

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TREND MICRO, INC.,
Petitioner,

v.

TAASERA LICENSING LLC,
Patent Owner.

IPR2023-00801
Patent 8,327,441 B2

Before JAMES P. CALVE, NEIL T. POWELL, and
STEVEN M. AMUNDSON, *Administrative Patent Judges*.

AMUNDSON, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
35 U.S.C. § 314

I. INTRODUCTION

Trend Micro, Inc. (“Petitioner”) filed a Petition requesting an *inter partes* review of claims 1–7 and 9 in U.S. Patent No. 8,327,441 B2 (Exhibit 1001, “the ’441 patent”) under 35 U.S.C. §§ 311–319.¹ Paper 2 (“Pet.”). Taasera Licensing LLC (“Patent Owner”) filed a Preliminary Response. Paper 6 (“Prelim. Resp.”). No further briefing was requested or authorized.

Under 37 C.F.R. § 42.4(a), we have authority to determine whether to institute an *inter partes* review. We may institute an *inter partes* review only if “the information presented in the petition filed under section 311 and any response filed under section 313 shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” 35 U.S.C. § 314(a) (2018). The “reasonable likelihood” standard is “a higher standard than mere notice pleading” but “lower than the ‘preponderance’ standard to prevail in a final written decision.” *Hulu, LLC v. Sound View Innovations, LLC*, IPR2018-01039, Paper 29 at 13 (PTAB Dec. 20, 2019) (precedential).

After considering the Petition, the Preliminary Response, and the evidence of record, and for the reasons explained below, we determine that Petitioner has not demonstrated a reasonable likelihood that it would prevail

¹ Although the Petition’s Introduction section identifies claims 1–12 as the challenged claims, the Petition’s Identification of Challenge section identifies claims 1–7 and 9 as the challenged claims. Pet. 1, 3. Additionally, Petitioner’s unpatentability arguments address only claims 1–7 and 9. *Id.* at 16–61. Further, Patent Owner recognizes claims 1–7 and 9 as the challenged claims. *See, e.g.*, Prelim. Resp. 1.

in proving the unpatentability of at least one claim challenged in the Petition. Hence, we deny the Petition and do not institute an *inter partes* review.

II. BACKGROUND

A. Real Parties in Interest

Petitioner identifies the following real parties in interest: Trend Micro, Inc., Trend Micro America, Inc., and Trend Micro Incorporated. Pet. 63–64. Patent Owner identifies itself as the sole real party in interest. Paper 3, 2. The parties do not raise any issue about real parties in interest.

B. Related Matters

Petitioner and Patent Owner identify the following civil actions as related matters:

- *Taasera Licensing LLC v. Trend Micro, Inc.*, No. 2:21-cv-00441-JRG (E.D. Tex.);
- *Trend Micro, Inc. v. Taasera Licensing LLC*, No. 2:22-cv-00303-JRG (E.D. Tex.);
- *Palo Alto Networks, Inc. v. Taasera Licensing LLC*, No. 2:22-cv-00314-JRG (E.D. Tex.);
- *Taasera Licensing LLC v. Musarubra US LLC*, No. 2:22-cv-00427-JRG (E.D. Tex.);
- *Taasera Licensing LLC v. CrowdStrike, Inc., et al.*, No. 2:22-cv-00468-JRG (E.D. Tex.); and
- *In re Taasera Licensing LLC, Patent Litigation*, No. 2:22-md-03042-JRG (E.D. Tex.) (the “Consolidated Litigation”).

Pet. 64; Paper 3, 2–3.

C. The '441 Patent (Exhibit 1001)

The '441 patent, titled “System and Method for Application Attestation,” issued on December 4, 2012, from an application filed on

February 17, 2012. Ex. 1001, codes (22), (45), (54). The patent states that the “disclosure relates to the field of data center virtualization and, more particularly, to a system and method to provide attestation of applications at runtime.” *Id.* at 1:16–18; *see id.* at code (57). The patent explains that software applications provided as service offerings, e.g., via a cloud infrastructure, present security challenges. *See id.* at 1:20–25, 5:64–67.

The ’441 patent discloses a system “to inspect web and non-web applications (e.g., all web and non-web applications)” running on an “instrumented target platform.” Ex. 1001, 1:52–56. The system may include “an attestation service (or broker) using an attestation server and configured to generate application artifacts and/or to issue application statements or claims.” *Id.* at 1:56–59. Application artifacts may represent “an identified application instance (e.g., at runtime, globally unique, opaquely and/or in a time-sensitive manner, for example, using an application artifact having a specified or predetermined lifetime) on an instrumented platform,” such as a mobile phone, personal computer, or gaming console. *Id.* at 1:59–2:8. Application statements may include “at least one statement (or claim) about an inspected runtime execution context and/or an intelligence based security context.” *Id.* at 2:8–11; *see id.* at 2:44–55, 9:40–46.

The ’441 patent explains that one or more embodiments verify applications “by signing-on or authenticating applications onto the network using” a multi-factor attestation process. Ex. 1001, 6:1–4. For example, in a two-factor attestation process, the two factors function as follows:

- (1) a first factor may “identify one or more attributes of the application (e.g., run time instance of the application) such as (1) a hash of the application, as an application

fingerprint, (2) other applications/processes called by the application, (3) ports used by the application, (4) computing, storage, and network resource usage by the application, and/or (5) the platform executing the application, among others”; and

- (2) a second factor may “identify what is known about the application based on the attributes of the application from security intelligence external to the operating environment (e.g., a third party (e.g., trusted third party) separate from the computing platform running the application).”

Id. at 6:4–12, 6:14–19.

The '441 patent's Figure 1 (reproduced below) depicts an exemplary system/architecture for one or more embodiments:

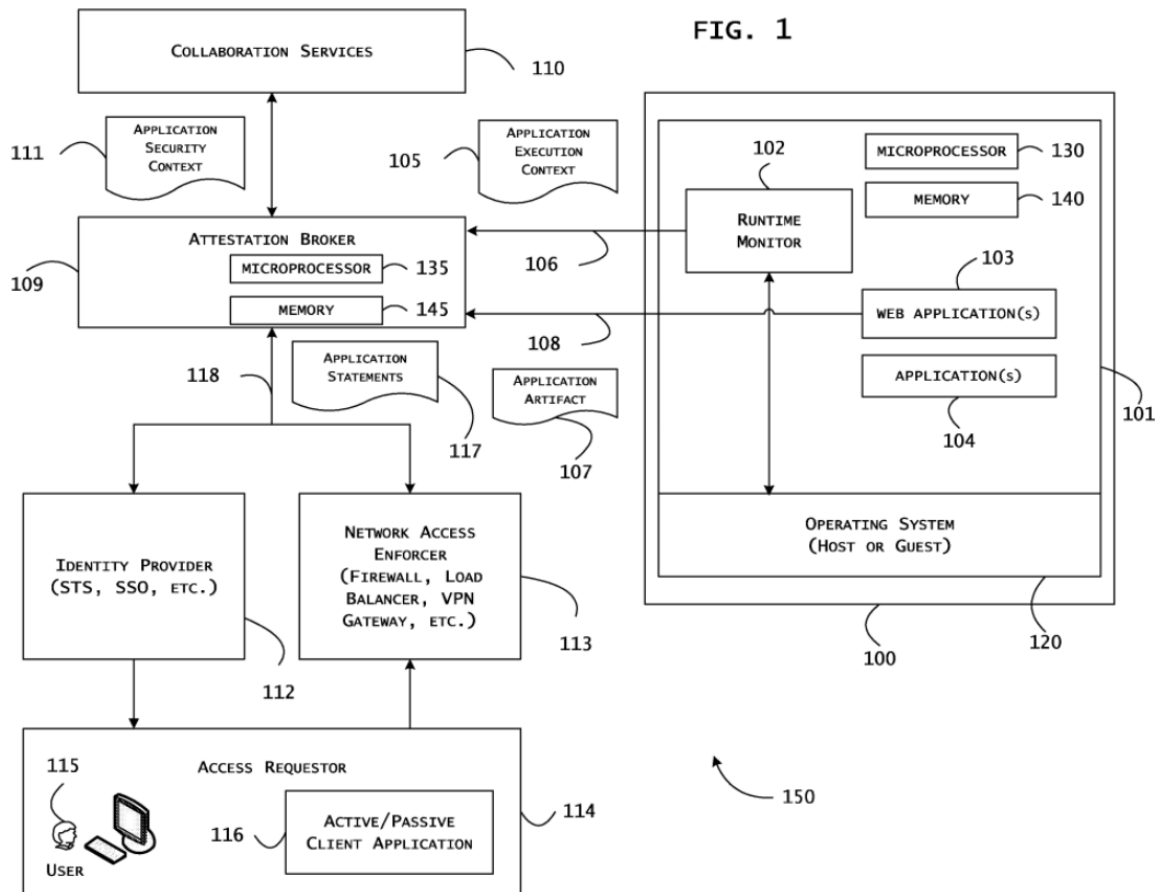


Figure 1 illustrates exemplary network 150 including the following components:

- instrumented computing platform 100 with computing device 101 containing runtime monitor 102, web-based application 103, and non-web-based application 104;
- attestation broker 109;
- collaboration services 110;
- identity provider 112, e.g., a Security Token Service (STS) or Single Sign On (SSO);
- network access enforcer 113, e.g., a network firewall, a load balancer, or a VPN gateway; and
- access requestor 114, e.g., another computing platform operated by user 115 and containing active or passive client application 116.

Ex. 1001, 6:62–7:12, 7:66–8:4, Fig. 1.

Client application 116 may desire an assurance or an attestation that a target application on computing platform 101 “is appropriate to be used.”

Ex. 1001, 7:7–12. Attestation broker 109 or collaboration services 110 may generate “application statements or reports 117” with “assertions about the running target application” and then deliver “the statements or reports 117 to the requestor, the third party, and/or end-user.” *Id.* at 7:13–17.

Collaboration services 110 may include (1) “an application whitelisting service[] (e.g., any application whitelisting services),” (2) “a vulnerability assessment service,” (3) “a patch management service,” (4) “an anti-virus service,” (5) “a predictive and behavioral analytics engine,” (6) an “orchestration service,” (7) “a network intrusion detection or prevention service,” (8) “an open flow switch controller,” and/or (9) “a malware analysis system.” Ex. 1001, 2:62–3:3.

Attestation broker 109 may issue “application artifacts 107 (e.g., that may maintain a record of the state of the discovered or identified applications running on the instrumented platform 100) to the runtime monitor 102 for discovered or identified applications running on the instrumented platform 100.” Ex. 1001, 7:27–32; *see id.* at 9:55–59. Runtime monitor 102 may use “application artifacts 107 to generate attribute value assertions (e.g., assertions regarding one or more attribute values associated with each respective running application) to describe the application execution context 105 of the applications.” *Id.* at 7:32–36.

