta pessoa e não pode ser divulgada de forma alguma para outrem pois então, caracterizaria uma quebra de segurança.

Da mesma forma que é utilizado um software para codificar uma informação - seja ela um simples texto ou um arquivo - é utilizado o mesmo software para decifrar esta informação. A diferença é que no processo para cifrar é aplicado um algoritmo baseado em determinada chave pública e, para decifrar, são utilizadas funções modulares de matemática que trabalham nas correspondentes chaves privadas.

Em tese, quanto mais divulgado fosse determinada chave pública de uma pessoa, melhor seria para ela. Cientistas na época de 1970 pensaram em criar uma

espécie de catálogo telefônico para mantê-las, porém, era inviável. Esta chave pública era uma sequência enorme de números que era basicamente a multiplicação de 2 números primos. Só era possível obter esta chave ou mesmo processá-la para criptografar determinada mensagem ou arquivo, se fosse através de meio digital, ou seja, através de uso de um computador.

Com o surgimento das redes de computadores nas universidades e nos governos, criou-se um novo conceito, chamado repositório de chaves públicas. Basicamente é uma lista de consulta comum entre os usuários de criptografia assimétrica, no caso a RSA, onde quando se quer mandar certa informação para determinado usuário, utiliza-se desta chave.

Com a popularização da Internet, a criptografia começou a ser utilizada cada vez mais pelas empresas privadas e pelas pessoas. O cenário de repositório de chaves públicas desenvolveu-se também.

Os usuários do software PGP (*Pretty Good Privace*) criado por *Phil Zimmermann* tinham acesso gratuito a um repositório de chaves públicas. Ao criptografar uma mensagem, o PGP automaticamente buscava neste repositório a determinada chave pública do destinatário e, o processo inverso de envio da mensagem, o destinatário se tornava o emissor e, da mesma forma, o PGP buscava a chave neste repositório.

Mesmo com toda a segurança da criptografia assimétrica, ainda existia uma falha de segurança a ser superada. Uma determinada pessoa ou organização má intencionada poderia criar uma chave privada e uma chave pública correspondente. Esta última seria divulgada num repositório de chaves públicas. Alguém que desejasse enviar uma mensagem a esta pessoa, iria se utilizar desta chave pública. Apesar da chave pública ser verdadeira a nível da computação, na realidade, pode não corresponder à pessoa que diz pertencer. Conforme descrito, é o processo normal de criptografia assimétrica, mas com um detalhe: e se a chave pública não fosse realmente da pessoa a quem a mensagem deveria ser criptografada?

Uma pessoa cria uma chave pública se passando por outra pessoa, enganando quem for o emissor da mensagem. Equivale a situação de uma pessoa criar um e-mail falso e conversar com outra, através deste e-mail. Não há como saber se do outro lado está a pessoa que deveria estar.

A partir deste raciocínio que escancarava um problema de segurança na utilização desta criptografia, criou-se um conceito a partir da ideia do repositório de chaves públicas. Este conceito se chama certificado digital.

Empresas surgiram na década de 1990 que ofereciam este tipo de serviço de certificação digital. A função desta empresa era atestar a garantia que determinada chave pública divulgada (*criada com criptografia RSA*) era de quem deveria ser.

Estas empresas, verdadeiros cartórios digitais, tiveram rápido crescimento e lucratividade. Todos que queriam atestar quem eram no mundo digital, pagavam planos mensais, trimestrais ou anuais por este serviço.

Ainda no final da década de 1990 começou a surgir o comércio eletrônico. Ainda hoje as pessoas têm medo de fazer este tipo de compra, medo de expor seus dados pessoais e de cartões de crédito. Este medo tem fundamento. A falha de segurança não está na comunicação em si. Os fracos elos de segurança são o próprio computador do cliente, ou a segurança do banco de dados da empresa que tem o *site* de vendas. A comunicação em si, utilizando camada segura de comunicação, atestada através de certificado digital válido, é totalmente segura. Pois esta utiliza tanto a criptografia assimétrica RSA para transmissão de chave, bem como a ciptografia simétrica DES ou AES para cifrar as mensagens pessoais e de cartão.

Hoje em dia os governos para agilizar processos e reduzir a burocracia assinam e transmitem seus documentos pela internet sem medo de serem falsificados. O governo brasileiro faz suas licitações públicas apenas para empresas que possui um *token* que dá acesso a uma criptografia assimétrica com certificado digital.

Nos últimos anos, criou-se no Brasil, a nota fiscal eletrônica. Com isso, o governo resolveu vários problemas de uma só vez. As empresas privadas são obrigadas a ir a uma autoridade certificadora e adquirir um token para ter acesso a um certificado digital. Assim, ao enviar documentos, elas provam que são elas.

Foi apresentado neste texto todos os conceitos teóricos envolvendo o uso e manuseio da chave assimétrica RSA. No trabalho prático, será apresentado a implementação desta tecnologia, porém, não será utilizado uma autoridade certificadora para garantir que determinada chave pública pertence a determinada pessoa.

Da mesma forma, será criado uma exemplificação de um repositório de chaves públicas, mas este será uma mídia removível, como um disco de memória flash ou uma pasta compartilhada em uma rede local entre os computadores em que será feita a apresentação. Será feito desta forma pois não há garantia que haverá uma rede de computadores disponível na apresentação, bem como, não há certeza se haverá comunicação disponível via Internet. Caso houvesse 100% de garantia de Internet, seria utilizado um diretório FTP, por exemplo, de acesso comum e, neste, manter as chaves públicas afim de demonstrar a criptografia com computadores geograficamente distantes.

6) Estrutura da criptografia assimétrica RSA

Neste tópico, será descrito o funcionamento interno da criptografia RSA. Esta, conforme foi escrito anteriormente, se baseia na criação de uma chave privada e da transmissão da correspondente chave pública para outrem, afim de que faça uma cifra segura e única desta mensagem.

"**A criptografia assimétrica para ter segurança, precisa trabalhar com conceitos matemáticos como módulo e números primos gigantescos (pelas suas características únicas na Matemática). Apesar de sua descoberta, independe do computador existir ou não, sua utilização, na prática, precisa sim de computadores, pela alta quantidade de processamento a ser feito.**"

"Segundo http://pt.wikipedia.org/wiki/Autoridade_de_Certifica%C3%A7%C3%A3o"

Para facilitar a demonstração de nosso trabalho, vamos exemplificar o nome de 2 pessoas, estas , utilizando números primos muito pequenos. Estas pessoas serão Angelita e Dirceu, nossos professores e, estes, hipoteticamente, escolherão 2 números primos cada um.

