Cryptography and Security — Final Exam
Solution

Serge Vaudenay

12.1.2011

- duration: 3h
- no documents is allowed
- a pocket calculator is allowed
- communication devices are not allowed
- 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 3-Collisions

This exercise is inspired by "Improved Generic Algorithms for 3-Collisions" by Joux and Lucks, ASIACRYPT'09, LNCS vol. 5912, pp. 347–363, 2009.

Let f be a random-looking function from a set $\mathcal{X}$ to a set $\mathcal{Y}$. Let N denote the cardinality of $\mathcal{Y}$. We call an r-collision a set $\{x_1,\ldots,x_r\}$ of r elements of $\mathcal{X}$ such that $f(x_i) = f(x_j)$ for every i and j.

Q.1 Recall what preimage resistance and collision resistance mean.

Preimage resistance: given an arbitrary y, it is hard to find x such that $f(x) = y$.
Collision resistance: it is hard to find x and x′ such that $x \neq x'$ and $f(x) = f(x')$.

Q.2 Name the ideas behind two collision finding algorithms from the course and give their time and memory complexity.

We have several algorithms based on the ***birthday paradox****. For instance we can pick a new x and store $(f(x),x)$ in memory if there is no (y,x) entry with $y = f(x)$, or stop if there is such entry. This works with time and memory complexity $O(\sqrt{N})$.*
We have other algorithms based on ***cycle finding*** *with much lower memory complexity. For instance, the Floyd cycling algorithm works with memory complexity $O(1)$ and time complexity $O(\sqrt{N})$.*

Q.3 Let $y \in \mathcal{Y}$ be a target value. Provide an algorithm $\mathcal{A}_1$ such that upon input y it returns $x \in \mathcal{X}$ such that $f(x) = y$ with average complexity N in terms of f evaluations.
Make the complexity analysis.

Algorithm $\mathcal{A}_1$ picks a new $x \in \mathcal{X}$ at random until $f(x) = y$:

```
input: $y$
1: repeat
2:     pick $x$ at random
3: until $f(x) = y$
output: $x$
```

Let p_y be the probability that a random x is mapped onto y by f. So, the probability to stop after i iterations is $(1-p_y)^{i-1}p_y$. The average complexity for input y is thus

$$E(C_y) = \sum_{i=0}^{+\infty} i(1-p_y)^{i-1}p_y = \frac{1}{p_y}$$

Assuming that $p_y \approx N^{-1}$, we have $E(C) \approx N$.

Q.4 By using $\mathcal{A}_1$ as a subroutine, provide an algorithm $\mathcal{A}_2$ producing r-collisions with complexity rN in terms of f evaluations.
Make the complexity analysis.

Algorithm $\mathcal{A}_2$ picks a random $y \in \mathcal{Y}$ and repeatedly calls $\mathcal{A}_1$ until r pairwise different preimages of y are found. If we neglect the case where a preimage x is found several times, the complexity is rN.
To make it more rigorous, we shall make sure that $\mathcal{A}_1$ never picks an x twice through different calls. That is, we rewrite $\mathcal{A}_1$ for $\mathcal{A}_2$ so that a new x is picked at each iteration, it is stored if $f(x) = y$, and the algorithm stops if the rth preimage was found.
To improve it further, we first pick x and take $y = f(x)$ so that we have one less preimage to find:

```
1: pick $x$ at random
2: let $y = f(x)$ and $L = (x)$
3: repeat
4:     pick a new $x$ at random
5:     if $f(x) = y$ and $x$ does not appear in $L$ then
6:         insert $x$ in $L$
7:     end if
8: until $L$ has size $r$
output: $L$
```

