

**IN THE UNITED STATES DISTRICT COURT  
FOR THE DISTRICT OF DELAWARE**

ANCORA TECHNOLOGIES, INC.,

*Plaintiff,*

v.

LENOVO GROUP LTD.,  
LENOVO (UNITED STATES) INC., and  
MOTOROLA MOBILITY, LLC,

*Defendants.*

C.A. No. 19-1712-CFC

**JURY TRIAL DEMANDED**

**FIRST AMENDED COMPLAINT FOR PATENT  
INFRINGEMENT**

---

Plaintiff, ANCORA TECHNOLOGIES, INC. (“Ancora”), for its first amended Complaint against Lenovo Group Ltd., Lenovo (United States) Inc. (collectively “Lenovo”), and Motorola Mobility, LLC (“Motorola”) states the following:

**I. THE PARTIES**

1. Plaintiff Ancora Technologies, Inc. is a corporation organized and existing under the laws of the State of Delaware and having a place of business at 23977 S.E. 10<sup>th</sup> Street, Sammamish, Washington 98075.

2. Upon information and belief, Lenovo Group Ltd. is a corporation organized and existing under the laws of the People’s Republic of China with its principal place of business at No. 6 Chuang Ye Road, Haidian District, Shangdi Information, Industry Base, 100085 Beijing, China.

3. Upon information and belief, Lenovo (United States) Inc. is a wholly-owned subsidiary of Lenovo Group Ltd. and is a Delaware entity with its principal place of business at 1009 Think Place, Morrisville, North Carolina 27560. Lenovo (United States) Inc. may be served

via its registered agent, The Corporation Trust Company, Corporation Trust Center, 1209 Orange St., Wilmington, Delaware, 19801.

4. Upon information and belief, Lenovo (United States) Inc. offers for sale mobile devices, such as smartphones, through its website at <https://tinyurl.com/Lenovo-phones>. Upon information and belief, Lenovo (United States) Inc. offers for sale mobile devices, such as smartphones, through an Amazon storefront website at <https://tinyurl.com/Lenovo-Amazon-Store>. Upon information and belief, Lenovo (United States) Inc. directs sales of its mobile devices to Delaware residents through at least these websites.

5. Upon information and belief, Motorola Mobility LLC is a wholly owned subsidiary of Lenovo Group Ltd. (*See e.g.*, <https://tinyurl.com/Motorola-Lenovo>). Upon information and belief, Motorola Mobility LLC is a Delaware corporation with its principal place of business at 222 West Merchandise Mart Plaza, Suite 1800, Chicago, Illinois 60654. Motorola may be served via its registered agent, The Corporation Trust Company, Corporation Trust Center, 1209 Orange St., Wilmington, Delaware, 19801.

6. Upon information and belief, Motorola Mobility LLC offers for sale mobile devices, such as smartphones, through its website at <https://www.motorola.com/us/home>. Upon information and belief, Motorola Mobility LLC offers for sale mobile devices, such as smartphones, through an Amazon storefront website at <https://tinyurl.com/Motorola-Amazon-Store>. Upon information and belief, Motorola Mobility LLC directs sales of its mobile devices to Delaware residents through at least these websites.

## II. JURISDICTION

7. This is an action for patent infringement arising under the provisions of the Patent Laws of the United States of America, Title 35, United States Code, namely 35 U.S.C. §§ 271, 281, and 284-285, among others.

8. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

9. This Court has general and specific personal jurisdiction over Lenovo and Motorola because they have committed acts within this District giving rise to this action and have established minimum contacts with this forum such that the exercise of jurisdiction over Lenovo and Motorola would not offend traditional notions of fair play and substantial justice. Lenovo and Motorola, directly and through subsidiaries and intermediaries (including distributors, retailers, franchisees and others), alter egos, and/or agents have committed and continue to commit acts of infringement in this District by, among other things, using, selling, importing, and/or offering for sale products that infringe the Asserted Patents.

10. Venue is proper in this District as to Lenovo Group Ltd. because it is a foreign entity that may be sued in any judicial district under 28 U.S.C. § 1391(c).