Attestation broker 109 receives (1) “runtime execution context 105 indicating attributes of the application at runtime” and (2) “security context 111 providing security information about the application.” Ex. 1001, 7:37–41, 18:60–64, Fig. 8 (step 810). A runtime execution context may include any of the following:

- (1) “the file hash digest”;
- (2) “file attributes (for example, the file size, the publisher details, the creation and/or the modification timestamps, the digital signing, and/or the product version) for loaded components of the running target application”;
- (3) “parent-child process associations”;
- (4) “loader dependencies”;
- (5) “information about active listening and/or open network ports that may be well-known and/or ephemeral”; and/or
- (6) “executable file binaries.”

Id. at 11:42–52; *see id.* at 8:5–18, 14:26–31, 14:45–48, 16:9–13, 16:22–25.

A security context may include information received from a collaboration service, such as “evaluations and behavioral or predictive analytics based on

historic state information and measurements sampled over a period of time using a variety of inspection methods.” *Id.* at 9:40–46; *see id.* at 2:49–50, 2:59–61, 15:20–27. A security context may indicate “the level of risk (of confidence) associated with running the target application.” *Id.* at 11:61–63.

After receiving runtime execution context 105 and security context 111, attestation broker 109 generates application statement or report 117 as “an attestation result” that indicates “security risks associated with the application based on the received runtime execution context 105 and the received security context 111.” Ex. 1001, 7:41–45, 18:64–19:1, Fig. 8 (step 820); *see id.* at 8:21–26. Hence, “as an outcome of (or based on) the evaluation of the running target application 103 or 104 on the target instrumented platform 100 using collaboration services 110, the attestation broker 109 may issue application statements or reports 117 via application statement notification 118.” *Id.* at 8:21–26. Attestation broker 109 may provide “a confidence metric that may include one or more ratings (e.g., application confidence ratings for the targeted application).” *Id.* at 8:43–46.

After generating application statement or report 117, attestation broker 109 sends “the attestation result associated with the application,” e.g., to access requestor 114. Ex. 1001, 7:45–46, 19:1–3, Fig. 8 (step 830). Additionally, identity provider 112 and network access enforcer 113 may “subscribe for application statements or reports 117” from attestation broker 109. *Id.* at 7:66–8:4.

Among other things, an attestation broker may request that a collaboration service “perform a static, dynamic and/or virtual execution analysis of the executable file binary, for example, by a malware analysis system.” Ex. 1001, 14:45–48.

D. The Challenged Claims

Petitioner challenges independent method claim 1 and claims 2–7 and 9 that depend directly or indirectly from claim 1. Pet. 3, 16–61.

Claim 1 exemplifies the challenged claims and reads as follows (with bracketed numbers and letters added for reference purposes):²

1. [1a] A method of providing an attestation service for an application at runtime executing on a computing platform using an attestation server, comprising:

[1b] receiving, by the attestation server remote from the computing platform:

[1c] a runtime execution context indicating attributes of the application at runtime, wherein the attributes comprise one or more executable file binaries of the application and loaded components of the application; and

[1d] a security context providing security information about the application, wherein the security information comprises an execution analysis of the one or more executable file binaries and the loaded components;

[1e] generating, by the attestation server, a report indicating security risks associated with the application based on the received runtime execution context and the received security context, as an attestation result; and

[1f] sending, by the attestation server, the attestation result associated with the application.

Ex. 1001, 26:27–45.

² We use the same numbers and letters that Petitioner uses to identify the claim language. See Pet. 18–32, 45–54.

E. The Asserted References

For its challenges, Petitioner relies on the following references:

Name	Reference	Exhibit
Munetoh	Seiji Munetoh et al., <i>Integrity Management Infrastructure for Trusted Computing</i> , IEICE Transactions on Information and Systems, Vol. E91-D, No. 5, 1242–51 (May 2008) ³	1005
Rajan	Hridesh Rajan & Mahantesh Hosamani, <i>Tisa: Toward Trustworthy Services in a Service-Oriented Architecture</i> , IEEE Transactions on Services Computing, Vol. 1, No. 4, 201–13 (October-December 2008) ⁴	1006

Pet. 2–3, 11–61. Petitioner asserts that Munetoh and Rajan qualify as prior art under pre-AIA § 102(b). *Id.* at 2; *see* 35 U.S.C. § 102(b) (2006).⁵

At this stage of the proceeding, Patent Owner does not dispute that each reference qualifies as prior art. *See, e.g.*, Prelim. Resp. 5–9, 12–28.

F. The Asserted Challenges to Patentability

Petitioner asserts the following challenges to patentability:

Claim(s) Challenged	35 U.S.C. §	Reference(s)/Basis
1–7, 9	103(a)	Munetoh, Rajan
1–7, 9	103(a)	Rajan, Munetoh

³ The acronym “IEICE” stands for “Institute of Electronics, Information and Communication Engineers.”

⁴ The acronym “IEEE” stands for “Institute of Electrical and Electronics Engineers.”

⁵ The Leahy-Smith America Invents Act (“AIA”), Pub. L. No. 112-29, 125 Stat. 284 (2011), amended 35 U.S.C. § 102 and § 103 effective March 16, 2013. Because the filing date of the challenged claims predates the AIA’s amendments to § 102 and § 103, this decision refers to the pre-AIA versions of § 102 and § 103.

Pet. 2–3, 11–61.

G. Testimonial Evidence

To support its challenges, Petitioner relies on the declaration of Seth James Nielson, Ph.D. (Exhibit 1003). Dr. Nielson states, “I have been retained by counsel for Trend Micro, Inc. (‘Trend Micro’ or ‘Petitioner’) as a technical expert in connection with” this proceeding. Ex. 1003 ¶ 2.

At this stage of the proceeding, Patent Owner does not introduce testimonial evidence.

III. DISCRETIONARY DENIAL
IN VIEW OF PARALLEL LITIGATION

Under § 314(a), the Director possesses “broad discretion” in deciding whether to institute an *inter partes* review. *See* 35 U.S.C. § 314(a); *Saint Regis Mohawk Tribe v. Mylan Pharm. Inc.*, 896 F.3d 1322, 1327 (Fed. Cir. 2018). The Director is “permitted, but never compelled,” to institute an *inter partes* review. *Harmonic Inc. v. Avid Tech., Inc.*, 815 F.3d 1356, 1367 (Fed. Cir. 2016). The Board decides whether to institute an *inter partes* review on the Director’s behalf. 37 C.F.R. § 42.4(a) (2023).

Based on *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020) (precedential), Patent Owner argues that we should exercise our discretion under § 314(a) to deny institution in view of the Consolidated Litigation. *See* Prelim. Resp. 28–34; *supra* § II.B. Petitioner argues that we should decline to exercise our discretion under § 314(a) to deny institution. *See* Pet. 61–62.

For the reasons explained below, we determine that Petitioner has not demonstrated a reasonable likelihood that it would prevail in proving the unpatentability of at least one claim challenged in the Petition. *See infra*

§§ IV.D.3(g), IV.E.1(g). Hence, we do not consider discretionary denial under § 314(a).

IV. PATENTABILITY ANALYSIS

A. Legal Principles: Obviousness

A patent may not be obtained “if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.” 35 U.S.C. § 103(a) (2006). An obviousness analysis involves underlying factual inquiries including (1) the scope and content of the prior art; (2) differences between the claimed invention and the prior art; (3) the level of ordinary skill in the art; and (4) where in evidence, objective indicia of nonobviousness, such as commercial success, long-felt but unsolved needs, and failure of others.⁶ *Graham v. John Deere Co.*, 383 U.S. 1, 17–18, 35–36 (1966); *Apple Inc. v. Samsung Elecs. Co.*, 839 F.3d 1034, 1047–48 (Fed. Cir. 2016) (en banc). When evaluating a combination of references, an obviousness analysis should address “whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.” *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007).

We analyze the obviousness issues according to these principles.

B. Level of Ordinary Skill in the Art

Factors pertinent to determining the level of ordinary skill in the art include (1) the educational level of the inventor; (2) the type of problems encountered in the art; (3) prior-art solutions to those problems; (4) the

⁶ The parties do not address objective indicia of nonobviousness. *See, e.g.*, Pet. 11–61; Prelim. Resp. 12–28.

rapidity with which innovations are made; (5) the sophistication of the technology; and (6) the educational level of workers active in the field. *Envtl. Designs, Ltd. v. Union Oil Co.*, 713 F.2d 693, 696–97 (Fed. Cir. 1983). Not all factors may exist in every case, and one or more of these or other factors may predominate in a particular case. *Id.* These factors are not exhaustive, but merely a guide to determining the level of ordinary skill in the art. *Daiichi Sankyo Co. v. Apotex, Inc.*, 501 F.3d 1254, 1256 (Fed. Cir. 2007). Moreover, the prior art itself may reflect an appropriate skill level. *Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001).

Petitioner asserts that a person of ordinary skill in the art at the time of the alleged invention “would have had a bachelor’s degree in computer science, or a related field, or an equivalent technical degree or equivalent work experience, and an additional one to two years of education or experience in the field of computer security topics.” Pet. 10. Petitioner also asserts that “[m]ore education can supplement practical experience and vice versa.” *Id.* Dr. Nielson’s testimony supports Petitioner’s assertions. *See* Ex. 1003 ¶ 81.

For purposes of the Preliminary Response only, “Patent Owner utilizes Petitioner’s proposed level of skill in the art.” Prelim. Resp. 11–12 (quoting Pet. 10).

Based on the current record and for purposes of this decision, we accept Petitioner’s description of an ordinarily skilled artisan because it comports with the technology and claims of the ’441 patent as well as the asserted prior art.

C. Claim Construction

We construe claim terms “using the same claim construction standard” that district courts use to construe claim terms in civil actions under 35 U.S.C. § 282(b). *See* 37 C.F.R. § 42.100(b). Under that standard, claim terms “are given their ordinary and customary meaning, which is the meaning the term would have to a person of ordinary skill in the art at the time of the invention.” *Power Integrations, Inc. v. Fairchild Semiconductor Int’l, Inc.*, 904 F.3d 965, 971 (Fed. Cir. 2018) (citing *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312–13 (Fed. Cir. 2005) (en banc)). The meaning of claim terms may be determined by “look[ing] principally to the intrinsic evidence of record, examining the claim language itself, the written description, and the prosecution history, if in evidence.” *DePuy Spine, Inc. v. Medtronic Sofamor Danek, Inc.*, 469 F.3d 1005, 1014 (Fed. Cir. 2006) (citing *Phillips*, 415 F.3d at 1312–17).

