Cryptography and Security — Final Exam

Serge Vaudenay

26.1.2022

- duration: 3h
- no documents allowed, except one 2-sided sheet of handwritten notes
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will **not** answer any technical question during the exam
- readability and style of writing will be part of the grade
- answers should not be written with a pencil

1 Diffie-Hellman vs ElGamal

We consider a group (with multiplicative notation) of prime order q generated by a given element g. The Diffie-Hellman protocol was specified as follows in the course:

- Alice picks $x \in \mathbf{Z}_q^*$ uniformly, computes $X = g^x$, and sends X to Bob.
- Bob checks that $X \in \langle g \rangle - \{1\}$ and aborts if it is not the case.
- Bob picks $y \in \mathbf{Z}_q^*$ uniformly, computes $Y = g^y$, and sends Y to Alice.
- Bob computes $K = \mathsf{KDF}(X^y)$ and uses it as the output.
- Alice checks that $Y \in \langle g \rangle - \{1\}$ and aborts if it is not the case.
- Alice computes $K = \mathsf{KDF}(Y^y)$ and uses it as the output.

The ElGamal cryptosystem was specified as follows in the course:

- The receiver generates his key pairs as follows. He picks $x \in \mathbf{Z}_q$ uniformly as a secret decryption key and computes the public encryption key $y = g^x$.
- The sender who wants to encrypt $m \in \langle g \rangle$ with key y picks $r \in \mathbf{Z}_q$ and computes $u = g^r$ and $v = my^r$. The ciphertext is (u, v).
- The receiver who wants to decrypt (u, v) with key x computes $m = vu^{-x}$.

Q.1 Explain how the ElGamal cryptosystem is built from the Diffie-Hellman protocol. For this, explain each element of the ElGamal cryptosystem and how it comes from the Diffie-Hellman protocol.

Q.2 In the Diffie-Hellman protocol, x and y are selected in $\mathbf{Z}_q^*$. What is the problem if we select them in $\mathbf{Z}_q$? Explain the security threat.

Q.3 In the ElGamal cryptosystem, x and r are instead selected in $\mathbf{Z}_q$. Why isn't it a problem?

2 Immunity Certificate

Disclaimer: this exercise was imagined in 2021. This is a work of fiction. Any similarity to actual persons, living or dead, or actual events, is purely coincidental.

A virus is currently striking many countries. Fortunately, there exists an effective vaccine, people who caught the virus are immune, and there exists a reliable infection test. Hence, countries can assess if people are immune in three categories: *vaccinated*, *healed*, and (negatively) *tested*. In the country Weirdoland, the Badass Office for Public Health (BOPH) — the local health authority — has deployed a way to give digital immunity certificates to people and a way to show them from inspection and to verify them.

The immunity certificate consists of a message encoding several pieces of information together with a digital signature using ECDSA. The information includes:

- the signing authority and how to verify the immunity certificate,
- the identity of the holder (full name and date of birth),
- the type of immunity (*vaccinated*, *healed*, or *tested*),
- and extra information about immunity which contains a reference date.

If *vaccinated*, this includes the date of the vaccine. If *healed*, this includes the date of the positive diagnosis. If *tested*, this includes the date of the test. An immunity certificate can be issued through a specific online platform with appropriate access control by vaccine/test centers and medical doctors. Signing keys are often changed and a PKI certificate for the verification key is provided online by BOPH. The validity period of a certificate is announced by BOPH following a policy. It depends on the type of immunity certificate and the reference date.

BOPH provides two apps which people install on their smartphones. One app is the Certshow and the other is Certscan. Both frequently connect to the BOPH server to retrieve existing PKI certificates and possible lists of revoked certificates. Indeed, when it appears that a signing key leaked or was misused, the PKI certificate of the corresponding verification key can be revoked and the immunity certificates which have been signed by this key are not valid any more. Immunity certificates which have a correct signature, a correct PKI certificate which is also non-revoked, and which are still in their validity period are called *valid*. Certshow allows people to store their immunity certificate, to show it in the form of a QR code, and to verify its validity. The QR code encodes all information. Certscan allows to scan the QR code of an immunity certificate, to decode it, to display the data it contains, and to check its validity.

Q.1 The Weirdoland government decided that restaurants should only give access to customers holding a valid immunity certificate. Why are restaurants asking the ID of customers in addition to scanning their QR code?
Why should restaurants use their Certscan app instead of relying on the Certshow app of their customers to verify the immunity certificate?

Q.2 Mr Antigov wants to enter a restaurant without following the correct procedure (i.e. he had no vaccine, no recent test, and was not sick before). Suggest 3 ways to do so and sort them in increasing difficulty. (Note: the difficulty of some methods may be subjective, it is recommended to justify the order.)

Q.3 The Badass Officer for Privacy Protection (BOPP) complains that some medical information about the holder leaks from the certificate. For that, BOPH proposes a “blind certificate” which only says “valid” or not, together with the identity. The way to get this blind certificate consists of having Certshow to send the full certificate to BOPH and to get in return an immunity certificate of a new type (the type *blind*) which is valid for only one day. This question is split into 3 subquestions.
Compare the privacy issues with the regular immunity certificate and the blind certificate.

In parallel, the Weirdoland government decided that immunity certificate of type *tested* should not be allowed any more in restaurants. **Why** doesn't the blind certificate work in this case?
Next, the Weirdoland government requires that people going to bars must have a valid immunity certificate of type other than *tested* and be tested. **How** could we update the immunity certificate and the blind certificate to enforce that?

Q.4 BOPH claims that they store no private information. They claim that immunity certificates are decentralized and only stored on the holder's smartphone. However, when Mrs Jab had her last dose, she received a certificate containing her previous data and the new injection. Do you believe that BOPH really stores no information? Explain why.

