A discourse on the (in)security of Diffie-Hellman

J.D. Allersma
Open Universiteit
Heerlen
jan.allersma@student.ou.nl

ABSTRACT

The Diffie-Hellman protocol is used for exchanging keys between two machines. However, using Diffie-Hellman can be insecure: an NFS algorithm can be used to break Diffie-Hellman and a Logjam attack can be to attack a session in which Diffie-Hellman is used. This paper outlines the impact of those vulnerabilities, based on two papers. One paper outlines the vulnerabilities on Diffie-Hellman, whereas the other paper provides a proposal to modifiy Diffie-Hellman with use of the Blowfish algorithm in such a way that it provides protection against Logjam attacks. At the end of this paper we give our view on the impact of the Logjam attack and NFS algorithm, based on those two papers.

Keywords

Diffie-Hellman, Logjam, Blowfish, Vulnerable, Impact

1. INTRODUCTION

Diffie-Hellman is a protocol to exchange keys for establishing a connection with encrypted traffic [2] [1]. A study from 2015 has shown that using Diffie-Hellman as key-exchange protocol can be dangerous, as it can possibly be broken with a Logjam attack or get compromised with an NFS algorithm. This does imply that there is an insecurity for a lot of HTTPS servers. IKE, SSH and Mailserver (SMTP and IMAPS) protocols using DH-768 or DH-1024 can be compromised when state-level resources are within reach. [2]

There are some measures that could taken to make breaking the Diffie-Hellman protocol more time-consuming and hopefully infeasible, such as using a prime number p that has the property as defined with equation (5). Another measure could be not to use commonly used prime numbers. Another measure that could be taken when using Diffie-Hellman to set up TLS, is to disable support for legacy export-grade Diffie-Hellman.[2]

Adrian et al.[1] propose using a modified version of Diffie-Hellman. Essentially, the algorithm that breaks Diffie-Hellman needs to find a certain prime number. The modified Diffie-Hellman protocol uses messages instead of prime numbers, making the algorithm useless, since there is no more prime number to be found.

This paper is a discourse. In this paper, the findings of Adrian et al. [2] and Adrian et al. [1] are examined. We focus on the impact of Logjam attacks and the NFS algorithm. More specifically, we focus on what prevention measures are available and how many servers are vulnerable to this attack.

First, we give a brief summary of both papers, describing how the Diffie-Hellman protocol, the NFS algorithm and Logjam attack work, what the impact of the Logjam attack is as of 2015 and how a Logjam attack can be prevented. Secondly, we give an analysis of both papers in which we provide our critical view on the impact on Diffie-Hellman's insecurities based on both papers. We make clear what both papers outline and what the difference is between both papers. Lastly, we draw our conclusions based on our analysis and give suggestions for further research about the impact on Logjam attacks.

2. ANALYSIS

2.1 Vulnerabilities

2.1.1 Diffie-Hellman explained

Adrian et al.[2] show that the Diffie-Hellman protocol has a flaw, which has serious impact on 8.4% of Alexa Top Million HTTPS websites. Those websites support the Diffie-Hellman protocol with 512-bit size. The flaw can be exploited to compromise encrypted data that is exchanged between machines that use DH-512 to exchange keys for HTTPS traffic and allow legacy export-grade Diffie-Hellman[1]. Key exchange is done as follows: two machines agree on a prime number p and a generator number g. The first machine has a secret number a and the second machine has a secret number b. The first machine sends number A, calculated by

$$g^a \mod p \tag{1}$$

to the second machine. The second machine sends number B, calculated by

$$g^b \mod p \tag{2}$$

to the first machine. Lastly, both machines calculate key K by resolving

$$g^{ab} \mod p \tag{3}$$

An attacker can compromise encrypted data by finding the shared secret, which can be found by finding the discrete log x from

$$y = g^x \mod p \tag{4}$$

[1]This allows to use a a group of ciphersuites that is weaker than the regular ciphersuites[2]

The log can be obtained using a number field sieve (NFS) algorithm. This algorithm does some precomputing for p. The NFS algorithm consists of four stages: polynomial selection, sieving, linear algebra and descent. The first three stages are part of the precomputation. Only the last stage is used to find an individual discrete log x. How long it takes for the algorithm to calculate the discrete log x, depends on the prime number p that is used. If p has the property

$$p - 1 = 2q \tag{5}$$

with q being some prime number, the time to find discrete log x takes so long that it is impractical to use the NFS algorithm. Therefore, Diffie-Hellman can be safe to use, as long as p has the above mentioned property.[2]

