University of BRISTOL

Calling the Bluff in the Diffie-Hellman Key Exchange

Author Name: Iman Syed
Supervisor: Prof. Tim Dokchitser
Level of Study: M/7
Credit Points: 20CP

Date: May 25, 2020

This work complies with the guidelines given in the project handbook on preparing and handing in the project report.

Acknowledgement of Sources

For all ideas taken from other sources (books, articles, internet), the source of the ideas is mentioned in the main text and fully referenced at the end of the report.

All material which is quoted essentially word-for-word from other sources is given in quotation marks and referenced.

Pictures and diagrams copied from the internet or other sources are labelled with a reference to the web page,book, article etc.

Signed: 
Dated: May 25, 2020

Calling the Bluff in the Diffie-Hellman Key Exchange

Iman Syed

Abstract

The Diffie-Hellman (DH) key exchange is the first method of securely generating and exchanging keys over an insecure channel to come into widespread use. The generated keys are most often used in procedures known as symmetric-key algorithms that allow users to transmit information in the form of messages securely between each other. In particular, Elliptic-Curve Diffie-Hellman (ECDH) key exchanges are used for key pair generation in blockchain implementations such as Bitcoin and Ethereum. ECDH has numerous advantages over other key generation protocols, notably that it generates key pairs of comparably much shorter bit lengths. One implementation that relates to blockchain relates to sharing bitcoin addresses. A bitcoin receiver can publish some relevant ECDH information that the bitcoin sender can use to calculate a shared secret. This shared secret will be the bitcoin address of the receiver. Then, the receiver can compute the private key that allows access to that address. There are many other protocols that build from and rely on the same mathematical tools as DH and ECDH (such as the Elliptic Curve Digital Signature Algorithm), and thus its security is of great importance. The Standards for Efficient Cryptography Group (SECG) is an international consortium to develop commercial standards for efficient and interoperable cryptography based on elliptic curve cryptography (ECC). ECC is the larger subject of secure communications with elliptic curves in which the ECDH protocol lies. SECG give recommendations for parameter sets to be used in cryptographic implementations including DH and ECDH, and this paper supports the use of such recommendations.

In this paper, we consider the problem of maliciously chosen Diffie-Hellman parameters that are robust enough to pass given approaches to parameter validation but in which the Discrete Logarithm Problem (DLP) is relatively easy to solve. Such a value that passes as a probable prime, but is not in fact prime is known as a pseudoprime. The security of any DH implementation depends on the hardness of DLP and we explore a scenario in which a malicious developer could exploit this knowledge. We will not discuss in much depth the notion of hardness, but instead we will adopt a roughly computational definition of hardness. We will say that a problem is hard if there is no existing computer than can solve it in a reasonable amount of time or within a reasonable number of trials (more specifically, we say the problem cannot currently be solved in polynomial time). We consider the security of DH in two settings - finite-field encryption and then more briefly in the context of elliptic curve encryption.

In the finite-field setting, I will discuss existing methods of malicious DH parameter generation and explore what characteristics malicious parameters (p, q, g) should have. For these parameters q has many small prime factors (and so seems to be a sensible choice, for reasons we will go on to discuss), but fools random-base Miller-Rabin primality tests with a reasonably high chance, and g is the generator

for the cyclic subgroup of order $q \bmod p$. This is achieved as an amalgamation of known methods for generating Carmichael numbers. I provide an extension of these methods into the safe prime setting.

In the elliptic curve setting I describe how the algorithm of Bröker and Stevenhagen can be used to output an elliptic curve E of order n, over the finite-field $\mathbb{F}_p$. We intentionally select n to be of the form hq with h being some small co-factor and q having many small prime factors (and so seems to be a sensible choice, for reasons we will discuss as above) but fools random-base Miller-Rabin primality testing with significant probability, and E has a point of order q that should be used as the generator g. I will explain and exemplify how the attacker can use this to solve the ECDH problem.

In the finite-field safe prime case, I specify how an implementation would work but fall short of providing an implementation (but I do give an example implementation in the finite-field case without the safe prime constraint). In the elliptic curve setting I provide an implementation originally given in [Alb+18] but similarly fall short of providing an implementation in the elliptic curve safe prime setting.

The aim in doing so is to show the importance of performing effective DH parameter validation on cryptographic protocols if such protocols do not rely on standardised parameter sets with tested and known security, such as the ones given by SECG. We also briefly discuss the counterargument that attackers will favour the greater reward of successfully attacking widely used parameter sets.

Overview:

- Introduction: A summary of the finite-field DH algorithm, and the motivation to prevent the use of incorrectly validated DH parameter sets.
- Understanding pseudoprimes: A discussion of the Miller-Rabin primality test and the nature of pseudoprimes, and its relation to Carmichael numbers.
- Carmichael numbers: Discussing the Erdös method and the method of Granville and Pomerance for constructing Carmichael numbers., and how the two can be combined to produce Carmichael numbers of cryptographically interesting size. I will describe an adaptation that should help, in addition, to generate numbers satisying the congruence conditions discussed in the previous section.
- The safe prime setting: Using the methods in the previous section to produce parameter sets in the safe prime setting that have a good chance of passing validity tests, but for which the Discrete Logarithm Problem is relatively easy to solve.
- The Elliptic Curve setting: Describing and giving an example of how the algorithm of Bröker and Stevenhagen can be used to generate malicious DH parameters.
- Conclusions: Key findings and achievements are summarised and areas for further investigation are detailed.

Contents

1 Introduction

In the following section, I summarise the Diffie-Hellman (DH) algorithm and discuss the motivation to prevent the use of incorrectly validated DH parameter sets.

1.1 Motivations

The Diffie-Hellman key exchange [DDH76] is an algorithm for exchanging a shared key securely through an insecure channel. It allows two parties with no prior knowledge of each other to output a shared key over the channel, and the keys can go on to be used in the secure communication of messages (eg. in a symmetric key encryption protocol like the RSA). It finds application in SSL, SSH, TLS and other Public Key Infrastructure systems.

Albrecht et. al [Alb+18] explore primality testing and its relation to the Diffie-Hellman Key Exchange. They found several cryptographic libraries for which the chosen primality testing was susceptible to error. One example they noted was in instances with reliance on fixed-base Miller-Rabin primality testing, or if falling short of full reliance, instances where too few applications of the Miller-Rabin test lead to pseudoprimes slipping though (the DH exchange, what pseudoprimes are, and why this a problem will be described later on). They use these findings to show that when Diffie-Hellman is applied in the finite-field case, we can generate particular DH parameter sets of the form (p, q, g) in which $p = kq+1$ for some k, p is prime, q is composite (it can be written as the product of two smaller positive integers) but q passes a Miller-Rabin primality test with some probability, and the Discrete Logarithm Problem (DLP) is easy to solve using the Pohlig-Hellman algorithm in the order q subgroup generated by g. This could be used to compromise the security of the algorithm, as we will go on to understand. The goal of this paper is to build on this work, and similar pieces of work, in discouraging DH implementations from allowing the use of non-standard and poorly vetted parameter sets.

In practise, the implementation of procedures such as DH and ECDH depends on cryptographic libraries to both implement the protocols, and validate the security of the parameters. Thus, the security of the procedure in practise depends on the effectiveness of the parameter validation procedures. If a compromised parameter set was used in some symmetric key cryptographic protocol, an attacker could determine this key and thereby break the security of the protocol. OpenSSL is an example of a widely used cryptographic library that contains such parameter validation functions, amongst other functions. The parameter set given in [Alb+18] passed an OpenSSL DH parameter validation function with significant probability where the function only performed a few iterations of Miller-Rabin primality testing. [Alb+18] suggest a scenario plausibly occurring when a malicious developer hard-codes the DH parameter into the relevant protocol that she wishes to exploit.

Definition 1. *Safe prime. A safe prime number is a prime number of the form* $2p + 1$ *for some prime* p

[Alb+18] go on to point out that their methods for producing a malicious set of DH parameters do not work in instances where p is a safe prime number. Using safe primes in the finite-field setting helps to prevent certain attacks, such as small subgroup attacks (see [LL97]), and appears to make parameter validation easier. OpenSSL's Diffie-Hellman validation routine `DH_check` is one example of a cryptographic library implementation that insists on the use of safe primes in applications of DH; in particular, it requires DH parameters (p, q, g) for which $p = 2q + 1$, and both p, q are tested for primality.

The motivation behind this paper is to follow on from [Alb+18] in suggesting techniques a potential attacker could exploit find a large, composite q passing the Miller-Rabin primality test with a significant probability, where $p = 2q + 1$ is prime but for which the DLP can be efficiently solved.

1.2 The Diffie-Hellman Key Exchange

We will now provide the scheme, discuss why it is correct and finally why it is secure.

Algorithm 1. The Basic DH Algorithm [DDH76].

$$
\begin{array}{cc}
\mathcal{A}(g,q): & \mathcal{B}(g,q): \\
a \xleftarrow{\$} \mathbb{Z}_q & b \xleftarrow{\$} \mathbb{Z}_q \\
A \leftarrow g^a & B \leftarrow g^b \\
\multicolumn{2}{c}{\xrightarrow{A}} \\
\multicolumn{2}{c}{\xleftarrow{B}} \\
K_A \leftarrow B^a & K_B \leftarrow A^b \\
\text{return } K_A & \text{return } K_B
\end{array}
$$

Where $x \xleftarrow{\$} X$ means 'x is chosen uniformly at random from set X' and $y \xleftarrow{\$} Y$ means y is assigned value Y.

