Report Template
for The Security of Deffie-Hellman Algorithm

Andy Lum, Zach Coomer, and Jonathan Bess
The University of Tennessee at Martin
andlum@ut.utm.edu, zaccoom@ut.utm.edu, jondbess@ut.utm.edu

1 Abstract

The purpose of the Diffie-Hellman protocol is to enable two users to exchange a secret key securely that can then be used for subsequent encryption of messages. The protocol itself is limited to exchange of the key. But because of having no entity authentication mechanism, Diffie-Hellman protocol is easily attacked by the man-in-the-middle attack and impersonation attack in practice of today's society. The security of Diffie-Hellman relies on the difficulty of the discrete logarithm problem. Thus, in this project, we will describe the different steps necessary to solve the discrete logarithm problem (DLP), discuss the state-of-the-art results obtained for the solution of DLP, present the SUN NFS cryptosystem and discuss some of its deficiencies, and provide recommendations in order to have a secure size for the prime p used in the Diffie-Hellman algorithm.

2 Introduction

In this project, basic knowledge of number theory and abstract algebra is assumed. Many cryptosystems use public key cryptography and key exchange protocols for security. These modes of security rely on functions that are easy to compute but hard to invert or reverse. The seminal paper of Diffie and Hellman proposed the use of the discrete logarithm problem (DLP) or index as a good source for a "one-way function." One-way functions will be discussed in more detail and what they mean with DLP further in the discussion of this report. However, the focus will be regarding the different steps necessary to solve the discrete logarithm problem (DLP) and the state-of-the-art results obtained for the solution of DLP. Finding the solution of the DLP is possible, but no one has been able to find the solution in relatively short matter of time for a prime p that is large enough to be sufficient for security purposes. The process is very complicated leading to state-of-the-art results. The subject of key exchange was one of the first issues addressed by a cryptographic protocol. Securing data is more important than ever. As the internet has become more pervasive, security

attacks have grown. Today, software to secure data files is not in wide use because security software is not convenient to use. Thus, in this project, we will describe how a SUN NFS cryptosystem uses directories to organize data and programs into groups, allowing the management of several directories and files at one time. A Network File System is developed by SUN Microsystem for use on its UNIX-based workstations and allows users to access files and directories located on remote computers. In this project we will also tackle the subject of prime numbers and how big they should be to keep the Diffie-Hellman secure.

3 Discussion

(1) To begin, as the reader you should know what the discrete logarithm problem (DLP) or index is, it has a few names. As quoted from Kevin S. McCurley, who wrote on the discrete logarithm problem in 1990 regarding cryptographic purposes, the DLP is:

> "Let G be a group (written multiplicatively), and for g $\in G$, let $\langle g \rangle$ be the cyclic subgroup generated by g. The discreet logarithm problem for the group G may be stated as:
>
> Given g $\in G$ and a$\in\langle g \rangle$, find an integer such that $g^x = a$.
>
> Such an integer x is the discrete logarithm of a to the base g , and we shall use the notation $x = \text{ind}_g\ a$ (another word for discrete logarithm is index). Note that $\text{ind}_g\ a$ is only determined modulo the order of g."

Now if that does not make much sense, that is okay. We are going to discuss DLP more in terms of the Diffie-Hellman key exchange protocol or algorithm. The Diffie-Hellman algorithm makes use of the discrete logarithm problem for its security. The purpose of the algorithm is to allow users to exchange a secret key securely that can then be used to encrypt communications. The DLP is determined by a prime and a primitive root modulo that prime which generates a multiplicative group, using those determine a unique exponent which is the DLP solution. The DLP difficulty is determined by using large prime numbers. When using smaller prime numbers it is considered relatively simple to solve DLP through brute-force but when using a prime number to the magnitude of thousands of bits, it becomes practically intractable to solve. . This creates what we mentioned in the introduction, a one-way function. Think about a cookie. If I gave you flour, sugar, butter, eggs, and chocolate you could probably make a cookie with ease. However, If I gave you a cookie and asked you to tell me how much flour, sugar, butter, eggs, and chocolate was in that cookie then that would be quite a challenge.

That is the premise of one-way functions. Easy to create in one direction, but close to impossible to work in the reverse or inverted direction.

