Finding Malicious Domain Parameters

Jules Fasquelle

March 2019

Abstract

The studied paper [1] presents an attack on the Diffie-Hellman procedure, namely investigating for malicious parameters for the protocol. The paper explores what can be achieved once such parameters are accepted, but mostly focuses on how to get to this point. In particular, the question of fooling Miller-Rabin's primality test is widely addressed, with additional constraints based on OpenSSL's Diffie-Hellman validation procedure. The authors derive a detailed procedure to generate the malicious parameters and gave evidence of its feasibility.

1 Introduction

1.1 Diffie-Hellman and the Discrete Logarithm Problem

The security of the Diffie-Hellman key exchange relies on the Computational Diffie-Hellman (CDH) assumption, namely that in a cyclic group of order q with a generator G and given G, G^a, G^b with random $a, b \in \{0, ...q-1\}$, it is difficult to compute G^{ab}. [2]
In particular, one way to solve this problem is to derive a from G^a, which is known as the Discrete Logarithm Problem. It can be solved by the Pohlig-Hellman algorithm [3], which runs in $\mathcal{O}(\sqrt{B})$ where B is a bound on the greatest prime factor of q. Obviously, having q composite and smooth (having all its prime factors less than some pre-determined bound –as defined in [1]) allows this algorithm to run in reasonable time and makes the problem feasible. Thus, the goal of the attack is to make a target accept such a q as a Diffie-Hellman parameter.

1.2 Usual DH parameters requirements and verification

The parameters required to perform a DH key exchange are of the form (p, q, G) where p, q are prime numbers of cryptographic size (typically 1024 bits being the legacy standard level according to [4]), q the order of the subgroup induced by G and p the modulus under which operations are done.
It is required that p is of the form $kq + 1$, and the **safe prime** requirement enforces $k = 2$, which removes one degree of freedom to the attacker. It is on this setting, enabled by default on OpenSSL, that the authors focus.

2 Constructing a pseudoprime q

2.1 Miller-Rabin primality test and how to fool it

The goal is to derive a composite and smooth number q that will be detected as prime with a good probability. OpenSSL uses the Miller-Rabin primality test, which operates as follow: the target q is tested for a property all prime numbers satisfy, with respect to a base a. If successful, the test is repeated with an other base, and after several (typically 3 to 5, depending on the implementation) successful rounds q is declared probably prime.
Previous work has shown that if the bases are known in advance, one can find a composite q which always passes the test. Nowadays in practice, the bases are drawn at random so one can only hope to find a q having good probability (but strictly less than 1) to pass the test. This depends on the number of potential bases a which are not *witnesses* for the compositeness of q, i.e the number of bases which yields a successful Miller-Rabin round for q. The authors provide an upper bound for this quantity, namely $\frac{\varphi(q)}{2^{m-1}}$, with m the number of prime factors of q and φ Euler's totient function.
The authors argue that the composite numbers q which maximize this proportion of non-witnesses (and as a consequence the probability of success) are the Carmichael numbers with prime factors congruent to 3 mod 4.

2.2 Finding Carmichael numbers in two steps

2.2.1 The Erdös method

This procedure allows to draw a Carmichael number from an arbitrary composite number L. It makes use of the Korselt's criterion about Carmichael numbers. The idea is to derive from the factors of L a list of prime numbers which product is a Carmichael number if congruent to 1 mod q. In particular, a careful choice of L can ensure that the resulting Carmichael number has its factors congruent to 3 mod 4.
This method allows to quickly generate small Carmichael numbers, but scales very badly to greater sizes.

2.2.2 The Granville-Pomerance method

This routine allows to draw a (cryptographically) large Carmichael number N from a smaller one n, while preserving useful properties such as the condition on the factors being 3 mod 4. Yet, its success relies on the primality of all elements of a collection $\{q_i\}$ derived from the prime factors of n, which is not guaranteed. It follows that by enforcing constraints on the free parameters of the method, we can achieve **sieving** on the $\{q_i\}$.