We can show correctness as follows:

$$
\begin{aligned}
K_A &= B^a \\
&= (g^b)^a \\
&= g^{ba} \\
&= (g^a)^b \\
&= A^b \\
&= K_B
\end{aligned}
$$

The fourth equality follows from $\mathbb{Z}_q$ being an abelian group. In order to examine why the scheme is secure, we introduce the following definitions:

Definition 2. *The Discrete Logarithm Problem (DLP). given a group G and with generator g and some element $y \in \langle g \rangle$, find x so that $g^x = y$*

Definition 3. *The Computational Diffie-Hellman Problem (CDHP). Given a group G with generator g and some elements $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$ (but not x_1 and x_2), find $y = g^{x_1 x_2}$*

It is important (yet simple) to note that if there is a machine which efficiently solves the DLP (in this case, we call the DLP "computationally easy") then we can also efficiently solve the CDHP (just use the machine to compute x_2 from y_2 then calculate $y = (y_1)^{x_2}$). There are a few cases in which the DLP is computable in a feasible amount of time, however there is no known efficient method for computing a solution in general.

The security of DH depends on this fact. A potential eavesdropper only has the values of g, p, $A = g^a$ and $B = g^b$, but must compute $K = g^{ab}$. which is precisely a statement of the CDHP. Since we have established that if the DLP can be solved efficiently then the CDHP can be solved efficiently, for the remainder of this paper will centre our discussions around solving the DLP. We will say that the DLP is hard to solve when there does not exist a machine that can solve it in polynomial time, and easy otherwise.

If an adversary can trick users into using parameters (g, q) such that g is a generator of a small subgroup, the problem become much simpler for the adversary, as he can conduct a Lim-Lee style subgroup attack [LL97] or even do a brute force search for the logs a, $b \in \mathbb{Z}_q$, and succeed in determining $K = K_A = K_B$. For this reason, in this paper we will concentrate instead on the revised DH scheme first given by [Sch91] which ensures that the subgroup used is of sufficiently large prime order. The revised algorithm works by ensuring that the order of a chosen cyclic subgroup, $p-1$, has some large prime divisor q. There will then be a cyclic subgroup of order $q \bmod p$, and we choose g to be a generator of this subgroup. If it is chosen to be sufficiently large (in practice this means at least 160 bits [Alb+18]) a brute-force search will be unfeasible. Then, a user who is given a DH parameter set can easily check whether they are being fooled into using a non-secure small subgroup.

Algorithm 2. The DH Scheme against small subgroup attacks [Sch91].

$$\begin{array}{cc} \mathcal{A}(p,g,q): & \mathcal{B}(p,g,q): \\ a \xleftarrow{\$} [2,q-2] & b \xleftarrow{\$} [2,q-2] \\ A \leftarrow g^a \bmod p & B \leftarrow g^b \bmod p \end{array}$$

If $1 < B < p$ and $B^q \bmod p = 1$, we have failed, and we terminate. Otherwise,

$$\begin{array}{cc} K_A \leftarrow B^a \bmod p & K_B \leftarrow A^b \bmod p \\ \text{return } K_A & \text{return } K_B \end{array}$$

We can show correctness as follows:

$$\begin{aligned} K_A &= B^a \bmod p \\ &= (g^b)^a \bmod p \\ &= g^{ba} \bmod p \\ &= (g^a)^b \bmod p \\ &= A^b \bmod p \\ &= K_B \end{aligned}$$

The checks in before returning K_A and K_B protect from two things. We know that if the public keys $B = 0$ or $B = 1$ are used then the channel is totally insecure (since, if define $K_A = K_B := Z$ we will always have either $Z = 0$ or $Z = 1$, so we make sure $B > 1$. The same problem arises in the case where $B = kp$ or $B = 1 + kp$ for some non-zero integer k, so we also check that $B < p$. Further if we check that $B^q \bmod p = 1$ we can be sure that $g^q \bmod p = 1$. Thus we will meet the requirement that g is a generator for the subgroup of order $q \bmod p$. Finally, we have $B^q = (g^b)^q = (g^q)^b = 1^b = 1 \bmod p$.

2 When can we solve the DLP?

A key purpose of this paper is to detail how one can determine DH and ECDH pseudoprime DH parameters for which the DLP is easy to solve. We will focus on finding DH parameters to which the Pohlig-Hellman algorithm (PH) can be applied to the corresponding subgroup, which efficiently solves the DLP in subgroups with smooth order (that is, the prime factorisation contains only small primes). There are certainly other avenues to be explored for potential attacks on the DLP but for the sake of continuity from [Alb+18] we focus on the PH algorithm. The original Pohlig-Hellman algorithm is given in [PH78] but in what follows we draw from an explanation given by [Mus06] as it is simpler and easier to follow. The algorithm takes time $O(B^{1/2})$ to complete where B is a bound on the largest prime factor of q [GMP19]. For example, if q only has 3 prime factors then q can have a maximum of 384 bits if we want the algorithm to apply at most 2^{64} effort.

2.1 The Pohlig-Hellman Algorithm

The Pohlig-Hellman algorithm provides a method for solving the DLP in the scenario where the group with generator g is a finite abelian group whose order is a smooth integer q. The algorithm operates by computing a logarithm modulo each prime power in the group order, and applying the Chinese remainder theorem (to combine these to a logarithm in the full group).

Algorithm 3. The Pohlig-Hellman Algorithm for groups of prime-power order [PH78; Mus06].
Suppose that p is a prime number and α a generator for the cyclic group $\mathbb{F}_p^\times$. Assume that $\beta \in \mathbb{F}_p^\times$ is such that $\beta = \alpha^x$. We want to solve $x = \log_\alpha \beta$. Assume further that

$$p - 1 = \prod_{i=1}^{k} q_i^{r_i}$$

where each q_i is are prime numbers in the factorization of $p-1$. The general idea is to solve a system of congruences modulo each q_i , and then combine the congruences, using the Chinese Remainder Theorem (CRT), giving a solution to the original congruence.

Begin by fixing choices q, r and we will work mod q^r. Recall that the DLP is to search for a solution to $x = \log_\alpha \beta$. The following trick is key to the algorithm. Write x as follows:

$$x \equiv x_0 + x_1 q + x_2 q^2 + x_3 q^3 + \ldots + x_{r-1} q^{r-1} \mod q^r \text{ for } 0 \leq x_i \leq q-1 \qquad (1)$$

By considering x in this form we can successively compute the x_i 's. We can then take this equation for x, multiply by the constant $\frac{p-1}{q}$ and obtain

$$x\left(\tfrac{p-1}{q}\right) \equiv x_0\left(\tfrac{p-1}{q}\right) + x_1(p-1) + x_2 q(p-1) + x_3 q^2(p-1) + \ldots + x_{r-1}q^{r-2}(p-1)$$

or simply

$$x\left(\tfrac{p-1}{q}\right) \equiv x_0\left(\tfrac{p-1}{q}\right) + (p-1)m, \text{ for some } m \in \mathbb{Z}$$

(Where the above congruences hold true modulo q^r). The next step is to work with the generator α and our solution β all modulo p. Raising β to the exponent of $\frac{p-1}{q}$, we get:

$$\beta^{\frac{p-1}{q}} \equiv (\alpha^x)^{\frac{p-1}{q}} \equiv \alpha^{x_0\left(\frac{p-1}{q}\right)+(p-1)m} \equiv \alpha^{x_0\left(\frac{p-1}{q}\right)} \equiv \left(\alpha^{\frac{p-1}{q}}\right)^{x_0} \mod p$$

It is now possible to run through a list of stored values in an attempt to find a match for x_0. The stored values are obtained by setting $c \equiv g^{\frac{p-1}{q}} \mod p$ and computing $c^j \mod p$ where $0 \leq j \leq q-1$. The CRT implies that one and only one value will be congruent to $\beta^{\frac{p-1}{q}}$. When a match is found we are able to solve for x_0, namely $j = x_0$. Once we have solved for x_0 we can perform a similar trick to solve for x_1 this time we multiply (1) by the quantity $\frac{p-1}{q^2}$ to obtain

$$x\left(\frac{p-1}{q^2}\right) \equiv x_0\left(\frac{p-1}{q^2}\right) + x_1\left(\frac{p-1}{q}\right) + (p-1)m_1, \text{ for some } m_1 \in \mathbb{Z}. \qquad (2)$$

Unfortunately we cannot immediately raise β to this exponent; we first need to shift it somehow to compensate for the extra power of q that we have on the left hand side of (2). To do this, we set $\beta_1 = \beta \cdot \alpha^{-x_0}$. Thus we have that

$$\begin{aligned}
\beta_1^{\frac{p-1}{q^2}} &\equiv \left(\alpha^x \cdot \alpha^{-x_0}\right)^{\frac{p-1}{q^2}} \\
&\equiv \left(\alpha^{q(x_1+x_2q+\ldots)}\right)^{\frac{p-1}{q^2}} \equiv \alpha^{\frac{p-1}{q}\left(x_1+x_2q+x_3q^2+\ldots\right)} \\
&\equiv \alpha^{\left(\frac{p-1}{q}\right)x_1+(p-1)m_1} \equiv \alpha^{\left(\frac{p-1}{q}\right)x_1} \mod p
\end{aligned}$$

and again we can look in our table of precomputed values and determine x_1. This is then repeated until all the x_i 's are known and for every prime q_i appearing in the factorization of $p-1$. A final application of the CRT applied with each q_i gives us the final value of x allowing us to solve the DLP in this case.