For purposes of this proceeding only, “Petitioner proposes that all claim terms should be given their plain and ordinary meanings.” Pet. 10. Further, Petitioner asserts that “no specific claim constructions are necessary to apply the asserted prior art to” the challenged claims. *Id.*

For purposes of the Preliminary Response only, “Patent Owner believes that claim construction is not required to resolve any issues.” Prelim. Resp. 9. Additionally, Patent Owner notes that in the Consolidated Litigation Patent Owner and Petitioner have proposed constructions for certain claim language, including the following terms and limitations in claim 1:

Term/Limitation	Patent Owner's Proposed Construction	Petitioner's Proposed Construction
“attestation”	“verification”	“verifying/verifies the identity of an application”
“runtime”	“the time the application is running”	“the time the application is running”
“at runtime”	preamble limiting at least as to “at runtime”; meaning “at the time the application is running”	preamble limiting at least as to “at runtime”; meaning “at the time the application is running”
“receiving . . . a runtime execution context indicating attributes of the application at runtime, wherein the attributes comprise one or more executable file binaries of the application and loaded components of the application”	plain and ordinary meaning subject to the construction of “runtime”	receiving at the time the application is running an execution context that includes the executable file binaries of the application (as distinct from binary hashes) and loaded components of the application
“a security context providing security information about the application”	plain and ordinary meaning	“security information about the application provided by a collaboration service”

Id. at 9–11 (citing Ex. 2003 (Joint Claim Construction Chart), 39–55, 62–69).

“[O]nly those terms need be construed that are in controversy, and only to the extent necessary to resolve the controversy.” *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999); *see Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017

(Fed. Cir. 2017). Based on the current record, we determine that no claim term requires an explicit construction to decide whether Petitioner satisfies the “reasonable likelihood” standard for instituting trial.

D. Alleged Obviousness over Munetoh and Rajan: Claims 1–7 and 9

Petitioner contends that claims 1–7 and 9 are unpatentable under § 103(a) as obvious over Munetoh and Rajan. *See* Pet. 3, 16–43. Patent Owner disputes Petitioner’s contentions. *See* Prelim. Resp. 14–22, 28.

Below, we provide overviews of Munetoh and Rajan. Then, we consider the obviousness issues. For the reasons explained below, Petitioner does not establish sufficiently for purposes of institution that the combined disclosures in Munetoh and Rajan teach the subject matter in any of claims 1–7 and 9.

1. OVERVIEW OF MUNETOH (EXHIBIT 1005)

Munetoh is a paper titled “Integrity Management Infrastructure for Trusted Computing” by Seiji Munetoh et al., published in the IEICE Transactions on Information and Systems, Volume E91-D, Number 5 (May 2008). Ex. 1005, 1.⁷ Munetoh explains that “security tools based on software-rooted trust are sometimes not effective against rootkit and self-evolving malware that hide themselves and can sneak through the end-point protective mechanisms provided by the security tools.” *Id.* For example, a metamorphic virus that “modifies and encrypts virus code when copying itself is not detectable by protection mechanisms using blacklist pattern matching schemes.” *Id.*

⁷ For Exhibit 1005 (Munetoh), we follow Petitioner’s practice and cite to the page numbers that Petitioner applied to the exhibit rather than the page numbers that appear in the publication.

Munetoh also explains that an organization called the Trusted Computing Group (TCG) developed a “whitelist-based software component verification scheme” employing a “specific security chip” embedded in a computing platform. Ex. 1005, 1. In particular, “integrity data for each component running on the platform is recorded in the security chip and securely checked by remote attestation.” *Id.* That verification scheme “makes it possible to check the health of a computing platform” using remote attestation “in a more secure and trusted manner and to alert the user whenever the platform is contaminated by unknown software.” *Id.*

To that verification scheme, Munetoh adds a verification mechanism employing a trusted third party called a Platform Validation Authority (PVA). Ex. 1005, 2. The PVA “verifies the composition of each target platform reported in remote attestations” including platforms with “varieties of running software components.” *Id.* The PVA uses a Platform Trust Services (PTS) component that “deals with the integrity information of a platform” and a trusted platform module (TPM) that “measure[s] and report[s] platform integrity in a manner that cannot be compromised even by the platform owners or the software running on it.” *Id.* at 1–2.

The TPM includes Platform Configuration Registers (PCRs) that “record the integrity measurements,” such as hash values. Ex. 1005, 2–3. Each “PCR can store the fingerprints of multiple components using a hash-chain mechanism.” *Id.* at 3.

Munetoh discloses two types of integrity measurements for obtaining “the fingerprints of the hardware and software components that make up a platform”: (1) “the TCG trusted bootstrap” and (2) “other integrity measurements built on top of the trusted bootstrap.” Ex. 1005, 2. The

trusted bootstrap “comprises an iterative process of ‘measurement’, loading, and execution of software components.” *Id.* In particular, when powered on, “the immutable initial bootstrap code” on a client platform measures the next component on the client platform using a “hash value of the binary image” of the next component. *Id.* Then, “each software component recursively measures the next component” until “the operating system is loaded.” *Id.* The PCRs on the client platform “record the integrity measurements,” i.e., the hash values of the binary images. *Id.* at 2–3. After the operating system is loaded, a check of the hash values stored in the PCRs determines whether “any bits of the measured components are modified from the originals.” *Id.* at 3.

Although the “trusted bootstrap provides a basis for platform integrity measurement,” it “does not prove all aspects of trustworthiness” because various “modules and application programs that are loaded after the bootstrap process contribute to the trustworthiness of the platform.” Ex. 1005, 3. Thus, “other integrity measurements built on top of the trusted bootstrap” include “runtime integrity measurements on a computing platform with TPM.” *See id.* at 2, 4.

Typically, a computing platform with TPM seeks some interaction with a relying party, such as a service or an access to some resources, and the relying party needs to trust “the integrity of the computing platform.” Ex. 1005, 4. To permit integrity verification, the computing platform sends an integrity report to the relying party acting as a verifier. *Id.* When verifying an integrity report from the computing platform, the relying party (verifier) uses integrity metrics (hash values) called “reference manifests” (RMs) generated by hardware manufacturers and software vendors as “the

authoritative information to evaluate the integrity report.” *Id.*; *see id.* at 5 (explaining that “vendors release software with RMs”). In particular, the relying party (verifier) compares the runtime measurements (hash values) in an integrity report to the reference manifests (hash values). *Id.* at 4; *see id.* at 7.

An integrity report “represents a snapshot of the platform integrity measurement.” Ex. 1005, 5. An integrity report includes “the hash values of the files measured” and may include “assertions made by the components running on the client platform.” *Id.* at 5, 7.

Munetoh’s Figure 1 (reproduced below) depicts a client-server architecture with a PTS component on a client and a PTS component on a server acting as a Platform Validation Authority (PVA):

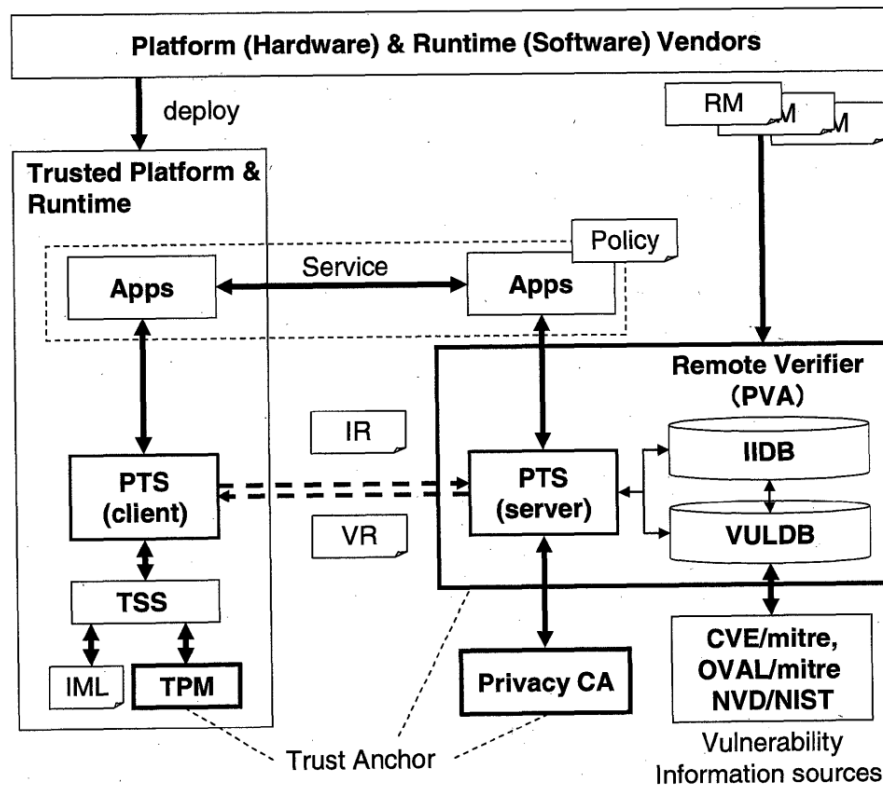


Fig. 1 PTS Client/Server architecture.

Figure 1 illustrates the following:

- a client with a PTS component, a TCG software stack (TSS), an integrity measurement log (IML), and a trusted platform module (TPM);
- a server-PVA with a PTS component, an integrity information database (IIDB), and a vulnerability database (VULDB);
- the server-PVA receiving reference manifests (RMs) generated by hardware manufacturers and software vendors; and
- the server-PVA receiving vulnerability information from external sources, such as Common Vulnerabilities and Exposures (CVE) identifiers “provided by MITRE Corporation” and the National Vulnerability Database (NVD) provided by the National Institute of Standards and Technology (NIST).

Ex. 1005, 4–5, 7.

The IIDB “maintains a whitelist (or blacklist) of hash values of the known components.” Ex. 1005, 6. The VULDB “is the central repository of the known vulnerabilities, by aggregating information from vendors and security advisories.” *Id.*

In Figure 1’s client-server architecture, the PTS components permit measurement and verification of all applications running on a client platform. Ex. 1005, 5, 8. Specifically, the client-side PTS generates an integrity report (IR in Figure 1) by collecting (1) hash values stored in the PCRs on the client platform and (2) information from an integrity measurement log (IML in Figure 1) containing “a list of components loaded on the platform and measured by the TPM, and how these components are composed.” *Id.* at 3–4; *see id.* at 6 (discussing the “PCR values and the IML in the IR”). The client-side PTS sends the integrity report to the server-side

PTS. *Id.* at 4–5. The server-side PTS verifies the integrity report from the client-side PTS by comparing the measurements in the integrity report to reference manifests. *See id.* at 4–5, 7. Then, the server-side PTS returns a verification result (VR in Figure 1) to the client-side PTS. *Id.* at 5.