2.1.2 *Diffie-Hellman's Insecurities*

Finding the discrete log x can take a lot of time, so for a successful attack, possible discrete logs have to be precomputed first, which requires p as input. Normally, this takes a lot of time if every server using Diffie-Hellman would use a different p. However, most servers use the same prime numbers. Of all Alexa Top one million domains that support 512-bit prime numbers as p. A group of 82% of those domains use the same prime number. A second group of 10% uses the same number and the other 8% use a total of 463 distinct prime numbers. Therefore, precomputing with prime numbers of those two domain groups would cover 92% of all Alexa top one million domains that support DF-512.[2]

Adrian et al.[2] take it a step further to see if the NFS algorithm can be used to break Diffie-Hellman with longer key sizes (768 and 1024 bits) and how much resources it would take to compromise, for example, IKE, SSH, HTTPS and SMTP. They conclude that theoretically with the computing power academics have within reach, DF-768 could be broken in a reasonable amount of time. NSA has theoretically enough resources to break DF-1024. Table 1 shows the amount of servers that are vulnerable if an attacker has a certain computing power to break Diffie-Hellman. For example, there are 98.500 servers of the Alexa Top one Million domains that are vulnerable to an attack if an attacker has the resources to precompute one 1024-bit Oakley group[2]. The percentage 17.9% represents the amount of servers that are vulnerable compared to the amount of servers that have been examined. Furthermore, there has been made a difference between 'HTTPS Top 1M w/ Active Downgrade' and 'HTTPS Top 1M'. The difference Active Downgrade and regular HTTPS is that with Active Downgrade the Logjam attack can be used. This means three things:

- Instead of precomputing primes, discrete log x is being found nearly real-time.
- Legacy export-grade Diffie-Hellman is used.
- Obviously, the handshake for an HTTPS takes longer because discrete log x has to be computed in the meantime.

[2]An Oakley group is a set of prime numbers which can be used as prime number p in Diffie-Hellman [2]

However, Adrian et al. tried to use an Logjam attack with Active Downgrade and it took 70 seconds to find the secret g^{ab}. We will discuss this further in the *Impact* section

Furthermore, the impact on Mailservers is depicted in Table 2.

Based on these findings, the outside world responded: modern browsers as of 2015[3] accepted prime numbers p of 512-bit size (and Safari even a less bigger size). After those findings of Adrian et al, [2] have been reported, all those browsers (and OpenSSL) are at least expected that they accept at least 1024-bit sizes prime numbers. Akamai does not support for export ciphersuites and many TLS-developers plan to gracefully reject negotiations when p is lower than 2048-bits.[2]

2.2 Prevention

2.2.1 *The protocol*

Adrian et al. [1] offer a modified version of the Diffie-Hellman protocol, which is invulnerable to Logjam attacks. According to them, a Blowfish algorithm can be used in combination with Diffie-Hellman to prevent a successful Logjam attack. Even though there is a stronger algorithm to use, the Rijndael algorithm, Blowfish is favored because of its speed. The Blowfish algorithm works as follows: The Blowfish algorithm chops messages in blocks of 64 bits, a block is called y. Messages can have a size that is not a multiple of eight bits. In that case, padding will be used to give messages an appropriate length for generating blocks. Then P will be created, a collection of 18 32-bit-sized blocks. And four S blocks will be created, each holding 256 blocks. S will be used for the *F function* [3], which will be used later on. Lastly K will be created, which is a collection of 32-bit-sized blocks from a 32-bits to 448-bits long sequence.[3] The first block of P (which we call *P1*) is XORed with the first block of K, resulting in *P'1*. Then the second block of P (*P2*) is XORed with the second block of K, resulting in *P'2* and so on. If there are no more blocks in K, but still blocks in P that needs to be processed, the first block of K will be used again, then the second block and so on. Then each y will be split in two 32-bit-sized blocks (yL and yR). Then four actions will done:

- yL is XORed with *P'n* (with n being the nth iteration).
- The XORed product *yL'* is processed by the *F function*.
- *yL'* is XORed with yR, resulting in *yR'*.
- The values of yL and *yR'* are swapped, so yL has the value or yR and *yR'* has the value of yL.

