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
2018/19

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.

Note também que os estudantes são livres de submeter uma nova proposta, desde que esteja em concordância com os objetivos da cadeira. Adicionalmente, a proposta deverá ser discutida com o Professor 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 prior to being accepted as a valid work topic.

Como de resto discutido durante as aulas, as equipas devem ter entre três e quatro elementos, e a implementação destas propostas deve ser complementada com um relatório em que se discutem as dificuldades encontradas ao longo do trabalho e se justificam todas as escolhas tomadas. Os trabalhos serão discutidos oralmente no final do semestre, recorrendo a um pequeno conjunto de diapositivos.

As discussed in the classes, the teams should be formed by three or four students, and the implementation of these proposals should be complemented with a report in which the difficulties faced along the work are discussed and the choices that have been taken are justified. The works will be orally discussed 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 do estudante. É também livre de decidir a linguagem de programação ou a tecnologia *web* (se aplicáveis) que quer usar (a não ser que os meios a utilizar estejam diretamente indicados na proposta). Só terão de ser levados em conta os dois apontamentos que se seguem: (i) não presuma nem esteja 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 encontrar durante o desenvolvimento destes projetos podem já ter sido tratados ou resolvidos nas aulas, e é livre 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. The student is free to decide the programming language or the web technology (if applicable) he or she is going to use also (unless the means that should be used are directly indicated in the work proposal). Nonetheless, you have to take the following two remarks into consideration: (i) do not expect the Professor to be able to answer to 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. 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 do(a) aluno(a) a pesquisa de detalhes específicos à solução dos problemas propostos, assim como de maneiras de testar a validade dessas soluções. Para além de incluir a solução do problema no relatório, o aluno deve também descrever o modo como validou o que fez (se aplicável) e incluir alguma teoria de como foi feito.

It is of the responsibility of the student to search for specific details concerning the proposal at hand and for ways to validate the solutions. Besides including the solution to the problem in the report, he

or she should also describe the means used to validate his or her work (if applicable) and include some of the theory behind what was done.

1.1 Estrutura do Relatório
Structure for the Report

De modo a facilitar a estruturação do documento técnico que deve acompanhar o código e a aplicação desenvolvida no âmbito do trabalho a desenvolver, fica aqui uma sugestão para a estrutura do relatório. O relatório pode ter mais capítulos ou secções para além das que são aqui indicadas.

1. Resumo
   Que é constituído por 2 frases onde introduzem o tema.
   2 frases (máximo) onde dizem como abordaram o tema.
   1 ou 2 frases onde dizem os objetivos ou resultados alcançados.
2. Introdução
   (a) Descrição do Problema
   Pequena secção onde descrevem o trabalho por palavras vossas. Não copiem o enunciado.
   (b) Constituição do Grupo
   (c) Organização do documento
   Exemplo:
   Este relatório está dividido em 4 capítulos principais:
   - O primeiro capítulo (Introdução) descreve o problema a tratar e os objetivos mais importantes a alcançar...
   - O segundo capítulo (Desenvolvimento) ....
   - ...
   - ...
3. Engenharia de Software
   No início de cada capítulo deve ser dito como o capítulo está estruturado.
   (a) Ferramentas e Tecnologias Utilizadas
   (b) Casos de Uso
   (c) Outros Diagramas
   (d) Conclusão
4. Implementação
   (a) Escolhas de Implementação
   Explicar porque é que escolheram fazer de uma maneira, e não de outra.
   (b) Detalhes de Implementação
   (c) Manual de Utilização
   Secção simples a indicar como se compila (se aplicável) ou como se usa o sistema implementado.
   (d) Conclusão
5. Reflexão Crítica e Problemas Encontrados
   Este capítulo...
   (a) Objetivos Propostos vs. Alcançados
   Descrever os objetivos que eram propostos e quais os que foram alcançados com exatidão.
   (b) Divisão de Trabalho pelos Elementos do Grupo
   Indicar as tarefas que cada membro do grupo fez.