Vamos a exemplificação:

6.1 Software de criptografia assimétrica

Angelita tem um software de criptografia assimétrica e, este vai gerar 2 números hipotéticos, chamados p e q. Sendo p =17 e q = 11;

6.2 Multiplicar 2 números

O software da Angelita vai multiplicar estes 2 números e, com isso, vai conseguir outro número primo. Que vamos chamar de N = 187. Além deste, ela vai escolher outro número primo, que vamos chamar de número relativo "e", que vai ter conteúdo de 7 (e = 7) ;

6.3 Repositório de chaves públicas

Angelita divulga num repositório de chaves públicas ou mesmo envia a seguinte informação para outra pessoa, no caso, Dirceu, contendo o seguinte: Nome: Angelita, Chave pública: 187 e 7, sendo *N=187* e *"e"=7*;

6.4 Cifrar uma letra

Supondo que alguém queira cifrar uma letra para Angelita, esta letra, primeiramente, deve ser convertida em um número M. Uma letra pode ser convertida em dígito binário em ASCII e, estes dígitos binários podem ser considerados como um número decimal M. Este número M vai ser cifrado produzindo o texto cifrado C, de acordo com a fórmula. $C = M^e \pmod N$;

6.5 Aplicada a cada caractere

Esta equação acima é aplicada a cada caractere da mensagem e, para o determinado número binário de uma letra, representado através de M. Este elevado a 7;

6.6 Número relativo primo

Ao número relativo primo “*e*”, aplicasse o módulo N, sendo N = 187. Se Dirceu deseja enviar apenas uma letra, por exemplo: X, para Angelita, no ASCII, é representado por 1011000, que equivale a 88 em decimais (*M = 88*);

6.7 Cifrar este número M

Para cifrar este número M, Dirceu vai buscar a chave pública de Angelita (N = 187 e “e” = 7). De posse da fórmula $C = M^e \pmod N$, ele chega a $C = 88^7 \pmod{187}$;

6.8 Fórmula numa calculadora convencional

Calcular esta fórmula numa calculadora convencional não é uma tarefa trivial devido ao tamanho padrão do mostrador. Nós temos que desmembrar utilizando a própria Matemática. Sabendo que, 7=4+2+1, temos que trocar o número 7 exponencial pelo 4, 2 e 1, em 3 fórmulas diferentes. Após isso, aplicamos a multiplicação destes e, no final, aplicamos o módulo 187. Assim, sabemos que o texto cifrado C = 11. Este número que Dirceu tem que enviar para Angelita, que representa o caractere “X” , cifrado;

6.9 Módulo da matemática

Esta equação pertence ao módulo da matemática chamado Aritimética Modular e, são funções de mão única. Mesmo sendo apenas um caractere X, seu correspondente criptografado é 11, é muito difícil para uma terceira pessoa descobrir este caractere, usando a engenharia reversa;

6.10 Dois números primos *p* e *q*

Somente a pessoa que conhece os dois números primos *p* e *q,* cujo resultado da multiplicação é o módulo 187, pode decifrar de uma maneira prática a mensagem. A pessoa que tem esses números é a Angelita, que criou a chave privada e repassou apenas a chave pública a Dirceu. Quando ela receber a mensagem cifrada *C = 11* ela vai conseguir decifrar pois, ela tem a chave privada *p = 17* e *q = 11*;

6.11 Conseguir decifrar a mensagem

O software da Angelita vai conseguir decifrar a mensagem cifrada usando uma sequência de comandos matemáticos. A primeira coisa a ser realizada, é achar um número *d*, através de um algoritimo (chamado Euclides). Segue a exemplificação;

e x d =1 (mod (p-1) x (q-1))

7 x d = 1 (mod 16 x 10)

7 x d = 1(mod 160)

d = 23

6.13 Software de Angelita

Agora o software de Angelita tem todos os recursos para conseguir decifrar a mensagem que foi enviada por Dirceu. Lembrando que Dirceu enviou o código C = 11, que indica a criptografia da letra X;

$$M = C^{d} \pmod{187}$$

$$M = 11^{23} \pmod{187}$$

$$M = 88$$

Sendo que, 88 é X em ASCII

6.14 Exemplificação prática

Esta foi uma exemplificação prática para converter apenas um caractere, isso, utilizando-se números primos muito pequenos. Foi usado apenas para fins didáticos. Lembramos que, na prática, softwares como o openPGP e outros de criptografia utilizam números primos muito grandes apesar de que, usam a mesma técnica demonstrada acima como critério de criptografia.

7) Estrutura da criptografia simétrica DES

Foi um padrão de criptografia criado pelo NBS (National Bureau of Standards), nos anos de 1970. Ele é baseado numa outra criptografia simétrica chamada Lúcifer, criada nos laboratórios da IBM.

É um sistema de criptografia que, da mesma forma que outras técnicas, precisa de um computador para ser processado. Utiliza-se de blocos de 64 bits, onde 8 destes bits servem apenas para o teste de paridade, sendo a parte útil, voltada para mensagem, ou seja, os outros 56 bits.

Para cifrar uma mensagem, existe uma sequência de comandos de mistura conforme descrito abaixo:

7.1 Mensagem

A mensagem é traduzida para uma longa fileira de dígitos binários;

7.2 Fileira

Esta fileira é divida em blocos de 64 dígitos;

7.3 Bloco

Em cada bloco é executada uma cifra;

7.4 64 dígitos

Para cada um dos blocos, os 64 dígitos são misturados e então divididos em dois blocos de 32 dígitos, denominados Direito(0) e Esquerdo(0);

7.5 Função mutiladora

Existe uma “função mutiladora”. Esta muda os dígitos de acordo com uma substituição complexa: O Direito(0) é mutilado e somado ao Esquerdo (0). Assim, cria-se um novo meio bloco de 32 bits chamado Direito (1);

7.6 Original

O Direito(0) original é renomeado para Esquerdo(1);

7.7 Conjunto de operação

Cada conjunto de operação (de 1 a 6) acima é chamado de rodada;

7.8 Processo é repetido

Este processo é repetido sobre todos os blocos de todas as fileira 16 vezes.

O processo de cifra DES, se comparado com uma massa de pão, seria algo como: Imaginemos uma longa peça de massa com uma mensagem escrita nela. Dividimos a peça em pequenos pedaços de 64 cm de tamanho. Então, pegamos metade de cada um dos pedaços, amassamos, dobramos, misturamos com outra metade e a esticamos para fazer um novo pedaço de 64 cm.

Conforme observamos é uma criptografia simétrica extremamente complexa. Ela foi para um padrão chamado 3DES, que basicamente falando, era fazer este processo com 3 níveis a mais de trabalho.

A DES é utilizada ainda hoje. Sua cifra, conforme podemos perceber, é muito difícil de ser quebrada, porém, não impossível. Nos anos de 1990, houve uma ONG, chamada Eletronic Frontier Foundation, que luta pela liberdade de expressão nos meios eletrônicos, criou um software, chamado DES Cracker.

8) **Estrutura da criptografia simétrica AES**

Em 1997, a NIST (National Institute of Standards of Tecnology) se deparou com a quebra de código do DES por uma ONG. Este padrão de cifra era adotado desde os anos 1970 e apesar de ainda ser necessário grande esforço para quebrá-la, ainda sim, isso é possível. Então a NIST lançou um concurso para buscar entre vários concorrentes um novo padrão de cifra simétrica que poderia substituir ou simplesmente conviver com a DES na atualidade.

