Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Propostas para Trabalhos de Grupo
Team Work Proposals

Segurança Informática
Computer Security

Departamento de Informática
Department of Computer Science
Universidade da Beira Interior
University of Beira Interior

Pedro R. M. Inácio
inacio@di.ubi.pt
2022/23

Conteúdo

1 Introdução

Introduction

As seguintes propostas de trabalho foram elaboradas no contexto da unidade curricular de Segurança Informática e servem primariamente o propósito de *exemplificar* o funcionamento de primitivas da criptografia e de aplicações que as integrem. Como tal, estas propostas não replicam necessariamente, ou em todo o detalhe, aplicações ou tecnologias atualmente em uso.

The following proposals were elaborated within the context of the subject of Computer Security and they serve primarily the purpose of exemplifying the functioning of criptography related primitives and of applications that make use of them. As such, they may not all replicate real life applications or technologies used nowadays.

Notem também que os estudantes são livres de submeter uma nova proposta, desde que esteja em concordância com os objetivos da unidade curricular. Adicionalmente, a proposta tem de ser discutida com o Professor e descrita num formato semelhante ao das demais antes de ser aceite como um tópico de trabalho válido.

Notice also that the students are free to submit a new proposal, as long as it is in accordance with the objectives of this subject. Additionally, the proposal has to be discussed with the Professor and described in a format similar to the one included herein prior to being accepted as a valid work topic.

Como de resto discutido durante as aulas, as equipas devem ter entre cinco e seis elementos, e a implementação destas propostas deve ser complementada com artefactos que provem a autoria e originalidade (e.g., mostrando evidências do uso de ferramentas de gestão do projeto de software) e com uma apresentação em que se discutem as dificuldades encontradas ao longo do trabalho e se justificam todas as escolhas tomadas. A apresentação ocorre no final do semestre, recorrendo a um pequeno conjunto de diapositivos.

As discussed in the classes, the teams should be formed by five or six students, and the implementation of these proposals should be complemented with artifacts that prove the authoring and originality (e.g., showing evidences of the usage of software management tools) and with a presentation in which the difficulties faced along the work are discussed and the choices that have been taken are justified. The presentation will take place at the end of the semester, resorting to a small set of slides.

As aplicações podem ser implementadas recorrendo a qualquer estúdio ou ambiente de desenvolvimento integrado, sendo esse pormenor deixado ao critério dos estudantes. São também livres de decidir a linguagem de programação ou as tecnologias *web* (se aplicáveis) a usar (a não ser que os meios a utilizar estejam diretamente indicados na proposta). Contudo, por favor tomem em consideração os dois apontamentos que se seguem: (i) não presumam nem estejam à espera que o Professor seja capaz de responder a todas as questões específicas às tecnologias que escolheu (por exemplo, não espere que ele saiba como trabalhar simultaneamente com o Netbeans, Eclipse, Visual Studio, etc.) e, (ii) a aplicação deve estar a funcionar corretamente no fim do semestre, independentemente das ferramentas ou tecnologias utilizadas. Alguns dos problemas que encontrarem durante o desenvolvimento destes projetos podem já ter sido tratados ou resolvidos nas aulas, e são livres de reutilizar, se aplicável, qualquer pedaço de código ou recurso desenvolvido ao longo das mesmas.

The applications can be implemented resorting to any development studio or integrated development environment of your choice. Students are free to decide the programming language or the web technologies (if applicable) they are going to use also (unless the means that should be used are directly indicated in the work proposal). Nonetheless, please take the following two remarks into consideration: (i) do not expect the Professor to be able to answer all technology specific questions (e.g., do not expect him to know how to work simultaneously with Netbeans, Eclipse, Visual Studio, etc.) and, (ii) the application should be correctly functioning at the end of the semester, independently of those choices. Some of the problems you may face during the development of these projects may have been already addressed during the practical classes, and you are free to reuse, if applicable, any code developed along the classes.

É da responsabilidade dos estudantes a pesquisa de detalhes específicos à solução dos

problemas propostos, assim como de maneiras de testar a validade dessas soluções. Para além de referirem a solução do problema na apresentação, os estudantes deve também descrever o modo como validaram o que foi feito (se aplicável) e incluir alguma teoria de como foi conseguido.
It is of the responsibility of the students to search for specific details concerning the proposal at hands and for ways to validate the solutions. Besides including the solution to the problem in the presentation, students should also describe the means used to validate the work (if applicable) and include some of the theory behind what was done.

1.1 Entrega do Trabalho
Delivery of the Works

A entrega do trabalho é feita única e exclusivamente via *moodle* até às 23:55 do dia de entrega do trabalho e os **nomes dos ficheiros devem seguir a especificação** incluída na secção dedicada a essa entrega na área da unidade curricular naquela plataforma. Por cada dia de atraso na entrega de qualquer elemento do trabalho (código de implementação, *scripts* de instalação ou outros artefactos), descontam-se 0,5 valores (aos 5).

