



(43) International Publication Date
15 December 2016 (15.12.2016)

- (51) International Patent Classification:
H04L 29/12 (2006.01) H04L 29/08 (2006.01)
H04L 29/10 (2006.01)
- (21) International Application Number:
PCT/US2016/031590
- (22) International Filing Date:
10 May 2016 (10.05.2016)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
62/1173,882 10 June 2015 (10.06.2015) US
14/865,987 25 September 2015 (25.09.2015) US
- (71) Applicant: MCAFEE, INC. [US/US]; 2821 Mission College Boulevard, Santa Clara, CA 95054-1838 (US).
- (72) Inventors: SMITH, Ned, M.; 375 SW Delta Drive, Beaverton, Oregon 97006 (US). HELDT-SHELLER, Nathan; 4039 NE 75th Avenue, Portland, OR 97213 (US). SCHRECKER, Sven; 2532 Corbel Way, San Marcos, CA 92078 (US).
- (74) Agent: CRANDALL, Sean, C.; Patent Capital Group, c/o CPA Global, P.O. Box 52050, Minneapolis, MN 55402 (US).
- (81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))

Published:

- with international search report (Art. 21(3))

(54) Title: DEVICE NAMING IN AN INTERNET OF THINGS

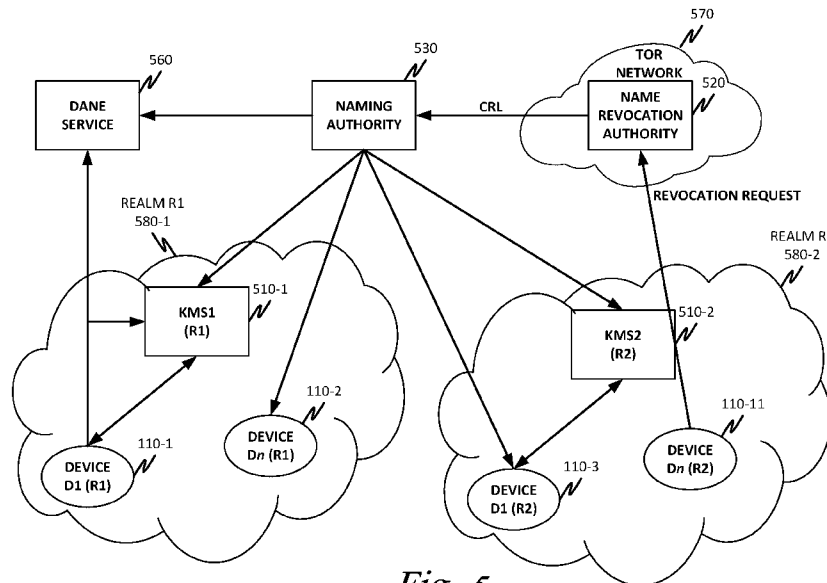


Fig. 5

(57) Abstract: In an example, there is disclosed a computing apparatus, having: a network interface; and one or more logic elements providing a name management engine, operable to: receive a self-assigned name registration request for a name N1 from an endpoint device via the network interface; compare N1 to a database of registered names; determine that the name has not been registered; and sign a certificate for N1. The engine is further operable to determine that the name has been registered, and send a notification that the name is not available. There is also disclosed a computer-readable medium having executable instructions for providing a name management engine, and a method of providing a name management engine.

WO 2016/200528 A1

DEVICE NAMING IN AN INTERNET OF THINGS

Cross Reference to Related Applications

[1] This Application claims priority to U.S. Provisional Application No. 62/173,882, titled "Internet of Things Device Name and Realm Name Selection and Certification", filed June 10, 2015 and U.S. Nonprovisional Utility Application No. 14/865,987, titled "Device Naming in an Internet of Things", filed September 25, 2015, both of which are incorporated herein by reference.

Field of the Specification

[2] This disclosure relates in general to the field of computer networking, and more particularly, though not exclusively to, a system and method for device naming in an internet of things.

Background

[3] The Internet of Things (IoT) is a loosely-defined network of physical objects (things) with embedded computing and communication capabilities, thus allowing the "things" to exchange data with one another. Within the IoT, real world phenomena can be sensed or observed electronically, and outputs from sensors or other data sources may be used as an input to a control system. In some cases, this allows a tighter coupling between the physical world and the virtual space. Each "thing" in the IoT may be uniquely identified with its physical computing platform, and some things are configured to operate within the existing internet infrastructure. Other devices operate over other network topologies, including *ad hoc* networks and direct data connections among others. In general terms, the IoT is considered to be highly-democratic (sometimes even anarchic) in that individual devices and networks may have broad autonomy in terms of what they do, how they do it, and how they communicate about it.

Brief Description of the Drawings

[4] The present disclosure is best understood from the following detailed description when read with the accompanying figures. It is emphasized that, in accordance with the standard practice in the industry, various features are not necessarily drawn to scale, and are used for illustration purposes only. Where a scale is shown, explicitly or implicitly, it provides only one illustrative example. In other embodiments, the dimensions of the various features may be arbitrarily increased or reduced for clarity of discussion.

[5] FIGURE 1 is a block diagram of a monitored network, which may be or comprise an IoT realm or domain, according to one or more examples of the present specification.

[6] FIGURE 2 is a block diagram of a data source according to one or more examples of the present specification.

[7] FIGURE 3 is a block diagram of a data aggregator according to one or more examples of the present specification.

[8] FIGURE 4 is a block diagram of an expert system according to one or more examples of the present specification.

[9] FIGURE 5 is a block diagram of naming architecture according to one or more examples of the present specification.

[10] FIGURE 6 is a flow chart of a name registration method according to one or more examples of the present specification.

[11] FIGURE 7 is a flow chart of a name revocation method according to one or more examples of the present specification.

[12] FIGURE 8 is a flow chart of a method of establishing a secured channel between two realms according to one or more examples of the present specification.

[13] FIGURE 9 is a block diagram of a key management server according to one or more examples of the present specification.

[14] FIGURE 10 is a block diagram of a naming authority according to one or more examples of the present specification.

[15] FIGURE 11 is a block diagram of a trusted execution environment according to one or more examples of the present specification.

Summary

[16] In an example, there is disclosed a computing apparatus, having: a network interface; and one or more logic elements providing a name management engine, operable to: receive a self-assigned name registration request from a name, N1, from an endpoint device via the network interface; compare N1 to a database of registered names; determine that the name has not been registered; and sign a certificate for N1. The engine is further operable to determine that the name has been registered, and send a notification that the name is not available. There is also disclosed a computer-readable medium having executable instructions for providing a name management engine, and a method of providing a name management engine

Embodiments of the Disclosure

[17] The following disclosure provides many different embodiments, or examples, for implementing different features of the present disclosure. Specific examples of components and arrangements are described below to simplify the present disclosure. These are, of course, merely examples and are not intended to be limiting. Further, the present disclosure may repeat reference numerals and/or letters in the various examples. This repetition is for the purposes of simplicity and clarity and does not in itself dictate a relationship between the various embodiments and/or configurations discussed. Different embodiments may have different advantages, and no particular advantage is necessarily required of any embodiment.

[18] The "Internet of Things" (IoT) is an explosive global aggregation of heterogeneous "smart" and "network-enabled" devices that often each provide a single, specialized function. The IoT includes smart appliances, smart sensors,

smart phones, and a plethora of other devices to which "smart" can be (and often is) prepended.

[19] Because the IoT is not a traditional network, it presents challenges that are sometimes new and unique. For example, in traditional networking, an oligarchy of naming authorities parcel out a relative handful of globally-unique internet protocol (IP) addresses, as in the IPv4 space, which has a theoretical maximum of approximately 4.3 billion unique addresses. This made global IPv4 addresses a relatively dear commodity to be managed centrally by an administrative body. But the cascade of IoT devices makes such a scheme both impractical and undesirable for many. Rather, IoT devices may operate within defined subnetworks using network address translation (NAT), or may self-declare a "Universally Unique Identifier" (UUID), which in one example is a 128-bit integer, and which may be usable as an IPv6 IP address. Such autonomous naming presents both new opportunities and new challenges that users and enterprises are still working to understand and appreciate. In one sense, the IoT may be viewed as a new, wild frontier, where rules are still evolving and where any device can be practically anything that it wants to be. While this provides exciting opportunities for innovators to experiment and try new things, it also provides a sometimes-lawless frontier where devices and their designers may not always be able to rely on traditional security solutions.

[20] By some estimates, 20 - 50 billion IoT devices will be deployed in the world by 2020. For these devices to be able to interoperate without conflict, each device may need a name or identifier that does not conflict with other names already in use. IoT devices may also be grouped or may cooperate with other devices in a context (e.g., a "realm") where the context or realm has its own name or identifier. In many cases, IoT networks are a conglomerate of disparate proprietary networks and transport layer technologies that do not agree on a single common network-naming scheme. So it is unlikely that all device names will neatly map to an IPv6 namespace (128-bit identifier). Thus, the existing internet naming infrastructure may fall short in handling some IoT device names.

[21] An alternative approach allows devices to self-assert a name such as a UUID where the UUID algorithm produces a pseudo-unique UUID. In this case,

pseudo-unique simply means that a self-asserted name (such as an IPv6 UUID) need not be parceled out as a finite, controlled resource the way 32-bit IPv4 addresses are currently handled. Rather, the device may choose its own name, and trust within a statistical probability that the name will be unique. For example, on a simple level, with a 128-bit address space, there is room for 3.4×10^{38} unique addresses. Thus, the instantaneous probability of a truly random algorithm selecting the same address twice is 1 in 3.4×10^{38} , which is beyond astronomical improbability. However, this result is skewed. Several factors complicate the probability, such as the large number of devices choosing names, and the fact that an IPv6 address includes structured fields rather than being simply a random 128-bit number. Some formal probability analyses have been published to evaluate the actual probability of IPv6 name collisions. And in short, the statistical likelihood that randomly-chosen, self-asserted UUID is genuinely unique is high enough that it is reasonable to permit devices to self-assert a name, and then provide mechanisms to deal with naming collisions if they occur.

[22] Nevertheless, self-asserted names may not be deemed authoritative except within the user's or organization's administrative domain. So even when a UUID is self-asserted, it may not be known whether the entity assigning the name is authorized, or if a rogue entity is engaged in a so-called "Sybil attack," in which a rogue actor seeks to subvert a peer-to-peer network with a forged identity.

[23] Organizations deploying IoT networks may find it helpful to group collections of devices in such a way that the collection itself (e.g., a "realm" or "domain") should be given a name. As with individual devices, there may be a collision in self-asserted realm names. Thus, it is recognized herein that there is a need for a method to name devices and collections of devices that deals with issues of collision, authenticity, and Sybil or similar attacks.

[24] An IoT network may also require assignment of cryptographic keys to devices so that device interactions can be protected. Key management services such as draft-hardjono-ace-fluffy-00 consider use of an IoT device that generates and assigns symmetric keys to peer devices following a Needham-Schroeder method. But at least some embodiments of those methods do not consider how a device name is reliably and securely associated to the device. They may also

presume that the key management server has knowledge of the expected device name and method for authenticating the device to the key management server. But the key management server may need to negotiate with other key management servers to determine device collection names that do not conflict and are not subject to Sybil attacks.

[25] Embodiments of the methods disclosed in this specification provide device and device group (a.k.a. realm) naming and assertion. The names may become authoritative names for IoT networks by combining a variety of Internet naming and certification technologies with IoT device and group naming conventions. This specification also provides methods for privacy protections in light of the intended objective of assigning unique identifiers to devices and collections.

[26] In an example, a naming authority (NA) supports a device or localized key management server that selects (asserts) a name to the NA, which establishes whether the name is already assigned to a different entity. The NA provides a certification (certificate) indicating simply that the name has not been previously assigned. Name assignment services may also support a business model where assigned names are rented or purchased as a way to pay for the cost of operating the NA. This specification also provides a method for efficiently determining if a name is previously assigned and resolving race conditions associated with name issuance.

[27] Once a name is issued, it is anticipated that other devices and key management servers may seek to verify whether or not the name is currently "in use," and if so, which authoritative entity is making the claim. A modified domain name system (DNS) infrastructure may be used according to RFC6698 where a certificate (subjectPublicKey) may be used by a DNS service to lookup a subjectAltName. This lookup allows the verifier to quickly establish that a device name or realm name has been reserved by a trusted NA and that another entity is not able to assert the same name. This may help to prevent Sybil attacks, for example.