Q.5 We consider an algorithm $\mathcal{A}_3$ for making r-collisions, defined by two parameters α and β. The algorithm works in two phases. In the first phase, it picks N^α random $x \in \mathcal{X}$ and stores $(f(x), L_{f(x)})$ in a hash table, where $L_{f(x)}$ is a list initialized to the single element x. In the second phase, it iteratively picks N^β random $x \in \mathcal{X}$. For each of these x's, it looks whether $y = f(x)$ has an entry in the hash table. If it does, and if x is not already in the list L_y, x is inserted into the list L_y. If L_y has r elements, the algorithm output L_y. We assume that $\mathcal{A}_3$ never picks the same x twice.

```
1: for $i = 1$ to $N^\alpha$ do
2:     pick a new $x$ at random
```

```
3:     set $y = f(x)$ and store $(y,(x))$ at place $h(y)$
4: end for
5: for $i = 1$ to $N^\beta$ do
6:     pick a new $x$ at random
7:     if there is an entry $(y, L_y)$ at place $h(f(x))$ such that $y = f(x)$ then
8:         insert $x$ in list $L_y$
9:         if $L_y$ has size $r$ then
10:            yield $L_y$ and stop
11:        end if
12:    end if
13: end for
14: algorithm failed
```

Q.5a Show that $\mathcal{A}_3$ either generates r-collisions or fails.

We show by induction that for every existing y in the hash table, the list L_y contain pairwise different entries x such that $f(x) = y$. It is true when the algorithm starts (the hash table is empty so there is no y). It is true when a new y entry is inserted (the list has a single element). It is true when a list is expanded. So, it is true throughout the execution of the algorithm.
The algorithm can only stop when it has found a list of cardinality r. So, this list is an r-collision.

Q.5b Show that the memory complexity is $M = O(N^\alpha r \log N)$ and that the time complexity in terms of f evaluations is $T = N^\alpha + N^\beta$.

Due the the structure of the algorithm, we have N^α entries in the hash table. Each entry has size up to $r\log_2 N$ in bits, so the memory complexity is $O(N^\alpha r \log_2 N)$. By neglecting $r\log_2 N$ we obtain $M \approx N^\alpha$.
We need N^α evaluations to prepare the hash table and N^β evaluations to run Phase 2, so the time complexity is $N^\alpha + N^\beta$. By using the approximation $\log(a+b) \approx \max(\log a, \log b)$ we obtain $T \approx \max(N^\alpha, N^\beta)$.

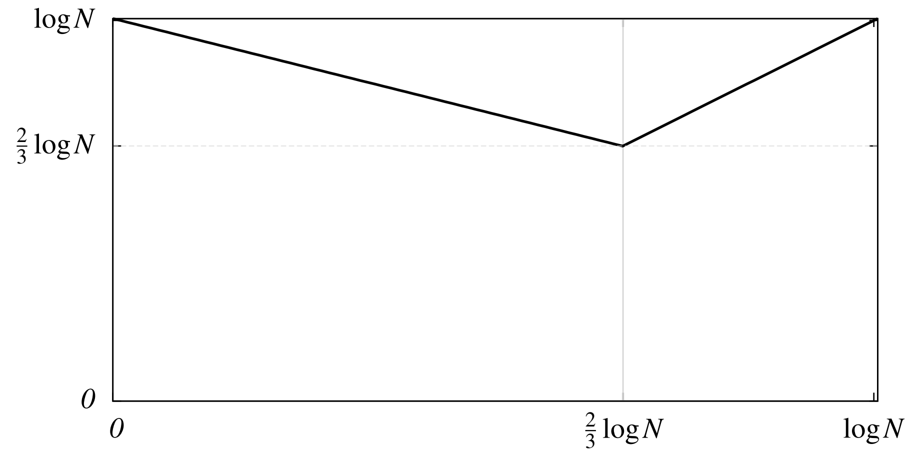
In what follows we will approximate $T \approx \max(N^\alpha, N^\beta)$ and $M \approx N^\alpha$.

Q.5c For $r = 2$, which inequality shall α and β satisfy to reach a constant probability of success?
For $r = 3$, show that this inequality becomes $\alpha + 2\beta \geq 2$.
Hint: apply the birthday paradox in Phase 2.