These four actions will be done in a total 16 rounds. After 16 rounds, *P'17* is XORed with yL and *P'18* is XORed with yR. Finally, yL and yR have to be reunited again as y and all blocks y have to be reunited to single encrypted message.[1] The modified Diffie-Hellman protocol is as follows:

[3]Modern browsers include Internet Explorer, Google Chrome, Mozilla Firefox, Opera and Safari

Table 1: Vulnerable servers per protocol and group of primes. Table based on Adrian et al. [2]

	All 512-bit groups	All 768-bit groups	One 1024-bit group	Ten 1024-bit group
HTTPS Top 1M w/ Active Downgrade	45.100 (8,4%)	45.100 (8,4%)	205.000(37,1%)	309.000(56,1%)
HTTPS Top 1M	118 (0,0%)	407 (0,1%)	98.500(17,9%)	132.000 (24,0%)
IKEv1 IPv4		64.700 (2,6%)	1.690.000 (66,1%)	1.690.000 (66,1%)
IKEv2 IPv4		66.000 (5,8%)	726.000 (63,9%)	726.000 (63,9%)
SSH IPv4			3.600.000 (25,7%)	3.600.000 (25,7%)

Table 2: Mailserver protocols and their support for security protocols.Table based on Adrian et al.[2]

	TLS	Diffie-Hellman	Legacy export-grade Diffie-Hellman	Ten most common 1024-bit groups
SMTP	50,7%	41,4%	14,8%	15,5%
IMAPS	100,0%	75,0%	8,4%	5,4%

Two machines agree on message M. The first machine has secret key a' and the second machine has secret key b'. The first machine sends cipher A', which is calculated by the equation

$$E(M, a') \tag{6}$$

Where E is a function to encrypt M using a modified Blowfish algorithm as described by Adrian et al. [1]. We do not go in further detail about these modifications, as the intent was to give a basic impression on how E works.

Next, the second machine sends cipher B' calculated in a similar fashion:

$$E(M, b') \tag{7}$$

Once both machines received a cipher C, the machines resolve the same key K' with their respective secret key S, using the following equation:

$$E(C, S) \tag{8}$$

2.2.2 *Applicability*

A crucial part of a Logjam attack is finding discrete log x, as mentioned earlier. A precondition is that p in equation (4) has to be a **prime** number. However, since the modified Diffie-Hellman protocol uses M instead of a prime number p, equation (4) does not apply to the modified protocol[4], and is therefore protected against Logjam attacks. [1]

2.3 Impact

Surprisingly, the research conducted by Adrian et al. [1] is a reaction to findings of Adrian et al. [2]. In fact, Adrian et al. [1] uses findings of Adrian et al. [2] as reference and basis for their research. However, Adrian et al. [1] focuses on the prevention of Logjam attacks and the performance of their proposed algorithm, but pay little attention to the impact that could be reduced by adapting their algorithm instead of the original Diffie-Hellman protocol. It would be interesting to examine whether the NSA could break the algorithm proposed by Adrian et al. [1] as opposed to the original protocol, which theoretically could be broken by the NSA [2].

When looking at the research methods of both papers, Adrian et al. [2] scan a random 1% of servers in the IPv4 address space to indicate how susceptible servers are to be compromised (for HTTPS, IKE, and SSH protocols) and to determine which security protocols servers support (for SMTP and IMAPS protocols). We think that using a random 1% of servers is a valid way to represent all active servers is the IPv4 address space.

Furthermore, Adrian et al. [2] use the Alexa top 1 million HTTPS domains as subject to see how many servers are vulnerable to the Logjam attack. From the paper it is not clear why this data set is used. It could be used as representation for all active servers. It could also be possible that the data set is used as representation for websites that have been visited the most, which in turn make Alex top 1 million servers that are vulnerable to a Logjam attack have a bigger impact security-wise than lesser visited websites. More clarity about why this data set is used would give more insight in, for example, what further research should be conducted to get a better understanding on the Logjam's impact.

Moreover, Adrian et al. [1] do mention the impact of the Logjam attack by stating how many HTTPS servers are vulnerable to the attack. This fact is taken from Adrian et al. [2]. The paper about the Logjam attack and its prevention differ three years in date of publication. One could argue that this statistic used by Adrian et al. [1] (the amount of vulnerable HTTPS servers) is rather outdated because measures could have been taken in the meantime (e.g. not supporting Diffie-Hellman anymore) to protect the servers from a Logjam attack. However, Adrian et al. [1] use a statistic from a more recent study as well, of which one could conclude that the Logjam attack is still a problem for a lot of websites. Even though one fact could give some indication on the impact of the Logjam attack, deeper inspection of the vulnerability's cause is neglected by Adrian et al. [1]. It is important to see whether servers still use the same prime numbers for Diffie-Hellman as this results in computation time to find discrete log x [2]. In a more general sense, statistics given by Adrian et al. [2] should be revised.

