

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NETSKOPE, INC.,
Petitioner,

v.

FORTINET, INC.,
Patent Owner.

Case No. IPR2023-00458
U.S. Patent No. 9,280,678

PATENT OWNER'S RESPONSE

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION	1
II. OVERVIEW OF THE '678 PATENT	3
III. OVERVIEW OF CERTAIN CITED ART	9
B. Ground 1 Cited Art.....	9
1. Cidon (Ex. 1004).....	9
2. Shikfa (Ex. 1006).....	10
3. Herrmann (Ex. 1005)	11
C. Ground 4 Cited Art.....	12
1. Auradkar (Ex. 1009)	12
2. Chiueh (Ex. 1011).....	14
3. Chambers (Ex. 1010)	15
IV. LEVEL OF ORDINARY SKILL IN THE ART.....	16
V. CLAIM CONSTRUCTION	17
VI. GROUND 1: PETITIONER FAILS TO ESTABLISH THAT CIDON, SHIKFA, AND HERRMANN RENDER CLAIMS 1- 3, 8-14, AND 16-27 OBVIOUS	21
A. Claims 1/16: Cidon In View of Shikfa Does Not Render Obvious “Creating Namespaces” or “Configuring The Namespaces ... Such That Content of The File is Encrypted in a Manner That Makes it Searchable”	21
1. Creating an Identifier is Not Creating a Namespace	23
2. Shikfa’s Mapping of Keywords to Identifiers In an Index Is Not “Configuring The Namespaces”	25

3.	Shikfa’s Mapping of Keywords to Identifiers in an Index Has No Bearing on the Manner in Which a File is Encrypted	27
4.	A POSITA Would Not Have Been Motivated To Combine Cidon And Hermann With Shikfa.....	28
B.	Claims 1/16: Cidon Does Not Disclose or Render Obvious a “Policy for Managing Access to and Processing a File to Be Stored on Cloud Platforms”	31
1.	Cidon’s Encryption Policy Is Not Assigned to a User	31
2.	Cidon’s “Encryption Policy” Does Not Govern “Processing of a File to Be Uploaded on a Cloud”	35
3.	Cidon’s “Encryption Policy” Does Not Govern Access Management	35
C.	Claims 1/16: Cidon Does Not Disclose or Render Obvious “Storing, by the Gateway Device” (Claim 1) or “a Storage Module” For Storing (Claim 16) “The Searchable Encrypted File on The One or More Cloud Platforms Based on the Policy”	39
D.	Claims 11/25: Cidon Does Not Disclose or Render Obvious Storing a File “Across Two Or More Cloud Platforms”	41
E.	Claim 14: Cidon Does Not Disclose or Render Obvious a Policy That Defines “One or More Authorized Locations to Which the File May be Stored”	42
VII.	GROUND 2 AND 3 FAIL ON THE MERITS.....	43
VIII.	GROUND 4: PETITIONER FAILS TO ESTABLISH THAT AURADKAR, CHIUUEH, AND CHAMBERS RENDER CLAIMS 1-3, 8-19, AND 21-27 OBVIOUS	44

A.	Claims 1/16: Auradkar in View of Chiueh Does Not Render Obvious “Creating Namespaces” or “Configuring The Namespaces ... Such That Content of The File is Encrypted in a Manner That Makes it Searchable”	44
1.	Creating an Identifier is Not Creating a Namespace	45
2.	Chiueh’s Mapping of Keywords to Document Identifiers in an Index Is Not “Configuring The Namespaces”	47
3.	Chiueh’s Mapping of Keywords to Identifiers in an Index Has No Bearing On the Manner in Which a File is Encrypted	49
4.	A POSITA Would Not Have Been Motivated to Combine Auradkar and Chambers and Chiueh ...	50
B.	Claims 1/16: Petitioner Fails to Show Auradkar Discloses “Cryptographic Key Information Defined by the Policy”	53
C.	Claims 1/16: Petitioner Fails to Show Auradkar Discloses a Policy for “Processing Of a File to be Uploaded on a Cloud”	54
IX.	GROUND 5 FAILS ON THE MERITS	55
X.	CONCLUSION	57

EXHIBIT LIST

No.	Exhibit
2001	Declaration of John Black, Jr., Ph.D.
2002	<i>Curriculum vitae</i> of John Black, Jr., Ph.D.
2003	Supplemental Declaration of John Black, Jr., Ph.D.
2004	Transcript of the Testimony of Dr. Wenke Lee (December 13, 2023)

I. INTRODUCTION

The Petition fails to meet its burden to establish unpatentability for several reasons. **First**, none of the references disclose the claimed “creating” and “configuring” of “namespaces,” even under Petitioner’s own definition of “namespace” as “the set of names available for naming things” in a particular context. Instead, Petitioner relies on the references’ teachings of a system that assigns an identifier (such as a filename or a hash) to a file or file chunk. An identifier does not constitute a namespace, and generating and assigning an identifier does not constitute **creating** and **configuring** the set of available names. By contrast, the ’678 patent introduces a novel process for encrypting a file where a namespace is created and configured for each file by, for example, creating a folder with the name of the file’s encrypted filename, and storing the file’s chunks in the folder where each file chunk is named with pertinent, encrypted data, such as encrypted keywords found within the file. As a result of this dynamically-generated namespace, the file chunks can be stored on any cloud storage service (*e.g.*, Dropbox) and searched by using the service’s built-in file search—unlike in the prior art, such files are portable and do not require any additional metadata to be searchable.

Second, for both of Petitioner’s primary references—Cidon and Auradkar—Petitioner does not point to a single, user-assigned policy that

performs **all** of the functions of a policy as recited in the claims, including “managing access to and processing a file” and defining “cryptographic key information.” As the Federal Circuit has recently held, “it does not suffice to have multiple [elements] able to perform just one of the recited functions; the claim language requires at least one [element] capable of performing each of the recited functions.” *Salazar v. AT&T Mobility LLC*, 64 F.4th 1311, 1318 (Fed. Cir. 2023). Moreover, the encryption policies that Petitioner relies on are policies that are **assigned to files** and not assigned to users, as claimed.

Third, the policies that Petitioner relies on are not policies for processing a file “**to be** uploaded” but are instead policies for processing a file that has already been uploaded.

Fourth, all grounds rely on at least three different prior art references combined in a way that was clearly driven by hindsight. As a result, Petitioner’s asserted motivations to combine generally lack detailed explanation and fail to consider the full scope of the references. For example, the Petition ignores the express teachings in Shikfa or Chiueh that encryption must occur on a user device because outside devices are untrustworthy, which establishes that a POSITA would lack motivation to combine them with the server-based encryption of Cidon and Auradkar.

For all these reasons, as well as others discussed below, Petitioner fails to meet its burden and the Board should uphold the challenged claims.

II. OVERVIEW OF THE '678 PATENT

The '678 patent provides a new vendor-independent “secure cloud storage” solution, introducing novel systems and methods for “implementing a policy based framework for encrypting, storing, accessing, querying and managing data across one or more cloud platforms” (e.g., Dropbox, Google Drive, Amazon Web Services, etc.). Ex. 1001, 3:23-28, 6:9-14; Ex. 2003, ¶¶29-30. Referring to the example in Figure 2 below, annotated by Dr. Black, the system operates as an intermediary “Gateway” [204] that facilitates “interactions between enterprise users [202a, 202b, 202c] and cloud service providers [206a, 206b, 206c] for storage, retrieval, and searching of data and content stored by the providers.” Ex. 1001, 8:27-9:4; Ex. 2003, ¶¶31-32.

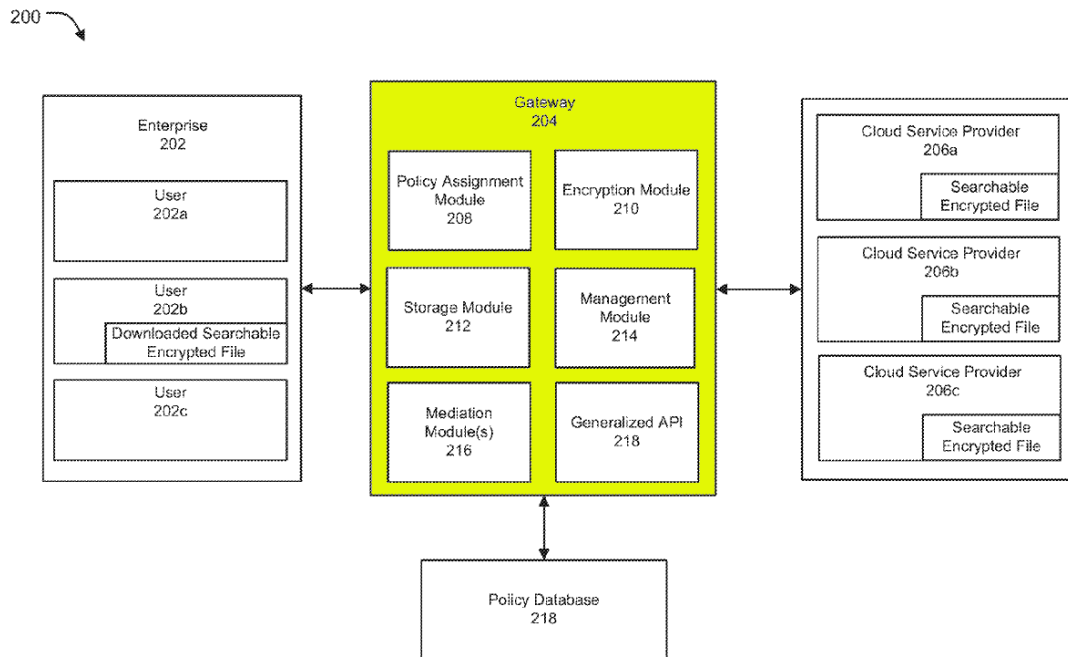


Fig. 2

Unlike prior art cloud storage systems that suffered from “rigid and non-flexible” interactions between enterprises and individual cloud service providers, the ’678 patent’s intermediary Gateway enables enterprises to more easily “store data across multiple cloud platforms to ensure more security, redundancy, and reduction in potential threat of data compromise,” and to implement global file storage policies that “allow storage of data across different service providers, allow different cloud storage access rights across users and computing devices, [and] allow searching of downloaded encrypted data across cloud service vendors.” Ex. 1001, 1:38-2:30. To this end, the Gateway may assign users a policy that “not only defines the manner in which the users can access and process content stored on the cloud, but can also

configure the mode in which the data is encrypted, stored, searched, and accessed to ensure secure and vendor independent cloud management.” *Id.*, 3:28-67; Ex. 2003, ¶33.