About sieving
It is possible to increase the probability of a number to be prime by constructing it so it is non-divisible by some given primes (for example, by the h first primes, which will give good results even for a small h).
To do so, we simply set the target number as $q_i = ka + b$, with a a number we want to sieve by, b coprime to a and k a free parameter.
It follows that $q_i = b \mod a$. Setting a as the product of the first h primes increases the probability for q_i to be prime.

Using this technique, we can ensure that the $\{q_i\}$ are coprime with all the primes we want, and draw a Carmichael number N (which is simply the product of the $\{q_i\}$). From an expected number of trials of $2^{43.8}$ with a naive computation, the sieving procedure allows to reduce this to 2^{32} for a target number N of 1024 bits with 8 factors of 128 bits.

3 Using the pseudoprime

3.1 The additional "safe-prime" requirement

Once we have a pseudoprime q which is likely (relatively to other composite numbers) to pass the Miller-Rabin primality test, we can draw a $p = kq + 1$ by just trying various k (in the case k is free) and test the result for primality. But in the "safe-prime" case ($k = 2$), one q is linked to one p and if the latter is not prime, computational resources have been wasted.

Fortunately, it appears that we can filter a lot of bad candidates for q and p by testing a divisibility criterion *before* applying the heavy Granville-Pomerance procedure, which saves a lot of time.

More formally, we define a subset $\{s_j\}$ of the primes we used to sieve for the $\{q_i\}$ in the last step of the Granville-Pomerance procedure[1].
For any s_i, we have

$$p(= 2q + 1) \equiv 2n + 1 \mod s_i \tag{1}$$

– n being the small Carmichael number we obtained with the Erdös method.
In other words, if $2n+1$ is divisible by any s_i, p will also be, and there is no need to even compute q as it would yield p non-prime. Moreover, it appears that we have some freedom about the $\{s_i\}$ and that it is possible to test divisibility for p –or equivalently for $2n + 1$– by any prime which is not a factor of n.

The authors of the original paper show that with every such prime, this property induces constraints on the value of n and consequently on the choice of L, the composite number used as a starting point in the Erdös method. They investigate the prime 3 (which we want to be part of the $\{s_j\}$ since it is the smallest odd prime) and find out that it should not divide L. They even suggest to remove it from the factors of n in some cases.
It follows that after such filtering, the probability for p to be prime is indeed beyond the baseline for odd numbers.

[1] We do not provide here the full technicalities used in the original paper. In the latter, the reader will find the detailed computation of the $\{q_i\}$ and a formal definition for the $\{s_i\}$.

3.2 The attack in application

The authors imagine two scenarios in which such malicious parameters can compromise the security:

- A TLS session between a client and a server. In this scenario, a malicious developer has hardcoded the computed DH parameters in the server and is able to decrypt the traffic. In practice, the attack would succeed in one out of 2^{24} attempts and the attacker cannot gain further advantage than eavesdropping.
- A PAKE protocol (SRP, J-PAKE) in which the attacker impersonates the server. In this scenario, the client's secret used in the DH protocol depends on her password, which the attacker can derive offline after aborting.

4 Conclusion and further observations

It is observed that such an attack is also applicable to elliptic-curves settings (ECDH) instead of finite fields. The key and modulus lengths required in such groups being smaller, the computation of malicious parameters would be way faster. Yet, protocols using elliptic curves operate on a well-known set of curves and it would be suspicious to suggest a new one, not to mention the difficulty of constructing it. The authors of the original paper conclude by observing that the actual security level of Diffie-Hellman parameter verification, notably on OpenSSL, is well under what is targeted. They recommend to limit the possible parameters to a fixed set –with the risk that one could carry out large pre-computations for discrete logarithms– and/or to significantly increase the number of rounds in the Miller-Rabin primality test.

References

[1] S. Galbraith, J. Massimo, and K. G. Paterson, "Safety in numbers: On the need for robust diffie-hellman parameter validation," 2019.

[2] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[3] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over gf (p) and its cryptographic significance (corresp.)," *IEEE Transactions on information Theory*, vol. 24, no. 1, pp. 106–110, 1978.

[4] "BlueKrypt cryptographic key length recommendation," https://www.keylength.com/en/3/.