[4]As long as M is not a prime number

Adrian et al. [2] state in their paper's introduction that "Diffie-Hellman is commonly implemented and deployed with these protocols and find that, in practice, it frequently offers less security that widely believed". Adrian et al. give two reasons for this, primely that a lot of servers support Diffie-Hellman key exchange that is too easy to break as of 2015. And secondly, that often the same parameters for Diffie-Hellman are used. Concluding that people believe that their 'secure' server is actually insecure, should not be derived from those two reasons. There could be numerous others reasons why people use a weak Diffie-Hellman protocol or use often used parameter. People might, for example, prefer weaker Diffie-Hellman protocols for their server to be more accessible for more devices. Or people use often used same parameters because of a lack of knowledge that other parameters can be used instead of the default ones. Or simply because they do not care about their security policy.

In section 2.2, we mentioned that the outside world responded on the findings of Adrian et al. [2]. The response of browsers was to enforce a bigger bitsize of ciphersuites. This is a very pragmatic solution for mitigating risks: let all (modern) browsers enforce stricter policy by only accepting stronger Diffie-Hellman implementations, making a Logjam attack more time costly and hopefully infeasible. Even though this suggestion does not prevent every Logjam attack from being successful, it does require only browsers to do something. This makes the suggestion more likely to be realised as opposed to other suggestion in which servers have to change their key-exchange protocol to, for example, stronger Diffie-Hellman groups based on elliptic curves [2] or a modified Diffie-Hellman protocol as proposed by Adrian et al. [1].

Lastly, a Logjam attack with Active Downgrade takes an average 70 seconds to complete[2]. When indicating the impact of a Logjam attack, it is important to consider the time it takes for the computation to complete. This is important because if a handshake takes too long, a user could get impatient and stop the request before the computation is finished, therefore prevent an attack from happening. We think it would be good if Adrian et al. [2] had mentioned this.

3. CONCLUSIONS AND FURTHER RESEARCH

Adrian et al. [2] give quite an impression on the impact of Diffie-Hellman's vulnerabilities. This is done in numbers of vulnerable servers to the NFS algorithm and the Logjam attack. Additionally, it gives insight on what the outside world (modern browsers, Akamai and TLS developers) does with the findings of Adrian et al. [2]. In contrast, Adrian et al. [1] give an proposal on how to prevent Logjam attacks, but give barely information on their proposal's impact: Adrian et al. [1] show that if servers would use their proposal, the servers are protected against NFS and Logjam attacks, but no insight is given on how many servers or key figures, such as developers of OpenSSL or Apache, actually do something with the proposal. Therefore, this subject is underexposed and some additional research could be conducted to provide insight in the impact of the proposal.

There is some presence of facts about the impact on the Logjam attack and the NFS algorithm. The facts that are given are, as of writing, slightly outdated. Further research about the current state of vulnerable servers would be appropriate.

Adrian et al. [2] conclude that DH-768 can be broken with academic power and DH-1028 theoretically with state-level resources. This conclusion has been drawn in 2015 [2]. As of writing, seven years later, one could question whether those conclusions are still relevant nowadays. Maybe DH-1024 can already be broken with the computing power of a regular PC. New research could be conducted to verify whether the conclusions of Adrian et al. [2] are still valid nowadays.

Furthermore, Adrian et al. [1] propose to use the Blowfish algorithm to encrypt messages as defined with equation (6) and (7). The reason for using Blowfish instead of Rijndael is that Blowfish is faster and more reliable than Rijndael [1]. Again, the research of Adrian et al. [1] is slightly outdated. Maybe the modified Diffie-Hellman protocol can be broken nowadays. If the modified protocol can be broken, further research could be conducted to determine whether using Rijndael could be used instead of Blowfish to modify the Diffie-Hellman protocol.

In the previous section, we discussed that considering computation time is important when indication the impact of a Logjam attack with Active Downgrade. Further research about this topic would be useful to investigate how big of a threat this attack is.

4. ACKNOWLEDGMENTS

Thanks to Piet Allersma and Floris Hooijmans for reviewing this paper.

5. REFERENCES

[1] A. Adrian, M. Cendana, and S. D. H. Permana. Diffie-hellman key exchange modification using blowfish algorithm to prevent logjam attack. *Journal of Telecommunication, Electronic and Computer Engineering*, 10(4), October - December 2018.

[2] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmerman. Imperfect forward secrecy: How diffie-hellman fails in practice. 22nd ACM Conference on Computer and Communications Security (CCS '15), October 2015.

[3] N. K. Valmik and K. V. K. Blowfish algorithm. *IOSR Journal of Computer Engineering*, 16(2), 2014.