Questão 2 - Pesquise por ferramentas que possam ser utilizadas pela linha de comando para gerar pares de chaves assimétricas. Enumere os comandos necessários para criar um par de chaves, explicando como o comando deve ser executado. (tutorial)

Ferramenta escolhida: OpenSSL

Como instalar:

- Todas as informações podem ser encontradas no GitHub [https://github.com/openssl/openssl](). Neste endereço é possível baixar o toolkit contendo libssl (implementação das versões de um protocolo chamado TLS), libcrypto (biblioteca de criptografia), e o openssl (ferramenta de linha de comando, para várias ações relacionadas a tarefas de realização, teste e análise de criptografia)
- Instruções de instalação se encontram em [https://github.com/openssl/openssl/blob/master/INSTALL.md]()
- No windows parece que há um instalador .exe já feito.
- Vai precisar de algumas coisas instaladas, como o Perl e make, ver no link acima.
- Rodar o Configure, no Mac ./Configure, no Windows, usando VSCode perlConfigure
- Vai perguntar qual tipo de configuração você vai querer
    - *Most likely you will be using the `VC-WIN64A` target for 64bit Windows binaries (AMD64) or `VC-WIN32` for 32bit Windows binaries (X86). The other two options are `VC-WIN64I` (Intel IA64, Itanium) and `VC-CE` (Windows CE) are rather uncommon nowadays.*
- Rodar o make. No Mac make, no Windows nmake
- Rodar make test. No Mac make test, no Windows nmake test
- Para instalar, tenha cuidado pois estará lidando com a pasta do sistema operacional. Rodar só esse comando com privilégios de admnistrador
    - no Mac sudo make install
    - no Windows, rodar o prompt como administrador, e rodar make install.
- Se o OpenSSL já estiver instalado, instalar em outra pasta.
    - No Mac usa-se $./Configure--prefix=/opt/openssl--openssldir=/usr/local/ssl
    - Não há informação na documentação do Git sobre como instalar em outro local no Windows, mas tenho impressão que é o mesmo.

Como gerar um par de chaves assimétricas? O que significa cada comando?

- Para essa questão, usará como exemplo o algoritmo RSA
- Obs: quando a ordem importar, é notado na descrição do comando.
- Havia, na ferramenta, o comando genrsa. Genrsa foi depreciado no OpenSSL 3.0. Para gerar uma chave privada é recomendado usar o genpkey ao invés do genrsa. O genpkey possui formato unificado PKCS#8 para gerar RSA, RSA-PSS, EC, X25519, X448, ED25519 e ED448 (https://www.misterpki.com/openssl-genrsa/)
    - *Os comandos devem ser precedidos de openssl*
- Para a chave pública, caso o arquivo seja do tipo PEM, o header vai ser -----BEGIN [ENCRYPTED] PRIVATE KEY-----

- O comando que gera a public key é específico para o rsa. Ele permite converter a chave (para outro tipo de arquivo, descriptografar etc.), também.
- No caso de gerar uma public key, o openSSL já tem ela pre-calculada e guardada no arquivo do privateKey.
- Para a chave pública, caso o arquivo seja do tipo PEM, o header vai ser -----BEGIN PUBLIC KEY-----

GERAR PRIVATE KEY (RSA)

```
openssl genpkey -algorithm RSA -out key.pem
openssl genpkey -algorithm RSA -out key.pem -aes-128-cbc -pass pass:hello
openssl genpkey -algorithm RSA -out key.pem -pkeyopt rsa_keygen_bits:2048 -pkeyopt
rsa_keygen_pubexp:3
```

