

VPN

Antivirus

Online backup

Streaming

Blog

ns

About Us

y Use
y OS/Device
y Country
guides
reviews

» What is the Diffie–Hellman key exchange and how does it work?

[readers](#) and we may receive a commission when you make purchases using the links on our site

What is the Diffie–Hellman key exchange and how does it work?

The Diffie–Hellman key exchange was one of the most important developments in public-key cryptography. Find out why it's still such an important aspect of internet security.

JOSH LAKE - SPECIALIST IN SECURITY, PRIVACY AND ENCRYPTION

UPDATED: March 23, 2021



comparitech

What is the Diffie–Hellman key exchange?



The Diffie–Hellman key exchange was **one of the most important developments in public-key cryptography** and it is still frequently implemented in a range of today's different security protocols.

It allows two parties who have not previously met to securely establish a key which they can use to secure their communications. In this article, we'll explain what it's used for, how it works on a step-by-step basis, its different variations, as well as the security considerations that need to be noted in order to implement it safely.

What is the Diffie-Hellman key exchange?

The Diffie-Hellman key exchange was the **first widely used method of safely developing and exchanging keys over an insecure channel**.

It may not seem so exciting or groundbreaking in the above terms, so let's give an example that explains why the Diffie-Hellman key exchange was such an important milestone in the world of cryptography, and why it is still so frequently used today.

Let's say you're a top secret spy and you need to send some important information to your headquarters. **How would you prevent your enemies from getting ahold of the message?**

The most common solution would be to encrypt the message with a code. The easiest way is to prearrange whichever type of code and key you plan on using beforehand, or to do it over a safe communication channel.

Let's say that you are a particularly bad spy, and you and your headquarters decide to use a weak shift-cipher to encode your messages. In this code, every "a" becomes "b", every "b" becomes "c", every "c" becomes "d", and so on, all the way up to the "z" becoming an "a".

Under this shift cipher, the message “Let’s get dinner” becomes “Mfu’t hfu ejooofs”. Thankfully, in our hypothetical situation, your adversaries are just as incompetent as you are and are unable to crack such a simple code, which keeps them from accessing the contents of the message.

But what happens if you couldn’t arrange a code with your recipient beforehand?

Let’s say you want to communicate with a spy from an allied nation who you have never met before. You don’t have a secure channel over which to talk to them. If you don’t encrypt your message, then any adversary who intercepts it will be able to read the contents. If you encrypt it without telling the ally the code, then the enemy won’t be able to read it, but neither will the ally.

This issue was one of the biggest conundrums in cryptography up until the 1970s:

How can you securely exchange information with someone if you haven’t had the opportunity to share the key ahead of time?

The Diffie-Hellman key exchange was the first publicly-used mechanism for solving this problem. The algorithm allows those who have never met before to safely create a shared key, even over an insecure channel that adversaries may be monitoring.

The history of the Diffie-Hellman key exchange

The Diffie-Hellman key exchange traces its roots back to the 1970s. While the field of cryptography had

developed significantly throughout the earlier twentieth century, these advancements were mainly focused in the area of symmetric-key cryptography.

It wasn't until 1976 that public-key algorithms emerged in the public sphere, when Whitfield Diffie and Martin Hellman published their [paper](#), *New Directions in Cryptography*. The collaboration outlined the mechanisms behind a new system, which would come to be known as the **Diffie-Hellman key exchange**.

The work was partly inspired by earlier developments made by Ralph Merkle. The so-called **Merkle's Puzzles** involve one party creating and sending a number of cryptographic puzzles to the other. These puzzles would take a moderate amount of computational resources to solve.

The recipient would randomly choose one puzzle to solve and then expend the necessary effort to complete it. **Once the puzzle is solved, an identifier and a session key are revealed to the recipient.** The recipient then transmits the identifier back to the original sender, which lets the sender know which puzzle has been solved.

Since the original sender created the puzzles, the identifier lets them know which session key the recipient discovered, and the two parties can use this key to communicate more securely. If an attacker is listening in on the interaction, they will have access to all of the puzzles, as well as the identifier that the recipient transmits back to the original sender.