(c) Problemas Encontrados
Problemas encontrados e resolvidos (ou não) durante a implementação.

(d) Reflexão Crítica Falar dos problemas de segurança do trabalho. I.e., fazer uma análise do ponto de vista de segurança, não sobre o que acham que devia ser melhorado na cadeira.

(e) Conclusão

6. Conclusões e Trabalho Futuro

   (a) Conclusões Principais
   Texto muito analítico onde descrevem o que de melhor tiram deste trabalho. Não vale a pena inventar lenga-lengas sobre segurança que possam demonstrar que não percebem do assunto. Sejam analíticos e sucintos.

   (b) Trabalho Futuro
   O que ficou por implementar.

1.2 Notas a ter em Conta na Elaboração de um Bom Relatório
Remarks Concerning the Elaboration of the Report

1. Comecem sempre por incluir uma introdução onde descrevam o problema a tratar, o contexto e a estrutura do documento.
2. Elaborem bem no esqueleto do documento (secções, subsecções, etc.). Uma boa estruturação do relatório é 90% do caminho para obter um bom trabalho.
3. Caso tenham efetuado trabalho de pesquisa, incluam no relatório todas as referências.
4. Sejam breves e sucintos, mas elaborem nos detalhes que acharam importantes.
5. Procurem resolver todos os problemas que enfrentarem no tempo que possuem. Caso tal se demonstre impossível, discutam o falhanço com o mesmo afinco que discutiriam o sucesso.
6. Procurem incluir formas que ilustrem melhor o trabalho, nomeadamente gráficos, figuras ou tabelas.
7. Sejam pontual. Um relatório entregue depois do prazo não vale nada.
8. Implementem mecanismos que tenham aprendido nas aulas, só para mostrar que estudaram a matéria ou estiveram presentes.
9. Faça documentos com qualidade. Preste atenção às regras da Língua em que escreve, nomeadamente pausas, sintaxe e semântica.

A folha de cálculo em `http://www.di.ubi.pt/~inacio/projeto/req-relatorio-v10.ods` também pode conter algumas dicas interessantes para a elaboração do relatório. Responder `SIM` a cada um dos critérios aí mencionados é meio caminho andado para uma boa nota na parte que se refere ao relatório.

1.3 Entrega do Trabalho
Delivery of the Works

Cada grupo deve entregar uma versão digital e outra impressa do relatório. Materiais adicionais, como ficheiros relativos a implementações de programas, podem ser entregues juntamente com a versão digital do trabalho. Os trabalhos devem ser submetidos usando a plataforma *moodle* até às 23:55 do dia de entrega do trabalho e os nomes dos ficheiros deve seguir a especificação incluída na secção respetiva da unidade curricular também no *moodle*. O relatório impresso deve ser entregue em mão ao docente da cadeira, ou

deixado na sua caixa do correio / cacifo no máximo até às 12:00 do dia útil seguinte ao estipulado para a entrega do trabalho. Contudo, a versão digital tem de ser submetida até às 23:55 do dia da entrega. Por cada dia de atraso na entrega de qualquer elemento do trabalho (relatório ou aplicação), descontam-se 0,5 valores (aos 6).

2 Estou-a-Ver: um Monitor para Integridade para Diretorias

Estou-a-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 resusmo, 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 assinaládas 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* SHA256 para todos os ficheiros e guarda-os numa base de dados/ficheiro juntamento com o nome dos ficheiros;
- A aplicação cifra o ficheiro da base de dados com a cifra 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 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.

3 `openguissl`: Uma janela para o `openssl`

`openguissl`: a window for `openssl`

O objetivo principal deste projeto é construir uma aplicação (e.g., Web local usando HTML5+CSS3+Javascript; 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.

