Cryptography and Security — Final Exam
Solution

Serge Vaudenay

20.1.2015

- 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

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

1 Hidden Collisions in DSA

The following exercise is inspired from Hidden Collisions on DSS *by Vaudenay, published in the proceedings of CRYPTO'96 pp. 83–88, LNCS vol. 1109, Springer 1996.*

We recall the DSA signature scheme:

Public parameters (p, q, g): pick a 160-bit prime number q, pick a large a random until $p = aq + 1$ is prime, pick h in $\mathbf{Z}_p^*$ and take $g = h^a \bmod p$ until $g \neq 1$.
Set up: pick $x \in \mathbf{Z}_q$ (the secret key) and compute $y = g^x \bmod p$ (the public key).
Signature generation for a message M**:** pick a random $k \in \mathbf{Z}_q^*$, compute

$$r = (g^k \bmod p) \bmod q \quad s = \frac{H(M) + xr}{k} \bmod q$$

the signature is $\sigma = (r, s)$.
Verification: check that $r = \left(g^{\frac{H(M)}{s} \bmod q} y^{\frac{r}{s} \bmod q} \bmod p\right) \bmod q$.

The hash function H is the SHA-1 standard. The output of H is a binary string which is implicitely converted into an integer. DSA was standardized by NIST with a usual suspicion that the NSA was behind it. It could be the case that some specific choices for (p, q, g) could indeed hide some special property making an attack possible. This is what we investigate in this exercise.

Q.1 What is the complexity of finding m and m' such that $m \neq m'$ and $H(m) = H(m')$?

Using the birthday paradox, the complexity has an order of magnitude of $\sqrt{\#\textsf{digest domain}}$. *Since SHA-1 hashes onto 160 bits, it is* 2^{80}.

Q.2 Describe a chosen-message signature-forgery attack based on the fact that an adversary knows two messages m and m' such that $m \neq m'$ and $H(m) = H(m')$.

The adversary asks the signer to sign m and gets the signature (r, s). Then, the forgery is $(m', (r, s))$. We can check that the signature is valid. Indeed,

$$\left(g^{\frac{H(m')}{s} \bmod q} y^{\frac{r}{s} \bmod q} \bmod p\right) \bmod q = \left(g^{\frac{H(m)}{s} \bmod q} y^{\frac{r}{s} \bmod q} \bmod p\right) \bmod q = r$$

Furthermore, $m \neq m'$. So, this is a valid forgery.

Q.3 Describe a chosen-message signature-forgery attack based on the fact that an adversary knows two messages m and m' such that $m \neq m'$ and $q = H(m) - H(m')$ (with the integer subtraction).

Propose a way for the NSA to generate public parameters (p, q, g) in such a way that it can later perform a forgery attack for a suitable message.

If q divides $H(m) - H(m')$, then $H(m) \bmod q = H(m') \bmod q$. We can see that the attack from the above question still works, since the digest of messages is always taken modulo q.
So, the NSA can select two random messages m and m' until $q = H(m) - H(m')$ is a 160-bit prime number. The message m could be a test message (such as "this is a test message to check that the signature works") while m' could be a payment order.

Q.4 To put more confidence, NIST added a way to certify that (p, q, g) were honestly selected. For this, we shall provide together with the public parameters a value seed such that

$$q = (H(\mathsf{seed}) \oplus H(\mathsf{seed} + 1)) \vee 2^{159} \vee 1$$

where $\oplus$ denotes the bitwise XOR, $\vee$ denotes the bitwise OR, and $+$ is the regular addition of integers. I.e., q is the XOR between $H(\mathsf{seed})$ and $H(\mathsf{seed} + 1)$ after which the least and the most significant bits are forced to 1 so that $2^{159} \leq q < 2^{160}$ and q is odd.

Propose a way to construct (seed, p, q, g) such that an attack is still possible.

HINT: take $m = \mathsf{seed}$ and $m' = \mathsf{seed} + 1$ and estimate the probability that $|H(m) - H(m')| = q$ for seed random, by looking at the propagation of carry bits in the subtraction.

HINT^2: you may skip this question.