The identifier doesn't tell the attacker which session key is being used, so the best approach for decrypting the

information is to **solve all of the puzzles to uncover the correct session key**. Since the attacker will have to solve half of the puzzles on average, it ends up being much more difficult for them to uncover the key than it is for the recipient.

This approach provides more security, but it is far from a perfect solution. The Diffie-Hellman key exchange took some of these ideas and made them more complex in order to create a secure method of public-key cryptography.

Although it has come to be known as the Diffie-Hellman key exchange, [Martin Hellman](#) has proposed that the algorithm be named the Diffie-Hellman-Merkle key exchange instead, to reflect the work that Ralph Merkle put towards public-key cryptography.

It was publicly thought that Merkle, Hellman and Diffie were the first people to develop public key cryptography until 1997, when the British Government declassified work done in the early 1970s by **James Ellis, Clifford Cox and Malcolm Williamson**.

It turns out that the trio came up with the first [public-key encryption scheme](#) between 1969 and 1973, but their work was classified for two decades. It was conducted under the Government Communication Headquarters (GCHQ), a UK intelligence agency.

Their discovery was actually the [RSA algorithm](#), so Diffie, Hellman and Merkle were still the first to develop the Diffie-Hellman key exchange, but no longer the first inventors of public-key cryptography.

Where is the Diffie-Hellman key exchange used?

The main purpose of the Diffie-Hellman key exchange is to **securely develop shared secrets that can be used to derive keys. These keys can then be used with symmetric-key algorithms to transmit information in a protected manner.** Symmetric algorithms tend to be used to encrypt the bulk of the data because they are more efficient than public key algorithms.

Technically, the Diffie-Hellman key exchange can be used to establish public and private keys. However, in practice, RSA tends to be used instead. This is because the RSA algorithm is also capable of signing public-key certificates, while the Diffie-Hellman key exchange is not.

The ElGamal algorithm, which was used heavily in PGP, is based on the Diffie-Hellman key exchange, so any protocol that uses it is effectively implementing a kind of Diffie-Hellman.

As one of the most common methods for safely distributing keys, the Diffie-Hellman key exchange is **frequently implemented in security protocols such as TLS, IPsec, SSH, PGP, and many others.** This makes it an integral part of our secure communications.