Each x in the second phase will hit one entry with probability $N^{\alpha-1}$. So, we have $N^{\alpha+\beta-1}$ hits on average for $r = 2$. We thus need $\alpha + \beta \geq 1$.
For $r = 3$, we have $N^{\alpha+\beta-1}$ hits in a set of N^α. When the number of hits exceed $N^{\frac{1}{2}\alpha}$, we obtain colliding hits with constant probability, thanks to the birthday paradox. So, $\alpha + \beta - 1 \geq \frac{1}{2}\alpha$ guarantees a constant probability of success. This is equivalent to $\alpha + 2\beta \geq 2$.

Q.5d Show that for parameters for $r = 3$ reaching a constant probability of success, $\log T$ is a function in terms of $\log M$.
Plot its curve.

We have $\log M = \alpha \log N$*,* $\log T = \max(\alpha, \beta) \log N$*, and* $\alpha + 2\beta \geq 2$*. The minimal value for* $\log T$ *is reached for* $\beta = 1 - \frac{1}{2}\alpha$*, so* $\log T = \max(\alpha, 1 - \frac{1}{2}\alpha) \log N$ *which can be computed in terms of* $\log M$*. The curve looks like what follows.*



Q.6 We consider another algorithm $\mathcal{A}_4$ for making 3-collisions, defined by parameters α and β. Now, $\mathcal{A}_4$ runs N^α times a collision-finding algorithm and stores the N^α obtained collisions in the same form (y, L_y) with $L_y = (x_1, x_2)$ as before. In a second phase, $\mathcal{A}_4$ picks N^β random x and check if $f(x)$ hits one of the y in the hash table. If it is the case, a 3-collision is found. (We assume that no x is picked several times.)

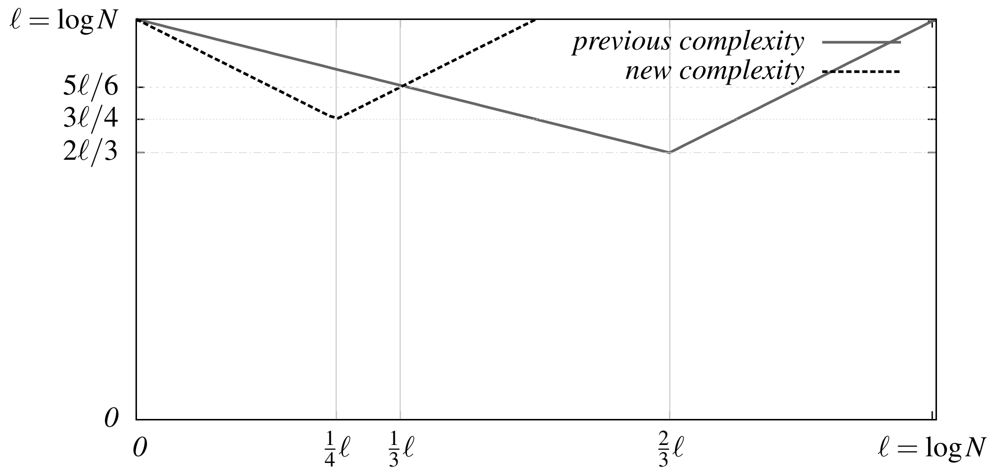
```
 1: **for** $i = 1$ to $N^\alpha$ **do**
 2:     run a collision-finding algorithm and get $x_1$ and $x_2$
 3:     set $y = f(x_1)$ and store $(y, (x_1, x_2))$ at place $h(y)$
 4: **end for**
 5: **for** $i = 1$ to $N^\beta$ **do**
 6:     pick a new $x$ at random
 7:     **if** there is an entry $(y, L_y)$ at place $h(f(x))$ such that $y = f(x)$ **then**
 8:         insert $x$ in list $L_y$
 9:         yield $L_y$ and stop
10:     **end if**
11: **end for**
12: algorithm failed
```

Q.6a Show that the memory complexity is $M \approx N^\alpha$ and that the time complexity in terms of f evaluations is $T \approx \max(N^{\alpha + \frac{1}{2}}, N^\beta)$.