genpkey		"Gerar uma chave privada"
-algorithm [Name]		"Do algoritmo [Name]" Pode ser RSA, RSA-PSS, EC, X25519, X448, ED25519 e ED448. No nosso caso Name = RSA Deve ser antes do -pkeyopt Não pode ocorrer junto de -paramfile.
-out [filename]		"Para o [filename]" O nome tanto faz, o importante vai ser a opção -outform
-outform [DER/PEM]		"Com formato [DER/PEM]" Default é PEM. Se tiver -genparam, esta opção é ignorada. Se pedir para criar um arquivo .der, e não olocar essa opção, a criação de chave seguirá o default, PEM Se pedir para criar um arquivo .pem, e colocar essa opção com DER, a criação de chave seguirá essa opção.
-aes128 **-aes192** **-aes256** **-aria128** **-aria192** **-aria56**	**-camellia128** **-camellia192** **-camellia256** **-des** **-des3** **-idea**	"Criptografada com o algoritmo..."
-pass [arg]		Ao indicar que quer criptografar, usar esse comando... "Com a passphrase [arg]" Pode usar, em conjunto, pass:, env:, file:, fd: ou stdin, para capturar de outros locais, também. Se colocar uma especificação de criptografia e não colocar esse comando, vai pedir uma passphrase logo depois. No caso de usar pass:, você não pode usar uma passphrase pequena. A passphrase deve ser digitada sempre que a key for ser usada, mas não quando for visualizada
-pkeyopt [opt:value]		"Para a opção do algoritmo de geração de chave pública, use [value]" Deve ser colocada sempre que for usar uma das opções abaixo, para cada uma.
rsa_keygen_bits:[num]		Opção 1 para o alg RSA.

	"Com a chave de tamanho [num]". A partir de 512. Default é 2048 e esse é o menor tamanho recomendado atualmente, mas leva mais tempo.
rsa_keygen_primes: [numprimes]	Opção 2 para o alg RSA. A quantidade de números primos na chave gerada. O default é 2 Na documentação diz que o máximo é 10, mas esse máximo é limitado pelo tamanho da chave gerada. No meu teste tentei com 1048576 e permitiu no máximo 5. Para 2048, 3.
rsa_keygen_pubexp: [value]	Opção 3 para o alg RSA. O valor expoente público do RSA, pode ser um large decimal ou um valor hexadecimal precedido de 0x. O default é 65537. Alguns usam o valor 3. Esse valor é aceito por razões legais, mas é depreciado. 65537 é o mais seguro.

GERAR PRIVATE KEY (RSA) - OUTRAS OPÇÕES

-help	"Printe a ajuda"
-quiet / -verbose	"Sem mostrar o status" / "Mostrando o status"
-cipher [Alg]	"Criptografada usando o algoritmo [ALG]" ALG deve seguir uma terminologia especifica pelo openSSL
-text	"Coloque um texto com os parâmetros (módulo [public key & n], expoente público [e], expoente privado e primos usados para criar a chave [p & q]) junto da chave"
-genparam	"Gere os parâmetros, ao invés de uma chave privada" Deve vir antes de -algorithm, -paramfile e -pkeyopts Não utilizada nesse caso.
-paramfile [file]	"Usando o set de parâmetros que estão nesse arquivo" Deve vir antes de -pkeyopt Não pode ocorrer junto de -algorithm. Usado por algoritmos que geram chaves privadas
-engine [id]	Opção depreciada. Era usado para indicar um hardware a ser usado
-provider [name]	Opção para fornecer um provedor para o OpenSSL implementar operações
-provider-path [path]	Local do provedor. Se for usada, deve ser usada antes de -provider
-propquery propq	Query para rodar no provedor.
-config configfile	Interno do OpenSSL, default é o arquivo *openssl.cnf*

GERAR PUBLIC KEY (RSA)

```
openssl rsa -in key.pem -pubout -out pubkey.pem
openssl rsa -in rsa.private -out rsa.public -pubout -outform PEM
```

