1 RSA Composition

Suppose Alice and Bob wish to communicate using their RSA public keys. Alice has a message for Bob which she wants to both sign using her own key, and encrypt using Bob’s public key. Which order should she perform these operations? One ordering is fundamentally insecure, please explain what attack is possible if the operations are not properly ordered.

2 Cipher Strengthening

Once it became clear that the 56-bit keys used in DES were insecure, cryptographers began to recommend Triple-DES, that is, defining a key $k$ as a 168-bit triple $(k_1, k_2, k_3)$ and computing an encryption $E'_k(m) = E_{k_3}(E_{k_2}(E_{k_1}(m)))$.

112-bit keys are still considered secure, so it’s worth examining why Double-DES was never recommended. Suppose an adversary has the computation resources to brute-force the keyspace of DES.

a. Explain an attack on the construction $E'_k(m) = E_{k_2}(E_{k_1}(m))$ for two randomly chosen 56-bit keys $k_1$ and $k_2$, given a few known plaintext-ciphertext pairs $(m, c)$. The “meet-in-the-middle” attack will involve brute-forcing the encryption of $m$ under $k_1$ and brute-forcing the decryption of $c$ under the key $k_2$.

b. Roughly how much harder is this attack than brute-forcing traditional DES? In particular, what memory requirements are there? Are there cases where an attacker who can easily brute-force DES encryption may be less prepared to brute-force decryption?

c. Given the above attack, estimate the effective bit-size of the popular Triple-DES construction, which ostensibly uses 168-bit keys.

3 Single Sign-on

Briefly explain how OpenID works and give a sketch of its protocol.
4  Hardware attacks

You are attempting to extract a 4-digit PIN from a secure crypto-processor on a smart card. You have reverse-engineering the PIN-verification routine:

```python
if failed_attempts < 3 && verify_pin(PIN):
    failed_attempts = 0
    startup()
else:
    failed_attempts++
```

a. Describe the cheapest hardware attack that can recover the PIN from this card.

b. Is this an attack that a typical bank adversary can afford?

c. Propose a re-factoring of the code that will eliminate this attack.

5  Any questions from the course that you may have...