[28] This specification also provides for a key management server (KMS), wherein a first KMS1 may engage with a second KMS2. In an example, each KMS

reserves a name using the NA and each dynamically asserts ownership of its respective name using a Diffie-Hellman key agreement protocol. If the certificate that is issued to KMS1 or KMS2 is used to sign the DH exchange, a MITM attacker is unable to disguise itself as either KMS1 or KMS2. So a secure channel can be established between KMS1 and KMS2, and there is no ambiguity over which name either KMS asserts as its own. This allows secure ad-hoc introduction of an IoT network (aka collection of devices) within a second IoT network with the knowledge that a third suspicious party cannot monitor the exchange or pose as KMS 1 or KMS2.

[29] Each KMS may obtain a realm name that is unique following a similar method as previously described, wherein the realm name is again unique (unused by another realm), and where the KMS asserts the realm name when issuing KMS “mini-tickets” to devices within a realm. A second KMS2 is unable to issue a mini-ticket with the same name because the KMS realm name may be independently verified using the NA certificate and a proof-of-possession block that accompanies mini-ticket issuance.

[30] In situations where privacy may be a concern (e.g., where there is danger that a device may be tracked based on its use of the identifier, knowing that the identifier is not used by any other device), a device may exhibit a privacy protection strategy by frequently changing its device identity.

[31] A separate (non-NA) service called a name revocation authority (NRA) may process identity revocations. This avoids the prospect that an NA may correlate requests for new identities with requests to revoke identities. The NRA accepts revocation requests from a large community of NA issued names and constructs a certificate revocation list (CRL), which in an example contains a minimum number of CRL entries (e.g. millions or more). The CRL is submitted to the NA *en masse*, preventing the NA from easily correlating subsequent NA certificate requests with a particular CRL entry. Unlike some existing CRL processing methods, the NRA accepts the revocation request on condition of a proof-of-possession. In other words, CRLs are only processed by the entity to which the name was assigned. If desired, the CRL may be post-dated so that the revocation occurs at a time chosen by the revoking entity, and in one example

CRL submissions may be made through a "Tor" anonymizing network to further confuse potential correlation activity.

[32] In an example, a DNS-based Authentication of Named Entities (DANE) service may be provided to efficiently find a previously allocated device or IoT realm name.

[33] In certain embodiments, a naming authority (NA) may singularly assign a name value to an asymmetric key.

[34] Also in an example, a trusted execution environment (TEE) may be used to protect an asymmetric key and to establish TEE properties that implement non-repudiation hardening.

[35] Advantageously, use of a name revocation authority (NRA) protects privacy of the entity performing the revocation request by batching CRL entries in a very large CRL, and where the CRL requestor may be obfuscated by a Tor network. The CRL requestor may prove possession of the private key as a condition of revocation to prevent unauthorized revocations of the assigned name. Thus, the owner of an assigned name may obtain a different name frequently to protect privacy.

[36] In some embodiments, a KMS may dynamically assign symmetric keys to IoT devices according to a Needham-Schroeder (or similar) protocol where an NA issued name is a device name and where a second NA issued name is a realm name.

[37] A system and method for device naming in an internet of things will now be described with more particular reference to the attached FIGURES. It should be noted that throughout the FIGURES, certain reference numerals may be repeated to indicate that a particular device or block is wholly or substantially consistent across the FIGURES. This is not, however, intended to imply any particular relationship between the various embodiments disclosed. In certain examples, a genus of elements may be referred to by a particular reference numeral ("widget 10"), while individual species or examples of the genus may be referred to by a hyphenated numeral ("first specific widget 10-1" and "second specific widget 10-2").

[38] FIGURE 1 is a network level diagram of a monitored network 100 according to one or more examples of the present specification. This figure illustrates a useful structure and application for an Internet of Things (IoT), though this embodiment should be understood to be nonlimiting. Monitored network 100 illustrates an example application in which the IoT provides inputs from a plurality of data sources, which are aggregated by a data aggregator. An expert system may then make decisions to drive a controlled system.

[39] In this example, monitored network 100 includes a plurality of data sources 120 connected to a network 170. Also connected to network 170 is a data aggregator 110, communicatively coupled to an expert system 130, controlling controlled systems 140. Controlled systems 140 provide feedback 142 to data aggregator 110.

[40] In one or more examples, data sources 120-1 through 120-N are disclosed. This is to illustrate that the number of data sources 120 may be large and indefinite, and may be in constant fluctuation as new data sources 120 are added to and removed from monitored network 100. Management of data sources 120 may be complicated both by the large number of data sources 120, and by the dynamic nature of monitored network 100. Thus in certain embodiments, it may be impractical for a human administrator to monitor and administer all of the various data sources 120. Furthermore, data sources 120 may not be statically located on network 170. For example, many cars carry data collection devices, and may provide data to network 170 as they hop from node to node on a mobile network. Thus, it may not be practical to predict in advance from which direction data will be coming, or what the nature of the data may be.

[41] By way of further complication, a plurality of data sources 120 may provide data features of similar or identical types, but in slightly different formats. In one example, each data source 120 is configured to provide a data stream accompanied by a metadata packet identifying the type and source of data. However, there may be no globally enforced or enforceable standard for such metadata packets. In an example, data sources 120 may at least standardize on a delivery format for the metadata, such as XML or a similar standards-compliant data format. In that case, the metadata may have a number of identifiable field

names, from which the feature type and source may be inferred. It should also be noted that in some cases data sources 120 may provide features of a compatible type, but in different formats. An example of this is a temperature feature provided by one data source 120-1 in Fahrenheit, and a second feature provided by a second data source 120-2 in Celsius. Similar issues may be encountered in any case where two or more data sources deliver similar features, with one data source providing the feature in metric units and the other data source providing the feature in Imperial or U.S. Customary units.

[42] This large collection of features of disparate types from different sources, and in different formats, is delivered via network 170 to a data aggregator 110. Data aggregator 110 collects the many features, and in an example attempts to classify the features according to a useful taxonomy. In one case, data aggregator 110 defines a taxonomy having an arbitrary number of classification levels, such as classes, sub-classes, genera, and species. For example, the class of environmental data may include the subclass of temperature, which may include further species of temperatures by location or source. In one example, all environmental features are classified as environmental features, temperature features are classified as temperature features, and temperature features of a common species may be aggregated by data aggregator 110 as being species that may be usefully combined. Data aggregator 110 may then provide one or more outputs to expert system 130.

[43] Expert system 130 may include one or more devices operable to collect features and to control one or more controlled systems 140. Expert system 130 may make decisions based on lookup tables, computer models, algorithms, or machine learning techniques. Features provided by data aggregator 110 may provide key inputs into the decisions that expert system 130 must make.

[44] Controlled systems 140 may include a number of real-world systems, such as air-conditioning, environmental systems, security systems, traffic systems, space-based systems, and any other system subject to automated control or data-driven operation. Controlled system 140 may include, in certain embodiments, facilities to measure the response of controlled systems 140 to inputs from expert system 130. Controlled systems 140 may then provide

feedback 142 to data aggregator 110. This may allow data aggregator 110 to measure the effect of combining or cross correlating certain features. In cases where data aggregator 110 determines that combining or cross correlating certain features has minimal impact on controlled systems 140, or in some cases even negative impact on controlled systems 140, data aggregator 110 may elect to unmerge certain features that are not found to be usefully combined.

[45] In an example, each data source 120 may include an appropriate operating system, such as Microsoft Windows, Linux, Android, Mac OSX, Apple iOS, Unix, or similar. Some of the foregoing may be more often used on one type of device than another. For example, desktop computers or engineering workstation may be more likely to use one of Microsoft Windows, Linux, Unix, or Mac OSX. Laptop computers, which are usually a portable off-the-shelf device with fewer customization options, may be more likely to run Microsoft Windows or Mac OSX. Mobile devices may be more likely to run Android or iOS. Embedded devices and dedicated appliances may run real-time operating systems such as real-time Linux, QNX, VxWorks, or FreeRTOS. For embedded devices without real-time demands, minimal Linux-based operating systems are currently very popular. However, all of the foregoing examples are intended to be nonlimiting.

[46] Network 170 may be any suitable network or combination of one or more networks operating on one or more suitable networking protocols, including for example, a local area network, an intranet, a virtual network, a wide area network, a wireless network, a cellular network, Bluetooth connections, or the Internet (optionally accessed via a proxy, virtual machine, or other similar security mechanism) by way of nonlimiting example. Importantly, network 170 need not be an IP-based network, but is broadly intended to encompass any suitable interconnect that allows devices to communicate with one another. This could include direct serial or parallel connections, Bluetooth, infrared communications, packet radio, telephony, or any other suitable communication link.

[47] Certain functions may also be provided on one or more servers, or one or more "microclouds" in one or more hypervisors. For example, a virtualization environment such as vCenter may provide the ability to define a plurality of "tenants," with each tenant being functionally separate from each other

tenant, and each tenant operating as a single-purpose microcloud. Each microcloud may serve a distinctive function, and may include a plurality of virtual machines (VMs) of many different flavors, including agentful and agentless VMs.

[48] In certain examples, monitored network 100 (or suitable portions thereof) may form an IoT "realm" or "domain," or may be part of a larger realm or domain.

[49] FIGURE 2 is a block diagram of a data source according to one or more examples of the present specification. Data source 120 may be any suitable computing device. In various embodiments, a "computing device" may be or comprise, by way of non-limiting example, a computer, workstation, server, mainframe, virtual machine (whether emulated or on a "bare-metal" hypervisor), embedded computer, embedded controller, embedded sensor, personal digital assistant, laptop computer, cellular telephone, IP telephone, smart phone, tablet computer, convertible tablet computer, computing appliance, network appliance, receiver, wearable computer, handheld calculator, or any other electronic, microelectronic, or microelectromechanical device for processing and communicating data. Any computing device may be designated as a host on the network. Each computing device may refer to itself as a "local host," while any computing device external to it may be designated as a "remote host."

[50] Data source 120 includes a processor 210 connected to a memory 220, having stored therein executable instructions for providing an operating system 222 and at least software portions of a data collection engine 224. Other components of data source 120 include a storage 250, network interface 260, and peripheral interface 240. This architecture is provided by way of example only, and is intended to be non-exclusive and non-limiting. Furthermore, the various parts disclosed are intended to be logical divisions only, and need not necessarily represent physically separate hardware and/or software components. Certain computing devices provide main memory 220 and storage 250, for example, in a single physical memory device, and in other cases, memory 220 and/or storage 250 are functionally distributed across many physical devices. In the case of virtual machines or hypervisors, all or part of a function may be provided in the form of software or firmware running over a virtualization layer to provide the

disclosed logical function. In other examples, a device such as a network interface 260 may provide only the minimum hardware interfaces necessary to perform its logical operation, and may rely on a software driver to provide additional necessary logic. Thus, each logical block disclosed herein is broadly intended to include one or more logic elements configured and operable for providing the disclosed logical operation of that block. As used throughout this specification, "logic elements" may include hardware, external hardware (digital, analog, or mixed-signal), software, reciprocating software, services, drivers, interfaces, components, modules, algorithms, sensors, components, firmware, microcode, programmable logic, or objects that can coordinate to achieve a logical operation.

[51] In an example, processor 210 is communicatively coupled to memory 220 via memory bus 270-3, which may be, for example, a direct memory access (DMA) bus by way of example, though other memory architectures are possible, including ones in which memory 220 communicates with processor 210 via system bus 270-1 or some other bus. Processor 210 may be communicatively coupled to other devices via a system bus 270-1. As used throughout this specification, a "bus" includes any wired or wireless interconnection line, network, connection, bundle, single bus, multiple buses, crossbar network, single-stage network, multistage network or other conduction medium operable to carry data, signals, or power between parts of a computing device, or between computing devices. It should be noted that these uses are disclosed by way of non-limiting example only, and that some embodiments may omit one or more of the foregoing buses, while others may employ additional or different buses.

[52] In various examples, a "processor" may include any combination of logic elements operable to execute instructions, whether loaded from memory, or implemented directly in hardware, including by way of non-limiting example a microprocessor, digital signal processor, field-programmable gate array, graphics processing unit, programmable logic array, application-specific integrated circuit, or virtual machine processor. In certain architectures, a multi-core processor may be provided, in which case processor 210 may be treated as only one core of a multi-core processor, or may be treated as the entire multi-core processor, as

appropriate. In some embodiments, one or more co-processors may also be provided for specialized or support functions.

[53] Processor 210 may be connected to memory 220 in a DMA configuration via DMA bus 270-3. To simplify this disclosure, memory 220 is disclosed as a single logical block, but in a physical embodiment may include one or more blocks of any suitable volatile or non-volatile memory technology or technologies, including for example DDR RAM, SRAM, DRAM, cache, L1 or L2 memory, on-chip memory, registers, flash, ROM, optical media, virtual memory regions, magnetic or tape memory, or similar. In certain embodiments, memory 220 may comprise a relatively low-latency volatile main memory, while storage 250 may comprise a relatively higher-latency non-volatile memory. However, memory 220 and storage 250 need not be physically separate devices, and in some examples may represent simply a logical separation of function. It should also be noted that although DMA is disclosed by way of non-limiting example, DMA is not the only protocol consistent with this specification, and that other memory architectures are available.

