Cryptography and Security — Final Exam
Solution

Serge Vaudenay

25.1.2012

- duration: 3h00
- no documents are allowed
- a pocket calculator is allowed
- communication devices are not allowed
- exam invigilators will not answer any technical question during the exam
- answers to every exercise must be provided on separate sheets
- readability and style of writing will be part of the grade
- do not forget to put your name on every sheet!

1 Security Issue in ECDSA

In Sony PS3, the bootup code can be changed when it comes from a valid signature from the manufacturer. The signature scheme is ECDSA. We briefly recall the scheme here.

The public key consists of a prime number n, a finite field $\mathsf{GF}(q)$, an elliptic curve over this field, a generator G of order n, and another point Q. The secret key is an integer $d \in \mathbf{Z}_n^*$ such that $Q = dG$. To sign a message M, the signer picks $k \in \mathbf{Z}_n^*$, computes the point $(x_1, y_1) = kG$, then $r = \bar{x}_1 \bmod n$ given a function $x \mapsto \bar{x}$ from $\mathsf{GF}(q)$ to $\mathbf{Z}$, and finally $s = \frac{H(M)+dr}{k} \bmod n$ given a hash function H. If $r = 0$ or $s = 0$, the signer restarts the computation until $r \neq 0$ and $s \neq 0$. The signature is the pair (r, s). To verify a signature (r, s) for a message M, the verifier checks that $Q \neq \mathcal{O}$, that Q lies on the curve, that $nQ = \mathcal{O}$, and that $r \in \mathbf{Z}_n^*$. Then, he computes $u_1 = \frac{H(M)}{s} \bmod n$, $u_2 = \frac{r}{s} \bmod n$, and $(x_1, y_1) = u_1 G + u_2 Q$, and finally checks that $r = \bar{x}_1 \bmod n$.

Q.1 ECDSA manipulates values of different *types* such as *points*, *field elements*, *integers*, etc. What are the types of k, r, s, y_1, $H(M)$? What is $\mathcal{O}$?

k, r, s, $H(M)$ are integers (taken modulo n). (Strictly speaking, $H(M)$ is a bitstring which shall be converted into an integer.) y_1 is a field element. $\mathcal{O}$ is the point at infinity, the neutral element of the elliptic curve.

Q.2 What kind of finite fields can we use in practice? Cite at least two and briefly explain how to perform computations in these structures.

There are two types of finite fields which are popular for ECDSA: the field $\mathbf{Z}_q$ when q is a prime number and the field $\mathsf{GF}(q)$ when q is a power of 2. In the former case, we manipulate integers and reduce them modulo q. In the latter case, we manipulate polynomials with coefficients modulo 2 and reduce them modulo a reference irreducible polynomial.

Q.3 If a key is valid and a signature is produced by the signing algorithm, show that the verification algorithm will accept the signature.

If the key is valid, we have $Q = dG$ and G is on the curve. So, Q lies on the curve. Then, $nQ = n(dG) = d(nG)$. Since G has order n, we have $nG = \mathcal{O}$. Furthermore, $d\mathcal{O} = \mathcal{O}$. So, $nQ = \mathcal{O}$. Then, since $d \in \mathbf{Z}_n^*$, $dG \neq \mathcal{O}$. So, $Q \neq \mathcal{O}$. Q passes all verifications.
Since r is the result of a modulo n computation, we have $r \in \mathbf{Z}_n$. Since $r = 0$ is excluded from the signature generation and n is prime, we have $r \in \mathbf{Z}_n^*$.
Finally, we have

$$u_1 G + u_2 Q = \left(\frac{H(M)}{s} G + \frac{r}{s} d\right) G = kG$$

Due to the signature generation, we know that $(x_1, y_1) = kG$ is such that $r = \bar{x}_1 \bmod n$, so the signature is valid.

Q.4 Why is it hard to recover the secret key given the public key?

Because ECDSA uses elliptic curves on which it is hard to compute the discrete logarithm. Computing d given Q and the group material is exactly the problem of computing the discrete logarithm of Q.

Q.5 For some reasons, the manufacturer produced signatures for different codes using the same random k. Given two codes M and M' and their signatures (r, s) and (r', s'), respectively, show that an adversary can recover d.

