**What is BIOS?**

BIOS (basic input/output system) is the program a personal computer's [microprocessor](https://whatis.techtarget.com/definition/microprocessor-logic-chip) uses to get the computer system started after you turn it on. It also manages data flow between the computer's [operating system](https://whatis.techtarget.com/definition/operating-system-OS) and attached devices such as the [hard disk](https://searchstorage.techtarget.com/definition/hard-disk), [video adapter](https://whatis.techtarget.com/definition/video-adapter), [keyboard](https://whatis.techtarget.com/definition/keyboard), [mouse](https://whatis.techtarget.com/definition/mouse) and printer. BIOS is an integral part of your computer and comes with it when you bring it home. (In contrast, the operating system can either be pre-installed by the manufacturer or vendor or installed by the user.) BIOS is a program that is made accessible to the microprocessor on an erasable programmable read-only memory ([EPROM](https://whatis.techtarget.com/definition/EPROM)) chip. When you turn on your computer, the microprocessor passes control to the BIOS program, which is always located at the same place on EPROM.When BIOS boots up (starts up) your computer, it first determines whether all of the attachments are in place and operational and then it loads the operating system (or key parts of it) into your computer's random access memory ([RAM](https://searchstorage.techtarget.com/definition/RAM-random-access-memory)) from your hard disk or diskette drive. With BIOS, your operating system and its applications are freed from having to understand exact details (such as hardware addresses) about the attached input/output devices. When device details change, only the BIOS program needs to be changed. Sometimes this change can be made during your system setup. In any case, neither your operating system or any applications you use need to be changed. Although BIOS is theoretically always the intermediary between the microprocessor and I/O device control information and data flow, in some cases, BIOS can arrange for data to flow directly to memory from devices (such as video cards) that require faster data flow to be effective.

**Purpose Of BIOS**

BIOS is an important part of the computer framework. It primary purpose is to initialize and test the system hardware components, as well as to load a boot loader or an operating system from memory. BIOS is usually pre-loaded on to the motherboard. It is typically loaded onto an erasable programmable read-only memory (EPROM) chip. When the computer it turned on, the BIOS is the first thing to boot up. Firstly, it determines whether all of the attachments are in place and are operational. It then loads the operating system from the hard disk to the random access memory (RAM), after which the RAM takes over.

**Booting Process**

Booting (also known as booting up) is the initial set of operations that a computer system performs when electrical power is switched on. The process begins when a computer that has been turned off is re-energized, and ends when the computer is ready to perform its normal operations. On modern general purpose computers, this can take tens of seconds and typically involves performing power-on self-test, locating and initializing peripheral devices, and then finding, loading and starting an operating system. Many computer systems also allow these operations to be initiated by a software command without cycling power, in what is known as a soft reboot, though some of the initial operations might be skipped on a soft reboot. A boot loader is a computer program that loads the main operating system or runtime environment for the computer after completion of self-tests.

The computer term boot is short for bootstrap or bootstrap load and derives from the phrase to pull oneself up by one’s bootstraps. The usage calls attention to the paradox that a computer cannot run without first loading software but some software must run before any software can be loaded. Early computers used a variety of ad-hoc methods to get a fragment of software into memory to solve this problem. The invention of integrated circuit Read-only memory (ROM) of various types solved the paradox by allowing computers to be shipped with a start up program that could not be erased, but growth in the size of ROM has allowed ever more elaborate start up procedures to be implemented.

There are numerous examples of single and multi-stage boot sequences that begin with the execution of boot program(s) stored in boot ROMs. During the booting process, the binary code of an operating system or runtime environment may be loaded from nonvolatile secondary storage (such as a hard disk drive) into volatile, or random-access memory (RAM) and then executed. Some simpler embedded systems do not require a noticeable boot sequence to begin functioning and may simply run operational programs stored in read-only memory (ROM) when turned on.

**What Is UEFI?**

The Unified Extensible Firmware Interface is a specification that defines a software interface between an operating system and platform firmware. The Unified Extensible Firmware Interface (UEFI) specification provides and defines a software interface that is between firmware and an operating system (OS). UEFI replaces BIOS, enhances the Extensible Firmware Interface (EFI) and provides an operational environment for OS and boot-time applications and services. UEFI works like BIOS, but with enhanced control, security and manageability of the system booting process. UEFI is programmable and allows for the addition of boot-time applications and services by original equipment manufacturer (OEM) developers.

The UEFI implementation of Windows 8 provides secure boot services that prevent the loading of malware into the root kit by evaluating and authenticating the certificate of each boot loader driver from the UEFI firmware that is stored on the system's motherboard. Thus, only UEFI certified applications and services can execute on boot. UEFI is also implemented directly on the OS to only authenticate operating systems that are digitally signed.

**Differentiate RAID & LVM**

**RAID**

**RAID** (**Redundant Array of Independent Disks**, originally **Redundant Array of Inexpensive Disks**) is a data [storage virtualization](https://en.wikipedia.org/wiki/Storage_virtualization) technology that combines multiple physical [disk drive](https://en.wikipedia.org/wiki/Disk_drive) components into one or more logical units for the purposes of [data redundancy](https://en.wikipedia.org/wiki/Data_redundancy), performance improvement, or both. This was in contrast to the previous concept of highly reliable mainframe disk drives referred to as "single large expensive disk.

Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the required level of [redundancy](https://en.wikipedia.org/wiki/Redundancy_(engineering)) and performance. The different schemes, or data distribution layouts, are named by the word "RAID" followed by a number, for example RAID 0 or RAID 1. Each scheme, or RAID level, provides a different balance among the key goals: [reliability](https://en.wikipedia.org/wiki/Reliability_engineering), [availability](https://en.wikipedia.org/wiki/Availability), [performance](https://en.wikipedia.org/wiki/Computer_performance), and [capacity](https://en.wikipedia.org/wiki/Computer_data_storage#Capacity). RAID levels greater than RAID 0 provide protection against unrecoverable [sector](https://en.wikipedia.org/wiki/Disk_sector) read errors, as well as against failures of whole physical drives.

**LVM**

Logical volume management (LVM) is a form of **storage** virtualization that offers system administrators a more flexible approach to managing disk **storage** space than traditional partitioning. This type of virtualization tool is located within the device-driver stack on the operating system. **Logical Volume Manager** (**LVM**) is a [device mapper](https://en.wikipedia.org/wiki/Device_mapper) target that provides [logical volume management](https://en.wikipedia.org/wiki/Logical_volume_management) for the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel). Most modern [Linux distributions](https://en.wikipedia.org/wiki/Linux_distribution) are LVM-aware to the point of being able to have their [root file systems](https://en.wikipedia.org/wiki/Root_file_system) on a [logical volume](https://en.wikipedia.org/wiki/Logical_volume).

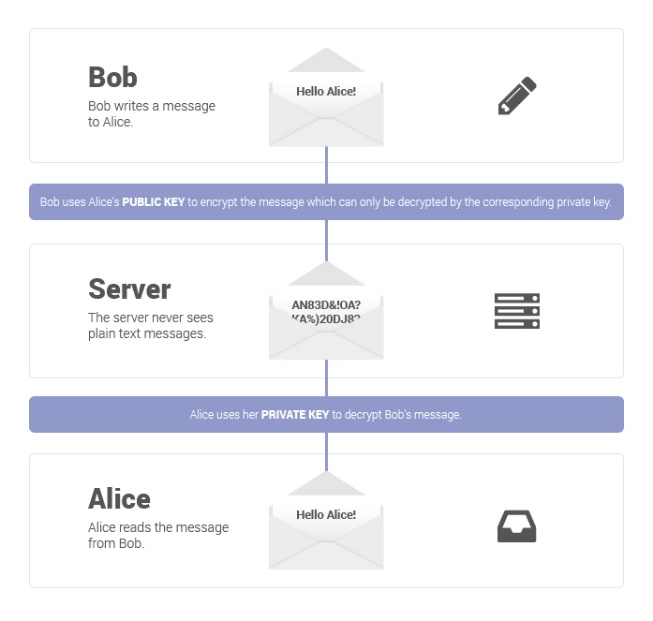
**What is end-to-end encryption (E2EE)?**

When you use E2EE to send an email or a message to someone, no one monitoring the network can see the content of your message — not hackers, not the government, and not even the company (e.g. [Proton Mail](https://protonmail.com/)) that facilitates your communication. This differs from the encryption that most companies already use, which only protects the data in transit between your device and the company’s servers. For example, when you send and receive an email using a service that does not provide E2EE, such as Gmail or Hotmail, the company has the ability to access the content of your messages because they also hold the encryption keys. **E2EE eliminates this possibility because the service provider does not actually possess the decryption key.**Because of this, E2EE is much stronger than standard encryption.

**How does end-to-end encryption work?**

To understand how E2EE works, it helps to look at a diagram. In the example below, Bob wants to say hello to Alice in private. Alice has a public key and a private key, which are two mathematically related encryption keys. The public key can be shared with anyone, but only Alice has the private key.

First, Bob uses Alice’s public key to encrypt the message, turning “Hello Alice” into something called cipher text — scrambled, seemingly random characters.

**[](https://protonmail.com/blog/wp-content/uploads/2015/05/bob-alice.jpg)**

Bob sends this encrypted message over the public internet. Along the way, it may pass through multiple servers, including those belonging to the email service they’re using and to their internet service providers. Although those companies may try to read the message (or even share them with third parties), it is impossible for them to convert the cipher text back into readable plaintext. Only Alice can do that with her private key when it lands in her inbox, as Alice is the only person that has access to her private key. When Alice wants to reply, she simply repeats the process, encrypting her message to Bob using Bob’s public key.

**Advantages of end-to-end encryption services**

There are several advantages of E2EE over the standard encryption that most services utilize:

* **It keeps your data safe from hacks.** E2EE means fewer parties have access to your unencrypted data. Even if hackers compromise the servers where your data is stored (e.g. Yahoo mail hack), they cannot decrypt your data because the does not possess the decryption keys.
* **It keeps your data private.**If you use Gmail, Google can know every intimate detail you put in your emails, and it can save your emails even if you delete them. E2EE gives you control over who reads your messages.
* **It’s good for democracy.** Everyone has the right to privacy. E2EE protects free speech and shields persecuted activists, dissidents, and journalists from intimidation.