When using a collision finding algorithm with constant memory complexity in Phase 1, the memory complexity is again $O(N^\alpha \log N)$ which is approximated by N^α.
When using a collision finding algorithm based on the birthday paradox, Phase 1 uses $O(N^\alpha + \sqrt{N})$ memory so the final memory complexity is approximated by $N^{\max(\alpha, \frac{1}{2})}$. This is not a very good choice since we get results which are not better than with the previous method.
The time complexity of a collision search is $O(N^{\frac{1}{2}})$. So, the time complexity of $\mathcal{A}_5$ is $O(N^{\alpha+\frac{1}{2}} + N^\beta)$ which is approximately $T \approx \max(N^{\alpha+\frac{1}{2}}, N^\beta)$.

Q.6b Show that for $\alpha + \beta \geq 1$ we obtain a constant probability of success.
Plot the curve of minimal $\log T$ in terms of $\log M$ to reach a constant probability of success.
Compare with $\mathcal{A}_3$.
When is it better?

Again, the logarithmic memory complexity is α. The logarithmic time complexity is now roughly $\max(\alpha + \frac{1}{2}, 1 - \alpha) \log N$. The curve looks like what follows.



As we can see, for $\alpha < \frac{1}{3}$, $\mathcal{A}_4$ has a better complexity.

2 Attack on some Implementations of PKCS#1v1.5 Signature with $e = 3$

This exercise is inspired by an attack originally presented by Bleichenbacher at CRYPTO'06 (unpublished), then improved by Kühn, Pyshkin, Tews, and Weinmann (in a technical available online). These attacks were later extended by Oiwa, Kobara, and Watanabe: "A New Variant for an Attack against RSA Signature Verification using Parameter Field", EuroPKI'07, LNCS vol. 4582, pp. 143–153, 2007.

In this exercise we represent bitstrings in hexadecimal by grouping bits into packets of 4, each packet (nibble) being denoted in hexadecimal with a figure between 0 and F. For instance, 2B represents the bitstring 0010 1011. Given a bitstring x, we denote by $\bar{x}$ the integer such that x is a binary expansion of $\bar{x}$. For instance, $\overline{\mathsf{00\,FF}} = 255$.

We call a *cube* an integer whose cubic root is an integer.

Given a message m and an integer ℓ_N, we define the bitstring of length ℓ_N

$$\mathsf{format}_{\ell_N}(m) = \mathsf{00\,01\,FF} \cdots \mathsf{FF\,00} \| D(m)$$

where $D(m)$ represents the identifier of the hash function H together with $H(m)$ following the ASN.1 syntax. As an example, in the SHA-1 case, we have

$$D(m) = \mathsf{30\,21\,30\,09\,06\,05\,2B\,0E\,03\,02\,1A\,05\,00\,04\,14} \| \mathsf{SHA\text{-}1}(m)$$

We denote by ℓ_D the bitlength of $D(m)$.

We recall that the PKCS#1v1.5 signature for a message m and a public key (e,N) is an integer s such that $0 \leq s < N$ and $s^e \bmod N$ can be parsed following the format $\mathsf{format}_{\ell_N}(m)$, where ℓ_N is the minimal bitlength of N. It is required that the padding field consisting of FF bytes is at least of 8 bytes.

Throughout this exercise we assume that $e = 3$.

Q.1 What is a signature scheme? Describe its components, its functionality, and give an intuition on its security.

*A signature scheme is defined by three algorithm: a key generator, a signature algorithm, and a verification algorithm. The **key generator** is a pseudorandom generator making a key pair of required length, one of the two keys is called the public key and the other is the secret key. The **signature algorithm** is a probabilistic algorithm taking a secret key and a message and producing a signature. The **verification algorithm** is a deterministic algorithm taking a public key, a message, and a putative signature, and telling whether the signature is valid.*
The signature scheme has the functionality such that when generating a key pair and making a signature on an arbitrary message m with the secret key, then the obtained signature passes the verification algorithm with m and the public key.
Security requires that it shall be impossible to create a valid signature without using the secret key.

Q.2 What is a valid signature for a message m in PKCS#1v1.5? Detail the verification algorithm.