11. Venue is proper in this District as to Lenovo (United States) Inc. and Motorola under 28 U.S.C. §§ 1391(b)-(d) and 1400(b) because each have committed acts of infringement in Delaware and each are organized under the laws of Delaware.

## III. BACKGROUND

12. On June 25, 2002, U.S. Patent No. 6,411,941 (“the ’941 patent”) entitled “Method Of Restricting Software Operation Within A License Limitation” was duly and legally issued. (*See* Exhibit A, U.S. Patent No. 6,411,941.) A reexamination certificate also issued to the ’941 patent

on June 1, 2010 where the patentability of all claims was confirmed by the United States Patent Office. (Exhibit B, *Ex Parte* Reexamination Certificate Issued Under 35 U.S.C. § 307.)

13. The '941 patent has been involved in litigation against Microsoft Corporation, Dell Incorporated, Hewlett Packard Incorporated, and Toshiba America Information Systems. (*See* 2009-cv-00270, Western District of Washington.)

14. The '941 patent has also been involved in litigation against Apple Incorporated. (*See* 2015-cv-03659, Northern District of California.)

15. The '941 patent is currently involved in litigation against HTC America, Inc. and HTC Corporation. (*See* 2016-cv-01919, Western District of Washington.)

16. The '941 patent is currently involved in litigation against Samsung Electronics America, Inc. and Samsung Electronics Co., Ltd. (*See* 2019-cv-00385, Western District of Texas.)

17. The '941 patent is currently involved in litigation against LG Electronics USA, Inc. and LG Electronics, Inc. (*See* 2019-cv-00384, Western District of Texas.)

18. The '941 patent was involved in a Covered Business Method proceeding before the U.S. Patent and Trademark Office (*See* PTAB-CBM2017-00054). The U.S. Patent and Trademark Office denied institution of the petition filed by HTC and found the '941 patent recites a “technological improvement to problems arising in prior art software and hardware methods of restricting an unauthorized software program’s operation.” (*See* PTAB-CBM2017-00054, Paper No. 7 at pg. 9.)

19. The U.S. Court of Appeals for the Federal Circuit further issued an order on November 16, 2018 regarding the validity of the '941 patent. (*See* CAFC 18-1404, Dkt. # 39.) In this appeal, the U.S. Court of Appeals for the Federal Circuit held:

[T]he claimed invention moves a software-verification structure to a BIOS location not previously used for this computer-security purpose and alters how the function is performed (in that the BIOS memory used for verification now interacts with distinct computer memory to perform a software-verification function), yielding a tangible technological benefit (by making the claimed system less susceptible to hacking).

CAFC 18-1404, Dkt. # 39, pg. 13.

20. The U.S. Court of Appeals for the Federal Circuit further issued an order on March 3, 2014 regarding claim construction and invalidity of the '941 patent. (*See* CAFC 13-1378, Dkt. # 57.)

21. Ancora is the owner of all right, title and interest in the '941 patent.

#### **IV. COUNT I – PATENT INFRINGEMENT**

22. Ancora realleges the preceding paragraphs as though set forth fully herein.

23. Claim 1 of the '941 patent recites “a method of restricting software operation within a license for use with a computer including an erasable, non-volatile memory area of a BIOS of the computer, and a volatile memory area; the method comprising the steps of: [1] selecting a program residing in the volatile memory, [2] using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS, the verification structure accommodating data that includes at least one license record, [3] verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS, and [4] acting on the program according to the verification.”

24. As explained in detail below, Lenovo and Motorola (collectively, “Defendants”) directly infringed the '941 patent in violation of 35 U.S.C. § 271(a) by, prior to the expiration of the '941 patent, using within the United States, and without authorization, the method recited in at least Claim 1 of the '941 patent literally or under the doctrine of equivalents.

25. Defendants design at least the following smartphones to use the method recited in Claim 1: Lenovo Phab, Lenovo Phab Plus, Moto X, Moto Z, Moto G4, Moto G4 Plus, Moto G4 Play, Moto E, and Droid Turbo 2. (*see e.g.*, <https://tinyurl.com/y9njhm24>.) (collectively, “Android Devices”)

26. The Android Devices include operating system software that is transmitted by Defendants or received under Defendants’ direction using over-the-air (“OTA”) servers and hardware (“the OTA Products”) that cause the Android Devices to perform the method of Claim 1 prior to the expiration of the ’941 patent.