If we take a random seed, *we let* $m = \mathsf{seed}$, $m' = \mathsf{seed} + 1$, m_1 *and* m_2 *such that* $\{m_1, m_2\} = \{m, m'\}$ *and* $H(m_1) > H(m_2)$. *We have* $|H(m) - H(m')| = H(m_1) - H(m_2)$. *We define the event*

$$E : |H(m) - H(m')| = (H(\mathsf{seed}) \oplus H(\mathsf{seed} + 1)) \vee 2^{159} \vee 1$$

We want to estimate $\Pr[E]$. *So, we repeat the above process* $1/\Pr[E]$ *times until* E *occurs and we can apply the same attack with* m *and* m'.
We let $\mathsf{msb}_2(s)$ *be the two most significant bits of a string* s. *We first assume that* $\mathsf{msb}(H(m_1)) \neq \mathsf{msb}(H(m_2))$.
When making the boolean subtraction $H(m_1) - H(m_2)$, *the difference of the least significant bits gives 1 with no carry only when making* $1 - 0$. *This occurs with probability* $\frac{1}{4}$. *Then, for each of the 157 next bit positions, assuming there is no carry, the subtraction of the two bits matched the XOR and makes no carry with probability* $\frac{3}{4}$ *(that is, in all cases except* $0 - 1$*). Finally, the two most significant bits with no carry have a difference matching the XOR and starting with 1 in the following cases:* $10 - 00$, $11 - 00$, $11 - 01$. *The remaining cases are* $10 - 01$, $11 - 10$, $01 - 00$. *So, this happens with probability* $\frac{1}{2}$. *Hence, we have*

$$\Pr\left[E \middle| \mathsf{msb}(H(m_1)) \neq \mathsf{msb}(H(m_2))\right] = \frac{1}{2} \times \frac{1}{4} \times \left(\frac{3}{4}\right)^{157}$$

When $\mathsf{msb}(H(m_1)) = \mathsf{msb}(H(m_2))$, *we can see that* E *cannot occur. So,*

$$\Pr[E] = \frac{1}{2^5} \times \left(\frac{3}{4}\right)^{157} \approx 2^{-70}$$

The attack has a complexity of 2^{70}.

2 DSA With Related Randomness

We recall the DSA signature scheme:

Public parameters (p, q, g): pick a 160-bit prime number q, pick a large a random until $p = aq + 1$ is prime, pick h in $\mathbf{Z}_p^*$ and take $g = h^a \bmod p$ until $g \neq 1$.

Set up: pick $x \in \mathbf{Z}_q$ (the secret key) and compute $y = g^x \bmod p$ (the public key).

Signature generation for a message M: pick a random $k \in \mathbf{Z}_q^*$, compute

$$r = (g^k \bmod p) \bmod q \quad s = \frac{H(M) + xr}{k} \bmod q$$

the signature is $\sigma = (r, s)$.

Verification: check that $r = \left(g^{\frac{H(M)}{s} \bmod q} y^{\frac{r}{s} \bmod q} \bmod p \right) \bmod q$.

Sampling the randomness k to sign is critical. This exercise is about bad sampling methods.

In what follows, we consider two messages m_1 and m_2, a signature (r_i, s_i) for message m_i using the randomness k_i, $i = 1, 2$.

Q.1 Sometimes, random sources are not reliable and produce twice the same value. If $k_1 = k_2$, show that from the values of $p, q, g, y, r_1, s_1, r_2, s_2, m_1, m_2$ we can recover x.

> *Modulo q, we have $k_i \equiv \frac{H(m_i)+xr_i}{s_i}$, $i = 1, 2$. If $k_1 = k_2$, we deduce $\frac{H(m_1)+xr_1}{s_1} \equiv \frac{H(m_2)+xr_2}{s_2}$. So, $x = \frac{s_2 H(m_1) - s_1 H(m_2)}{s_1 r_2 - s_2 r_1} \bmod q$.*

Q.2 To avoid the previous problem, a crypto apprentice decides to sample k based on a counter. Redo the previous question with $k_2 = k_1 + 1$.

> *The equation is now $\frac{H(m_1)+xr_1}{s_1} + 1 \equiv \frac{H(m_2)+xr_2}{s_2}$ which leads us to*
>
> $$x = \frac{s_2 H(m_1) + s_1 s_2 - s_1 H(m_2)}{s_1 r_2 - s_2 r_1} \bmod q$$

Q.3 To avoid the previous problem, a crypto apprentice decides to sample k by iterating an affine function. Redo the previous question for $k_2 = \alpha k_1 + \beta$ with α and β known.