4 CTRL+C: Um Gestor de *Clipboard* Porreirinho

CTRL+C: An Extremely Nice Clipboard Manager

O objetivo principal deste projeto é desenvolver um gestor do *Clipboard* com alguma segurança embutida. Estes gestores são programas que permitem guardar todo o histórico da utilização do *Clipboard*, e até aceder às entradas anteriores com bastante facilidade, sendo esta uma funcionalidade de grande utilidade e uma das principais motivações para a sua existência. O problema é que alguns (senão todos os) gestores de *Clipboard* guardam o histórico num ficheiro de texto em texto-limpo, inclusive palavras-passe que possam estar a ser copiadas de um gestor de palavras-passe para um formulário online, ficando assim

expostas a cavalos de tróia ou até a utilizadores concorrentes do sistema. Pretende-se que o programa desenvolvido no contexto deste projeto resolva esse problema e adicione valor através do suporte das seguintes funcionalidades:

- o histórico do *Clipboard* deve ser guardado cifrado com uma cifra (e modo de cifra) de qualidade reconhecida (e.g., AES);
- a chave de cifra deve ser gerada aquando da execução do gestor e guardada apenas em memória. A geração da chave de cifra deve ser feita de forma segura;
- o histórico deve ser salvo no disco (cifrado) a cada 5 minutos (apagando o anterior);
- devem ser criados valores de *hash* de cada entrada no histórico, que devem ser guardados em texto limpo num ficheiro à parte;
- deve ser possível verificar se determinada entrada existe no último ficheiro do histórico cifrado, sem o decifrar, e recorrendo apenas aos valores de *hash*;
- deve ser feita uma assinatura digital do ficheiro cifrado aquando este é guardado no disco (i.e., a cada 5 minutos). Para tal, deve ser gerado um par de chaves RSA que acompanha a aplicação.

Este projeto pressupõe o desenvolvimento de uma aplicação que procura obter as entradas do *Clipboard* em segundo plano. Para tal, pode pensar em colocá-la a correr como um *daemon* que faz CTRL+V a cada 2 segundos, na expetativa de encontrar uma entrada nova. Eventualmente, pode ainda desenvolver uma aplicação para configuração de alguns parâmetros (e.g., quantas entradas são suportadas). A aplicação principal deve permitir ver as várias entradas no *Clipboard* e também escolher qual deve ir para lá através de duplo clique. Note que a versão mais simples da aplicação, o histórico é perdido sempre que esta é terminada (porque a chave de cifra que permite decifrar o ficheiro está apenas guardada em memória).

Uma versão mais elaborada da aplicação deve ter funcionalidades adicionais, como as que se sugerem a seguir:

- o gestor deve suportar a geração de chaves de cifra a partir de uma pitada de sal e de uma palavra-passe mestra, configurada pelo utilizador aquando da primeira utilização. Deve ser usada, para o efeito, uma função de derivação de chaves de cifra como a PBKDF2;
- tomando a funcionalidade anterior como garantida, o gestor deve conseguir decifrar ficheiros de históricos anteriores;
- tomando a funcionalidade anterior como garantida, o gestor deve conseguir decifrar, de forma muito seletiva, apenas as entradas selecionadas pelo utilizador. Esta funcionalidade requer que seja usado um modo de cifra adequado;
- o gestor deve suportar vários utilizadores (cada utilizador terá o seu histórico);
- as assinaturas devem ser verificadas a pedido do utilizador;
- permitir que os programas cliente utilizem certificados digitais para validação das assinaturas digitais;
- escrever um *help* bastante completo.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe uma secção no relatório.

5 SAFEBOOK: Plataforma para Publicar Mensagens Online Protegidas

SAFEBOOK: Platform to Publish Protected Online Messages

Este projeto aponta para o desenvolvimento de uma plataforma para publicar mensagens *online* tipo *Twitter*. A maior diferença é que as mensagens são publicadas cifradas, e só depois do recetor dessas mensagens as decifrar e ver, é que são disponibilizadas em texto limpo. Por outras palavras, cada utilizador do sistema deverá ter a possibilidade de publicar uma mensagem direcionada para outro utilizador, e só quando esse utilizador ler essa mensagem é que esta aparece em texto limpo no site. Todas as mensagens devem ser assinadas digitalmente pelo autor original das mesmas e, após serem lidas pelo destinatário, também devem conter uma assinatura deste último. As funcionalidades básicas da plataforma são:

- registo de um novo utilizador;
- cifragem, assinatura digital e publicação de mensagens;
- decifragem de mensagens direcionadas ao utilizador no programa cliente;
- verificação da assinatura digital das mensagens decifradas;
- notificação de nova mensagem.