The verification result has one of three values: (1) “valid” meaning that “the IR is verified as trusted”; (2) “invalid” meaning that “the IR is verified to be invalid”; and (3) “unverified” meaning that “there is not enough information for making a decision.” Ex. 1005, 5 (emphases omitted). If the verification result is “invalid” or “unverified,” the communication from the server-side PTS to the client-side PTS includes a “detailed reason.” *Id.*

Munetoh discloses a finite state machine (FSM) behavior model that represents the trusted-bootstrap process and “subsequent measurement processes” of a client platform. Ex. 1005, 6. “Each state in the behavior model represents a platform state, and each transition represents the integrity of the component that is loaded and executed.” *Id.* The behavior model describes how the PCR values and the integrity measurement log in an integrity report “are measured” and “how they should be verified.” *Id.*

Munetoh also discloses using vulnerability information from external sources to generate a report that “describes the presence of vulnerabilities and the degrees of security risks.” Ex. 1005, 7. For instance, an Open Vulnerability and Assessment Language (OVAL) interpreter “collects the detailed states of the platform, tests them against the OVAL definitions, and reports the results of the tests.” *Id.*

2. OVERVIEW OF RAJAN (EXHIBIT 1006)

Rajan is a paper titled “Tisa: Toward Trustworthy Services in a Service-Oriented Architecture” by Hridesh Rajan and Mahantesh Hosamani, published in the IEEE Transactions on Services Computing, Volume 1, Number 4, (October-December 2008). Ex. 1006, 1.⁸ Rajan explains that service-oriented computing (SoC) is “a new programming paradigm that promotes dealing with modules or components,” often called services, that may form “larger applications or solutions.” *Id.* (endnote omitted). Rajan describes a verification mechanism based on a trusted platform module (TPM) “used as a root of trust in the presence of distributed services in a service-oriented computing environment.” *Id.* at 2. “A TPM can extend its trust to higher layers of the system by building a chain of trust starting from the hardware and subsequently linking upper layers.” *Id.*

To implement the verification mechanism, Rajan provides architectural extensions and algorithms for the following three types of entities: clients, brokers, and providers. *See* Ex. 1006, 1–4. For instance, each entity already includes a service-negotiation interface, and Rajan adds a trust-negotiation and verification interface to each entity. *Id.* at 1, 3.

⁸ For Exhibit 1006 (Rajan), we follow Petitioner’s practice and cite to the page numbers that Petitioner applied to the exhibit rather than the page numbers that appear in the publication.

Rajan's Figure 1 (reproduced below) depicts architectural extensions for a client, a broker, and a provider:

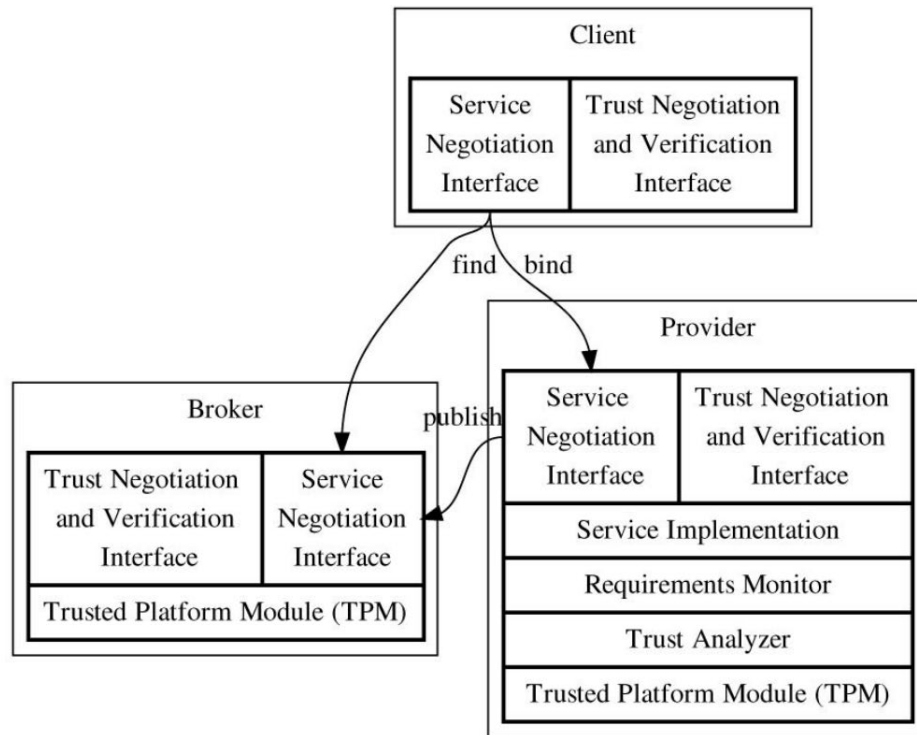


Fig. 1. Proposed architectural extensions.

Figure 1 illustrates the following:

- a client including (i) a service-negotiation interface and (ii) a trust-negotiation and verification interface;
- a broker including (i) a service-negotiation interface, (ii) a trust-negotiation and verification interface, and (iii) a trusted platform module; and
- a provider including (i) a service-negotiation interface, (ii) a trust-negotiation and verification interface, (iii) a service implementation (software application), (iv) a requirements monitor, (v) a trust analyzer, and (vi) a trusted platform module.

Ex. 1006, 4.

The broker “negotiates the desired integrity-level with the provider and provides assurance to the client using the trust-negotiation and verification interface.” Ex. 1006, 3. The trusted platform module on the broker attests to the broker’s identity. *Id.* The trusted platform module on the provider attests to the provider’s identity and “the integrity of the provider’s software stack.” *Id.* at 3, 6. The requirements monitor on the provider verifies conformance of the service implementation (software application) to “the desired nonfunctional requirements” in a service request from the client. *Id.* at 4. The trust analyzer on the provider “serves to verify the integrity of the requirements monitor” and “provides a report that includes an immutable integrity measurement of the local architecture along with a time stamp to indicate the freshness of the measurement.” *Id.* at 6; *see id.* at 4.

Rajan’s Figure 2 (reproduced below) depicts interactions between a client, a broker, and a provider:

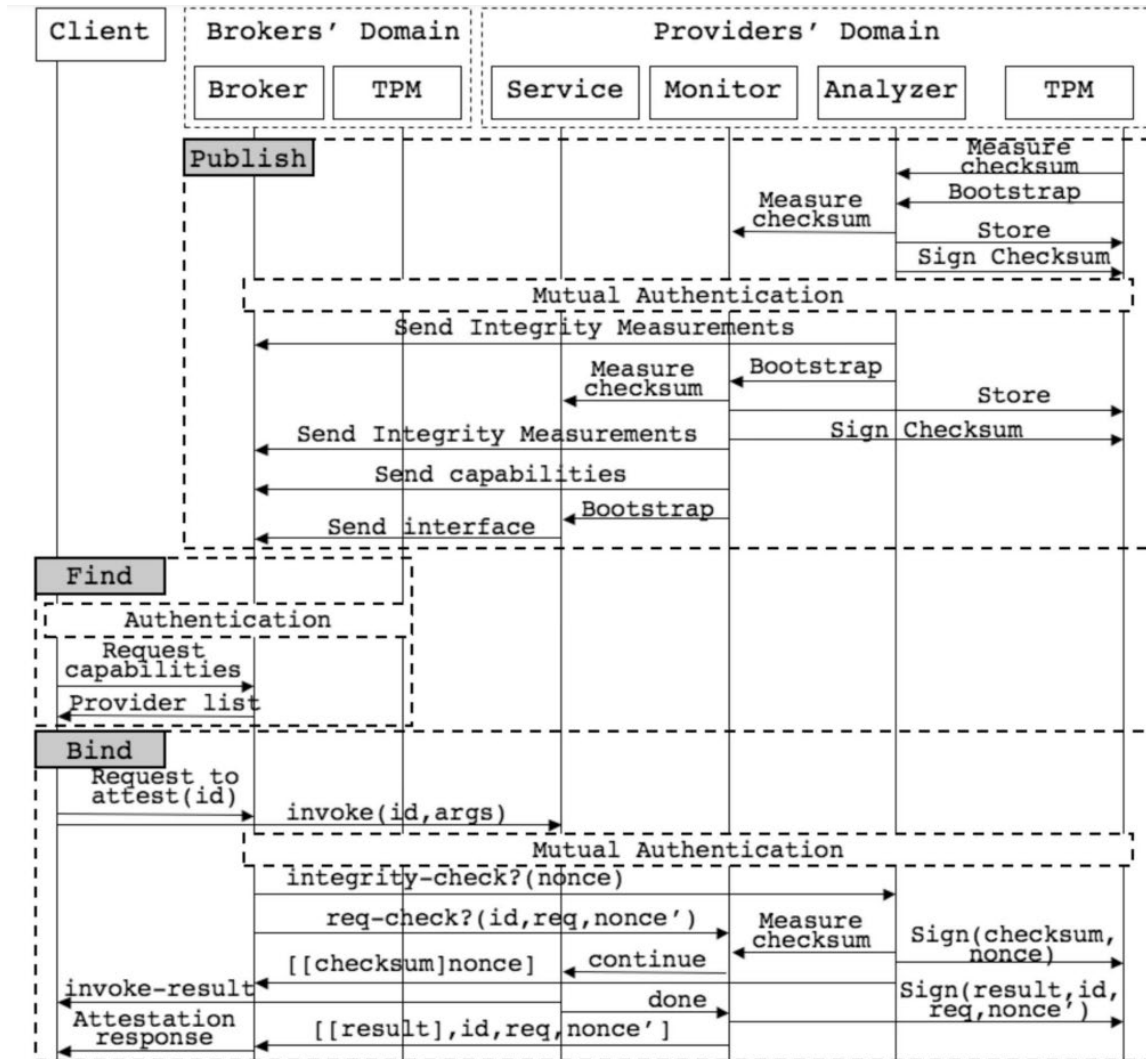


Fig. 2. Overview of Interaction between Components.

Figure 2 illustrates three steps: (1) a publish step, (2) a find step, and (3) a bind step. Ex. 1006, 5; *see id.* at 6–9.

During the publish step, the trust analyzer and the requirements monitor on the provider “send integrity measurements to the broker after mutual authentication,” and the broker stores the integrity measurements for future reference. Ex. 1006, 4, 7. During the publish step, the broker also receives the “capabilities of the requirements monitor available to the

broker,” i.e., “the type of nonfunctional requirements that the monitor can verify.” *Id.* at 3; *see id.* at 7.

