SEE-U-L4TER: Uma Máquina do Tempo Criptográfica

O objetivo deste trabalho é desenvolver um sistema que permita cifrar mensagens (ficheiros) que só podem ser decifradas em determinado dia e hora. Para que seja seguro, o sistema deve implementar uma arquitetura cliente servidor (por exemplo, uma aplicação web), em que o componente do servidor está claramente separado do cliente, e que tem como principais funcionalidades a geração de chaves de cifra para qualquer momento especificado e a cifra de mensagens com essas chaves.
Assim, o sistema deve ter três grandes funcionalidades, eventualmente disponibilizadas em três interfaces:

- Usando o sistema, um utilizador deve poder pedir que lhe seja, por exemplo, cifrado um ficheiro que só pode ser aberto no dia e hora 25 de maio de 2030, 12h00, introduzindo a data e o ficheiro através de alguma interface (por exemplo, um formulário). O sistema deve então gerar uma chave de cifra única AES que dependa de um segredo que só o sistema sabe, do nome de utilizador (e-mail) do utilizador que submete e da data e hora. Depois deve cifrar o ficheiro e devolver o criptograma ao utilizador (sem a chave de cifra), para que o utilizador dele faça o que quiser, inclusive enviá-lo a outro utilizador;

- Qualquer outro utilizador deve poder aceder ao sistema e verificar qual a chave que abre os critptogramas da data e hora atual numa interface não autenticada. Por outras palavras, qualquer utilizador tem acesso a chaves de decifra a todo o momento, mas cada chave só é mostrada nesse período de tempo, e nunca antes ou depois;

- Adicionalmente, o sistema pode ter também uma interface que permita a submissão de uma chave de cifra e de um ficheiro cifrado anteriormente, devolvendo em resultado o ficheiro decifrado (se a decifra for bem sucedida).

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

1. O sistema gera chaves de cifra pseudo aleatoriamente usando um algoritmo criptográfico, mostrando-as numa interface isolada para a data atual na página de landing. As chaves de cifra são dependentes de um e-mail, de um segredo (que só o sistema sabe), e do dia e hora atuais. Por exemplo SHA256(e-mail || segredo || data e hora);

2. A pedido de um utilizador, o sistema gera chaves de cifra para um determinado dia e hora (usando a mesma técnica referida no ponto anterior), e cifra um ficheiro (submetido pelo utilizador) com essa chave;

3. Aquando da operação no ponto anterior, o sistema gera também um código de autenticação de mensagem HMAC-SHA256, que é acoplado ao criptograma;

4. A pedido de um utilizador, o sistema tenta a decifra de um criptograma que lhe seja submetido usando a chave que o utilizador também fornecer;

5. Aquando da execução da tarefa no ponto anterior, o sistema verifica o código de autenticação de mensagem e avisa o utilizador o código não verificar.

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 ou tem-nas instaladas pelo administrador do sistema);

4. O sistema permite 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);

5. Caso a funcionalidade anterior seja implementada, o sistema deve permitir a utilizadores registados acederem a qualquer chave de cifra do passado (mas não as futuras);

6. Outras funcionalidades que considere interessantes.

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 (não precisa necessariamente executar em rede).

SEE-U-L4TER: A Cryptographic Time Machine

The aim of this work is to develop a system that makes it possible to encrypt messages (files) that can only be decrypted on a given day and time. In order to be secure, the system must implement a client-server architecture (for example, a web application), in which the server component is clearly separated from the client, and whose main features are the generation of cipher keys for any specified time and the encryption of messages with these keys.
The system should therefore have three main functions, possibly available in three interfaces:

- Using the system, a user should be able to request that, for example, a file be encrypted that can only be opened on May 25, 2030, 12:00 PM, by entering the date and the file through some interface (for example, a form). The system must then generate a unique AES cipher key that depends on a secret that only the system knows, the username (e-mail) of the user submitting it and the date and time. It must then encrypt the file and return the cryptogram to the user (without the encryption key), so that the user can do whatever they want with it, including sending it to another user;

- Any other user must be able to access the system and check which key opens the current date and time cryptograms in an unauthenticated interface. In other words, any user has access to decryption keys at all times, but each key is only shown in that time period, and never before or after;

- In addition, the system can also have an interface that allows you to submit an encryption key and a previously encrypted file, returning the decrypted file as a result (if the decryption is successful).

Thus, the basic functionalities to be supported by the system are:

1. The system generates pseudo-random cipher keys using a cryptographic algorithm, displaying them in an isolated interface for the current date on the landing page. The cipher keys are dependent on an e-mail, a secret (which only the system knows), and the current day and time. For example SHA256(e-mail || secret || date and time);

2. At a user's request, the system generates cipher keys for a given day and time (using the same technique as in the previous point), and encrypts a file (submitted by the user) with that key;

3. During the operation in the previous point, the system also generates an HMAC-SHA256 message authentication code, which is attached to the cryptogram;

4. At a user's request, the system attempts to decrypt a cryptogram submitted to it using the key that the user also provides;

5. When the task in the previous point is carried out, the system checks the message authentication code and warns the user if the code is not checked.

You can strengthen your work and knowledge by implementing the following functionalities:

1. The system must allow you to choose the cipher used between AES-128-CBC and AES-128-CTR;

2. The system must allow you to choose the HMAC function used between HMAC-SHA256 and HMAC-SHA512;

3. In addition to using message authentication codes to check whether a message has been properly decrypted, the system uses RSA digital signatures (in this case, the system has a way of generating an RSA key pair or has them installed by the system administrator);

4. The system allows simple user registration using an e-mail address and password (a secure representation of the password must be stored in the database);

5. If the above functionality is implemented, the system must allow registered users to access any past cipher keys (but not future ones);

6. Other features that you consider interesting.

Think of a way to attack the system (a flaw in its implementation) and dedicate a short time slot to it in the presentation. Note that, for evaluation and prototyping purposes, the system developed can run all its components/applications/programs locally, as long as it simulates or realizes the suggested architecture (it doesn't necessarily have to run on a network).