Let $(x_1, y_1) = kG$. We have $r = \bar{x}_1 \bmod n = r'$ and

$$s = \frac{H(M) + dr}{k} \bmod n \qquad s' = \frac{H(M') + dr}{k} \bmod n$$

So,

$$\frac{H(M)}{s} + d\frac{r}{s} \equiv \frac{H(M')}{s'} + d\frac{r}{s'} \pmod{n}$$

thus

$$d = \frac{sH(M') - s'H(M)}{(s' - s)r} \bmod n$$

which can be computed.

2 Hard Disk Encryption

A hard disk is made of sectors of various length (e.g., 4 096 bytes). We want to encrypt data on the disk using the following constraints:

- we want security (no information leakage);
- we want to use symmetric encryption with a single secret key K for the entire hard disk;
- we prefer to use a block cipher;

– we want to be able to access or update a random piece of information without having to process an entire sector; and
– encryption should be "in-place", i.e., ciphertexts must not be larger than plaintexts.

Q.1 Let ℓ be the block length in bits for the block cipher. We assume that each sector has a length L which is a multiple of ℓ. If i is the index of a sector and j is the index of a block in the sector, we let $x_{i,j}$ denote the plaintext block we would have had at position (i,j) with an unencrypted hard disk. Further, we let $y_{i,j}$ denote the ciphertext block we have in the encrypted hard disk.
What is the value of ℓ in the case of AES? Which mode of operation could we propose to meet all the requirements?

$\ell = 128$ bits.
We can propose a CTR mode of AES where

$$y_{i,j} = x_{i,j} \oplus \mathsf{Enc}_K(i,j)$$

This uses a block cipher, we can access to data randomly, and the length is preserved. We could not use the CBC, OFB, or CFB modes which require processing more blocks to access to a random one. We could not use the ECB mode for security reasons. (We can see when two plaintext blocks are equal by comparing the ciphertext blocks, which leaks information.)

Q.2 We still assume that each sector has a length L which is a multiple of ℓ. We define the XTS mode by having a key K composed of two subkeys $K = (K_1, K_2)$ and by having

$$y_{i,j} = \mathsf{Enc}_{K_1}(x_{i,j} \oplus t_{i,j}) \oplus t_{i,j} \quad \text{with} \quad t_{i,j} = \alpha^j \times \mathsf{Enc}_{K_2}(i),$$

where α is a constant and $\alpha^j \times u$ is defined by $\mathsf{GF}(2^\ell)$ operations. Explain how to decrypt and show that it meets all requirements. What is the problem if L is not a multiple of ℓ?

Decryption is using a similar formula:

$$x_{i,j} = \mathsf{Dec}_{K_1}(y_{i,j} \oplus t_{i,j}) \oplus t_{i,j} \quad \text{where} \quad t_{i,j} = \alpha^j \times \mathsf{Enc}_{K_2}(i)$$

Indeed,

$$\mathsf{Dec}_{K_1}(y_{i,j} \oplus t_{i,j}) \oplus t_{i,j} = \mathsf{Dec}_{K_1}(\mathsf{Enc}_{K_1}(x_{i,j} \oplus t_{i,j}) \oplus t_{i,j} = x_{i,j}$$

The problem if L is not multiple of ℓ is that there remains an incomplete block and it is not clear how to encrypt it.
It is (hopefully) secure, using symmetric encryption with a single key $K = (K_1, K_2)$, using a block cipher, we can have random access to a block, and it is length-preserving. So it meets all requirements.

Q.3 We assume that there is at most one incomplete block per sector and that the size of the sector is $L \geq \ell$. We assume that there are n_i blocks in sector i, that $j \in \{1, \ldots, n_i\}$, and that the incomplete block (if any) is the one of index n_i. We use the XTS mode from the previous question with the *ciphertext stealing* technique for the special blocks of index $n_i - 1$ and n_i. Ciphertext stealing consists of using a special rule to compute y_{i,n_i-1} and y_{i,n_i} from x_{i,n_i-1} and x_{i,n_i}:

– if the size of x_{i,n_i} is ℓ, proceed as in the previous question;
– otherwise, split $\mathsf{Enc}_{K_1}(x_{i,n_i-1} \oplus t_{i,n_i-1}) \oplus t_{i,n_i-1}$ into $y_{i,n_i} \| u$, where y_{i,n_i} is an incomplete block, having the same length as x_{i,n_i}, and u is the leftover information in the block. Then, $y_{i,n_i-1} = \mathsf{Enc}_{K_1}((x_{i,n_i} \| u) \oplus t_{i,n_i}) \oplus t_{i,n_i}$. (The $\|$ symbol denotes the concatenation operation.)

