(Applied) Cryptography

Tutorial #4

Manuel Barbosa (mbb@fc.up.pt) Rogério Reis (rvreis@fc.up.pt)

MSI/MCC/MERSI – 2022/2023

1 - Implement the universal hash function of poly1305 in Sage

- recall $H((K_1, K_2), (M_1, M_2, \ldots)) = K_1 + P(K_2)$ where $P(X) = K_1 + M_1 X + M_2 X^2 + \ldots$
- use F = FiniteField(2**130-5) to define the type of coefficients
- use PR.<X> = PolynomialRing(F) to define the type of polynomials
- define the key to the hash as a pair of elements in F
- define the message as a list in F, which you can cast to a polynomial (careful with 0-th coefficient, which comes from 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?

2 - Use Python to encrypt a file with AES-GCM

- Make sure you can decrypt it with openSSL (if the command line does not support AEAD in your machine, use this tool `https://github.com/jforissier/aesgcm`).
- Modify the encrypted file
- See if you can still decrypt it with openSSL
- How would this be different if you were using AES-CTR?

3 - A length extension attack works as follows.

- Application generates secret key K, which is kept hidden
- At some point application computes $h = H(K \| M)$ for some message M and publishes (M, h).
- Intuitively it should be impossible for some attacker to compute $H(K \| M')$ for $M \neq M'$.
- However, for some hash functions, it is possible to compute such a value using only (M, h).

This technique has been explained in theoretical classes for the SHA-2 family.

Demonstrate the attack by constructing:

- A Python program that generates K, computes $h = \text{SHA2}(K \| M)$ for some M and saves K, M and h into different files.
- Another Python program that reads M and h (but not K!) and generates some M' and h' into different files.

Is must be the case that $\text{SHA2}(K \| M') = h'$ and that $M \neq M'$.