Security Protocols and Application — Final Exam
Solution

Philippe Oechslin and Serge Vaudenay

28.7.2019

- duration: 3h00
- no document allowed
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will not answer any technical question during the exam
- the answers to each 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!

The exam grade follows a linear scale. In each exercise, each question has the same weight. Both exercises have the same weight.

1 Finding Malicious Domain Parameters

This exercise is inspired from Galbraith-Massimo-Paterson, Safety in Numbers: On the Need for Robust Diffie-Hellman Parameter Validation, *PKC 2019, IACR, also https://eprint.iacr.org/2019/032.*

Let $n = 2^e d + 1$ where e and d are positive integers and d is odd. Let a be an integer such that $1 \leq a < n$. We say that n is a *pseudoprime to base a* if and only if

$$a^d \bmod n = 1 \quad \text{or} \quad \exists i \in \{0, 1, \ldots, e-1\} \quad (a^{2^i d} + 1) \bmod n = 0$$

We also define

$$S(n) = \{a \in \{1, 2, \ldots, n-1\}; n \text{ is a pseudoprime to base } a\}$$

It was proven that $\#S(n) \leq \frac{\varphi(n)}{2^{m-1}}$, where m is the number of pairwise different prime factors of n.

Q.1 Explain the acronyms CDH, TLS, PAKE, ECDH.

CDH: computational Diffie-Hellman problem. This is the problem of computing g^{xy} from input (g, g^x, g^y).
TLS: transport layer security. This is the IETF-standard protocol for secure TCP communication such as the https protocol.
PAKE: password-based authenticated key exchange. This is a class of cryptographic protocols which use a password as common input and produce a symmetric key as output. The lecture mentioned SRP and J-PAKE as examples of such protocols.
ECDH: elliptic-curve Diffie-Hellman. This is a variant of the Diffie-Hellman protocol adapted to elliptic curves.

Q.2 Explain what is a safe prime, a smooth number, and by which efficient algorithm we can compute discrete logarithms in a smooth ordered cyclic group.

> *A safe prime is a prime number p such that $\frac{p-1}{2}$ is also a prime number.*
> *A smooth number is an integer whose prime factors are all small.*
> *The Pohlig-Hellman algorithm computes discrete logarithms in a cyclic group. Its complexity relates the the largest prime factor of the order of the group. Hence, if the order is a smooth number, the complexity is small.*

Q.3 Explain what are Diffie-Hellman parameters and which mathematical properties we should normally verify on those parameters.

> *DH parameters consist of three elements (p,q,g), where p and q are two integers and g is a group element of $\mathbf{Z}_p$ (i.e. another integer). We must verify that*
> - *p and q are both prime,*
> - *g is an element of $\mathbf{Z}_p^*$,*
> - *g has order q in this group.*

Q.4 Compute $S(33)$.

> *For $n = 33$, we have $e = 5$ and $d = 1$. It is useful to have the table of the $x \mapsto x^2 \bmod n$ function:*

x	1	2	4	5	7	8	10	13	14	16
$x^2 \pmod n$	1	4	16	-8	16	-2	1	4	-2	-8

> *If we iteratively square any number from $\{\pm 2, \pm 4, \pm 5, \pm 7, \pm 8, \pm 13, \pm 14, \pm 16\}$, we never reach 1. If no power of $a \notin \mathbf{Z}_n^*$ can be equal to 1 (otherwise, the previous power of a would be an inverse of a modulo n). Thus, we can see that $S(33) = \{\pm 1, \pm 10\}$.*

Q.5 Depending on $\#S(n)$ and the number t of iterations, what is the probability of the Miller-Rabin primality test to be wrong when n is a composite number?

> *The Miller-Rabin test essentially picks a random a and check the pseudoprimality condition to base a. It repeats this t times. If n is composite, the probability to be wrong in one iteration is the probability to pick a in $S(n)$. Hence, the probability is $\frac{\#S(n)}{n-1}$. With t iterations, the probability to be wrong becomes*
>
> $$\left(\frac{\#S(n)}{n-1}\right)^t$$

Q.6 Explain the following quote:

"The primality test that OpenSSL uses [...] performs t rounds of random-base Miller-Rabin testing, where t is determined by the bit-size of p and q. Since p and q are 1 024 and 1 023 bits respectively, $t = 3$ rounds of Miller-Rabin are performed, at least in versions prior to OpenSSL 1.1.0i (released 14th August 2018). From version 1.1.0i onwards, t was increased to 5, with the aim of achieving 128 bits of security instead of 80 bits."