To expand more on DLP through the Diffie-Hellman key exchange I need to explain how the key exchange works. A common representation includes Alice and Bob. We have a group g which is known. Alice has a secret integer a and transmits g^a. Bob has a secret integer b and transmits g^b. Alice and Bob need to establish a secret number to encrypt their communications so by raising each other's transmitted group elements by the secret integers that they have, they reach the same secret number. This completes the key exchange and now they have an encrypted line of communication. An attacker can decipher their encryption by solving the DLP, which would likely take an unreasonable amount of time since the hacker does not know Alice's or Bob's secret number.

Here is an example of solving a simple DLP in terms of the Diffie-Hellman key exchange from the textbook *Computer Security Principles and Practice*:

> "Key exchange is based on the use of the prime number q = 353 and a primitive root of 353, in this case $\alpha = 3$. A and B select secret keys $X^A = 97$ and $X^B = 233$, respectively. Each computes its public key:
>
> A computes $Y_A = 3^{97} \bmod 353 = 40$.
>
> B computes $Y_B = 3^{233} \bmod 353 = 248$.
>
> After they exchange public keys, each can compute the common secret key:
>
> A computes $K = (Y_B)^{XA} \bmod 353 = 248^{97} \bmod 353 = 160$
>
> B computes $K = (Y_A)^{XB} \bmod 353 = 40^{233} \bmod 353 = 160$
>
> We assume an attacker would have available the following information:
>
> q = 353; $\alpha = 3$; $Y_A = 40$; $Y_B = 248$
>
> In this simple example, it would be possible by brute-force to determine the secret key 160. In particular, an attacker E can determine the common key by discovering a solution to the equation $3^a \bmod 353 = 40$ or the equation $3^b \bmod 353 = 248$. The brute force approach is to calculate powers of 3 modulo 353, stopping when the result equals either 40 or 248. The desired answer is reached with the exponent value of 97, which provided $3^{97} \bmod 353 = 40$."

Clearly the prime picked in the preceding scenario was relatively small. So one could imagine the amount of work involved to compute the solution to a DLP with a prime of the magnitude of thousands of bits.

There are a series of steps and unique algorithms that can solve the DLP, none may be practical in terms of real world use mainly due to the time required to commit such a task. These methods or algorithms take vast number of calculations to complete and hours upon hours because of the shear size of the prime used to calculate the DLP. The algorithms I will briefly explain are the Shank's BSGS(Baby Steps Giant Steps) also known as Meet in the Middle and The Pohlig-Hellman algorithm. Also, I must mention that while it is probably the most infeasible approach, it is possible that one could find the solution to DLP through brute force attempts. If one took tables of multiples of the prime and given numbers and entered them one by one, this approach would quite possibly take an unfathomable amout of time.

Shank's Baby Steps Giant Steps:

We say if $a^m \equiv b \bmod N$, then m is the discrete logarithm of b, mod N.

Come up with two sets of numbers and look for a number that appears in both sets

1. Choose some k
2. Evaluate a, a^2, a^3, . . . , a^{k-1}.
3. Evaluate ba^{-k}, ba^{-2k}, ba^{-3k}, ... ba^{-rk}, where rk > N.
4. If the DLP has a solution, there will be (at least) one value common to both lists.

Let the common elements be a^n and ba^{-mk}. We have:

$a^n \equiv ba^{-mk} \bmod N$

$a^{mk+n} \equiv b \bmod N$

which solves the DLP.

Pohlig-Hellman:

This requires the use of the Euler-Fermat Theorem which is:

If (a,N) = 1, then $a^d \equiv 1 \bmod N$ for some divisor d of $\varphi(N)$.

And the Chinese remainder theorem:

Let m and n be relatively prime positive integers. For all integers a and b, the pair of congruences

$x \equiv a \bmod m$, $x \equiv b \bmod n$

has a solution, and this solution is uniquely determined modulo mn.