Depois de 5 anos, em 2002, a NIST conseguiu achar uma cifra que tinha várias condições como: sem direitos autorais, ser de domínio público, ser do tipo simétrico, e ser cifrada a partir de blocos de 128 bits. A cifra ganhadora foi uma chamada Rijindael, uma fusão do nome de seus criadores. Posteriormente ao ser escolhida no concurso, esta cifra foi ligeiramente modificada pela NIST, daí, chamada de AES. A mudança principal de um modelo para outro é que a AES aceita blocos fixos de 128 bits, um padrão que visou facilitar a adoção em âmbito mundial e, o fato de ter blocos diferentes de tamanho, seria um entrave maior a padronização.

A AES é muito superior, tanto em segurança, rapidez de processamento e facilidade de implementação. Devido a estas características, ela é muito utilizada em nível de hardware, por exemplo, roteadores wifi, smartphones, ipads, etc.

A AES, da mesma forma que a DES, utiliza rodadas para desenvolver a criptografia de determinado código. Só que, cada turno, é dividido em alguns estágios, denominado: AddRoundKey, SubBytes, ShiftRows e MixColuns.

Definitivamente o processo para cifrar com a AES é o padrão simétrico mais seguro, está em operação desde 2002 e, gradativamente está sendo adotado pelas empresas. Hoje os certificados digitais utilizam a criptografia AES junto com a RSA, em contrapartida, muitos outros, utilizam DES com RSA.

O descobrimento da AES, não significa que a DES está inútil. Ambas são muito difíceis de serem quebradas. Entretanto, a AES ainda não foi quebrada.

9) **Relatório com as linhas de códigos das páginas**

```
<Window
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:Properties="clr-namespace:Criptografia_com_RSA.Properties" x:Name="formPrincipal"
x:Class="Criptografia_com_RSA.MainWindow"
      Title="Criptografia assimétrica com RSA - prof. Dirceu e Angelita - Novembro/2012" Height="310" Width="783"
WindowStartupLocation="CenterScreen" Icon="favicon.ico" ResizeMode="NoResize" Initialized="formPrincipal_Initialized"
Loaded="formPrincipal_Loaded" Closed="formPrincipal_Closed">
   <Grid HorizontalAlignment="Left" Height="205" VerticalAlignment="Top" Width="673" Margin="10,10,0,0">
      <Grid.RowDefinitions>
         <RowDefinition Height="57*"/>
         <RowDefinition Height="83*"/>
         <RowDefinition Height="55*"/>
      </Grid.RowDefinitions>
      <Rectangle Fill="#FFF4F4F5" HorizontalAlignment="Left" Height="57" Margin="480,10,-251,0" Grid.Row="1"
Stroke="Black" VerticalAlignment="Top" Width="270"/>
      <Rectangle Fill="#FFF4F4F5" HorizontalAlignment="Left" Height="95" Margin="480,71,-77,-21" Grid.Row="1"
Stroke="Black" VerticalAlignment="Top" Width="270" Grid.RowSpan="2"/>
      <Rectangle Fill="#FFF4F4F5" HorizontalAlignment="Left" Height="60" Margin="480,3,-251,0" Stroke="Black"
VerticalAlignment="Top" Width="270" Grid.RowSpan="2"/>
      <Rectangle x:Name="Retangulo1" Fill="#FFF4F4F5" HorizontalAlignment="Left" Height="223" Margin="0,3,0,-21"
Stroke="Black" VerticalAlignment="Top" Width="473" Grid.RowSpan="3"/>
      <Button x:Name="btnGerarChavePrivada" Content="Gerar Chave Privada" Margin="205,86,0,-50" Width="129"
VerticalAlignment="Top" HorizontalAlignment="Left" Panel.ZIndex="1" Click="btnGerarChavePrivada_Click" Grid.Row="2"/>
      <Button x:Name="btnCifrar" Content="Cifrar &amp;gt;&amp;gt;" Margin="193,35,0,0" Width="81" HorizontalAlignment="Left"
Panel.ZIndex="1" Click="btnCifrar_Click" Grid.Row="1" Height="22" VerticalAlignment="Top"/>
      <Button x:Name="btnDecifrar" Content="&amp;lt;&amp;lt; Decifrar" Margin="193,72,0,0" Width="81" VerticalAlignment="Top"
HorizontalAlignment="Left" Panel.ZIndex="1" Click="btnDecifrar_Click" Grid.Row="1" Grid.RowSpan="2"/>
      <Button x:Name="btnFechar" Content="Fechar" Margin="356,86,0,-50" Width="119" VerticalAlignment="Top"
HorizontalAlignment="Left" IsCancel="True" Panel.ZIndex="1" Click="btnFechar_Click" Grid.Row="2"/>
      <Label x:Name="lblCaptionMensagemOriginal" Content="Mensagem Original" HorizontalAlignment="Left"
Margin="6,8,0,0" VerticalAlignment="Top" Width="186"/>
      <TextBox x:Name="textoOriginal" HorizontalAlignment="Left" Height="179" Margin="6,31,0,-15" TextWrapping="Wrap"
VerticalAlignment="Top" Width="186" TextChanged="textoOriginal_TextChanged" Grid.RowSpan="3"/>
```

<Label x:Name="lblCaptionMensagemCifrada" Content="Mensagem Cifrada" HorizontalAlignment="Left" Margin="273,8,0,0" VerticalAlignment="Top" Width="186"/>

<TextBox x:Name="textoCifrado" HorizontalAlignment="Left" Height="151" Margin="277,31,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="186" Grid.RowSpan="3"/>

<Label x:Name="lblTamanhoChave" Content="Nível de segurança do RSA:" Height="29" Margin="487,7,-251,0" VerticalAlignment="Top" FontWeight="Bold" RenderTransformOrigin="4.8,1"/>

<Label x:Name="lblTamanhoMaximoTexto" Content="Label" HorizontalAlignment="Left" Height="29" Margin="480,10,-244,0" VerticalAlignment="Top" FontWeight="Bold" Grid.Row="1" Width="263"/>

<Label x:Name="lblCaptionCaminhoRepositorioChavePublica" Content="Repositório chaves públicas" HorizontalAlignment="Left" Height="26" Margin="480,68,-152,0" Grid.Row="1" VerticalAlignment="Top" Grid.RowSpan="2" FontWeight="Bold" Width="171"/>

<Button x:Name="btnProcurarRepositorio" Content="..." HorizontalAlignment="Left" Margin="716,0,-68,0" Grid.Row="2" Width="25" RenderTransformOrigin="0.741,3.679" Click="btnProcurarRepositorio_Click" Height="25" VerticalAlignment="Top"/>

<Slider x:Name="sliderTamanhoTexto" HorizontalAlignment="Left" Margin="490,39,-244,0" VerticalAlignment="Top" Width="253" Minimum="128" Maximum="1000" ValueChanged="sliderTamanhoTexto_ValueChanged" SmallChange="1" Grid.Row="1"/>

<TextBox x:Name="edtCaminhoRepositorio" HorizontalAlignment="Left" Height="22" Margin="488,3,-36,0" Grid.Row="2" TextWrapping="Wrap" VerticalAlignment="Top" Width="221"/>

<RadioButton x:Name="rdgRSA2048" Content="2048 bits" HorizontalAlignment="Left" Margin="490,36,-91,0" VerticalAlignment="Top" Width="100" IsChecked="True" GroupName="RSA" Click="rdgRSA2048_Click"/>