> *The equation is now $\alpha \frac{H(m_1)+xr_1}{s_1} + \beta \equiv \frac{H(m_2)+xr_2}{s_2}$ which leads us to*
>
> $$x = \frac{s_2 \alpha H(m_1) + s_1 s_2 \beta - s_1 H(m_2)}{s_1 r_2 - s_2 \alpha r_1} \bmod q$$

Q.4 To avoid the previous problem, a crypto apprentice decides to sample k by iterating an quadratic function. Redo the previous question for $k_2 = \alpha k_1^2 + \beta k_1 + \gamma$ with α, β, and γ known.

The equation is now

$$\alpha\left(\frac{H(m_1)+xr_1}{s_1}\right)^2+\beta\frac{H(m_1)+xr_1}{s_1}+\gamma\equiv\frac{H(m_2)+xr_2}{s_2}$$

We can rewrite it as

$$x^2\frac{\alpha r_1^2}{s_1^2}+x\left(2\frac{\alpha H(m_1)r_1}{s_1^2}+\frac{\beta r_1}{s_1}-\frac{r_2}{s_2}\right)+\left(\frac{\alpha H(m_1)^2}{s_1^2}+\frac{\beta H(m_1)}{s_1}+\gamma-\frac{H(m_2)}{s_2}\right)\equiv 0$$

By writing this as $ux^2+vx+w\equiv 0$*, we find that* x *is one of the two solutions*

$$x=\frac{-v\pm\sqrt{v^2-4uw}}{2u}\bmod q$$

We check the correct solution with $y=g^x \bmod p$.

3 Reset Password Recovery

We consider a non-uniform distribution D of passwords. Passwords are taken from a set $\{k_1, \ldots, k_n\}$ and each password k_i is selected with probability $\Pr_D[k_i]$. (We omit the subscript D when there is no ambiguity in the distribution.) For simplicity, we assume that $\Pr[k_1] \geq \Pr[k_2] \geq \cdots \geq \Pr[k_n]$. We consider a game in which a cryptographer apprentice plays with a black-box device which has two buttons — a *reset* button and a *test* button — and a keyboard.

- When the player pushes the reset button, the device picks a new password K, following the above distribution, and stores it into its memory. The game cannot start before the player pushes this button.
- The player can enter an input w on the keyboard and push the test button. This makes the device compare K with w. If $K = w$, the device opens, the player wins, and the game stops. Otherwise, the device remains closed and the player continues.

A strategy is an algorithm that the player follows to play the game. Given a strategy, we let C denote the expected number of times the player pushes the test button until he wins. The goal of the player is to design a strategy which uses a minimal C.

In this exercise, we consider several strategies. To compare them, we use a toy distribution T defined by the parameters a, p and

$$\Pr_T[k_1] = \cdots = \Pr_T[k_a] = \frac{p}{a} \quad , \quad \Pr_T[k_{a+1}] = \cdots = \Pr_T[k_n] = \frac{1-p}{n-a}$$

and assuming that $\frac{p}{a} \geq \frac{1-p}{n-a}$.

Q.1 We consider a strategy in which the player always pushes the reset button before pushing the test button. For a general distribution D, give an optimal strategy and the corresponding value of C.
Apply the general result to the toy distribution T.

The best strategy is to test the most likely possible password, i.e. k_1, following the algorithm

1: ***loop***
2:     *reset*
3:     *test k_1*
4: ***end loop***

The expected number of test queries is

$$\sum_{i=1}^{+\infty} i \Pr[k_1](1 - \Pr[k_1])^{i-1} = \frac{1}{\Pr[k_1]} = 2^{-H_\infty}$$

where H_∞ is called the min-entropy.
For the toy distribution T, we have

$$C = \frac{a}{p}$$

Q.2 We consider a strategy in which the reset button is never used again after the initial reset. For a general distribution D, give an optimal strategy and the corresponding value of C. Apply the general result to the toy distribution T.

The best strategy is to test the possible passwords by decreasing order of likelihood, i.e.