Because storing files on cloud platforms in an “unencrypted format” may expose them to “network attack[s] that may lead to the [files] being compromised” (Ex. 1001, 1:52-54), the Gateway can also “encrypt one or more files to be uploaded/stored” using “cryptographic key information” defined by the assigned policy. *Id.*, 4:1-12. But recognizing that conventional encryption techniques also “make[] the files ... unsearchable and hence unfriendly for user access” (*id.*, 1:54-57), the Gateway can produce a “searchable encrypted file” via a novel process that involves “dividing the file into [] chunks; creating namespaces for one or more of the [] chunks; and configuring the namespaces of the one or more chunks such that content of the file is encrypted in a manner that makes it searchable.” *Id.*, 19:62-20:3; Ex. 2003, ¶34.

In prior art systems, such as those described in the Shikfa and Chiueh references that the Petition relies on, searches could be performed on encrypted files by first extracting the keywords from the unencrypted file and storing them in an index (*e.g.* a mapping of keywords to file identifiers), and then encrypting both the file and the index using known encryption

techniques. *See, e.g.*, Ex. 1006, ¶29 (“The searchable encryption mechanism used to encrypt the indexes 24 may be any searchable encryption mechanism known in the art”); Ex. 1011, ¶40; Ex. 2003, ¶¶35-36. The encrypted file could then be stored in a cloud storage service (*e.g.* Dropbox), and searched by querying the encrypted index running on a user device. However, this approach requires that queries be run through the encrypted index, which has several problems: (1) the user device hosting the index may not be powerful enough to perform large, complex queries, and (2) only the device that has the index can perform queries, which reduces portability and reliability. Ex. 2003, ¶36. In other words, the encrypted file itself is not searchable—searches are performed through an index. *Id.*

The ’678 Patent introduces a novel process for generating searchable encrypted files that are portable and advantageously make use of the more powerful searching capabilities of the cloud service. Ex. 2003, ¶37. Specifically, “indices, also referred to as keywords hereinafter, which would typically be used as search terms in the form of search queries are stored as ***filenames*** of the chunks.” Ex. 1001, 12:7-9.

For example, as shown in Figure 4A, Container 1 (a folder in the cloud service) contains four file chunks (files in the cloud service), with “[t]he name of the folder ... the encrypted version of the original file name” (such as

foo.txt) and “the names of the files may include one or more searchable indices,” such as the fourth file “uetywo” being the encrypted keyword “Ronn.” *Id.*, 12:58-13:35. Thus, when a user wants to search for “Ronn,” the gateway device need merely encrypt the word “Ronn” to generate the cipher text “uetywo,” and then use the cloud storage service’s built-in search functionality to search for the file “uetywo.” Ex. 2003, ¶38; Ex. 1001, 10:17-19 (“Access to such searchable encrypted files can allow the users to search through the content of the files within the context of the cloud service providers 206”).

The Rozier paper, which the specification incorporates by reference as “Appendix A” (*see* Ex. 1001, 11:31-36) explicitly explains that the system “makes use of the most basic functionality that almost all of the existing cloud providers provide,” *i.e.* “the file-name search functionality.” Ex. 1003, 40. This novel process provides multiple benefits over the prior art encrypted indices, such as (1) search being performed by the powerful hardware of the cloud service instead of the gateway device or a user device, (2) search not being limited to a device that hosts the index—any device can perform the query, as the keywords are encrypted using a public key. Ex. 1001, 11:63-67; Ex. 2003, ¶39.

Moreover, as Dr. Black explains, implementing this searchable encryption process at the intermediary Gateway provides other important advantages over prior art systems. Ex. 2001, ¶40. For example, some prior art systems relied on the individual client device to encrypt files before storing them on the cloud platform, thereby inhibiting the enterprise from implementing policies that allow other authorized client devices and users in the enterprise to search for and access those encrypted files. In contrast, the '678 patent's Gateway readily provides these capabilities. *See, e.g.*, Ex. 1001, 17:21-18:37. Other prior art systems relied on the cloud service provider to implement file encryption, thereby risking that the files could be “exposed to attack within the cloud, insider jobs and subpoena—all without the knowledge or consent of the data owner.” *Id.*, 1:59-62. In contrast, the '678 patent's Gateway can implement the patent's searchable encryption process “independent of the vendor/service provider of the cloud platforms.” *Id.*, 10:1-6; Ex. 2001, ¶40.

As Dr. Black explains, the combination of features in the '678 patent provides a more secure, flexible, and user-friendly cloud storage solution than what existed before. Ex. 2001, ¶¶41-48.

III. OVERVIEW OF CERTAIN CITED ART

The petition challenges the '678 patent's independent claims 1 and 16 on two grounds. Ex. 2003, ¶49. Ground 1 asserts that the combination of Cidon, Shikfa, and Herrmann renders the independent claims obvious. Ground 4 asserts that the combination of Auradkar, Chiueh, and Chambers also renders the independent claims obvious. Below are short summaries of these six references.

B. Ground 1 Cited Art

1. Cidon (Ex. 1004)

Cidon is directed to systems and methods for encrypting files stored on cloud-based storage services. Ex. 1004, ¶10. Cidon's file encryption methods are implemented by a "management server." Ex. 1004, ¶102, Fig. 2. The management server includes an "IT Policy Enforcement Module" that enforces user "access policies" controlling user access to secure files. *Id.*, ¶108; Ex. 2003, ¶50.

Separately, Cidon discloses placement policies, which can be "defined on documents which are ancestors of other documents." Ex. 1004, ¶207. Examples of placement policies include the "encryption policy," defined by the file ancestry tree (*i.e.*, "a tree of files where the oldest file is at the top of

the tree”). *Id.*, ¶191, 206-212. Under an encryption policy, all descendent files of an encrypted parent must be encrypted. *Id.*, ¶227; Ex. 2003, ¶51.

2. Shikfa (Ex. 1006)

Shikfa is directed to a “system and method for enabling searchable encryption of encrypted documents stored by a client on one or more storage providers.” Ex. 1006, Abstract. Shikfa’s system includes a “broker server” that “transfer[s] the encrypted documents between the client and the one or more storage providers[,] maintain[s] information indicating where the encrypted documents are transferred[,] [and] stores information for at least one encrypted index for the encrypted documents.” *Id.* However, Shikfa teaches that it is not this broker server, but the **client** that “encrypts the one or more documents” and “generate[s] [the] one or more encrypted indexes.” *Id.*, ¶29. As Shikfa explains, this “ensures that the broker server 12 cannot access and/or discover the content of the client's data because the broker server 12 does not know the encryption used on the encrypted documents.” *Id.*, ¶48. As Dr. Black explains, Shikfa teaches that this arrangement is important for “maintaining the confidentiality of data external to the client.” Shikfa, ¶48; Ex. 2003, ¶52.

Shikfa further discloses that the client may encrypt documents and generate encrypted indexes using any “encryption mechanism known in the

art.” Shikfa, ¶29. However, as Dr. Black explains, Shikfa does **not** disclose performing an encryption process that divides the documents into chunks and creates and configures namespaces for one or more of the chunks. Ex. 2003, ¶53.

3. Herrmann (Ex. 1005)

Herrmann describes a system and methods for “anti-virus cooperative enforcement.” Ex. 1005, Abstract, ¶¶48-50. In Herrmann’s system, “[b]efore ... a client computer[] is permitted to join a network or access protected resources, a check is made to evaluate the antivirus engine and virus definitions that are installed and in use on [the client computer].” *Id.*, ¶50; Ex. 2003, ¶54.

As an example, “when a client computer requests a VPN connection to a gateway server, the system ... determines whether or not a client computer is using an approved anti-virus application.” Ex. 1005, ¶50. The system makes this determination via a series of interactions between the client computer and a separate “integrity server.” *Id.*, ¶29; *see also id.*, Figs. 3-4. Then, “[a]fter the integrity server 370 has evaluated compliance with anti-virus policies or rules, it notifies the gateway server about whether to permit or restrict access by this particular client device,” and “[t]he gateway server may then decide how much access to permit by this particular client device

(i.e., the gateway server 350 assigns an access policy to the connection/session between the gateway server 350 and the client computer 310) based upon the notification.” *Id.*, ¶77; Ex. 2003, ¶55.

However, as Dr. Black explains, Herrmann does **not** disclose using its anti-virus cooperative enforcement system in connection with encrypting, storing, or controlling access to files stored on the cloud. It does not reference cloud storage at all. Ex. 2003, ¶56.

C. Ground 4 Cited Art

1. Auradkar (Ex. 1009)

Auradkar is directed to a comprehensive “trustworthy platform” in which “data services, including mathematical transformation techniques, such as searchable encryption, deassembling/reassembling or distribution techniques, for data, are provided in a way that distributes trust across multiple entities ... and decouples data protection requirements ... from the containers that hold the data (e.g., file systems, databases, etc.) enabling the data to act as its own custodian.” Ex. 1009, ¶¶9-10. Auradkar explains that Figure 1, below, depicts “a general environment for providing one or more embodiments of secure, private and selectively accessible network data services as described herein.” *Id.*, ¶110; Ex. 2003, ¶57.