[54] Storage 250 may be any species of memory 220, or may be a separate device. Storage 250 may include one or more non-transitory computer-readable mediums, including by way of non-limiting example, a hard drive, solid-state drive, external storage, redundant array of independent disks (RAID), network-attached storage, optical storage, tape drive, backup system, cloud storage, or any combination of the foregoing. Storage 250 may be, or may include therein, a database or databases or data stored in other configurations, and may include a stored copy of operational software such as operating system 222 and software portions of data collection engine 224. Many other configurations are also possible, and are intended to be encompassed within the broad scope of this specification.

[55] Network interface 260 may be provided to communicatively couple data source 120 to a wired or wireless network. A "network," as used throughout this specification, may include any communicative platform operable to exchange data or information within or between computing devices, including by way of non-limiting example, an ad-hoc local network, an internet architecture providing

computing devices with the ability to electronically interact, a plain old telephone system (POTS), which computing devices could use to perform transactions in which they may be assisted by human operators or in which they may manually key data into a telephone or other suitable electronic equipment, any packet data network (PDN) offering a communications interface or exchange between any two nodes in a system, or any local area network (LAN), metropolitan area network (MAN), wide area network (WAN), wireless local area network (WLAN), virtual private network (VPN), intranet, direct parallel or serial connection, packet radio, or any other appropriate architecture or system that facilitates communications in a network or telephonic environment.

[56] Data collection engine 224, in one example, is operable to carry out computer-implemented methods as described in this specification. Data collection engine 224 may include one or more tangible non-transitory computer-readable mediums having stored thereon executable instructions operable to instruct a processor to provide a data collection engine 224. As used throughout this specification, an "engine" includes any combination of one or more logic elements, of similar or dissimilar species, operable for and configured to perform one or more methods provided by the engine. Thus, data collection engine 224 may comprise one or more logic elements configured to provide methods as disclosed in this specification. In some cases, data collection engine 224 may include a special integrated circuit designed to carry out a method or a part thereof, and may also include software instructions operable to instruct a processor to perform the method. In some cases, data collection engine 224 may run as a "daemon" process. A "daemon" may include any program or series of executable instructions, whether implemented in hardware, software, firmware, or any combination thereof, that runs as a background process, a terminate-and-stay-resident program, a service, system extension, control panel, bootup procedure, BIOS subroutine, or any similar program that operates without direct user interaction. In certain embodiments, daemon processes may run with elevated privileges in a "driver space," or in ring 0, 1, or 2 in a protection ring architecture. It should also be noted that data collection engine 224 may also include other

hardware and software, including configuration files, registry entries, and interactive or user-mode software by way of non-limiting example.

[57] In one example, data collection engine 224 includes executable instructions stored on a non-transitory medium operable to perform a method according to this specification. At an appropriate time, such as upon booting data source 120 or upon a command from operating system 222 or a user 120, processor 210 may retrieve a copy of the instructions from storage 250 and load it into memory 220. Processor 210 may then iteratively execute the instructions of data collection engine 224 to provide the desired method.

[58] Peripheral interface 240 may be configured to interface with any auxiliary device that connects to data source 120 but that is not necessarily a part of the core architecture of data source 120. A peripheral may be operable to provide extended functionality to data source 120, and may or may not be wholly dependent on data source 120. In some cases, a peripheral may be a computing device in its own right. Peripherals may include input and output devices such as displays, terminals, printers, keyboards, mice, modems, data ports (e.g., serial, parallel, USB, Firewire, or similar), network controllers, optical media, external storage, sensors, transducers, actuators, controllers, data acquisition buses, cameras, microphones, speakers, or external storage by way of non-limiting example.

[59] In one example, peripherals include display adapter 242, audio driver 244, and input/output (I/O) driver 246. Display adapter 242 may be configured to provide a human-readable visual output, such as a command-line interface (CLI) or graphical desktop such as Microsoft Windows, Apple OSX desktop, or a Unix/Linux X Window System-based desktop. Display adapter 242 may provide output in any suitable format, such as a coaxial output, composite video, component video, VGA, or digital outputs such as DVI or HDMI, by way of nonlimiting example. In some examples, display adapter 242 may include a hardware graphics card, which may have its own memory and its own graphics processing unit (GPU). Audio driver 244 may provide an interface for audible sounds, and may include in some examples a hardware sound card. Sound output may be provided in analog (such as a 3.5mm stereo jack), component ("RCA")

stereo, or in a digital audio format such as S/PDIF, AES3, AES47, HDMI, USB, Bluetooth or Wi-Fi audio, by way of non-limiting example.

[60] In an example, peripherals include one or more sensors 290, which may be configured and operable to collect data about real-world phenomena and to process the data into a digital form. In one operative example, data collection engine 224 collects data from sensor 290 via peripheral interface 240. The collected data may then be stored in storage 250 and/or sent over network interface 260.

[61] FIGURE 3 is a block diagram of a data aggregator 110 according to one or more examples of the present specification. Data aggregator 110 may be any suitable computing device, as described in connection with FIGURE 2. In general, the definitions and examples of FIGURE 2 may be considered as equally applicable to FIGURE 3, unless specifically stated otherwise.

[62] Data aggregator 110 includes a processor 310 connected to a memory 320, having stored therein executable instructions for providing an operating system 322 and at least software portions of a aggregation engine 324. Other components of data aggregator 110 include a storage 350, network interface 360, and peripheral interface 340. As described in FIGURE 2, each logical block may be provided by one or more similar or dissimilar logic elements.

[63] In an example, processor 310 is communicatively coupled to memory 320 via memory bus 370-3, which may be for example a direct memory access (DMA) bus. Processor 310 may be communicatively coupled to other devices via a system bus 370-1.

[64] Processor 310 may be connected to memory 320 in a DMA configuration via DMA bus 370-3, or via any other suitable memory configuration. As discussed in FIGURE 2, memory 320 may include one or more logic elements of any suitable type.

[65] Storage 350 may be any species of memory 320, or may be a separate device, as described in connection with storage 250 of FIGURE 2. Storage 350 may be, or may include therein, a database or databases or data stored in other configurations, and may include a stored copy of operational software such as operating system 322 and software portions of aggregation engine 324.

[66] Network interface 360 may be provided to communicatively couple server 140 to a wired or wireless network, and may include one or more logic elements as described in FIGURE 2.

[67] Aggregation engine 324 is an engine as described in FIGURE 2 and, in one example, includes one or more logic elements operable to carry out computer-implemented methods as described in this specification. Software portions of aggregation engine 324 may run as a daemon process.

[68] Aggregation engine 324 may include one or more non-transitory computer-readable mediums having stored thereon executable instructions operable to instruct a processor to provide a security engine. At an appropriate time, such as upon booting server 140 or upon a command from operating system 322 or a user or security administrator, processor 310 may retrieve a copy of aggregation engine 324 (or software portions thereof) from storage 350 and load it into memory 320. Processor 310 may then iteratively execute the instructions of aggregation engine 324 to provide the desired method. Operationally, aggregation engine 324 may be configured to collect and classify data provided by data sources 120.

[69] Peripheral interface 340 may be configured to interface with any auxiliary device that connects to data aggregator 110 but that is not necessarily a part of the core architecture of data aggregator 110. Peripherals may include, by way of non-limiting examples, any of the peripherals disclosed in FIGURE 2. In some cases, data aggregator 110 may include fewer peripherals than data source 120, reflecting that it may be more focused on providing processing services rather than interfacing directly with users.

[70] FIGURE 4 is a block diagram of an expert system 130 according to one or more examples of the present specification. Expert system 130 may be any suitable computing device, as described in connection with FIGURE 2. In general, the definitions and examples of FIGURE 2 may be considered as equally applicable to FIGURE 4, unless specifically stated otherwise.

[71] Expert system 130 includes a processor 410 connected to a memory 420, having stored therein executable instructions for providing an operating system 422 and at least software portions of a expert system engine 424. Other

components of expert system 130 include a storage 450, network interface 480, and peripheral interface 440. As described in FIGURE 2, each logical block may be provided by one or more similar or dissimilar logic elements.

[72] In an example, processor 410 is communicatively coupled to memory 420 via memory bus 470-3, which may be for example a direct memory access (DMA) bus. Processor 410 may be communicatively coupled to other devices via a system bus 470-1.

[73] Processor 410 may be connected to memory 420 in a DMA configuration via DMA bus 470-3, or via any other suitable memory configuration. As discussed in FIGURE 2, memory 420 may include one or more logic elements of any suitable type.

[74] Storage 450 may be any species of memory 420, or may be a separate device, as described in connection with storage 250 of FIGURE 2. Storage 450 may be, or may include therein, a database or databases or data stored in other configurations, and may include a stored copy of operational software such as operating system 422 and software portions of expert system engine 424.

[75] Network interface 460 may be provided to communicatively couple controlled systems 140 to a wired or wireless network, and may include one or more logic elements as described in FIGURE 2.

[76] Expert system engine 424 is an engine as described in FIGURE 2 and, in one example, includes one or more logic elements operable to carry out computer-implemented methods as described in this specification. Software portions of expert system engine 424 may run as a daemon process.

[77] Expert system engine 424 may include one or more non-transitory computer-readable mediums having stored thereon executable instructions operable to instruct a processor to provide a security engine. At an appropriate time, such as upon booting controlled systems 140 or upon a command from operating system 422 or a user or security administrator, processor 410 may retrieve a copy of expert system engine 424 (or software portions thereof) from storage 450 and load it into memory 420. Processor 410 may then iteratively execute the instructions of expert system engine 424 to provide the desired method. Operationally, expert system engine 424 may be configured to receive

aggregated data from data aggregator 110 and to make decisions about how to control controlled system 140.

[78] Peripheral interface 440 may be configured to interface with any auxiliary device that connects to expert system 130 but that is not necessarily a part of the core architecture of expert system 130. Peripherals may include, by way of non-limiting examples, any of the peripherals disclosed in FIGURE 2. In some cases, expert system 130 may include fewer peripherals than data source 120, reflecting that it may be more focused on providing processing services rather than interfacing directly with users.

[79] FIGURE 5 is a block diagram of a naming scheme for an IoT architecture. This embodiment includes three services for managing IoT device names and IoT realm names. These services should be understood to be logical functions, and need not be provided on separate physical or virtual machines. Rather, they represent logical functional divisions that may be implemented on any suitable combination of logic elements.

[80] In this example, DANE service 560 handles name requests using an issued certificate subject public key or using the issued name. DANE is a protocol that allows X.509 certificates, commonly used for Transport Layer Security (TLS), to be bound to DNS names using Domain Name System Security Extensions (DNSSEC). X.509 is disclosed as a nonlimiting example. In other cases, other formats may be for signing documents, such as JSON Web Signature (JWS), or any other suitable format. In an example, the lookup request may return information about the device or realm including network addresses or other device or host attributes. A lookup request may also be signed by DANE service 560 to authenticate the lookup results.

[81] Naming authority (NA) 530 issues certificates corresponding to device and realm name issue requests, where the objective is to establish that no other entity has been issued the same name. NA 530 may operate on a database of names where a database locking scheme such as a hierarchical lock manager and a hash table index may be used to quickly identify the name entry and prevent duplicate assignment due to race conditions. The name database may be replicated for high-availability and resistance to denial-of-service attacks. A

distributed lock manager may be used to ensure replicated shares do not result in duplicate name issuance.

[82] Name revocation authority (NRA) 520 issues CRLs that revoke issued names in a privacy-protecting manner to prevent unauthorized tracking of devices and realm activity using an issued name. In an example, the CRL contains a large number of name revocation requests so that a correlation between revocation requests and new name issuances is impractical.

[83] In an embodiment, a name revocation request may protect the requestor by routing the request through a Tor network 570 to obfuscate the requester's identity. The revocation request may further be obfuscated by post-dating when the request may be processed. The NRA prevents unauthorized name revocations (which may allow to denial of service) by verifying the requestor has high certification veracity consisting of a TEE that protects the private key.

[84] Devices 110 (which may be IoT data sources) and key management servers (KMS) 510 may request device and realm names from the NA according to their need.

[85] Realm R1 580-1 includes a plurality of devices D1 110-1 – Dn 110-2. R1 580-1 also includes KMS1 510-1, which in some cases may act as a local naming authority for R1 580-1.