There are two issues that we should be aware of for this attack. The first is that we are assuming that $p-1$ can be factored efficiently and that it contains only small primes in its factorization. The second issue lies in the precomputation of the c^j values. since the $p-1$ contains only small primes, the values of the q_i 's are small and hence so are the values of the c^j 's. since these values are small, they can be efficiently computed, and require relatively small storage for each list. The expected running time for this algorithm is $O\left(\sum_{i=1}^{k} r_i\left(\log_2(p-1) + \sqrt{q_i}\right)\right)$, or equivalently $O(\sqrt{q_i'})$ where q_i' is the largest prime factor of $p-1$. To avoid this attack in its entirety, one can choose the value of p carefully so that $p-1$ has a large prime factor.

3 Understanding pseudoprimes

We now go on to discuss the nature of pseudoprimes, how they should be determined and give some important results that we will eventually use to manipulate them.

3.1 The Miller-Rabin Primality Test

Clearly, we can only hope to fool a primality test in the way we are hoping for if the test is not a deterministic one. There exists a deterministic primality-proving algorithm known as the AKS primality test [AKS04] which can be conducted in polynomial time. However, it is not implemented by existing cryptographic libraries as although it runs faster than any other algorithm for problems that are sufficiently large, in reality "sufficiently large" problems are so big that the algorithm is never used in practice. We will focus on the Miller-Rabin primality test [Rab80], which is the most widely implemented probabilistic primality test and is implemented, for example, in the function `DH_check`, which is OpenSSL's Diffie-Hellman parameter validation procedure. The test works by taking a particular system of congruences and translates them over to the setting of composite numbers to see if these new congruences fail. However, as we will go on to discuss, there are values amongst a type of number known as the Carmichael numbers that have good chances of fooling this test.

Theorem 1. [Rab80]. Let n be prime and $1 \leq a \leq n-1$. Factor out the largest power of 2 from $n-1$, and we denote this $n-1 = 2^e d$ where $e \geq 1$ and k is odd. Then

$$a^d \equiv 1 \bmod n \text{ or } a^{2^i d} \equiv -1 \bmod n \text{ for some } i \in \{0, \ldots, e-1\}. \tag{3}$$

Proof. Let n be an odd integer such that $n > 1$.
We now show that the polynomial $x^{n-1} - 1 = x^{2^2 d} - 1$ can be repeatedly factorised as often as there continues to be powers of 2 in the exponent:

$$\begin{aligned} x^{2^2 d} - 1 &= (x^{2^{e-1} d})^2 - 1 \\ &= (x^{2^{e-1} d} - 1)(x^{2^{e-1} d} + 1). \\ &= (x^{2^{e-2} d} - 1)(x^{2^{2-2} d} + 1)(x^{2^{e-1} d} + 1). \\ &\vdots \\ &= (x^d - 1)(x^d + 1)(x^{2d} + 1)(x^{4d} + 1) \cdots (x^{2^{e-1} d} + 1) \end{aligned}$$

Suppose n is prime and $1 \leq a \leq n-1$. Then by Fermat's little theorem $a^{n-1} - 1 \equiv 0 \bmod n$. Thus:

$$\left(a^d - 1\right)\left(a^d + 1\right)\left(a^{2d} + 1\right)\left(a^{4d} + 1\right) \cdots \left(a^{2^{c-1} d} + 1\right) \equiv 0 \bmod n$$

So we must have that at least one of these factors is 0 mod n, and the result follows.

□

We will use this result to justify our next definition.

Definition 4. *Miller-Rabin Witness. Let $n > 1$ be odd. Write $n-1 = 2^e d$ with d odd (as in the proof of the last theorem) and pick $a \in \{1, \ldots, n-1\}$. a is known as a Miller-Rabin witness for n if both of the congruences in (3) are false. In particular:*

$$\textbf{\textit{both}}\ a^d \neq 1 \bmod n\ \textbf{\textit{and}}\ a^{2^i d} \not\equiv -1 \bmod n\ \textbf{\textit{for all}}\ i \in \{0, \ldots, e-1\}$$

Definition 5. *Miller-Rabin Nonwitness. Let $n > 1$ be odd. Write $n - 1 = 2^e d$ with d odd (as in the proof of the last theorem) and pick $a \in \{1, \ldots, n-1\}$. a is known as a Miller-Rabin nonwitness for n if one of the congruences in 3 is false. In particular:*

$$\textbf{\textit{either}}\ a^d \neq 1 \bmod n\ \textbf{\textit{or}}\ a^{2^i d} \not\equiv -1 \bmod n\ \textbf{\textit{for some}}\ i \in \{0, \ldots, e-1\}$$

The term 'witness' here means some number that proves n is composite. An odd prime must have no Miller-Rabin non-witness, so if n has a Miller-Rabin witness it is certainly composite and thus is certainly not prime. However, the converse is not strictly true; if n has no witnesses, it is only likely to be prime because, as we will go on to see, there are certain pseudoprimes that return no witnesses.

Theorem 2. [Rab80]. Let $n > 1$ be an odd composite.
Over 75% of integers from 2 to $n-2$ are Miller-Rabin witnessses for n. Equivalently, less than 25% of integers from 2 to $n-2$ are Miller-Rabin nonwitnesses.

We now formulate the Miller-Rabin test as follows, to decide whether an odd number $n > 1$ is prime. Steps 3 and 5 are a direct consequence of Theorems 1 and 2 respectively.

1. Select some $t \geq 1$ to be the total number of trials for the test.
2. Set $a \leftarrow [2, n-2]$
3. If a is a Miller–Rabin witness for n, terminate the test. We determine with certainty that n is composite.
4. If a is not a Miller–Rabin witness for$n$ then return to 2 and repeat with a different value of $a \leftarrow [2, n-2]$
5. If the test has not terminated after t trials we conclude that n is prime with probability $1 - 1/4^t$ or greater

We now proceed to examine the relationship between a composite n and its non-witnesses. Let us denote the number of non-witnesses a composite n posseses as $S(n)$. In order to get a composite that fools the Miller-Rabin test for any random base a then it is clearly desirable for $S(n)$ to be as large as possible. On the other hand, four our purposes we need to find an n for which it is possible to efficiently solve the DLP in a subgroup of order n, so we attempt to find an n which is relatively smooth, in order to apply the Pohlig-Hellman algorithm. We can calculate S(n) exactly via the following theorem:

Theorem 3. [Mon80]. Suppose n is an odd composite. Suppose $n = 2^e d + 1$ for some odd d. Further, suppose that n has prime factorisation $n = \prod_{i=1}^{m} p_i^{q_i}$ where each prime p_i can be expressed as $n = 2^{e_i} d_i + 1$ with each d_i odd. Then:

$$S(n) = \left(\frac{2^{\min(e_i) \cdot m} - 1}{2^m - 1} + 1 \right) \prod_{i=1}^{m} \gcd(d, d_i) \tag{4}$$

The following result is a bound on the number of non-witnesses $S(n)$.

Theorem 4. [Mon80] Monier-Rabin Bound. Let $n \neq 9$ be an odd composite. Then:

$$S(n) \leq \frac{\varphi(n)}{4}$$

where φ is the Euler's totient function: $\varphi(n)$ is the number of positive integers that are relatively prime to n.

We also know from [Mon80] that the Monier-Rabin bound reaches equality for composites of the form $n = (2k + 1)(4k + 1)$ with each multiple prime, and k odd.

Definition 6. *Carmichael numbers. Let n be an odd composite number. Then n is a Carmichael number if $a^{n-1} = 1 \bmod n$ for all a co-prime to n.*

Another case of equality is in the instance where n is a Carmichael number can be written as a prime factorisation of exactly three primes $n = p_1 p_2 p_3$ and where each factor p_i is congruent to 3 mod 4.

Theorem 5. Ss given by [Alb+18]. Korselt's Criterion. Let n be an odd composite. Then n is a Carmichael number if and only iff

Its prime factorisation does not have any repeated factors (ie. n is square free).

AND for all prime divisors p of n , we have $p - 1 \mod n - 1$.

m	C_m	$S(C_m)$
3	$7 \cdot 19 \cdot 67$	$\varphi(C_m)/4$
4	$7 \cdot 19 \cdot 67 \cdot 199$	$\varphi(C_m)/8$
5	$7 \cdot 11 \cdot 19 \cdot 103 \cdot 9419$	$\varphi(C_m)/16$
6	$7 \cdot 11 \cdot 31 \cdot 47 \cdot 163 \cdot 223$	$\varphi(C_m)/32$
7	$19 \cdot 23 \cdot 31 \cdot 67 \cdot 71 \cdot 199 \cdot 271$	$\varphi(C_m)/64$
8	$11 \cdot 31 \cdot 43 \cdot 47 \cdot 71 \cdot 139 \cdot 239 \cdot 271$	$\varphi(C_m)/128$
9	$19 \cdot 31 \cdot 43 \cdot 67 \cdot 71 \cdot 103 \cdot 239 \cdot 307 \cdot 631$	$\varphi(C_m)/256$
10	$7 \cdot 11 \cdot 19 \cdot 31 \cdot 47 \cdot 79 \cdot 139 \cdot 163 \cdot 271 \cdot 2347$	$\varphi(C_m)/512$

Table 1: [Pin06]. The smallest number C_m with m prime factors that meets the upper bound of $\varphi(C_m)/2^{m-1}$ on $S(C_m)$.