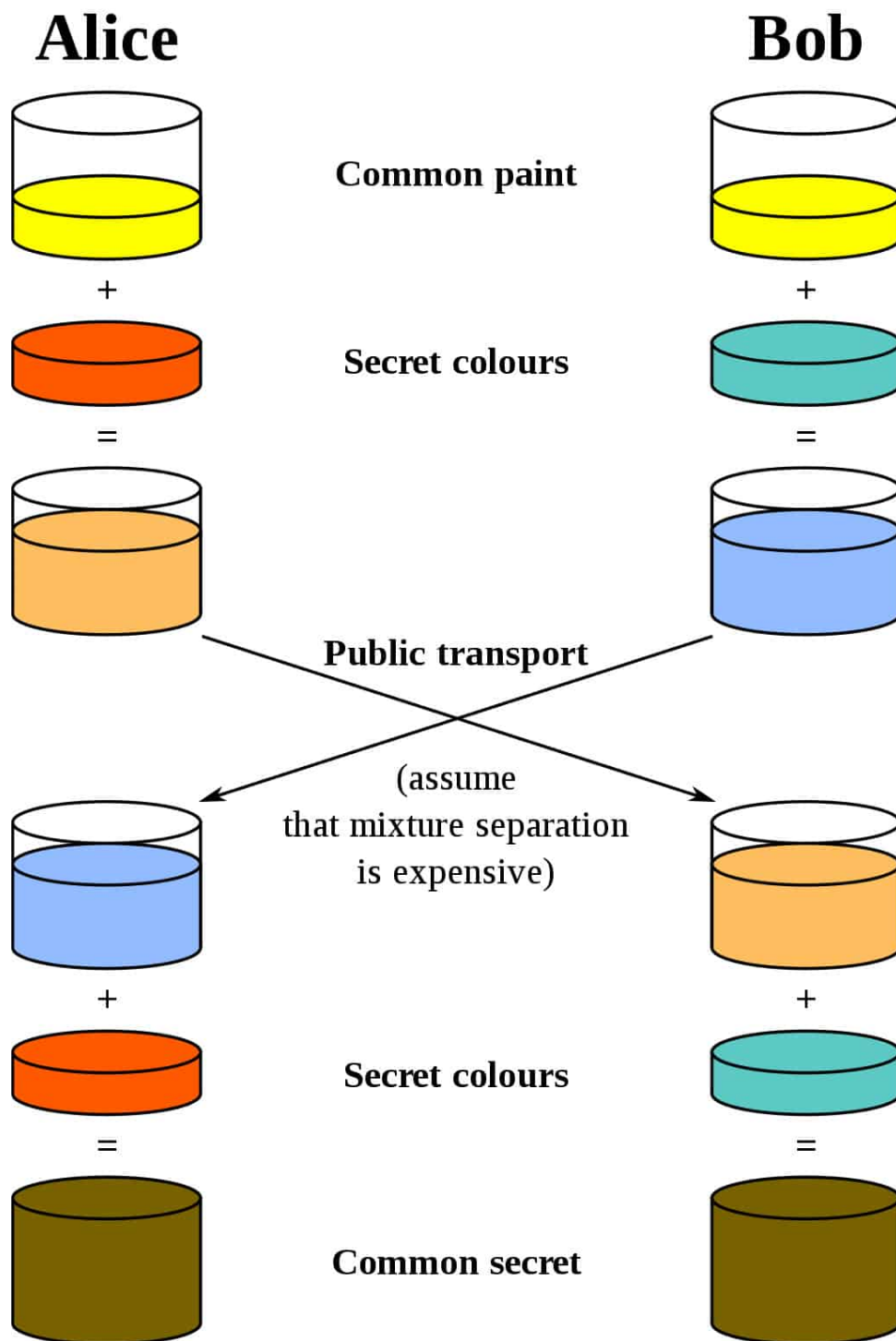
As part of these protocols, the Diffie-Hellman key exchange is often used to help secure your connection to a website, to remotely access another computer, and for sending encrypted emails

How does the Diffie-Hellman key exchange work?

The Diffie-Hellman key exchange is complex and it can be difficult to get your head around how it works. **It uses very large numbers and a lot of math**, something that many of us still dread from those long and boring high school lessons.

To make things a bit easier to understand, **we will start by explaining the Diffie-Hellman key exchange with an analogy**. Once you have a big-picture idea of how it works, we'll move on to a more technical description of the underlying processes.

The best analogy for the Diffie-Hellman scheme is to think of **two people mixing paint**. Let's use the cryptography standard, and say that their names are Alice and Bob. **They both agree on a random color to start with**. Let's say that they send each other a message and **decide on yellow as their common color**, just like in the diagram below:



They set their own color. They do not tell the other party their choice. Let's say that Alice chooses **red**, while Bob chooses a **slightly-greenish blue**.

The next step is for both Alice and Bob to mix their secret color (red for Alice, greenish-blue for Bob) with the yellow that they mutually agreed upon. According to

the diagram, Alice ends up with an **orangish mix**, while Bob's result is a **deeper blue**.

Once they have finished the mixing, they send the result to the other party. **Alice receives the deeper blue**, while **Bob is sent the orange-colored paint**.

Once they have received the mixed result from their partner, they then add their secret color to it. **Alice takes the deeper blue and adds her secret red paint**, while **Bob adds his secret greenish-blue to the orange mix he just received**.

The result? **They both come out with the same color**, which in this case is a disgusting brown. It may not be the kind of color that you would want to paint your living room with, but it is a shared color nonetheless. This shared color is referred to as the **common secret**.

The critical part of the Diffie-Hellman key exchange is that both parties end up with the same result, without ever needing to send the entirety of the common secret across the communication channel.

Choosing a common color, their own secret colors, exchanging the mix and then adding their own color once more, gives both parties a way to arrive at the same common secret without ever having to send across the whole thing.

If an attacker is listening to the exchange, all that they can access is the common yellow color that Alice and Bob start with, as well as the mixed colors that are exchanged. Since this is really done with enormous numbers instead of paint, **these pieces of information aren't enough for the attack to discern either of the initial secret colors, or the common secret**

(technically it is possible to compute the common secret from this information, but in a secure implementation of the Diffie-Hellman key exchange, it would take an unfeasible amount of time and computational resources to do so).