During the find step, the client sends “a request to the broker for services that satisfy desired functional properties as well as provide monitoring capabilities to check desired nonfunctional properties.” Ex. 1006, 3–4. During the find step, the broker responds to the client’s request with “a list of only those services that satisfy both functional requirements as well as provide monitoring capabilities for nonfunctional requirements.” *Id.* at 4.

During the bind step, the client sends a service request to the provider for a service implementation (software application) and sends an attestation request to the broker for an assurance that the provider “satisfied the previously agreed upon nonfunctional requirements during the service request.” Ex. 1006, 4; *see id.* at 7. The “integrity of a transaction is assumed only if an assurance is received from the broker.” *Id.* at 7–8. “Upon receiving this assurance, the results of the service can be used as intended.” *Id.* at 8.

The cryptographic capabilities of the trusted platform modules on the provider and the broker are “leveraged to establish a secure communication channel between” the provider and the broker. Ex. 1006, 5. “This channel is used to convey the integrity measurements to the broker.” *Id.*

Rajan discloses a trace-based requirements monitor where (1) the requirements are “specified as logical formulas over program traces” or “expressed as fields and methods” and (2) the monitor “determines whether the trace of service execution satisfied the desired nonfunctional requirements” in a service request from the client. Ex. 1006, 5–6; *see id.*

at 8–9. As an example, a trace-based requirement for a payment-processing service may specify that “in no trace of the service implementation, the argument credit card number directly or indirectly flows to a persistence-related method, e.g., File.write(...).” *Id.* at 5–6 (ellipsis in original). As another example, a trace-based requirement for a Web service may specify that “the execution trace of the program involving the variables and methods dealing with client data labeled as sensitive, should not include APIs dealing with persistence or serialization.” *Id.* at 6.

With a trace-based requirements monitor, the integrity measurements sent from the provider to the broker include (1) a trace “generated and maintained by the requirements monitor” on the provider and (2) a “static checksum of the software stack” on the provider “up to the requirements monitor.” Ex. 1006, 6–7. The “static checksum is created by computing the SHA1 hash of the software stack” on the provider “up to the requirements monitor” and is “essential to validate the current state of the service implementation.” *Id.* at 7. In particular, the requirements monitor’s integrity “has not been violated if its checksum as computed by the trust analyzer and signed by” the trusted platform module matches clean-room measurements. *Id.* at 2; *see id.* at 12.

Further, the broker checks “the trace received from the provider” to determine “if it contains a sequence that violates the requirements” from the client. Ex. 1006, 7. The trace received from the provider during the publish step “represents the normal execution of the service implementation.” *Id.* A comparison of the trace received from the provider during the publish step to the current trace “quickly identif[ies] aberrations in the current trace.” *Id.*

Rajan's Figures 4 and 5 (reproduced below) depict example traces generated by a trace-based requirements monitor for a Web service:

```
ENTRY Connect.initialize()
PUT  Connect.name="John_Doe"
...
GET  Connect.name "John_Doe"
...
Order successfully processed: John Doe
```

Fig. 4. Output trace for the original service.

```
ENTRY com.mysql.jdbc.Statement.executeQuery
("SHOW_VARIABLES")
ENTRY com.mysql.jdbc.Connection.prepare
Statement
("INSERT INTO _cnumbers (name, ccn, cvc) _VALUES
(?, ?, ?) ")
...
```

Fig. 5. Trace for compromised Web service.

Above on the left, Figure 4 shows “the trace generated for the clean-room Web service.” Ex. 1006, 8. Above on the right, Figure 5 shows “the execution trace of the compromised Web service.” *Id.*

3. INDEPENDENT CLAIM 1

(a) Preamble 1a

Claim 1 recites “[a] method of providing an attestation service for an application at runtime executing on a computing platform using an attestation server.” Ex. 1001, 26:27–29.

Petitioner contends that Munetoh teaches claim 1's preamble because Munetoh discloses the following:

- (1) “a client-server architecture where the client is the platform (with applications) to be verified/attested and the server is the remote verifier or Platform Validation Authority (PVA)”;
- (2) both the client and the server include Platform Trust Services (PTS) elements that “between them determine whether or not to validate or attest the client platform”;
- (3) the server-side PTS receives “attestation data,” including measurements of “applications running on the platform,” from the client-side PTS;
- (4) the server-side PTS confirms “the integrity of all components on the platform” based on (i) the information

from the client-side PTS and (ii) “reference manifests for the applications”; and

- (5) the server-side PTS sends “a validation result” to the client-side PTS.

Pet. 19–21; *see id.* at 18–19. Further, Petitioner asserts that the server-PVA “trusts an application on the client (computer platform) based on the attestation data sent by the client.” *Id.* at 20.

Petitioner identifies the server-PVA (with the server-side PTS) as an “attestation server” providing an “attestation service” that “measures and attests to applications on” a client where at least one of the applications is “an application at runtime executing on a computing platform” according to claim 1’s preamble. *See* Pet. 19–21.

At this stage of the proceeding, Patent Owner makes no arguments specific to claim 1’s preamble. *See, e.g.,* Prelim. Resp. 15–22. Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015).

Generally, a preamble does not limit a claim. *Allen Eng’g Corp. v. Bartell Indus., Inc.*, 299 F.3d 1336, 1346 (Fed. Cir. 2002). We need not decide whether claim 1’s preamble limits the claim because, based on the current record and for the reasons advanced by Petitioner and supported by Dr. Nielson’s testimony, Petitioner establishes sufficiently that Munetoh teaches claim 1’s preamble. *See* Pet. 18–21; Ex. 1003 ¶¶ 123–132.

(b) Limitation 1b

Claim 1 recites “receiving, by the attestation server remote from the computing platform.” Ex. 1001, 26:30–31 (limitation 1b).

Petitioner contends that Munetoh teaches limitation 1b because Munetoh discloses that the server-PVA receives information from a client, and thus receives information from a “computing platform.” Pet. 21. Further, Petitioner asserts that the server-side PTS in the server-PVA complies with the Trusted Computing Group (TCG) standard for component verification. *Id.* According to Petitioner, the server-side PTS (1) receives integrity reports from the client-side PTS, (2) “verifies the integrity reports” based on “reference manifests (RMs) received at the server from software vendors,” and (3) “returns a Verification Result (VR) or Validation Result (used interchangeably in Munetoh) (corresponding to ‘attestation’).” *Id.* at 21–22.

At this stage of the proceeding, Patent Owner makes no arguments specific to limitation 1b. *See, e.g.,* Prelim. Resp. 15–22.

Based on the current record and for the reasons advanced by Petitioner and supported by Dr. Nielson’s testimony, Petitioner establishes sufficiently that Munetoh teaches limitation 1b. *See* Pet. 21–22; Ex. 1003 ¶¶ 145–152.

(c) Limitation 1c

Claim 1 recites receiving “a runtime execution context indicating attributes of the application at runtime, wherein the attributes comprise one or more executable file binaries of the application and loaded components of the application.” Ex. 1001, 26:32–35 (limitation 1c).

(i) Petitioner’s Contentions

Petitioner contends that Munetoh teaches limitation 1c for two related reasons. *See* Pet. 22–24. First, Petitioner asserts that an integrity report sent from the client-side PTS to the server-side PTS “represents a snapshot of the

platform integrity measurement” and may include “assertions made by the components running on the client platform.” *Id.* at 22 (quoting Ex. 1005, 5). Petitioner asserts that an integrity report corresponds to the claimed “runtime execution context indicating attributes of the application at runtime.” *Id.* Petitioner further asserts that Platform Configuration Registers (PCRs) on the client platform store “the hashes of binary components” on the client platform “as a fingerprint in the chain of trust.” *Id.* According to Petitioner, “a running application can be determined to be trusted based on its binary code being unmodified.” *Id.*

Second, Petitioner asserts that Munetoh teaches that the received attributes comprise “one or more executable file binaries of the application and loaded components of the application” according to limitation 1c because Munetoh describes a trusted-bootstrap process that “comprises an iterative process of ‘measurement’, loading, and execution of software components.” Pet. 23 (quoting Ex. 1005, 2). According to Petitioner, the “software components” executed during Munetoh’s trusted-bootstrap process correspond to “one or more executable file binaries of the application and loaded components of the application.” *Id.*

Further, Petitioner asserts that an integrity report sent from the client-side PTS to the server-side PTS includes “the software boot measurements and the application measurements.” Pet. 24. Petitioner also asserts that an integrity report may include “additional information,” such as an integrity measurement log containing “a list of components loaded on the platform and measured by the TPM [trusted platform module], and how these components are composed.” *Id.* (quoting Ex. 1005, 3) (citing Ex. 1003 ¶ 163).

(ii) Patent Owner's Contentions

Patent Owner disputes that Munetoh teaches limitation 1c. *See* Prelim. Resp. 15–18. Specifically, Patent Owner asserts that “Petitioner and Munetoh are silent on executable file binaries of the application and loaded components of the application.” *Id.* at 15 (emphases omitted). Patent Owner also asserts that Munetoh’s trusted-bootstrap process “comprises actions” and does not correspond to “executable file binaries and loaded components.” *Id.* at 15–16. According to Patent Owner, Munetoh does not disclose “executable file binaries, let alone executable file binaries of the application for the purpose of analysis.” *Id.* at 17 (emphasis omitted).

Further, Patent Owner asserts that the ’441 patent discloses “executable file binaries that are the subject of an execution analysis, such as by a malware analysis system, and that executable file binaries are related to running processes of an application on an instrumented target platform, but are not processes of the platform itself.” Prelim. Resp. 16. As support, Patent Owner quotes from the patent as follows: “The attestation broker 402 may also request on [sic] or more collaboration services 403 to perform a static, dynamic and/or virtual execution analysis of the executable file binary, for example, by a malware analysis system.” *Id.* (alteration by Patent Owner) (quoting Ex. 1001, 14:45–48).

Additionally, Patent Owner contends that “Munetoh only discloses monitoring loaded components on the platform rather than components of the application.” Prelim. Resp. 17 (emphases omitted). According to Patent Owner, “Petitioner glosses over this apparent distinction between loaded components on the platform versus loaded components of the application.” *Id.* at 18 (emphases omitted).

(iii) Analysis

Based on the current record, Petitioner does not establish sufficiently for purposes of institution that Munetoh teaches limitation 1c. *See* Pet. 22–24; Ex. 1003 ¶¶ 158–165. In particular, we disagree with Petitioner that Munetoh teaches that the received attributes comprise “one or more executable file binaries of the application and loaded components of the application” according to limitation 1c. *See* Pet. 23–24.

