Cryptography and Security — Final Exam
Solution

Serge Vaudenay

17.1.2017

- 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 <u>**not**</u> answer any technical question during the exam
- readability and style of writing will be part of the grade

The exam grade follows a linear scale in which each question has the same weight.

1 Stealing Bitcoins

We recall the ECDSA signature scheme.

- We are given a point G of an elliptic curve and its prime order n.
- A secret key is a value $d \in \mathbf{Z}_n$ and the public key is the point $Q = dG$.
- To sign a message M, we pick a random $k \in \mathbf{Z}_n^*$ and we compute $r = \overline{(kG)_1} \bmod n$, where $(kG)_1$ denotes the x-coordinate of kG and $\overline{(kG)_1}$ denotes its conversion into an integer; we compute $s = \frac{H(M)+dr}{k} \bmod n$, where $H(M)$ is the digest of M; the signature of M is (r, s).
- To verify a signature (r, s) of a message M for Q, we just compare r with a function V of (G, n, Q, r, s, M).

Q.1 Say what is the verification function V.

We compare r with

$$V(G, n, Q, r, s, M) = \overline{(\frac{H(M)}{s}G + \frac{r}{s}Q)_1}$$

because

$$\frac{H(M)}{s}G + \frac{r}{s}Q = \frac{H(M)}{s}G + \frac{dr}{s}G = \frac{H(M)+dr}{s}G = kG$$

Q.2 Assuming that a signing algorithm is implemented using a terrible random number generator (namely, with one for which the generated number often repeats), show that given two signed messages (M, r, s) and (M', r', s') for the public key Q, an adversary may extract the secret key d.

We know that $k = k'$ often occurs. If $k = k'$, we can see that $r = r'$ and we have $s - s' = \frac{H(M)-H(M')}{k} \bmod n$ so

$$d = \frac{1}{r}\left(\frac{s}{s-s'}(H(M) - H(M')) - H(M)\right) \bmod n$$

By computing d this way as soon as $r = r'$, we may obtain the correct secret value d. We normally have no cases $r = r'$ and $k \neq k'$ but we can still rule them out by checking that the above gives d satisfying $Q = dG$. Here, even though ECDSA is believed to be secure, a bad random number generator compromises the long term secret.

Q.3 We recall that a bitcoin transaction for an account Q is a signature of a will to collect the UTXO $\mathsf{utxo}_1, \ldots, \mathsf{utxo}_t$ owned by Q and split them to some accounts $Q_1, \ldots, Q_u$. Assuming that a user has implemented his ECDSA signing algorithm using a terrible random number generator, show how we can steal his bitcoins.

By scanning the blockchain we look at pairs of signatures for Q and apply the above formula until it gives the right secret d. Then, we forge a signature using d to transfer all UTXO owned by Q to another account.

2 Kleptography

Alice wants to communicate securely with Bob. For that, she buys a highly secure tamperproof device from the company mole.com and uses it to communicate. Upon reset, this device generates its own RSA secret key with random primes p and q, modulus n, and exponents e and d. In this implementation, the exponent e is random and of the same size as the modulus. It outputs n and e. Then, when we input a ciphertext, it decrypts it and returns the plain message.

Q.1 Suggest a modulus length for good security and describe the algorithm to generate e.

> *A secure modulus length could be 2 048 bits. To generate e, we pick a random e repeatedly until it is coprime with $\varphi(n) = (p-1)(q-1)$.*

Q.2 The company mole.com is hiding a trapdoor to be able to decrypt messages. For this, there is a symmetric secret key t which is put inside the device and the exponent e is chosen of form $\mathsf{SymEnc}_t(p)\|\mathsf{random}$.
Explain how mole.com can decrypt messages sent to Alice.

> *From the public key, mole.com gets e, extracts $\mathsf{SymEnc}_t(p)$ and decrypts it with the trapdoor t. Then, the company can divide n by p to obtain q and compute $d = e^{-1} \bmod \varphi(n)$. Given d and n, the company can decrypt messages using the RSA decryption algorithm.*

Q.3 Some agencies from the "axis of evil" (in the sense of G.W. Bush) succeed to reverse engineer devices from mole.com. Show that this creates a major national security issue for the country in which these devices are sold.