So,

$a^{\varphi(N)} \equiv 1 \bmod N$

Let $\varphi(N) = pq$, where $(p, q) = 1$

Doing this allows us to solve the DLP by solving a Chinese Remainder problem

The Pohlig-Hellman for solving $a^x \equiv b \bmod N$ follows:

Let $x = a_0 + a_1 p$

Then:

$a^{qx} \equiv b^q$

$(a^q)^{a_0} \equiv b^q$

Since b^q and a^q can be found, a_0 can be determined by trial and error.

So instead of guessing for x which would be between 0 and N – 1, we can try for a_0 which possibilities are between 0 and p.

Since $x = a_0 + a_1 p$, then $x \equiv a_0 \bmod p$

Similarly $x \equiv b_0 \bmod q$ can be found, then use the Chinese remainder to solve for x.

Which solves the DLP.

Those are just a couple of potential ways to solve DLP, there are quite a few combinations of other types of algorithms and no set in stone way to approach the task. The most important goal in tackling the discrete logarithm problem is to find a method in which the least amount of time is necessary to solve the problem.. During my investigation I came across many methods and algorithms that have been developed over a few decades at this point. The preceding algorithm descriptions I have recorded are extremely brief representations to an extremely complex theory and problem. I have recorded those summaries from Mathematician Jeff Suzuki and I recommend looking him up on YouTube for a much more in depth representation.

(2) With everything I have mentioned about the discrete logarithm problem including what it is and the steps necessary to solve it, which as previously stated are not so set in stone regarding the implication of DLP in cryptographical uses, I want to briefly describe the state-of-the-art solution that goes with DLP. Remember, a one-way function is a function that is easily determined in one direction, but

practically unfeasible to invert. The solution to DLP is when one is able to bypass the constraints of practically unfeasible and actually produce a solution to a one way function. For this, I believe that the best way to represent the true magnificence of the solution to DLP and it's complexity in a cryptographical scenario is by discussing the results of Damian Weber and Thomas Denny in The Solution of McCurley's Discrete Log Challenge from 1998.

In this report Weber and Denny are solving a discrete logarithm problem in the form of a Diffie-Hellman key which was presented as a challenge by Kevin S. McCurley, who wrote the Discrete Logarithm Problem in 1989. McCurley offered a 100-dollar prize to the first person to solve this DLP problem and just to give you, the reader some context on how intensive of a problem this was: the problem was presented in 1989 and the solution was written in a report in 1998. Almost a decade to solve.

McCurley's challenge was a DLP used in the Diffie-Hellman key exchange protocol and used a 129-digit prime. Think about the number of bits that would require and that what is being referred to is from the late 1980s and late 1990s, 20 to 30 years ago. If 0 through 7 in base 10 can be represented via 3 bits, and 8 and 9 in base 10 can be represented via 4 bits, then on average it take 3.2 bits to represent a single digit. In this particular case, the 129-digit prime is 426 bits. The specific numbers are available online but I want to reference that McCurley provided the number that Alice would send raised to her secret key, the number Bob would send raised to his secret key, p which is the prime and q which has a correspondence to the multiplicative group. Using this information, which would typically be the information that is easily recovered in the one-way function, they produced a solution. They used a combination of algorithms and mathematical techniques to reduce and ultimately find the solution, some of the methods they mention are the Number Field Sieve, Index Calculus, and Lanczos algorithm. Note, that these algorithms are not fundamental methodologies and those are just some of the algorithms mentioned there were others. Towards the end of the report, Weber and Denny have a table listing complete computation times and with preprocessing algorithms they were able to compute in 911 hours. The solution K that they produced is K = 3812728041119001413807839150792963419399864355101867028505637561504552396692940392210217251405327092887263942637006353279774080 8. I just wanted to include that to show the shear scale of the solutions being referred to. The chance that one is able to brute-force enter a number of that scale is highly unlikely. Today's solutions use primes of thousands of bits and even more complex groups.

The solutions to these problems are truly state-of-the-art and it is quite clear why these computations are used in cryptography.

(3) A file system is a hierarchical structure (file tree) of files and directories. With this file tree, it uses directories to organize data and programs into small groups and allows the management of several directories and files at one time. Some tasks are performed more efficiently on a file system than on each directory within the file system. A Network File System is a distributed file system developed by SUN Micro-systems for use on its UNIX-based workstations. It allows users to access files and directories located on a remote computer, but data potentially stored on another machine. Its major goal is to provide a simple crash recovery and a reasonable performance.