[86] Realm R2 580-2 includes a plurality of devices D1 110-3 – Dn 110-n. R2 580-2 also includes KMS2 510-2, which in some cases may act as a local naming authority for R2 580-2.

[87] In an embodiment, names are self-assigned and asserted by devices 110 or by KMS 510, and are certified by NA 530. However, in other embodiments, names can be assigned by NA 530. Certified names are unique (not used by another entity), thus mitigating Sybil-type attacks.

[88] Peer KMS entities (e.g., KMS 510-1 and KMS 510-2) may establish an ad-hoc trust relationship by negotiating a Diffie-Hellman (DH) key exchange where the NA-issued certificate is used to sign the DH exchange. This establishes that the KMS entities are not spoofed by a rogue MITM and establishes realm namespaces that are distinct such that a device D1 110-1 (called D1R1 here) with a name not assigned by NA 530 can be disclosed to a second realm R2 having a

device with the same name D1 110-3 (called D1R2 here). D1R1 110-1 is distinguishable from the D1R2 110-3 because they are in separate realms, even though each may use the name "D1" internally. KMS entities may issue mini-tickets that provision symmetric keys for end-to-end device interactions dynamically. Use of the naming services allows cross-realm key management without confusion regarding misdirected symmetric keys.

[89] FIGURE 6 is a flow chart of name assignment method 600 according to one or more examples of the present specification. In an embodiment, name assignments follow a process similar to a PKCS10 certificate signing request, but in this case the certificate proves that the requested name has not been claimed by another device or realm. It also may establish that the requester has a sufficiently hardened environment, such as a TEE. This may help to prevent a "confused deputy" style, particularly where key non-repudiation properties are in force. An attestation protocol such as Intel Sigma, SigmaCE, or TPM attestation may be used to establish trust properties. The certificate may reflect the ability of the client device's environment to protect the key and perform other security and privacy related operations. NA 530 may further implement a method of payment for issued names where the name is rented or purchased for a period of time and where issuance is conditional on payment of agreed remuneration.

[90] In block 610, a device 110 or KMS 510 may self-assign a device or realm name, N1.

[91] In block 620, KMS 510 generates an appropriate key pair K1, which may be used to attest name N1.

[92] In block 630, KMS 510 may generate a certified name request, optionally with a proof of possession (PoP) private key.

[93] In block 640, KMS 510 establishes an attestation connection to NA 530.

[94] In decision block 650, NA 530 checks whether the endpoint (device or KMS) has a verified TEE. If there is no verified TEE, then in block 664, a "low veracity" certificate may be assigned. This means that the name is certified, but the device has a lower trust level because it has not provided hardened security features.

[95] In block 660, if there is a TEE, then NA 530 checks whether the PoP signature is valid. If it is not, then again, in block 664, NA 530 assigns a low veracity certificate to the endpoint.

[96] If the PoP signature is valid, then in block 670, NA 530 assigns the endpoint a high veracity certificate. This certificate indicates that the name is certified, and the device has a relatively high degree of trust.

[97] In block 680, NA 530 checks whether the proposed name has already been assigned to some other device. Optionally, this may include determining whether the name has been revoked on a CRL. In other embodiments, the name space is large enough that it is reasonable to enforce universal uniqueness.

[98] If the name has already been used, then NA 530 may send a signal informing the endpoint device that the name is not available. The device may still choose to use the name, but the name will not be certified and trusted.

[99] If the name has not already been used, then in block 690, NA 530 signs the certificate for N1. The endpoint device (realm 580 or device 110) may then use the name as a trusted, certified name.

[100] In block 699, the method is done.

[101] FIGURE 7 is a flow chart of name revocation method 700 according to one or more examples of the present specification. It should be noted that the method of FIGURE 7 could be performed on NA 530, on NRA 520, or on any other suitable device. A strict separation of NRA 520 from NA 530 is disclosed herein by way of nonlimiting example, with an illustration of enhanced security that is realized by separating the functions. NRA 520 may aggregate and batch CRLs so that it is difficult or impossible to trace a specific CRL to a specific actor at a specific time. This can enhance security and privacy.

[102] Absent a separate NRA or some other method to anonymize the CRL, it may be possible for unauthorized entities to use NA issued names to track a device or realm knowing that no other device or realm is using the same name. This has privacy implications. A strategy for minimizing negative privacy implications is to limit the use of the key. This is achieved by revoking the name using a CRL issued independently by NRA 520. In an embodiment, NRA 520 is separate and independent from NA 530.

[103] NRA 520 helps to ensure that revocation requests are not correlated with a name issuance event. This is achieved by NRA 520 forming a large CRL batch (for example, in the millions) so that correlation of name issuance events and name revocation events is impractical. The NRA may establish a business model where use of the service to revoke is subject to a fee. Having a separate business justification for NRA 520 and NA 530 helps prevent collusion between the separate entities that may result in loss of privacy.

[104] Revocation requests may be further protected by submitting requests through a Tor network, where the submission event may not be traced to a particular device or host and to a particular time of day. The request itself may include post-dated revocation to further disassociate the revocation event from a name issuance event.

[105] The endpoint device (e.g., device 110 or realm 580) may sign the request to allow the NRA to validate PoP of the private key, further establishing authorization to revoke. This helps prevent rogue entities from performing denial of service attack on the name.

[106] A condition of revocation may be to inform other devices of the revocation, including for example a peer KMS2 where KMS1 may have issued symmetric key tickets or established other context based on the name. This ensures that if re-issuance occurs, it does not represent a threat to the previous owner of the name.

[107] A condition of revocation may also be that a revoked name may not be reused or may not be reused for a period of time determined to be safe by the previous owner or the NRA.

[108] In block 710, an endpoint device generates a name revocation request for name N1.

[109] In block 720, the endpoint device signs the name revocation request using key K1.

[110] In block 730, the endpoint device submits the name revocation request to NRA 520, for example via Tor network 570.

[111] In decision block 740, NRA 520 checks whether the signature on the revocation request is valid. If it is not, then the name is not added to the CRL, and in block 799, the method is done.

[112] If the signature on the name revocation request is valid, then in decision block 760, NRA 520 checks whether it has reached its threshold of revocations before submitting the CRL. As discussed above, this may be a large number, for example in the millions, to help prevent easy correlation between revocations and issuance of new names. If the threshold has not been reached, then NRA 520 continues to wait for additional name revocation requests.

[113] If the threshold has been reached, then in block 770, NRA 520 submits the CRL to NA 530.

[114] In block 799, the method is done.

[115] FIGURE 8 is a flow chart of a method 800 for realm naming. This method may be particularly suited to preventing Sybil-type attacks. This may have value for IoT deployments where ad-hoc interactions are required across multiple realms. The KMS1 for a first realm may assert a realm name that is different from a second KMS2 managed realm, and where both are certain a third party has not been assigned a duplicate name. This prevents MITM attacks. The DH exchange may be signed by the name certificate to achieve this goal. The DANE DNS system may be consulted to verify the name assignment and to obtain contextual information that may facilitate the ad-hoc collaboration between KMS hosts. Symmetric keys issued within the context facilitate a common use case scenario where mobile and ad-hoc device-device interactions are expected but where end-to-end security is also expected.

[116] In block 810, a first realm 580-1 obtains a certified realm name R1.

[117] In block 820, realm 580-1 discovers a second realm 580-2.

[118] In block 830, KMS1 510-1 establishes a Diffie-Hellman session with KMS2 510-2.

[119] In block 850, the DH parameters are signed with the R1 and R2 certificates respectively.

[120] In decision block 860, KMS1 510-1 performs a DANE DNS lookup with DANE service 560 to determine whether R1 or R2 are valid. If they are not, then in block 899, the method is done.

[121] In block 870, if R1 or R2 is valid, KMS1 510-1 issues a KMS ticket for D1 in R1 to access D2 in R2.

[122] In block 899, the method is done.

[123] FIGURE 9 is a block diagram of a KMS 510 according to one or more examples of the present specification. KMS 510 may be any suitable computing device, as described in connection with FIGURE 2. In general, the definitions and examples of FIGURE 2 may be considered as equally applicable to FIGURE 9, unless specifically stated otherwise.

[124] KMS 510 includes a processor 910 connected to a memory 920, having stored therein executable instructions for providing an operating system 922 and at least software portions of a key management engine 924. Other components of KMS 510 include a storage 950, network interface 960. As described in FIGURE 2, each logical block may be provided by one or more similar or dissimilar logic elements.

[125] In an example, processor 910 is communicatively coupled to memory 920 via memory bus 970-3, which may be for example a direct memory access (DMA) bus. Processor 910 may be communicatively coupled to other devices via a system bus 970-1.

[126] Processor 910 may be connected to memory 920 in a DMA configuration via DMA bus 970-3, or via any other suitable memory configuration. As discussed in FIGURE 2, memory 920 may include one or more logic elements of any suitable type.

[127] Storage 950 may be any species of memory 920, or may be a separate device, as described in connection with storage 250 of FIGURE 2. Storage 950 may be, or may include therein, a database or databases or data stored in other configurations, and may include a stored copy of operational software such as operating system 922 and software portions of key management engine 924.

[128] Network interface 960 may be provided to communicatively couple server 140 to a wired or wireless network, and may include one or more logic elements as described in FIGURE 2.

[129] Key management engine 924 is an engine as described in FIGURE 2 and, in one example, includes one or more logic elements operable to carry out computer-implemented methods as described in this specification. Software portions of key management engine 924 may run as a daemon process.

[130] Key management engine 924 may include one or more non-transitory computer-readable mediums having stored thereon executable instructions operable to instruct a processor to provide a security engine. At an appropriate time, such as upon booting server 140 or upon a command from operating system 922 or a user or security administrator, processor 910 may retrieve a copy of key management engine 924 (or software portions thereof) from storage 950 and load it into memory 920. Processor 910 may then iteratively execute the instructions of key management engine 924 to provide the desired method. Operationally, key management engine 924 may be configured to collect and classify data provided by data sources 120.

[131] FIGURE 10 is a block diagram of a naming authority 530 according to one or more examples of the present specification. Data aggregator 110 may be any suitable computing device, as described in connection with FIGURE 2. In general, the definitions and examples of FIGURE 2 may be considered as equally applicable to FIGURE 10, unless specifically stated otherwise. It should also be noted that the structures disclosed herein may be equally applicable to naming authority 530, name revocation authority 520, and DANE service 560. In certain embodiments, one or more of these features may be provided by a common hardware platform, or by virtual machines running on common hardware.

[132] Data aggregator 110 includes a processor 1010 connected to a memory 1020, having stored therein executable instructions for providing an operating system 1022 and at least software portions of a aggregation engine 1024. Other components of data aggregator 110 include a storage 1050, and network interface 1060. As described in FIGURE 2, each logical block may be provided by one or more similar or dissimilar logic elements.

[133] In an example, processor 1010 is communicatively coupled to memory 1020 via memory bus 1070-3, which may be for example a direct memory access (DMA) bus. Processor 1010 may be communicatively coupled to other devices via a system bus 1070-1.

[134] Processor 1010 may be connected to memory 1020 in a DMA configuration via DMA bus 1070-3, or via any other suitable memory configuration. As discussed in FIGURE 2, memory 1020 may include one or more logic elements of any suitable type.

[135] Storage 1050 may be any species of memory 1020, or may be a separate device, as described in connection with storage 250 of FIGURE 2. Storage 1050 may be, or may include therein, a database or databases or data stored in other configurations, and may include a stored copy of operational software such as operating system 1022 and software portions of aggregation engine 1024.

[136] Network interface 1060 may be provided to communicatively couple server 140 to a wired or wireless network, and may include one or more logic elements as described in FIGURE 2.

[137] Name Management Engine 1024 is an engine as described in FIGURE 2 and, in one example, includes one or more logic elements operable to carry out computer-implemented methods as described in this specification. Software portions of Name Management Engine 1024 may run as a daemon process.

[138] Name Management Engine 1024 may include one or more non-transitory computer-readable mediums having stored thereon executable instructions operable to instruct a processor to provide a security engine. At an appropriate time, such as upon booting server 140 or upon a command from operating system 1022 or a user or security administrator, processor 1010 may retrieve a copy of Name Management Engine 1024 (or software portions thereof) from storage 1050 and load it into memory 1020. Processor 1010 may then iteratively execute the instructions of Name Management Engine 1024 to provide the desired method. Operationally, Name Management Engine 1024 may be configured to collect and classify data provided by data sources 120.

[139] FIGURE 11 is a block diagram of a trusted execution environment (TEE) 100 according to one or more examples of the present specification.