This structure of the Diffie-Hellman key exchange is what makes it so useful. **It allows the two parties to communicate over a potentially dangerous connection and still come up with a shared secret that they can use to make encryption keys for their future communications.** It doesn't matter if any attackers are listening in, because the complete shared secret is never sent over the connection.

The technical details of the Diffie-Hellman key exchange

Time for some math...

Don't worry, we'll take it slow and try to make the whole process as easy to understand as possible. It follows a similar premise as the analogy shown above, but instead of mixing and sending colors, **the Diffie-Hellman scheme actually makes calculations based on exceptionally-large prime numbers, then sends them across.**

To ensure security, it is recommended that the **prime (p) is at least 2048 bits long**, which is the binary equivalent of a decimal number of about this size:

```
415368757628736598425938247569843765827634879128375827365928736
84273684728938572983759283475934875938475928475928739587249587
29873958729835792875982795837529876348273685729843579348795827
```

93857928739548772397592837592478593867045986792384737826735267
3547623568734869386945673456827659498063849024875809603947902
7945982730187439759284620950293759287049502938058920983945872
0948602984912837502948019371092480193581037995810937501938507913
95710937597019385089103951073058710393701934701938091803984091804
98109380198501398401983509183501983091079180395810395190395180935
8109385019840193580193840198340918093851098309180019

To prevent anyone's head from exploding, we'll run this explanation through with much smaller numbers. Be aware that **the Diffie-Hellman key exchange would be insecure if it used numbers as small as those in our example**. We are only using such small numbers to demonstrate the concept in a simpler manner.

In the most basic form of the Diffie-Hellman key exchange, **Alice and Bob begin by mutually deciding upon two numbers to start with**, as opposed to the single common point in the example above. These are **the modulus (p) and the base (g)**.

In practical use, **the modulus (p) is a very large prime number**, while **the base (g) is relatively small to simplify calculations**. The base (g) is derived from a cyclic group (G) that is normally generated well before the other steps take place.

For our example, let's say that the modulus (p) is **17**, while the base (g) is **4**.

Once they have mutually decided on these numbers, Alice settles on a secret number (**a**) for herself, while Bob chooses his own secret number (**b**). Let's say that they choose:

$$a = 3$$

$$b = 6$$

Alice then performs the following calculation to give her the number that she will send to Bob:

$$A = g^a \bmod p$$

In the above calculation, **mod** signifies a modulo operation. These are essentially calculations to figure out the remainder after dividing the left side by the right. As an example:

$$15 \bmod 4 = 3$$

If you understand how modulo operations work, you can do them yourself in the following calculations, otherwise you can use an [online calculator](#).

So let's put our numbers into the formula:

$$A = 4^3 \bmod 17$$

$$A = 64 \bmod 17$$

$$A = 13$$

When we do the same for Bob, we get:

$$B = 4^6 \bmod 17$$

$$B = 4096 \bmod 17$$

$$B = 16$$

Alice then sends her result (A) to Bob, while Bob sends his figure (B) to Alice. Alice then calculates the shared secret (s) using the number she received from Bob (B) and her secret number (a), using the following formula:

$$s = B^a \bmod p$$

$$s = 16^3 \bmod 17$$

$$s = 4,096 \bmod 17$$

$$s = 16$$

Bob then performs what is essentially the same calculation, but with the number that Alice sent him (A), as well as his own secret number (b):

$$s = A^b \bmod p$$

$$s = 13^6 \bmod 17$$

$$s = 4,826,809 \bmod 17$$

$$s = 16$$

As you can see, both parties ended up with the same result for s , **16**. This is the shared secret, which only Alice and Bob know. They can then use this to set up a key for symmetric encryption, allowing them to safely send information between themselves in a way that only they can access it.