A valid signature for m is a string which passes the verification algorithm. The algorithm first converts the string into an integer s. Then, it checks that $s < N$. Then, it computes the binary expansion of $s^3 \bmod N$. It parses this string into

$$\mathsf{00\,01\,FF\cdots FF\,00}\|D$$

and checks that D a valid ASN.1 string representing a digest using some valid hash function H. Finally, it checks that the digest equals $H(m)$.

Q.3 Let $u = \mathsf{format}_{\ell_N}(m)$.

Q.3a If $\bar{u}$ is a cube, show that we can easily forge a signature for m without any secret information.

If it is a cube, we can compute the cubic root s and s^3 has a valid format. Since, $s^3 \bmod N = s^3 = \bar{u}$, it is a valid signature.

Q.3b We assume that $\bar{u}$ looks like a random number less than $a = 2^{\ell_N - 15}$. How many cubes are less than a?
What is the probability for $\bar{u}$ to be a cube?

We have

$$u = \mathsf{00\,01\,FF\cdots FF\,00\,30\,21\,30\,09\,06\,05\,2B\,0E\,03\,02\,1A\,05\,00\,04\,14}\|H(m)$$

which is less than $2^{\ell_N - 15}$. The number of cubes less than a is exactly $\lfloor \sqrt[3]{a} \rfloor$. So, we roughly have $2^{\frac{1}{3}\ell_N - 5}$ cubes. Therefore, $\bar{u}$ is a cube with probability roughly $2^{-\frac{2}{3}\ell_N + 10}$.
Of course, u does not really look like random. We can do some thinner analysis and get that the probability for $\bar{u}$ to be a cube is $\frac{1}{3}2^{-\frac{2}{3}\ell_N+10}$. Indeed, $\bar{u}$ is between a and $a+b$ for $b < 2^{160}$ and $a \approx 2^{\ell_N - 15}$. The number of cubes between a and $a+b$ is roughly

$$(a+b)^{\frac{1}{3}} - a^{\frac{1}{3}} = a^{\frac{1}{3}}\left(\left(1+\frac{b}{a}\right)^{\frac{1}{3}} - 1\right) \approx a^{\frac{1}{3}}\frac{b}{3a}$$

so the probability to be a cube is about $\frac{1}{3}a^{-\frac{2}{3}}$.

Q.3c Deduce an algorithm to forge a signature for m which works with a success probability $2^{-\frac{2}{3}\ell_N+10}$.
It this practical?

The algorithm first computes u. If $\bar{u}$ is not a cube, the algorithm fails. Otherwise, it extracts the cubic root and produce a valid signature. Since it is easy to compute cubic roots over the integers, the algorithm is very fast. Its probability of success is $2^{-\frac{2}{3}\ell_N+10}$ which is much too low to be practical. E.g. for $\ell_N = 1\,024$, the probability is 2^{-673}.

Q.4 Bleichenbacher observed that some parsers just scan the bytes from the formatting rule but do not check that the string terminates after the digest. That is, these implementations accept the following format

$$\mathsf{00\,01\,FF\cdots FF\,00}\|D(m)\|g$$

where g is any garbage string, provided that the padding field has at least 8 bytes and that the total length (including the garbage) is ℓ_N.
In this question we assume $\ell_N = 3\ell$. We further assume that $\ell_N \geq 84 + 6\ell_D$.

Q.4a Let $P = \mathsf{FF}\cdots\mathsf{FF}$ be a string of FF bytes with bitlength ℓ_P. Show that the ℓ_N-bit string $u = \mathsf{00\,01}\|P\|\mathsf{00}\|D(m)\|\mathsf{00}\cdots\mathsf{00}$ is such that $\overline{u} = 2^{3\alpha} - x2^{\gamma}$ for some integer x, where $\alpha = \ell - 5$ and $\gamma = \ell_N - 24 - \ell_D - \ell_P$.

Let $x = 2^{8+\ell_D} - \overline{D(m)}$. We easily see that $2^{3\alpha} - x2^{\gamma}$ writes

$$\mathsf{00\,01}\|P\|\mathsf{00}\|D(m)\|\mathsf{00}\cdots\mathsf{00}$$

in hexadecimal.