<RadioButton x:Name="rdgRSA4096" Content="4096 bits" HorizontalAlignment="Left" Margin="575,36,-176,0" VerticalAlignment="Top" Width="100" GroupName="RSA" Click="rdgRSA4096bits_Click"/>

<RadioButton x:Name="rdgRSA6144" Content="6144 bits" HorizontalAlignment="Left" Margin="665,36,-244,0" VerticalAlignment="Top" Width="78" GroupName="RSA" Click="rdgRSA6144bits_Click"/>

<Button x:Name="btnTextoCopiar" HorizontalAlignment="Left" Height="28" Margin="277,47,0,-20" Grid.Row="2" VerticalAlignment="Top" Width="43" Content="Copiar" Click="btnTextoCopiar_Click"/>

<Button x:Name="btnColar" Content="Colar" HorizontalAlignment="Left" Margin="320,47,0,-17" Grid.Row="2" Width="40" Height="28" Click="btnColar_Click"/>

<Button x:Name="btnTextoSalvar" Content="Salvar" HorizontalAlignment="Left" Margin="360,47,0,-20" Grid.Row="2" VerticalAlignment="Top" Width="38" Height="28" Click="btnTextoSalvar_Click"/>

<Button x:Name="btnTextoCarregar" Content="Carregar" HorizontalAlignment="Left" Margin="398,47,0,-20" Grid.Row="2" VerticalAlignment="Top" Width="65" Height="28" RenderTransformOrigin="0.2,1" Click="btnTextoCarregar_Click"/>

<Label x:Name="lblCaptionCaminhoRepositorioTrocaMensagem" Content="Repositório troca mensagens cifradas" HorizontalAlignment="Left" Height="26" Margin="480,26,-210,0" Grid.Row="2" VerticalAlignment="Top" FontWeight="Bold" Width="229"/>

<TextBox x:Name="edtCaminhoTrocaMensagem" HorizontalAlignment="Left" Height="22" Margin="488,48,-36,-12" Grid.Row="2" TextWrapping="Wrap" VerticalAlignment="Top" Width="221"/>

```
        <Button x:Name="btnProcurarTrocaMensagem" Content="..." HorizontalAlignment="Left" Margin="716,45,-68,-12"
Grid.Row="2" Width="25" RenderTransformOrigin="0.741,3.679" Click="btnProcurarTrocaMensagem_Click"/>
    </Grid>
</Window>
```