> *The evil agencies can extract the trapdoor t by reverse engineering. Then, it can decrypt all messages generated by any of these devices, just like mole.com in the previous question.*

Q.4 Using asymmetric encryption, propose a new generation of mole.com devices which allows the company to continue to decrypt messages without risking a security break in the case of reverse engineering.

> *One idea would be to use a public key to encrypt p, with the secret key only known by mole.com and take $e = \mathsf{Enc}_{\mathsf{mpk}}(p)\|\mathsf{random}$ where mpk is the public key of mole.com and the trapdoor t is the associated secret key.*

Q.5 In the above question, when the selected asymmetric encryption is RSA, observe that due to moduli sizes, this decreases the security of the encryption. Propose a way to fix this.

> *One requirement in the solution of the previous question is that the asymmetric encryption produces ciphertexts smaller than the RSA modulus so that we have enough space in e to store $\mathsf{Enc}_{\mathsf{mpk}}(p)$. With RSA, the problem is to that the modulus size for mpk must be smaller than the modulus size of the encryption in the device. So, the security decreases. To fix this, we could use a public key cryptosystem with short messages. For instance, we could use ElGamal over elliptic curves and use point compression for the ciphertext.*

3 AES-GCM Issues

We recall the GCM mode of AES. We modified it a bit for simplicity in this exercise: we assume no associated data, all messages have a length multiple of 128 bits, and the authentication tag has 128 bits. To encrypt a message P with an AES key K and a 96-bit nonce IV, we split it into m 128-bit blocks $P = (P_1, \ldots, P_m)$ and run the following algorithm (written in pseudocode).

1: $J_i = \mathsf{IV} \| (i + 1 \bmod 2^{32})_{32}$, $i = 0, \ldots, m$, where x_{32} is the binary representation of x in 32 bits
2: $C = (C_1, \ldots, C_m)$ where $C_i = P_i \oplus \mathsf{AES}_K(J_i)$
3: $H = \mathsf{AES}_K(0^{128})$ (called the *authentication key*)
4: $S = C_1 H^m \oplus \cdots \oplus C_m H$ (with multiplications in $\mathsf{GF}(2^{128})$)
5: $T = S \oplus \mathsf{AES}_K(J_0)$
6: the output is (C, T)

Q.1 Give the description of decryption/authentication in pseudocode.

To decrypt $(C_1, \ldots, C_m, T)$ with a key K and nonce IV, we proceed as follows.

1: $J_i = \mathsf{IV} \| (i + 1 \bmod 2^{32})_{32}$, $i = 0, \ldots, m$
2: $P = (P_1, \ldots, P_m)$ *where* $P_i = C_i \oplus \mathsf{AES}_K(J_i)$
3: $H = \mathsf{AES}_K(0^{128})$
4: $S = C_1 H^m \oplus \cdots \oplus C_m H$ *in* $\mathsf{GF}(2^{128})$
5: if $T \neq S \oplus \mathsf{AES}_K(J_0)$, *abort (output nothing)*
6: the output is P

Q.2 Assuming that a user encrypts a message with m larger than 2^{32}, show how an adversary can recover the XOR of some plaintext blocks from the ciphertext (C, T).

We have $J_{i+2^{32}} = J_i$ for all i. So, $C_{i+2^{32}} \oplus C_i = P_{i+2^{32}} \oplus P_i$.

Q.3 Assuming that a user encrypts two messages P and P' with the same nonce IV, show how an adversary can recover a set of small cardinality which contains the authentication key H. (For simplicity, we assume that P and P' have the same length.)