27. The following comparison between the limitations of Claim 1 of the ’941 patent and Defendant’s Over-the-Air update process (the “Accused Process”) used to update the Android Devices establishes Defendants’ infringement of the ’941 patent.

28. For instance, on information and belief, the Accused Process allowed one of the Android Devices (i.e., the Moto Z Play) to be updated to the Android 7.0 Nougat software on March 7, 2017. (*see, e.g.*, <https://www.androidauthority.com/android-7-0-update-679175/> and <https://www.androidauthority.com/moto-z-play-update-747977/>.)

**“A method of restricting software operation within a license for use with a computer including an erasable, non-volatile memory area of a BIOS of the computer, and a volatile memory area;”**

29. The Accused Process is a method of restricting software operation within a license because, if the “Verified Boot” aspect of the Accused Process fails, the OTA update will not complete and the updated software will not execute.

**Verified Boot**

Verified Boot strives to ensure all executed code comes from a trusted source (usually device OEMs), rather than from an attacker or corruption. It establishes a full chain of trust, starting from a hardware-protected root of trust to the bootloader, to the boot partition and other verified partitions including `system`, `vendor`, and optionally `oem` partitions. During device boot up, each stage verifies the integrity and authenticity of the next stage before handing over execution.

In addition to ensuring that devices are running a safe version of Android, Verified Boot check for the correct version of Android with [rollback protection](#). Rollback protection helps to prevent a possible exploit from becoming persistent by ensuring devices only update to newer versions of Android.

In addition to verifying the OS, Verified Boot also allows Android devices to communicate their state of integrity to the user.

<https://source.android.com/security/verifiedboot>

30. If the operating system update image is not cryptographically signed with the expected cryptographic keys, the update process Defendants use will reject the update:

**Signing Builds for Release**

Android OS images use cryptographic signatures in two places:

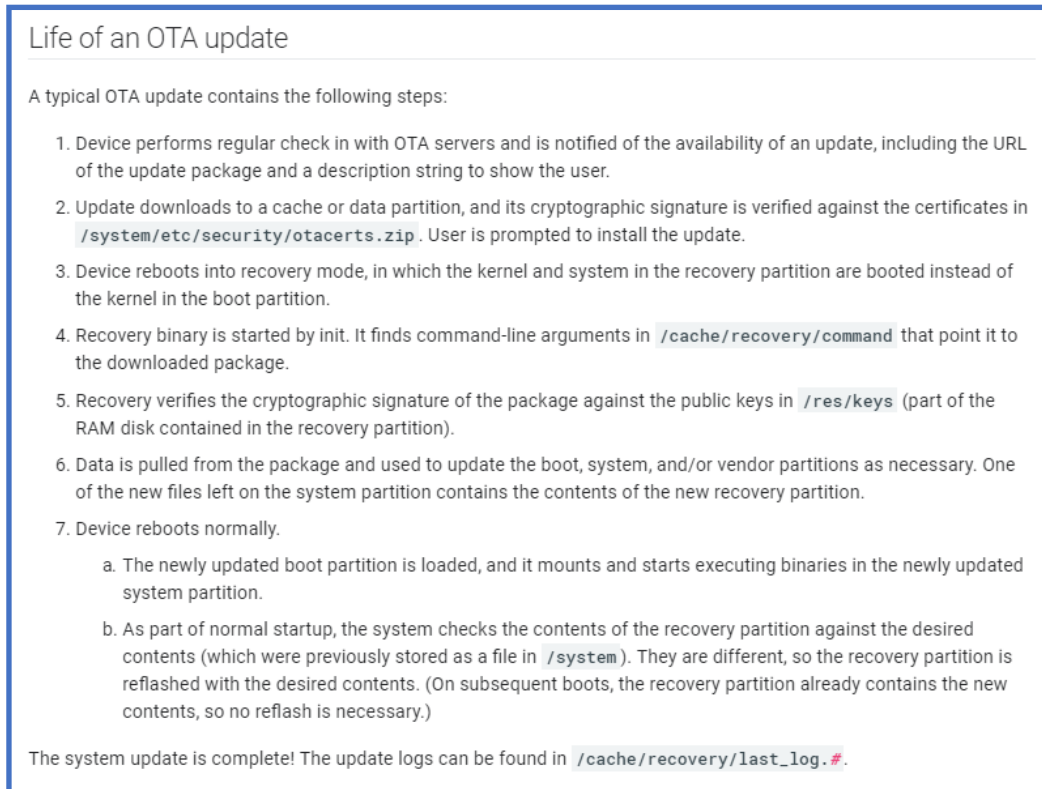
1. Each `.apk` file inside the image must be signed. Android's Package Manager uses an `.apk` signature in two ways:
  - When an application is replaced, it must be signed by the same key as the old application in order to get access to the old application's data. This holds true both for updating user apps by overwriting the `.apk`, and for overriding a system app with a newer version installed under `/data`.
  - If two or more applications want to share a user ID (so they can share data, etc.), they must be signed with the same key.
2. OTA update packages must be signed with one of the keys expected by the system or the installation process will reject them.

