Homework #3
RSA

CNS Course Sapienza

Riccardo Prinzivalle, 1904064

November 20, 2020

1 Homework Goal

This homework contains an implementation of RSA algorithm based on major libraries for the mathematical functions, a comparison with the insecure *PyCryptoDome* RSA and with the *PyCryptoDome* AES.

2 RSA Implementation

RSA basically is divided in two part: the initialization and the encryption.
The encryption phase is simpler than the initialization: it uses only an exponentiation and a modulo reduction. The modulo reduction is already implemented in python (used in this homework), instead what it is needed is an efficient implementation of the exponentiation. To do so, I used the proposed **Square And Multiply** in the slides of the course: at first with small values of base and exponent it worked flawlessly, but during the test with bigger messages and keys, the computation explodes, so I thought to reduce in modulo after every performed computation (since we have to do it after the exponentiation, so why don't do it at every step?) and the **SAM** computation time dropped to some seconds. The code can be seen in Fig. 1

```
def sam(base, exp, n):
    f = 1
    while exp > 0:
        lsb = 0x1 & exp
        exp >>= 1
        if lsb:
            f *= base
        base *= base
        base = base % n
        f = f % n
    return f
```

Figure 1: Improved version of Square And Multiply

Now the easy task is done; the initialization is the part which requires more attention to guarantee the security of our implementation. The operation to be performed are:

1. find two large prime numbers p and q
2. compute $n = p \cdot q$
3. compute Euler Phi function as $\Phi(n) = (p-1)(q-1)$ exploiting prime properties of p and q
4. select an exponent $e \in 1, 2, \ldots, \Phi(n)$ such that $gcd(e, \Phi(n) = 1$
5. compute d such that $d \cdot e = 1 \mod \Phi(n)$

3 RSA Comparison

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer Factorization	RSA	1024	3072	7680	15360
Discrete Logarithm	DH, DSA, Elgamal	1024	3072	7680	15360
Elliptic Curves	ECDH, ECDSA	160	256	384	512
Symmetric key	AES, 3DES	80	128	192	256

Table 1: Key length comparison in public key and symmetric key algorithm

4 Conclusion

After this brief introduction on elliptic curves, it is obvious why they have been widely adopted in many cases of asymmetric encryption: they use less bits for the same level of security, so are more efficient to compute and their base concepts are easier to visualize. As suggested on section **??**, EC can be easily used in hybrid encryption scheme, in the key exchange phase.

References

[1] C. Paar and J. Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer Publishing Company, Incorporated, 1st edition, 2009.