Munetoh’s trusted-bootstrap process “comprises an iterative process of ‘measurement’, loading, and execution of software components.” Ex. 1005, 2. In particular, when powered on, “the immutable initial bootstrap code” on a client platform measures the next component on the client platform using a “hash value of the binary image” of the next component. *Id.* Then, “each software component recursively measures the next component” until “the operating system is loaded.” *Id.* The PCRs on the client platform “record the integrity measurements,” i.e., the hash values of the binary images. *Id.* at 2–3.

Munetoh’s trusted-bootstrap process takes place on a client platform. Ex. 1005, 2–3. Petitioner maps the claimed “one or more executable file binaries of the application and loaded components of the application” to the “software components” executed during Munetoh’s trusted-bootstrap process. Pet. 23. Petitioner maps the claimed “attestation server” to Munetoh’s server-PVA. *Id.* at 19, 21, 27. But Petitioner fails to show that Munetoh’s server-PVA receives the “software components” executed during Munetoh’s trusted-bootstrap process. *See id.* at 22–24; Ex. 1003 ¶¶ 158–165.

Additionally, Munetoh discloses that the client-side PTS generates an integrity report (IR) by collecting (1) hash values stored in the PCRs on the client platform and (2) information from an integrity measurement log (IML) containing “a list of components loaded on the platform and measured by the TPM, and how these components are composed.” Ex. 1005, 3–4; *see id.* at 6 (discussing the “PCR values and the IML in the IR”); Ex. 1003 ¶¶ 149, 172. The client-side PTS sends the integrity report to the server-side PTS. Ex. 1005, 4–5; *see* Ex. 1003 ¶¶ 152, 165.

Based on the current record, Petitioner fails to show that an integrity report sent from the client-side PTS to the server-side PTS includes “one or more executable file binaries of the application and loaded components of the application” according to limitation 1c. *See* Pet. 22–24; Ex. 1003 ¶¶ 158–165. As for the hash values included in an integrity report, the ’441 patent distinguishes hash values from executable file binaries and loaded components. *See* Ex. 1001, 8:5–13, 8:16–18, 11:42–52, 14:26–31, 14:45–48, 16:9–13, 16:22–25. For example, the patent explains that a runtime execution context may include any of the following:

- (1) “the file hash digest”;
- (2) “file attributes (for example, the file size, the publisher details, the creation and/or the modification timestamps, the digital signing, and/or the product version) for loaded components of the running target application”; and/or
- (3) “executable file binaries.”

Id. at 11:42–52; *see id.* at 8:5–18.

Further, in a related litigation, Petitioner asserted that “binary hashes” are distinct from executable file binaries. Ex. 2003, 62–63.

As for the information from an integrity measurement log included in an integrity report, i.e., “a list of components loaded on the platform and measured by the TPM, and how these components are composed,” Petitioner fails to show how that information corresponds to “one or more executable file binaries of the application and loaded components of the application” according to limitation 1c. *See* Pet. 22–24; Ex. 1003 ¶¶ 158–165.

(d) Limitation 1d

Claim 1 recites receiving “a security context providing security information about the application, wherein the security information comprises an execution analysis of the one or more executable file binaries and the loaded components.” Ex. 1001, 26:36–39 (limitation 1d).

(i) Petitioner’s Contentions

Petitioner contends that Munetoh teaches limitation 1d and alternatively that the combined disclosures in Munetoh and Rajan teach limitation 1d. *See* Pet. 24–27. Regarding Munetoh alone, Petitioner contends that Munetoh discloses:

- (1) collecting “client-side integrity data into an integrity measurement log (IML) that includes a real time log of state changes in applications of the platform”;
- (2) recording “the integrity data for each component” running on a client platform in a security chip on the client platform; and
- (3) securely checking the integrity data “by remote attestation” with the server-PVA.

Id. at 24–25 (quoting Ex. 1005, 1) (citing Ex. 1003 ¶ 172). Petitioner contends that the integrity measurement log corresponds to “a security context providing security information about the application” according to limitation 1d. *Id.* at 24.

Further, Petitioner asserts that Munetoh teaches “the security information comprises an execution analysis of the one or more executable file binaries and the loaded components” according to limitation 1d because Munetoh discloses that:

- (1) the integrity measurement log includes “a list of components loaded on the platform and measured by the TPM” (trusted platform module) and “how they are composed”;
- (2) “a finite state machine (FSM) that analyzes” the integrity report (including the integrity measurement log) from the client-side PTS “based on a behavioral model that represents platform states and ‘the integrity of the component that is loaded and executed’”; and
- (3) the server-PVA “uses this information to ‘verif[y] the security state from the latest components.’”

Pet. 25–26 (alteration by Petitioner) (quoting Ex. 1005, 3, 6–7).

Regarding the combined disclosures in Munetoh and Rajan, Petitioner contends that Rajan teaches “an execution analysis” of the applications running on a computer platform because Rajan discloses a requirements monitor on a provider that “traces the operations of running programs and traces variables, methods, and arguments of the programs” as they execute “to ensure information security during runtime.” Pet. 26–27 (citing Ex. 1003 ¶¶ 176–178; Ex. 1006, 6, 8). Further, Petitioner asserts that Rajan discloses “sending the trace data (i.e., the execution analysis)” to a broker that corresponds to the claimed “attestation server.” *Id.* at 27; *see id.* at 21, 26–27.

(ii) Patent Owner’s Contentions

Patent Owner disputes that Munetoh teaches limitation 1d and that the combined disclosures in Munetoh and Rajan teach limitation 1d. *See Prelim.*

Resp. 19–21. Regarding Munetoh alone, Patent Owner asserts that Munetoh does not disclose “one or more executable file binaries of the application and loaded components of the application” according to limitation 1c and, therefore, cannot disclose “an execution analysis of the one or more executable file binaries and the loaded components” according to limitation 1d. *Id.* at 19. Patent Owner also asserts that Munetoh’s disclosures about the integrity measurement log, the integrity report from the client-side PTS to the server-side PTS, and the finite state machine (FSM) “fail to mention analyzing executable file binaries or loaded components, let alone executable file binaries or loaded components of the application.” *Id.* at 19–20 (emphasis omitted).

Further, Patent Owner asserts that “Rajan does not cure the deficiencies of Munetoh” because Rajan’s requirements monitor provides traces “used for the purpose of compliance with certain policies or ‘requirements [that] are specified as logical formulas.’” Prelim. Resp. 20 (alteration by Patent Owner) (quoting Ex. 1006, 5). Patent Owner also asserts that Rajan (1) “explicitly describes that it does not measure applications” and (2) “is devoid of any mention of executable file binaries and loaded components of the application.” *Id.* at 21 (citing Ex. 1006, 10).

(iii) Analysis

Based on the current record, Petitioner establishes sufficiently that Munetoh teaches limitation 1d and alternatively that the combined disclosures in Munetoh and Rajan teach limitation 1d. *See* Pet. 24–27; Ex. 1003 ¶¶ 170–179. Regarding Munetoh alone, Munetoh discloses that the client-side PTS generates an integrity report (IR) by collecting (1) hash values stored in the PCRs on the client platform and (2) information from an

integrity measurement log (IML) containing “a list of components loaded on the platform and measured by the TPM, and how these components are composed.” Ex. 1005, 3–4; *see id.* at 6 (discussing the “PCR values and the IML in the IR”); Ex. 1003 ¶¶ 149, 172. The client-side PTS sends the integrity report to the server-side PTS. Ex. 1005, 4–5; *see* Ex. 1003 ¶¶ 152, 165.

The hash values stored in the PCRs on the client platform and contained in an integrity report include hash values resulting from the trusted-bootstrap process and hash values resulting from “runtime integrity measurements” of software components on the client platform. Ex. 1005, 2–4; *see* Ex. 1003 ¶¶ 118, 129–130, 161–163, 165, 173. The trusted-bootstrap process includes executing software components to generate hash values of the binary images of the software components. Ex. 1005, 2–3; *see* Ex. 1003 ¶ 162.

Hence, the hash values stored in the PCRs on the client platform and contained in an integrity report correspond to “an execution analysis of the one or more executable file binaries and the loaded components” according to limitation 1d. *See* Ex. 1003 ¶¶ 170–174. Accordingly, we disagree with Patent Owner that Munetoh’s disclosures about the integrity report from the client-side PTS to the server-side PTS “fail to mention analyzing executable file binaries or loaded components.” *See* Prelim. Resp. 19–20.

Further, because a check of the hash values stored in the PCRs on the client platform and contained in an integrity report determines whether “any bits of the measured components are modified from the originals,” those hash values provide “security information” according to limitation 1d. Ex. 1005, 3; *see* Ex. 1003 ¶ 172.

Additionally, as Petitioner contends, Rajan teaches “an execution analysis” of the applications running on a computer platform because Rajan discloses a requirements monitor on a provider that “traces the operations of running programs and traces variables, methods, and arguments of the programs” as they execute “to ensure information security during runtime.” Ex. 1006, 4–8; *see* Ex. 1003 ¶¶ 176–178; Pet. 26–27. Additionally, Rajan explains that the integrity measurements sent from a provider to a broker include traces “generated and maintained by the requirements monitor” on the provider. Ex. 1006, 7; *see id.* at 4; Ex. 1003 ¶ 178.

Patent Owner’s assertion that Rajan “explicitly describes that it does not measure applications” mischaracterizes Rajan because Rajan distinguishes between directly measuring applications and indirectly measuring applications. *See* Ex. 1006, 10; Prelim. Resp. 21 (citing Ex. 1006, 10). In particular, Rajan explains the “static and inflexible” nature of remote attestation “does not affect the viability of” its architecture because its architecture monitors “the specific requirements of the applications that are not supposed to change” instead of “directly measuring the applications.” Ex. 1006, 10. For instance, a requirements monitor for a Web service may determine whether “the execution trace of the program involving the variables and methods dealing with client data labeled as sensitive” improperly “include[s] APIs dealing with persistence or serialization.” *Id.* at 6.

(e) Limitation 1e

Claim 1 recites “generating, by the attestation server, a report indicating security risks associated with the application based on the

received runtime execution context and the received security context, as an attestation result.” Ex. 1001, 26:40–43 (limitation 1e).

(i) Petitioner’s Contentions

Petitioner contends that Munetoh teaches limitation 1e and alternatively that the combined disclosures in Munetoh and Rajan teach limitation 1e. *See* Pet. 27–31. Regarding Munetoh alone, Petitioner asserts that the server-PVA:

- (1) analyzes data in the integrity report (including the integrity measurement log) from the client-side PTS;
- (2) determines whether to trust and validate the client platform that sent the integrity report; and
- (3) “issues a validation result (VR) of Valid, Invalid, or Unverified, and includes in the result details to explain the decision.”