MainWindow.xaml.cs

```
using System;
using System.IO;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Windows;
using System.Windows.Forms;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Security.Cryptography;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;


namespace Criptografia_com_RSA
{

    public partial class MainWindow : Window
    {
        private int tamanhotexto; // tamanho máximo do texto permitido a criptografar
        public int tamanhoTexto
        {
            get
            { return tamanhotexto; }
            set
            {
                tamanhotexto = value;
                lblTamanhoMaximoTexto.Content = "Tamanho máximo texto: " + value.ToString()+" caracteres";
                setMudancaTextoOriginal();
            }
        }

        private int rsabits = 2048; // quanto maior o número , mais difícil quebrar
        public int rsaBits
        {
            get
            { return rsabits; }
            set
            {
                rsabits = value;
            }
        }

        public MainWindow()
        {
            InitializeComponent();
        }

        private void setMudancaTextoOriginal()
```

```
{
    this.lblCaptionMensagemOriginal.Content =
        "Mensagem Original - " +
        this.textoOriginal.Text.Length.ToString() + "/" +
        this.tamanhoTexto.ToString();
}

private Parametros GetParametros()
{
    Parametros parametros = null;
    if (File.Exists("Parametros.bin"))
    {
        StreamReader streamReader = new StreamReader("Parametros.bin");
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        parametros = (Parametros)binaryFormatter.Deserialize(streamReader.BaseStream);
        streamReader.Close();
    }
    return parametros;
}

private void copiarTextoCifradoAreaTransferencia()
{
    if (textoCifrado.Text.Length == 0)
    {
        System.Windows.MessageBox.Show("Não há texto criptografado para ser copiado!",
                                       "Erro");

    }
    else
    {
        textoCifrado.SelectAll();
        System.Windows.Forms.Clipboard.SetText(textoCifrado.Text);

        System.Windows.MessageBox.Show("Texto copiado com sucesso a área de transferência!",
                                       "Área de transferência");
    }
}

private void setParametros(string settingChanged)
{
    Parametros parametros = new Parametros();

    if (File.Exists("Parametros.bin"))
    {
        StreamReader streamReader = new StreamReader("Parametros.bin");
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        parametros = (Parametros)binaryFormatter.Deserialize(streamReader.BaseStream);
        streamReader.Close();

        switch (settingChanged)
        {
            case "REPOSITORIO":
                {
                    parametros.diretorioRepositorio = edtCaminhoRepositorio.Text;
                    break;
                }
            case "TAMANHOTEXTO":
                {
                    parametros.tamanhoTexto = this.tamanhoTexto;
                    break;
                }
            case "RSABITS":
                {
                    parametros.rsaBits = this.rsaBits;
                    break;
                }
            case "TROCAMENSAGEM":
```

```
            {
                parametros.diretorioTrocaMensagem = edtCaminhoTrocaMensagem.Text;
                break;
            }
        }

        StreamWriter streamWriter = new StreamWriter("Parametros.bin", false);
        binaryFormatter.Serialize(streamWriter.BaseStream, parametros);
        streamWriter.Close();
    }
    else
    {
        parametros.diretorioRepositorio = edtCaminhoRepositorio.Text;
        parametros.diretorioTrocaMensagem = edtCaminhoTrocaMensagem.Text;
        parametros.tamanhoTexto = this.tamanhoTexto;
        parametros.rsaBits = this.rsaBits;

        StreamWriter streamWriter = new StreamWriter("Parametros.bin", false);
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        binaryFormatter.Serialize(streamWriter.BaseStream, parametros);
        streamWriter.Close();
    }

}

private void btnGerarChavePrivada_Click(object sender, RoutedEventArgs e)
{
    try
    {
        ManipulacaoArquivo manipulacaoarquivo = new ManipulacaoArquivo(rsaBits);

        if (!Directory.Exists(edtCaminhoRepositorio.Text))
        {
            throw new Exception("Diretório do caminho do repositório não foi encontrado!");
        }

        string mensagem = "";

        bool continuarProcesso = true;

        if (rsaBits == 4096)
        {
            mensagem = "A chave será gerada com tamanho de " + rsaBits.ToString() + " bits!\n" +
                    "Este processo poderá levar alguns segundos.\n" +
                    "Deseja criar nova chave privada?";
        }
        else if (rsaBits == 6144)
        {
            mensagem = "A chave será gerada com tamanho de " + rsaBits.ToString() + " bits!\n" +
                    "Este processo poderá levar alguns minutos.\n" +
                    "Deseja criar nova chave privada?";
        }

        // se a chave for mais exigir mais tempo pra ser gerada, fazer uma pergunta anterior
        if (mensagem != "")
        {
            continuarProcesso =
            System.Windows.MessageBox.Show(mensagem,
                                "Confirmação",
                                System.Windows.MessageBoxButton.YesNo) == System.Windows.MessageBoxResult.Yes;
        }

        if (continuarProcesso)
        {
            // nome e path do arquivo onde será gravado cada chave
            string arquivoChavePrivada = "";
            string arquivoChavePublica = "";
```

```
            // contém o XML das chaves
            string chavePrivada = "";
            string chavePublica = "";

            UtilizacaoRSA rsa = new UtilizacaoRSA(rsaBits);

            manipulacaoarquivo.getNomeArquivoChavePrivadaNovo(ref arquivoChavePrivada); // Selecionar novo Arquivo de
chave Privada

            if (arquivoChavePrivada != "")
            {
               // Alimentar a variável para apontar onde deve ser gravado a chave Pública
               arquivoChavePublica = System.IO.Path.ChangeExtension(arquivoChavePrivada, "pub"); // trocando a extensão
               arquivoChavePublica = System.IO.Path.GetFileName(arquivoChavePublica); // nome do arquivo sem o path

               arquivoChavePublica = edtCaminhoRepositorio.Text + System.IO.Path.DirectorySeparatorChar +
arquivoChavePublica; // Novo Caminho

               Cursor = System.Windows.Input.Cursors.Wait;

               btnGerarChavePrivada.Content = "Aguarde...";
               btnGerarChavePrivada.IsEnabled = false;
               System.Windows.Forms.Application.DoEvents(); // Libera Mensagens do processamento do Windows

               try
               {
                  rsa.gerarChavePublicaePrivada(ref chavePrivada, ref chavePublica);

                  manipulacaoarquivo.salvarConteudoEmArquivo(arquivoChavePrivada, chavePrivada); // salvando chave
privada no disco removível
                  manipulacaoarquivo.salvarConteudoEmArquivo(System.IO.Path.ChangeExtension(arquivoChavePrivada,
"pub"), chavePublica); // salvando chave pública no disco removível (backup)

                  manipulacaoarquivo.salvarConteudoEmArquivo(arquivoChavePublica, chavePublica); // salvando chave
pública no repositório

                  System.Windows.MessageBox.Show( "Chaves pública e privada foram geradas e gravadas com sucesso!\n"+
                                                  "Atenção: A chave privada deve ser mantida em segurança. Esta em hipótese alguma
deve ser divulgada.",
                                                  "Gravação Ok");
               }
               catch (Exception E)
               {
                  System.Windows.MessageBox.Show(E.Message, "Erro");
               }
               finally
               {
                  this.Cursor = System.Windows.Input.Cursors.Arrow;
               }

               btnGerarChavePrivada.IsEnabled = true;
               btnGerarChavePrivada.Content = "Chave Privada";
            }
         }
      }
      catch (Exception E)
      {
         System.Windows.MessageBox.Show(E.Message, "Erro");
      }
   }

   private void btnCifrar_Click(object sender, RoutedEventArgs e)
   {
      try
      {
         ManipulacaoArquivo manipulacaoarquivo = new ManipulacaoArquivo(rsaBits);
```

```
        if (textoOriginal.Text.Length == 0 ||
                    this.textoOriginal.Text.Length > this.tamanhoTexto)
        {
            throw new ArgumentException("Digite uma mensagem de até " + tamanhoTexto.ToString() + " caracteres!");
        }

        if (!Directory.Exists(edtCaminhoRepositorio.Text))
        {
            throw new ArgumentException("Não foi encontrado o caminho do repositório de chaves públicas: " +
                                edtCaminhoRepositorio.Text);
        }