A natureza desta plataforma define implicitamente dois programas:

- Um `servidor` que gere as informações de todos os utilizadores (tipo `username` e `passwords`) e publica as mensagens;
- e um `cliente`, que produz assinaturas digitais, cifra ou decifra mensagens (dependendo do papel que o seu utilizador está a tomar nesse momento), e que se liga ao servidor para publicar mensagens.

Note-se que a ideia princípal deste projeto é a de que esta plataforma faça uso de mecanismos de criptografia simétricas e assimétrica. Isto é, quando determinado utilizador quiser publicar uma mensagem, deve escolher (interativamente) o destinatário da mesma, pedir ao servidor que lhe envie a chave pública (ou o certificado) desse destinatário e cifrar a mensagem com essa chave. O destinatário deve posteriormente receber uma notificação de *nova mensagem recebida*, decifrá-la com a sua chave privada e, caso concorde, substituir o texto cifrado publicado pelo texto limpo que decifrou.

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

- ter um *help* bastante completo;
- Usar certificados digitais X.509, e implementar uma infraestrutura de chave pública para o sistema e validar cadeias de certificados;
- Pensar numa forma correta de fornecer certificados digitais a utilizadores;
- Permitir escolher entre dois ou mais algoritmos de cifra e funções de *hash*;
- Adicionar outros serviços, e.g., envio de uma mensagem para vários utilizadores (ao invés de um só).

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe uma secção no relatório.

6 MR.SMITH: Plataforma para Cifrar e Assinar usando um Agente de Confiança

MR.SMITH: Platform for Encrypting and Digitally Signing Files using a Trusted Third Party

O objetivo principal deste trabalho é o de construir uma plataforma para assinar e transmitir ficheiros de um modo seguro, entre dois ou mais utilizadores, recorrendo apenas a mecanismos da criptografia simétrica e a um agente de confiança. Entre outras, apontam-se as seguintes funcionalidades básicas do sistema a desenvolver:

- deve ser possível fazer o registo junto do agente de confiança;
- o utilizador, depois de devidamente autenticado no sistema, deve poder escolher se quer cifrar ou assinar um ficheiro antes de o enviar; deve também poder escolher o utilizador para o qual quer enviar o ficheiro a partir de uma lista ligados ao sistema;
- caso o utilizador escolha não cifrar o ficheiro a transmitir, este deve ser acompanhado de um código de autenticação da mensagem (deve ser usado um *Hash based Message Authentication Code* (HMAC) para o efeito);
- as chaves de sessão devem ser geradas por um dos clientes e trocadas via agente de confiança;
- o agente de confiança deve ser capaz de fazer assinaturas digitais de ficheiros transmitidos por utilizadores autenticados do sistema.

Note que a definição deste projeto indicia que o sistema só poderá estar completamente operacional quando estão ligados, pelo menos, três intervenientes: (i) a aplicação (cliente) que vai ser usada para transmitir o ficheiro, (ii) a aplicação (cliente) que vai ser usada para receber o ficheiro e (iii), o agente de confiança. Pressupõe-se que o agente de confiança já está a correr quando qualquer utilizador se tenta ligar ao sistema, que é este que gera todas as chaves de sessão para confidencialidade nas comunicações, e que alimenta cada cliente com as informações acerca de outros clientes ligados ao sistema (para que determinado cliente possa enviar um ficheiro a outro, este deve estar *online*). Pressupõe-se que as chaves de cifra de chaves de sessão são inseridas manualmente no sistema aquando da inicialização de determinado cliente.