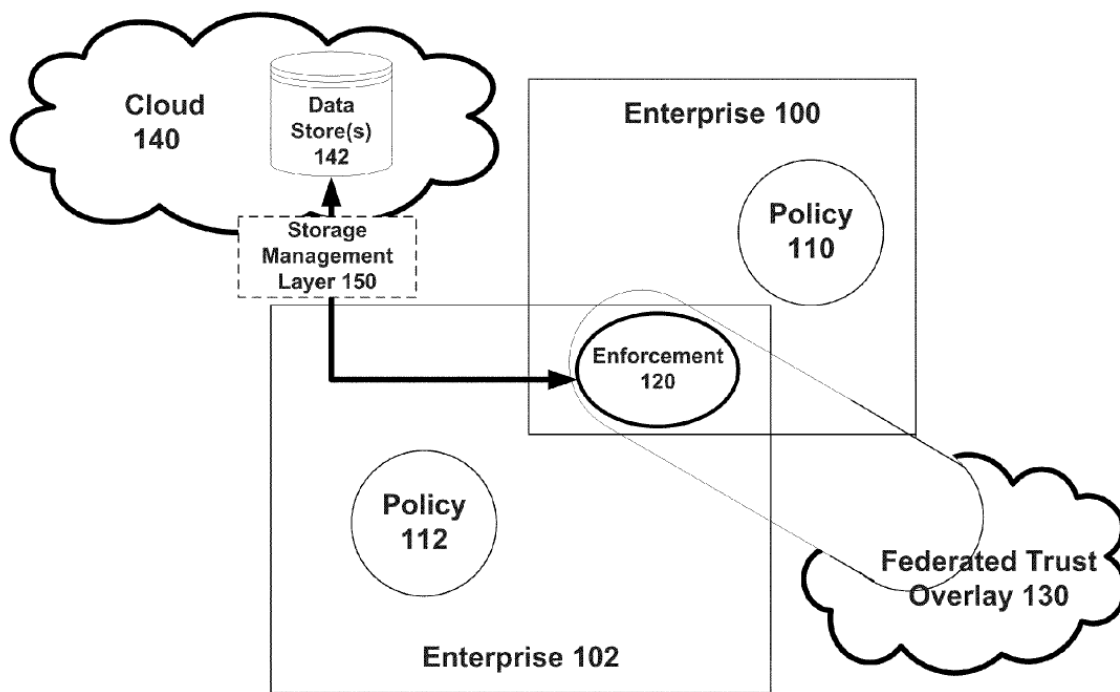
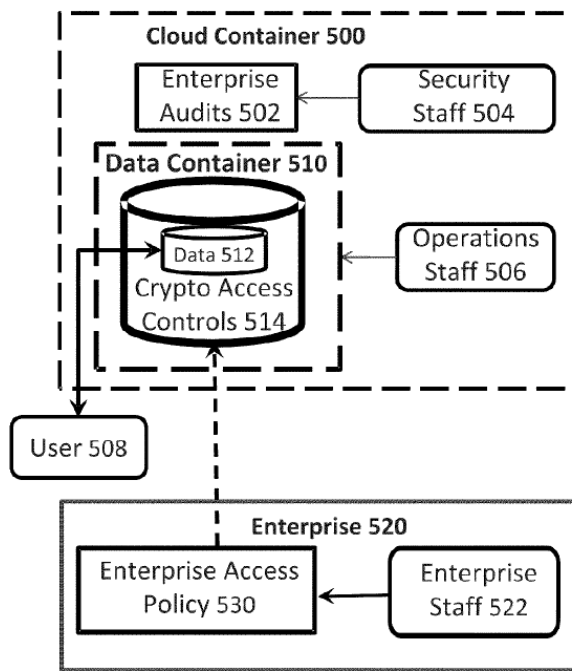


FIG. 1

Auradkar further explains that Figure 5, below, illustrates “aspects of data acting as its own custodian,” wherein “an enterprise 520 has enterprise staff 522 that can are subject to enterprise access policy 530 ... [that] can affect how data 512 stored in a data container 510 of a cloud container 500 can be accessed, manipulated, retrieved, searched, etc.” Ex. 1009, ¶¶116-117. However, as Dr. Black explains, Auradkar does **not** disclose that these “access polic[ies]” governing “users seek[ing] access to data” ever define cryptographic key information or otherwise control the encryption and storage of this data on cloud storage systems. Ex. 2003, ¶58.



Data as Custodian

- Containers are Redundant for Security
- Access is Enforced by Crypto Wrapper
- Policy is Set by Owner and guaranteed by Crypto Wrapper

FIG. 5

2. Chiueh (Ex. 1011)

Chiueh is directed to a “document processing method and ... system” intended to “improve information security of remote storage documents, and facilitate a remote server to deal with various data processing requirements,” including keyword searching. Ex. 1011, ¶7. The system includes a “client [] connected to [a] remote server through a communication network.” *Id.*, ¶9, Fig. 1. Before storing a document, the client in Chiueh performs a series of steps to create “encrypted pages” and a “significant word set (SWS),” which “the client transmits ... to [the] remote server for storage.” *Id.*, ¶¶8-9, Fig. 2; Ex. 2003, ¶59.

As Dr. Black explains, Chiueh (like Shikfa) teaches that its document processing method is performed by the **client**, and that Chiueh's method should **not** be performed by remote servers. Ex. 2003, ¶60. Consequently, in Chiueh's system, "the remote server does not have the keys [and it can] not decrypt the document pages and the significant word set." Ex. 1011, ¶10. Chiueh explains that this is important because "the remote server is probably not reliable," and if "the remote server ha[d] the decryption capability, the client could not prevent an unfaithful query processing of the remote server," and "the document stored in the remote server could be peeked/leaked." *Id.*, ¶5; Ex. 2003, ¶60.

3. Chambers (Ex. 1010)

Chambers is directed to "[t]echniques for managing access control policies" that control access to computer networks (Chambers, Abstract) and, more specifically, it describes "[t]echniques for automatically assigning access control policies (ACPs) based on device types of network client devices (NCDs)." Ex. 1010, ¶21. "When a network device first connects to the network, it is examined by the system, identified by type, and assigned an access policy according to the specified rules for that device type." *Id.* The assigned network access policies specify network parameters such as "bandwidth limits and traffic shaping rules, VLAN assignment, firewall rules,

whether a captive portal should be applied to that device, etc.” *Id.*, ¶¶32-40; Ex. 2003, ¶61.

The network access policies are assigned and enforced by “network access devices (NADs) ... managed by a management server.” Ex. 1010, ¶24, Fig. 1. The NADs “may operate as a gateway device, an access point (AP), a network switch, or a combination thereof.” *Id.* The network access policies are “downloaded from management server 101 to [the] network access device,” and then “[w]hen a client device ... is associated with [the] network access device 102, [the NAD’s] access control module 116 controls [network] access of [the] client device 108 by enforcing [the] ACPs.” *Id.*, ¶28, Fig. 6; Ex. 2003, ¶62.

As Dr. Black explains, Chambers (like Herrmann) does **not** disclose using its network access control policy management techniques in connection with encrypting, storing, or controlling access to files on cloud storage systems. Chambers does not reference cloud storage at all. Ex. 2003, ¶63.

IV. LEVEL OF ORDINARY SKILL IN THE ART

A person of ordinary skill in the art (“POSITA”) relevant to the ’678 patent at the time of the invention would have had a Bachelor of Science degree in electrical engineering and/or computer science, and two years of work or research experience in the fields of network and data security, or a

Master's degree in electrical engineering and/or computer science and one year of work or research experience in related fields. Ex. 2003, ¶18. Patent Owner's expert was a POSITA by the earliest priority date of the '678 patent. *Id.*, ¶20.

Patent Owner's description of the level of ordinary skill in the art is almost the same as that of the Petitioner. See Pet. 15. The positions set forth in this response would be the same under either parties' proposal. *Id.*, ¶19.

V. CLAIM CONSTRUCTION

In an *inter partes* review, claim terms receive their plain and ordinary meaning in light of the specification of the patent in which it appears, the same standard as used in district court. 37 C.F.R. § 42.100(b); see *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005). Petitioner states that “no constructions are necessary.” Pet. 15. Patent Owner agrees that no claim terms presently require construction to resolve any current dispute between the parties. Ex. 2003, ¶¶64-65. However, Patent Owner provides the following discussion with respect to the term “namespaces.”

All of the Challenged Claims recite the term “namespaces,” and require, as part of an encryption process, “creating namespaces for one or more of the plurality of chunks” and “configuring the namespaces of the one

or more chunks such that content of the file is encrypted in a manner that makes it searchable.”

A “namespace” has a plain meaning to a POSITA: a set of names available within a particular context. Ex. 2003, ¶¶66-67. This is the interpretation advanced by Patent Owner’s own expert, Dr. Lee. For example, Patent Owner’s expert Dr. Lee describes a “namespace” as “the ***set of names available for naming things.***” Ex. 1002, ¶101 (quoting Ex. 1014 (“Dictionary of Computer and Internet Terms”)); Ex. 1002, ¶¶98, 101, 217. Dr. Lee further explains that “a namespace in a flat structure can simply be a number chosen from within a range (e.g., 0 to $2^{128}-1$).” Ex. 1002, ¶42. Thus, in this example, the namespace defines that a name must be a number and must be within a certain range. Ex. 2003, ¶67. If an object is given the name “1234,” that is a valid name within the namespace—although a POSITA would not consider the name “1234” to itself be a namespace. *Id.*

Dr. Lee also states that “[a] namespace ensures that every object within the ***context*** of the namespace can be identified uniquely by its name(s).” *Id.*, ¶42. For example, Dr. Lee testified that a directory named “documents” is a namespace for the files within that directory—within that “documents” namespace, files can be uniquely identified by simply referring to the file name, such as “foo.” Ex. 2004, 38:18-23. Other contexts, such as other

directories, may have a different file also named “foo,” but they belong to a different namespace. *Id.*, 35:19-36:8; 39:11-40:1.