3.2 The relationship between nonwitnesses and prime factorisation

We begin with a key result following on from Theorem 4.

Theorem 6. [GMP19]. Factor bound on $S(n)$. Let n be an odd composite with prime factorisation $n = \prod_{i=1}^{m} p_i^{q_i}$. Write $n = 2^e d + 1$ where d is odd and $p_i = 2^{e_i} d_i + 1$ where each d_i is odd. Then $S(n) \leq \frac{\varphi(n)}{2^{m-1}}$, where $\varphi(\cdot)$ denotes Euler's function, with equality if and only if n is square-free.

Notice that the case of Theorem 6 in which $m = 2$ is precisely the statement of the Monier-Rabin bound. The next theorem will show that for $m \geq 3$, the bound given in Theorem 6 is reached if and only if n is a Carmichael number whose prime factors are all congruent to 3 mod 4.

Theorem 7. [Mon80]. Let n be a Charmichael number with $m \geq 3$ prime factors, each congruent to 3 mod 4. Then $S(n) = \frac{\varphi(n)}{2^{m-1}}$. Conversely, if n has $m \geq 3$ prime factors and $S(n) = \frac{\varphi(n)}{2^{m-1}}$, then n is a Carmichael number whose prime factors are all congruent to 3 mod 4.

Table 1 provides, for each $3 \leq m \leq 10$, the least number with m prime factors C_m such that C_m reaches the factor bound of Theorem 6. Theorem 7 implies that all such values of C_m are Carmichael numbers with prime factors each congruent to 3 mod 4. We note that this table is simply for the purpose of exemplifiying the factor bound, but they cannot be put into cryptographic use as they are much too small.

4 Producing Carmichael Numbers

The results that were given in the last section give us sufficient reason to search for appropriately sized Carmichael numbers, with each factor congruent to 3 mod 4. To recap, we wish to do so in order to maximise the number of nonwitnesses to any such number n and so increase the chance that n is accepted by the Miller-Rabin test as a probable prime. The next section will discuss two existing methods for determining such numbers: The Erdös method and the method of Granville and Pomerance.

4.1 The Erdös Method.

[Erd56] provide one method of constructing Carmichael numbers with a large number of small prime factors. In the following section, we will begin with the notation L for an initial highly composite number (one with many prime factors). We will also retain notation for the following set: $P(L) = \{p : p \text{ prime}, p-1 \mid L, p \nmid L\}$.

Lemma 1. *[Erd56]. If for some subset $\{p_1, p_2, \cdots, p_m\} \subseteq P(L)$, we have $p_1 p_2 \cdots p_m = 1 \bmod L$, then $n = p_1 p_2 \cdots p_m$ is a Carmichael number.*

Proof. . By construction, $p_i - 1 \mid L$. Clearly n is an odd composite. Since $n = 1 \bmod L$ is equivalent to the claim $L \mid n-1$; it follows that $p_i - 1 \mid n-1$. Also, n is evidently square-free (all p_i are distinct). The result follows by satisfaction of Korselt's criterion (Theorem 5). □

Now we are in a position to present what is a relatively simple and efficient approach to generating Carmichael numbers with a chosen number of prime factors m, and for reasonably large values of L as given by [Erd56].

1. Select an appropriate composite L (In the next subsection we consider how to choose this value).

2. Generate $P(L)$. We do this by considering each factor $a-1$ of L and applying the Miller-Rabin primality test on a:

   (a) If a is not prime, it does not belong to $P(L)$ so we discard it and move on to the next value.

   (b) If a turns out to be prime, it belongs to $P(L)$.

3. After detemining $P(L)$, examine all product combinations of m distinct elements of $P(L)$. Call one such product $n = p_1 \cdots p_m$. Then if $n = 1 \bmod L$, n is a Carmichael number.

Table 2, as given by [GMP19] gives examples of Carmichael numbers that can be produced by the Erdös method, where L_{best} is the value less than or equal to a given L_{bound} such

L_{bound}	L_{best}	$\lvert\mathcal{P}\left(L_{\text{best}}\right)\rvert$
2^{20}	$810810 = 2 \cdot 3^4 \cdot 5 \cdot 7 \cdot 11 \cdot 13$	39
2^{21}	$2088450 = 2 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 13 \cdot 17$	50
2^{22}	$4054050 = 2 \cdot 3^4 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13$	58
2^{23}	$7657650 = 2 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	65
2^{24}	$13783770 = 2 \cdot 3^4 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	73
2^{25}	$22972950 = 2 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	89
2^{26}	$53603550 = 2 \cdot 3^2 \cdot 5^2 \cdot 7^2 \cdot 11 \cdot 13 \cdot 17$	93

Table 2: For a given L_{bound} (column 1), the value L_{best} (column 2) gives the value of $L \leq L_{\text{bound}}$ resulting in the largest set of primes $\mathcal{P}(L)$, subject to the additional restriction that $p = 3 \bmod 4$ for all $p \in \mathcal{P}(L)$

that the set $\mathcal{P}(L_{best})$ is of the largest possible size subject to the additional restriction that $p = 3 \bmod 4$ for all $p \in \mathcal{P}(L)$. $|\mathcal{P}\left(L_{\text{best}}\right)|$ represents the number of prime factors of an outputted Carmichael number n.

4.2 Selecting values for L

Here follow the work of [GMP19] in discusing how we should selected the initial composite L for our purposes. It is fairly clear that we should choose L to be even, otherwise the integers a for which $a - 1 \mid L$ will all be even. Secondly, to ensure all primes $p \in P(L)$ satisfy $p = 3 \bmod 4$ we should set $L = 2 \bmod 4$. This is because then each factor f of L must also be $2 \bmod 4$, and so $p = f + 1 = 3 \bmod 4$ as required.

Thirdly, in order to produce a Carmichael number n with m prime factors we need to find some product $n = p_1 p_2 \cdots p_m$ such that $p_1 p_2 \cdots p_n = 1 \bmod L$. We can see that the number of possible such products is

$$\begin{pmatrix} |\mathcal{P}(L)| \\ m \end{pmatrix}$$

Thus, to maximise the possible values returned by this method and thus maximise the chance of finding a Carmichael number within them we should find L such that $\mid P(L) \mid$ is as large as possible. Such an L would have many factors $f - 1$ and so many possible candidates f that are considered in step 2 of the Erdös method, and will be added to $P(L)$ if prime. Thus, we should choose L to have as many factors as possible. In this argument we have assumed that the primality of the different values of f will be in some some independent from the factors $f - 1$ of L.

It is important to remember that for our purposes, any method we use to generate Carmichael numbers must produce numbers of cryptographically relevant size. However, it turns out in practice that it is difficult to use the Erdös method to do so. To illustrate,

suppose we wanted to construct a 1024-bit Carmichael number n with, say, $m = 8$ prime factors, each of 128 bits. This would mean using an L much larger than 2^{128}. This would make it very long to conduct a search to find a product $p_1 \cdots p_8 = 1 \bmod L$.

4.3 The method of Granville and Pomerance

In this section we will consider another method of generating Carmichael numbers, which is attributed to Granville and Pomerance [GP01]. The method takes a small Carmichael with some known number of factors m, and outputs a larger Carmichael number also with m factors.

Theorem 8. [GP01] Granville and Pomerance. Let $n = p_1 p_2 \cdots p_m$ be a Carmichael Number. Let $L = lcm(p_i - 1)$ and let M be any integer with $M = 1 \bmod L$. Set $q_i = 1 + M(p_i - 1)$. Then $N = q_1 \cdots q_m$ is a Carmichael number whenever each q_i is prime.

However, this theorem alone is incomplete for our purposes, since as mentioned, we only desire Carmichael numbers whose prime factors are congruent to 3 mod 4. We remedy this concern with the following lemma; using the same notation as Theorem 8:

Lemma 2. *[Alb+18] If $p_i = 3 \bmod 4$, then $q_i = 3 \bmod 4$ for all $i \in \{1, \cdots, m\}$*

Proof. Choose some $i \in \{1, \cdots, m\}$i. Since $L = lcm(p_i - 1)$ and $p_i - 1$ is always an even number, it follows that L is an even number. But since $M = 1(\bmod L)$ we must have that M is odd; write $M = 2s + 1$. Moreover, since $p_i = 3 \bmod 4$, we have $p_i - 1 = 2d_i$ with d_i odd; write $d_i = 2t_i + 1$. Then $q_i = 1 + M(p_i - 1) = 1 + (2s+1)(4t_i+2) = 3 + 4(2st + i + s + t_i)$. The result follows. □

We can see that the the two crucial choices of variable in this method are the choice of M, and the input Carmichael number n.

It is fairly clear from the Granville and Pomerance theorem that the initial choice n affects the properties of the outputted Carmichael number, for example by determining the number of prime factors m of N. The result of Lemma 2 is also determined by n.