We use obvious notations by adding a $'$ for values depending on P'. If $\mathsf{IV}' = \mathsf{IV}$, then $J'_0 = J_0$. So, $T' \oplus T = S' \oplus S$. With a tag of 128 bits, we obtain

$$T' \oplus T = (C_1 \oplus C'_1)H^m \oplus \cdots (C_m \oplus C'_m)H$$

By solving a polynomial equation, we recover H. We may recover up to m solutions.

Q.4 If the adversary knows a set of small cardinality to which H belongs, show how he can decrypt any ciphertext (C, T) with a nonce IV by a chosen ciphertext attack.

We write $C = (C_1, \ldots, C_m)$. For each guess of H, the adversary sets $C' = (C_1, \ldots, C_m \oplus 1_{128})$ and $T' = T \oplus H$. If the decryption of (C', T') with nonce IV does not abort, the adversary deduces that his guess for H is correct. Using the small set of values, he can thus recover H. If the decryption is $P' = (P'_1, \ldots, P'_m)$, then the adversary deduces $\mathsf{AES}_K(J_i) = P'_i \oplus C'_i$, then $P_i = C_i \oplus \mathsf{AES}_K(J_i)$.

4 Prime Reuse in RSA

The following exercise is inspired from Mining your Ps and Qs: Detection of Widespread Weak Keys in Network Devices *by Heninger, Durumeric, Wustrow, Halderman, published in the proceedings of USENIX Security Symposium'12, 2012.*

In this exercise, we consider a pool of D RSA keys with a modulus length of s bits.

We recall that the probability of a uniformly distributed random number in $\{1, \ldots, n\}$ to be prime is approximately $\frac{1}{\ln n}$.

Q.1 Say how to check if two different RSA moduli use a prime factor p in common and why it is a security problem.

> *Given two RSA moduli n and n', we can easily compute $\mathsf{gcd}(n, n')$ to see common factors. If there is one in common, it will be the gcd.*
> *It is a major security problem because if $p = \mathsf{gcd}(n, n')$ can be computed, then we can compute $q = n/p$ and $q' = n'/p$ and deduce the RSA secret keys.*

Q.2 Using truly random prime numbers, estimate the probability that there exist two RSA keys on the Internet which have a prime factor in common (or estimate the number of pairs with a common prime factor). Justify your answer precisely.

> *We have $2D$ prime number generations in total. The number of $\frac{s}{2}$-bit prime numbers is roughly $m = 2^{\frac{s}{2}} / \ln 2^{\frac{s}{2}}$. We consider two ways to answer to this question.*
>
> – *The expected number of collisions based on the birthday paradox is about*
>
> $$\frac{(2D)^2}{2} \times \frac{1}{m} = \frac{2D^2}{m}$$
>
> – *The probability of having a collision in $2D$ trials in a set of m is*
>
> $$1 - \left(1 - \frac{1}{m}\right)^{2D^2} \approx 1 - e^{-\frac{2D^2}{m}} \approx \frac{2D^2}{m}$$
>
> *In both cases, we have*
>
> $$\frac{2D^2}{m} \approx sD^2 2^{-\frac{s}{2}} \ln 2 \approx 2^{-456}$$
>
> *(For the numeric computation, we use the figures from the next question.) So, we are almost sure never to see common prime factors.*

Q.3 By scanning public keys over the Internet, one can find about $D = 11\,170\,883$ keys of size $s = 1\,024$. We observed that $16\,717$ RSA keys share a common prime factor. What can we deduce?

> *We deduce that the random generator is bad.*

5 Subgroup Issues in the Diffie-Hellman Protocol

Let g be an element of a (multiplicative) Abelian group G and let $\langle g \rangle$ denote the subgroup of G generated by g. Let q denote the order of g. Let n denote the order of G. We consider the Diffie-Hellman protocol in which Alice picks a secret $x \in \mathbf{Z}_q^*$, computes $X = g^x$, sends X to Bob. Bob picks a secret $y \in \mathbf{Z}_q^*$, computes $Y = g^y$, sends Y to Alice. Alice checks that $Y \in \langle g \rangle$ and computes $K = Y^x$. Bob checks that $X \in \langle g \rangle$ and computes $K = X^y$.

