Security

Assignment 11, Friday, December 2, 2016

Handing in your answers: For the full story, see

`http://www.sos.cs.ru.nl/applications/courses/security2016/exercises.html`

To summarize:

- Include your name and student number **in** the document (they will be printed!), as well as the name of your teaching assistant (Hans or Joost). When working together, include **both** your names and student numbers.
- Submit one single **pdf** file – when working together, only hand in **once**.
- Hand in via Blackboard, before the deadline.

Deadline: Monday, December 12, 09:00 sharp!

Goals: After completing these exercises successfully you should be able to

- notice the risks of shared RSA moduli;
- analyse relations among certificate within a certificate chain;
- keep a structured overview while performing larger computations

Marks: You can score a total of 100 points.

1. **(20 points)** We assume that Alice and Bob use RSA public keys with the same modulus n but with different public exponents e_A and e_B. We further assume that Alice and Bob still have their p and q such that $p \cdot q = n$.

   (a) Show that Alice can decrypt messages sent to Bob. (*Hint:* How can Alice calculate d_B?)

   (b) Assume that $\gcd(e_A, e_B) = 1$. This implies that Eve can apply Extended Euclidean gcd algorithm to find integers x and y such that $x \cdot e_A + y \cdot e_B = \gcd(e_A, e_B) = 1$. Now if the message m was sent encrypted as c_A to Alice and as c_B to Bob, how can Eve obtain this message m from $c_A^x \pmod n$ and $c_B^y \pmod n$? Show the steps clearly.

2. **(20 points)** Let C_X^Y be (ad hoc) notation for a public-key certificate[1] of X, signed by Y. Consider a system that consists of the agents P, Q, R, S and only the following three certificates

$$C_Q^P, \; C_R^P, \; C_S^R.$$

   Assume further that everybody knows his own key pair and also P's public key.

   Explain briefly which certificates (and thus: which public keys) are needed by both agents S and Q in order to securely perform the following tasks.

   (a) S sends a confidential message to Q.

   (b) S sends a signed message to Q, and Q verifies this message.

[1] In this exercise, a certificate C_X^Y simply contains the public key of X and the signature of Y on it. For a more detailed discussion, see Wikipedia: `http://en.wikipedia.org/wiki/Public_key_certificate`

3. **(60 points)** Asymmetric cryptography is not used to encrypt the actual text in the message directly; instead, it is used to encrypt something that can be used to derive a symmetric key, which is in turn used to encrypt the actual text. In one of the past lectures, RSA-KEM was explained (see slide 57 for a description). In this exercise, we will use this to look at a somewhat more realistic scenario than we have seen so far.

   Imagine Alice and Bob are using RSA-KEM, and we have intercepted the random value r, encrypted with the following RSA public key (n, e):

$$\begin{aligned} \text{n} &= 902140983721750316999465244309489804973 3 \\ \text{e} &= 65537 \end{aligned}$$

   The RSA-encryption of the random value r is:

$$8972163497987314734169999025202261871445$$

   The session key is a 128-bit AES key. It is derived by taking the first 128 bits of output from SHAKE128(r). AES is a standard block cipher[2] - for this exercise, that is all you need to know about it. AES was used in CBC-mode[3], to produce the following ciphertext:

```
4A 20 EF EE 84 F3 85 BD 00 2D 5B DF 3F 2B F0 C2
9B F0 B3 18 74 6A 08 78 96 13 3D CE D9 17 B7 4F
69 5A 8F 72 B2 71 59 36 EC D5 E7 55 54 3C C3 BB
```

   The following IV was used:

```
55 BB 82 09 2A 18 AA A9 EF 68 0A 6C 2C 94 8F 00
```

   The plaintext is a string encoded using ASCII-encoding. Find the plaintext.

   To help you do this, we have set up the following web pages that enables you to do AES encryption/decryption, SHAKE128, as well as XOR operations on large values:

   - `http://www.sos.cs.ru.nl/applications/courses/security2016/aes/`
   - `http://www.sos.cs.ru.nl/applications/courses/security2016/shake128/`
   - `http://www.sos.cs.ru.nl/applications/courses/security2016/xor/`

   Note that such tools typically require you to convert your input into base-16 (i.e. hexadecimal notation).

   Furthermore, it will be helpful to use tools like Wolfram Alpha[4] to help you with the computations, or self-written code. Make sure that whatever you use is able to support large integers.

   First, write down your strategy in steps, using terms such as RSA, ϕ, factoring, d, CBC, IV, block, XOR, decrypt, encrypt, decode, *etc.* Be as specific as possible. Second, perform each step and make the computation explicit: what tool did you use, and how? What was your input, and what was the result? **Write down your intermediate steps and results!** This will also make it easier to stay organized while solving this exercise.

   Hint: especially for the first step, think like an attacker – everything is allowed!

---

[2] `https://en.wikipedia.org/wiki/Advanced_Encryption_Standard`
[3] `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`
[4] `http://www.wolframalpha.com/`