Advantages:

- Stateless server and client
- Server can be rebooted and user on client might be unaware of the reboot
- Client/Server distinction occurs at the application/user level not the system level
- It is highly flexible, so we need to be disciplined in our administration/configuration.

Disadvantages:

- Uses RPC(Remote Control Call) authentication which is easily spoofed
- File System data is transmitted in clear text (data could be copied)
- Network slower than local disk
- Complexity, Security issues.

This requires three important parts:

- The protocol
- The server side
- The client side

The **protocol** uses the Sun RPC mechanism and Sun eXternal Data Representation(XDR) standard. It is defined as a set of remote procedures. Protocol is stateless and each procedure call contains all

the information necessary to complete the call. The server maintains no "between call" information.

The **server** side implements a write-through policy that is required by statelessness. In addition, any blocks modified by a write request (including i-nodes and indirect blocks) must be written back to disk before the call completes. File handle consists of: Filesystem id which identifies disk partition, the I-node number which identifies file within partition, and the generation number which changes every time i-node is reused to store a new file. The server will store Filesystem id which is in filesystem super-block and it will store the I-node generation number in i-node.

The **client** side provides transparent interface to NFS and a mapping between remote file names and remote file addresses is done a server boot time through remote mount. In addition, it provides transparent access to NFS and other file systems (including UNIX FFS). The new virtual filesystem interface supports VFS calls, which operate on whole file system and VNODE calls, which operate on individual files. However, it does treat all files the same fashion.

(4) It is recommended to have large prime numbers to keep your Diffie-Hellman secure. The recommended size is at least 2,048 bits. The larger the prime number the more secure it will be. There are ways to generate large prime numbers that are safe to use. Two of the ways this can be done as Thomas Pornin states:

> "A good DH modulus and generator is what you get when generating DSA key parameters; see the DSA specification. You get to choose the subgroup order (q, a prime number), the modulus (p, such that p-1 is a multiple of q), and a generator for the subgroup of size q....some effort has been invested into generating so called 'safe primes': prime integers p such that (p-1)/2 is also a prime. A safe prime is called such because it does not suffer from some attacks which may make discrete logarithm easy (or at least easier) on some 'weak' modulus – but a randomly generated modulus will not be weak, with an overwhelming probability, so there is no real worry here. Also, "safe primes" have the advantage of allowing g=2 as generator, which promotes computational efficiency."

Despite the need for large prime numbers, it is important that the prime number isn't too big due to it affecting the efficiency of the Diffie-Hellman process.

4 Results

(1) The investigation of the steps required to solve the DLP, make it very clear that there is no size fits all. The primes that are picked in the generation of the DLP can belong to additive or multiplicative groups, and there is no general equation to solve the discrete logarithm problem. Also, in today's cryptographic climate, if system users are using a large enough prime, it is unlikely that solving the DLP will be of any practical real world application. Most calculations take times in the exponential order corresponding to the size of the prime.

(2) The state-of-the-art results obtained for the solution of DLP are very extreme numbers taking hundreds if not thousands of hours to reach. Therefore, the solution is very sufficient in supplying a one-way function in the field of security and encryption between two parties on public networks.

(3) In conclusion, Network File System (NFS) succeeded because it was robust, reasonably efficient, and tuned to the needs of disk-less workstations. In addition, NFS was able to evolve and incorporate concepts such as close-to-open consistency which means that any changes made by you are flushed to the server on closing the file, and a cache re-validation occurs when you re-open it.

(4) Ways to make sure you have a secure prime number to use in the Diffie-Hellman algorithm is by using a good DH modulus and generator or generating a safe prime. Large prime numbers are always more secure than smaller ones but primes that are too big won't add more in terms of security. Primes that are too big will become a detriment.