        string arquivoChavePublica = "";

        manipulacaoarquivo.getNomeArquivoChavePublica(edtCaminhoRepositorio.Text, ref arquivoChavePublica);

        if (arquivoChavePublica != "")
        {
            Cursor = System.Windows.Input.Cursors.Wait;

            btnCifrar.Content = "Aguarde...";
            btnCifrar.IsEnabled = false;

            System.Windows.Forms.Application.DoEvents(); // Libera Mensagens do processamento do Windows

            string xmlChavePublica = manipulacaoarquivo.carregarConteudo(arquivoChavePublica);

            if (xmlChavePublica != "")
            {
                string bitsString = xmlChavePublica.Substring(0, xmlChavePublica.IndexOf("</RSABits>") + 10);
                xmlChavePublica = xmlChavePublica.Replace(bitsString, ""); // Limpando parte da sring
                int bits = Convert.ToInt32(bitsString.Replace("<RSABits>", "").Replace("</RSABits>", ""));

                UtilizacaoRSA rsa = new UtilizacaoRSA(bits);
                rsa.mensagem = textoOriginal.Text;
                rsa.xmlChave = xmlChavePublica;

                textoCifrado.Text = rsa.Cifrar();
                textoOriginal.Text = "";
            }

        }
    }
    catch (Exception E)
    {
        System.Windows.MessageBox.Show(E.Message, "Erro");
    }
    finally
    {
        Cursor = System.Windows.Input.Cursors.Arrow;
        btnCifrar.Content = "Cifrar >>";
        btnCifrar.IsEnabled = true;

    }
}

private void btnDecifrar_Click(object sender, RoutedEventArgs e)
{
    try
    {
        if (!Directory.Exists(edtCaminhoRepositorio.Text))
        {
            throw new Exception("Diretório do caminho do repositório não foi encontrado!");
        }

        ManipulacaoArquivo manipulacaoarquivo = new ManipulacaoArquivo(rsaBits);
```

```
        if (textoCifrado.Text.Length == 0)
        {
            System.Windows.MessageBox.Show("Cole ou carrege algum texto cifrado!", "Erro");
        }
        else
        {
            string arquivo = "";

            manipulacaoarquivo.getNomeArquivoChavePrivadaExistente(ref arquivo);

            if (arquivo != "")
            {
                Cursor = System.Windows.Input.Cursors.Wait;

                btnDecifrar.Content = "Aguarde...";
                btnDecifrar.IsEnabled = false;

                System.Windows.Forms.Application.DoEvents(); // Libera Mensagens do processamento do Windows

                string xmlChavePrivada = manipulacaoarquivo.carregarConteudo(arquivo);

                if (xmlChavePrivada != "")
                {
                    string bitsString = xmlChavePrivada.Substring(0, xmlChavePrivada.IndexOf("</RSABits>") + 10);
                    xmlChavePrivada = xmlChavePrivada.Replace(bitsString, ""); // Limpando parte da sring

                    int bits = Convert.ToInt32(bitsString.Replace("<RSABits>", "").Replace("</RSABits>", ""));

                    UtilizacaoRSA rsa = new UtilizacaoRSA(bits);
                    rsa.mensagem = textoCifrado.Text;
                    rsa.xmlChave = xmlChavePrivada;

                    textoOriginal.Text = rsa.Decifrar();
                    textoCifrado.Text = "";
                }
            }
        }
    }
    catch (Exception E)
    {
        this.textoOriginal.Text = "";
        System.Windows.MessageBox.Show(E.Message, "Erro");
    }
    finally
    {
        Cursor = System.Windows.Input.Cursors.Arrow;
        btnDecifrar.Content = "<< Decifrar";
        btnDecifrar.IsEnabled = true;

    }
}

private void btnFechar_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}

private void textoOriginal_TextChanged(object sender, TextChangedEventArgs e)
{
    setMudancaTextoOriginal();
}

private void formPrincipal_Initialized(object sender, EventArgs e)
{
    edtCaminhoRepositorio.Text = "";

    rsaBits = 128;
```

```
    tamanhoTexto = 128;

    sliderTamanhoTexto.Value = tamanhoTexto;
}

private void btnProcurarRepositorio_Click(object sender, RoutedEventArgs e)
{
    FolderBrowserDialog openFolder = new FolderBrowserDialog();

    if (openFolder.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        edtCaminhoRepositorio.Text = openFolder.SelectedPath;
    }
}

private void formPrincipal_Loaded(object sender, RoutedEventArgs e)
{
    try
    {
        textoCifrado.IsReadOnly = true;
        edtCaminhoRepositorio.IsReadOnly = true;

        if (File.Exists("Parametros.bin"))
        {
            Parametros parametros = GetParametros();

            edtCaminhoRepositorio.Text = parametros.diretorioRepositorio;

            edtCaminhoTrocaMensagem.Text = parametros.diretorioTrocaMensagem;

            tamanhoTexto = parametros.tamanhoTexto;
            sliderTamanhoTexto.Value = this.tamanhoTexto;

            rsaBits = parametros.rsaBits;

            switch (this.rsaBits)
            {
                case 2048:
                    rdgRSA2048.IsChecked = true;
                    break;
                case 4096:
                    rdgRSA4096.IsChecked = true;
                    break;
                case 6144:
                    rdgRSA6144.IsChecked = true;
                    break;
                default:
                    {
                    rdgRSA2048.IsChecked = true;
                    break;
                    }

            }

        }
    }
    catch (Exception E)
    {
        System.Windows.MessageBox.Show("Deve-se Excluir o arquivo Parametros.bin manualmente!\n\n"+
                                E.Message, "Erro");
    }
}

private void sliderTamanhoTexto_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    tamanhoTexto = Convert.ToInt16(Math.Truncate(sliderTamanhoTexto.Value));
}
```

```
private void formPrincipal_Closed(object sender, EventArgs e)
{
   setParametros("TAMANHOTEXTO");
   setParametros("RSABITS");
   setParametros("REPOSITORIO");
   setParametros("TROCAMENSAGEM");
}

private void rdgRSA2048_Click(object sender, RoutedEventArgs e)
{
   rsaBits = 2048;
}

private void rdgRSA4096bits_Click(object sender, RoutedEventArgs e)
{
   rsaBits = 4096;
}

private void rdgRSA6144bits_Click(object sender, RoutedEventArgs e)
{
   rsaBits = 6144;
}

private void btnTextoCopiar_Click(object sender, RoutedEventArgs e)
{
   copiarTextoCifradoAreaTransferencia();
}

private void btnColar_Click(object sender, RoutedEventArgs e)
{
   try
   {
      string areaTransferencia = System.Windows.Forms.Clipboard.GetText();

      if (areaTransferencia.Trim().Length == 0)
      {
         throw new Exception("Não há texto na área de transferência!");
      }

      textoCifrado.Text = areaTransferencia;
      textoOriginal.Text = "";
   }
   catch (Exception E)
   {
      System.Windows.MessageBox.Show(E.Message,
                              "Erro");
   }
}

private void btnTextoSalvar_Click(object sender, RoutedEventArgs e)
{
   if (textoCifrado.Text.Length == 0)
   {
      System.Windows.MessageBox.Show("Não há texto criptografado para ser salvo!",
                              "Erro");

   }
   else if (!Directory.Exists(edtCaminhoTrocaMensagem.Text))
   {
      System.Windows.MessageBox.Show("Diretório de troca de mensagens não encontrado!",
                              "Erro");
   }
   else
   {
      string arquivo = "";
```

```
            ManipulacaoArquivo manipulacaoarquivo = new ManipulacaoArquivo(rsaBits);
            manipulacaoarquivo.getNomeArquivoMensagemCriptografadaNovo(ref arquivo, edtCaminhoTrocaMensagem.Text);

            if (arquivo != "")
            {
               manipulacaoarquivo.salvarConteudoEmArquivo(arquivo, textoCifrado.Text);

               System.Windows.MessageBox.Show("Texto copiado com sucesso em " + arquivo,
                                    "Gravação em disco");

            }
         }
      }

      private void btnTextoCarregar_Click(object sender, RoutedEventArgs e)
      {
         if (!Directory.Exists(edtCaminhoTrocaMensagem.Text))
         {
            System.Windows.MessageBox.Show("Diretório de troca de mensagens não encontrado!",
                                  "Erro");
         }
         else
         {
            string arquivo = "";

            ManipulacaoArquivo manipulacaoarquivo = new ManipulacaoArquivo(rsaBits);

            manipulacaoarquivo.getNomeArquivoMensagemCriptografadaExistente(ref arquivo,
edtCaminhoTrocaMensagem.Text);

            if (arquivo != "")
            {
               textoCifrado.Text = manipulacaoarquivo.carregarConteudo(arquivo);
               textoOriginal.Text = "";
            }
         }
      }

      private void btnProcurarTrocaMensagem_Click(object sender, RoutedEventArgs e)
      {
         FolderBrowserDialog openFolder = new FolderBrowserDialog();

         if (openFolder.ShowDialog() == System.Windows.Forms.DialogResult.OK)
         {
            edtCaminhoTrocaMensagem.Text = openFolder.SelectedPath;
         }

      }

   }
}
```