*Note that although **B** and **s** are the same in the example above, this is just a coincidence based on the small numbers that were chosen for this illustration. Normally, these values would not be the same in a real implementation of the Diffie-Hellman key exchange.*

Even though much of the above data is sent across the channel in cleartext (p , g , A and B) and can be read by potential attackers, the shared secret (s) is never transmitted. It would not be practical for an attacker to calculate the shared secret (s) or either of the secret

numbers (a and b) from the information that is sent in cleartext.

Of course, this assumes that the Diffie-Hellman key exchange is properly implemented and sufficiently large numbers are used. As long as these provisions are adhered to, the Diffie-Hellman key exchange is considered a safe way to establish a shared secret which can be used to secure future communications.

Establishing a shared key between multiple parties

The Diffie-Hellman key exchange can also be used to set up a shared key with a greater number of participants. It works in the same manner, except further rounds of the calculations are needed for each party to add in their secret number and end up with the same shared secret.

Just like in the two-party version of the Diffie-Hellman key exchange, some parts of the information are sent across insecure channels, but not enough for an attacker to be able to compute the shared secret.

Why is the Diffie-Hellman key exchange secure?

On a mathematical level, the Diffie-Hellman key exchange relies on one-way functions as the basis for its security. These are calculations which are simple to do one way, but much more difficult to calculate in reverse.

More specifically, it relies on the Diffie-Hellman problem, which assumes that under the right parameters, it is

infeasible to calculate gab from the separate values of g , ga and gb . There is currently no publicly known way to easily find gab from the other values, which is why the Diffie-Hellman key exchange is considered secure, despite the fact that attackers can intercept the values p , g , A , and B .

Authentication & the Diffie-Hellman key exchange

In the real world, the Diffie-Hellman key exchange is rarely used by itself. The main reason behind this is that **it provides no authentication, which leaves users vulnerable to man-in-the-middle attacks.**

These attacks can take place when the Diffie-Hellman key exchange is implemented by itself, because **it has no means of verifying whether the other party in a connection is really who they say they are.** Without any form of authentication, **users may actually be connecting with attackers** when they think they are communicating with a trusted party.

For this reason, the Diffie-Hellman key exchange is generally implemented alongside some means of authentication. This often involves using digital certificates and a public-key algorithm, such as RSA, to verify the identity of each party.

Variations of the Diffie-Hellman key exchange

The Diffie-Hellman key exchange can be implemented in a number of different ways, and it has also provided the basis for several other algorithms. Some of these

implementations provide authorization, while others have various cryptographic features such as perfect forward secrecy.

Elliptic-curve Diffie-Hellman

Elliptic-curve Diffie-Hellman takes advantage of the algebraic structure of elliptic curves to allow its implementations to achieve a similar level of security with a smaller key size. A 224-bit elliptic-curve key provides the [same level of security](#) as a 2048-bit RSA key. This can make exchanges more efficient and reduce the storage requirements.

Apart from the smaller key length and the fact that it relies on the properties of elliptic curves, elliptic-curve Diffie-Hellman operates in a similar manner to the standard Diffie-Hellman key exchange.

TLS

[TLS](#), which is a protocol that is used to secure much of the internet, can use the Diffie-Hellman exchange in three different ways: anonymous, static and ephemeral. In practice, only ephemeral Diffie-Hellman should be implemented, because the other options have security issues.

- **Anonymous Diffie-Hellman** – This version of the Diffie-Hellman key exchange doesn't use any authentication, leaving it vulnerable to man-in-the-middle attacks. It should not be used or implemented.
- **Static Diffie-Hellman** – Static Diffie-Hellman uses certificates to authenticate the server. It does not

authenticate the client by default, nor does it provide forward secrecy.

- **Ephemeral Diffie-Hellman** – This is considered the most secure implementation because it provides perfect forward secrecy. It is generally combined with an algorithm such as DSA or RSA to authenticate one or both of the parties in the connection. Ephemeral Diffie-Hellman uses different key pairs each time the protocol is run. This gives the connection perfect forward secrecy, because even if a key is compromised in the future, it can't be used to decrypt all of the past messages.

ElGamal

ElGamal is a public-key algorithm built on top of the Diffie-Hellman key exchange. Like Diffie-Hellman, it contains no provisions for authentication on its own, and is generally combined with other mechanisms for this purpose.

