Cryptography for Absolute Beginners

medium.com/@hashelse/cryptography-for-absolute-beginners-3e274f9d6d66

January 16, 2019

We live in a post-Snowden world. For many, that means assuming none of your digital assets are safe from surveillance.

There are ways, however, to use the internet with insane mathematics in your favor to ensure that no one can see whatever it is that you're sending to someone else.

I'll be trying to explain in detail how all of this works.Feel free to skim over the theory and play with the fun stuff.

Cryptography is an ancient mathematical science that was originally used for military communications, and designed to conceal the contents of a message should it fall into the hands of the enemy. Recent developments in cryptography have added additional uses, including mechanisms for authenticating users on a network, ensuring the integrity of transmitted information and preventing users from repudiating (i.e. rejecting ownership of) their transmitted messages.

Today, encryption is an integral part of many of the tools and protocols we rely on to protect the security of our everyday transactions and online communications. Encryption can be used on the physical layer of the Internet to scramble data that's being transmitted via cable or radio communications. It adds support for secure communications to plaintext protocols like the Hypertext Transfer Protocol (HTTP), which enables Web browsing, and can protect the integrity of data exchanged through applications like email and mobile messengers. You can also encrypt data that is stored on devices like cellphones or computers, shielding the local copies of emails, text messages, documents, and photos from unauthorized snooping.

How and at what layer your data is encrypted makes a huge difference. Just because a product or service uses encryption doesn't necessarily mean that everything that's stored on or sent over that platform is completely private. For example, Google now makes the HTTPS protocol (HTTP over an encrypted connection) the default for all Gmail traffic, which prevents unauthorized users from reading emails while they travel between Google's email servers and end users' computers — but it does nothing to stop Google itself from accessing plaintext copies of those conversations. If you don't want your email provider to be able to read your messages, you have to take additional steps to implement end-to-end encryption, which refers to a system in which "messages are encrypted in a way that allows only the unique recipient of a message to decrypt it, and not anyone in between." With end-to-end encryption, you encrypt the contents of a message on your local machine or device. That data is then transmitted as ciphertext by the email provider to the intended recipient, who is the only person who can decrypt and read it.

Is all of law enforcement and US government against encryption?

Not necessarily. Law enforcement and intelligence officials have often said they appreciate the benefits of encryption when it comes to protecting data from threats such as hackers or foreign governments. They just want to be sure there's a way to access encrypted data — especially communications — for their investigations.

First, how does encryption protect my data?

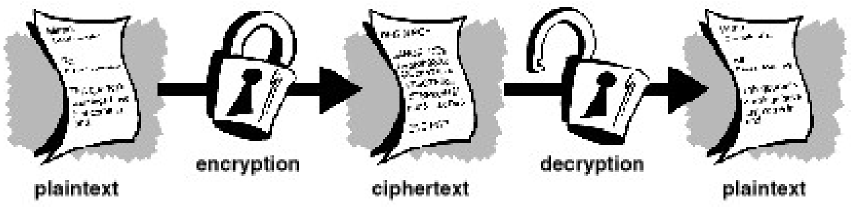
Encryption algorithms use math to "scramble" data so it can't be read by an unauthorized person — such as a hacker or government seeking to break in.

Data can be encrypted in two places: First, it can be encrypted "in transit," such as when you send information from your browser to a website. Second, data can be encrypted when it's "at rest," such as when it is stored on a computer or on a server.

Data that can be read and understood without any special measures is called plaintext or cleartext. The method of disguising plaintext in such a way as to hide its substance is called encryption. Encrypting plaintext results in unreadable gibberish called ciphertext. You use encryption to ensure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting ciphertext to its original plaintext is called decryption.