ManipulacaoArquivo.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

public class ManipulacaoArquivo
{
   private int bitsRSA;

   List<string> listaRemovivel; // lista de drivers removíveis
```

```
public ManipulacaoArquivo(int abitsRSA)
{
   bitsRSA = abitsRSA;

   listaRemovivel = new List<string>(); // criando instância para lista de drivers removíveis
}

private long getTamanhoArquivo(string arquivo)
{
   FileInfo info = new FileInfo(@arquivo);

   return info.Length;
}

public void getListaDriversRemoviveis()
{
   listaRemovivel.Clear();

   // Gerar lista de drivers removíveis
   foreach (var drivers in System.IO.DriveInfo.GetDrives())
   {
      if (drivers.DriveType == DriveType.Removable)
      {
         listaRemovivel.Add(drivers.RootDirectory.ToString());
      }
   }

   if (listaRemovivel.Count == 0)
   {
      throw new Exception("Não foi encontrado um driver removível para guardar a chave privada!");
   }
}

public void getNomeArquivoMensagemCriptografadaExistente(ref string arquivo, string diretorioTrocaMensagem)
{
   try
   {
      arquivo = "";

      System.Windows.Forms.DialogResult resultado;

      OpenFileDialog dialog = new System.Windows.Forms.OpenFileDialog();
      dialog.Title = "Carregar mensagem criptografada em arquivo texto";
      dialog.Filter = "Arquivo Texto (*.txt)|*.txt";
      dialog.DefaultExt = "txt";
      dialog.AutoUpgradeEnabled = true;
      dialog.InitialDirectory = diretorioTrocaMensagem;

      bool continuarLoop = true;

      do
      {
         resultado = dialog.ShowDialog();

         if (resultado == System.Windows.Forms.DialogResult.OK)
         {
            if (System.IO.Path.GetExtension(dialog.FileName).ToUpper() != ".TXT")
            {
               throw new Exception("A mensagem criptografada deve estar em arquivo do tipo TXT!");
            }
            else
            {
               continuarLoop = false;

               arquivo = dialog.FileName;
            }
```

```
            }
            else
            {
                continuarLoop = false;
            }

        }
        while (continuarLoop);

    }
    catch (Exception E)
    {
        System.Windows.MessageBox.Show(E.Message, "Erro ao salvar mensagem");
    }
}

public void getNomeArquivoMensagemCriptografadaNovo(ref string arquivo, string diretorioTrocaMensagem)
{
    try
    {
        arquivo = "";

        System.Windows.Forms.DialogResult resultado;

        SaveFileDialog dialog = new System.Windows.Forms.SaveFileDialog();
        dialog.Title = "Gerar mensagem criptografada em arquivo texto";
        dialog.Filter = "Arquivo Texto (*.txt)|*.txt";
        dialog.InitialDirectory = diretorioTrocaMensagem;

        dialog.OverwritePrompt = false;
        dialog.DefaultExt = "txt";
        dialog.AutoUpgradeEnabled = true;

        bool continuarLoop = true;

        do
        {
            resultado = dialog.ShowDialog();

            if (resultado == System.Windows.Forms.DialogResult.OK)
            {
                if (System.IO.Path.GetExtension(dialog.FileName).ToUpper() != ".TXT")
                {
                    throw new Exception("A mensagem criptografada deve estar em um arquivo TXT!");
                }
                else if (File.Exists(dialog.FileName))
                {
                    continuarLoop = true;
                    System.Windows.MessageBox.Show("Já existe um arquivo gravado neste diretório com o mesmo nome!",
"Erro");
                }
                else
                {
                    continuarLoop = false;

                    arquivo = dialog.FileName;
                }
            }
            else
            {
                continuarLoop = false;
            }

        }
        while (continuarLoop);

    }
```

```
        catch (Exception E)
        {
            System.Windows.MessageBox.Show(E.Message, "Erro ao salvar mensagem");
        }
    }

    public void getNomeArquivoChavePrivadaNovo(ref string arquivo)
    {
        try
        {
            getListaDriversRemoviveis();

            arquivo = "";

            System.Windows.Forms.DialogResult resultado;

            SaveFileDialog dialog = new System.Windows.Forms.SaveFileDialog();
            dialog.Title = "Gerar chave primária ("+bitsRSA.ToString()+" bits) em disco removível";
            dialog.Filter = "Chave privada (*.prv)|*.prv";
            dialog.OverwritePrompt = false;
            dialog.DefaultExt = "prv";
            dialog.AutoUpgradeEnabled = true;

            dialog.CustomPlaces.Clear();

            foreach (var drivers in listaRemovivel)
            {
                dialog.CustomPlaces.Add(drivers);
            }

            bool continuarLoop = true;

            do
            {
                dialog.InitialDirectory = listaRemovivel[0].ToString();

                resultado = dialog.ShowDialog();

                if (resultado == System.Windows.Forms.DialogResult.OK)
                {
                    if (listaRemovivel.IndexOf(System.IO.Path.GetPathRoot(dialog.FileName)) < 0)
                    {
                        continuarLoop = true;
                        System.Windows.MessageBox.Show("Chave primária, por segurança, deve ser gravada em disco removível!",
"Erro");
                    }
                    else if (System.IO.Path.GetExtension(dialog.FileName).ToUpper() != ".PRV")
                    {
                        throw new Exception("A chave privada deve ser um arquivo do tipo PRV!");
                    }
                    else if (File.Exists(dialog.FileName))
                    {
                        continuarLoop = true;
                        System.Windows.MessageBox.Show("Já existe uma chave privada com este nome neste diretório com o
mesmo nome!", "Erro");
                    }
                    else
                    {
                        continuarLoop = false;

                        arquivo = dialog.FileName;
                    }
                }
                else
                {
                    continuarLoop = false;
                }
```

```
        }
        while (continuarLoop);
    }
    catch (Exception E)
    {
        System.Windows.MessageBox.Show(E.Message, "Erro ao salvar chave privada");
    }
}

public void getNomeArquivoChavePrivadaExistente(ref string arquivo)
{
    try
    {
        getListaDriversRemoviveis();

        arquivo = "";

        System.Windows.Forms.DialogResult resultado;

        OpenFileDialog dialog = new System.Windows.Forms.OpenFileDialog();
        dialog.Title = "Abrir chave privada em disco removível";
        dialog.DefaultExt = "prv";
        dialog.AutoUpgradeEnabled = true;
        dialog.Filter = "Chave privada (*.prv)|*.prv";

        dialog.CustomPlaces.Clear();

        foreach (var drivers in listaRemovivel)
        {
            dialog.CustomPlaces.Add(drivers);
        }

        bool continuarLoop = true;

        do
        {
            dialog.InitialDirectory = listaRemovivel[0].ToString();

            resultado = dialog.ShowDialog();

            if (resultado == System.Windows.Forms.DialogResult.OK)
            {
                if (listaRemovivel.IndexOf(System.IO.Path.GetPathRoot(dialog.FileName)) < 0)
                {
                    continuarLoop = true;
                    System.Windows.MessageBox.Show("Chave primária, por segurança, deve estar localizada em um disco
removível!", "Erro");
                }
                else if (System.IO.Path.GetExtension(dialog.FileName).ToUpper() != ".PRV")
                {
                    throw new Exception("A chave privada deve ser um arquivo do tipo PRV!");
                }
                else
                {
                    continuarLoop = false;

                    arquivo = dialog.FileName;
                }
            }
            else
            {
                continuarLoop = false;
            }

        }
        while (continuarLoop);
```

```
    }
    catch (Exception E)
    {
        System.Windows.MessageBox.Show(E.Message, "Erro ao localizar chave privada");
    }
}

public string carregarConteudo(string arquivo)
{
    StreamReader streamReader = new StreamReader(arquivo, true);
    string conteudo = streamReader.ReadToEnd();
    streamReader.Close();

    return conteudo;
}

public void salvarConteudoEmArquivo(string arquivo, string conteudo)
{
    try
    {
        StreamWriter streamWriter = new StreamWriter(arquivo, false);
        streamWriter.Write(conteudo);
        streamWriter.Close();
    }
    catch (Exception E)
    {
        System.Windows.MessageBox.Show(E.Message, "Erro ao salvar conteúdo");
    }
}

public void getNomeArquivoChavePublica(string caminhoRepositorio, ref string arquivo)
{
    try
    {
        System.Windows.Forms.DialogResult resultado;