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

- permitir que apenas as primeiras chaves de cifra de chaves de sessão sejam trocadas usando criptografia assimétrica.
- garantir que as credenciais de autenticação no servidor / agente de confiança (i.e. *username* e *password*) são trocadas de forma segura.
- assegurar que as chaves de cifra de chaves de sessão mudam sempre que o utilizador faz *login* no sistema, sem que se perca sincronismo e nunca recorrendo a mecanismos de criptografia simétrica (pesquisar por *one time password*);
- permitir que o utilizador escolha a cifra a utilizar e o comprimento da chave de cifra;
- ter um *help* bastante completo.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe uma secção no relatório.

7 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 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 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).

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 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 uma secção no relatório.

8 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 uma secção no relatório.

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* 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 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., AES-CBC e 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 uma secção no relatório.

10 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 actua 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 uma secção no relatório.

11 OpenEasySSL: Comandos Integrados e Interativos para OpenSSL

OpenEasySSL: Integrated and Interactive Commands for OpenSSL

Nota: este tema só está disponível para Informática Web.

A ferramenta `openssl` é difícil de utilizar para utilizadores habituados apenas a interfaces gráficas, e mesmo para utilizadores com conhecimentos básicos em criptografia têm dificuldade em ajustar todos os parâmetros de muitos dos seus comandos. O objetivo deste projeto é que sejam desenvolvidos vários comandos (*scripts*) interativos para *command line interface* que facilitem a utilização de comandos `openssl`. A ideia é substituir comandos como

```
$  openssl enc -rc4 -K 0de583210e14a0cfc2f45365 -in texto-limpo.txt -out
criptograma.rc4
```

por also semelhante a

```
$  cifrar-com-openssl

#  Escolha a cifra a utilizar:
1 - RC4 (desencorajada)
2 - AES-128
3 - AES-256
...
1

#  Introduza a chave de cifra:
(deixar em branco para gerar uma aleatoriamente)
0de583210e14a0cfc2f45365

#  Introduza o nome do ficheiro de entrada:
(deixar em branco para cifrar uma mensagem do teclado)
texto-limpo.txt

#  Introduza o nome do ficheiro de saída:
(deixar em branco para gerar um nome automaticamente)
criptograma.rc4

#  Pretende remover o ficheiro de entrada depois de cifrar?  (sim/não)
sim

#  Ficheiro cifrado com sucesso:
*  Cifra utilizada:    RC4
*  Chave de cifra:     0de583210e14a0cfc2f45365
*  Ficheiro de saída:  criptograma.rc4
*  Ficheiro de entrada eliminado.
```

Portanto, a ideia é substituir um comando complexo por um comando simples `$ cifrar-com-openssl`, que depois ajude o utilizador a fazer as escolhas importantes e necessárias. O conjunto base de funcionalidades/comandos/partes do *openssl* a suportar são:

- A cifra e decifra de ficheiros (parte `enc`), conforme ilustrado em cima;
- O cálculo de valores de *hash*;
- O cálculo e verificação de valores HMAC (mínimo SHA256 e SHA512);
- Geração e manipulação simples de chaves RSA (gerar chaves, extrair chave pública para ficheiros à parte, etc.);
- O cálculo de assinaturas digitais RSA-SHA256;
- A verificação de assinaturas digitais RSA-SHA256.

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

- São geradas chaves de cifra simétrica aleatórias com o tamanho adequado à cifra quando o utilizador opta por deixar a chave em branco (ver exemplo em cima);
- É possível cifrar mensagens diretamente do teclado quando não se especifica o nome do ficheiro de entrada;
- As chaves privadas RSA são geradas e guardadas de forma segura imediatamente (cifradas com uma cifra de chave simétrica);

- Implementar opções de codificação (e.g., BASE64);
- É suportada a geração e manipulação de certificados digitais X.509 (geração de pedidos de certificado, assinatura, manuseamento de cadeias de certificação, etc.)
- Pensar em outras formas de acrescentar valor e interatividade aos comandos;
- escrever um *help* bastante completo.

Pensem numa forma de atacar o sistema (uma falha da sua implementação) e dediquem-lhe uma secção no relatório.