[https://source.android.com/devices/tech/ota/sign\\_builds](https://source.android.com/devices/tech/ota/sign_builds)

31. Each Android Device used with the Accused Process includes a computer having a non-volatile memory area of a BIOS (also referred to as Unified Extensible Firmware Interface (UEFI)) in the form of ROM or Flash memory (also described as “RAM disk”) and volatile memory in the form of RAM memory. The BIOS included within each Android Device comprises data that is maintained when the power is removed and contains the set of essential startup operations that run when a computer is turned on, which tests hardware, starts the operating system, and supports the transfer of data among hardware devices of the computer.

**“the method comprising the steps of: selecting a program residing in the volatile memory,”**

32. The Accused Process loads at least a portion of the updated operating system program image into the Android Device’s RAM (volatile memory) and selects the program for execution.



<https://source.android.com/devices/tech/ota/nonab>

**“using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS, the verification structure accommodating data that includes at least one license record,”**

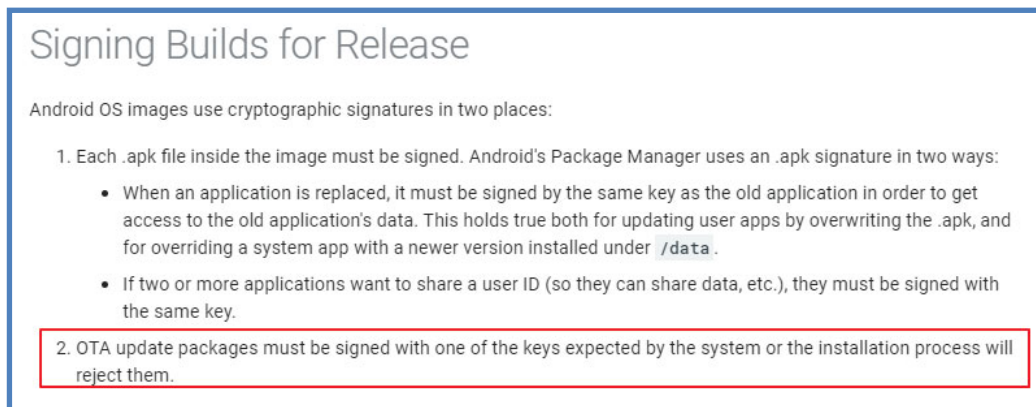
33. The Accused Process uses an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS of the Android Devices. For example, Defendants implement an OTA Install program or subroutine that provides to the Android Devices an OTA update containing a verification structure. The OTA Install program or subroutine also stores a



verification structure within a partition (*e.g.*, the “cache” or “A/B” partitions) of the erasable, non-volatile memory of the Android Device’s BIOS.

34. For instance, on information and belief, the OTA Process employs a daemon program called “update\_engine” that checks for an OTA update and will download a full OTA package and verify the OTA Process is completed. (*see, e.g.*, <https://source.android.com/devices/tech/ota/ab>.) On further information and belief, the OTA Process may initiate a subroutine called “FullOTA\_InstallBegin” to perform the installation of the OTA package. (*see, e.g.*, [https://source.android.com/devices/tech/ota/nonab/device\\_code](https://source.android.com/devices/tech/ota/nonab/device_code).)