1.2 Sugestão de Alinhamento para a Apresentação
Suggested Lineup for the Presentation

A defesa do trabalho deve ser acompanhada por um breve conjunto de diapositivos. A apresentação oral pelos elementos do grupo não pode demorar mais do que 10 minutos. Devem considerar fazer um conjunto de 10 diapositivos para guiar o discurso com o seguinte alinhamento (façam as adaptações que considerem necessárias):

- 1 diapositivo com o titulo do trabalho e elementos do grupo;
- 1 diapositivo com os objetivos do trabalho;
- 1 diapositivo dedicado ao levantamento de requisitos de segurança;
- 1 diapositivo dedicado à Engenharia do Software;
- 1 diapositivo dedicado à implementação;
- 1 diapositivo dedicado à apresentação do sistema/aplicação desenvolvida (este diapositivo é só para lembrar para fazer o *switch* para um demonstrador real ou para um video da aplicação a correr);
- 1 diapositivo a referir os testes efetuados;
- 1 diapositivo com a análise crítica;
- 1 diapositivo dedicado aos objetivos alcançados / conclusões e trabalho futuro.
- 1 diapositivo a dizer `Bem haja pela atenção. Perguntas?`

2 Propostas de Trabalho
Work Proposals

2.1 CANTTOUCHME: Um Livro de Registos Pessoal e Seguro
CANTTOUCHME: A Personal and Safe Book of Records

A ideia deste trabalho é desenvolver um programa que permita guardar registos de texto imutáveis para vários utilizadores. Embora as aplicações ou sistemas que suportam mais

do que um utilizador serem especialmente beneficiadas em arquiteturas cliente/servidor, esta pode ser pensada como uma aplicação para computador pessoal (quem quiser, pode fazer uma aplicação *web*), que eventualmente é usado por mais do que um utilizador. A aplicação deve permitir registar os utilizadores e que estes registem atividades do seu dia através da introdução de uma descrição em texto. Sempre que um utilizador entra na aplicação, este deve apenas ver um campo onde pode escrever essa descrição, que fica guardada juntamente com a data, e ter a opção de ver registos anteriores. A aplicação deve guardar todos os registos cifrados no sistema (para que ninguém os consiga ver, mesmo que abra os ficheiros com outro programa externo). A chave de cifra deve ser única por utilizador, e ser derivada da palavra-passe do utilizador. Embora se apresente como um aspeto mais avançado, recomenda-se que a aplicação guarde cada um dos registos com um código de autenticação de mensagens que o cole ao bloco anterior, ao estilo de uma *blockchain* (i.e., cada novo registo é adicionado como sendo um novo bloco da *blockchain*). Assim, cada novo bloco contém
`[valor hash do bloco anterior, data do registo, texto do registo, código de auteticação de mensagem]`.
A chave de integridade deve ser também única por utilizador, e portanto ser derivada da sua palavra-passe.

Assim, o conjunto de funcionalidades básicas a suportar pela aplicação (e que lhe definem também um pouco melhor o comportamento) são:

1. a aplicação permite que o utilizador se registe com um *e-mail* e palavra-passe, devendo ser guardada uma representação segura da palavra-passe na base de dados ou ficheiro de utilizadores;
2. a aplicação deve gerar automaticamente uma chave para cifra/decifra dos registos a partir da palavra-passe do utilizador e de um número aleatório associado a esse utilizador;
3. a aplicação deve gerar automaticamente uma chave para integridade (código de autenticação de mensagens) dos registos a partir da palavra-passe do utilizador e de um outro número aleatório associado a esse utilizador;
4. a aplicação deve permitir que um utilizador insira registos e cifrar todos os registos com a chve de cifra/decifra mencionada em cima utilizando AES-128-CBC;
5. a aplicação deve guardar códigos de autenticação de mensagens para cada registo feito pelo utilizador. A chave de integridade a usar é a que foi mencionada em cima;
6. a aplicação deve permitir que um utilizador veja os seus registos anteriores. Portanto, quando um utilizador entra na aplicação, esta deve permitir decifrar e mostrar quaisquer registos anteriores.

Podem fortalecer o trabalho e conhecimento através da implementação das seguintes funcionalidades:

1. a aplicação verifica o código de autenticação de mensagens antes de mostrar registos anteriores ao utilizador, e mostra o resultado da validação (`válido` ou `não válido`) junto com o registo;
2. a aplicação deve permitir escolher a cifra que é utilizada entre AES-128-CBC e AES-128-CTR, aquando do registo do utilizador;
3. a aplicação deve permitir escolher a função de HMAC usada entre HMAC-SHA256 e HMAC-SHA512, aquando do registo do utilizador;
4. para além de usar códigos de autenticação de mensagens para verificar se um bloco/registo foi bem decifrado, o sistema usa assinaturas digitais RSA (neste caso, o sistema tem forma de gerar um par de chaves RSA de sistema);

5. a aplicação implementa o sistema de *blockchain* mencionado na introdução.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação.