[140] In the example of FIGURE 11, memory 220 is addressable by n -bits, ranging in address from 0 to $2^n - 1$. Within memory 220 is an OS 222, enclave 1140, trusted execution framework (TEF) 1124, application stack 1120, and application code 1130.

[141] In this example, enclave 1140 is a specially-designated portion of memory 220 that cannot be entered into or exited from except via special instructions, such as Intel® SGX or similar. Enclave 1140 is provided as an example of a secure environment which, in conjunction with a secure processing engine 1110, forms a trusted execution environment (TEE) 1100. A TEE 1100 is a combination of hardware, software, and/or memory allocation that provides the ability to securely execute instructions without interference from outside processes, in a verifiable way. By way of example, TEE 1100 may include memory enclave 1140 or some other protected memory area, and a secure processing engine 1110, which includes hardware, software, and instructions for accessing and operating on enclave 1140. Non-limiting examples of solutions that either are or that can provide a TEE include Intel® SGX, ARM TrustZone, AMD Platform Security Processor, Kinibi, securiTEE, OP-TEE, TLK, T6, Open TEE, and SierraTEE, CSE, VT-x, MemCore, Canary Island, Docker, and Smack. Thus, it should be noted that in an example, secure processing engine 1110 may be a user-mode application that operates via trusted execution framework 1124 within enclave 1140. TEE 1100 may also conceptually include processor instructions that secure processing engine 1110 and trusted execution framework 1124 require to operate within enclave 1140.

[142] Secure processing engine 1110 and trusted execution framework 1124 may together form a trusted computing base (TCB), which is a set of programs or computational units that are trusted to be secure. Conceptually, it may be advantageous to keep TCB relatively small so that there are fewer attack vectors for malware objects or for negligent software. Thus, for example, operating system 222 may be excluded from TCB, in addition to the regular application stack 1120 and application code 1130.

[143] In certain systems, computing devices equipped with the Intel Software Guard Extension (SGX) or equivalent instructions may be capable of

providing an enclave 1140. It should be noted however, that many other examples of TEEs are available, and TEE 1100 is provided only as one example thereof. Other secure environments may include, by way of nonlimiting example, a virtual machine, sandbox, testbed, test machine, or other similar device or method for providing a TEE 1100.

[144] In an example, enclave 1140 provides a protected memory area that cannot be accessed or manipulated by ordinary computer instructions. Enclave 1140 is described with particular reference to an Intel® SGX enclave by way of example, but it is intended that enclave 1140 encompass any secure processing area with suitable properties, regardless of whether it is called an “enclave.”

[145] One feature of an enclave is that once an enclave region 1140 of memory 220 is defined, as illustrated, a program pointer cannot enter or exit enclave 1140 without the use of special enclave instructions or directives, such as those provided by Intel® SGX architecture. For example, SGX processors provide the ENCLU[EENTER], ENCLU[ERESUME], and ENCLU[EEXIT]. These are the only instructions that may legitimately enter into or exit from enclave 1140.

[146] Thus, once enclave 1140 is defined in memory 220, a program executing within enclave 1140 may be safely verified to not operate outside of its bounds. This security feature means that secure processing engine 1110 is verifiably local to enclave 1140. Thus, when an untrusted packet provides its content to be rendered with secure processing engine 1110 of enclave 1140, the result of the rendering is verified as secure.

[147] Enclave 1140 may also digitally sign its output, which provides a verifiable means of ensuring that content has not been tampered with or modified since being rendered by secure processing engine 1110. A digital signature provided by enclave 1140 is unique to enclave 1140 and is unique to the hardware of the device hosting enclave 1140.

[148] The foregoing outlines features of several embodiments so that those skilled in the art may better understand the aspects of the present disclosure. Those skilled in the art should appreciate that they may readily use the present disclosure as a basis for designing or modifying other processes and structures for carrying out the same purposes and/or achieving the same advantages of the

embodiments introduced herein. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the present disclosure, and that they may make various changes, substitutions, and alterations herein without departing from the spirit and scope of the present disclosure.

[149] The particular embodiments of the present disclosure may readily include a system on chip (SOC) central processing unit (CPU) package. An SOC represents an integrated circuit (IC) that integrates components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and radio frequency functions: all of which may be provided on a single chip substrate. Other embodiments may include a multi-chip-module (MCM), with a plurality of chips located within a single electronic package and configured to interact closely with each other through the electronic package. In various other embodiments, the digital signal processing functionalities may be implemented in one or more silicon cores in Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), and other semiconductor chips.

[150] Additionally, some of the components associated with described microprocessors may be removed, or otherwise consolidated. In a general sense, the arrangements depicted in the figures may be more logical in their representations, whereas a physical architecture may include various permutations, combinations, and/or hybrids of these elements. It is imperative to note that countless possible design configurations can be used to achieve the operational objectives outlined herein. Accordingly, the associated infrastructure has a myriad of substitute arrangements, design choices, device possibilities, hardware configurations, software implementations, equipment options, etc.

[151] Any suitably-configured processor component can execute any type of instructions associated with the data to achieve the operations detailed herein. Any processor disclosed herein could transform an element or an article (for example, data) from one state or thing to another state or thing. In another example, some activities outlined herein may be implemented with fixed logic or programmable logic (for example, software and/or computer instructions executed by a processor) and the elements identified herein could be some type of a

programmable processor, programmable digital logic (for example, a field programmable gate array (FPGA), an erasable programmable read only memory (EPROM), an electrically erasable programmable read only memory (EEPROM)), an ASIC that includes digital logic, software, code, electronic instructions, flash memory, optical disks, CD-ROMs, DVD ROMs, magnetic or optical cards, other types of machine-readable mediums suitable for storing electronic instructions, or any suitable combination thereof. In operation, processors may store information in any suitable type of non-transitory storage medium (for example, random access memory (RAM), read only memory (ROM), field programmable gate array (FPGA), erasable programmable read only memory (EPROM), electrically erasable programmable ROM (EEPROM), etc.), software, hardware, or in any other suitable component, device, element, or object where appropriate and based on particular needs. Further, the information being tracked, sent, received, or stored in a processor could be provided in any database, register, table, cache, queue, control list, or storage structure, based on particular needs and implementations, all of which could be referenced in any suitable timeframe. Any of the memory items discussed herein should be construed as being encompassed within the broad term 'memory.'

[152] Computer program logic implementing all or part of the functionality described herein is embodied in various forms, including, but in no way limited to, a source code form, a computer executable form, and various intermediate forms (for example, forms generated by an assembler, compiler, linker, or locator). In an example, source code includes a series of computer program instructions implemented in various programming languages, such as an object code, an assembly language, or a high-level language such as OpenCL, Fortran, C, C++, JAVA, or HTML for use with various operating systems or operating environments. The source code may define and use various data structures and communication messages. The source code may be in a computer executable form (e.g., via an interpreter), or the source code may be converted (e.g., via a translator, assembler, or compiler) into a computer executable form.

[153] In one example embodiment, any number of electrical circuits of the FIGURES may be implemented on a board of an associated electronic device. The

board can be a general circuit board that can hold various components of the internal electronic system of the electronic device and, further, provide connectors for other peripherals. More specifically, the board can provide the electrical connections by which the other components of the system can communicate electrically. Any suitable processors (inclusive of digital signal processors, microprocessors, supporting chipsets, etc.), memory elements, etc. can be suitably coupled to the board based on particular configuration needs, processing demands, computer designs, etc. Other components such as external storage, additional sensors, controllers for audio/video display, and peripheral devices may be attached to the board as plug-in cards, via cables, or integrated into the board itself. In another example embodiment, the electrical circuits of the FIGURES may be implemented as stand-alone modules (e.g., a device with associated components and circuitry configured to perform a specific application or function) or implemented as plug-in modules into application specific hardware of electronic devices.

[154] Note that with the numerous examples provided herein, interaction may be described in terms of two, three, four, or more electrical components. However, this has been done for purposes of clarity and example only. It should be appreciated that the system can be consolidated in any suitable manner. Along similar design alternatives, any of the illustrated components, modules, and elements of the FIGURES may be combined in various possible configurations, all of which are clearly within the broad scope of this specification. In certain cases, it may be easier to describe one or more of the functionalities of a given set of flows by only referencing a limited number of electrical elements. It should be appreciated that the electrical circuits of the FIGURES and its teachings are readily scalable and can accommodate a large number of components, as well as more complicated/sophisticated arrangements and configurations. Accordingly, the examples provided should not limit the scope or inhibit the broad teachings of the electrical circuits as potentially applied to a myriad of other architectures.

[155] Numerous other changes, substitutions, variations, alterations, and modifications may be ascertained to one skilled in the art and it is intended that the present disclosure encompass all such changes, substitutions, variations,

alterations, and modifications as falling within the scope of the appended claims. In order to assist the United States Patent and Trademark Office (USPTO) and, additionally, any readers of any patent issued on this application in interpreting the claims appended hereto, Applicant wishes to note that the Applicant: (a) does not intend any of the appended claims to invoke paragraph six (6) of 35 U.S.C. section 112 (pre-AIA) or paragraph (f) of the same section (post-AIA), as it exists on the date of the filing hereof unless the words "means for" or "steps for" are specifically used in the particular claims; and (b) does not intend, by any statement in the specification, to limit this disclosure in any way that is not otherwise reflected in the appended claims.

Example Implementations

[156] There is disclosed in one example, a computing apparatus, comprising: a network interface; and one or more logic elements comprising a name management engine, operable to: receive a self-assigned name registration request from a name N1 on an endpoint device via the network interface; compare N1 to a database of registered names; determine that the name has not been registered; and sign a certificate for N1.

[157] There is further disclosed an example, wherein the name management engine is further operable to determine that the name has been registered, and send a notification that the name is not available.

[158] There is further disclosed an example, wherein the name management engine is further operable to determine that the endpoint device has a trusted execution environment (TEE) meeting a minimum security requirement, and wherein the certificate is a high veracity certificate.

[159] There is further disclosed an example, wherein the name management engine is further operable to determine that the endpoint device does not have a trusted execution environment (TEE) meeting a minimum security requirement, and wherein the certificate is a low veracity certificate.

[160] There is further disclosed an example, wherein the name management engine is further operable to determine that the name registration

request is signed by a valid signature, and wherein the certificate is a high veracity certificate.

[161] There is further disclosed an example, wherein the valid signature is a proof of possession valid signature.

[162] There is further disclosed an example, wherein the name management engine is further operable to determine that the name registration request is not signed by a valid signature, and wherein the certificate is a low veracity certificate.

[163] There is further disclosed an example, wherein the name management engine is operable to receive a certificate revocation list (CRL).

[164] There is further disclosed an example, wherein receiving the CRL comprises receiving a name revocation request via the network interface.

[165] There is further disclosed an example, wherein the name revocation request is post-dated.

[166] There is further disclosed an example, wherein receiving the CRL comprises receiving a batch CRL via the network interface.

[167] There is further disclosed an example, wherein the name management engine is further operable to mark a plurality of names in the registered name database as revoked, wherein the plurality of names appear on the CRL.

[168] There is further disclosed an example, wherein the CRL is anonymized.

[169] There is further disclosed an example of one or more tangible, non-transitory computer-readable storage mediums having stored thereon executable instructions for instructing one or more processors for providing a name management engine operable for performing any or all of the operations of the preceding examples.

[170] There is further disclosed an example of a method of providing a name management engine comprising performing any or all of the operations of the preceding examples.

[171] There is further disclosed an example of an apparatus comprising means for performing the method.

[172] There is further disclosed an example wherein the means comprise a processor and a memory.

[173] There is further disclosed an example wherein the means comprise one or more tangible, non-transitory computer-readable storage mediums.

[174] There is further disclosed an example wherein the apparatus is a computing device.

[175] There is further disclosed, in an example, a computing apparatus, comprising: a network interface; and a key management engine operable to: determine that a first device D1 in a first realm R1 with the computing apparatus needs to establish a connection with a second device D2 in a second realm R2; establish a secure channel with a secure management service of the second realm R2; and issue a key management ticket for D1 to securely communicate with D2.

[176] There is further disclosed an example, wherein the key management engine is further operable to perform a domain name system (DNS) lookup of a name for R1 or R2 on a DNS-based authentication of named entities (DANE) service.

[177] There is further disclosed an example of one or more tangible, non-transitory computer-readable storage mediums having stored thereon executable instructions for instructing one or more processors for providing a key management engine operable for performing any or all of the operations of the preceding examples.

[178] There is further disclosed an example of a method of providing a key management engine comprising performing any or all of the operations of the preceding examples.

[179] There is further disclosed an example of an apparatus comprising means for performing the method.

[180] There is further disclosed an example wherein the means comprise a processor and a memory.