35. The verification structure includes data accommodating at least one license record. Examples of such a license record Defendants use in the Accused Process include a cryptographic signature or key:



The image is a screenshot of a document titled "Signing Builds for Release". The text explains that Android OS images use cryptographic signatures in two places. It lists two points: 1. Each .apk file inside the image must be signed, and 2. OTA update packages must be signed with one of the keys expected by the system or the installation process will reject them. The second point is highlighted with a red rectangular box.

**Signing Builds for Release**

Android OS images use cryptographic signatures in two places:

1. Each .apk file inside the image must be signed. Android's Package Manager uses an .apk signature in two ways:
  - When an application is replaced, it must be signed by the same key as the old application in order to get access to the old application's data. This holds true both for updating user apps by overwriting the .apk, and for overriding a system app with a newer version installed under `/data`.
  - If two or more applications want to share a user ID (so they can share data, etc.), they must be signed with the same key.
2. OTA update packages must be signed with one of the keys expected by the system or the installation process will reject them.

[https://source.android.com/devices/tech/ota/sign\\_builds](https://source.android.com/devices/tech/ota/sign_builds)

## Verifying Boot

Verified boot requires cryptographically verifying all executable code and data that is part of the Android version being booted before it is used. This includes the kernel (loaded from the `boot` partition), the device tree (loaded from the `dtbo` partition), `system` partition, `vendor` partition, and so on.

Small partitions, such as `boot` and `dtbo`, that are read only once are typically verified by loading the entire contents into memory and then calculating its hash. This calculated hash value is then compared to the *expected hash value*. If the value doesn't match, Android won't load. For more details, see [Boot Flow](#).

Larger partitions that won't fit into memory (such as, file systems) may use a hash tree where verification is a continuous process happening as data is loaded into memory. In this case, the root hash of the hash tree is calculated during run time and is checked against the *expected root hash value*. Android includes the `dm-verity` driver to verify larger partitions. If at some point the calculated root hash doesn't match the *expected root hash value*, the data is not used and Android enters an error state. For more details, see [dm-verity corruption](#).

The *expected hashes* are typically stored at either the end or beginning of each verified partition, in a dedicated partition, or both. Crucially, these hashes are signed (either directly or indirectly) by the root of trust. As an example, the AVB implementation supports both approaches, see [Android Verified Boot](#) for details.

<https://source.android.com/security/verifiedboot/verified-boot>.

### **“verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS, and”**

36. Defendants use the Accused Process to confirm whether the operating system update is licensed using at least the verification structure from the erasable, non-volatile memory of the Android Device's BIOS. For instance, once the verification structure has been set up in the BIOS, Defendants use the Accused Process to reboot into recovery mode, load the OTA update into its volatile memory (e.g., RAM), and use the at least one license record from the BIOS to verify the OTA update (e.g., Step 5 below):

## Life of an OTA update

A typical OTA update contains the following steps:

1. Device performs regular check in with OTA servers and is notified of the availability of an update, including the URL of the update package and a description string to show the user.
2. Update downloads to a cache or data partition, and its cryptographic signature is verified against the certificates in `/system/etc/security/otacerts.zip`. User is prompted to install the update.
3. Device reboots into recovery mode, in which the kernel and system in the recovery partition are booted instead of the kernel in the boot partition.
4. Recovery binary is started by init. It finds command-line arguments in `/cache/recovery/command` that point it to the downloaded package.
5. Recovery verifies the cryptographic signature of the package against the public keys in `/res/keys` (part of the RAM disk contained in the recovery partition).
6. Data is pulled from the package and used to update the boot, system, and/or vendor partitions as necessary. One of the new files left on the system partition contains the contents of the new recovery partition.
7. Device reboots normally.
  - a. The newly updated boot partition is loaded, and it mounts and starts executing binaries in the newly updated system partition.
  - b. As part of normal startup, the system checks the contents of the recovery partition against the desired contents (which were previously stored as a file in `/system`). They are different, so the recovery partition is reflashed with the desired contents. (On subsequent boots, the recovery partition already contains the new contents, so no reflash is necessary.)