Patent Owner’s expert, Dr. Black, agrees with this description of a namespace. Ex. 2003, ¶69. One thing that is immediately apparent from this description of a namespace is that a name itself (such as an identifier that identifies an object) is **not** a namespace. *Id.* Instead, for identifiers, a namespace both defines **the set of allowable identifiers** within a particular context. *Id.* Thus, in the preceding example, a file has an object identifier (*i.e.* filename) “foo” **within** the namespace “documents” (*i.e.* the directory it belongs). Ex. 2004, 38:18-23; Ex. 1002, ¶42. That same file may have a different identifier in a different namespace—for example, in the root context, that file may have the identifier “/home/wenke/documents/foo,” *i.e.* the identifier contains the full path and filename. Ex. 2003, ¶69; Ex. 2004, 37:15-38:8.

The Challenged Claims recite “creating” and “configuring” namespaces for file chunks as part of an encryption process. Ex. 2003, ¶70. In other words, the Challenged Claims do not merely recite assigning an identifier to those file chunks, but require that the system, as part of the process of encrypting the file chunks, create and configure the namespace—

i.e. the “set of available names” within a particular context. Ex. 1002, ¶¶41-42; Ex. 2003, ¶70.

As an example of creating a namespace for file chunks as part of the encryption process, the ’678 Patent teaches that “each chunk can be encrypted based on public and private/secret keys defined by the encryption module 210,” where keys are managed based on user-assigned policies. Ex. 1001, 9:40-41, 11:63-65. These policies therefore create a namespace that defines the naming convention for the file chunks (*e.g.* encryption based on a particular key). Ex. 2003, ¶71. The ’678 Patent further explains that file chunks are stored in folders within the cloud storage service, and “[t]he name of the folder may be the encrypted version of the original file name.” Ex. 1001, 13:2-6. Hence, storing the file chunks in a folder creates a namespace scope for the file chunks. Ex. 2003, ¶71.

As an example of configuring a namespace for file chunks, the ’678 Patent describes that encrypted keywords “are stored as filenames of the chunks.” Ex. 1001, 12:9; *see also* cl. 6 (“configuring the namespaces comprises storing content of the file as filenames”). Thus, in this example, the namespace of the file chunks is configured to include the set of specific encrypted keywords and configured to be used as the filename of the file chunk stored within the cloud storage service. Ex. 2003, ¶72. This allows the

file (as split into file chunks) to be encrypted in a manner that makes it searchable. *Id.*

The prior art did not disclose these steps of creating and configuring namespaces as a part of an encryption process. Ex. 2003, ¶73. Instead, as discussed below, the prior art assigns to a document an identifier (like a filename or an SHA2 hash), which Petitioner conflates with a namespace. *Id.* Based on Petitioner’s own definition, a namespace defines the set of available names and the scope/context for those names. To the extent an identifier is even a name, they are names *within* a namespace, but they are not in and of themselves the namespace. *Id.* The use of an identifier within an already-defined namespace does not *create* or *configure* the namespace. *Id.*

VI. GROUND 1: PETITIONER FAILS TO ESTABLISH THAT CIDON, SHIKFA, AND HERRMANN RENDER CLAIMS 1-3, 8-14, AND 16-27 OBVIOUS

A. Claims 1/16: Cidon In View of Shikfa Does Not Render Obvious “Creating Namespaces” or “Configuring The Namespaces ... Such That Content of The File is Encrypted in a Manner That Makes it Searchable”

Claims 1 and 16 both require “creating namespaces for one or more of the plurality of chunks” and then “configuring the namespaces of the one or more chunks such that content of the file is encrypted in a manner that makes it searchable.” Ex. 1001, claim 1[b][ii] and 16[D]. A file encrypted in this

manner is referred to as a “searchable encrypted file.” *E.g., id.*, at 20:4-5, 21:5; Ex. 2003, ¶74.

Petitioner argues that Cidon discloses “creating namespaces” because Cidon creates a file segment identifier using an SHA2 hash, and a POSITA would understand that this identifier is a namespace. Pet. 36-37 Petitioner further argues that, while Cidon does not expressly disclose the “configuring the namespaces” limitation, Shikfa discloses it because it uses an index to associate extracted keywords with file pointers/identifiers. Pet. 40-42. Identifiers and namespaces, however, are fundamentally different. As discussed above, Petitioner’s own definition of a namespace is that it defines the set of names available in a particular context. Cidon and Shikfa’s identifiers are, at best, members of a pre-defined namespace, but not the namespace itself.

Thus, Petitioner’s arguments that Cidon in view of Shikfa render obvious these limitations fail for at least the following reasons: (1) creating an identifier to assign to a file chunk does not create a namespace, (2) Shikfa’s use of identifiers in an encrypted index is not configuring a namespace, (3) Shikfa’s use of identifiers in an encrypted index has no bearing on the manner in which a file is encrypted, and (4) a POSITA would not be motivated to

combine Cidon's server-based encryption with Shikfa's device-based encryption. Ex. 2003, ¶75.

1. Creating an Identifier is Not Creating a Namespace

Petitioner's assertion that creating an identifier to assign to a file chunk meets the "creating namespaces" limitation fails because an identifier is not a namespace. Ex. 2003, ¶¶76-80.

Petitioner asserts that Cidon discloses the "creating namespaces" limitation because Cidon "calculat[es] (and *creat[es]*) a 'unique' file segment identifier (or file segment signature) for each of the file segments (chunks) based on the content of a file segment using, e.g., SHA2 signature computation." Pet. 36-37. Petitioner argues that segment identifiers "constitute[] 'namespaces' because these identifiers were 'the set of names available for naming' corresponding file segments consistent with the plain and ordinary meaning of this term." Pet. 39-40. In other words, Petitioner alleges that when Cidon calculates a SHA2 signature to be assigned to a file segment, it has created a namespace for the file chunk. Ex. 2003, ¶76.

Under Petitioner's own definition of namespace as "the set of names available for naming," the creation of a namespace must therefore involve the *creation of the set of available names*. An identifier is not the "set of names available for naming" because an identifier is, at best, a single name within

that set of names. Ex. 2003, ¶77; Pet. 10 (“[A] namespace ensures that every object within it can be identified uniquely by its name(s).”). Likewise, the group of identifiers created for the segments in a file is also not the “set of names available for naming” because the group of identifiers is still merely a group of items within a larger set of available names—the creation of the identifiers within the namespace is not a creation of the namespace itself. Ex. 2003, ¶77. Petitioner does not point to anything in Cidon that states the set of available names—*i.e.* the universe of possible SHA2 signatures—is created or defined during the encryption process. *Id.* Nor would a POSITA consider the namespace to be created when an object is assigned an identifier—again, the identifier is at best a name ***within*** the namespace (*i.e.* the set of SHA2 signatures). *Id.* Instead, it is quite clear that the set of available names—SHA2 signatures—is simply defined by the Cidon reference in advance. *Id.* Thus, the system does not ***create*** the namespace during the encryption process because it was created in advance.

The conflation of an identifier and a namespace can be illustrated by an example from Dr. Black. Ex. 2003, ¶78. A URL (*e.g.* <https://www.google.com/home.html>) has a defined set of available names. *Id.* That namespace was not created by any computer process but by a standards body that defined it in a standards documents. *Id.* When generating an URL

to point to a file on a web server, this is not the creation of the namespace which already exists. *Id.*

Also, a POSITA would understand that “www.google.com” is a namespace for the local name “home.html.” Ex. 2003, ¶79; *see also* Ex. 1002, ¶42. In this example, the namespace “www.google.com” was created when Google registered that domain name and assigned to its web servers. Ex. 2003, ¶79.

Moreover, as discussed above in Section V, *supra*, an identifier is not a namespace. Ex. 2003, ¶80. A namespace defines the set of allowable names within a particular context. *Id.* Identifiers are simply individual items within the larger set of allowed names that was created before the encryption process. *Id.* They do not define the set of allowable names. *Id.*

2. Shikfa’s Mapping of Keywords to Identifiers In an Index Is Not “Configuring The Namespaces”

Shikfa’s discussion of an encrypted index that maps keywords to pointers/identifiers to encrypted documents does not meet the limitation “configuring the namespaces” because identifiers are not namespaces. Ex. 2003, ¶¶81-84.

Shikfa discloses “generat[ing] one or more indexes 24 for one or more documents that are to be encrypted,” where “[e]ach index 24 includes one or

more keywords for the encrypted documents 14 and pointers [or identifiers] to the encrypted documents 14 containing the keywords.” Ex. 1006, ¶29. Petitioner contends that “Shikfa’s pointers/identifiers were namespaces because they allowed identification of encrypted documents” and that “[t]hese identifiers ... were configured to allow the encrypted documents to be searched” because “they mapped search keywords and encrypted documents.” Pet. 42-43; Ex. 2003, ¶82.

But as discussed above in Section V, *supra*, an identifier is not a namespace. Ex. 2003, ¶83. A namespace defines the set of names available within a particular context. *Id.* Standing alone, an identifier is simply a single item within the set of allowed names. *Id.* It does not define the set of allowable names. *Id.*

Moreover, in Petitioner’s Cidon-Shikfa combination, the “set of allowed names”—*i.e.* the alleged namespace—is the universe of SHA2 signatures that may be used as identifiers for file chunks, as disclosed in Cidon. *See* Pet. 38-40; Ex. 2003, ¶84. However, this namespace is never “configured” by Shikfa’s encryption process. Ex. 2003, ¶84. Instead, in the proposed Cidon-Shikfa combination, the SHA2 identifiers are ***used*** (not configured) within Shikfa’s indexes as pointers to file chunks. *Id.* No configuration of the alleged namespace is performed, because the

namespace—*i.e.* the universe of available SHA2 signatures—was defined in advance. *Id.*

3. Shikfa’s Mapping of Keywords to Identifiers in an Index Has No Bearing on the Manner in Which a File is Encrypted

In addition, Shikfa’s use of identifiers within an encrypted index also does not involve a configuration “such that content of the file is encrypted in a manner that makes it searchable.” Ex. 2003, ¶¶85-87.

