Finding malicious domain parameters

Jules Fasquelle

Original paper :
Safety in Numbers: On the Need for Robust Diffie-Hellman Parameter Validation, Steven Galbraith, Jake Massimo, and Kenneth G. Paterson

Reminder : Diffie-Hellman key exchange

- Alice and Bob agree on parameters *(p, q, G)*

  -- p a prime, G a generator for a group of order q
- Alice sends $\mathbf{A = G^a \quad mod\ p}$ to Bob
- Bob responds $\mathbf{B = G^b \quad mod\ p}$
- Alice computes $\mathbf{B^a = G^{ba} \quad mod\ p}$
- Alice computes $\mathbf{A^b = G^{ab} \quad mod\ p}$

Given $\mathbf{G^a}$ or $\mathbf{G^b}$, trying to derive $\mathbf{G^{ab}}$ is difficult (*Computational Diffie–Hellman assumption*)

Deriving **a** or **b** is the **Discrete Logarithm** Problem (DLP) Infeasible ?

It depends on **p** and **q** !

Diffie-Hellman requirements about *(p, q, G)*

- Operations are **mod p**
- *G* induces a subgroup of order *q*
- Implementations test if *p* is a good prime and therefore *(p,q,G)* a good group

Most implementations (e.g. default OpenSSL) require *p* to be a ***safe prime:***

p = 2q + 1 with q also prime

In some cases, **p = kq + 1** is accepted

- *p*, *q* should be of cryptographic size (p≈3072 bits)- Source : https://www.keylength.com/en/3/

The attacker's arsenal 🔪🔨🔫

- The **Pohlig-Hellman** algorithm:

  Solves the Discrete Logarithm Problem…

  in O(√*b*), with *b* a bound on the largest prime factor

- Cryptographically large composite numbers looking like primes

  A composite number is said **smooth** if its factors are bounded

The Goal: Initiate a DH exchange with such a parameter, and solve the DLP!

Miller-Rabin primality test and how to fool it

<u>Input:</u> a number ***n***
<u>Output:</u> whether ***n*** is prime, with high probability

- Pick a base ***a*** — How is it chosen ?
- Test if a property holds for *n* with respect to this base
- Repeat

$$a^d \equiv 1 \pmod{n}$$

or

$$a^{2^r \cdot d} \equiv -1 \pmod{n}$$

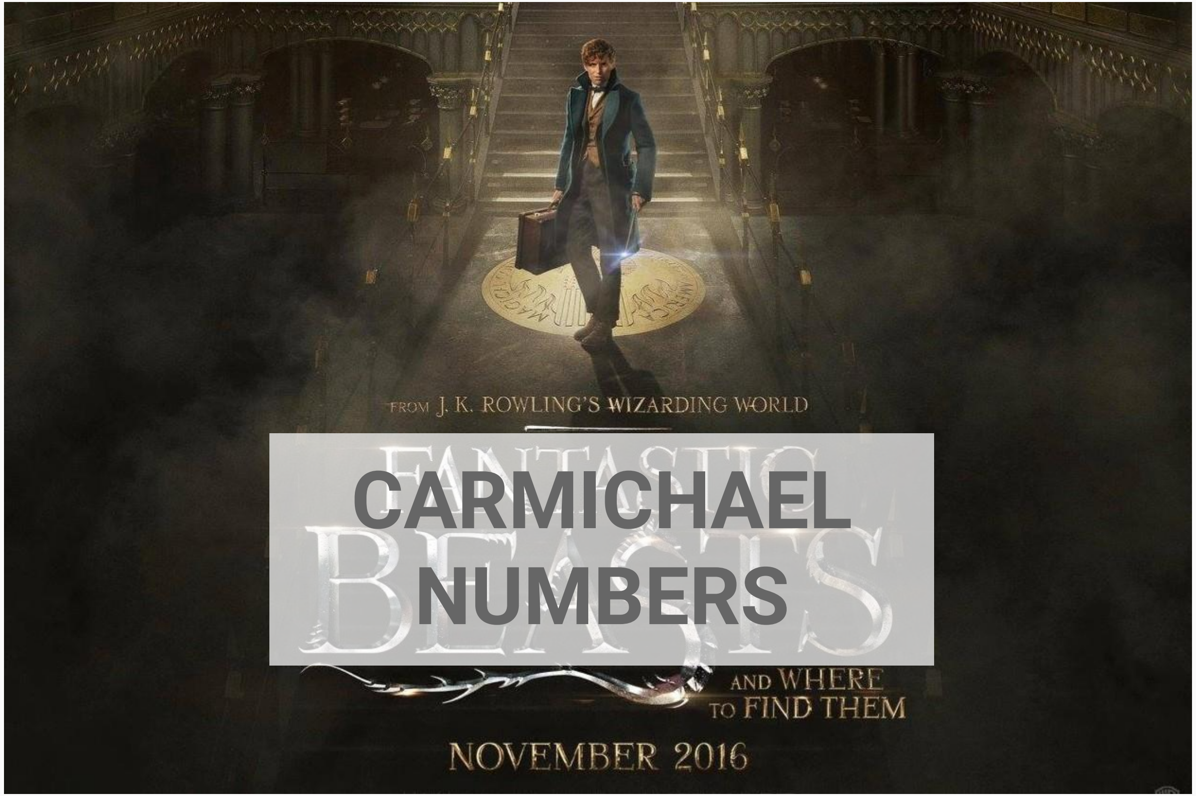
a is a base
r and *d* are derived from *n*