Ciphertext

If this article was encrypted, anyone who intercepts the encrypted version of it would instead see a very long string of unintelligible numbers and letters, such as: "SNaqi82xleab92lkafdtuijgjf0dgfdojtkr8vcp2dso"



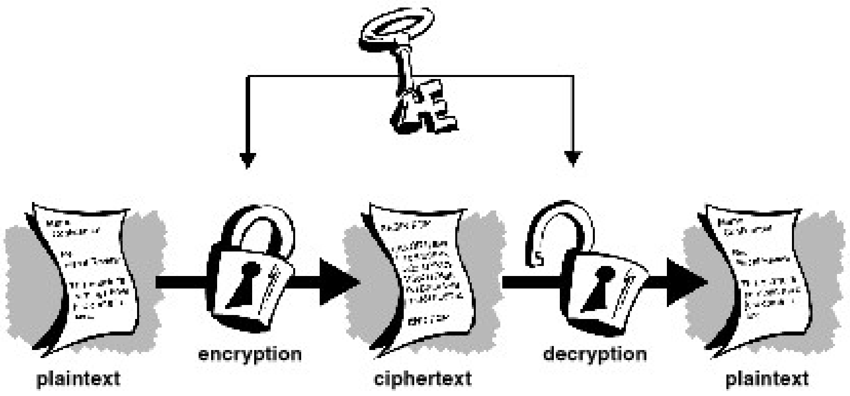
Symmetric-key encryption

To unscramble the encrypted data, you will need an encryption "key."(kinda like a Password) The key is a very large number that an encryption algorithm uses to change the data back into a readable form. Without the key, no one but the owner of the encrypted data will be able to access a readable version. This unscrambling process is called "decryption."This is what's known as symmetric-key encryption.

Symmetric-key encryption has benefits. It is very fast. It is especially useful for encrypting data that is not going anywhere. However, conventional encryption alone as a means for transmitting secure data can be quite expensive simply due to the difficulty of secure key distribution.

Recall a character from your favorite spy movie: the person with a locked briefcase handcuffed to his or her wrist. What is in the briefcase, anyway? It's probably not the missile launch code/ biotoxin formula/ invasion plan itself. It's the key that will decrypt the secret data.

For a sender and recipient to communicate securely using Symmetric-key encryption, they must agree upon a key and keep it secret between themselves. If they are in different physical locations, they must trust a courier, the Bat Phone, or some other secure communication medium to prevent the disclosure of the secret key during transmission. Anyone who overhears or intercepts the key in transit can later read, modify, and forge all information encrypted or authenticated with that key. From DES to Captain Midnight's Secret Decoder Ring, the persistent problem with Symmetric-key encryption is key distribution: how do you get the key to the recipient without someone intercepting it?



Public key cryptography (Also known as Asymmetric encryption)

The problems of key distribution are solved by public key cryptography. Public key cryptography is an asymmetric scheme that uses a pair of keys for encryption: a public key, which encrypts data, and a corresponding private, or secret key for decryption. You publish your public key to the world while keeping your private key secret. Anyone with a copy of your public key can then encrypt information that only you can read. Even people you have never met.

It is computationally infeasible to deduce the private key from the public key. Anyone who has a public key can encrypt information but cannot decrypt it. Only the person who has the corresponding private key can decrypt the information.

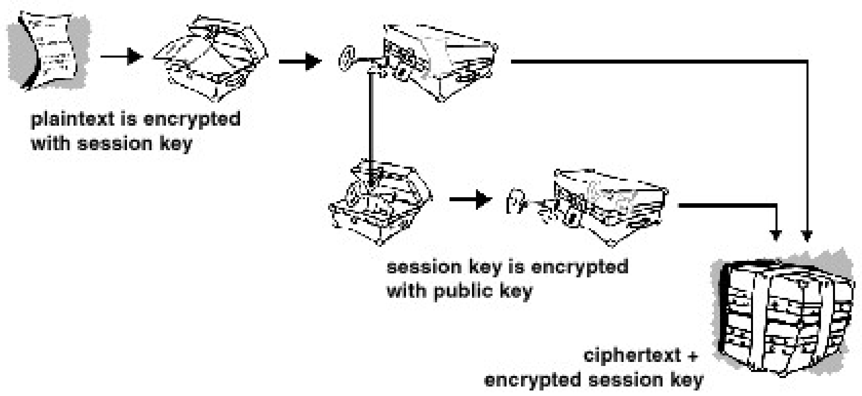
The primary benefit of public key cryptography is that it allows people who have no preexisting security arrangement to exchange messages securely. The need for sender and receiver to share secret keys via some secure channel is eliminated; all communications involve only public keys, and no private key is ever transmitted or shared. Some examples of public-key cryptosystems are Elgamal (named for its inventor, Taher Elgamal), RSA (named for its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman), Diffie-Hellman (named, you guessed it, for its inventors), and DSA, the Digital Signature Algorithm (invented by David Kravitz).