Shikfa’s use of identifiers within an index has no bearing on or relationship to the manner in which a file is encrypted. Ex. 2003, ¶86. Shikfa is quite clear that its index is created *before* a document is encrypted. Ex. 1006, ¶29. Moreover, Shikfa states that the document is encrypted “using any desired encryption mechanism known in the art.” *Id.* Dr. Lee confirmed at his deposition that the creation of the index is “independent” and “orthogonal” to the encryption of the document. Ex. 2004, 136:13-137:7. Thus, even if the identifiers are used in an index for searching an encrypted file, those indexes are completely independent of the file itself. Ex. 2003, ¶86. The manner in which the file is encrypted is, as Dr. Lee put it, “orthogonal” to the use of identifiers in Shikfa’s index. *Id.*

This is not a trivial distinction. Ex. 2003, ¶87. As discussed above, whereas the prior art describes the use of external encrypted indexes, the ’678

Patent describes a novel process where an encrypted index is embedded into the encrypted file by using the encrypted keywords as filenames for the file's chunks. *Id.* As a result, queries can be performed using a cloud storage service's built-in searching mechanism. Ex. 1001, 10:17-19; Ex. 1003, 40. The encrypted file itself becomes searchable. Indeed, the '678 Patent explains that the users can “download[] the searchable encrypted files ..., which would then allow the user 202b to search metadata and/or content associated with the searchable encrypted files even in offline mode when cloud service providers 206 are inaccessible via a network connection.” Ex. 1001, 10:23-28. Such portability is not possible using Shikfa's teachings, because the encrypted files must be accompanied by the index in order to be searched—the encrypted files themselves are not searchable. Ex. 2003, ¶87.

4. A POSITA Would Not Have Been Motivated To Combine Cidon And Hermann With Shikfa

Petitioner fails to demonstrate a motivation to combine Cidon and Hermann with Shikfa. Ex. 2003, ¶¶88-94. Petitioner asserts that Shikfa's identifiers would have been implemented in Hermann's gateway server, which Petitioner reads the claimed “gateway device” on. Pet. at 44-45. As an initial matter, Shikfa already discloses its own “gateway device,” namely the broker server, which acts as an intermediary between the client and cloud

storage providers. *See* Ex. 1006, ¶28 (“The broker server 12 acts as an intermediary between the client 16 and the cloud storage provider 18...”); ¶¶42-43; Ex. 2003, ¶89. Thus, even if a POSITA were to be motivated to implement a gateway device in Cidon’s system, there is no reason he/she would have incorporated Hermann’s gateway server into the modified system (because Shikfa already discloses one). Ex. 2003, ¶89.

Further, Shikfa is explicit that the client, not the broker server, performs the encryption of the document. *See* Ex. 1006, ¶¶30, 36; Ex. 2003, ¶90. Dr. Lee confirmed that Shikfa is “pretty clear that the client performs the encryption.” Ex. 2004, 148:22-25. Dr. Lee also confirmed that “the broker doesn’t know the encryption algorithm being used and the key, so they cannot decrypt ... the file content.” *Id.*, 151:21-23.

In Shikfa’s system, isolating the broker service from the client ensures that only the client can access the content of the client’s data. *Id.*, ¶48 (“The broker system 10 also ensures that the broker server 12 cannot access and/or discover the content of the client’s data because the broker server 12 does not know the encryption used on the encrypted documents 14. Thus, the broker system 10 provides searchable encryption while maintaining the confidentiality of data external to the client 16.”); Ex. 2003, ¶91. Dr. Lee confirmed the importance of keeping the encryption on the user device,

testifying that if a gateway device (such as a router) has the capability of encrypting data, that makes the data susceptible to a “man in the middle” attack. Ex. 2004, 80:10-82:5. Dr. Black agrees with Dr. Lee. Ex. 2003, ¶91.

Thus, a POSITA seeking to modify Cidon, which describes performing encryption by a “management server,” would not have been motivated to look to Shikfa, which emphasizes the importance of using the user device to encrypt files (because of the lack of trust of other devices). Ex. 2003, ¶92.

Moreover, Shikfa teaches away from the claimed invention of a gateway device that performs encryption. Ex. 2003, ¶93. A reference “teaches away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken in the claim.” *Meriesonne v. Google, Inc.*, 849 F.3d 1379, 1382 (Fed. Cir. 2017). Here, as discussed above, there is ample evidence that a POSITA would be led away from the claimed invention because Shikfa explains why the lack of trust in external devices would necessitate performing encryption on a user’s device. Ex. 2003, ¶¶93-94. Shikfa’s teachings are further corroborated by Dr. Lee’s testimony. Although the Institution Decision states that Patent Owner merely “attacks the references individually rather than as Petitioner proposes,” respectfully, the Board may not “ignore[] the express disclosure ...

that teaches away from the claimed invention.” *Chermours Co. FC, LLC v. Daikin Industries*, 4 F.4th 1370, 1376 (Fed. Cir. 2021). That ruling from the Institution Decision should not be adopted in the final decision.

B. Claims 1/16: Cidon Does Not Disclose or Render Obvious a “Policy for Managing Access to and Processing a File to Be Stored on Cloud Platforms”

Claim 1 recites *assigning to one or more users* “a policy for managing access to and processing a file to be stored on one or more cloud platforms.” Ex. 1001, claim 1[a][i]. Similarly, claim 16 recites implementing a policy *for a user* where “the policy defines and manages access to and processing of a file to be uploaded on a cloud.” *Id.*, claim 16[c][i][A]. Both independent claims therefore require a *user*-assigned policy that (1) manages access to one or more cloud platforms (hereinafter, the “access management” limitation) and (2) processes a file to be stored on the cloud platforms (hereinafter, the “processing” limitation). Petitioner has failed to show that Cidon discloses such a policy. Ex. 2003, ¶¶95-112.

1. Cidon’s Encryption Policy Is Not Assigned to a User

Petitioner failed to show that Cidon discloses a *user*-assigned policy that performs the “processing” limitation. Ex. 2003, ¶¶97-104. Petitioner relies exclusively on an “encryption policy” disclosed in Cidon’s paragraphs 224 and 227. Pet. 25 (“Cidon’s policy also ‘dictate[d]’ whether certain files

were to be encrypted (policy for processing a file).”). But Cidon’s encryption policy is assigned to a “***specific file***,” not a user. See Ex. 1004, ¶227 (“[A] ***specific file*** was found to be under the encryption policy as defined by the file ancestry tree. This means that the original parent is encrypted and the policy dictates that all descendent files must be encrypted too.”); Ex. 2003, ¶99. The context of Cidon’s disclosure also makes clear that an encryption policy, as part of placement policies, is “***defined on documents***” as a result of a file’s ancestry tree, not assigned to users. *Id.*, ¶207 (“Policies can be ***defined on documents*** which are ancestors of other documents. For example, inheritance rules can be defined which imply that a policy that is defined on a document ... can be extended to its sub-tree.”); Ex. 2004, 57:6-9 (“[E]ncryption [i]s one of the policies you can define ***on a document***.”). As Dr. Lee explains it, “ancestry information for a file includes which file it was copied from,” thus if a parent file is encrypted, the encryption policy “dictates that all descendent files must be encrypted too.” Ex. 2004, 43:8-20; 61:22-62:11. Dr. Black agrees with Dr. Lee. Ex. 2003, ¶100.

Indeed, Cidon is clear that encryption policies assigned to a document are not assigned to a user, because “files in a single ancestry tree may belong to ... different users,” yet the encryption policy would still require all of those files to be encrypted. Ex. 1004, ¶¶195, 227; Ex. 2004, 48:14-17 (“B is a child

of A regardless who owns B or who owns A”); Ex. 2003, ¶101. Dr. Lee confirmed that Cidon does “not require [that parent and child files] belong to the same user.” Ex. 2004, 48:14-17.

Assigning an encryption policy to a **file** is fundamentally different than assigning an encryption policy to a **user**. Ex. 2003, ¶102. A POSITA would understand that a user-assigned encryption policy would allow defining encryption requirements for files owned by specific users, whereas a file-assigned encryption policy defines encryption requirements for files regardless of which user owns the files. Ex. 2003, ¶103. For example, with user-assigned encryption policies, users who typically create highly sensitive documents could have a policy where their documents are encrypted—such a policy is not possible with file-assigned encryption policies. *Id.*, ¶101.

Given the fundamental difference between the two, user-assigned encryption policies would not be obvious in view of file-assigned policies. *Id.*, ¶¶102-03. The Petition never argues it would be obvious. In fact, far from explaining how it would be obvious, the Petition never even acknowledges that there is difference, nor mention that Cidon’s encryption policy is file-assigned instead of user-assigned despite their fundamental differences. *Id.*

Consistent with the claim language, the specification of the ‘678 patent affirms that a policy must be assigned to a user, not a file. Ex. 2003, ¶103.

For example, the specification describes that “each client 102 is assigned a policy, which is configured to define the rights of the client 102,” and such rights “control the manner in which the respective client 102 encrypts, stores, access, and manages files ...” Ex. 1001, 7:61-8:4. The ‘678 patent also explains that “a policy can be implemented for a client 102 or a group of clients 102, also interchangeably referred to as users 102 hereinafter, based on their role and responsibility in the organization,” and “[t]he selected policy can then be used for defining the manner in which the client 102 interacts with multiple cloud storage providers ...” *Id.*, 8:4-14; *see also id.*, 2:17-21 (“A file storage policy is assigned by the cloud storage gateway device to each user. The assigned file storage policy defines access rights, storage diversity requirements and a type of encryption to be applied to files ***for the corresponding user.***”).

