Network Security Technology Spring, 2019
Tutorial 4, Week 5 (March 27) LIU Zhen
Due Date: April 3

Questions:

1. **Hash Function (50 Points)** Assume prime $p = 13$ and a generator $g = 7$.

   (a) Find two distinct positive integers x and y such that:

   $$g^x \mod p = g^y \mod p$$

   (b) Given the question 1a above, explain why $h(x) = g^x \mod p$ is not a good hash function.

   (c) Does the hash function $h(x) = g^x \mod p$ satisfy one-wayness, under the condition that p is a large prime?

   Solution

   (a) We first set x to any positive integer $<= 12$, for example $x = 1$. From Euler's theorem we know that for any $y = x + k \cdot (p - 1)$, k an integer, we have $g^x \mod p = g^y \mod p$. We can thus set y for example to $y = x + 1 \cdot 12 = 13$ and then easily verify that

   $$g^x \mod p = 7^1 \mod 13 = 7$$
   $$g^y \mod p = 7^{13} \mod 13 = 7$$

   (b) From question 1a we have seen that it is trivial, given an input x, to create an input y such that $h(x) = h(y)$, and $h(\cdot)$ thus does not satisfy weak collision resistance (i.e., given x, it should be hard to find a y such that $h(x) = h(y)$).

   (c) Yes, under the assumption that the discrete logarithm problem is hard, $h(x) = g^x \mod p$ satisfies one-wayness.

2. **Digital Certificates (50 Points)**

   (a) Why can a public key not just be transmitted through email or be posted on a website?

   (b) A certificate does not necessarily have to be signed directly by a CA, there is something called a *certificate chain*. Can you imagine what a certificate chain is?

   (c) Who signs the certificate of a CA?

   Solution

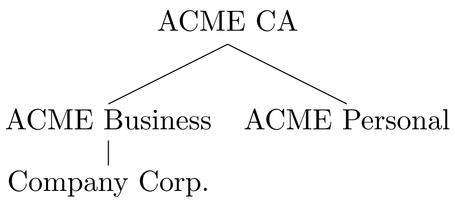
   (a) Both email and web browsing are not normally secure and can easily be compromised. Email, for example, typically travels through several mail servers along the way, and anyone of them could replace the public key with a different one. Web browsing can also be intercepted and modified by an attacker.

   (b) For organizational reasons it is sometimes desirable that not all certificates are signed using the master key of a CA. Instead, there exists intermediate certificates which are used to certify keys.
   For example, imagine a company named *ACME CA* possesses a master key and the

corresponding certificate. It then creates two organizational units *ACME Business* and *ACME Personal*. Both of these two units have their own private/public key-pair.
The parent *ACME CA* now creates a certificate for each of the two units, certifying that they own their respective public key.
If a company *Company Corp.* wants to certify their public key, they contact *ACME Business*, pay lots of money (digital certificates are sometimes called the "most expensive bits in the internet") and then get a certificate from *ACME Business* certifying that they own the public key.
If John then wants to verify the public key of *Company Corp.* he does the following:

- Using the certificate of *ACME Business* he verifies that *ACME Business* did indeed sign the certificate of *Company Corp.*.
- Using the certificate of *ACME CA* John verifies that the certificate of *ACME Business* was indeed signed by *ACME CA*.
- Because John trusts *ACME CA*, he therefore also trusts *ACME Business* and by extension now also *Company Corp.*.

John in fact verified the certificate chain up to a *trust anchor* (i.e., *ACME CA*). Graphically, the chain would look like this:



(c) The CA itself. CA certificates are so-called *self-signed* certificates, because they use their own private key to sign their certificate.