The system update is complete! The update logs can be found in `/cache/recovery/last_log.#`.

<https://source.android.com/devices/tech/ota/nonab>

### **“acting on the program according to the verification.”**

37. The Accused Process acts on the program (the operating system update) according to the verification. If the OTA update is verified, the Accused Process will load and execute the update (e.g., Steps 6 and 7 below):

## Life of an OTA update

A typical OTA update contains the following steps:

1. Device performs regular check in with OTA servers and is notified of the availability of an update, including the URL of the update package and a description string to show the user.
2. Update downloads to a cache or data partition, and its cryptographic signature is verified against the certificates in `/system/etc/security/otacerts.zip`. User is prompted to install the update.
3. Device reboots into recovery mode, in which the kernel and system in the recovery partition are booted instead of the kernel in the boot partition.
4. Recovery binary is started by init. It finds command-line arguments in `/cache/recovery/command` that point it to the downloaded package.
5. Recovery verifies the cryptographic signature of the package against the public keys in `/res/keys` (part of the RAM disk contained in the recovery partition).
6. Data is pulled from the package and used to update the boot, system, and/or vendor partitions as necessary. One of the new files left on the system partition contains the contents of the new recovery partition.
7. Device reboots normally.
  - a. The newly updated boot partition is loaded, and it mounts and starts executing binaries in the newly updated system partition.
  - b. As part of normal startup, the system checks the contents of the recovery partition against the desired contents (which were previously stored as a file in `/system`). They are different, so the recovery partition is reflashed with the desired contents. (On subsequent boots, the recovery partition already contains the new contents, so no reflash is necessary.)

The system update is complete! The update logs can be found in `/cache/recovery/last_log.#`.

<https://source.android.com/devices/tech/ota/nonab>

38. If the verification fails, however, the update is rejected:

## Signing Builds for Release

Android OS images use cryptographic signatures in two places:

1. Each `.apk` file inside the image must be signed. Android's Package Manager uses an `.apk` signature in two ways:
  - When an application is replaced, it must be signed by the same key as the old application in order to get access to the old application's data. This holds true both for updating user apps by overwriting the `.apk`, and for overriding a system app with a newer version installed under `/data`.
  - If two or more applications want to share a user ID (so they can share data, etc.), they must be signed with the same key.
2. OTA update packages must be signed with one of the keys expected by the system or the installation process will reject them.

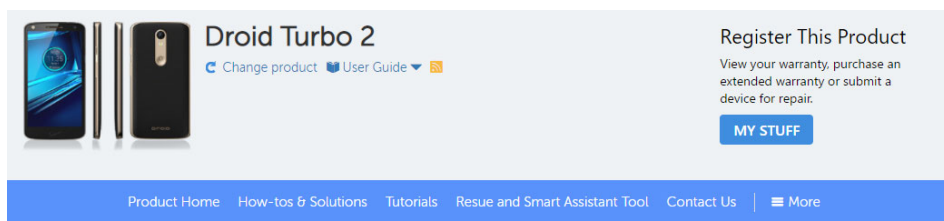
[https://source.android.com/devices/tech/ota/sign\\_builds](https://source.android.com/devices/tech/ota/sign_builds)

39. Defendants use and control the use of the Accused Process to perform OTA software updates on the Android Devices, practicing each limitation of Claim 1 as described above. Defendants directly infringed Claim 1 of the '941 patent by using the Accused Process with Android Devices by itself.

40. Once Defendants have set up the verification structure by transmitting to a device an OTA update, the Accused Process is configured to automatically perform each of the remaining Claim 1 steps.

41. In addition, Ancora alleges that Defendants jointly infringe the '941 patent with Android Device owners in the United States being responsible as a single entity as set forth below.

42. Defendants condition participation in the Accused Process and the receipt of the benefit of a software update on the performance of each of the above steps. For instance, Defendants condition participation by customers in using the Accused Process in order to gain access to new or upgraded Android operating systems.



## Software updates - moto g7

[Back to Previous Page](#)

How can I check for software updates on the moto g7?

Software updates will be sent to your phone automatically over-the-air as they become available.

You can manually check for updates using the instructions below.

Go to **Settings > System > Advanced > System Updates**

If an update is available, follow the instructions on your phone to download and install it.