Cidon’s encryption policy, on the other hand, would not differentiate encryption requirements of a file for different users: regardless of their role or responsibility in the organization, all users would be required to encrypt a file if the file is subject to an encryption policy (*e.g.*, if the file’s original parent is encrypted). Ex. 2003, ¶104. Cidon’s encryption policy is therefore neither “assign[ed]” to a user (claim 1) or “implement[ed]” for a user (claim 16).

2. Cidon's "Encryption Policy" Does Not Govern "Processing of a File to Be Uploaded on a Cloud"

Cidon's encryption policy also does not meet the "processing" limitation because it does not govern the "processing of a file *to be uploaded* on a cloud." Instead, Cidon's encryption policy applies to files that have already been uploaded. Ex. 2003, ¶¶105-107.

Specifically, Cidon explains that encryption of a file is performed by a "management server" which "encrypts files *on the user Dropbox accounts*," *i.e.*, files already uploaded to the cloud. Ex. 1004, ¶223; Ex. 2003, ¶106. Moreover, Cidon explains that encryption occurs after "a specific file was found to be under the encryption policy as defined by the file ancestry tree." *Id.*, ¶227. Dr. Lee confirmed that encryption does not occur until after "the user uploads an unencrypted file to [a Dropbox] secured folder." Ex. 2004, 87:4-7.

Thus, Cidon is clear that its encryption policy applies to files that have already been uploaded. Ex. 2003, ¶107. It therefore does not meet the claimed policy that governs "processing of a file *to be uploaded* on a cloud."

3. Cidon's "Encryption Policy" Does Not Govern Access Management

Petitioner has failed to show that Cidon's encryption policy meets the "access management" limitation. As noted above, the claims require that a

policy govern not only the “processing” of a file to be stored on cloud platforms but also access management. *See* claim 1 (“a policy for managing access to **and** processing of a file to be stored on one or more cloud platforms”); claim 16 (“the policy defines and manages access to **and** processing of a file to be uploaded on a cloud”). As such, there must be at least **one** policy, assigned to a user (not a file), that governs **both** access management and processing. Ex. 2003, ¶¶108-112.

Petitioner has failed to identify any such single policy in Cidon. For the access management limitation, Petitioner relies on Cidon’s paragraphs [0102] and [0108], which mention enforcement of an “access policy” as one of the functions performed by a management server. Pet. 25. But in Cidon’s disclosures, an access policy is distinct from an encryption policy. Ex. 2003, ¶109. Unlike an encryption policy which is defined by a file’s ancestry, access policies are defined by an administrator; an encryption policy is specific to a file, whereas an access policy is assigned for a user or users; and an encryption policy dictates whether a file must be encrypted, whereas an access policy determines if a user can be granted access to a file. Ex. 1004, ¶108, 227. Because Petitioner failed to identify a policy in Cidon that governs (1) managing access to cloud platforms **and** (2) processing of a file to be stored on the cloud platforms, Ground 1 should be rejected. Ex. 2003, ¶109.

The Institution Decision rejected a similar argument made in Patent Owner's Preliminary Response. See Paper 10, at 17. The Institution Decision stated that "a policy" need not be limited to a single policy because "[a]s a general rule, the word 'a' or 'an' in a patent claim carry the meaning of 'one or more.'" *Id.* Patent Owner does not disagree that there could be more than one policy. Ex. 2003, ¶110. But respectfully, this case is different than the typical case where "a" means "one or more." In this case, the claims plainly require "a policy for managing access to and processing a file," and that this policy is "assign[ed] to one or more users." Ex. 2003, ¶111. The claim also requires that the policy defines "cryptographic key information." *Id.* The claim is clear on its face that **a** policy must perform **both** the "managing" **and** "processing" steps, as well as defining "cryptographic key information."

In this case, therefore, while the use of the term "a" certainly does not preclude multiple policies, at least one of those policies must perform both the managing **and** processing steps. Ex. 2003, ¶111. The specification of the '678 Patent is consistent with the requirement that there is at least one policy that performs all of the recited functions. Ex. 2003, ¶112. For example, the specification describes that "each client 102 is assigned **a** policy, which is configured to define the rights of the client," which "control the manner in

which the respective client 102 encrypts, stores, accesses, and manages files.”

Ex. 1001, 7:65-8:2; *see also id.*, 3:45-67.

A recent Federal Circuit decision supports Patent Owner’s reading of “a policy” as requiring at least one policy that meets all the recited functions. In *Salazar v. AT&T Mobility LLC*, 64 F.4th 1311 (Fed. Cir. 2023), the claim recited “a microprocessor” that was claimed to perform several functions. The Federal Circuit found that while the claims allowed for “one or more microprocessors,” it further found that the claims require “at least one of which is configured to perform the [recited] generating, creating, retrieving, and generating functions.” *Id.*, at 1314. The Court held that “it does not suffice to have multiple microprocessors able to perform just one of the recited functions; the claim language requires at least one microprocessor capable of performing each of the recited functions.” *Id.* at 1318.

Similarly, in *Tivo, Inc. v. EchoStar Communications Corp.*, 516 F.3d 1290 (Fed. Cir. 2008), the Court noted that “the question whether ‘a’ or ‘an’ is treated as singular or plural depends heavily on the context of its use.” *Id.*, 516 F.3d at 1303. The Court then held that the term “assembl[ing] said video **and** audio components into an MPEG stream” “clearly indicates that two separate components are assembled into **a single stream**, not that the video components are assembled into one stream and the audio components into a

second stream.” *Id.* This language is similar to the ’678 patent’s recitation of “a policy for managing access to ***and*** processing a file.”

These holdings squarely apply to this situation and, respectfully, the Board should construe “a policy” to require at least one policy which performs all the recited functions.

C. **Claims 1/16: Cidon Does Not Disclose or Render Obvious “Storing, by the Gateway Device” (Claim 1) or “a Storage Module” For Storing (Claim 16) “The Searchable Encrypted File on The One or More Cloud Platforms Based on the Policy”**

Claim 1 requires “storing, by the gateway device, the searchable encrypted file on the one or more cloud platforms based on the policy.” Ex. 1001, claim 1[c]. The term “***the*** policy” has antecedent basis to “a policy” recited in limitation 1[a][i]. Similarly, claim 16 requires “a storage module, which, when executed by the one or more processors, stores the searchable encrypted file on the one or more cloud platforms based on ***the*** policy” (*id.*, claim 1[c][iii]), where the policy refers to the earlier-recited “a policy.” Petitioner asserts that the “storing” / “storage module” limitation is met because Cidon’s policies included “[p]lacement [of files] in specific directories or shares.” Pet. at 48-49. While Cidon’s placement policies, standing alone, may meet the “storing” / “storage module” limitation, the placement policies are not the same “policy” that Petitioner pointed to for the

earlier-recited “a policy.” Ex. 2003, ¶113. As discussed above, for the “access management” function of limitation 1[a][i], Petitioner relies on Cidon’s paragraphs [0102] and [0108], which disclose an “access policy” defined for the users. Pet. at 25. The “placement policies” referred to in [0208] and [0209], however, are policies “defined on documents.” Ex. 1004, ¶¶206-07. In other words, Cidon’s placement policies apply as a result of file-specific properties such as “similarities and ancestry trees” and are not assigned to the users. Ex. 2003, ¶113.

As the Board noted (Paper 10, at 25), Cidon enumerates the different functions performed by a management server in paragraph [0102], but these functions are distinct from the “placement policies” described in paragraphs [0206]-[0228]. *See* Ex. 1004, ¶102 (describing that a management server may be arranged to (1) provide access to file repositories, (2) store file repository information, (3) analyze and encrypt files, (4) store and retrieve encryption keys, (5) provide access policy configuration for secure file access, (6) enforce a secure file access policy, and (7) report file access use and devices used for access.”). Cidon does not disclose that a management server performs these functions based on the placement policies. Ex. 2003, ¶114. Indeed, unlike the placement policies, Cidon is explicit that access policies are assigned by IT departments for different users (*see* Ex. 1004, ¶108), further undermining

that Cidon discloses a policy that governs **both** access management and storage of an encrypted file on the cloud. Ex. 2003, ¶114.

D. Claims 11/25: Cidon Does Not Disclose or Render Obvious Storing a File “Across Two Or More Cloud Platforms”

Claims 11 and 25 require that the encrypted file “is stored across two more cloud platforms.” Petitioner fails to establish that Cidon disclosure of “archiving” and “backup” meets this limitation because archiving/backup is merely copying of a file to another location, as opposed to storing a file **across** multiple locations. Ex. 2003, ¶¶115-118.

Consistent with the specification, the plain meaning of storing “across” multiple platforms is that a single file has different parts stored on different platforms. Ex. 2003, ¶116. For example, the ’678 Patent explains that “Clients may also store such content across two or more stores 114 such that the content ***is divided into multiple chunks and each chunk is stored on a different cloud store*** 114.” Ex. 1001, 7:43-45.

Petitioner does not cite to anything in Cidon that discloses storing a file across multiple platforms. Instead, Petitioner relies on Cidon’s disclosure of “archiving and backup to other storage services.” Ex. 1004, ¶174. As an initial matter, Cidon’s disclosure of “archiving and backup” is with respect to “DNA extraction,” not encrypted documents. *Id.*; Ex. 2003, ¶117.

More importantly, archiving and backup of a file does not constitute splitting up a file into chunks and storing chunks on different platforms. Instead, archiving and backup would be understood to mean merely storing a **copy** of the file to a different store. Ex. 2003, ¶118. Dr. Lee confirmed this, stating that “[b]ackup normally means that I ... put a **copy** of ... your file in some kind of backup.” Ex. 2004, 107:24-108:1. By storing a **copy**, this allows recovery of a lost or corrupt file. *Id.* Dr. Lee also confirmed that archiving and backup are synonymous. *Id.*, 108:5-23. Dr. Black agrees with Dr. Lee. Ex. 2003, ¶118. For these reasons, archiving and copying do not meet storing a file “across two or more cloud platforms.”