Explain how to decrypt and show that it meets all requirements.

This system meets all requirements since it uses a block cipher, keeps the data size, and can access data randomly. In the worst case, two blocks will be processed to recover one.
To decrypt $y_{i,j}$ for $j < n_i - 1$ or y_{i,n_i} of length ℓ, we proceed as before. To decrypt y_{i,n_i-1} and y_{i,n_i} when y_{i,n_i} has length smaller than ℓ, we proceed as follows: split $\mathsf{Dec}_{K_1}(y_{i,n_i-1} \oplus t_{i,n_i-1}) \oplus t_{i,n_i-1}$ into $x_{i,n_i} \| u$ where x_{i,n_i} has the same length as y_{i,n_i} and u is the leftover information in the block, then $x_{i,n_i-1} = \mathsf{Dec}_{K_1}((y_{i,n_i} \| u) \oplus t_{i,n_i}) \oplus t_{i,n_i}$.

3 Attack on 2K-3DES

This exercise is based on "On the security of multiple encryption" by Merkle and Hellman, Communications of the ACM, Vol. 24(7), July 1981.

Q.1 What are the block length and the key length in DES? What is the complexity of key recovery exhaustive search in terms of *data*, *known plaintexts* versus *chosen ciphertexts*, *memory*, and *time*?

Blocks have 64 bits. The key has 56 effective bits.
With a single plaintext-ciphertext pair (x, y) with a known plaintext, it is enough to characterize the correct key as no wrong key shall be consistent with probability $\left(1 - 2^{-64}\right)^{2^{56}} \approx e^{-2^{-8}}$ which is very close to 1. The average complexity is of 2^{55} trials with a small memory (just enough to store the data and a counter).

Q.2 Double DES is defined by

$$y = \mathsf{DES}_{K_1}\left(\mathsf{DES}_{K_2}(x)\right).$$

Explain how the meet-in-the-middle attack works. What is its complexity in terms of *data*, *known plaintexts* versus *chosen ciphertexts*, *memory*, and *time*?

We now need two pairs (x_i, y_i), $i = 1, 2$ to characterize the correct key uniquely. With 2 known plaintexts, we prepare a dictionary of 2^{56} records $(\mathsf{DES}_b^{-1}(y_1), b)$ for all b. Records are sorted by their first two values. The dictionary takes 8×2^{56} bytes. There would be tricks to shrink it a bit but the order of magnitude should stay 2^{56}. Then, for all a we compute $(\mathsf{DES}_a(x_1))$ and check if this is in the dictionary. When it is, b is given by the dictionary and we check if $y_2 = \mathsf{DES}_b\left(\mathsf{DES}_a(x_2)\right)$. If it matches, then $K_1 = a$ and $K_2 = b$ is the correct key. The time complexity consists of 4×2^{56} DES encryptions. Again, there would be tricks to reduce it a bit but the order of magnitude should stay 2^{56}.

Q.3 Two-key triple DES is defined by

$$y = \mathsf{DES}_{K_1}\left(\mathsf{DES}_{K_2}^{-1}\left(\mathsf{DES}_{K_1}(x)\right)\right).$$

By preparing a dictionary of all $(\mathsf{DES}_k^{-1}(0), k)$ pairs, show that we can break this using many chosen plaintexts and within a time/memory complexity similar to in the previous question.
Hint: Make an exhaustive search on K_1, i.e., guess K_1, do something, then use the dictionary to recover K_2.