The choice of M affects the output in a couple of ways. Firstly, it is necessary to choose an M so that all the $q_i = 1 + M(p_i - 1)$ are prime (by the definition of a Carmichael number). If we use the heuristic assumption that the values the even q_i are as likely to be prime as random choices of odd q_i of the same size, the probability that a random choice of M yields m primes is approximately $(2/ln(B))^m$ where B is a bound on the q_i . So this probability will be very small for cryptographically sized N with even moderately sized m. So we should not hope to obtain Carmichael numbers of suitable cryptographic size by making guesses. Instead, we now try to develop methods for determining M that will increase the likelihood that all the q_i are prime.

4.4 Choosing M

As we can infer from Theorem 8, there is only one limitation on the choice of the value of M That is, $M = 1(\mathrm{mod} L)$, where $L = lcm(p_i - 1)$. By being particular in our choice of M we should be able to choose such a value, and also ensure that the resulting values $q_i = 1 + M(p_i - 1)$ has a greater likelihood of being prime than if M was randomly chosen.

In pursuing this task we first return to the work of [GMP19] who found use in techniques from [JPV00] for generating primes on low-end processors. First, we consider numbers of the form $p = kH + \delta$ for a free parameter k where H is the product of the first h distinct primes, , $H = \prod_{i=1}^{h} s_i$, and where δ is co-prime to H.

We can be sure that p is divisible by each s_i since $p = \delta = 0 \bmod s_i$, To generate different candidates for p, we can choose different values for k, and then test for primality. An important technique is known as 'sieving' a positive integer n by some primes $x_1, x_2, \cdots, x_\gamma$ where $x_i \neq n$ for all i. By this, we mean that we are checking if n is divisible by each of the x_1, and if it is not then we say it has passed sieving by each x_i thus improving the likelihood that n is prime. The process by which each generated number p is tested by trial divisions by each of the small primes diving H is an example of sieving. By this process we can say that the numbers generated have a greater probability of being prime than a randomly generated counterpart, since by construction they are guaranteed to pass sieving by the primes $s_1, s_2, \cdots, s_h$.

[GMP19] present an adaption of this method such that the resulting $q_i = 1 + M(p_i - 1)$ are guaranteed to be indivisible by many small primes, as follows.

Since $M = 1 \bmod L$, we can write $M = kL + 1$, where k is the new free parameter in the construction method. Then

$$q_i - 1 = M(p_i - 1) = (kL + 1)(p_i - 1) = kLp_i + p_i - kL - 1.$$

Which is equivalent to

$$q_i = kLp_i + p_i - kL = kL(p_i - 1) + p_i.$$

We can observe now that many small primes will divide L since $lcm(p_i - 1, p_j - 1) = L \neq j$. If we use the Erdös method to generate the starting Carmichael number n, we will end up with even more small primes that divide L since we will start with a smooth number with all of the $p_i - 1$ divide.

We cannot have that any of the primes diving L can be a value of p_i, again since $lcm(p_i - 1, p_j - 1) = L$ for $i \neq j$. Let p be a prime dividing L. We know that for each such prime p

$$q_i = p_i \neq 0 \bmod p.$$

Thus, we can be sure that every q_i does not have any of the prime divisors of L as a factor: so any sieving on q_i for every such divisor will pass the sieving process. .

Despite this, there are still other primes which are neither equal to any of the p_i, nor that divide L, which must be considered. Let s be such a prime. Suppose we choose some k such that $s|k$.

Since we have $M = kL + 1$ for some free parameter k, we get:

$$q_i = kL(p_i - 1) + p_i = p_i \neq 0 \bmod s$$

Thus, we should choose k so that it is divisible by some product of the primes s_j, such $s_j \neq p_i$ for all i, j, and s_j does not equal any of the divisors of L for any j. This ensures we can pass sieving by all of the s_j. We could certainly includes additional constraints for k to ensure that the generated q_i are of a desired bit-size and so that there are ultimately enough choices for M, but we will not discuss them here. Let us write $k = k' \prod_j s_j$ for some collection of primes s_j that obey the above constraints; we now choose k' instead of k as the free parameter in the construction.

The effectiveness of sieving will depend on the collection of two sources of primes: the prime factors present in L, and the set $\{s_1, \cdots, s_j\}$. Let us call the total collection of primes from these wo sources $s_1, ..., s_h$. Then the fraction of candidates for each q_i that are successfully removed by the sieving (since they are not prime) is given as follows:

$$\sigma = 1 - \prod_{i=1}^{h} \left(1 - \frac{1}{s_i}\right) \tag{5}$$

Thus, the prime values of q_i will be within $1 - \sigma$ of the initial set of candidates, so that a random selection amongst these values that were not removed by sieving is $1/(1 - \sigma)$ times more likely to result in a prime than a random selection amongst the initial set. Notice that the prime $s = 3$ is very powerful in sieving, contributing $2/3$ as a factor in the product $\prod_{i=1}^{h} \left(1 - \frac{1}{s_i}\right)$ that determines (5). This construction is an improvement on the success probability of each trial from the modified Granville-Pomerance construction (each with some choice difference choice of free parameter k' from $(2/ln(B))^m$ to $(2/(1 - \sigma)ln(B))^m$.

In this section we have explained how the modified method of Granville and Pomerance given by [GMP19] shows that it is possible to generate large Carmichael numbers n with the following properties:

1. n is of a cryptographically interesting size
2. n has a selectable number of small prime factors (and so can fulfil some smoothness bound to be a candidate for a subgroup order to which the Pohlig-Hellman algorithm

can be applied)

3. n achieves the upper bound in Theorem 4 for the number of its Miller-Rabin non-witnesses and therefore maximises the probability of being a Miller-Rabin pseudo-prime

4.5 An example of the modified GP method

Using a C implementation of the modified Granville-Pomerance construction, with the Carmichael number C_8 of Table 1 as the starting value n and $L = 53603550$, [GMP19] found that choosing

$$k = 7891867750444302551322686487$$

produces the 8 -factor, 1024 -bit Carmichael number $N = q_1 \cdots q_8$ where:

$$\begin{aligned}
q_1 &= 761457829597791649244915744232411931 9 \\
q_2 &= 9306706806195231268548970207285034723 \\
q_3 &= 1776734935728180514904803403208961174 3 \\
q_4 &= 1006816463579302291779388595151744665 39 \\
q_5 &= 3629615654416140194734098380841163541 59 \\
q_6 &= 3926584207959278937939615521091804194983 \\
q_7 &= 4850486374537932805690113290760464005567 \\
q_8 &= 1026064425383024247357523965353175078100 51
\end{aligned}$$

Here, q_s, the largest prime factor, has 137 bits.

With B of 128 bits and $m = 8$ (so that the target N has 1024 bits), they estimated the standard Granville-Pomerance construction to have a success rate for producing an appropriate Carmichael number of $(2/\ln(B))^m \approx 2^{-43.8}$ per trial, so that the expected number of trials would be about $2^{43.8}$. With their modified version of the Granville-Pomerance construction each of the q_i passed sieving by the primes 3,5,7,11,13,17 that divide L. This gave them $\sigma = 0.6393$ and therefore a reduction in the expected number of trials by a factor of about $1/(1-\sigma)^m \approx 2^{11.8}$ to roughly 2^{32} trials.

5 The Safe-Prime Setting

5.1 The primality of $2q + 1$

In the previous section, we showed that in our pursuit of finding malicious DH parameters in the safe prime setting (and any finite-field setting) we need to construct a number q so that q passes Miller-Rabin primality testing, but has sufficiently many small prime factors so that the adversary can use the Pohlig-Hellman algorithm to solve the DLP the subgroup generated by q. The final parameter set will be of the form (p, q, g) where p is prime and $g \in \mathbb{Z}_p$ generates a group of prime order q, where $q \mid p - 1$. The randomly generated values of a and b given by Algorithm 1 are then generated by this latter subgroup. In the safe prime setting, we have the additional requirement that $p = 2q + 1$.

[Ble05] use the results given in the last section to construct a large Carmichael number q with m prime factors f_i, and for which $f_i = 3 \bmod 4$, and q will pass random-base Miller-Rabin primality testing with the highest possible probability of all other Carmichael numbers with m prime factors. We build on this work with an additional step in the safe prime setting, which is to then test $2q + 1$ for primality and accept the value if it passes. If $2q + 1$ is indeed prime then the DLP in the subgroup of order q can be solved with $O(mB^{1/2})$ effort where B is an upper bound on the prime factors of q.

There are two reasons why this approach will fail in practice. Firstly, standard density estimates for primes tell us that the probability that $2q+1$ is prime by chance is approximately $1/\ln q$ and thus that it is very unlikely [VS13]. Secondly, for certain constructions there is reason to believe that $2q + 1$ will never be prime. In what follows I will describe further and attempt to resolve these issues.

5.2 Choice of sieving primes

First, we will evaluate the problem of sieving for $2q+1$ where the prime q is produced by the Method of Granville and Pomerance:

Assume we have some starting Carmichael number $n = p_1 \cdots p_m$. Suppose we apply the method of Granville and Pomerance: Set $q_i = M(p_i - 1) + 1$ where $M = 1 + kL$ and $L = lcm(p_i - 1)$. Assume k is chosen such that all the resulting q_i are all prime. Write $q = q_1 \cdots q_m$ for the outputted Carmichael number. I now present the most interesting result to be used in this paper, which although simple turns out to be remarkably useful.