Q.4b By using the assumption $\ell_N \geq 84 + 6\ell_D$, show that we can select ℓ_P such that $\gamma \geq 2\alpha$ and $x \leq 2^{\frac{1}{2}(3\alpha-\gamma)}$.

Since $\gamma - 2\alpha = \ell - 14 - \ell_D - \ell_P$, by selecting $\ell_p \leq \ell - 14 - \ell_D$, we have $\gamma \geq 2\alpha$.
Since $x \leq 2^{8+\ell_D}$ and $3\alpha - \gamma = 9 + \ell_D + \ell_P$, by selecting $\ell_P \geq \ell_D + 7$, we have $x \leq 2^{\frac{1}{2}(3\alpha-\gamma)}$.
When $\ell_N \geq 84 + 6\ell_D$, we have $\ell - 28 \geq 2\ell_D$, so $\ell_D + 7 \leq (\ell - 14 - \ell_D) - 7$. Therefore, we can select ℓ_P such that $\ell_P \bmod 8 = 0$ and $\ell_D + 7 \leq \ell_P \leq \ell - 14 - \ell_D$.

Q.4c We assume that $x \bmod 3 = 0$. Let $y = \frac{1}{3}x2^{\gamma-2\alpha}$ and $s = 2^{\alpha} - y$. Show that $\overline{u} \leq s^3 < \overline{u} + 2^{\gamma}$.

Note that since $x \bmod 3 = 0$ and $\gamma \geq 2\alpha$, y is an integer.
We use $(A-B)^3 = A^3 - 3A^2B + 3AB^2 - B^3$ with $A = 2^{\alpha}$ and $B = y$. Clearly, $A^3 - 3A^2B = \overline{u}$.
Since $s = A - B$, we have $s^3 = \overline{u} + 3AB^2 - B^3$. Thus, we only have to show that $0 \leq 3AB^2 - B^3 < 2^{\gamma}$.
Since $2^{3\alpha} - x2^{\gamma} \geq 0$, we have $A \geq 3B$, so $3AB^2 - B^3 \geq 0$.
Since $x \leq 2^{\frac{1}{2}(3\alpha-\gamma)}$, we have $3AB^2 < 2^{\gamma}$.

Q.4d Deduce an algorithm to forge signatures on a random message m with success probability $\frac{1}{3}$ based on Bleichenbacher's observation when 3 divides ℓ_N and $\ell_N \geq 84 + 6\ell_D$.

We take m hash it to compute $D(m)$, then x. If $x \bmod 3 \neq 0$, the attack fails (that is, with probability $\frac{2}{3}$). Then, we construct s as above. Due to the above inequalities, s^3 parses

$$\mathsf{00\,01\,FF}\cdots\mathsf{FF\,00}\|D(m)\|g$$

where g is some "garbage" string. So, it is accepted as a valid signature.

Q.4e Finally, apply the attack to $\ell_N = 3\,072$ with SHA-1. Show that the attack applies and that

$$s = 2^{1\,019} - \frac{1}{3}(2^{288} - \overline{D(m)})2^{34}$$

is a valid signature with probability $\frac{1}{3}$ over the random selection of the message.

For $\ell_N = 3\,072$, we have $\ell_N \bmod 3 = 0$. The $\ell_N \geq 84 + 6\ell_D$ constraint is equivalent to $498 \geq \ell_D$. For SHA-1, the digest length is 160. We can see that there is an overhead of 120 bits in the ASN.1 syntax of $D(m)$. So, we have $\ell_D = 280$ and the constraint is satisfied. We have $s = 2^{1\,019} - \frac{1}{3}x2^{730-\ell_P}$. By replacing x with its value, we obtain

$$s = 2^{1\,019} - \frac{1}{3}(2^{288} - \overline{D(m)})2^{730-\ell_P}$$

We must select ℓ_P such that $\ell_P \leq 730$, $\ell_P \geq 287$, and $\ell_P \bmod 8 = 0$. By selecting $\ell_P = 696 = 8 \times 87$ we obtain the expression in the question. Whenever its content of the parentheses is divisible by 3, s is a valid signature of the message. That is, with a probability $\frac{1}{3}$.