Once updated, your phone will restart to complete the installation.

**To find the Android version on your phone:**

Go to **Settings > About phone > Android version**

### Timing for Android software updates

There are two types of updates:

- **Security updates**, which contain fixes and improvements from Google for your current version of the Android operating system. Motorola provides these updates to most phones on a regular basis.
- **Android OS updates**, which are new versions of the Android operating system. Motorola provides OS updates from Google to eligible phones as soon as possible.

To see if an OS update will be available for your phone, or if your phone is still eligible for security updates, visit our [software update website](#). Our support agents get their Android update information from here too. If the site doesn't have a update release date for your phone, then we don't know the release date yet.

For Android OS updates, with several phone models, regions, and distribution channels we may have hundreds of software versions to test before releasing an update to your phone. Because dependencies on carriers and other key partners for certifications, independent testing, and requests for changes take more time, all phone owners don't receive updates at the same time, even in the same region.

<https://support.motorola.com/us/en/Solution/MS135865>.)

43. Defendants take steps to ensure that the Accused Process cannot install an OTA update except by performing each of the above described steps.

44. Defendants emphasize the benefits associated with updating the software using the Accused Process. For instance, Defendants have stated:

There are two types of updates:

**Security updates**, which contain fixes and improvements from Google for your current version of the Android operating system. Motorola provides these updates to most phones on a regular basis.

**Android OS updates**, which are new versions of the Android operating system. Motorola provides OS updates from Google to eligible phones as soon as possible.

(<https://support.motorola.com/us/en/Solution/MS135865>.)

45. Defendants control the manner of the performance of the Accused Process. As set forth above, Defendants configured each Android Product such that, upon receiving an OTA update, it would automatically perform each remaining step of the Accused Process. For example, using the Accused Process, Defendants may require immediate installation of the OTA updates onto the Android Devices. (<https://source.android.com/devices/tech/admin/ota-updates>.) Or using the Accused Process, Defendants may allow the customer to postpone installation of the OTA update for a specified period.

46. Defendants controlled the timing of the performance of the Accused Process by determining when to utilize the Accused Process to set up a verification structure in the Android Devices.

47. Defendants had the right and ability to stop or limit infringement by not using the Accused Process but failed to do so.

48. Defendants' infringement has caused damage to Ancora, and Ancora is entitled to recover from Defendants those damages Ancora has sustained as a result of Defendant's infringement.

**V. DEMAND FOR RELIEF**

WHEREFORE, Plaintiff respectfully requests that this Court enter judgment against Lenovo and Motorola as follows:

A. Declaring that Lenovo and Motorola have infringed United States Patent No. 6,411,941 in violation of 35 U.S.C. § 271;

B. Awarding damages to Ancora arising out of this infringement, including enhanced damages pursuant to 35 U.S.C. § 284 and prejudgment and post-judgment interest, in an amount according to proof;

C. Awarding Ancora its costs and expenses in this action;

D. Declaring that this case is exceptional, and that Ancora is entitled to its reasonable attorneys' fees pursuant to 35 U.S.C. § 285; and

E. Awarding such other and further relief the Court deems just and proper, including any relief that the Court may deem appropriate under 35 U.S.C. § 285.

**VI. DEMAND FOR JURY TRIAL**

Ancora respectfully demands a trial by jury of any and all issues triable of right by a jury.

Dated: September 8, 2020

SMITH, KATZENSTEIN & JENKINS, LLP

OF COUNSEL:

BROOKS KUSHMAN P.C.

John P. Rondini

Mark A Cantor

John S. LeRoy

Marc Lorelli

1000 Town Center, 22nd Floor

Southfield, MI 48075-1238

(248) 358-4400

jrondini@brookskushman.com

mcantor@brookskushman.com

jleroy@brookskushman.com

mlorelli@brookskushman.com

/s/ Eve H. Ormerod

Neal C. Belgam (No. 2721)

Eve H. Ormerod (No. 5369)

1000 West Street, Suite 150

Wilmington, DE 19801

302.652.8400

nbelgam@skjlaw.com

eormerod@skjlaw.com

*Attorneys for Plaintiff*

*Ancora Technologies, Inc.*