[181] There is further disclosed an example wherein the means comprise one or more tangible, non-transitory computer-readable storage mediums.

[182] There is further disclosed an example wherein the apparatus is a computing device.

Claims

1. A computing apparatus, comprising:

a network interface; and

one or more logic elements comprising a name management engine, operable to:

receive a self-assigned name registration request from a name N1 an endpoint device via the network interface;

compare N1 to a database of registered names;

determine that the name has not been registered; and

sign a certificate for N1.
2. The computing apparatus of claim 1, wherein the name management engine is further operable to determine that the name has been registered, and send a notification that the name is not available.
3. The computing apparatus of claim 1, wherein the name management engine is further operable to determine that the endpoint device has a trusted execution environment (TEE) meeting a minimum security requirement, and wherein the certificate is a high veracity certificate.
4. The computing apparatus of claim 1, wherein the name management engine is further operable to determine that the endpoint device does not have a trusted execution environment (TEE) meeting a minimum security requirement, and wherein the certificate is a low veracity certificate.
5. The computing apparatus of claim 1, wherein the name management engine is further operable to determine that the name registration request is signed by a valid signature, and wherein the certificate is a high veracity certificate.

6. The computing apparatus of claim 5, wherein the valid signature is a proof of possession valid signature.
7. The computing apparatus of claim 1, wherein the name management engine is further operable to determine that the name registration request is not signed by a valid signature, and wherein the certificate is a low veracity certificate.
8. The computing apparatus of any of claims 1 – 7, wherein the name management engine is operable to receive a certificate revocation list (CRL).
9. The computing apparatus of claim 8, wherein receiving the CRL comprises receiving a name revocation request via the network interface.
10. The computing apparatus of claim 9, wherein the name revocation request is post-dated.
11. The computing apparatus of claim 8, wherein receiving the CRL comprises receiving a batch CRL via the network interface.
12. The computing apparatus of claim 8, wherein the name management engine is further operable to mark a plurality of names in the registered name database as revoked, wherein the plurality of names appear on the CRL.
13. The computing apparatus of claim 8, wherein the CRL is anonymized.
14. One or more tangible, non-transitory computer readable storage mediums having stored thereon executable instructions for providing a name management engine, wherein the name management engine is configured to:
 - receive a self-assigned name registration request from a name N1 an endpoint device via a network interface;
 - compare N1 to a database of registered names;

determine that the name has not been registered; and

sign a certificate for N1.

15. The one or more tangible, computer-readable storage mediums of claim 14, wherein the name management engine is further operable to determine that the name has been registered, and send a notification that the name is not available.

16. The one or more tangible, computer-readable storage mediums of claim 14, wherein the name management engine is further operable to determine that the endpoint device has a trusted execution environment (TEE) meeting a minimum security requirement, and wherein the certificate is a high veracity certificate.

17. The one or more tangible, computer-readable storage mediums of claim 14, wherein the name management engine is further operable to determine that the endpoint device does not have a trusted execution environment (TEE) meeting a minimum security requirement, and wherein the certificate is a low veracity certificate.

18. The one or more tangible, computer-readable storage mediums of claim 14, wherein the name management engine is further operable to determine that the name registration request is signed by a valid proof of possession signature, and wherein the certificate is a high veracity certificate.

19. The one or more tangible, computer-readable storage mediums of claim 14, wherein the name management engine is further operable to determine that the name registration request is not signed by a valid signature, and wherein the certificate is a low veracity certificate.

20. The one or more tangible, computer-readable storage mediums of any of claims 14 – 19, wherein the name management engine is operable to receive an anonymized certificate revocation list (CRL).

21. The one or more tangible, computer-readable storage mediums of claim 20, wherein receiving the CRL comprises receiving a name revocation request via the network interface.

22. The one or more tangible, computer-readable storage mediums of claim 21, wherein the name revocation request is post-dated.

23. The one or more tangible, computer-readable storage mediums of claim 20, wherein receiving the CRL comprises receiving a batch CRL via the network interface.

24. A computing apparatus, comprising:

a network interface; and

a key management engine operable to:

determine that a first device D1 in a first realm R1 with the computing apparatus needs to establish a connection with a second device D2 in a second realm R2;

establish a secure channel with a secure management service of the second realm R2; and

issue a key management ticket for D1 to securely communicate with D2.

25. The computing apparatus of claim 24, wherein the key management engine is further operable to perform a domain name system (DNS) lookup of a name for R1 or R2 on a DNS-based authentication of named entities (DANE) service.