Id. at 28–29; *see id.* at 29–30.

Further, Petitioner asserts that the server-PVA issues another report, i.e., a report “generated from the evaluating or testing of” the integrity reports to “determine vulnerabilities using information (e.g., CVE identifiers) in a vulnerabilities database (VULDB).” Pet. 29. According to Petitioner, an Open Vulnerability and Assessment Language (OVAL) tool can “describe the test cases to detect the presence of vulnerabilities.” *Id.* (quoting Ex. 1005, 7).

Regarding the combined disclosures in Munetoh and Rajan, Petitioner contends that “Rajan discloses consistent and compatible features corresponding to those” recited in limitation 1e. Pet. 30. But Petitioner does not identify any specific “consistent and compatible features” in Rajan. *See id.* at 30–31.

(ii) Patent Owner's Contentions

Patent Owner disputes that Munetoh teaches limitation 1e and that the combined disclosures in Munetoh and Rajan teach limitation 1e. *See* Prelim. Resp. 22. Specifically, Patent Owner asserts the references alone or in combination do not disclose limitation 1c's received "runtime execution context" or limitation 1d's received "security context" and, therefore, cannot disclose "generating, by the attestation server, a report indicating security risks associated with the application based on the received runtime execution context and the received security context, as an attestation result" according to limitation 1e. *Id.*

(iii) Analysis

Based on the current record, Petitioner does not establish sufficiently for purposes of institution that Munetoh teaches limitation 1e or that the combined disclosures in Munetoh and Rajan teach limitation 1e. *See* Pet. 27–31; Ex. 1003 ¶¶ 182–197. Regarding Munetoh alone, for the reasons discussed above for limitation 1c, we agree with Patent Owner that Munetoh does not disclose limitation 1c's received "runtime execution context" and, therefore, does not disclose generating a report "based on the received runtime execution context" according to limitation 1e. *See supra* § IV.D.3(c)(iii); Prelim. Resp. 22.

Regarding the combined disclosures in Munetoh and Rajan, Petitioner's contention that "Rajan discloses consistent and compatible features corresponding to those" recited in limitation 1e fails to show how Rajan cures Munetoh's deficiencies for limitation 1e. *See* Pet. 30–31.

(f) Limitation 1f

Claim 1 recites “sending, by the attestation server, the attestation result associated with the application.” Ex. 1001, 26:44–45 (limitation 1f).

Petitioner contends that Munetoh teaches limitation 1f because Munetoh discloses that the server-PVA (the claimed “attestation server”) sends the following:

- (1) a validation result to a client by HTTPS “where the validation result includes the detailed conditions at the computer platform that led to the result”;⁹ and
- (2) “the test results for vulnerabilities.”

Pet. 31.

At this stage of the proceeding, Patent Owner makes no arguments specific to limitation 1f. *See, e.g.*, Prelim. Resp. 15–22.

Based on the current record and for the reasons advanced by Petitioner and supported by Dr. Nielson’s testimony, Petitioner establishes sufficiently that Munetoh teaches limitation 1f. *See* Pet. 31; Ex. 1003 ¶¶ 198–204.

(g) Conclusion About Obviousness/Nonobviousness

Because Petitioner does not establish sufficiently for purposes of institution that the combined disclosures in Munetoh and Rajan teach each limitation in claim 1, Petitioner does not establish sufficiently for purposes of institution that the combined disclosures teach claim 1’s subject matter. *See* Pet. 22–24, 27–31; Ex. 1003 ¶¶ 158–165, 182–197; *supra* §§ IV.D.3(c)(iii), IV.D.3(e)(iii). Hence, Petitioner does not demonstrate a

⁹ The acronym “HTTPS” stands for “Hypertext Transfer Protocol Secure.”

reasonable likelihood of proving that claim 1 is unpatentable under § 103(a) as obvious over Munetoh and Rajan.

4. DEPENDENT CLAIMS 2–7 AND 9

Claims 2–7 and 9 depend directly or indirectly from claim 1. Ex. 1001, 26:46–27:7, 27:12–17. Hence, claims 2–7 and 9 incorporate all the limitations in claim 1. 35 U.S.C. § 112 ¶ 4 (2006).

Based on the current record and for the reasons discussed above for claim 1, Petitioner does not demonstrate a reasonable likelihood of proving that claims 2–7 and 9 are unpatentable under § 103(a) as obvious over Munetoh and Rajan. *See supra* §§ IV.D.3(c)(iii), IV.D.3(e)(iii), IV.D.3(g).

E. Alleged Obviousness over Rajan and Munetoh: Claims 1–7 and 9

Petitioner contends that claims 1–7 and 9 are unpatentable under § 103(a) as obvious over Rajan and Munetoh. *See* Pet. 3, 43–61. Patent Owner disputes Petitioner’s contentions. *See* Prelim. Resp. 14, 23–28.

Above, we provided overviews of Rajan and Munetoh. *See supra* §§ IV.D.1–IV.D.2. Below, we consider the obviousness issues. For the reasons explained below, Petitioner does not establish sufficiently for purposes of institution that the combined disclosures in Rajan and Munetoh teach the subject matter in claims 1–7 and 9.

1. INDEPENDENT CLAIM 1

(a) Preamble 1a

Claim 1 recites “[a] method of providing an attestation service for an application at runtime executing on a computing platform using an attestation server.” Ex. 1001, 26:27–29.

Petitioner contends that Rajan teaches claim 1’s preamble because Rajan discloses the following:

- (1) a provider that provides many services including a software stack or an application;
- (2) the provider includes a trusted platform module (TPM) that attests to “the integrity of the provider’s software stack as well its identity”;
- (3) the provider includes a requirements monitor that performs “detailed monitoring” and provides “traces of program operations during execution”;
- (4) a client that provides service requirements for an application available from the provider;
- (5) the provider “needs to be attested” before a client may “safely use the provider”; and
- (6) a broker that “receives a client’s service requirements and checks them against the measurements” from the provider’s TPM using the broker’s TPM.

Pet. 45–46 (quoting Ex. 1006, 3).

Petitioner identifies the broker as an “attestation server” and the provider as a “computing platform” according to claim 1’s preamble.

Pet. 45; *see id.* at 47.

At this stage of the proceeding, Patent Owner makes no arguments specific to claim 1’s preamble. *See, e.g.,* Prelim. Resp. 23–28.

Generally, a preamble does not limit a claim. *Allen Eng’g*, 299 F.3d at 1346. We need not decide whether claim 1’s preamble limits the claim because, based on the current record and for the reasons advanced by Petitioner and supported by Dr. Nielson’s testimony, Petitioner establishes sufficiently that Rajan teaches claim 1’s preamble. *See* Pet. 45–47; Ex. 1003 ¶¶ 123–124, 134–142, 273–276.

(b) Limitation 1b

Claim 1 recites “receiving, by the attestation server remote from the computing platform.” Ex. 1001, 26:30–31 (limitation 1b).

Petitioner contends that Rajan teaches limitation 1b because Rajan discloses that the broker receives from the provider integrity measurements “describing the security state of a monitor and analyzer on the service provider via a secure channel.” Pet. 47–48.

At this stage of the proceeding, Patent Owner makes no arguments specific to limitation 1b. *See, e.g.*, Prelim. Resp. 23–28.

Based on the current record and for the reasons advanced by Petitioner and supported by Dr. Nielson’s testimony, Petitioner establishes sufficiently that Rajan teaches limitation 1b. *See* Pet. 47–48; Ex. 1003 ¶¶ 145–146, 153–155, 277–278.

(c) Limitation 1c

Claim 1 recites receiving “a runtime execution context indicating attributes of the application at runtime, wherein the attributes comprise one or more executable file binaries of the application and loaded components of the application.” Ex. 1001, 26:32–35 (limitation 1c).

(i) Petitioner’s Contentions

Petitioner contends that Rajan teaches limitation 1c because Rajan discloses a requirements monitor “tracing software execution in the software stack” during software operation that corresponds to the claimed “runtime execution context indicating attributes of the application.” Pet. 48–49. Petitioner also contends that Rajan’s traces are (1) a “static checksum of the software stack” that corresponds to the claimed attributes comprising “one or more executable file binaries of the application” and (2) “logs of running

services” that correspond to the claimed attributes comprising “one or more . . . loaded components of the application.” *Id.* at 49 (quoting Ex. 1006, 7) (citing Ex. 1003 ¶¶ 166–167, 281–282).

(ii) Patent Owner’s Contentions

Patent Owner disputes that Rajan teaches limitation 1c. *See* Prelim. Resp. 23–24. Specifically, Patent Owner asserts that Rajan’s “static checksum of the software stack is inapposite” because (1) Rajan does not refer to the software stack as an application and (2) “Rajan does not measure applications.” *Id.* at 23 (citing Ex. 1006, 10). Patent Owner also asserts that “Rajan only describes that ‘[o]ur current experimental results have looked at static checksum as a method of ensuring the integrity of the monitor.’” *Id.* (alteration by Patent Owner) (quoting Ex. 1006, 12).

(iii) Analysis

Based on the current record, Petitioner does not establish sufficiently that Rajan teaches limitation 1c. *See* Pet. 48–49; Ex. 1003 ¶¶ 158–159, 166–167, 279–283. Specifically, Rajan discloses a trace-based requirements monitor where (1) the requirements are “specified as logical formulas over program traces” or “expressed as fields and methods” and (2) the monitor “determines whether the trace of service execution satisfied the desired nonfunctional requirements” in a service request from a client. Ex. 1006, 5–6; *see id.* at 8–9; Ex. 1003 ¶¶ 100, 103, 116, 119, 140–141, 176.

For verification purposes in Rajan, a “static checksum is created by computing the SHA1 hash of the software stack” on a provider “up to the requirements monitor” on the provider. Ex. 1006, 7; *see* Ex. 1003 ¶ 102. A broker receives the static checksum and a trace of a service implementation (software application) generated by the requirements monitor on the

provider. Ex. 1006, 4, 7; *see* Ex. 1003 ¶¶ 102, 166, 281. Then, the broker checks “the trace received from the provider” to determine “if it contains a sequence that violates the requirements” from a client. Ex. 1006, 7; *see* Ex. 1003 ¶¶ 104, 194.

Based on the current record, Petitioner fails to show that the static checksum (hash value) and the trace received by the broker include “one or more executable file binaries of the application and loaded components of the application” according to limitation 1c. *See* Pet. 48–49; Ex. 1003 ¶¶ 158–159, 166–167, 279–283. As for the static checksum (hash value) received by the broker, the ’441 patent distinguishes hash values from executable file binaries and loaded components. *See* Ex. 1001, 8:5–13, 8:16–18, 11:42–52, 14:26–31, 14:45–48, 16:9–13, 16:22–25; *see supra* § IV.D.3(c)(iii). Further, in a related litigation, Petitioner asserted that “binary hashes” are distinct from executable file binaries. Ex. 2003, 62–63.