2.2 NOTHING2LOSE: Uma Lotaria que Compromete Tempo e Prémios

NOTHING2LOSE: A Lottery that Compromises Time and Prizes

A criptografia contém ferramentas excelentes para desenvolver jogos de mistério com desafios. A ideia deste trabalho é desenvolver um sistema que gere quatro bilhetes de lotaria com prémios cifrados e permita que os seus utilizadores decifrem um desses bilhetes. Os quatro bilhetes devem ter valores diferentes (e.g., prémio simples de 1 runa; prémio médio de 2 runas; prémio raro de 4 runas e prémio lendário de 8 runas), e serem incrementalmente dificeis de abrir. Por outras palavras, se um utilizador escolher um bilhete que tem um prémio maior (sem saber), este prémio deve demorar mais tempo a decifrar (por exemplo, o bilhete com o prémio simples deve demorar 30 segundos, enquanto o prémio médio deve demorar o dobro, i.e., 60 segundos; o prémio lendário deve demorar 4 minutos). O sistema a construir deve, portanto, simular tudo isto:

1. O sistema gera quatro bilhetes com diferentes prémios (os prémios devem ser aleatórios, mas com valores definidos por uma exponencial);
2. O sistema gera quatro chaves de cifra com complexidade diferente (e.g., a chave para o prémio simples tem 20 bits aleatórios + 108 bits preenchidos com 0s; a chave para o prémio médio tem 21 bits aleatórios + 107 bits preenchidos com 0s; a chave para o prémio raro tem 22 bits aleatórios + 106 bits preenchidos com 0s; e a chave para o prémio lendário tem 23 bits aleatórios + 105 bits preenchidos com 0s);
3. O sistema calcula códigos de autenticação de mensagens para os vários prémios;
4. O sistema cifra os quatro bilhetes;
5. O sistema pede ao utilizador que escolha um prémio e inicia a decifra desse prémio até devolver o resultado ao utilizador (o utilizador tem de esperar até receber o prémio ou desistir, e escolher outro).

Assim, o conjunto de funcionalidades básicas a suportar pelo sistema são:

1. registo simples de um utilizador através de um *e-mail* e palavra-passe (deve ser guardada uma representação segura da palavra-passe na base de dados);
2. o sistema gera bilhetes com diferentes prémios seguindo o esquema em cima;
3. o sistema gera chaves de cifra com diferentes complexidades (testem as complexidades para saberem quantos bits devem ter para demorarem 30, 60, 120 e 240 segundos a decifrar);
4. o sistema calcula códigos de autenticação de mensagem com HMAC-SHA256;
5. o sistema cifra os bilhetes com as chaves geradas usando AES-128-CBC;
6. o sistema permite a um utilizador registado escolher um bilhete e tenta decifrá-lo (demorando o tempo necessário), e mostrando o resultado no final;
7. o sistema permite que o utilizador desista da decifra de um bilhete (se estiver demorar muito), e que escolha outro, até que não tenham mais nenhum para escolher.

Podem fortalecer o trabalho e conhecimento através da implementação das seguintes funcionalidades:

1. o sistema deve permitir escolher a cifra que é utilizada entre AES-128-CBC e AES-128-CTR;
2. o sistema deve permitir escolher a função de HMAC usada entre HMAC-SHA256 e HMAC-SHA512;
3. para além de usar códigos de autenticação de mensagens para verificar se uma mensagem foi bem decifrada, o sistema usa assinaturas digitais RSA (neste caso, o sistema tem forma de gerar um par de chaves RSA de sistema);
4. Inventar esquemas de ajuda inteligentes para decifra (e.g., responder corretamente a uma pergunta permite reduzir o tempo de decifra por desvendar parte da chave de cifra).

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação.

2.3 ESTOU-TA-VER: um Monitor para Integridade para Diretorias

ESTOU-TA-VER: an Integrity Monitor for Directories

O objetivo principal deste projeto é desenvolver um programa que detete alterações em ficheiros dentro de uma diretoria. A ideia é fazer uso extenso de funções de *hash*, *Hash base Message Authentication Codes* (HMAC) ou assinaturas digitais, bem como cifras, para construir bases de dados de valores resumo, códigos de autenticação ou assinaturas digitais de ficheiros de uma diretoria à escolha do utilizador. Esta base de dados (que pode ser um ficheiro) deve ser feita na primeira vez que o programa é executado para uma diretoria e deve conter o nome de todos os ficheiros da diretoria, bem como o resumo, código ou assinatura digital que os representa. A base de dados deve ser sempre guardada cifrada com uma chave de cifra derivada de uma palavra-passe do utilizador. Sempre que o utilizador quiser, o programa volta a pedir a palavra-passe, decifra a base de dados e verifica se houve alterações em alguns dos ficheiros comparando valores de *hash* ou verificando HMACs ou assinaturas digitais. Caso sejam detetadas alterações, estas devem notificadas ao utilizador e deve ser feita uma nova base de dados da diretoria (guardando a base de dados anterior).

O conjunto de funcionalidades básicas a incluir são assim as seguintes:

- a aplicação funciona apenas quando é executada pelo utilizador (execução isolada);
- a aplicação calcula valores de *hash Secure Hash Algorithm 256* (SHA256) para todos os ficheiros e guarda-os numa base de dados/ficheiro juntamente com o nome dos ficheiros;
- a aplicação cifra o ficheiro da base de dados com a cifra *Advanced Encryption Standard* no modo *Cipher Block Chaining* e chaves de 128 bits (AES-128-CBC). A chave de cifra é derivada de uma palavra-passe usando um algoritmo de derivação de chaves de cifra seguro (e.g., *Password Based Key Derivation Function 2* (PBKDF2)), junto com um *salt*. A chave de cifra ou palavra-passe nunca são guardadas de forma persistente no disco e a base de dados só é decifrada para a memória (i.e., nunca é guardada no disco em texto texto-limpo);
- a aplicação guarda HMAC-SHA512 juntamente com os valores de *hash* e nomes de ficheiros na base de dados. A chave de integridade para cálculo de HMACs é derivada da mesma palavra-passe dada pelo utilizador (ver ponto anterior) através de uma função segura de derivação de chaves, mas usando um valor de *salt* diferente;
- quando executada pela primeira vez, a aplicação calcula os valores de *hash* e HMAC e guarda-os na base de dados; nas vezes seguintes, a aplicação faz verificação de integridade para cada ficheiro e avisa o utilizador acerca de eventuais alterações;

