Applied Cryptography

Week #11 Extra

Bernardo Portela and Rogério Reis

2024/2025

Important

- Your answers must **always** be accompanied by a justification. Presenting the final result (e.g. the result of a calculation) without the rationale that laid to said result will result in a grade of 0.
- Submit your answers via e-mail to *bernardo.portela@fc.up.pt*, with adequate identification of the group and its members.

Q1: Demonstrating forgeability of plain RSA

A key property of digital signatures is that of existential unforgeability, which is expressed as a security game as follows.

Existential Unforgeability

1 - We generate a keypair (sk, pk)

2 - We give pk to the adversary, and allow him to request signatures of arbitrary messages. This means that the adversary is free to request signatures of anything, but **every signed message** is recorded.

3 - The adversary must then produce a pair (m, s). He wins the game if i.) s is a valid signature for m and ii.) the signature of m was **not** requested in step 2.

Plain RSA signature scheme

Plain RSA as a signature scheme sets (e, n) as the public key and (d, n) as the private key. Signatures of m are m^d and validation of (m, s) is checking if $m = s^e$

It is easy to see that **plain RSA** is not existentially unforgeable. Indeed, anyone can produce forgery $(1, 1)$, as $1^e = 1$ regardless of the value of e. However the issues of signing with plain RSA go above and beyond this singular case. Let's go for a slightly more challenging scenario.

Question: Describe how an adversary can produce a valid forgery for the following experiment with probability 1.

Existential unforgeability with a twist

1 - We generate a keypair (sk, pk) and select a bad value b.

2 - We give pk and b to the adversary, and allow him to request signatures of arbitrary messages. This means that the adversary is free to request signatures of anything, but **every signed message** is recorded.

3 - The adversary must then produce a signature s. He wins the game if i.) s is a valid signature for b and ii.) the signature of b was **not** requested in step 2.

Q2: Shamir Secret Sharing

In the extra work of class 1, we overviewed a technique for secret sharing that allowed messages to be split in parts, such that it could only be reconstructed if a subset of participants agreed to participate – i.e. revealed their "shares" of the original value.

Shamir Secret Sharing is based on polynomial interpolation over finite fields. Consider a point $(0, 1)$. It is easy to observe that there is an overwhelming number of polynomials p of degree 1 such that $p(0) = 1$.

E.g.

$p(x) = x + 1$; $p(x) = 2x + 1$; $p(x) = 4x + 1$; etc.

However, when given two points $(0, 1)$ and $(1, 3)$ there is only **one** polynomial p for which these points are valid: $p(x) = 2x + 1$

The intuition for Shamir Secret Sharing is that secrets are represented as polynomials, and shares are represented as points in said polynomial. As such, one can very flexibly select the number of necessary shares and the threshold for secret reconstruction by adjusting the parameters of the system.

We will consider integer polynomial coefficients, and standard integer arithmetic. A practical implementation of shamir secret sharing requires computation over a finite field to ensure privacy. However, for this didatic exercise, *we can disregard this very simple adaptation*, as the intuitions we want to understand can be observed in integer arithmetic, which is slightly more straightfoward to explore.

Question - P1:

Implement the secret sharing function that takes value x and produces a set of four shares x_1, x_2, x_3, x_4 such that any three can be used to reconstruct the original value.

This will entail responding to the following problems:

1- What degree should the polynomial be, so that three points are sufficient to reconstruct the value, but two points can never be enough?

2- How can we generate a polynomial f of that degree, such that $f(0) = x$?

Question - P2:

Implement a function for polynomial interpolation, that takes n points, and recovers the only polynomial of degree $n - 1$ that contains those points.

Show how this allows for the secret to be recovered:

1- Generate a polynomial f for secret number 1001 and shares (points) for that secret.

2- Take only the minimum amount of necessary shares (randomly selected) and retrieve the polynomial f'

3- Show that, for $x = 0$, $f'(x) = 1001$, and thus the secret can be recovered.

Question - P3:

Use what you implemented to test the following:

1- Generate a polynomial f for secret number 100 and shares (points) for that secret: x_1, x_2, x_3, x_4.

2- Generate another polynomial g for secret number 550 and shares (points) for that secret y_1, y_2, y_3, y_4

3- Calculate $z_1 = x_1 + y_1$; $z_2 = x_2 + y_2$ and $z_3 = x_3 + y_3$

4- Use your secret recovery method using points z_1, z_2, z_3. What can you conclude from the result? Justify what happened.