As for the trace of a service implementation (software application) generated by the requirements monitor on the provider and received by the broker, Petitioner fails to show how the trace corresponds to “one or more executable file binaries of the application and loaded components of the application” according to limitation 1c. *See* Pet. 48–49; Ex. 1003 ¶¶ 158–159, 166–167, 279–283.

(d) Limitation 1d

Claim 1 recites receiving “a security context providing security information about the application, wherein the security information comprises an execution analysis of the one or more executable file binaries and the loaded components.” Ex. 1001, 26:36–39 (limitation 1d).

(i) Petitioner's Contentions

Petitioner contends that the combined disclosures in Rajan and Munetoh teach limitation 1d. *See* Pet. 49–51. Specifically, Petitioner asserts that Rajan discloses a requirements monitor on a provider that “traces the operations of running programs and traces variables, methods, and arguments of the programs” as they execute “to ensure information security during runtime.” *Id.* at 49. Petitioner also asserts that Rajan discloses “sending the trace data (i.e., the execution analysis)” to a broker that corresponds to the claimed “attestation server.” *Id.* at 49–50; *see id.* at 46–47.

Additionally, Petitioner contends that Munetoh discloses:

- (1) collecting “client-side integrity data into an integrity measurement log (IML) that includes a real time log of state changes in applications of the platform”;
- (2) recording “the integrity data for each component” running on a client platform in a security chip on the client platform; and
- (3) securely checking the integrity data “by remote attestation” with the server-PVA.

Pet. 50 (quoting Ex. 1005, 1) (citing Ex. 1003 ¶ 172). Petitioner contends that the integrity measurement log corresponds to “a security context providing security information about the application” according to limitation 1d. *Id.*

Further, Petitioner asserts that Munetoh teaches “the security information comprises an execution analysis of the one or more executable file binaries and the loaded components” according to limitation 1d because Munetoh discloses that:

- (1) the integrity measurement log includes “a list of components loaded on the platform and measured by the TPM” (trusted platform module) and “how they are composed”;
- (2) “a finite state machine (FSM) that analyzes” the integrity report (including the integrity measurement log) from the client-side PTS “based on a behavioral model that represents platform states and ‘the integrity of the component that is loaded and executed’”; and
- (3) the server-PVA “uses this information to ‘verif[y] the security state from the latest components.’”

Pet. 50–51 (alteration by Petitioner) (quoting Ex. 1005, 3, 6–7).

(ii) Patent Owner’s Contentions

Patent Owner disputes that the combined disclosures in Rajan and Munetoh teach limitation 1d. *See* Prelim. Resp. 24–25. Specifically, Patent Owner asserts that the references alone or in combination do not disclose “one or more executable file binaries of the application and loaded components of the application” according to limitation 1c and, therefore, cannot disclose “an execution analysis of the one or more executable file binaries and the loaded components” according to limitation 1d. *Id.* at 24. Patent Owner also asserts that Rajan’s disclosures about a requirements monitor and Munetoh’s disclosures about the integrity measurement log, the integrity report from the client-side PTS to the server-side PTS, and the finite state machine (FSM) do not “disclose analyzing one or more executable file binaries and the loaded components.” *Id.* at 24–25.

(iii) Analysis

Based on the current record, Petitioner establishes sufficiently that the combined disclosures in Rajan and Munetoh teach limitation 1d. *See* Pet. 49–51; Ex. 1003 ¶¶ 170–179, 284–288. Petitioner makes essentially the

same arguments why the combined disclosures in the references teach limitation 1d for the challenge with Rajan as the primary reference as for the challenge with Munetoh as the primary reference. *Compare* Pet. 24–27, *with id.* at 49–51. For the reasons discussed above for the challenge with Munetoh as the primary reference, Petitioner establishes sufficiently that the combined disclosures in the references teach limitation 1d. *See id.* at 24–27; Ex. 1003 ¶¶ 170–179; *supra* § IV.D.3(d)(iii).

(e) Limitation 1e

Claim 1 recites “generating, by the attestation server, a report indicating security risks associated with the application based on the received runtime execution context and the received security context, as an attestation result.” Ex. 1001, 26:40–43 (limitation 1e).

(i) Petitioner’s Contentions

Petitioner contends that the combined disclosures in Rajan and Munetoh teach limitation 1e. *See* Pet. 51–53. Specifically, Petitioner asserts that Rajan discloses “determining an attestation response” for “a service or application based on operational traces transmitted from the service to the broker.” *Id.* at 52. Petitioner asserts that the attestation response from the broker is “calculated during runtime based on both the static hash of the data and the trace data of the application.” *Id.* at 52–53.

Further, Petitioner asserts that an ordinarily skilled artisan would have been motivated to combine Rajan’s “determination of violations of the runtime execution of the application” with Munetoh’s “validation result report having vulnerability data” to “obtain the fine-grained, operation-level vulnerability information and obtain static and dynamic analysis of the system.” Pet. 53 (citing Ex. 1003 ¶¶ 194–195, 291). Petitioner does not

identify any disclosures in Munetoh relevant to limitation 1e other than Munetoh's "validation result report having vulnerability data." *See id.* at 51–53.

(ii) Patent Owner's Contentions

Patent Owner disputes that the combined disclosures in Rajan and Munetoh teach limitation 1e. *See* Prelim. Resp. 25–28. Specifically, Patent Owner asserts the references alone or in combination do not disclose limitation 1c's received "runtime execution context" or limitation 1d's received "security context" and, therefore, cannot disclose "generating, by the attestation server, a report indicating security risks associated with the application based on the received runtime execution context and the received security context, as an attestation result" according to limitation 1e. *Id.* at 26.

Further, Patent Owner asserts that the attestation response from the broker in Rajan "does not indicate security risks associated with the application based on the received runtime execution context and the received security context" and instead "includes an immutable integrity measurement of the local architecture." Prelim. Resp. 26–27 (emphasis omitted) (citing Ex. 1006, 6). Patent Owner also asserts that Rajan's immutable integrity measurement of the local architecture is "irrespective of any application running on that architecture, which is why Rajan states it does not directly measure applications 'which may change frequently.'" *Id.* at 27 (quoting Ex. 1006, 10).

Additionally, Patent Owner contends that Rajan concerns "verifying the integrity of a remotely-hosted requirements monitor, and specifically 'look[ing] at static checksum as a method of ensuring the integrity of the

monitor.” Prelim. Resp. 27–28 (alteration by Patent Owner) (quoting Ex. 1006, 12).

(iii) Analysis

Based on the current record, Petitioner does not establish sufficiently for purposes of institution that the combined disclosures in Rajan and Munetoh teach limitation 1e. *See* Pet. 51–53; Ex. 1003 ¶¶ 182–197, 289–293. For the reasons discussed above for limitation 1c, we agree with Patent Owner that Rajan does not disclose limitation 1c’s received “runtime execution context” and, therefore, does not disclose generating a report “based on the received runtime execution context” according to limitation 1e. *See supra* § IV.E.1(c)(iii); Prelim. Resp. 26. Further, Petitioner fails to show how Munetoh cures Rajan’s deficiencies for limitation 1e. *See* Pet. 51–53; Ex. 1003 ¶¶ 182–197, 289–293.

(f) Limitation 1f

Claim 1 recites “sending, by the attestation server, the attestation result associated with the application.” Ex. 1001, 26:44–45 (limitation 1f).

Petitioner contends that Rajan teaches limitation 1f because Rajan discloses that (1) the broker sends the attestation response to the client seeking the service implementation (software application) from the provider and (2) the broker notifies the provider “being attested of violations such as an API call.” Pet. 53–54.

At this stage of the proceeding, Patent Owner makes no arguments specific to limitation 1f. *See, e.g.*, Prelim. Resp. 23–28.

Based on the current record and for the reasons advanced by Petitioner and supported by Dr. Nielson’s testimony, Petitioner establishes

sufficiently that Rajan teaches limitation 1f. *See* Pet. 53–54; Ex. 1003 ¶¶ 198–199, 205, 294–297.

(g) Conclusion About Obviousness/Nonobviousness

Because Petitioner does not establish sufficiently for purposes of institution that the combined disclosures in Rajan and Munetoh teach each limitation in claim 1, Petitioner does not establish sufficiently for purposes of institution that the combined disclosures teach claim 1’s subject matter. *See* Pet. 48–49, 51–53; Ex. 1003 ¶¶ 158–159, 166–167, 182–197, 279–283, 289–293; *supra* §§ IV.E.1(c)(iii), IV.E.1(e)(iii). Hence, Petitioner does not demonstrate a reasonable likelihood of proving that claim 1 is unpatentable under § 103(a) as obvious over Rajan and Munetoh.

2. DEPENDENT CLAIMS 2–7 AND 9

Claims 2–7 and 9 depend directly or indirectly from claim 1. Ex. 1001, 26:46–27:7, 27:12–17. Hence, claims 2–7 and 9 incorporate all the limitations in claim 1. 35 U.S.C. § 112 ¶ 4.

Based on the current record and for the reasons discussed above for claim 1, Petitioner does not demonstrate a reasonable likelihood of proving that claims 2–7 and 9 are unpatentable under § 103(a) as obvious over Rajan and Munetoh. *See supra* §§ IV.E.1(c)(iii), IV.E.1(e)(iii), IV.E.1(g).

V. CONCLUSION

After considering the Petition, the Preliminary Response, and the evidence of record, and for the reasons discussed above, we determine that Petitioner has not demonstrated a reasonable likelihood that it would prevail in proving the unpatentability of at least one claim challenged in the Petition. Hence, we deny the Petition and do not institute an *inter partes* review.

VI. ORDER

Accordingly, it is

ORDERED that the Petition is denied and no trial is instituted.

IPR2023-00801
Patent 8,327,441 B2

For PETITIONER:

Robert M. Hansen
Scott B. Amankwatia
Matthew J. Anderson
THE MARBURY LAW GROUP, PLLC
rhansen@marburylaw.com
samankwatia@marburylaw.com
manderson@marburylaw.com
pat-docketing@marburylaw.com

For PATENT OWNER:

Peter Lambrianakos
Vincent J. Rubino, III
Alfred R. Fabricant
Joseph M. Mercadante
FABRICANT LLP
plambrianakos@fabricantllp.com
vrubino@fabricantllp.com
ffabricant@fabricantllp.com
jmercadante@fabricantllp.com
ptab@fabricantllp.com