- caso sejam detetadas alterações, é criada e guardada uma nova base de dados atualizada, sendo a chave de cifra derivada da mesma palavra-passe mas usando um valor de *salt* diferente.

Como forma de fortalecer o trabalho, podem considerar o seguinte:

- a aplicação funciona como um serviço em segundo plano, detetando alterações em tempo-real (execução *daemon*);
- a aplicação faz uso de assinaturas digitais (e.g., *Rivest, Shamir e Adleman* (RSA)) em vez de HMACs;
- a chave privada (do par RSA para assinaturas digitais) é guardada de modo seguro na diretoria a monitorizar (i.e., é guardada cifrada);
- ter um *help* bastante completo.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação.

2.4 XIUUU: Troca de Segredos Criptográficos Seguro

XIUUU: Safe Sharing of Cryptographic Secrets

O objetivo principal deste trabalho é implementar um sistema que permita trocar segredos criptográficos (só as chaves) entre duas entidades de uma forma fiável e segura. A ideia é que o sistema integre e disponibilize uma série de esquemas de distribuição e protocolos de acordo de chaves criptográficas, e também formas de as gerar a partir de palavras-passe. Em princípio, o sistema deve poder ser concretizado numa única aplicação que, depois de executada em dois computadores ou dispositivos distintos, permita a geração e troca de segredos entre ambas as instâncias. Entre outras a pensar, o sistema deve fornecer as seguintes funcionalidades base:

- geração de um segredo criptográfico a partir de palavras-passe inseridas pelo utilizador, nomeadamente através de algoritmos como o *Password Based Key Derivation Function 2* (PBKDF2);
- troca de um segredo criptográfico usando o protocolo de acordo de chaves Diffie-Hellman;
- troca de um segredo criptográfico usando Puzzles de Merkle;
- troca de um segredo criptográfico usando o *Rivest, Shamir e Adleman* (RSA);
- distribuição de novas chaves de cifra a partir de chaves pré-distribuídas;
- distribuição de novas chaves de cifra usando um agente de confiança (neste caso, a aplicação desenvolvida deve permitir que uma das instâncias possa ser configurada como agente de confiança);
- implementar forma de ter a certeza de que o segredo partilhado é o mesmo dos dois lados.

A aplicação desenvolvida pode funcionar em modo *Client Line Interface* (CLI) ou fornecer uma *Graphical User Interface* (GUI). Eventualmente, este sistema pode ser implementado para dispositivos móveis, nomeadamente para a plataforma Android. Conforme já sugerido em cima, a aplicação deve poder funcionar em modo cliente ou servidor sendo que, idealmente, deve deixar que seja o utilizador a escolher o modo sempre que é iniciada. Caso o modo escolhido seja o de servidor, deve ser então pedida o número da porta em que vai ficar à escuta; caso contrário, deve ser pedido o endereço *Internet Protocol* (IP)

e porta do destino. Uma aplicação que esteja a funcionar como servidor deve ser capaz de fornecer uma lista de utilizadores disponíveis e facultar uma forma de se iniciarem ligações dedicadas entre quaisquer dois utilizadores para posterior troca de segredos. O trabalho e conhecimento podem ser fortalecidos através da implementação das seguintes funcionalidades:

- usar certificados digitais X.509 nas trocas de segredos que recorrem ao RSA;
- implementar uma infraestrutura de chave pública para o sistema e validar cadeias de certificados nas trocas de segredos que recorrem ao RSA (e.g., definir um certificado raiz para o sistema e que já vem embutido no código ou com a aplicação, gerando depois certificados digitais para cada um dos utilizadores do sistema);
- pensar numa forma correta de fornecer certificados digitais a utilizadores;
- implementar mecanismos de assinatura digital para verificação de integridade em trocas de chave efémeras usando o Diffie-Hellman;
- possibilitar a escolha de diferentes algoritmos de cifra para os Puzzles de Merkle;
- possibilitar a escolha de diferentes funções de *hash* para o PBKDF2;
- ter um *help* bastante completo e ser de simples utilização.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação. Notem que, para efeitos de avaliação e prototipagem, o sistema desenvolvido pode executar localmente todos os seus componentes/aplicações/programas, desde que simule ou concretize a arquitetura sugerida (i.e., não precisa necessariamente executar em rede).

2.5 iCHATO: Aplicação para Conversação Online com Mensagens Assinadas

iCHATO: Application for Online Chatting with Signed Messages

Nota: este tema só está disponível para Engenharia Informática.