For each a we compute $x = \mathsf{DES}_a^{-1}(0)$ and use x as a chosen plaintext. We obtain y. Then, we check if $\mathsf{DES}_a^{-1}(y)$ is in the dictionary. If it is, it means that $\mathsf{DES}_a^{-1}(y) = \mathsf{DES}_b^{-1}(0)$ for some b and it gives b. Clearly, x encrypts to y with key (a, b). With a previous plaintext-ciphertext pair we can check if this key is correct. Clearly, when a becomes equal to K_1 (which would happen after an average number of trials equal to 2^{55}), this attack recovers K_2. So, it works with a number of DES operations equal to 3×2^{55}, 2^{55} chosen plaintexts, and a dictionary of 2^{56} entries.

4 Collisions with a Subset

In a classroom, we have x female students and y male students. We assume that their birthday is uniformly distributed in a calendar of N possible dates, e.g., $N = 365$.

Q.1 Let p_{xx} denote the *exact* probability, that there are two different female students with the same birthday. Express p_{xx} in terms of N and x.

The probability that we have no collision is

$$1 \cdot \left(1 - \frac{1}{N}\right) \cdot \left(1 - \frac{2}{N}\right) \cdots \left(1 - \frac{x-1}{N}\right) = \frac{N!}{N^x (N-x)!}$$

So, the probability to have a female-female collision is

$$p_{xx} = 1 - \frac{N!}{N^x (N-x)!}$$

Q.2 Let $p_{xy|\neg xx}$ denote the *exact* probability, that there is at least one female-male pair of students who share the same birthday conditioned to that female students have pairwise different birthdays. Express $p_{xy|\neg xx}$ in terms of N, x, and y.

The probability of no female-male collision under the condition that there are exactly x female birthdays is $(1 - \frac{x}{N})^y$. So, the probability to have a male-female collision is

$$p_{xy|\neg xx} = 1 - \left(1 - \frac{x}{N}\right)^y$$

Q.3 Show that $p_{xy|\neg xx} \approx 1 - e^{-\frac{xy}{N}}$.

We have

$$p_{xy|\neg xx} = 1 - e^{y \log(1-\frac{x}{N})}$$

Since $\log(1-\varepsilon) \approx -\varepsilon$, *we obtain the result.*

Q.4 Based on the previous computations, what is the *exact* probability $p_{x\star}$ that at least one female student shares the same birthday with another student (either female or male)?

It is

$$p_{x\star} = p_{xx} + (1 - p_{xx})p_{xy|\neg xx} = 1 - \frac{N!}{N^x(N-x)!} + \frac{N!}{N^x(N-x)!}\left(1 - \left(1 - \frac{x}{N}\right)^y\right)$$

Q.5 Show that $p_{x\star} \approx 1 - e^{-\frac{x(x+2y)}{2N}}$.
Hint: $p_{xx} \approx 1 - e^{-\frac{x^2}{2N}}$.

We have

$$p_{x\star} = p_{xx} + (1 - p_{xx})p_{xy|\neg xx} \approx 1 - e^{-\frac{x^2}{2N}} + e^{-\frac{x^2}{2N}}\left(1 - e^{-\frac{xy}{N}}\right) = 1 - e^{-\frac{x(x+2y)}{2N}}$$

Q.6 In a community of n_u users each having a password, we assume that there is a public directory for the hash of the passwords. We consider an attacker who tries to find password matches with the existing database of n_u password hashes. He is allowed to try n_t many random passwords and hash them. We say that he succeeds if he gets any match. That is to say, he succeeds if either he finds at least one password with a hash in the directory, or if he finds two users having the same password hash in the directory. What is his success probability?

We take $x = n_u$ *and* $y = n_t$. *The attacker succeeds if either there is a collision between the list* x *and the list* y, *or there is a collision inside the list* x. *If the range of the hash function is* N, *this probability is thus* $p_{x\star}$.

5 Secure Communication Across the Röstigraben

Warning: this exercise asks you to propose a real solution for a real problem. You are requested to precisely describe your proposed solution so that we could assess on correctness, feasibility, efficiency, and security. Take this exercise as if it was for a hiring interview for an engineer position.

You want to communicate securely with your friend in Zurich, but you forgot to prepare for it the last time you met. Fortunately, you are making MSc studies with courses in cryptography, so you are familiar with communication systems and computers, and so is your friend.