        OpenFileDialog dialog = new System.Windows.Forms.OpenFileDialog();
        dialog.Title = "Localizar chave pública para criptografar a mensagem";
        dialog.DefaultExt = "pub";
        dialog.InitialDirectory = caminhoRepositorio;
        dialog.AutoUpgradeEnabled = true;
        dialog.Filter = "Chave pública (*.pub)|*.pub";

        bool continuarLoop = true;

        do
        {
            dialog.InitialDirectory = caminhoRepositorio;

            resultado = dialog.ShowDialog();

            if (resultado == System.Windows.Forms.DialogResult.OK)
            {
                if (System.IO.Path.GetExtension(dialog.FileName).ToUpper() != ".PUB")
                {
                    throw new Exception("A chave pública deve ser um arquivo do tipo PUB!");
                }
                else if (getTamanhoArquivo(dialog.FileName) == 0)
                {
                    throw new Exception("Arquivo de chave pública inválido!");
                }
                else
                {
                    continuarLoop = false;

                    arquivo = dialog.FileName;
                }
```

```
            }
            else
            {
                continuarLoop = false;
            }
        }
        while (continuarLoop);
    }
    catch (Exception E)
    {
        System.Windows.MessageBox.Show(E.Message, "Erro ao abrir chave pública");
    }

  }
}
```

Parametros.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

[System.Serializable()]

public class Parametros
{
  private string diretoriotrocamensagem;
  public string diretorioTrocaMensagem
  {
    get
    { return diretoriotrocamensagem; }
    set
    { diretoriotrocamensagem = value; }

  }

  private string diretoriorepositorio;
  public string diretorioRepositorio
  {
    get
    { return diretoriorepositorio; }
    set
    { diretoriorepositorio = value;}

  }

  private int tamanhotexto;
  public int tamanhoTexto
  {
    get
    { return tamanhotexto; }
    set
    { tamanhotexto = value;}
  }

  private int rsabits;
  public int rsaBits
  {
    get
    { return rsabits; }
    set
    { rsabits = value;}
  }
```

```
}
```

UtilizacaoRSA.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.Collections;

//[System.Serializable()]

public class UtilizacaoRSA
{
   private RSACryptoServiceProvider provider;
   private int bitsRSA;
   private int tamanhoChave;
   private int tamanhoMaximoTexto;

   private string xmlchave;
   public string xmlChave // contém o XML da chave privada ou pública
   {
      get
      {
         return xmlchave;
      }
      set
      {
         xmlchave = value;
         provider.FromXmlString(value);
      }
   }


   public string mensagem; // contém o texto original ou o texto cifrado

   public UtilizacaoRSA(int abitsRSA) // Construtor com sobrecarga
   {
      tamanhoChave = abitsRSA / 8;
      tamanhoMaximoTexto = tamanhoChave - 42;

      bitsRSA = abitsRSA;

      provider = new RSACryptoServiceProvider(bitsRSA);
   }


   public string Cifrar()
   {
      try
      {
         byte[] bytes = Encoding.UTF32.GetBytes(mensagem); // vamos ter que criptografar cada byte da Mensagem

         int tamanhoArrayBytes = bytes.Length;
         int loops = tamanhoArrayBytes / tamanhoMaximoTexto;

         StringBuilder stringCifrada = new StringBuilder();

         for (int i = 0; i <= loops; i++)
         {
            byte[] temp = new byte[(tamanhoArrayBytes - tamanhoMaximoTexto * i > tamanhoMaximoTexto) ?
tamanhoMaximoTexto : tamanhoArrayBytes - tamanhoMaximoTexto * i];

            Buffer.BlockCopy(bytes, tamanhoMaximoTexto * i, temp, 0, temp.Length);
```

```
            byte[] byteCifrado = provider.Encrypt(temp, true);

            stringCifrada.Append(Convert.ToBase64String(byteCifrado));
        }

        string retorno = stringCifrada.ToString();

        if (retorno.Trim().Length == 0)
        {
            throw new Exception("Não foi possível criptografar a mensagem!");
        }


        return retorno;
    }
    catch (Exception E)
    {
        throw new Exception("Erro ao criptografar mensagem com chave de " + bitsRSA.ToString() + " bits!\n\n" +
                    "Mensagem original do erro:\n" + E.Message);
    }
}

public string Decifrar()
{
    try
    {
        int baseBloco64 = (tamanhoChave % 3 != 0) ? ((tamanhoChave / 3) * 4) + 4 : (tamanhoChave / 3) * 4;
        int loops = mensagem.Length / baseBloco64;

        ArrayList arrayDecifrado = new ArrayList();

        for (int i = 0; i < loops; i++)
        {
            byte[] encryptedBytes = Convert.FromBase64String(mensagem.Substring(baseBloco64 * i, baseBloco64));

            arrayDecifrado.AddRange(provider.Decrypt(encryptedBytes, true));
        }

        string retorno = Encoding.UTF32.GetString(arrayDecifrado.ToArray(Type.GetType("System.Byte")) as byte[]);

        if (retorno.Trim().Length == 0)
        {
            throw new Exception("Não foi possível descriptografar a mensagem!");
        }

        return retorno;
    }
    catch (Exception E)
    {
        throw new Exception("Não foi possível descriptografar mensagem.\n"+
                    "Esta mensagem não foi cifrada com a chave pública que faz par a esta chave privada!\n\n"+
                    "Mensagem original do erro:\n"+E.Message);
    }
}

public void gerarChavePublicaePrivada(ref string chavePrivada, ref string chavePublica)
{
    chavePrivada = "<RSABits>" +
                bitsRSA.ToString() +
                "</RSABits>" +
                provider.ToXmlString(true); // true identifica a inserção de conteúdo Privado

    chavePublica = "<RSABits>" +
                bitsRSA.ToString() +
                "</RSABits>" +
```

```
            provider.ToXmlString(false);
    }
}
```

BIBLIOGRAFIA

Livros:
O livro dos códigos, escrito por Simon Singh
Visual C# 2010, escrito por John Sharp

Sites pesquisados (acessados no dia 30/10/2012)
National Institute of Standards Technology
http://csrc.nist.gov/groups/ST/toolkit/index.html
Token bancário
http://silviolobo.com.br/DOMINIOPUBLICO/joomla/informatica-a-tecnologia/798-token-bancario-como-funciona

Cifra de Vigenere
http://ciencia.hsw.uol.com.br/cracker3.htm

Máquina Enigma
http://www.inf.ufsc.br/~davigp/INE-5386/Enigma/

Quem foi Alan Turing
http://pt.wikipedia.org/wiki/Alan_Turing

Marian Rejewski
http://pt.wikipedia.org/wiki/Marian_Rejewski

Phil Zimmermann e o PGP
http://pt.wikipedia.org/wiki/Phil_Zimmermann
Gato de Schrödinger
http://pt.wikipedia.org/wiki/Gato_de_Schr%C3%B6dinger

Dinheiro quântico de Stephen Wiesner:
http://cs-exhibitions.uni-klu.ac.at/index.php?id=258

Verisign
http://pt.wikipedia.org/wiki/VeriSign

Eletronic Fontier Foundation
http://pt.wikipedia.org/wiki/Electronic_Frontier_Foundation

Riindael ou AES
http://pt.wikipedia.org/wiki/AES

OpenPGP Alliance

http://www.openpgp.org/