Lemma 3. *[GMP19]. Fix the notation above. Then for all primes s that divide kL, we have $2q + 1 = 2n + 1 \bmod s$.*

Proof. We know that $q_i = M(p_i - 1) + 1 = (1 + kL)(p_i - 1) + 1$. Thus, if s is prime with $s \bmod kL$ we must have $q_i = p_i \bmod s$. The result follows. □

Lemma 3 shows that we can the only information we need to determine whether $2q+1$ will be divisible by each of the primes s or not is the factorisation of the small starting Carmichael number n . Since we require $2q+1$ to be prime, we can discard any n for which $2n+1 = 0 \bmod s$ for any of the primes s dividing L or k. We can be fairly confident that there are many such primes s, since L is usually has a large number of prime factors, coming about as the least common multiple of the $p_i - 1$. We can be even more sure of this when the Erdös method is used to construct n as we will see in what follows.

The prime 3 willhave a large impact as part of the process of sieving used for the method of Granville and Pomerance. As mentioned in the previous sections, contributes a factor 2/3 to the product term $\prod_{i=1}^{h}\left(1-\frac{1}{s_i}\right)$ in the computation of the value of σ. This is sufficient reason that we would want to preserve $s_i = 3$ as a factor of kL in the construction. However, from Lemma 3 we know that if we want $2q+1$ to be prime, we must have $2n+1 \neq 0 \bmod 3$. This implies that either $n = 0 \bmod 3$ or $n = 2 \bmod 3$. In turn, we now consider each of these two cases:

1. Suppose $n = 0 \bmod 3$:

   Since 3 divides n we can set $p_1 = 3$. In our approach, we obtain the initial $n = p_1 \cdots p_m$ using the Erdös method, in which case $p_1 = 3$ belongs to the set $P(L*)$ (where $L*$ denotes the input number used in the Erdös method; $L*$ may be different from $L = lcm(p_i - 1)$ in the method of Granville and Pomerance, even though they may turn out to be equal). From the definition $P(L*)$, we deduce that $3 \nmid L*$.

   We know that if $p \in P(L*)$ then $p = f + 1$ for some $f \in L*$. Thus $p = 2 \bmod 3$ for each $p \in P(L*)\backslash\{3\}$. We also know that $p = 3 \bmod 4$ by choice of $L*$, so we know that $p = 11 \bmod 12$ for every $p \in P(L*)\backslash\{3\}$. Thus, if n is generated by the Erdös method and 3 is a factor of n, then the remaining primes, aside from 3, that are factors of n must all be 11 mod 12.

2. Suppose now $n = 2 \bmod 3$:

   Claim: $p_i = 2 \bmod 3$ for all primes p_i such that p_i is a factor of n.

   Proof: Suppose, to the contrary, that $p_i = 1 \bmod 3$ for some i. This implies $3 \mid p_i - 1$. ByTheorem 5, we infer that $3 \mid n - 1$, and thus $n = 1 \bmod 3$. Since we assumed that $n = 2 \bmod 3$ we have a clear contradiction.
   Also, $n = \prod_{i=1}^{m} p_i = 2^m \bmod 3$, which implies that $n = 2 \bmod 3$ if and only if m is odd. As a result, m, must be odd in the case where $n = 2 \bmod 3$.

   So in the instance where $n = 2 \bmod 3$, is it necessary to use a starting Carmichael number with m odd in which $p_i = 2 \bmod 3$ for each prime factor p_i. Perhaps these conditions may seem too stringent. But, in the discussion of the Erdös method we have already shown how to execute these conditions: we simply need to ensure that

3 does not divide $L*$, where $L*$ is the value corresponding to the notation used in the Erdös construction. The consequence is that there is only one value of $p \in P(L*)$ (the exception being $p = 3$) that will meet this condition. For the final step in the Erdös method use the set $P(L*)\backslash\{3\}$.

5.3 A summary in the finite-field safe prime setting

The cumulation of the previous sections have shown that it is possible to produce a particular Carmichael number n so that if the method of Granville and Pomerance is used to produce q from starting value n, then $2q + 1 \neq 0 \bmod s_i$ for many small primes s_i. We know that q will attain the Monier-Rabin bound on $S(q)$, that is, q will attain its maximum number of Miller-Rabin non-witnesses for q, as desired. I now summarise this procedure as follows:

1. Carry out the first step of the Erdös method with a value of $L*$ such that $2 \mid L*$, $4 \nmid L*$, $3 \nmid L*$. We do this so that the set $3 \in P(L*)$, and other primes in $P(L)$ are all 11 mod 1.

2. Remove 3 from $P(L*)$ and carry out the second step of the Erdös method with some choice of odd m to find a subset of $P(L*)$ consisting of primes $p_1 \cdots p_m$ such that $n = p_1 \cdots p_m = 1 \bmod L$. From the previous discussion concerning the prime 3, we know that n will be a Carmichael number with m prime factors that are each 11 mod 12. Thus, they will also be both 3 mod 4 and 2 mod 3.

3. Set $L = lcm(p_i - 1)$. By Lemma 3 if n is a Carmichael number then $2n+1 \neq 0 \bmod s$ for each prime factor s of L. Check this is satisfied, but if this fails, go back to the first step and begin again to generate a different value of n.

4. Use n to denote the same value in the method of Granville of Pomerance to find candidates for q (such that all q_i are all prime). We know that 3 does not divide L in the Granville-Pomerance method, but we are seeking a value such that $3 \mid kL$ since sieving by 3 will be highly effective. Thus, we should set k to be some sufficiently large multiple of 3 in this step.

5. Lastly, we run the Miller-Rabin primality test on $2q + 1$. We can be sure that $2q + 1 \neq 0 \bmod 3$ and $2q + 1 \neq 0 \bmod s$ for each prime divisor s of L as a result of the choices made in the second step. Thus, $2q + 1$ will certainly not be divisible by 3 and other certain small primes. If $p = 2q + 1$ passes then we have found a Carmichael number q that passes the Miller-Rabin primality test with probability approximately 1/4 but for which solving the DLP in the subgroup of order $q \bmod p$ is relatively easy via the Pohlig-Hellman algorithm.

I fall short of providing an implementation for this strategy in the paper.

6 Elliptic-Curve Diffie Hellman

Definition 7. *Elliptic Curve. An elliptic curve E, over a finite-field $\mathbb{F}_p$ is defined by the following equation: $y^2 = x^3 + ax + b$ where $a, b \in \mathbb{F}_p$. The curve also needs to be non-singular (i.e it has no cusps, self-intersections or isolated points). A curve is non-singular if, and only if, the discriminant Δ is non-zero, where: $\Delta = -16(4a^3 + 27b^2)$*

Elliptic curve cryptography is an industrial standard cryptosystem. Its popularity is due to an increase in speed during implementation, the use of less memory, and smaller key sizes. For example, a key size of 4096 bits for RSA gives the same level of security as 313 bits in an elliptic curve system [Mus06]. Its security lies in the difficulty of solving the Elliptic Curve Discrete Log Problem (ECDLP), which will be described shortly. The ECDLP is known to be at least as hard as the DLP. If the elliptic curve is chosen carefully, the ECDLP is believed to be infeasible to solve, even with today's computational power. On the other hand, this obstacle has not deterred people in their attempts to crack elliptic curve cryptosystems. A multitude of attacks have been developed, tested, and analyzed when attacking the ECDLP. For the most, part the ECDLP has withstood all attempts; however, in some special cases the problem is actually quite easy. It is these simple cases that must be avoided when building such a cryptosystem. One example of such a weakness will be the subject of this section.

An Elliptic Curve Diffie-Hellman (ECDH) scheme can operate in different ways, but in this section we will look at the method in which an explicit definition of an elliptic curve is given. We will denote an ECDH parameter set by (p, E, P, q, h), where p is the order of the finite-field $\mathbb{F}_p$, E is the equation of the chosen elliptic curve, P is the generator for a subgroup of order q on the curve (so that $\#E(\mathbb{F}_p = \langle P \rangle)$ and h is the cofactor of this subgroup. The "cofactor" is $h = \frac{1}{q}\left|E\left(\mathbb{F}_p\right)\right|$. The significance of the cofactor is that for every point r on the curve the point hr is either the "point at infinity", or it has order p. This means that if we take a point on the curve and multiply it by the cofactor then we necessarily return a point in the subgroup of prime order p.

Algorithm 4. [Rab05]. Elliptic Curve Diffie-Hellman.

$$
\begin{array}{cc}
\mathcal{A}(p, E, P, q, h): & \mathcal{B}(p, E, P, q, h): \\
d_A \xleftarrow{\$} [1, q-1] & d_B \xleftarrow{\$} [1, q-1] \\
Q_A \leftarrow d_A \cdot P & Q_B \leftarrow d_B \cdot P \\
& \xrightarrow{Q_A} \\
& \xleftarrow{Q_B} \\
(x_A, y_A) \leftarrow hd_A \cdot Q_B & (x_B, y_B) \leftarrow hd_B \cdot Q_A \\
\text{return } x_A & \text{return } x_B
\end{array}
$$

If $h > 1$, we know that there are points on E with small order, we'll call such a point H (there won't always be a point of order h, however to make this argument slightly simpler, we'll assume there is). We can show correctness as follows:

$$\begin{aligned}(x_A, y_A) &= d_A \cdot Q_B \\ &= hd_A \cdot (d_B \cdot P + H) \\ &= hd_A \cdot d_B \cdot P + hd_A \cdot H \\ &= hd_A \cdot d_B \cdot P \\ &= hd_B \cdot d_A \cdot P \\ &= hd_B \cdot (d_A \cdot P + H) \\ &= hd_B \cdot Q_A \\ &= (x_B, y_B)\end{aligned}$$

Where the fourth equality follows from the fact that the point H is of order h.