Q.1 How would you generate a private/public key pair on your computer?

You could use a software such as GPG to set up your own key.
You could also try to develop your own RSA at your own risks. For instance, you could generate two prime numbers p and q large enough, compute $n = p \times q$ and pick a random e until it is coprime with $(p-1)\times(q-1)$. The public key would be $K_p^L = (n, e)$ while the private one would be $K_s^L = (n, d)$ with $d = e^{-1} \bmod ((p-1)(q-1))$.

Q.2 How would you and your friend *securely* exchange your public keys?

You could exchange your public keys over the Internet (e.g., by email), then authenticate them using a second channel.
To authenticate your public key K_p^L to your counterpart, you can pick a random string r_L, compute $\mathsf{SHA1}(K_p^L \| r_L)$, take the 80 leftmost bits σ_L, send r_L by email. You friend would do the same computation to obtain σ'_L. Then, you call each other over the telephone and recognize each other by your voice. Them, you would spell σ_L to your friend who will compare it with σ'_L. If they match, then your key is authenticated. The authentication of his public key K_p^Z would be similar: you receive r_Z by email, compute $\mathsf{SHA1}(K_p^Z \| r_Z)$, take the 80 leftmost bits σ'_Z. When you call each other over the phone, he would spell σ_Z and you would have to check $\sigma_Z = \sigma'_Z$.
There are better interactive protocols using shorter strings than 80 bits and which are called SAS-based authentication protocols. One advantage of the above one is that it is essentially non-interactive. It can work even with a very slow channel instead of a telephone. For instance, you could exchange σ's by regular mail with enough handwriting so that you could authenticate the message. Concrete implementations heavily depends on the properties on this second channel.
If voice recognition is not enough, you could also transmit the 80-bit σ_Z by several channels (telephone, email, sms, regular mail) and assume that no adversary will be able to attack all channels.

Q.3 How would you use public keys to set up a symmetric key with your friend?

You could use the public keys to exchange symmetric keys.
To exchange symmetric keys, you would exchange some secret random numbers by encrypting them with each other's public key, then XOR them, and use the result as a seed to generate symmetric keys. For instance, you pick x_L using a pseudorandom generator with a seed set to some secret stuff with large enough entropy and send $y_L = \mathsf{Enc}_{K_p^Z}(x_L)$ to your friend. You receive y_Z and compute $x_Z = \mathsf{Dec}_{K_s^L}(y_Z)$. Then, $s = x_L \oplus x_Z$ can be used as a seed for a pseudorandom generator to generate a symmetric key K. As for pseudorandom generator, you can use

$$\mathsf{SHA1}(1\|\mathsf{stuff})\|\mathsf{SHA1}(2\|\mathsf{stuff})\|\cdots$$

With GPG you would just send encrypted emails with public key K_p^Z You can decrypt the email from your counterpart using K_s^L.
With your hand-made RSA you would take a random number w which is twice larger than your friend's modulus (to avoid biases in distributions), and reduce it modulo that number to get x_L. You would compute $y_L = x_L^{eZ} \bmod N_Z$ and send it by email. As your hand-made implementation of RSA would essentially be the plain RSA cryptosystem, it would be wise to throw away your RSA keys after the protocol completes in order to avoid all the problems of plain RSA.

Q.4 How would you implement a secure communication channel based on this key?

A key K would have two parts $K = (K_1, K_2)$. Let $c-1$ be the number of exchanged messages. To send the cth message x, your can send $\mathsf{Enc}_{K_1}(\mathsf{Mac}_{K_2}(c\|x)\|x)$. It is assumed that you would send even numbered messages and that you would receive odd numbered messages.
The synchronized counter protects message sequentiality. For instance it defeats replay attacks. Encryption and MAC protect message confidentiality, authentication, and integrity.

Q.5 Under which assumptions would your system be secure?

The described solution requires
- *that σ's are well authenticated (e.g., by voice recognition);*
- *that the public-key solution is secure (e.g., GPG is secure or your home-made RSA is secure);*
- *that your random numbers are really random, based on a large enough entropy;*
- *that the block cipher and the MAC are secure.*