Week 7

Andre Rodrigues - up201505639
Pedro Antunes - up201507254

Consolidation questions

1 - What security guarantees does a MAC give?

Um MAC garante a autenticidade e a integridade da mensagem porque apenas quem conhece a chave pode alterar a mensagem e dessa forma garante também a autenticidade pois o receptor da mensagem sabe quem tem o resto das chaves e portanto sabe que a mensagem veio de um desses emissores. O MAC não garante contudo que um emissor que não conheça a chave lhe envie um par (mensagem, tag) que lhe foi enviado no passado usando essa mesma chave, designamos este fenômeno por “replay”. Para corrigir este tipo de ataque, ambos devem guardar um counter que deve ser anexado junto com a mensagem, antes de autenticar para que caso o receptor receba uma mensagem inferior ao seu counter, saber que a pode descartar.

2 - Why is every PRF a MAC, but not necessarily the converse?

Todas as PRF podem ser um MAC porque a definição de MAC diz que o seu output não pode ser adivinhado o que é uma redução da definição da PRF que diz que o seu output não pode ser distinguido de uma string aleatória.

O contrário, nem todos os MAC podem ser PRF, porque basta olhar para este contra-exemplo:

Digamos que temos uma PRF1 e queremos construir uma PRF2 através da PRF1 tal que: PRF2(K,M) = PRF1(K,M) || 0

Devido a PRF2 fazer um append do bit 0 podemos distinguir sempre o seu output de uma string aleatória. Portanto a PRF2 não é uma PRF, contudo continua a ser um MAC porque apesar do 0 o resto continua a não ser possível de ser adivinhado.

Conclusão: a noção de PRF é mais segura do que a de um MAC.

3 - Explain why finding a collision in the hash function may not necessarily allow you to break HMAC.

Como a chave não é conhecida é muito mais improvável obter uma colisão porque agora não é possível fazer a pré-computação de colisões apenas conhecendo as mensagens.

Contudo mesmo que ocorram colisões numa função de hash que representa um valor interno com outros valores internos de execuções anteriores isso não é um problema pois esse valor será usado em conjunto com outro valor para recalcular o valor MAC. A colisão destes dois valores ou de valores MAC é que podem levar a quebra do HMAC.

4 - What does the Wegman-Carter construction do? What MAC construction that you know uses Wegman-Carter?

Como a construção HMAC utilizava uma cifra por blocos para implementar a função de compressão subjacente à função hash utilizada posteriormente pela construção HMAC para gerar um MAC, foi pensado em formas mais simples de gerar um MAC através de cifras por blocos sem recorrer a uma função de hash tradicional, uma função mais fraca criptograficamente, contudo mais rápida.

Portanto foi criada a função de hash universal que tem uma noção de resistência a colisões diferente, com base em probabilidades. E onde o MAC era calculado através de um polinômio, só que continha uma vulnerabilidade que permitia forjar tags e assim quebrar o MAC.

Posto isto, surgiu a construção de wegman-carter que corrigia este problema bem como permite a autenticação de múltiplas mensagens (com a mesma chave) usando uma função de hash universal, o que era uma limitação da mesma.

Para isso esta construção somava o output de UM (polinômio) com uma PRF que utilizava um nonce uma chave diferente de UH, isto para esconder os valores do polinômio computados por UH, que era o que dava origem a vulnerabilidade referida anteriormente.

A Google utiliza esta construção para garantir a segurança das conexões HTTPS (HTTP over TLS) e também utilizado pelo openSSH. É utilizado através do poly1305

que utiliza a construção de wegman-carter só que ao invés de utilizar uma PRF utiliza o AES porque PRP => PRF para um output de tamanho grande.

Guided practical assignment

1 - Implement the universal hash function ofpoly1305 in Sage

- recall H((K1, K2),(M1, M2, . . .)) =K1+P(K2) where P(X) =K1+M1X+M2X2+. . .
- use F = FiniteField(2**130-5) to define the type of coefficients
- usePR.<X> = PolynomialRing(F) to define the type of polynomials
- define the key to the hash as a pair in F
- define the message as a list in F, which you can cast to a polynomial (careful with 0-th coefficient, which comesfrom the key)
- computing the hash is evaluating the polynomial at the other key component
- What is the probability that the hash of two fixed messages collide for a randomly sampled key?

Ficheiro poly1305.py

What is the probability that the hash of two fixed messages collide for a randomly sampled key?

A probabilidade de haver uma colisão de duas mensagens diferentes para uma mesma chave:

PR[H(K,M1) = H(K,M2)]

É menor do que n/p.

Onde n = min(M1,M2) e corresponde ao número de blocos da mensagem e p no caso do poly1305 é 2^(130) - 5

2.

Message.txt -> “crypto”

openssl dgst -hmac '1234' message.txt

HMAC-SHA256(message.txt)=
1e9171335e87712eb5c2b33f0ad9bf685b938957a551031204acafaf584c0b0c

Message.txt -> “crypto123crypto”

HMAC-SHA256(message.txt)=
9911052dd3e35106f034b4008c665005c6bb214604e8373172f5487a0559de2e

3.

Ficheiro ex3_python.py