In practise, the cofactor $h = 1$ is often used. One might ask why we should parametrise the cofactor at all, as the correctness result remains the same regardless. The reason is that it helps avoid a small potential data leakage if Bob does not correctly implement the protocol. Then, suppose an attacker didn't give Alice the value $d_A \cdot P$, he gave her the value $d_A \cdot P + H$. Alice would then compute $d_A(d_B P + H) = d_A d_B P + d_A H = d_A d_B P + (d_A \bmod h)H$. As Bob knows the value $d_A d_B P$ (he does his half of the second phase honestly), he can compute Alice's shared secret with h different alternatives, and so recover $d_A \bmod h$.

By internally multiplying by h in the second phase, this doesn't happen; we have seen that correctness remains independent of what $d_A \bmod h$ is.

Now, if this is the only place that Alice uses d_A (that is, if she is using ECDH correctly), this leakage doesn't matter at all - Bob learns the shared secret as required, and since $d_A \bmod h$ is not used by Alice elsewhere he can't leverage that into any knowledge of any other secret. However, if Alice reuses her $(d_A, d_A \cdot P)$ pair for other exchanges, the leakage is relevant.

Alice and Bob exchange their public keys Q_A and Q_B, as they do not share the randomly generated values d_A and d_B so a potential attacker only has access to these values and the values of the parameters. Thus, the following problem must be solved in order to determine the shared key.

Definition 8. *Elliptic Curve Discrete Logarithm Problem (ECDLP). Given points $P, Q \in E(\mathbb{F}_p)$ to find an integer l, if it exists, such that $Q = lP$.*

Given the ECDLP $K_A = aK_B$, the finite-field Pohlig-Hellman algorithm can be interpreted in the elliptic context, and similarly used recursively by computing discrete logarithms in the prime order subgroups. Each of these smaller subproblems can then be

solved using methods, such the Pollard's rho algorithm, but these subtools will not be discussed in this paper.

The Pohlig-Hellman algorithm, as given in [Mus06], to solve the ECDLP works as follows:

Algorithm 5. [Mus06]. The Pohlig-Hellman algorithm for ECDLP.
Input. An elliptic curve E of order n with prime factorisation $n = \prod_{i=1}^{r} p_i^{e_i}$, and two elements $Q, P \in E$ such that P has order n and $Q \in \langle P \rangle$. Notice that n has r distinct prime factors.
Output. The unique integer $l \in \{0, \ldots, n-1\}$ so that $Q = lP$

1. For each $i \in \{1, \cdots, r\}$:

   i Compute $l_i = l \bmod p_i^{e_i}$

2. Solve the simultaneous congruence $l = \quad (\mathrm{mod} p_i^{e_i}) \quad \forall i \in \{1, \ldots, r\}$ for l using the Extended Euclidean Algorithm. The Chinese Remainder Theorem guarantees there exists a unique solution $l \in \{0, \ldots, n-1\}$

3. Return l

Let's take a closer look at how this works. For the moment fix a prime say $p_1^{\ell_1}$ We compute l_1 as follows. We write the base-p_1 representation of l_1

$$l_1 = a_0 + a_1 p_1 + a_2 p_1^2 + \ldots + a_{e_1-1} p_1^{\ell_1 - 1} \mod p_1^{\ell_1}, \mathrm{for} a_i \in [0, p_1 - 1] \tag{6}$$

We begin by computing a list of small values for each prime divisor p_i of n. Set $T_i = \left\{ [j] \left(\left[\frac{n}{p_i} \right] P \right) : 0 \leq j \leq p_i - 1 \right\}$. We will look for a match with these points and values that we will determine below. When we find a match we have solved for a given coefficient in the base- p_1 expansion of l. We can now compute the following,

$$\begin{aligned} \left[\frac{n}{p_1} \right] Q &= \left[\frac{n}{p_1} \right] \left(\left[a_0 + a_1 p_1 + \ldots + a_{e_1-1} p_1^{e_1 - 1} \right] P \right) \\ &= [a_0] \left[\frac{n}{p_1} \right] P + \left([a_1 + a_2 p_1 + \ldots] \right) [n] P = [a_0] \left[\frac{n}{p_1} \right] P \end{aligned}$$

Thus we can now look in our list T_1, find the matching point in the list and read off the coefficient a_0.

To solve for the next coefficient, a_1, we have to change our starting point which can be easily done. since we have already solved for a_0 we can use it and set $Q_1 = Q - [a_0] P$ then perform the above calculation using Q_1 instead, and shifting by the proper quantity

to isolate for a_1. If we multiply (6) by $\frac{n}{p_1^2}$, after a_0 has been removed this will then give us

$$\left[\frac{n}{p_1^2}\right] Q_1 = ([a_1 + a_2 p_1 + \ldots]) \left[\frac{n}{p_1}\right] P = [a_1] \left(\left[\frac{n}{p_1}\right] P \right)$$

and again we look in our list T_1 for a matching solution. This then gives us a result for a_1. We continue in this way until we have solved for each coefficient in the base- p_1 expansion of l_1. We then continue and solve for each l_i in the same manner. When this is done we solve the system of congruences in step 2 of Algorithm 5 and recover the original value of l in our original problem $Q = [k]P$, thus solving the ECDLP.

Just as an attacker can seek to find malicious parameter sets in the context of finite-field DH, the same approach is possible in the ECDH context. In the following section, we will describe and explain a sketch of one possible attack. Firstly, we will show that the algorithm of Bröker and Stevenhagen can be used to construct a composite number q that would be declared 'probably prime' by the Miller-Rabin Primality Test, but for which the ECDLP is relatively easy to solve via Algorithm 5. Then we will use this knowledge and construct a curve of order $n = hq$ and of suitable size.

6.1 The algorithm of Bröker and Stevenhagen

In this section we will describe how we can use the algorithm of Bröker and Stevenhagen to determine a curve over a prime p of some particular order n. We will begin by stating certain theorems that the algorithm exploits, which are all quoted from [BS05] via [GMP19].

Theorem 9. An elliptic curve E over $\mathbb{F}_p$ has $\#E(\mathbb{F}_p) = p + 1 - t$ points for some t where $\mid t \mid < 2\sqrt{p}$.

Theorem 10. The endomorphism ring of E contains $\mathbb{Z}[\sqrt{t^2 - 4p}]$, which is a subring of the imaginary quadratic field $K = \mathbb{Q}(\sqrt{t^2 - 4p})$.

Theorem 11. If E is an elliptic curve over a number field whose endomorphism ring is the ring of integers of K, then the reduction modulo p of E is an elliptic curve over $\mathbb{F}_p$ and, by taking a suitable isomorphism, we may ensure that the reduced curve has $p+1-t$ points.

The strategy of the algorithm is roughly to determine some elliptic curve whose endomorphism ring is $K = \mathbb{Q}(\sqrt{t^2 - 4p})$, then find its reduction modulo p according to Theorem 11. The algorithm is fairly complicated and I will not seek to give a full exposition here, but instead I will describe the first part of the algorithm.

Suppose we begin with an integer n chosen to be the number of points on the final desired elliptic curve. According to the Algorithm, we would first like to construct a prime p and an integer t such that $p + 1 - t = n$ and such that $\mathbb{Q}(\sqrt{t^2 - 4p})$ has some

small discriminant D (a result of Theorem 10 and Theorem 11 - we will omit the proof of this). In what follows, the algorithm will attempt to solve this problem. Recall that that algorithm takes in an integer n and outputs an elliptic curve consisting of n points.

Let $D < 0$ be some discriminant of some imaginary quadratic field F. We will try to find point(s) (p, t) such that $t^2 - 4p = (f^2)D$ for some $f \in \mathbb{N}$.

By Theorem 9 we know that we must ensure $p + 1 - t = n$ and so $p = n + t - 1$. If $t^2 - 4p = (f^2)D$ then

$(t - 2) - (f^2)D = t^2 - (f^2)D - 4t + 4 = 4(p - t + 1) = 4n$.