ElGamal was mainly used in PGP, GNU Privacy Guard and other systems because its main rival, RSA, was patented. RSA's patent expired in 2000, which allowed it to be implemented freely after that date. Since then, ElGamal has not been implemented as frequently.

STS

The Station-to-Station (STS) protocol is also based on the Diffie-Hellman key exchange. It's another key agreement scheme, however it provides protection against man-in-the-middle attacks as well as perfect forward secrecy.

WHAT'S IN THIS ARTICLE?

[What is the Diffie-Hellman key exchange?](#)

[Where is the Diffie-Hellman key exchange used?](#)

[How does the Diffie-Hellman key exchange work?](#)

[The technical details of the Diffie-Hellman key exchange](#)

[Show More](#)

It requires both parties in the connection to already have a keypair, which is used to authenticate each side. If the parties aren't already known to each other, then certificates can be used to validate the identities of both parties.

The Diffie-Hellman key exchange & RSA

As we discussed earlier, **the Diffie-Hellman key exchange is often implemented alongside RSA or other algorithms to provide authentication for the connection.** If you are familiar with [RSA](#), you may be wondering **why anyone would bother using the Diffie-Hellman key exchange as well**, since RSA enables parties who have never previously met to communicate securely.

RSA allows its users to encrypt messages with their correspondent's public key, so that they can only be decrypted by the matching private key. However, in practice, **RSA isn't used to encrypt the entirety of the communications—this would be far too inefficient.**

Instead, RSA is often only used as a means to authenticate both parties. It does this with **the digital certificates of each party, which will have been verified by a certificate authority** to prove that a certificate owner is truly who they say they are, and that the public key on the certificate actually belongs to them.

For mutual authentication, **each party will sign a message using their private key and then send it to their communication partner.** Each recipient can then **verify the identity of the other party by checking the signed messages against the public key on their**

communication partner's digital certificate (see the above-mentioned article on RSA for more detail on how this works, particularly the **Signing messages** section).

Now that both parties have been authenticated, it's technically possible to continue using RSA to safely send encrypted messages between themselves, however it would end up being too inefficient.

To get around this inefficiency, many security protocols use an algorithm such as **the Diffie-Hellman key exchange to come up with a common secret that can be used to establish a shared symmetric-key. This symmetric key is then used in a symmetric-key algorithm, such as AES, to encrypt the data** that the two parties intend to send securely between themselves.

It may seem like a complex and convoluted process, but it ends up being much quicker and less-demanding on resources when compared to using a public-key algorithm for the whole exchange. This is because **symmetric-key encryption is orders of magnitude more efficient than public-key encryption.**

In addition to the inefficiencies that we just mentioned, there are some other downsides that would come from solely using RSA. **RSA needs padding to make it secure**, so an additional algorithm would need to be implemented appropriately alongside it to make it safe.

RSA doesn't provide perfect forward secrecy, either, which is another disadvantage when compared to the ephemeral Diffie-Hellman key exchange. Collectively, these reasons are why, in many situations, it's best to only apply RSA in conjunction with the Diffie-Hellman key exchange.

Alternatively, the Diffie-Hellman key exchange can be combined with an algorithm like the Digital Signature Standard (DSS) to provide authentication, key exchange, confidentiality and check the integrity of the data. In such a situation, RSA is not necessary for securing the connection.

Security issues of the Diffie-Hellman key exchange

The security of the Diffie-Hellman key exchange is dependent on how it is implemented, as well as the numbers that are chosen for it. As we stated above, it has no means of authenticating the other party by itself, but in practice other mechanisms are used to ensure that the other party in a connection is not an impostor.

Parameters for number selection

If a real-world implementation of the Diffie-Hellman key exchange used numbers as small as those in our example, it would make the exchange process trivial for an attacker to crack. But it's not just the size of the numbers that matter – the numbers also need to be sufficiently random. If a random number generator produces a predictable output, it can completely undermine the security of the Diffie-Hellman key exchange.

The number p should be 2048 bits long to ensure security. **The base, g , can be a relatively small number like 2**, but it needs to come from an **order of G** that has a large prime factor

The Logjam attack

The Diffie-Hellman key exchange was designed on the basis of the discrete logarithm problem being difficult to solve. The most effective publicly known mechanism for finding the solution is the [number field sieve algorithm](#).