5 Conclusions

In Conclusion, it is easy to see why DLP is used in cryptography and especially the Diffie-Hellman key exchange algorithm. The steps required to solve a complex iteration of DLP get exponentially more difficult based on the prime used. There are many ways to approach the problem, which can lead to ambiguity for many people who attempt to solve the DLP. The amazing part of solving DLP is how easy a basic level or low prime problem can be, requiring just a few steps. The state-of-the-art results to the solution strengthen the argument of why using DLP as a one-way function is so useful to cryptology. The solution take hours upon hours to reach for a DLP with a large prime with today's current mathematical knowledge. Yet it is quite simple to generate the problem and for two entities to share that problems solution as a means of encryption. These are just scratching the surface on why DLP is the perfect match for cryptography and why it is used

in the Diffie-Hellman key exchange protocol. For a NFS, we need a protocol, a client, and a server. Some recent models on the client side on the virtual file system are UNIX file system, other file system, and an NFS Client. If the file is local, then we access the UNIX file system for the local file. However, if the file is remote, then the NFS Client will send a request to NFS Server. The NFS client will use the NFS Protocol to communicate with the server. On the server side, the virtual file system contains an NFS server and a UNIX file system. A Virtual File System is used to distinguish between local and remote files. The file identifiers used in NFS are called file handles. There are three types of file handles, **Filesystem id** which identifies disk partition, the **I-node number** which identifies file within partition, and **the generation number** which changes every time i-node is reused to store a new file.

6 Recommendations

It is recommended that the reader take some time and look up more in-depth information about solving discrete logarithm problems. These problems can range from relatively simple to nearly impossible. There are many methodologies and algorithms that can be used in conjunction with one another to solve DLPs and many ways that have yet to be developed. It is possible with the nature of DLP and the state-of-the-art solutions involved with complex examples of DLP that they could be implemented in much more than just a cryptographic key exchange algorithm. I recommend that if the reader has a desire to learn more about the step to solve DLP and the state-of-the-art solution, that they should read the publications from Kevin S. McCurley and also from Damian Weber and Thomas Denny to start. They give an in-depth introduction to the subjects and some history to them too.

7 References

(1) Rajat Kateja, After Hours Academic. *"Sun's Network File System."* After Hours Academic, 18 Oct.2021, https://afterhoursacademic.medium.com/suns-network-file-system-b23b1b3eb7c0.

(2) Sandberg, R., Goldberg, D., Kleiman, S.L., Walsh, D., & Lyon, B. (1985). *"Design and implementation of the Sun network filesystem."* USENIX, pages 119-130.

(3) Sandberg, R.. *"The Sun Network Filesystem: Design, Implementation and Experience."* (2001).

(4) Buck, B. (2020). *Cryptography: What is the discrete logarithm problem?* Retrieved from Quora: https://www.quora.com/Cryptography-What-is-the-discrete-logarithm-problem

(5) Katz, J., Lindell, Y. (2007). *Introduction to Modern Cryptography.* CRC Press.

(6) McCurley, K. S. (1990). *The Discrete Logarithm Problem.*
Retrieved from mccurley: https://www.mccurley.org/papers/dlog.pdf

(7) Pomerance, C. (n.d.). *Discrete Logarithms.* Retrieved from
chrome-extension:
//efaidnbmnnnibpcajpcglclefindmkaj/https://math.dartmouth.edu/ carlp/dltalk09.pdf

(8) Stallings, W., Brown, L. (2018). *Computer Security Principles and Practice.* Hoboken, New Jersey: Pearson.

(9) Suzuki, J. (2015, February 26). *Solving the DLP: Baby Step/Giant Step Algorithm.* Retrieved from YouTube: https://www.youtube.com/watch?v=007MVsELvQw&t=36s

(10) Suzuki, J. (2015, March 5). *pohlig hellman.* Retrieved from YouTube: https://youtu.be/SmzUe_-e7oA

(11) Weber, D., Denny, T. (1998). *The Solution of McCurley's Discrete Log Challenge.* Advances in Cryptology – CRYPTO '98, 458-471.

(12) Pornin, Thomas (2011, July 13). *Where do I get prime numbers for Diffie-Hellman? Can I use them twice?* Retrieved from Security Stack Exchange: https://security.stackexchange.com/questions/5263/where-do-i-get-prime-numbers-for-diffie-hellman-can-i-use-them-twice

(13) Computerphile. (2017, December 20). *Diffie Hellman -the Mathematics bit-Computerphile.* Retrieved from YouTube: https://youtu.be/Yjrfm_oRO0w