Com este projeto pretende-se que o grupo de trabalho desenvolva uma aplicação para conversasão *online* (vulgarmente conhecida como *chat*) tipo *Internet Relay Chat* (IRC), mas cujas mensagens são assinadas e, opcionalmente, cifradas. A plataforma desenvolvida deve suportar as seguintes funcionalidades:

- registo de novos utilizadores (neste caso, a plataforma deve gerar um par de chaves pública e privada automaticamente e atribui-las ao utilizador de modo seguro);
- permitir que os utilizadores falem todos uns com os outros num canal público (neste caso, as mensagens só seguem assinadas, não cifradas);
- permitir que dois utilizadores falem um com o outro diretamente através de um canal dedicado, com mensagens cifradas e assinadas (as chaves de sessão podem ser trocadas usando criptografia assimétrica ou o protocolo de acordo de chaves Diffie-Hellman).

Notem que este projeto pressupõe o desenvolvimento de duas aplicações diferentes: uma que atua como cliente; e outra que atua como servidor. A aplicação cliente é a que serve de interface para o sistema, e a que permite que um utilizador se registe da primeira vez que se liga à plataforma. É o servidor que guia o processo de registo, guardando o `nome de utilizador`, a `palavra-chave` e algumas informações do utilizador. É também o servidor que gera e envia o par de chaves pública/privada (e o certificado, se aplicável - ver funcionalidade extra em baixo) para o cliente. Notem que devem tomar as devidas precauções para que o referido envio seja feito de modo seguro (i.e. trocando chaves de

sessão e cifrando as chaves ou certificados durante este envio). As chaves públicas (ou certificados) podem ser guardadas no servidor e distribuídas a quem as requisitar durante o funcionamento da aplicação, mas as chaves privadas devem ficar somente no lado do cliente a quem pertencem.

O utilizador registado acede ao sistema via introdução do seu `nome de utilizador` e da `palavra-passe`, e após validado, passa a poder escrever no canal público do sistema. Caso deseje falar com alguém em particular, deve ser gerada e trocada uma chave de sessão, usada para garantir a confidencialidade do canal assim criado.

Podem fortalecer o trabalho e conhecimento através da implementação das seguintes funcionalidades:

- guardar as palavras-chave de uma forma segura;
- permitir que mais do que dois utilizadores falem de modo privado com mensagens cifradas;
- permitir que os programas cliente utilizem certificados digitais para validação das assinaturas digitais;
- a implementação da funcionalidade anterior implica que a aplicação servidor deve adicionalmente facilitar a criação do certificado digital aquando da geração do par de chaves pública/privada que ocorre durante o registo de um utilizador.
- escrever um *help* bastante completo.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação. Notem que, para efeitos de avaliação e prototipagem, o sistema desenvolvido pode executar localmente todos os seus componentes/aplicações/programas, desde que simule ou concretize a arquitetura sugerida (i.e., não precisa necessariamente executar em rede).

2.6 CHALLENGE-ACCEPTED: Um Sistema de Publicação de Desafios Criptográficos

CHALLENGE-ACCEPTED: System to Publish Cryptographic Challenges

A criptografia contém ferramentas excelentes para desenvolver jogos de mistério com desafios. A ideia deste trabalho é desenvolver um sistema que permita que diferentes utilizadores publiquem e resolvam desafios. Os desafios podem ser mensagens cifradas com diferentes cifras ou encontrar determinados valores de *hash*. O sistema deve permitir que os utilizadores se registem (todos os utilizadores são iguais) e deixem desafios criptográficos para que outros utilizadores registados os possam resolver. Só deve existir um tipo de utilizador (i.e., um Utilizador pode simultaneamente publicar ou resolver desafios). Devem existir dois tipos básicos de desafios (embora a equipa que desenvolve o trabalho possa definir mais):

1. Desafios de decifra de mensagens, em que é preciso adivinhar uma palavra-passe para decifrar um ficheiro;
2. Desafios de descoberta de mensagens para os quais são conhecidos os valores de *hash*.

Assim, o conjunto de funcionalidades básicas a suportar pelo sistema são:

1. registo simples de um utilizador através de um *e-mail* e palavra-passe (deve ser guardada uma representação segura da palavra-passe na base de dados);

2. cada utilizador deve poder colocar no sistema desafios do tipo cifra de mensagens. Neste caso, para além da mensagem, o sistema deve pedir uma palavra-passe, derivar uma chave de cifra, e cifrar essa mensagem, codificando-a em BASE64;

3. para que a funcionalidade anterior resulte, o sistema deve também calcular um código de autenticação de mensagens (com uma chave secreta colocada pelo administrador do sistema) que permite verificar se uma mensagem foi bem decifrada ou não, quando uma solução é submetida;

4. cada utilizador deve poder colocar no sistema desafios do tipo valor de *hash*. Neste caso, o sistema deve pedir apenas a mensagem, calcular o valor de *hash* e publicar esse valor em hexadecimal ou BASE64;

5. cada utilizador deve poder escolher um desafio (diferente dos que ali tiver colocado) e submeter uma solução a esse desafio. O sistema deve dizer se a solução submetida está correta ou não. No caso das mensagens cifradas, deve também mostrar a mensagem;

6. o sistema deve permitir fazer desafios com as cifras AES-128-ECB, AES-128-CBC e AES-128-CTR;

7. o sistema deve permitir fazer desafios com as funções de *hash* MD5, SHA256 e SHA512.