How was t computed?

This quote seems to mean that the probability to fool the primality test on one round is of order of magnitude 2^{-27}. For t rounds, it is 2^{-27t} which is roughly 2^{-80} for $t=3$ and 2^{-128} for $t=5$.
How 2^{-27} is obtained is not explained. The incorrectness bound is known to be $\varphi(n)/2^{m-1}$. So, the probability should be of order of magnitude $1/2^{m-1}$. It is known that for a random n, m is around $\ln\ln n$. For a 1024-bit number m is thus around 6.5.

Q.7 The quote of the previous question continues as follows:

> "For the DH parameter set [there is] a probability of approximately $1/2^8$ of being declared prime by a single round of Miller-Rabin testing. Hence this DH parameter set will be accepted by DH_check as being valid with probability approximately 2^{-24} (and the lower probability of 2^{-40} since version 1.1.0i of OpenSSL)."

Why is this not a contradiction with the previous quote?

The probability to pass is 2^{-8t} with $t=3$ then $t=5$. Actually, the 2^{-8} for one round comes from $1/2^{m-1}$ with $m=9$.
The previous quote was assuming a randomly generated n for which one iteration was wrong with probability much less than 2^{-8}. Here, we face to a maliciously generated one.

Q.8 Is this attack a threat to the Diffie-Hellman protocol? If not, when could it be a threat?

In the normal Diffie-Hellman protocol, keys are ephemeral and specific to the parameters. So, this does not seem to be of any threat.
This attack is a threat for PAKE. In this case, the password may leak from the protocol, although the protocol is normally made so that there is no leak when the Diffie-Hellman parameters are correct.

2 NSEC5 and Zone Enumeration

2.1 NSEC and NSEC3

Q.1 NSEC and NSEC3 have a weakness that NSEC5 aims to eliminate. Answer the following 3 questions:

- What is this weakness ?
- What advantage does NSEC3 give regarding this weakness ?
- Why is this not sufficient ?

- Existing zone names can be enumerated from the non-existance records.
- Instead of enumerating zone names, you can only enumerate hashes of names.
- Hashes can be cracked with dictionnaries or rainbow tables. The entropy of host names is too low.

2.2 NSEC5 properties

In NSEC5, PSR stands for Primary-Secondary-Resolver systems. Explain the following properties for a PSR system:

Q.2 Completeness:

When all parties follow the protocol, then the responses are correct and verification of the proof returns 1 with $P \geq 1 - \mu(k)$

Q.3 Soundness:

Even a malicious secondary cannot convince a resolver of a false statement.

Q.4 Privacy in NSEC5 is defined using f-zero knowledge proofs (f-zk proofs). Explain what the f means and what it is in NSEC5

We tolerate leaking $f(R)$ *in the proofs. In NSEC5,* $f(R) = |R|$

2.3 NSEC5 signatures

NSEC5 uses two key pairs, the primary and secondary keys. They are used for two different types of signatures. Let's call them primary signatures and secondary signatures.

Q.5 How many primary and how many secondary signatures must the primary resolver generate when setting up a zone with N host names ?

N primary for each answer, $N+2$ secondary for each element of a pair, including a lower and an upper bound, $N+1$ primary for each pair, including pairs with upper and lower bound.

Q.6 How many primary and how many secondary signatures must the secondary server generate when answering a request ?

When $x \in R$ then none, else one secondary.

Q.7 How many primary and how many secondary signature verifications must the resolver carry out to verify the answer ?

When $x \in R$ then one primary, else one primary (the pair) and one secondary (to verify π_y)

2.4 NSEC5 attacks

Q.8 Looking at the answers of the last two questions, describe a method for creating a denial of service on the secondary server. What is the cost for the attacker ?

Make many requests for an inexisting domain, do not verify the answer.

Q.9 Describe a method that allows an attacker to know the number of names that exist in a domain

Make many different requests, count the number of different responses.

Q.10 If a secondary server is compromised by an attacker, can the attacker
a) know all existing names of the domain ?
b) fake a positive response for a name that is not in the domain?
c) fake a negative response for a name that is in the domain?
Justify

Make many requests for an inexisting domain, do not verify the answer.

Q.11 What attack could an attacker carry out if he was in possession of the private key of a secondary server?

The same attack as on NSEC3: Get non-existance answers and brutefoce the y_j and y_{j+1} to find the existing servers.

Q.12 There is a very small probability that a fully functioning secondary server can not generate a proof of non-existence of a name. In what situation does this happen ?

When there is a collision on h_1 or h_2 resulting in a collision on y