Because conventional cryptography was once the only available means for relaying secret information, the expense of secure channels and key distribution relegated its use only to those who could afford it, such as governments and large banks (or small children with secret decoder rings). Public key encryption is the technological revolution that provides strong cryptography to the adult masses. Remember the courier with the locked briefcase handcuffed to his wrist? Public-key encryption puts him out of business (probably to his relief).

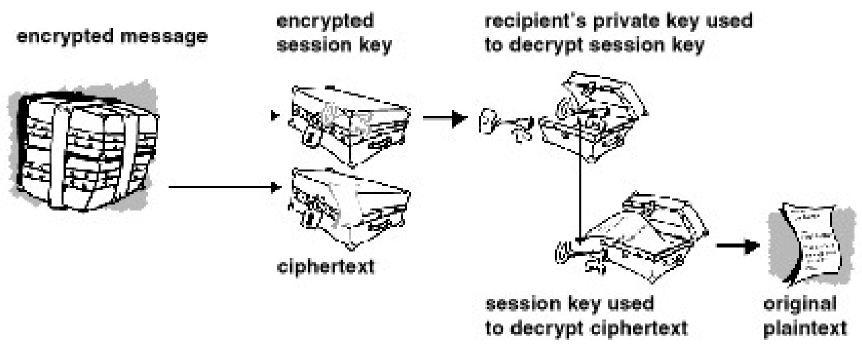
PGP combines some of the best features of both conventional and public key cryptography. PGP is a hybrid cryptosystem. When a user encrypts plaintext with PGP, PGP first compresses the plaintext. Data compression saves modem transmission time and disk space and, more importantly, strengthens cryptographic security. Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher. Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis. (Files that are too short to compress or which don't compress well aren't compressed.)

PGP then creates a session key, which is a one-time-only secret key. This key is a random number generated from the random movements of your mouse and the keystrokes you type. This session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext; the result is ciphertext. Once the data is encrypted, the session key is then encrypted to the recipient's public key. This public key-encrypted session key is transmitted along with the ciphertext to the recipient.

If you did'nt get this it's okay, watch this video


plaintext is encrypted with session key
session key is encrypted with public key
ciphertext + encrypted session key

Decryption works in the reverse. The recipient's copy of PGP uses his or her private key to recover the temporary session key, which PGP then uses to decrypt the conventionally-encrypted ciphertext.


encrypted message
encrypted session key
recipient's private key used to decrypt session key
ciphertext
session key used to decrypt ciphertext
original plaintext

The combination of the two encryption methods combines the convenience of public key encryption with the speed of symmetric encryption. symmetric encryption is about 1,000 times faster than public key encryption. Public key encryption in turn provides a solution to key distribution and data transmission issues. Used together, performance and key distribution are improved without any sacrifice in security.

Keys

A key is a value that works with a cryptographic algorithm to produce a specific ciphertext. Keys are basically really, really, really big numbers. Key size is measured in bits; the number representing a 1024-bit key is darn huge. In public key cryptography, the bigger the key, the more secure the ciphertext.

While the public and private keys are mathematically related, it’s very difficult to derive the private key given only the public key; however, deriving the private key is always possible given enough time and computing power. This makes it very important to pick keys of the right size; large enough to be secure, but small enough to be applied fairly quickly. Additionally, you need to consider who might be trying to read your files, how determined they are, how much time they have, and what their resources might be.