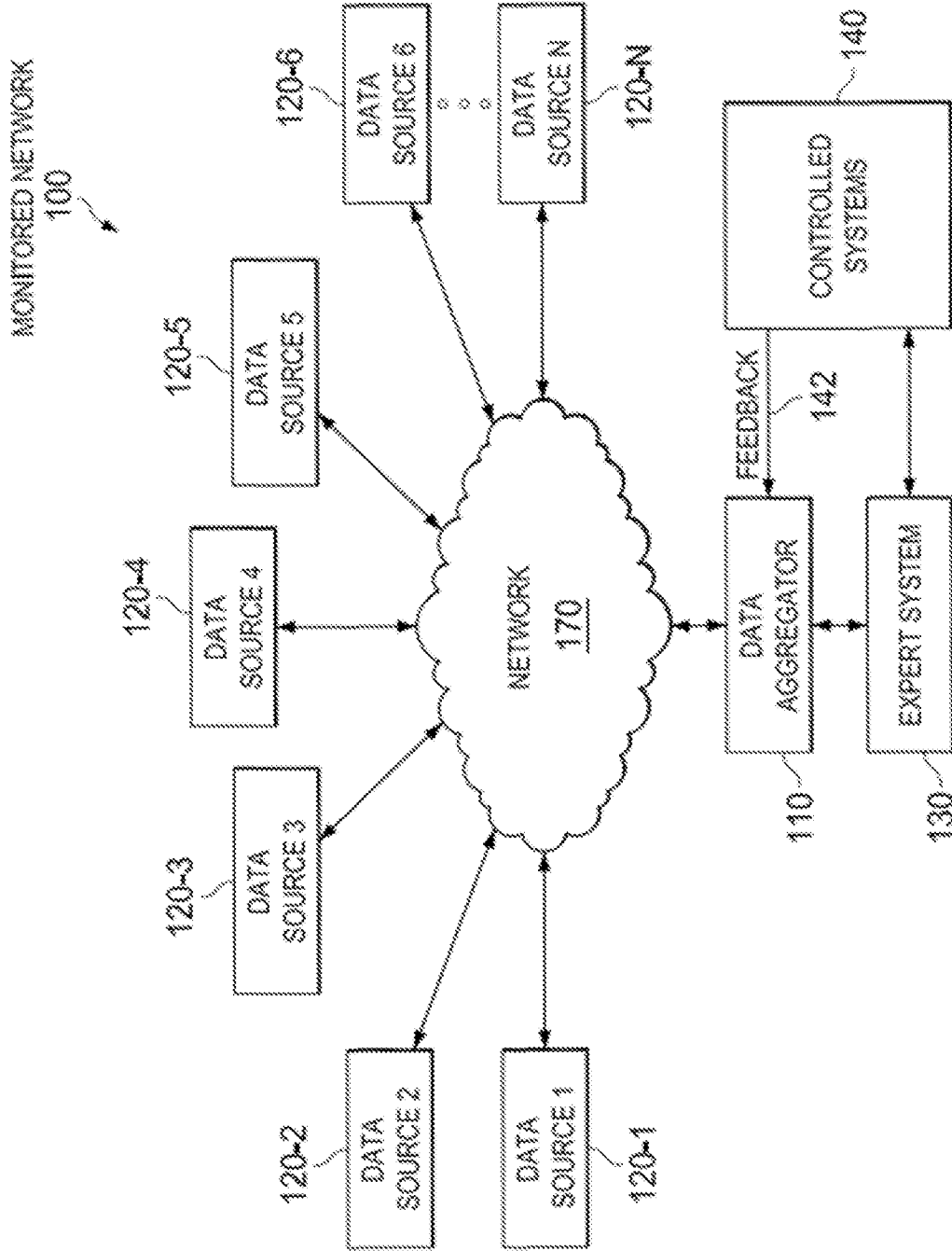


Fig. 1

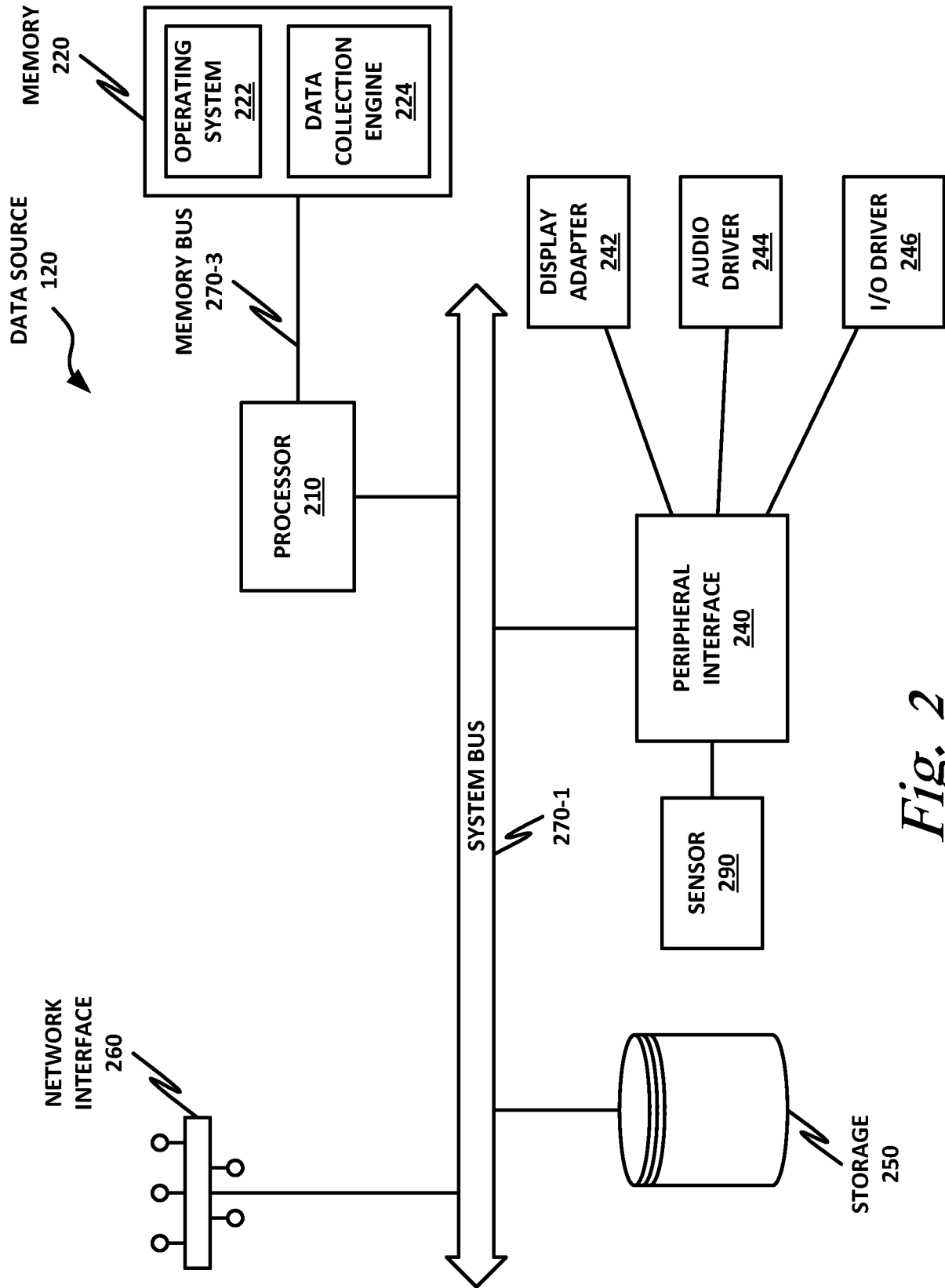


Fig. 2

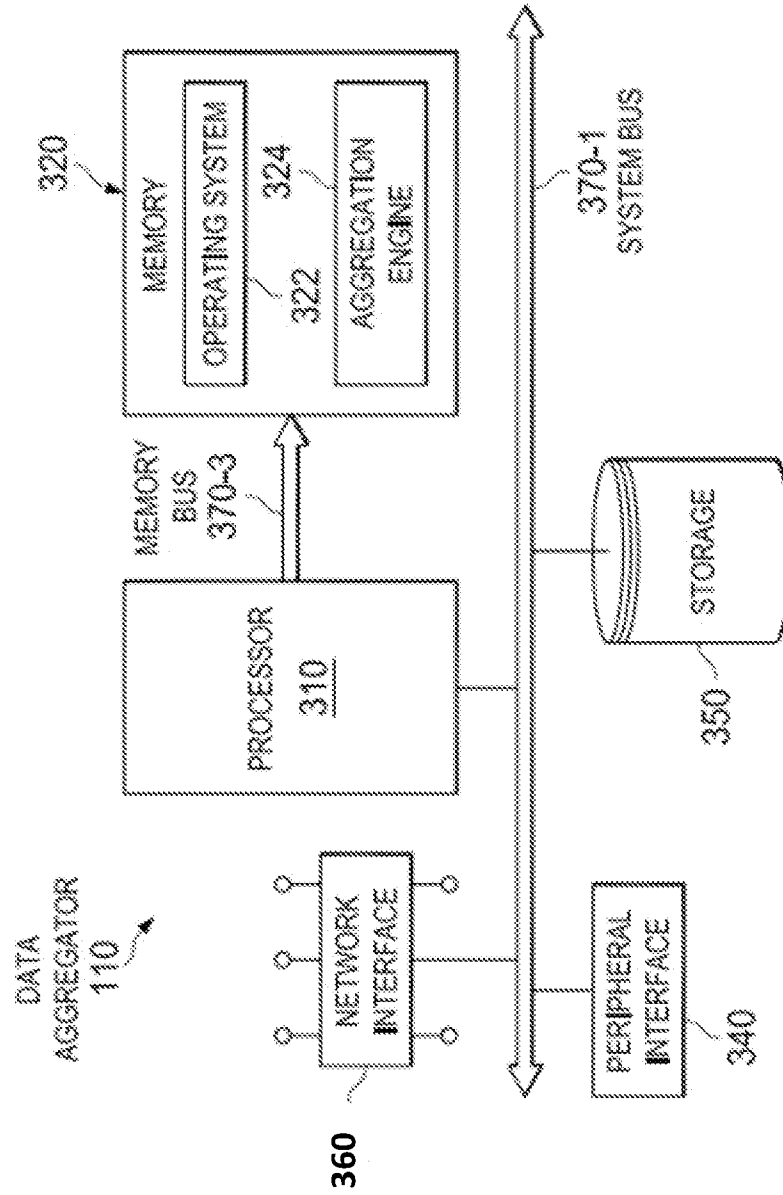


Fig. 3

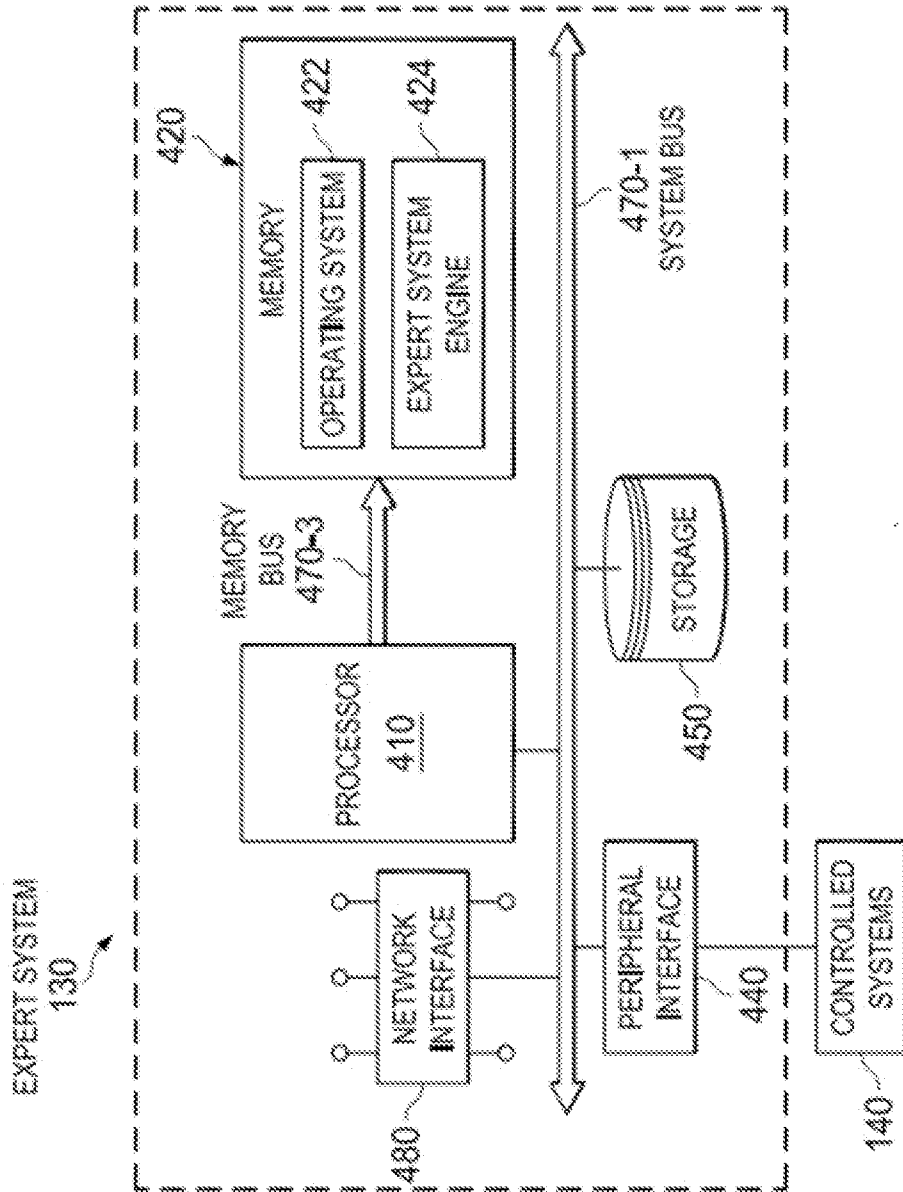


Fig. 4

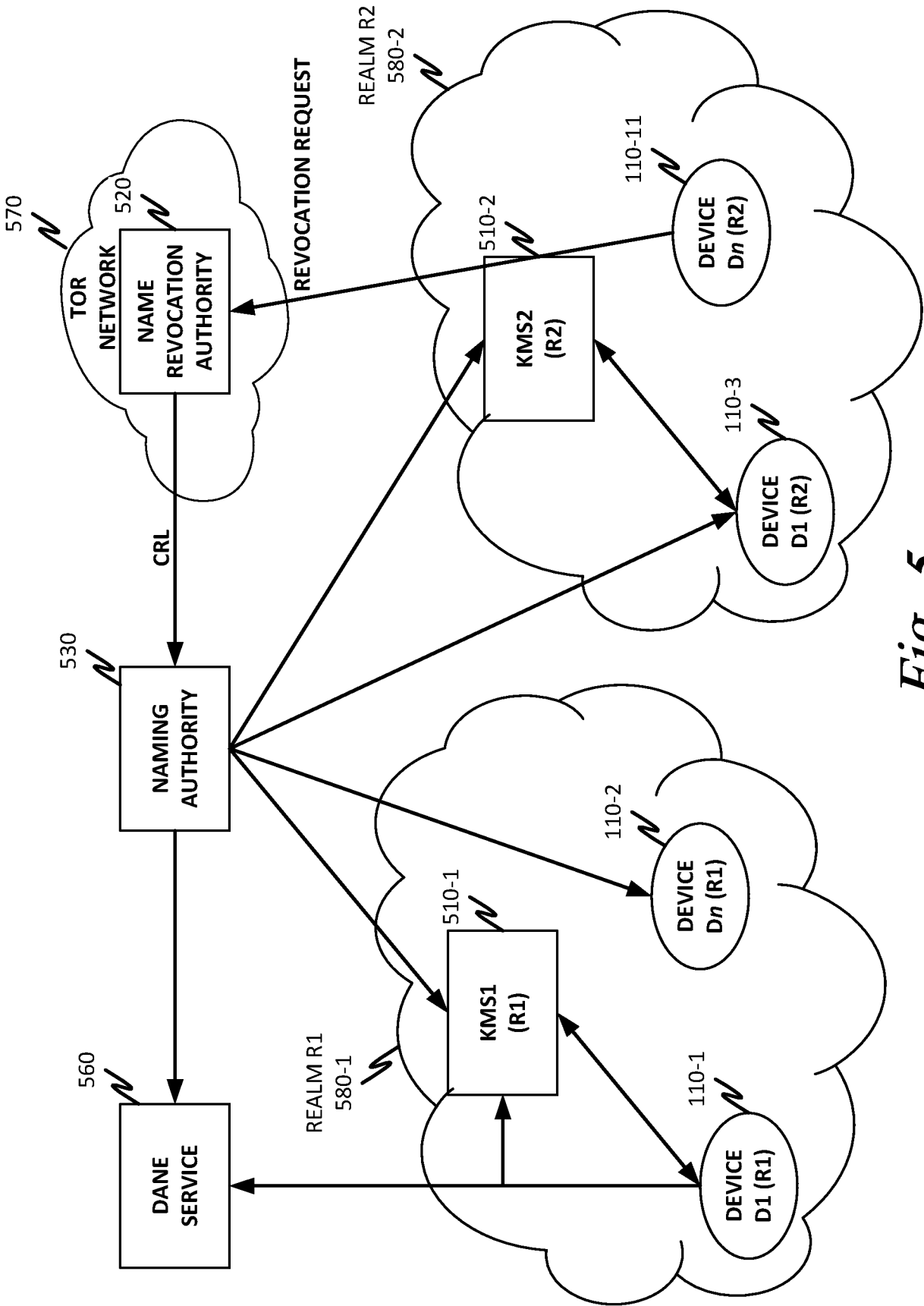


Fig. 5