Podem fortalecer o trabalho e conhecimento através da implementação das seguintes funcionalidades:

1. o sistema não deve permitir mais do que uma tentativa a cada 15 segundos para os desafios cifrados (i.e., o processo de derivação de chaves de cifra a partir da palavra-passe deve ser propositadamente lento, para que a decifra seja propositadamente lenta também – ver projeto anterior);

2. para além de usar códigos de autenticação de mensagens para verificar se uma mensagem foi bem decifrada, o sistema usa assinaturas digitais RSA (neste caso, o sistema tem forma de gerar um par de chaves RSA para o administrador do sistema);

3. para além de suportar as cifras referidas nos pontos básicos, o sistema suporta ainda a ElGamal;

4. o sistema contém outros tipos de desafios criptográficos.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação. Notem que, para efeitos de avaliação e prototipagem, o sistema desenvolvido pode executar localmente todos os seus componentes/aplicações/programas, desde que simule ou concretize a arquitetura sugerida (i.e., não precisa necessariamente executar em rede).

2.7 TWO-HEADED-BEAST: Aplicação para Transmitir Ficheiros entre Dois Utilizadores

TWO-HEADED-BEAST: Application to Transmit Files Between Two Users

O objectivo primordial deste projeto é o de construir uma aplicação que permita transmitir ficheiros potencialmente grandes entre dois utilizadores, de um modo seguro. Entre outras, encontram-se as seguintes funcionalidades básicas da solução:

- para ficheiros maiores que 12Mb, o programa deve repartir o ficheiro e operar sobre estes blocos; é claro que é da responsabilidade do programa garantir que a integridade de *cada* bloco é assegurada;

- caso o utilizador escolha não cifrar o ficheiro a transmitir, este deve ser acompanhado de um código de autenticação da mensagem;
- as chaves de sessão devem ser trocadas utilizando o protocolo de acordo de chaves Diffie-Hellman ou através de criptografia de chave pública (pode recorrer ao `OpenSSL` para esta funcionalidade específica);
- o utilizador deve poder optar por comprimir o ficheiro antes de o cifrar.

Note que, a definição deste projeto presume que dois programas são colocados a correr em ambos os lados da comunicação, e que um deles vai ficar à espera que o outro se ligue a ele. Logo que se liguem, acorda-se a chave de sessão, o emissor cifra o ficheiro e envia para o segundo. É da responsabilidade do emissor a partição do ficheiro em blocos (se aplicável), a compressão, a cifragem e a aplicação dos mecanismos de integridade, garantia de autenticação da mensagem ou assinatura digital (ver abaixo). É da responsabilidade do receptor a decifragem, decompressão e montagem do ficheiro recebido, bem como a verificação de que o ficheiro está integro e veio de fonte segura. Caso algo corra mal (por exemplo, o ficheiro não passa na verificação de integridade), deve ser exibido um erro em ambos os lados da comunicação.

Podem fortalecer o trabalho e conhecimento através da implementação das seguintes funcionalidades:

- possibilitar a anexação da assinatura digital ao ficheiro através da especificação do ficheiro com a chave privada e recorrendo, como opção, ao `OpenSSL`.
- permitir validar a assinatura digital de um ficheiro recorrendo a uma chave pública (supostamente de quem assinou) ou de um certificado digital (podem recorrer ao `OpenSSL` para esta tarefa).
- permitir que o utilizador escolha a cifra a utilizar e o comprimento da chave de cifra;
- permitir que o utilizador escolha a função de *hash* a usar;
- ter um *help* bastante completo.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação.

2.8 SHARESAFE: Aplicação para Distribuir Ficheiros entre um Grupo de Utilizadores

SHARESAFE: Application to Share Files in a Group of Users

A ideia princípal no âmago deste projeto é a de criar uma plataforma para distribuir ficheiros dentro de um grupo restrito de utilizadores (por exemplo, dentro de uma organização ou grupo de trabalho) de uma forma segura. Na sua forma básica, a plataforma deve permitir que:

- um utilizador faça o seu registo no sistema (com nome de utilizador e palavra-passe) e receba uma chave (de um modo seguro) para uso oportuno;
- que um utilizador faça a transmissão segura (cifra), correta e autenticada (código de autenticação de mensagens) de um ficheiro a todos os utilizadores;
- que um utilizador faça a transmissão segura (cifra), correta e autenticada (código de autenticação de mensagens) de um ficheiro a um subconjunto dos utilizadores da aplicação, mediante escolha dos destinatários desse ficheiro.

Dado a especificidade do projeto em questão, sugere-se que se implemente um programa que (i) atua como servidor e que guarda as informações de utilizadores registados, (ii) gera chaves de sessão e coordena a sua troca antes da transmissão de um ficheiro, e distribui o ficheiro cifrado a todos os utilizadores do grupo restrito (depois do utilizador original ter feito o *upload* do ficheiro cifrado para o servidor). Os programas clientes são aqueles que permitem que o utilizador se autentique no sistema (mediante apresentação do nome de utilizador e palavra-chave), e que transmita ficheiros para os utilizadores que especificar. No modo de envio, o programa cliente é aquele que cifra e protege a integridade das mensagens; no modo de receção, o programa cliente é aquele que decifra as mensagens e verifica a sua integridade.