rsa		"RSA..."
-pubout		"Gere uma chave pública no output" O default é gerar uma chave privada (converter, etc.) Caso o input seja uma chave pública (veja *-pubin*), essa opção vai ser setada automaticamente
-in [NAME/URI]		"Com o input..." Na documentação diz que se não informar ele pega um valor padrão, mas no meu teste ele ficou em loop eterno Se rodar *openssl rsa -text -in private_key.pem* aparece no prompt os parâmetros (módulo [public key & n], expoente público [e], expoente privado e primos usados para criar a chave [p & q]) da chave que veio no -in
-out [NAME]		"E gere como output..." Na documentação diz que não pode usar o mesmo nome de arquivos, mas no meu teste permitiu e sobrescreveu. Na documentação diz que se não informar ele pega um valor padrão, mas no meu teste ele ficou em loop eterno O nome tanto faz, o importante vai ser a opção -outform *Caso receba uma chave criptografada, vai descriptografar.*
-passin [arg] **-passout [arg]**		"Com a passphrase do input/output..." Pode usar, em conjunto, pass:, env:, file:, fd: ou stdin, para capturar de outros locais, também. Se colocar uma especificação de criptografia e não colocar esse comando, vai pedir uma passphrase logo depois. A passphrase deve ser digitada sempre que a key for ser usada.
-outform [DER/PEM/PVK]		"Com o formato de chave de saída informado" Default é PEM. Se pedir para criar um arquivo .der, e não olocar essa opção, a criação de chave seguirá o default, PEM Se pedir para criar um arquivo .pem, e colocar essa opção com DER, a criação de chave seguirá essa opção.
-aes128 **-aes192** **-aes256** **-aria128** **-aria192** **-aria56**	**-camellia128** **-camellia192** **-camellia256** **-des** **-des3** **-idea**	"criptografe a chave privada com..." Só pode usar se o output for do tipo PEM "Criptografada com AES 256" Opcional, mas recomendado. Vai pedir uma senha em seguida Vai adicionar mais dados ao arquivo, como "Proc-Type" e "DEK-Info"

GERAR PUBLIC KEY (RSA) – OUTRAS OPÇÕES

-help	"Printe a ajuda"
-inform [DER/PEM/P12/ENGINE]	"Com o formato de chave de entrada informado" Desde o OpensSSL 3.0, caso isso não seja informado, vai tentar com todos os formatos, automaticamente. Caso informe o formato, vai tentar somente com esse formato.
-traditional	Usar PKCS#1 ao invés de PKCS#8, para a chave privada
-pubin	"Leia uma chave pública ao invés de uma chave privada, no input"
-text	"Coloque um texto com os parâmetros (módulo [public key & n], expoente público [e], expoente privado e primos usados para criar a chave [p & q]) da chave privada, junto do arquivo gerado"
-noout	"Não mostre nenhuma mensagem quando tiver terminado"
-modulus	"Printe o valor do modulo n da chave"
-check	"Cheque a consistência da chave RSA privada"
-RSAPublicKey_in	O mesmo de -pubin, mas espera uma chave RSAPublicKey
-RSAPublicKey_out	O mesmo de -pubout, mas espera uma chave RSAPublicKey
-pvk-strong	"Com nível de encoding PVK forte" (padrão)
-pvk-weak	"Com nível de encoding PVK fraco"
-pvk-none	"Sem encoding PVK"
-engine [id]	Opção depreciada. Era usado para indicar um hardware a ser usado
-provider-path [path]	Local do provedor. Se for usada, deve ser usada antes de -provider
-provider [name]	Opção para fornecer um provedor para o OpenSSL implementar operações
-propquery [propq]	Query para rodar no provedor.

Referências

- https://github.com/openssl/openssl/blob/master/INSTALL.md
- https://www.misterpki.com/openssl-genrsa/
- https://www.scottbrady91.com/openssl/creating-rsa-keys-using-openssl
- https://en.wikibooks.org/wiki/Cryptography/Generate_a_keypair_using_OpenSSL#cite_note-5
- https://www.devco.net/archives/2006/02/13/public_-_private_key_encryption_using_openssl.php
- https://rietta.com/blog/openssl-generating-rsa-key-from-command/
- https://www.ccuec.unicamp.br/ccuec/material_apoio/comandos-uteis-openssl
- https://www.feistyduck.com/library/openssl-cookbook/online/openssl-command-line/key-generation.html#openssl-key-generation_variablelist1_varlistentry3_listitem1_para1_footnote1-fnote
- https://www.ibm.com/docs/en/secure-proxy/6.0.0?topic=certificates-certificate-formats-used-secure-proxy
- https://www.openssl.org/docs/man3.0/man1/openssl-rsa.html
- https://www.openssl.org/docs/man3.0/man1/openssl-genpkey.html
- Help, da ferramenta, no prompt de comando.