  - Bases which fail the test are called **witnesses** *(for compositeness)*
  - If no test has failed, declare *n* prime

We are looking for smooth numbers *n* with few witnesses

Recap

- We want to break Diffie-Hellman
- In this case, by solving the DLP
- Feasible for non-prime parameters
- We need to fool the primality test performed in the DH handshake
- Find composite numbers with maximum non-witnesses



Carmichael numbers

A composite number *n* which satisfies $b^{n-1} \equiv 1 \pmod{n}$ for *b* coprime to n.

Theorem 3 (Korselt's Criterion). *Let n be odd and composite. Then n is a Carmichael number if and only if n is square-free and for all prime divisors p of n, we have $p-1 \mid n-1$.*

The upper bound on the number of non-witnesses is $S(n) \leq \frac{1}{2^{m-1}}\varphi(n).$ (with Euler's totient function)

The Carmichael numbers reach it on the additional constraint of having prime factors congruent to 3 mod 4.

Finding Carmichael numbers

- The **Erdös Method** allows to generate Carmichael numbers, but of small size
- The **Granville-Pomerance method** allows to draw a much bigger Carmichael number from a smaller one

The Erdös method

Recall:

- Korselt's criterion:
  n is a Carmichael number iff p_i - 1 divides n-1 for all prime factors p_i of n
- We need a Carmichael number congruent to 3 mod 4 to reach the max probability of looking prime

- Select a composite number L
- Define $\mathcal{P}(L) = \{p : p \text{ prime}, p - 1 \mid L, p \nmid L\}$
- Find a product $n = p_1 p_2 \cdots p_m$ such that n = 1 mod L
- It is a Carmichael number!
  - Proof by Korselt's criterion
  - The number of possible products is $\binom{|\mathcal{P}(L)|}{m}$
    - The smoother L, the more possible products
- If we choose L to be 2 mod 4, we have n = 3 mod 4

Scales very badly !

1024-bits n with 8 factors, all bounded to 128 bits -> $L \simeq 2^{128}$

The Granville-Pomerance method

Recall:

- We want to draw large Carmichael numbers from smaller ones
- We want them smooth so they really help with the Discrete Logarithm
- And to keep the $p_i = 3 \bmod 4$ property!

- Take a Carmichael number N and its prime factors p_i
- Set **L' = lcm({p_i - 1})** and **M = 1 + kL'** for a free k
- Construct **{q_i = 1+ M(p_i - 1)}**
- For prime $q_1, \ldots, q_m$, $N = q_1 \cdots q_m$ is a Carmichael number
- If we had $p_i = 3 \bmod 4$, we have $q_i = 3 \bmod 4$
  - Proof: L' is even, M is odd…

Where is the difficulty ?

