Week 10

Andre Rodrigues - up201505639
Pedro Antunes - up201507254

Consolidation questions

1 - What is the difference between IND-CPA security in symmetric and public-key settings

A diferença é que no caso da chave pública o atacante recebe a chave pública que encripta os plaintexts, parecido com enviar plaintexts e receber o ciphertext, só que aqui é o atacante que consegue fazer esse processo. E com acesso a chave pública o atacante poderá tentar descobrir a chave privada pois elas estão relacionadas (embora não irá conseguir porque nós acreditamos que estas funções sejam one-way). Contudo para passar o teste IND-CPA, o algoritmo de chave publica não pode ser determinístico porque um atacante pode encriptar os seus plaintexts e assim descobrir qual deles corresponde ao ciphertext enviado e assim ganhar o jogo “IND-CPA”.

2 - Why are all asymmetric public-key encryption schemes in practice hybrid? Explain their structure.

Todos os esquemas de cifrar com chave pública são esquemas híbridos porque combinam os esquemas de chave pública com os esquemas de chave simétrica, ou seja, o esquema de chave pública é usado para cifrar uma chave simétrica que depois será usada para cifrar os dados transmitidos. Isto acontece porque o esquema de chave pública é bastante moroso e necessita de muitos recursos, que aumentam muito consoante o tamanho da mensagem. Só que a vantagem de usar estes esquemas é que pode-se trocar chaves simétricas sem que as entidades envolvidas na comunicação nunca antes se tenham encontradas, pois um emissor cifra a chave simétrica com a chave pública do receptor e envia-a esse ciphertext para o receptor, que depois será decifrado com a chave privada obtendo assim a chave simétrica (o ciphertext só pode ser decifrado pelo portador da chave privada).

3 - What security properties are guaranteed by digital signatures and how are they different to MACs?

As assinaturas digitais permitem confirmar a identidade de quem assinou a mensagem, permite também usar essa mensagem assinada como prova visto que quem a assinou não a pode negar (não repúdio) e permite saber que o documento não foi alterado depois de ser assinado.

As assinaturas digitais são diferentes dos MACs porque aqui não é recalculada a assinatura como forma de verificação porque para calcular a assinatura é necessário ter a chave privada enquanto que para verificar é usada a chave pública, ou seja, neste caso tem de existir um algoritmo que recebe a assinatura, a mensagem e depois a chave pública e depois verifica se é consistente.
Outra diferença é que os MACs não oferecem a vantagem de não-repúdio porque duas entidades possuem a mesma chave, portanto ambos podem criar autenticação das mensagens e negar a sua criação.

4 - Explain why the raw RSA function is not a secure public-key encryption scheme

Tal como referido na pergunta 1, a função RSA (textbook) não satisfaz o IND-CPA porque esta é determinística, ou seja, para um determinado plaintext dá sempre o mesmo ciphertext. Como o IND-CPA é o mínimo de segurança que uma cifra pode ter, dizemos que a função RSA por si só não é segura, é necessário uma construção que implemente o RSA de forma mais segura. E outro problema que a torna insegura é o facto de ser maleável, ou seja, através de dois ciphertexts consegue criar um outro ciphertext válido.

5 - Explain why the raw RSA function is not a secure digital signature scheme.

Tal como a função RSA não é segura para cifrar, também não é segura para criar assinaturas digitais devido a sofrer de um ataque chamado de “Blind Attack”.

Com este ataque conseguimos obter uma assinatura numa mensagem que talvez a entidade responsável pela assinatura não quisesse validar (M). Vou pegar nessa mensagem e vou máscara-la através de um R tal que R^eM e depois vou tentar obter uma assinatura nessa mensagem camuflada, obtendo S = (R^eM)^d o que nos vai permitir obter S = R^edM^d ⇔ S = RM^d, onde depois iremos dividir S/R e obter a mensagem pretendida devidamente assinada (M^d).

Guided practical assignment

1 - Use openSSL option `smime`:

- **first decrypt the message using `decryptionkey.pem`**

`$` openssl smime -decrypt -in testmessage.eml -inkey decryptionkey.pem -out t.pem
Password: 1234

- **then verify the signature using `verificationkey.pem`**

```
$ openssl smime -verify -in t.pem -CAfile rootcacert.pem -certfile
verificationkey.pem
```

- **you can disable certificate verification in the previous step**

$ openssl smime -verify -noverify -in t.pem