The capabilities of this algorithm were taken into account when the Diffie-Hellman key exchange was designed. By 1992, it was known that for a given group, G , three of the four steps involved in the algorithm could potentially be [computed beforehand](#). If this progress was saved, the final step could be calculated in a comparatively short time.

This wasn't too concerning until it was realized that a significant portion of internet traffic uses the same groups that are 1024 bits or smaller. In 2015, an [academic team](#) ran the calculations for the most common 512-bit prime used by the Diffie-Hellman key exchange in TLS.

They were also able to downgrade 80% of TLS servers that supported DHE-EXPORT, so that they would accept a 512-bit export-grade Diffie-Hellman key exchange for the connection. This means that **each of these servers is vulnerable to an attack from a well-resourced adversary**.

The researchers went on to extrapolate their results, **estimating that a nation-state could break a 1024-bit prime. By breaking the single most-commonly used 1024-bit prime, the academic team estimated that an adversary could monitor 18% of the one million most popular HTTPS websites.**

They went on to say that a second prime would enable the adversary to decrypt the connections of 66% of VPN servers, and 26% of SSH servers. Later in the report, the academics suggested that the NSA may already have these capabilities.

“A close reading of published NSA leaks shows that the agency’s attacks on VPNs are consistent with having achieved such a break.”

Despite this vulnerability, the Diffie-Hellman key exchange can still be secure if it is implemented correctly. As long as a 2048-bit key is used, the Logjam attack will not work. Updated browsers are also secure from this attack.

Is the Diffie-Hellman key exchange safe?

While the Diffie-Hellman key exchange may seem complex, it is a fundamental part of securely exchanging data online. As long as it is implemented alongside an appropriate authentication method and the numbers have been selected properly, it is not considered vulnerable to attack.

The Diffie-Hellman key exchange was an innovative method for helping two unknown parties communicate safely when it was developed in the 1970s. While we now implement newer versions with larger keys to protect against modern technology **the protocol itself looks like it will continue to be secure until the arrival of quantum computing** and the advanced attacks that will come with it.

How will quantum computing affect the Diffie-Hellman key exchange?

Quantum computing is an emerging branch of computing that continues to make breakthroughs. The specifics of how quantum computers work are complicated and out of the scope of this article, however the technology does present significant problems to the field of cryptography.

The simple explanation is that quantum computers are expected to be able to solve certain problems that are currently not feasible for classical computers. This will open up a lot of doors and bring about new possibilities. Sufficiently powerful quantum computers will be able to run quantum algorithms that can more effectively solve various mathematical problems.

While this may sound great, the security of many of our current cryptographic mechanisms rely on these problems being difficult to solve. If these mathematical problems become easier to compute, it also becomes easier to break these cryptographic mechanisms.

One of these quantum algorithms is [Grover's algorithm](#). When quantum computers become powerful enough, it will speed up attacks against symmetric-key ciphers like [AES](#). However, it can easily be mitigated by doubling the key size.

The biggest concern is how [Shor's algorithm](#) will affect public-key cryptography. This is because the security of most common public-key algorithms rely on the immense difficulty of solving one of these three computations:

- The [discrete logarithm problem](#)
- The [integer factorization problem](#)
- The [elliptic-curve discrete logarithm problem](#)

The specifics of each don't really matter, but you can follow the links if you want additional information. The important thing is that once sufficiently powerful quantum computers arrive, it will become much more practical to solve these problems with Shor's algorithm. As these problems become easier to solve, the cryptographic systems that rely on them will become less secure.

Public-key cryptography plays a fundamental role in protecting our communications, which is why quantum computing presents a huge challenge for cryptographers.

In the case of the Diffie-Hellman key exchange, its security relies on the impracticality of being able to solve the discrete logarithm problem with current technology and resources. However, the threats from Shor's algorithm loom closer with each advance in quantum computing.

It's hard to come up with a rough timeline of when quantum computing will seriously threaten the Diffie-Hellman key exchange because some researchers are far more optimistic than others. Despite this, replacements for the Diffie-Hellman key exchange and other public-key algorithms are being developed to make sure we are prepared for when the time comes.