E. **Claim 14: Cidon Does Not Disclose or Render Obvious a Policy That Defines “One or More Authorized Locations to Which the File May be Stored”**

Claim 14 requires that “the policy” of claim 1 define, among other things, “one or more authorized locations to which the file may be stored.” Ex. 1001, cl. 14. Petitioner asserts that “[a] POSITA would have found it obvious that the file locations would be part of Cidon’s security policy that could also define (and manage) such location information.” Pet. 64. Petitioner does not identify what “Cidon’s security policy” refers to. Petitioner relies on Cidon’s paragraph [0364]. Ex. 2003, ¶119. That paragraph, however, simply discloses moving an encrypted file to another

location, and Cidon does not disclose that such relocation is done based on any “policy.” Ex. 1004, ¶364 (“The modifying action (2130) may include ... generating the modified file to comprise access information to another location in which the modified file is to be stored, and storing the modified file at the other location . . .”).

Further, as discussed above for claim 1, Petitioner does not identify any policy based on which the claimed access management and processing of a file to be stored are performed. Ex. 2003, ¶120. Petitioner’s argument for claim 14 fails for the additional reason that Petitioner does not point to any policy based on which the functions recited in claim 1 *and* claim 14 are performed.

VII. GROUNDS 2 AND 3 FAIL ON THE MERITS

As discussed above, Petitioner fails to establish that Cidon in view of Shikfa and Herrmann renders independent claims 1 and 16 obvious. Because Grounds 2 and 3 rely on Ground 1 for the limitations of the independent claims, Grounds 2 and 3 also fail for the same reasons. Ex. 2003, ¶121.

VIII. GROUND 4: PETITIONER FAILS TO ESTABLISH THAT AURADKAR, CHIUEH, AND CHAMBERS RENDER CLAIMS 1-3, 8-19, AND 21-27 OBVIOUS

A. Claims 1/16: Auradkar in View of Chiueh Does Not Render Obvious “Creating Namespaces” or “Configuring The Namespaces ... Such That Content of The File is Encrypted in a Manner That Makes it Searchable”

Similar to Ground 1, Petitioner argues that Auradkar and/or Chiueh disclose “creating namespaces” because both references identify an object by its filename, and a POSITA would understand that a filename is a namespace. Pet. 94-95. Petitioner further argues that Chiueh discloses “configuring the namespaces” because it uses an index to map extracted keywords with filenames. Pet. 105-06. But Auradkar and Chiueh’s filenames are, at best, members “local name[s] *within* the namespace” (Pet. 11), not the namespace itself.

Thus, Petitioner’s arguments that Auradkar in view of Chiueh render obvious these limitations fail for at least the following reasons: (1) creating an identifier to assign to a file chunk does not create a namespace, (2) Chiueh’s use of identifiers in an encrypted index is not configuring a namespace, (3) Chiueh’s use of identifiers in an encrypted index has no bearing on the manner in which a file is encrypted, and (4) a POSITA would not be motivated to combine Auradkar’s server-based encryption with Chiueh’s device-based encryption. Ex. 2003, ¶¶122-144.

1. Creating an Identifier is Not Creating a Namespace

As with the Cidon-Shikfa combination, the Auradkar-Chiueh combination likewise fails because creating an identifier to assign to a file chunk does not meet the claimed “creating namespaces” because an identifier is not a namespace. Ex. 2003, ¶¶123-29.

Petitioner asserts that Auradkar discloses the “creating namespaces” limitation because Auradkar discloses “creating tags for payloads (e.g., payload 1200, XML fragment 1902 (chunks)), such as tag 1200 that included the filename of the payload (FName: fox.tx) and XML tags 1900, and encrypting them.” Pet. 94-95. Petitioner argues that “[a] POSITA would have found it obvious that these tags constituted namespaces for the payloads (e.g., file 1202), because, consistent with the plain and ordinary meaning of a namespace, they were a ‘set of names available for naming things’ such as files or segments.” Pet. 96; Ex. 2003, ¶124.

Petitioner also provides an alternative theory using Chiueh, arguing that “Chiueh’s system also created and assigned a document identification or ID (namespace) to each of the encrypted pages,” the document ID being a “combination of a filename and a page offset in the document.” Pet. 96-97; Ex. 2003, ¶125.

Under Petitioner’s own definition of namespace as “the set of names available for naming things,” the creation of a namespace must therefore involve the *creation of the set of available names*. Ex. 2003, ¶126. A filename is not the “set of names available for naming things” because, at best, an identifier like a filename is merely a single item within a set. *Id.* Indeed, Dr. Lee admitted that a filename is simply a local name *within* the namespace, not the namespace itself. Ex. 1002, ¶42; Ex. 2004, 37:8-38:12 (admitting that filename “foo” is a local name “within the namespace of documents”).

Likewise, a group of filenames is also not the “set of names available for naming” because the group of filenames is still merely a group of items within a larger set of available names—the creation of the identifiers within the namespace is not a creation of the namespace itself. Ex. 2003, ¶127.

Petitioner does not point to anything in Auradkar or Chiueh that states the set of available names—*i.e.* the universe of valid filenames (or filenames and page offsets)—is created or defined during the encryption process. Ex. 2003, ¶128. Instead, it is quite clear that the set of available names—valid identifiers—is simply defined by the Auradkar and Chiueh references in advance. *Id.* Thus, the system does not *create* the namespace during the encryption process because it was created in advance. *Id.*

Moreover, as discussed above in Section V, *supra*, an identifier is not a namespace. Ex. 2003, ¶129. A namespace defines the set of allowable names within a particular context. *Id.* Standing alone, an identifier is simply a single item or name within the set of allowed names. *Id.* For example, the Petition relies on the file name “fox.tx” as meeting the claimed namespace. Pet. 95. The file “fox.tx,” however, is just one available file name within a namespace and not the namespace itself. Ex. 2003, ¶129.

2. Chiueh’s Mapping of Keywords to Document Identifiers in an Index Is Not “Configuring The Namespaces”

Chiueh’s teaching of an encrypted index that maps keywords document identifiers does not meet the limitation “configuring the namespaces” because identifiers are not namespaces. Ex. 2003, ¶¶130-33.

Chiueh discloses generating an “SWS” (*i.e.*, an index) that maps keywords extracted from an unencrypted document to a document identifier that “may include a file name of the document and a page offset in the document.” Ex. 1011, ¶¶21-23, 40. Petitioner contends that “[a] POSITA would have found it obvious that Chiueh’s system configured document IDs (namespaces) of the encrypted pages (chunks) because it mapped those IDs (indicated in the index information) to keywords appearing in the

corresponding pages in a global search index (GSI), such that the page content in encrypted form was searchable.” Pet. 105-06; Ex. 2003, ¶131.

But as discussed above in Section V, *supra*, an identifier is not a namespace. Ex. 2003, ¶132. A namespace defines the set of names in a particular context. *Id.* Standing alone, an identifier is simply a single item or name within the set of allowed names, not the namespace. *Id.*

Moreover, in Petitioner’s Auradkar-Chiueh combination, the “set of allowed names”—*i.e.*, the alleged namespace—is the universe of filenames and page offsets, as disclosed in Chiueh. Ex. 2003, ¶133. However, this namespace is never “configured” by Chiueh’s encryption process. *Id.* Instead, in the proposed Auradkar-Chiueh combination, the document identifiers (filenames) are ***used*** (not configured) within Chiueh’s indexes as index information in order to locate a document page. *Id.* No configuration of the alleged namespace is performed, because the alleged namespace—*i.e.* the universe of valid filenames and page offsets—was defined in advance by the reference. *Id.*

3. **Chiueh's Mapping of Keywords to Identifiers in an Index Has No Bearing On the Manner in Which a File is Encrypted**

In addition, Chiueh's use of document identifiers within an encrypted index also does not involve a configuration "such that content of the file is encrypted in a manner that makes it searchable." Ex. 2003, ¶¶134-36.

Chiueh's use of document identifiers within an index has no bearing on or relationship to the manner in which a file is encrypted. Ex. 2003, ¶135. Chiueh is quite clear that its index is created **before** a document is encrypted. Ex. 1011, ¶21 ("the client 110 removes a part of words from the document pages that **are still not encrypted**"). Moreover, Dr. Lee confirmed at his deposition that the creation of the index is "orthogonal" to the encryption of the document. Ex. 2004, 184:24-185:8. Thus, even the identifiers are used in an index for searching an encrypted file, those indexes are completely independent of the file itself. Ex. 2003, ¶135. The manner in which the file is encrypted is, as Dr. Lee put it, "orthogonal" to the use of document identifiers in Chiueh's index. *Id.*

This is not a trivial distinction. Ex. 2003, ¶136. As discussed above, whereas the prior art describes the use of external encrypted indexes, the '678 Patent describes a novel process where an encrypted index is embedded into the encrypted file by using the encrypted keywords as filenames for the file's

chunks. *Id.* As a result, queries can now be performed using a cloud storage service’s built-in searching mechanism. Ex. 1001, 10:17-19; Ex. 1003, 40; Ex. 2003, ¶136. The encrypted file itself becomes searchable. Ex. 2003, ¶136. Indeed, the ’678 Patent explains that the users can “download[] the searchable encrypted files ..., which would then allow the user 202b to search metadata and/or content associated with the searchable encrypted files even in offline mode when cloud service providers 206 are inaccessible via a network connection.” Ex. 1001, 10:23-28. Such portability is not possible using Chiueh’s teachings, because the encrypted document pages must be accompanied by the index in order to be searched—the encrypted document pages themselves are not searchable. Ex. 2003, ¶136.