This result allows us to conduct the first part of algorithm of Bröker and Stevenhagen used to construct a curve with $n = h_1 \cdots h_k$ points:

1. Choose some D such that
   - $(D/h_i) \geq 0$ for all $h_i \mid n$.
   - $D < 0$ so that there are only finitely many solutions
   - either $D = 0 \bmod 4$ or $D = 1 \bmod 4$ (since the algorithm requires that $gcd(1, D) = 1$ and $gcd(D, 4n) = 1$),
   - $\mid D \mid < D_{\text{bound}}$ for some bound
2. Determine all solutions $x_0 \in \mathbb{Z}/4n\mathbb{Z}$ to the equation ${x_0}^2 = D \bmod 4n$
3. For each solution, setting $x_0 = w$ use Cornacchia's algorithm [BS05] to solve the equation $w^2 - f^2 D = 4n$ for w
4. Check whether $n + (w + 2) - 1 = n + w + 1$ is a probable prime.
5. If so, output (p, t) where $t^2 - 4p = (f^2)D$
6. Repeat 1 to 4 until a value for step 5 is obtained

The rest of the algorithm is more complex and requires an extended set of mathematical tools. Roughly speaking, we use the chosen discriminant D in conjunction with a tool known as Hilbert class polynomial whose roots can be used to return an elliptic curve with $n = p + 1 - t$. To find the whole algorithm and its proof, refer to [BS05].

One may sensibly speculate that the algorithm will completely fail for a given value of n since we restrict to $\mid D \mid < D_{\text{bound}}$. But for the purposes of ECDH this should not be a problem since there are many possible values of n we can work with.

Now let us consider the outcomes for a few different values of n. $n = 4q$ for the prime order q is often used in practise (eg. the group order of Edwards and Montgomery curves

is divisible by 4). If D is odd then any solution (w, f) to $w^2 - f^2 D = 4n$ means that w will be odd, and thus t will be odd.

If n is odd then $p = n + w + 1$ will be odd, as desired. However, if n is even and D is chosen to be odd, p will be even and so obviously not prime. Thus it is necessary to insist on an odd discriminant D in the case where n is odd. This restriction is unnecessary in the case where n is even (since w and t will be even and so $p = n + w + 1$ will be odd), so we can also take D to be even.

6.2 A summary in the elliptic curve setting

1. As a malicious developer, use the methods developed in Section 4 to generate a sufficiently large Carmichael number q which is congruent to 3 mod 4. By design, this will pass the Miller-Rabin primality test with a relatively high success probability.

2. Apply the algorithm of Bröker and Stevenhagen, or otherwise, to obtain an elliptic curve E over $\mathbb{F}_p$ of desired order n (eg $n = 4q$ for applications that use Montgomery and Edwards curves) such that n is a product of many small primes. Use the parameter set (p, E, P, q, h) (where h is the cofactor of E and P is a generator for E) in the ECDH implementation that you wish to eventually attack.

3. Once an implementation has been established and a pair of users have exchanged a secret key x, use the Pohlig-Hellman algorithm, or otherwise, to solve (with notation as in Algorithm 4) the ECDLP $Q_A = d_A \cdot P$ for the curve E with generator P. Then $(x_A, y_A) = (x_B, y_B) = hd_A \cdot Q_B$, and the shared secret is $x_A = y_A = x$.

6.3 An example in the elliptic curve setting

[GMP19] implemented the algorithm of Bröker and Stevenhagen [BS05], and ran it with q that are 256 -bit Carmichael numbers with 3 and 4 prime factors, all congruent to 3 mod 4. These were generated using methods described in Section 4. By design, these values of q pass random-base Miller-Rabin primality testing with probability 1/4 and 1/8 per iteration, respectively. Using an early abort approach for each q they estimate a success probability of roughly 1/4 for each q considered. When successful, the computations took under a minute on a personal device.

Set $q = q_1 q_2 q_3$ where:

$$q_1 = 120969320416809549586937 71$$
$$q_2 = 362907961250428648760813 11$$
$$q_3 = 133066252458490504545631471$$

They found q to be a Carmichael number with 3 prime factors that are all congruent to 3 mod 4, so that q passes random-base Miller-Rabin primality testing with probability 1/4 per iteration. Using the algorithm of Bröker and Stevenhagen, they obtained the elliptic

curve $E(\mathbb{F}_p)$ defined by $y^2 = x^3+5$, where = 58417055476151343628013443570006259007184622249465663594746403634665595 3 such that $\#E(\mathbb{F}_p) = q$ and p has 256 bits. They found that every point P on this curve satisfies $[q]P = \mathcal{O}$, the point at infinity, so any point can be used as a generator (although it is possible that not all such points will have order q, if q is accepted by the MR test as being prime then this will not hinder anything). The Pohlig-Hellman algorithm can be used to solve the ECDLP on this curve using about $3 \cdot 2^{42.5}$ group operations, since the largest prime factor. of q has 85 bits.

7 Conclusions

The discussions of this essay were intended to show that the use of poorly established parameter sets in DH and ECDH can lead to malicious attacks. We should instead use well vetted and tested parameter sets. This message is not so important in the case of ECDH, not necessarily as a result of security concerns of the difficulty of parameter validation, but simply because parameter generation in this case is non-trivial to begin with. We can be more sure of secure implementation if we stick to a limited set of well-understood curves. Even then, [Ber+15] demonstrate that it is very difficult to rule out that even well-established curves don't have hidden security flaws unless the generation process for the curve is thoroughly explained, with demonstrably little room for manipulation. If a library insists, for some reason, on allowing the use of bespoke parameters, then implementations should employ robust primality testing as part of parameter validation, using, for example, many rounds of Miller-Rabin tests, or using the Baillie-PSW primality test for which there are no known pseudoprimes, cf. [GMP19].

The counterargument to this is that the potential reward of solving many logarithms if an attacker uses the best-known algorithms for solving discrete logarithms, outweighs the cost of large-precomputations that these algorithms require. Despite the fact that even our best algorithms are very slow, the reward outweighs the cost when there are far more use cases to be exploited with common sets of DH parameters. The Logjam attack on 512-bit DH described by [Adr+15] is an example of a common set of DH parameters where this argument certainly applied.

The work of this paper puts greater weight in favour of the first argument, suggesting that we should stick to limited sets of well-understood DH parameters in both the finite-field and elliptic curve cases.

References

[Alb+18] Martin R. Albrecht, Jake Massimo, et al. "Prime and prejudice: Primality testing under adversarial conditions". In: *Proceedings of the ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2018, pp. 281–298.

[DDH76] Whitfield Diffie, Whitfield Diffie, and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[LL97] Chae Hoon Lim and Pil Joong Lee. "A key recovery attack on discrete log-based schemes using a prime order subgroup". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 1294. Springer Verlag, 1997, pp. 249–263.

[Sch91] C. P. Schnorr. "Efficient signature generation by smart cards". In: *Journal of Cryptology* 4.3 (1991), pp. 161–174.

[PH78] Stephen C. Pohlig and Martin E. Hellman. "An Improved Algorithm for Computing Logarithms over GF(p) and Its Cryptographic Significance". In: *IEEE Transactions on Information Theory* 24.1 (1978), pp. 106–110.

[Mus06] Matthew Musson. *Attacking the Elliptic Curve Discrete Logarithm Problem*. 2006.

[GMP19] Steven Galbraith, Jake Massimo, and Kenneth G. Paterson. "Safety in Numbers: On the Need for Robust Diffie-Hellman Parameter Validation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11443 LNCS. Springer Verlag, 2019, pp. 379–407.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". In: *Annals of Mathematics* 160.2 (2004), pp. 781–793.

[Rab80] Michael 0 Rabin. *Probabilistic Algorithm for Testing Primality*. Tech. rep. 1980, pp. 128–138.

[Mon80] Louis Monier. "Evaluation and comparison of two efficient probabilistic primality testing algorithms". In: *Theoretical Computer Science* 12.1 (1980), pp. 97–108.

[Pin06] Richard G. E. Pinch. "The Carmichael numbers up to 10^{18}". In: (2006). arXiv: `0604376 [math]`.

[Erd56] Paul Erdos. "On pseudoprimes and Carmichael numbers". In: *Publ. Math Debrecen* 4 (1956), pp. 201–206.

[GP01] Andrew Granville and Carl Pomerance. *TWO CONTRADICTORY CONJECTURES CONCERNING CARMICHAEL NUMBERS*. Tech. rep. 2001.

[JPV00] Marc Joye, Pascal Paillier, and Serge Vaudenay. "Efficient Generation of Prime Numbers". In: Springer, Berlin, Heidelberg, 2000, pp. 340–354.

[Ble05] Daniel Bleichenbacher. “Breaking a cryptographic protocol with pseudoprimes”. In: *Lecture Notes in Computer Science*. Vol. 3386. Springer, Berlin, Heidelberg, 2005, pp. 9–15.

[VS13] Joachim Von Zur Gathen and Igor E. Shparlinski. “Generating safe primes”. In: *Journal of Mathematical Cryptology* 7.4 (2013), pp. 333–365.

[Rab05] Kefah Rabah. “Implementation of Elliptic Curve Diffie-Hellman and EC Encryption Schemes”. In: *Information Technology Journal* 4.2 (2005), pp. 132–139.

[BS05] Reinier Broker and Peter Stevenhagen. “Constructing elliptic curves in almost polynomial time”. In: (2005). arXiv: `0511729 [math]`.

[Ber+15] Daniel J. Bernstein, Tung Chou, et al. “How to manipulate curve standards: A white paper for the black hat”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9497. Springer Verlag, 2015, pp. 109–139.

[Adr+15] David Adrian, Karthikeyan Bhargavan, et al. “Imperfect forward secrecy: How diffie-hellman fails in practice”. In: *Proceedings of the ACM Conference on Computer and Communications Security*. Vol. 2015-Octob. New York, New York, USA: Association for Computing Machinery, 2015, pp. 5–17.