Podem fortalecer o trabalho e conhecimento através da implementação das seguintes funcionalidades:

- guardar as palavras-chave de uma forma segura;
- implementar o Diffie-Hellman de grupo para estabelecimento de chaves entre grupos de utilizadores;
- possibilitar a anexação da assinatura digital ao ficheiro através da especificação do ficheiro com a chave privada e recorrendo, como opção, ao `OpenSSL`;
- permitir validar a assinatura digital de um ficheiro recorrendo a uma chave pública (supostamente de quem assinou) ou de um certificado digital (podem recorrer ao `OpenSSL` para esta tarefa);
- permitir que o utilizador escolha a cifra a utilizar e o comprimento da chave de cifra;
- permitir que o utilizador escolha a função de *hash* a usar;
- testar as implementações utilizadas usando programas já devidamente validados (e.g., `OpenSSL`);
- ter um *help* bastante completo.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação. Notem que, para efeitos de avaliação e prototipagem, o sistema desenvolvido pode executar localmente todos os seus componentes/aplicações/programas, desde que simule ou concretize a arquitetura sugerida (i.e., não precisa necessariamente executar em rede).

2.9 FALL-INTO-OBLIVION: Um Estranho Cesto do Lixo

FALL-INTO-OBLIVION: A Strange Recycle Bin

O objetivo principal deste projeto é construir uma aplicação que simule a funcionalidade do *Recycle Bin* dos sistemas operativos modernos, mas de uma forma alternativa pouco convencional. Na verdade, a aplicação a desenvolver deve estar constantemente a monitorizar uma pasta chamada `FALL-INTO-OBLIVION`, e cifrar automaticamente todos os ficheiros que aí forem colocados. Deve também calcular um valor resumo, um *Message Authentication Code* (MAC) ou uma assinatura digital do ficheiro. A chave usada para cifrar o ficheiro e calcular o MAC deve ser gerada automaticamente para cada ficheiro, mas derivada de um *Personal Identification Number* (PIN) de 3 ou 4 dígitos. Se um utilizador desejar reaver o seu ficheiro mais tarde, tem de adivinhar o código com que foi cifrado e autenticado. Tem 3 hipóteses para conseguir decifrar o ficheiro. Depois dessas 3 tentativas, o ficheiro deve ser eliminado do sistema operativo.

De uma forma geral, pode-se dizer que a aplicação a desenvolver deve:

- permitir cifrar ficheiros, guardando o resultado numa pasta chamada `FALL-INTO-OBLIVION`;

- calcular o valor de *hash* do ficheiro, guardando também o resultado junto com o criptograma (em ficheiros separados);
- gerar automaticamente um PIN, e usá-lo como chave para cifrar cada ficheiro;
- calcular o MAC dos criptogramas;
- permitir decifrar o ficheiro por via da adivinhação do PIN. Só devem ser permitidas até 3 tentativas;
- verificar a integridade do ficheiro no caso do PIN ter sido adivinhado.

A aplicação pode correr em modo *Client Line Interface* (CLI) ou em modo gráfico (fica ao critério dos executantes). Devem usar cifras e mecanismos de autenticação de mensagens de qualidade (e.g., *Advanced Encryption Standard* em modo *Cipher Block Chainign* (AES-CBC) e *Hash* MAC *Secure Hash Algorithm 256* (HMAC-SHA256)). Podem fortalecer o trabalho e solidificar o conhecimento através da implementação das seguintes funcionalidades:

- substituir os MACs por assinaturas digitais (o programa deve então também permitir gerar as chaves pública e privadas);
- permitir que o utilizador escolha a cifra a utilizar e o comprimento da chave de cifra;
- permitir que o utilizador escolha a função de *hash* a usar;
- ter um *help* completo e intuitivo.

Uma versão muito básica deste trabalho pode utilizar chamadas ao sistema (comandos OpenSSL). A forma ideal de implementar o trabalho passar por integrar os mecanismos criptográficos na própria aplicação. Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação.

2.10 MON-AMOUR: Um Sistema para Trocar Cartas de Amor, mas com um Senão...

MON-AMOUR: System to Exchange Love Letters, with a Twist...

Algumas cifras simples ou formas de esconder mensagens ou significados estão ligadas a episódios de trocas de segredos românticos. A ideia deste trabalho é fazer um sistema para cifrar e decifrar mensagens, eventualmente com a índole referida, mas com um pequeno senão: a cifra de uma mensagem deve ser feita a partir de uma palavra-passe (e.g., *"azul"*) e esta deve garantir que as tentativas de decifra do respetivo criptograma demoram sempre, no mínimo, 15 segundos (para criar *suspense*). Para se conseguir o efeito desejado, o sistema deve ser constituído por dois programas (um para cifra, outro para decifra). O programa para cifrar deve:

1. pedir ao utilizador para colocar uma pergunta para a qual o destinatário da mensagem deve conhecer a resposta (e.g., *"Qual a minha cor favorita?"*);
2. pedir ao utilizador para escrever a resposta à pergunta (e.g., *"azul"*), colocando todas as letras da resposta em minúsculas;
3. pedir ao utilizador para escrever a mensagem;
4. gerar um número aleatório n com 128 bits;
5. juntar a palavra-passe e o número aleatório, e calcular repetidamente o valor de *Secure Hash Algorithm 256* (SHA256) dessa junção e valores de *hash*, parando quando o tempo de execução for de 15 segundos ou mais, guardando o resultado final e o número de vezes que calculou o SHA256.

