Applied Cryptography

Week #11 Extra

Bernardo Portela and Rogério Reis

2023/2024

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: Man-in-the-Middle

Implement a prototype that demonstrates how a Man-in-the-Middle attack can occur in a standard unauthenticated Diffie-Hellman key exchange. As usual, this happens between usual suspects *Alice* and *Bob*. In appendix, you can find three files:

- `config` is a configuration file that is used by Alice and Bob to know who to talk to.
- `alice.py` establishes a connection with Bob (hopefully), and performs the Diffie-Hellman key exchange.
- `bob.py` mirrors the behavior of Alice.

The goal of the work is to design the man-in-the-middle adversary code: `mitm.py`. Your code must convince Alice and Bob to instead talk to him, and perform a key exchange with him. **Your attack must not change the source code of Alice or Bob**. Your attack is successful if Alice and Bob are **not** agreeing on the same secret, and instead the secrets they have agreed to are both known to the adversary.

Suggestion: Start by analysing the code for Alice and Bob, what are they using to communicate? How can we subvert this mechanism to be more... convenient?

References: To facilitate communication, this code uses *pwntools* (reference). It is not mandatory to use this, but the library considerably facilitates communication.

Q2: ECC

The following is a naive attempt at an elliptic curve signature scheme. Consider a global elliptic curve, prime p and generator G. The scheme works as follows.

- Alice picks a private signing key sk_A and forms the public verifying key by computing $pk_A \leftarrow sk_A \cdot G$
- To sign message m, Alice picks a random value k, and computes the signature $\sigma \leftarrow m - k \cdot sk_A \cdot G$. It then sends to Bob the tuple (m, k, σ)
- To verify the signature, Bob checks that $m = \sigma + k \cdot pk_A$. If this is true, the signature is validated.

Question - P1: Show that the scheme works, i.e. show that, for correctly signed messages, the verification algorithm works accordingly.

Question - P2: Show that this scheme is vulnerable, by describing a simple technique for forging a signature on an arbitrary message, without knowledge of the secret key sk_A. *Hint:* consider what computations can one do using simply pk_A

Q3: Post-quantum Cryptography

The evolution of quantum computation technology poses a looming threat to cryptographic mechanisms of common usage, such as public-key encryption, digital signatures and key agreement protocols. This is mainly due to Shor's algorithm for quantum computation, which is theoretically capable of solving problems such as integer factorization or the discrete logarithm in polynomial time, given sufficient stable qubits. This has led to the development of post-quantum cryptographic (PQC) algorithms, which rely on different mathematical objects which, to the best knowledge of cryptographers worldwide, are computationally hard to be solved by quantum computers.

Question - P1: A common misconception is that, to develop post-quantum cryptographic algorithms, one needs to leverage quantum computation. Explain why this is not the case, by distinguishing *quantum computation* from *post-quantum cryptography*.

Shor's algorithm is expected to be able to efficiently break RSA-2048 using roughly 20 million qubits (reference). Cutting-edge quantum computers are barely surpassing 1000 qubits. However, developing PQC algorithms is an extremely pressing concern.

Question - P2: Investigate an attack technique called store-now-decrypt-later (ENDL), and discuss why this approach is a good argument for dealing with quantum threats as soon as possible.