3 Blind Schnorr Signatures

We recall the Schnorr signature scheme (with a slight difference). We assume a group (with multiplicative notations) of prime order p generated by an element G. For key generation, the key holder selects $x \in \mathbf{Z}_p$ as a signing key and publishes $X = G^x$ as a verifying key. A valid signature for a message m is a pair (R, s) made of a group element R and an integer s which satisfies $G^s = X^{H(R,m)}R$. To sign m, the signer picks uniformly a random $r \in \mathbf{Z}_p$, computes $R = G^r$ and $s = (x \cdot H(R, m) + r) \bmod p$. We adapt the scheme into a *blind* signature scheme where the signer signs in a blind way, i.e. he does not see what he is signing but he signs one and only one document. In the blind signature protocol, the enquirer has (X, m) as input and the signer has x as input (in addition to the public group parameters such as p and G). Then, the protocol works as follows:

- The enquirer sends a signature request to the signer.
- The signer picks $r \in \mathbf{Z}_p$ uniformly at random and sends back $R = G^r$.
- The enquirer picks $\alpha, \beta \in \mathbf{Z}_p$ uniformly at random, computes $R' = G^\alpha X^\beta R$ and $c = (H(R', m) + \beta) \bmod p$, and sends c to the signer.
- The signer responds with $s = (c \cdot x + r) \bmod p$.
- The enquirer sets $s' = (s + \alpha) \bmod p$ and checks that (R', s') is a valid signature for m.

The security is such that by querying the signer ℓ times, the enquirer should not be able to produce $\ell + 1$ different triplets (m_i, R'_i, s'_i) such that (R'_i, s'_i) is a valid signature of m_i. We model the signer with secret x by the following stateful oracle which takes as input a *session identifier* sid:

```
OSig(sid, $c$):
1: if state{sid} does not exist then
2:     pick $r \in \mathbf{Z}_p$ uniformly
3:     set $R \leftarrow G^r$
4:     set state{sid} $\leftarrow r$
5:     return $R$
6: end if
7: if state{sid} $= \perp$ then abort
8: set $r \leftarrow$ state{sid}
9: set $s \leftarrow (c \cdot x + r) \bmod p$
10: set state{sid} $\leftarrow \perp$
11: return $s$
```

Here, state is an associative array (a.k.a. a dictionary). $\mathsf{OSig}(\mathsf{sid}, c)$ can be queried the first time with a dummy input $c = \perp$ (which is unused) to get R, then the second time with $c \in \mathbf{Z}_p$ to get s. Both oracle calls with the same sid count for *one* signature query.

Q.1 Define correctness for blind signatures and prove that the above scheme is correct.

Q.2 Define the blind signature security with ℓ sessions by using a game Γ_ℓ and properly define the advantage of an adversary.

Q.3 We consider an adversary making ℓ oracle calls normally as follows:

```
$\mathcal{A}(p, G, X)$:
1: for $i = 0$ to $\ell - 1$ do
2:     $R_i \leftarrow \mathsf{OSig}(i, \perp)$
3:     pick $\alpha_i, \beta_i \in \mathbf{Z}_p$ uniformly
4:     set $R'_i \leftarrow G^{\alpha_i} X^{\beta_i} R_i$
5: end for
6: [to be continued...]
```

After that, the adversary selects 2ℓ pairwise different random messages m_i^b for $i = 0, \ldots, \ell-1$ and $b = 0, 1$ such that $H(R'_i, m_i^0) \bmod p \neq H(R'_i, m_i^1) \bmod p$ for every i. Then, the adversary defines $c_i^b = (H(R'_i, m_i^b) + \beta_i) \bmod p$, $z_i = \frac{2^i}{c_i^1 - c_i^0} \bmod p$, and

$$z_\ell = -\sum_{i=0}^{\ell-1} z_i . c_i^0$$

Note that in notations m_i^b and c_i^b, the superscript b represents an index and not an exponent.

Prove that for any set of ℓ bits $b_0, \ldots, b_{\ell-1}$, we have

$$z_0 . c_0^{b_0} + \ldots + z_{\ell-1} . c_{\ell-1}^{b_{\ell-1}} + z_\ell = b_0 + 2b_1 + \ldots + 2^{\ell-1} . b_{\ell-1}$$

Q.4 We first assume that the α_i and β_i that the adversary selected are all equal to 0 for simplicity. Next, the adversary selects a random message m_ℓ which is different from the 2ℓ previous ones. The adversary defines $R'_\ell = {R'_0}^{z_0} \cdots {R'_{\ell-1}}^{z_{\ell-1}} X^{-z_\ell}$ and $c_\ell = H(R'_\ell, m_\ell) \bmod p$. We assume that $\ell > \log_2 p$ so that the adversary can make the binary decomposition $c_\ell = \sum_{i=0}^{\ell-1} 2^i b_i$ in ℓ bits b_i. The adversary sets $m_i = m_i^{b_i}$ and $c_i = c_i^{b_i}$. Finally, the adversary runs $s'_i \leftarrow \mathsf{OSig}(i, c_i)$ for $i = 0, \ldots, \ell - 1$.

Show that the adversary can find s'_ℓ and make $\ell + 1$ signed messages.

Q.5 In order to make the attack undetectable, we now assume that the α_i and β_i are totally random. The adversary further selects $\alpha_\ell, \beta_\ell \in \mathbf{Z}_p$ at random. The adversary now defines

$$R'_\ell = G^{\alpha_\ell} X^{\beta_\ell} {R'_0}^{z_0} \cdots {R'_{\ell-1}}^{z_{\ell-1}} X^{-z_\ell - \beta_0 z_0 - \cdots - \beta_{\ell-1} z_{\ell-1}}$$

Show that the previous attack adapts with a new s'_ℓ.