Security

Assignment 10, Wednesday, November 25, 2015

Handing in your answers: the full story, see

`http://www.sos.cs.ru.nl/applications/courses/security2015/exercises.html`

Briefly,

- submission via Blackboard (`http://blackboard.ru.nl`);
- one single pdf file;
- make sure to write all names and student numbers and the name of your teaching assistant (Brinda or Joost).

Deadline: Thursday, December 3, 24:00 (midnight) sharp!

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

- relate the security goals to public-key cryptography;
- understand how to combine symmetric and asymmetric crypto in protocols;
- understand how a blind RSA signature works;
- understand the problem with reusing primes.

Note: You may use a calculator (like `bc` under Linux) but you have to write down and explain intermediate steps.

Marks: You can score a total of 100 points.

1. **(10 points)** Suppose Alice wants to send a message m to Bob using public key cryptography. How can this message be sent such that the following requirements are satisfied:

   - no-one but Bob can read the message (Confidentiality),
   - Bob is sure that Alice was the sender of the message (Authenticity),
   - Bob is sure the message has not been tampered during the transit (Integrity),
   - Alice cannot deny later that she sent the message (Non-repudiation).

   Explain the tasks both Alice and Bob have to perform to achieve all this (using the terms *Encrypt*, *Decrypt*, *Sign* and *Verify*). Also, describe the computation they would need to perform if they were to use RSA.

   Use the values $\langle e_A, d_A \rangle$ for Alice's key-pair and $\langle e_B, d_B \rangle$ for Bob's key-pair. You can assume that Alice and Bob have access to each other's public keys.

2. **(40 points)** In this exercise, we have another look at RSA, this time in the context of signatures. For each question, give intermediate steps to show how you got your results.

   (a) Bob has chosen primes $p = 17$ and $q = 23$, compute n and $\varphi(n)$.

   (b) Take $e = 5$ and, applying the *extended Euclidean algorithm*, compute d such that $d \cdot e = 1 \pmod{\varphi(n)}$. Now (n, e) and (n, d) are Bob's public and private keys, respectively.

   (c) Compute Bob's signature on $m = 28$.

   (d) Verify the signature using Bob's public key and the *square-and-multiply* method. If it does not verify, also consider carefully checking (c) again.

(e) Now Alice wants Bob to sign $m = 28$ **blindly**[1]. Take random $r = 11$ and compute m', i.e., the message that Alice sends to Bob to sign.

(f) Compute Bob's signature on m'.

(g) Un-blind the signature from (f) and compare your results with (c).

3. **(30 points)** Public key protocols.

   For this question, consider the following protocol (refer to the lecture slides for the notation):

$$\begin{array}{lrcl} 1. & A \longrightarrow B & : & N_A, \{A, K_{AB}\}_{e_B} \\ 2. & B \longrightarrow A & : & K_{AB}\{N_A, m\} \end{array}$$

   Alice and Bob use this protocol so that Bob can share the secret message m with Alice. Assume that Alice and Bob each have their own private keys d_A and d_B, and everyone knows their public keys e_A and e_B.

   (a) As you can see, Alice first sends Bob a symmetric key K_{AB}, encrypted with Bob's public key e_B. Bob then uses this symmetric key to encrypt m for Alice. Why would Bob do that, instead of just encrypting m with Alice's public key?

   (b) This protocol can be (ab)used to make Bob send m to Eve, the attacker. Show how this can be achieved (use arrow notation).

   (c) Show how this can be fixed (using asymmetric cryptography, but still encrypting m with a symmetric key).

4. **(20 points)** It is generally hard to factor integers; this is why RSA is secure. It is however easy to find *common factors* of two integers. This was used in independent research by two groups in 2012 to factor various SSL, SSH and GnuPG keys [2,3], and was used as a starting point to find various private keys of Taiwanese citizen smartcards[4].

   For this exercise, it is useful to be able to compute the GCD of two large values in some automated way. You can do this using your favorite programming language[5], but we have also set up an online page to make this more convenient, at the following URL. Note that many other online GCD calculators do not support integers of this size.

   - `http://www.sos.cs.ru.nl/applications/courses/security2015/gcd/`

   Find all common factors in the following RSA moduli $N = p \cdot q$. Note that some of the moduli N may not have common factors with any of the others, so not all of them can be factored this way. Use this to factor those values N (*i.e.* find both p and q).

   - 153593241046674892978867376676801703195333499261944069748317
   - 595581987651106688365284842778515858399666547859870373300567
   - 732521324063413291774595255009269986704084399047286433357607
   - 697998237255232517803133139640937207091669333334886072165381
   - 665759389457622825753076124570026166878147870317677657070179
   - 115583164418843644012534609117494469512367874677960825637222 9
   - 176294427788887166758409622538881387638478405478915857712513
   - 592339248856319601455928821705423109007342115448431777433343

[1] See Slides 56 and 57 of the lecture or Wikipedia: `http://en.wikipedia.org/wiki/Blind_signature`.
[2] `https://factorable.net/`
[3] `http://eprint.iacr.org/2012/064`
[4] `http://cr.yp.to/papers.html#smartfacts`
[5] See for example the `gcd` function in Python's `fractions` module