Primes distribution and sieving

The probability that a random choice of *M* yields *m* primes smaller than *B* is $(2/\ln(B))^m$

- Our previous example (1024-bits number with 8 factors of 128-bits each) gives 2^{-43}…

What constraints do we need on *M* to lift up this probability ?

Sieving is constructing a number *p* of the form $p = kH + \delta$ with H the product of some primes $s_1, \ldots, s_h$. and δ coprime to H.

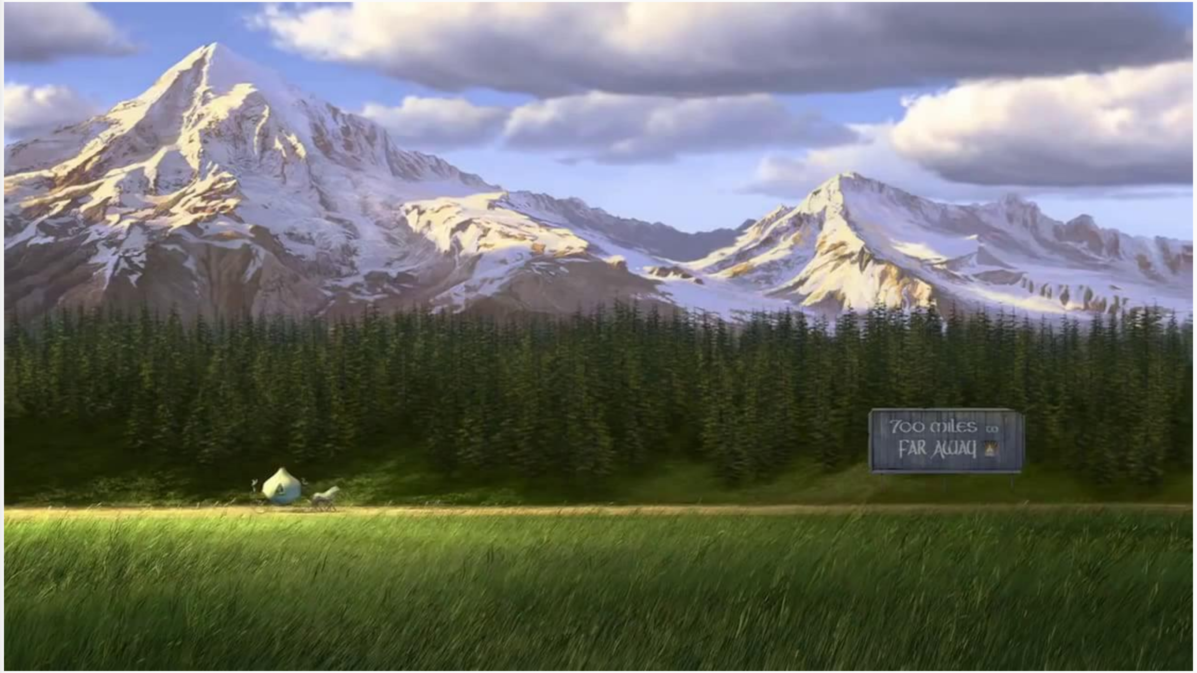
It yields *p* coprime to any s_i , increasing its chances to be prime, particularly if $s_1, \ldots, s_h$ are the first *h* primes.

Sieving for the q_i

- Recall we have **M = kL' + 1** and **q_i = 1+ M(p_i - 1)**, with p_i prime factors of our small Carmichael number.
- We want the q_i to be prime, i.e to not be divisible by any other prime
    - We already have q_i coprime to all p_j --recall **L' = lcm({p_i - 1})**
    - We have $q_i = kL'(p_i - 1) + p_i$, so if we choose *k* as the product of some primes, we achieve sieving on them.
- Sieving on 3, 5, 7, 11, 13, 17 gives a 8-primes family { q_i} in 2^{32} trials (in expectation) instead of 2^{43} !
- Recall we wanted prime { q_i } so $N = q_1 \cdots q_m$ is a Carmichael number.

Recap #2

- We want to find malicious parameters which will help break Diffie-Hellman security
- Carmichael numbers fool the primality test with highest probability
- We have a method to generate small Carmichael numbers, and a method to draw bigger ones
- We learned about **sieving** in the process



Are we there yet ?