Let B be a given bound. Given $q = q_s q_l$ with $q_s = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$ where the p_i are pairwise different "small" primes (i.e. $p_i \leq B$) and q_l has no "small" factor except 1, we denote $\mathcal{F}_B(q) = \{(p_1, \alpha_1), \ldots, (p_r, \alpha_r)\}$.

We recall that we have "efficient" (i.e. polynomial in $B + \log q$) partial factoring algorithms to compute $\mathcal{F}_B$.

We also recall that if p is a "small" prime (i.e. $p \leq B$) and $g' \in G$ is an element of order p^α, then there is an "efficient" (i.e. polynomial in αB) algorithm to compute the discrete logarithm in $\langle g' \rangle$. More precisely, there is an algorithm $\mathcal{L}_0$ such that $\mathcal{L}_0(B, p, \alpha, g', g'^x) = x \bmod p^\alpha$.

Q.1 If g has order q and $(p_1, \alpha_1) \in \mathcal{F}_B(q)$, show that there is an "efficient" algorithm to compute $x \bmod p_1^{\alpha_1}$ from g^x. More precisely, show that there is an algorithm $\mathcal{L}_1$ such that $\mathcal{L}_1(B, q, p_1, \alpha_1, g, g^x) = x \bmod p_1^{\alpha_1}$ for any x. Justify your answer.

> *Let* $\mathcal{L}_1(B, q, p_1, \alpha_1, g, g^x) = \mathcal{L}_0(B, p_1, \alpha_1, g^{q/p_1^{\alpha_1}}, (g^x)^{q/p_1^{\alpha_1}})$. *Clearly,* $g' = g^{q/p_1^{\alpha_1}}$ *has order* $p_1^{\alpha_1}$ *and* $(g^x)^{q/p_1^{\alpha_1}} = g'^{x \bmod p_1^{\alpha_1}}$ *has a discrete logarithm equal to* $x \bmod p_1^{\alpha_1}$.

Q.2 If g has order $q = q_s q_l$ with $q_s = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$ where $\mathcal{F}_B(q) = \{(p_1, \alpha_1), \ldots, (p_r, \alpha_r)\}$, show that there is an "efficient" algorithm to compute the modulo q_s part of the discrete logarithm in $\langle g \rangle$. More precisely, show that there is an algorithm $\mathcal{L}$ such that $\mathcal{L}(B, q, g, g^x) = x \bmod q_s$ for any x. Justify your answer.

> *Once we compute* $x \bmod p_i^{\alpha_i} = \mathcal{L}_1(B, q, p_i, \alpha_i, g, g^x)$ *we can recombine them into* $x \bmod q_s$ *by using the Chinese Remainder Theorem.*

Q.3 With the previous notation, show that if instead of picking x in $\mathbf{Z}_q^*$ Alice picks x uniform in $\{1, \ldots, q_s - 1\} \cap \mathbf{Z}_q^*$, then a passive adversary can recover x.

> *We have* $\mathcal{L}(B, q, g, X) = x \bmod q_s = x$ *when* $x < q_s$ *so we can compute* x *easily in that case.*

Q.4 We now assume that y is a static key (i.e. Bob runs many sessions of the protocol by using the same value y). We consider that after the Diffie-Hellman protocol, Bob sends the encryption of a known message m (e.g. the null message $m = 0$) with the key $\mathsf{KDF}(X^y)$. Encryption is done using a deterministic symmetric encryption. We write $n = pqr$ where p is a "small" prime (i.e. $p \leq B$).
Show that if Bob does not verify that $X \in \langle g \rangle$, an active adversary can recover $y \bmod p$ by malciously selecting X.

> *By choosing* X *set to an element of order* p *in* G*, there will be only* p *possible values for* X^y *so the adversary can make an exhaustive search using* $\mathsf{Enc}_{\mathsf{KDF}(X^y)}(0)$ *and obtain* $y \bmod p$.