Potential replacements for the Diffie-Hellman key exchange

The danger from quantum computers isn't immediate, so the cryptographic community is yet to settle on specific alternatives to the Diffie-Hellman key exchange. However, numerous paths are being pursued. These include:

- [Lattice-based cryptography](#)
- [Multivariate cryptography](#)
- [Elliptic-curve isogeny cryptography](#)

We still don't know exactly how the post-quantum world will look for cryptography, but the security community is actively working on the problems and keeping up with the advances in the quantum computing world. While there will be big changes in the future, it's nothing that the average person needs to fear—you probably won't even notice when any changes do take place.

3 Comments

Leave a comment



paolo66

June 28, 2021 at 8:02 pm

Hi, very interesting article, but I have a question: how can someone divide whatever number by a 2048 bit long number, as suggested?? furthermore, to have a meaningful quotient the dividend has to be greater than divisor...

Thanks,
paolo

Reply

**Josh Lake***June 28, 2021 at 11:16 pm*

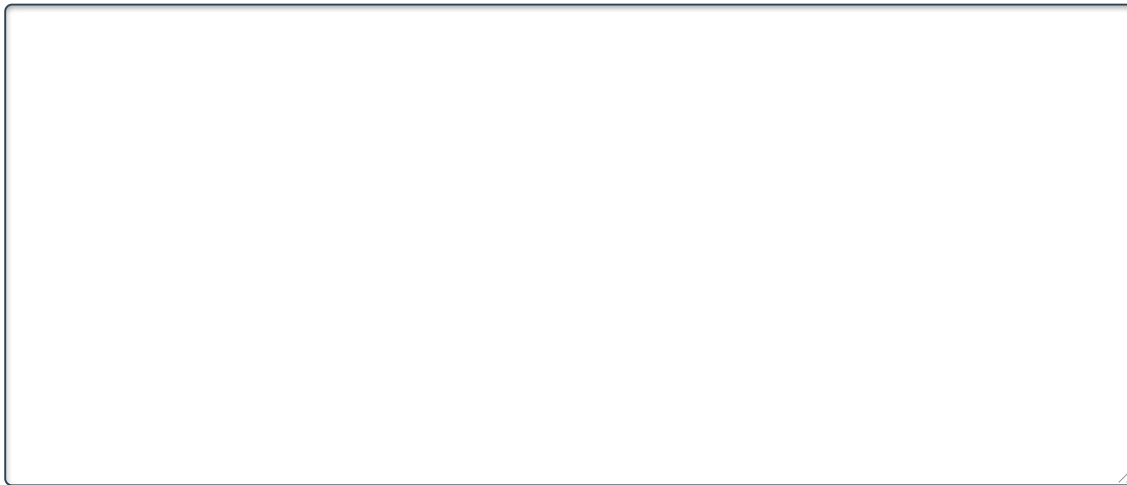
Hi Paolo,

You would need a calculator that is programmed to deal with 2048-bit numbers (617 digits long, I think). These numbers are so large that most people never need to deal with them, so most calculators aren't set up to handle them either. As for your second point, this is a modular operation, not normal division. If you were to reverse the numbers and try $4 \bmod 15$, your answer would be 4, because 15 goes into 4 zero times, with a remainder of 4. You can try it for yourself on the calculator.

[Reply](#)**Janak***September 15, 2020 at 12:39 pm*

Best article for understanding Diffie Hellman exchange. Btw disgusting brown was hilarious.

[Reply](#)[Leave a Reply](#)[Comment](#)



Name *

Leave Comment

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)



[Home](#) [Blog](#) [Authors](#) [Privacy policy](#) [Cookies Policy](#) [Terms of use](#) [Disclosure](#)
[About Comparitech](#) [Contact Us](#) [Accessibility](#)

© 2021 Comparitech Limited. All rights reserved.

Comparitech.com is owned and operated by Comparitech Limited, a registered company in England and Wales (Company No. 09962280), Suite 3 Falcon Court Business Centre, College Road, Maidstone, Kent, ME15 6TF, United Kingdom. Telephone +44(0)333 577 0163