Back to the Diffie-Hellman requirements: “safe prime”

- We wanted (p, q, G) such that:
  - q looks prime but is actually a large smooth Carmichael number
  - $p = kq + 1$ is prime

If we are free to choose k, this is feasible by just trying a lot of different values.

- The “safe prime” requirement demands **k = 2**

Which makes it way harder since if $2q + 1$ is not prime, we have to generate a new q and try again...

What to change to pass as safe prime

Luckily, we can (once again) improve the probability that $2q+1$ is prime

- Say n is our small Carmichael number and q is the big one. Recall:

$n = p_1 \cdots p_m$, $q_i = M(p_i - 1) + 1$, $L' = \mathrm{lcm}(p_i - 1)$ and $M = 1 + kL'$.

Lemma 2. *With notation as above, for all primes s dividing kL, we have that* $2q + 1 = 2n + 1 \pmod{s}$.

As a consequence, we can move the test for coprimality with the $\{ s_i \}$ to n, before computing q.

(The idea, as before, is that ensuring p is not divisible by some primes lifts the chance that p is prime itself)

Here, L' being smooth, the set of si is large and the condition will discard a lot of composite numbers!

Assemble!

<u>Requirements:</u>

(p, q, G) such that:

- p, q pass the primality tests
  - $p = 2q + 1$
- Discrete Logarithm solving is possible on order q
  - q composite, smooth
  - q fools the primality test with correct probability
    - It's a Carmichael number
    - Its factors are 3 mod 4
  - q is big enough

- Run the Erdös method to generate a small Carmichael number n
  - Pick L, define $\mathcal{P}(L) = \{p : p \text{ prime}, p-1 \mid L, p \nmid L\}$
  - Construct n = 1 mod L by product of some p_i
  - Before that, filter P(L) by removing the number 3
- Derive **L' = lcm({pi - 1})** and check for each of its prime factors if it divides 2n+1
  - If yes, go back to step 1
- Use n in the Granville-Pomerance method
  - Remember $q_i = kL'(p_i - 1) + p_i$
  - Sieving doesn't work on the p_i !
  - If we want to surely sieve by a specific x, we must make sure x is not a p_i
  - Now we can choose k as a multiple of x
- Test $p = 2q + 1$ for primality

Application to cryptographic sizes

- For $p = 2q+1$ a prime of 1024 bits, q having 9 factors bounded to 121 bits:
  - q fools Miller-Rabin primality test with probability 2^{-8}
  - The Pohlig-Hellman algorithm solves the DLP in 2^{64} operations
  - Generating such q with 9 factors yields:
    - Standard Granville-Pomerance needs 2^{48} trials to succeed, enhanced one needs 2^{34}
    - q yields a $2q+1$ prime in 2^{7} trials
    - The authors found one in $2^{38.15}$ trials total (136 core-days on 3.2GHz GPUs)

- For similar p but q having 11 factors bounded to 100 bits each:
  - q declared prime with probability 2^{-10}
  - q being smoother, the Discrete Logarithm is solved in 2^{54}
  - Generated q in $2^{44.83}$ trials (1680 core-days on 3.3 GHz GPUs)

Elliptic curves setting

- Elliptic Curve Diffie-Hellman exchanges are also defined by a prime, a generator and the order q of the induced subgroup
- The idea is still to fool a primality test but then to reconstruct a curve with the algorithm of Bröker and Stevenhagen
- Once we have the curve, Pohlig-Hellman solves the DLP

How hard is it?

- Required key lengths are smaller in this setting
- Thus we can have fewer factors for q and pass as prime with high probability, but:
- “Suitable parameter generation is non-trivial”
- “Safe and efficient implementation is much easier with a limited and well-understood set of curves”

Conclusion: the attacks

- To compromise honestly established TLS sessions: “evil developer”
  - The malicious parameters should be hardcoded into a server for later use
  - This will not work very often and careful validations can avoid it

- To impersonate a server and recover past passwords of a client
  - PAKE scenario where the client uses OpenSSL’s DH validation (3 to 5 random bases for Miller-Rabin)
  - For specific PAKE protocols (SRP, J-PAKE), it allows to recover a secret since the client sends password-dependent protocol messages
  - SRP specifications warn against malicious DH parameters!