No final, o programa para cifrar deve cifrar a mensagem com a *Advanced Encryption Standard* (AES), num modo conveniente e usando como chave o valor de *hash* resultante do último passo. Finalmente, deve devolver um ficheiro com a seguinte estrutura:

`número de iterações` *hash* `| Pergunta | Número aleatório | Criptograma` .

Por sua vez, programa para decifrar deve:

1. Pedir o nome do ficheiro onde está o criptograma (ou este deve ser-lhe dado como parâmetro);
2. Mostrar a pergunta ao utilizador e pedir a resposta;
3. Tentar decifrar o criptograma, voltando a calcular o valor de *hash* iterativamente com os parâmetros fornecidos (i.e., número de iterações necessárias, palavra-passe e número aleatório).

Notem que o *senão* é o facto da decifra demorar algum tempo e ser inerentemente resistente a ataques de dicionário.

Assim, o conjunto de funcionalidades básicas a suportar pelo sistema são:

- permitir cifrar uma mensagem seguindo o esquema referido antes;
- permitir decifrar o criptograma após correta inserção da resposta à pergunta;
- inclusão de um *Hash based Message Authentication Code* (HMAC) (para além da estrutura acima referida) aquando da cifra de uma mensagem;
- verificação automática da integridade da decifra através do HMAC.

Podem fortalecer o trabalho e conhecimento através da implementação das seguintes funcionalidades:

- o programa de cifra permite a um utilizador gerar ou importar um par de chaves *Rivest, Shamir e Adleman* (RSA);
- para além do HMAC, é calculada e incluída uma assinatura digital aquando da cifra de uma mensagem (neste caso, a chave pública de verificação também pode ser incluída na estrutura do ficheiro produzido);
- aquando de uma decifra bem sucedida, é também verificada a assinatura digital da mensagem;
- as chaves privadas RSA são geradas e guardadas de forma segura imediatamente (cifradas com uma cifra de chave simétrica);
- considerar desenvolver um esquema de confiança para as chaves públicas, eventualmente através do desenvolvimento de um programa que simule um agente de confiança que gera chaves RSA e as certifica;
- escrever um *help* bastante completo.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe um pequeno intervalo de tempo na apresentação. Notem que, para efeitos de avaliação e prototipagem, o sistema desenvolvido pode executar localmente todos os seus componentes/aplicações/programas, desde que simule ou concretize a arquitetura sugerida (i.e., não precisa necessariamente executar em rede).

2.11 `openguissl`: Uma Janela para o `openssl`

`openguissl`: a Window for `openssl`

O objetivo principal deste projeto é construir uma aplicação (e.g., Java; C; ou Python+XML) que sirva como interface gráfica para o programa de linha de comandos `openssl`. O trabalho pressupõe que a parte da programação será sobretudo focada na criação de um ambiente gráfico integrado e fácil de usar que permita a um utilizador escolher gradualmente o que quer fazer com a ferramenta (e.g., se quer cifrar um ficheiro, criar chaves de cifra RSA, assinar digitalmente um documento, etc.), para depois configurar parâmetros específicos da funcionalidade escolhida (como o nome do ficheiro a cifrar e daquele que irá guardar o criptograma, a chave de cifra, etc.). Também pressupõe que este ambiente gráfico irá fazer chamadas ao sistema, montando o comando textttopenssl respetivo. Os mecanismos criptográficos e funcionalidades a integrar são os seguintes:

- A aplicação deve começar com um ecrã inicial onde mostra todas as funcionalidades suportadas numa grelha;
- A aplicação tem uma secção para cifra de ficheiros com cifras de chaves simétricas (pedindo nome do ficheiro de entrada, saída e chave de cifra, bem como o algoritmo a usar);
- A aplicação tem uma secção dedicada à geração de palavras-passe aleatórias seguras;
- A aplicação tem uma secção dedicada ao cálculo de valores de *hash* (pedindo o nome do ficheiro de entrada e o nome do algoritmo a usar);
- A aplicação tem uma secção dedicada ao cálculo de HMACs (pedindo o nome do ficheiro de entrada, o nome do algoritmo a usar e a chave em hexadecimal);
- A aplicação tem uma secção dedicada à geração de chaves RSA (pedindo o nome do ficheiro de saída);
- A aplicação tem uma secção dedicada ao cálculo de assinaturas digitais (pedindo o nome do ficheiro de entrada, a chave privada e o nome do ficheiro onde a assinatura será guardada).

Como forma de fortalecer o trabalho, podem considerar o seguinte:

- A aplicação tem uma secção dedicada à verificação de assinaturas digitais (pedindo o nome do ficheiro de entrada, a chave pública e o nome do ficheiro onde a assinatura está guardada);
- A aplicação tem uma secção dedicada à criação de certificados digitais e gestão de uma infraestrutura de chave pública, nomeadamente certificados auto-assinados e assinados por uma entidade de confiança.