4. A POSITA Would Not Have Been Motivated to Combine Auradkar and Chambers and Chiueh

Petitioner failed to demonstrate a motivation to combine Auradkar and Chambers with Chiueh. Ex. 2003, ¶¶137-41. Petitioner asserts that Chiueh’s encryption techniques would be incorporated into the Auradkar and Chambers combination. However, Chiueh is explicit that the user client device performs the encryption of the document pages and the keywords. See Ex. 1004, ¶¶20-24.

The Institution Decision rejected a similar argument, stating that “Chiueh does not counsel against encrypting document pages and word sets at an enterprise system of Auradkar or at an intermediate gateway of the Auradkar-Chambers combination.” Paper 10, 44. With respect, Chiueh actually explains why it is important the client device performs encryption (and decryption) instead of the server: because “the remote server is probably not reliable” as “the document stored in the remote server could be peeked/leaked” if the remote server is given decryption capabilities. *Id.*, ¶15; Ex. 2003, ¶138.

In Chiueh’s system, isolating the remote server from the client ensures that only the client can access the content of the client’s data. Ex. 2003, ¶139. As explained by Chiueh, when a user wishes to query an encrypted index, the remote server only receives an “encrypted keyword,” searches the encrypted index using the encrypted keyword, and “transmits the targeted encrypted page.” Ex. 1004, ¶25. Thus, Chiueh’s system where the user device encrypts the document is contrary to the claim requirement that encryption of the file be performed “by the gateway device.” Ex. 2003, ¶139. Dr. Lee likewise confirmed the importance of keeping the encryption on the user device, testifying that if a gateway device (such as a router) has the capability of

encrypting data, that makes the data susceptible to a “man in the middle” attack. Ex. 2004, 80:10-82:5; Ex. 2003, ¶139.

Thus, a POSITA seeking to modify Auradkar in view of Chambers, which Petitioner asserts performs encryption on a remote gateway device, would not have been motivated to look to Chiueh, which emphasizes the importance of using the user device to encrypt files (because of the lack of trust of other devices). Ex. 2003, ¶140.

Moreover, Chiueh teaches away from the claimed invention of a gateway device that performs encryption. Ex. 2003, ¶141. A reference “teaches away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken in the claim.” *Meriesonne v. Google, Inc.*, 849 F.3d 1379, 1382 (Fed. Cir. 2017). Here, as discussed above, there is ample evidence that a POSITA would be led away from the claimed invention because Chiueh explains why the lack of trust in external devices would necessitate performing encryption on a user’s device. Ex. 2003, ¶141. Chiueh’s teachings are further corroborated by Dr. Lee’s testimony of dangerous man-in-the-middle attacks if a gateway device is able to encrypt and decrypt transmissions. Although the Institution Decision stated that Patent Owner merely “attacks the references individually

rather than as Petitioner proposes,” respectfully, the Board should not “ignore[] the express disclosure ... that teaches away from the claimed invention.” *Chermours Co. FC, LLC v. Daikin Industries*, 4 F.4th 1370, 1376 (Fed. Cir. 2021). For these reasons, the holding in the Institution Decision should not be adopted in the final decision.

B. Claims 1/16: Petitioner Fails to Show Auradkar Discloses “Cryptographic Key Information Defined by the Policy”

Claims 1 and 16 recite “encrypting ... using cryptographic key information defined by the policy.” Petitioner has failed to show any policy that defines cryptographic key information. Ex. 2003, ¶¶142-45.

The ’678 Patent describes that “a selected policy can be used to define encryption keys, decryption keys, and encryption type, among other attributes for carrying out the encryption of data.” Ex. 1001, 4:4-7. A POSITA would therefore understand that “cryptographic key information” at least includes defining the encryption and/or decryption keys and encryption type. Ex. 2003, ¶143.

Petitioner argues that Auradkar discloses this limitation because it discloses that “a policy could be created that requires that (sensitive) medical data must be backed up once a day using AES 256 encryption to a secure

server in a trusted data center.” Ex. 1009, ¶96. However, there are several fatal issues with Petitioner’s argument. Ex. 2003, ¶144.

Auradkar states that this policy is with respect to a “paragraph’s classification,” *i.e.*, the secrecy classification of a paragraph within a document. Ex. 1009, ¶95. But Auradkar does not state that such a policy is the same Enterprise Access Policy that Petitioner relies on for the antecedent basis of “policy.” See Pet. 80-81. The antecedent basis for “the policy” that defines cryptographic key information is “a policy for managing access to and processing a file to be stored.” Ex. 2003, ¶145. There is no assertion, nor any teaching in Auradkar they are the same policies.

As discussed above, the Institution Decision rejected a similar argument made in Patent Owner’s Preliminary Response, stating that the claims do not require a single policy that performs all the recited functions. See Paper 10, 17-18. However, as discussed above, Patent Owner respectfully submits that this reading is inconsistent with the Federal Circuit’s decision in *Salazar*, and should not be adopted in the final decision.

C. Claims 1/16: Petitioner Fails to Show Auradkar Discloses a Policy for “Processing Of a File to be Uploaded on a Cloud”

Auradkar’s policy also does not meet the “processing” limitation because it does not govern the “processing of a file *to be uploaded* on a cloud.”

Instead, Auradkar’s policy is with respect to files that have already been uploaded to a cloud service. Ex. 2003, ¶¶146-47.

Specifically, Auradkar explains that its policy “affect[s] how data 512 **stored in a data container 510 of a cloud container 500** [could] be accessed, manipulated, retrieved, searched.” Ex. 1009, ¶117. Indeed, Auradkar is explicit that access to the data is controlled by “crypto access controls” that reside within the cloud container—these controls are “guided by, but separated from, enterprise access policy 530” and “protect the data 512 from unwarranted access by users 508.” *Id.*; Fig. 5. As a result, Auradkar’s policy only governs data that has already been uploaded—it cannot affect data that is “**to be uploaded** on a cloud” because the crypto access controls that implement the policy only reside on the cloud server. Ex. 2003, ¶147.

IX. GROUND 5 FAILS ON THE MERITS

As discussed above, Petitioner fails to establish that Auradkar in view of Chiueh and Chambers renders independent claims 1 and 16 obvious. Because Ground 5 relies on Ground 4 with regard to the limitations of the independent claims, Ground 5 also fails for the same reasons. Ex. 2003, ¶148.

Ground 5 also fails because Inoue does not disclose or render obvious claim 6’s “wherein the step of configuring the namespaces comprises storing content of the file as filenames.” Ex. 2003, ¶¶149-51. Claim 6 depends on

claim 1’s “configuring the namespaces of the one or more chunks such that content of the file is encrypted in a manner that makes it searchable.” Specifically, Inoue does not teach that a filename stores *content of the file* nor does Inoue teach that the content of the file becomes *searchable*. Ex. 2003, ¶149.

Inoue is a system where a user can label an image by providing a voice input with keywords. Ex. 2003, ¶150. Specifically, Inoue teaches a method where an image file is given a filename using *one* keyword extracted from a *voice input*. Ex. 1012, ¶101 (“In the voice recognition processing, one or more keywords (words) can be extracted *from the input voice data* using a word spotting voice recognition technology.”); ¶104 (“[T]he voice information setting section 707 ... sets *one* of the keywords as the title (a filename) of the image data.”). The filename does not store content of the file (*i.e.* the image) because the keyword comes from a voice input, not the image itself. As a result, because the keyword does not come from the content of the file (and indeed, could be any arbitrary word provided by the speaker), using Inoue’s method does not result in making the *content of the file* searchable. Ex. 2003, ¶150.

Thus, Petitioner’s assertion that “[a] POSITA would have been motivated to incorporate Inoue’s functionality of assigning a keyword

extracted from file content ... into the Auradkar-Chiueh-Chambers system” is based on a false predicate. Ex. 2003, ¶151. Inoue teaches assigning to a filename a keyword extracted from a ***voice input***, not from the file itself.

X. CONCLUSION

For the foregoing reasons, Patent Owner respectfully submits that the Board should uphold the challenged claims.

Date: December 22, 2023

By: /s/ James Glass

James M. Glass (Reg. No. 46,729)
QUINN EMANUEL URQUHART &
SULLIVAN LLP

51 Madison Avenue, 22nd Floor
New York, NY 10010

Email: jimglass@quinnemanuel.com

Phone: 212-849-7142

Fax: 212-849-7100

*Counsel for Patent Owner
Fortinet, Inc.*

CERTIFICATE OF LENGTH (37 C.F.R. §§ 42.24)

The undersigned hereby certifies that, according to the word-processing system used to prepare the foregoing document, this document has 11,891 words and thus complies with the applicable word limit.

Date: December 22, 2023

By: /s/ James Glass

James M. Glass (Reg. No. 46,729)
QUINN EMANUEL URQUHART &
SULLIVAN LLP
51 Madison Avenue, 22nd Floor
New York, NY 10010
Email: jimglass@quinnemanuel.com
Phone: 212-849-7142
Fax: 212-849-7100

*Counsel for Patent Owner
Fortinet, Inc.*

CERTIFICATE OF SERVICE (37 C.F.R. §§ 42.6(E), 42.105(A))

The undersigned hereby certifies that the foregoing document was served in its entirety on December 22, 2023 upon the following parties via Electronic Mail:

Thomas N. Millikan (millikan-ptab@perkinscoie.com)

Babak Tehranchi (tehranchi-ptab@perkinscoie.com)

Andrew N. Klein (klein-ptab@perkinscoie.com)

Wei Yuan (yuan-ptab@perkinscoie.com)

Netskope-FortinetIPRTeam@perkinscoie.com

Date: December 22, 2023

By: /s/ James Glass

James M. Glass (Reg. No. 46,729)
QUINN EMANUEL URQUHART &
SULLIVAN LLP

51 Madison Avenue, 22nd Floor
New York, NY 10010

Email: jimglass@quinnemanuel.com

Phone: 212-849-7142

Fax: 212-849-7100

*Counsel for Patent Owner
Fortinet, Inc.*