Larger keys will be cryptographically secure for a longer period of time. If what you want to encrypt needs to be hidden for many years, you might want to use a very large key. Of course, who knows how long it will take to determine your key using tomorrow’s faster, more efficient computers? There was a time when a 56-bit symmetric key was considered extremely safe.

Digital signatures

A major benefit of public key cryptography is that it provides a method for employing digital signatures. Digital signatures enable the recipient of information to verify the authenticity of the information’s origin, and also verify that the information is intact. Thus, public key digital signatures provide authentication and data integrity. A digital signature also provides non-repudiation, which means that it prevents the sender from claiming that he or she did not actually send the information. These features are every bit as fundamental to cryptography as privacy, if not more.

A digital signature serves the same purpose as a handwritten signature. However, a handwritten signature is easy to counterfeit. A digital signature is superior to a handwritten signature in that it is nearly impossible to counterfeit, plus it attests to the contents of the information as well as to the identity of the signer.

Some people tend to use signatures more than they use encryption. For example, you may not care if anyone knows that you just deposited $1000 in your account, but you do want to be darn sure it was the bank teller you were dealing with.

Instead of encrypting information using someone else's public key, you encrypt it with your private key. If the information can be decrypted with your public key, then it must have originated with you.

Note that you will need to have GnuPG (GPG) installed before starting the tutorial.

GPG (PGP) Command Line — Basic Tutorial

It's called PGP, which stands for "pretty good privacy," and it's a way to encrypt your messages. Encryption, at its most basic form, is a way to cypher a message so that if anyone that sees the data in transit they have no way to know what the message says. OpenPGP is the most popular standard for digital encryption.

In fact, Edward Snowden first contacted journalist Laura Poitras to inform her of his trove of documents using PGP.

So let's take a look at what PGP is and how easy it is to use.

Quick recap (if you skipped the theory):

Encryption is basically a way of jumbling digital data so that no one can see what it really says while it's being sent. For the purposes of this explainer, we're going to focus on what's called "public key encryption". This uses a multitude of cryptographic techniques to cipher every message using two factors that are constant to every person using PGP: a public key and a private key.

A public key is the information that is needed to encrypt a message. People wishing to receive encrypted messages make their public key readily available, as it's the only way for sources to begin the process of sending secure messages

How it works

Note: There are gui apps to do this process simpler and easier, but here we are using the Command Line to achieve the same. (We'll be using the Linux Command Line to achieve this, Windows users can check out a tutorial on GUI Application which does the same)

First, you'll want to **generate a key for yourself:**

```
gpg --gen-key
```

You’ll be asked to enter a few details. **Don’t forget these details.**

To list your public keys:

```
gpg --list-keys
```

Now, let’s say your name is John Doe, and you want to send a message to Jane Doe. This is how you would do it (note that all names used must be the names you see when listing the keys).

First, export your public key:

```
gpg --export --armor youremail@example.com > publickey.asc
```

or

```
gpg --export --armor yourname > publickey.asc
```

Send this file to Jane Doe. Get her to do the same.

To import someone else’s public key:

```
gpg --import publickey.asc
```

Now that you’ve imported Jane Doe’s key, let’s send her an encrypted message.

To encrypt a file to send to Jane Doe:

```
gpg --encrypt --recipient receiversname filename.txt
```

Example:

```
gpg --encrypt --recipient Jane Doe secretmessage.txt
```

or, if the previous command doesn’t work:

```
gpg -e -u “sender’s name (you)” -r “name of the receiver’s key” filename.txt
```

Example:

```
gpg -e -u “John Doe” -r “Jane Doe” secretmessage.txt
```

This will create a file called secretmessage.txt.pgp. Send this to Jane Doe.

Now Jane has received your file. This is how she decrypts it:

To decrypt to command line (meaning that you’ll only see the message in the command line, and it won’t be saved decrypted to your hard drive):

```
gpg --decrypt filename.txt.gpg
```

To decrypt to disk:

```
gpg filename.txt.gpg
```

Done!