1: *reset*
2: ***for*** $i = 1$ *to* n ***do***
3:     *test* k_i
4: ***end for***

The expected number of test queries is

$$\sum_{i=1}^{n} i \Pr[k_i] = G$$

which is called the guesswork entropy*.*
For the toy distribution T, we have

$$\begin{aligned} G &= \frac{p}{a}\sum_{i=1}^{a} i + \frac{1-p}{n-a}\sum_{i=a+1}^{n} i \\ &= p\frac{a+1}{2} + (1-p)\frac{n+a+1}{2} \\ &= (1-p)\frac{n}{2} + \frac{a+1}{2} \end{aligned}$$

Q.3 For $n = 3$ and $a = 1$, propose one value for p in the toy distribution T so that the strategy in Q.2 is better and one value for p so that the strategy in Q.1 is better.
We recall that we must have $\frac{p}{a} \geq \frac{1-p}{n-a}$.

For $n = 3$ and $a = 1$, the condition $\frac{p}{a} \geq \frac{1-p}{n-a}$ simplifies to $p \geq \frac{1}{3}$.
We could already look at the extreme cases with $p = \frac{1}{3}$, making T the uniform distribution with $n = 3$, and $p = 1$, making T having zero probability on k_2 and k_3. For the uniform distribution, we have $G = 2$ and $2^{-H_\infty} = 3$. So, the strategy of Q.2 is better. For $p = 1$, we have $G = 1$ and $2^{-H_\infty} = 1$. So, both strategies are equally good.
In the general case, we have

$$G = \frac{5-3p}{2}$$

and

$$2^{-H_\infty} = \frac{1}{p}$$

We have equality between G and 2^{-H_∞} if and only if $3p^2 - 5p + 2 = 0$ which have roots $p = 1$ and $p = \frac{2}{3}$. So, for $\frac{1}{3} \leq p \leq \frac{2}{3}$, G is lower, and for $\frac{2}{3} \leq p \leq 1$, 2^{-H_∞} is lower. We can propose $p = \frac{5}{6}$ for which the strategy of Q.1 is better.

Q.4 We consider a strategy in which the player always pushes the reset button after m tests have been made since the last reset. For a general distribution D, give an optimal strategy and the corresponding value of C.
Check that your result is consistent with those from Q.1 and Q.2 with $m = 1$ and $m = n$.

The best strategy is to test the m most likely possible passwords by decreasing order of likelihood, i.e.

```
loop
    reset
    for i = 1 to m do
        test k_i
    end for
end loop
```

Let

$$p_m = \Pr[k_1] + \cdots + \Pr[k_m]$$

We define the distribution $D' = D|K \in \{k_1, \ldots, k_m\}$ conditioned to $K \in \{k_1, \ldots, k_m\}$. We have $\Pr_{D'}[k_i] = \frac{1}{p_m} \Pr_D[k_i]$. The distribution D' has a guesswork entropy G_m defined by

$$G_m = \frac{1}{p_m} \sum_{i=1}^{m} i \Pr_D[k_i]$$

The expected number of iterations of the outer loop is $\frac{1}{p_m}$. The number of tests during the last iteration is G_m. The number of tests during each of the previous iteration is exactly m. So, the expected number of tests is

$$C_1 = m\left(\frac{1}{p_m} - 1\right) + G_m$$

Note that for $m = 1$, we have $p_1 = \Pr[k_1]$, $G_1 = 1$, and $C_1 = \frac{1}{\Pr[k_1]}$. For $m = n$, we have $p_n = 1$, $G_n = G$, and $C_n = G$.

4 A Bad EKE with RSA

In this exercise we want to apply the EKE construction with the RSA cryptosystem and the AES cipher to derive a password-based authenticated key exchange protocol (PAKE). For that, Alice and Bob are assumed to share a (low-entropy) password w. The protocol runs as follows:

Alice		**Bob**
password: w		password: w
$\mathsf{RSAGen} \to (\mathsf{sk}, N)$, $B_1\|\cdots\|B_8 \leftarrow N$		
$C_1\|\cdots\|C_8 \leftarrow \mathsf{AESCBC}_w(B_1\|\cdots\|B_8)$	$\xrightarrow{C_1\|\cdots\|C_8}$	$B_1\|\cdots\|B_8 \leftarrow \mathsf{AESCBC}_w^{-1}(C_1\|\cdots\|C_8)$
		$N \leftarrow B_1\|\cdots\|B_8$, pick $K \in \{0,1\}^{128}$
		$\Gamma \leftarrow \mathsf{RSAEnc}_{N,3}(K)$, $B'_1\|\cdots\|B'_8 \leftarrow \Gamma$
$B'_1\|\cdots\|B'_8 \leftarrow \mathsf{AESCBC}_w^{-1}(C'_1\|\cdots\|C'_8)$	$\xleftarrow{C'_1\|\cdots\|C'_8}$	$C'_1\|\cdots\|C'_8 \leftarrow \mathsf{AESCBC}_w(B'_1\|\cdots\|B'_8)$
$\Gamma \leftarrow B'_1\|\cdots\|B'_8$, $K \leftarrow \mathsf{RSADec}_{\mathsf{sk}}(\Gamma)$		
output: K		output: K

Here are some explanations:

- Alice generates an RSA modulus N such that $\mathsf{gcd}(3, \varphi(N)) = 1$. This modulus is supposed to have exactly 1024 bits. The modulus N is written in binary and splits into 8 blocks $N = B_1\|\cdots\|B_8$. The blocks $B_1, \ldots, B_8$ are then encrypted with AES in CBC mode with IV set to the zero block and the key set to w. The obtained ciphertext blocks $C_1, \ldots, C_8$ are sent to Bob.
- Bob decrypts $C_1, \ldots, C_8$ following the AES-CBC decryption algorithm with IV set to the zero block and the key set to w. He recovers $B_1, \ldots, B_8$ and can reconstruct N. He picks a random 128-bit key K and computes the RSA-OAEP encryption of K with key N and $e = 3$. He then obtains a ciphertext Γ. This is split into 8 blocks $\Gamma = B'_1\|\cdots\|B'_8$ and the blocks $B'_1, \ldots, B'_8$ are then encrypted with AES in CBC mode with IV set to the zero block and the key set to w. The obtained ciphertext blocks $C'_1, \ldots, C'_8$ are sent to Alice.
- Alice decrypts $C'_1, \ldots, C'_8$ following the AES-CBC decryption algorithm with IV set to the zero block and the key set to w. She recovers $B'_1, \ldots, B'_8$ and can reconstruct Γ. She applies the RSA-OAEP decryption on Γ with her secret key and obtains K.

So, Alice and Bob end the protocol with the secret K.

Q.1 Assume (*only in this question*) that we use plain RSA instead of RSA-OAEP. Show that Eve can easily recover w and K in a *passive* attack with a single execution of the protocol.
HINT: show that the plain RSA decryption of Γ is easy in this case.

> *Since $K < 2^{128}$, we have $K^3 < 2^{384}$ which is small. So, $K^3 \bmod N = K^3$. Eve could do an exhaustive search on W to decrypt $C'_1\|\cdots\|C'_8$ to obtain some candidate values for Γ. She could then compute $\sqrt[3]{\Gamma}$. For wrong guesses for the password, this is unlikely to be an integer. For the correct w, this gives K. So, Eve recovers w and K.*

Q.2 Propose a *passive* attack allowing Eve to deduce the password w after a few executions of the protocol. Estimate the number of executions needed to recover a password with less than 48 bits of entropy with a high probability.
HINT: N is not an arbitrary bitstring. You could think of eliminating some password guesses.

We apply the principle of the partition attack*: the set of valid N does not corresponds to the set of messages. Namely, the most significant and the least significant bits of N must always be 1 (since N is odd and between 2^{1023} and 2^{1024}). We could also note that $N \bmod 3 = 1$, so the set of valid N is at least $\frac{1}{8}$ of the full space. Eve could do an exhaustive search on all w, decrypt $C_1\|\cdots\|C_8$ with the trial passwords and discards all trials not satisfying the above conditions. The set of possible passwords would thus be reduced by at least a factor 8. By using k executions of the protocol, the set of possible passwords is reduced by 8^k until it contains the correct password w. For instance, if w has an entropy lower than 48 bits, $k = 16$ iterations are enough to isolate the correct password.*