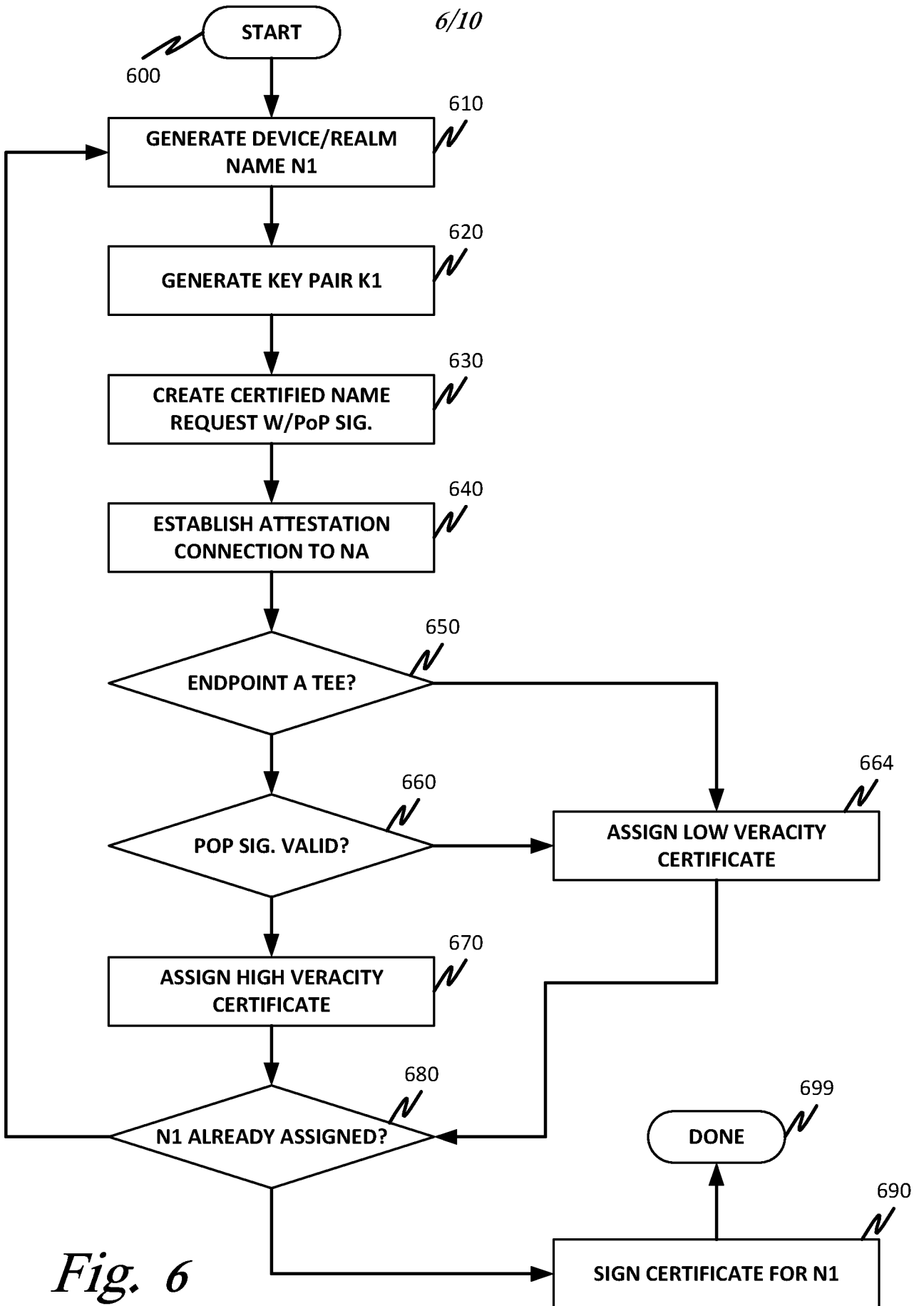


Fig. 6

7/10

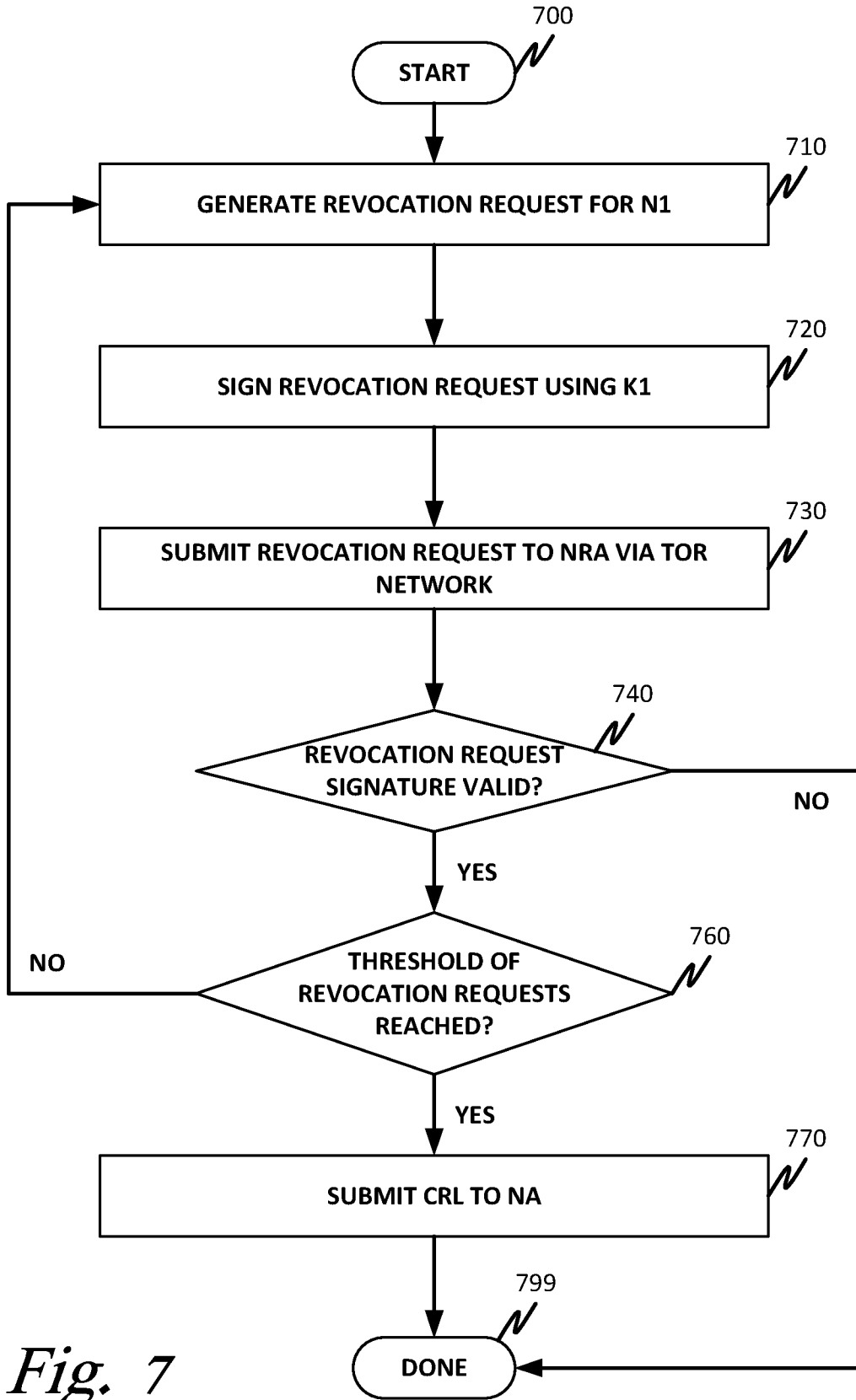


Fig. 7

8/10

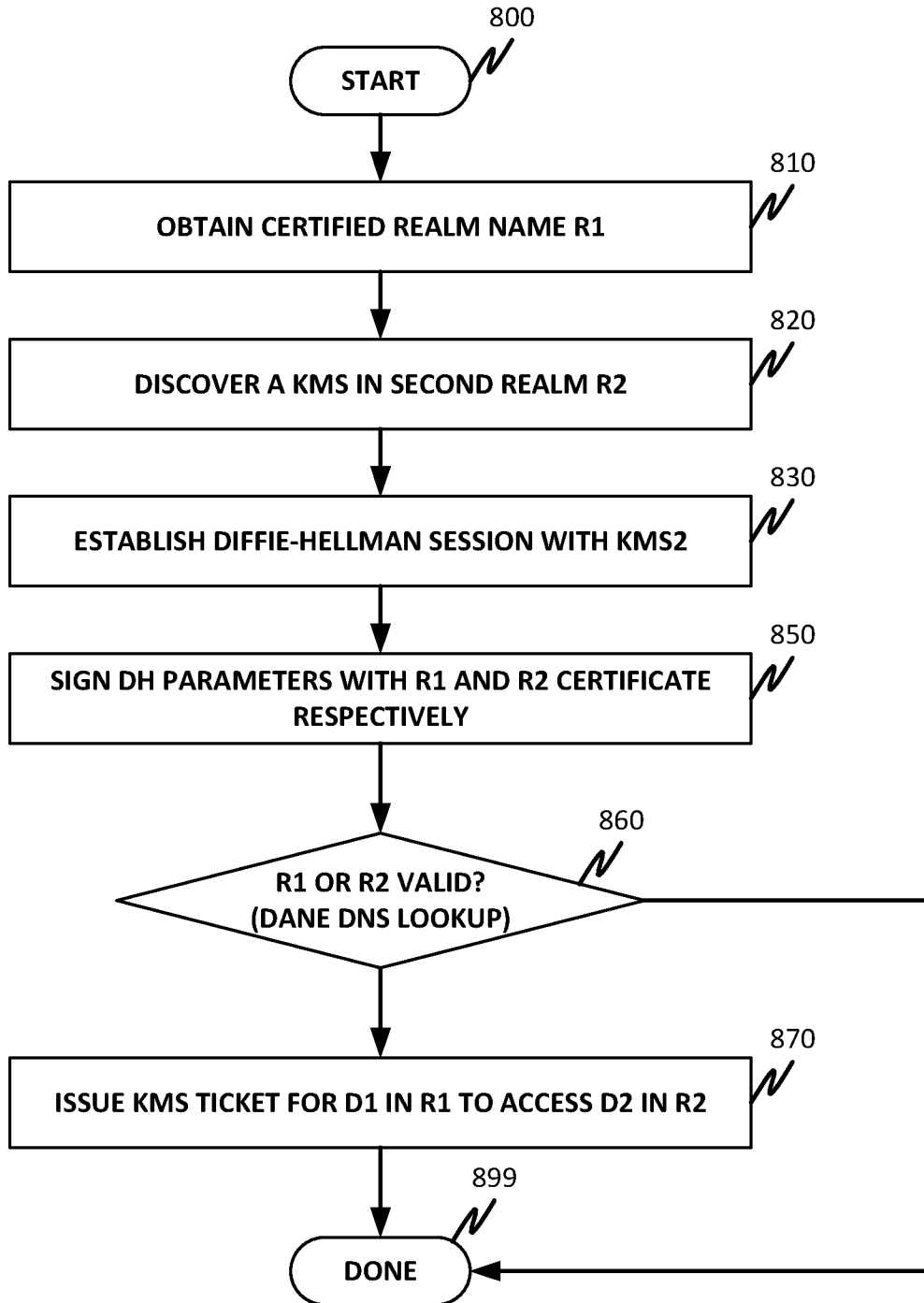


Fig. 8

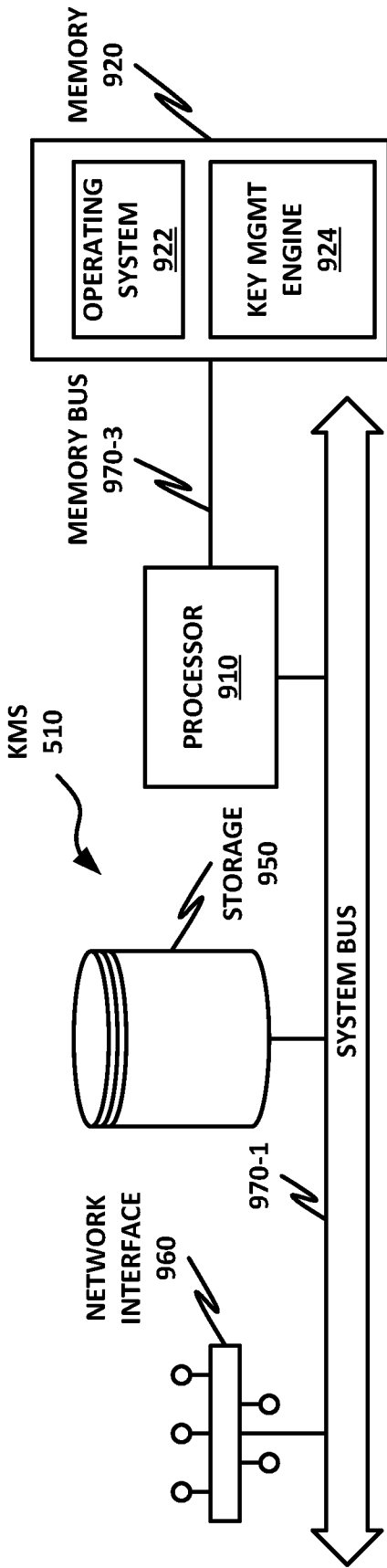


Fig. 9

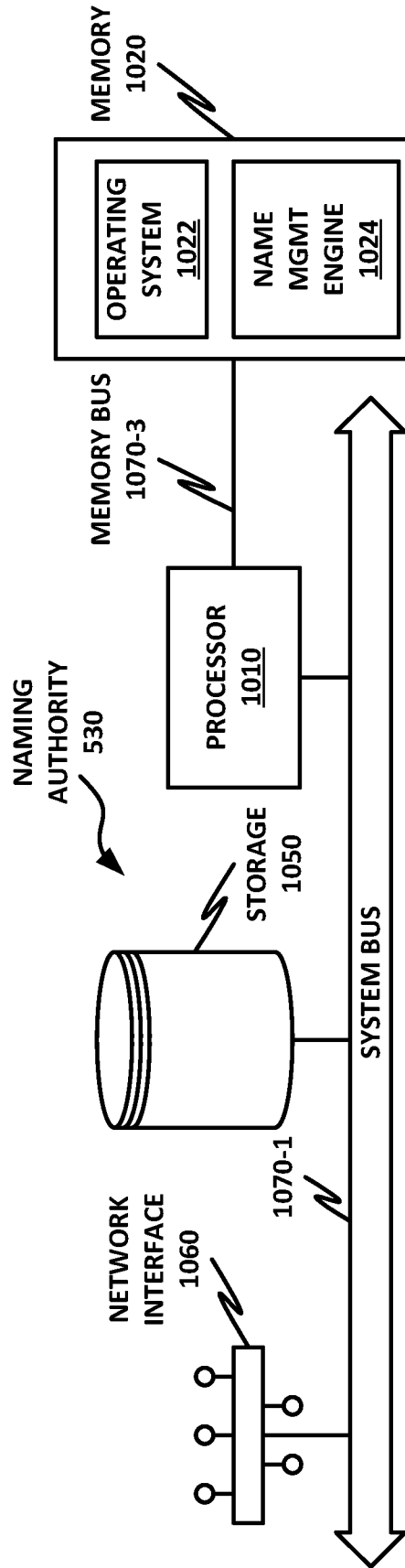


Fig. 10

10/10

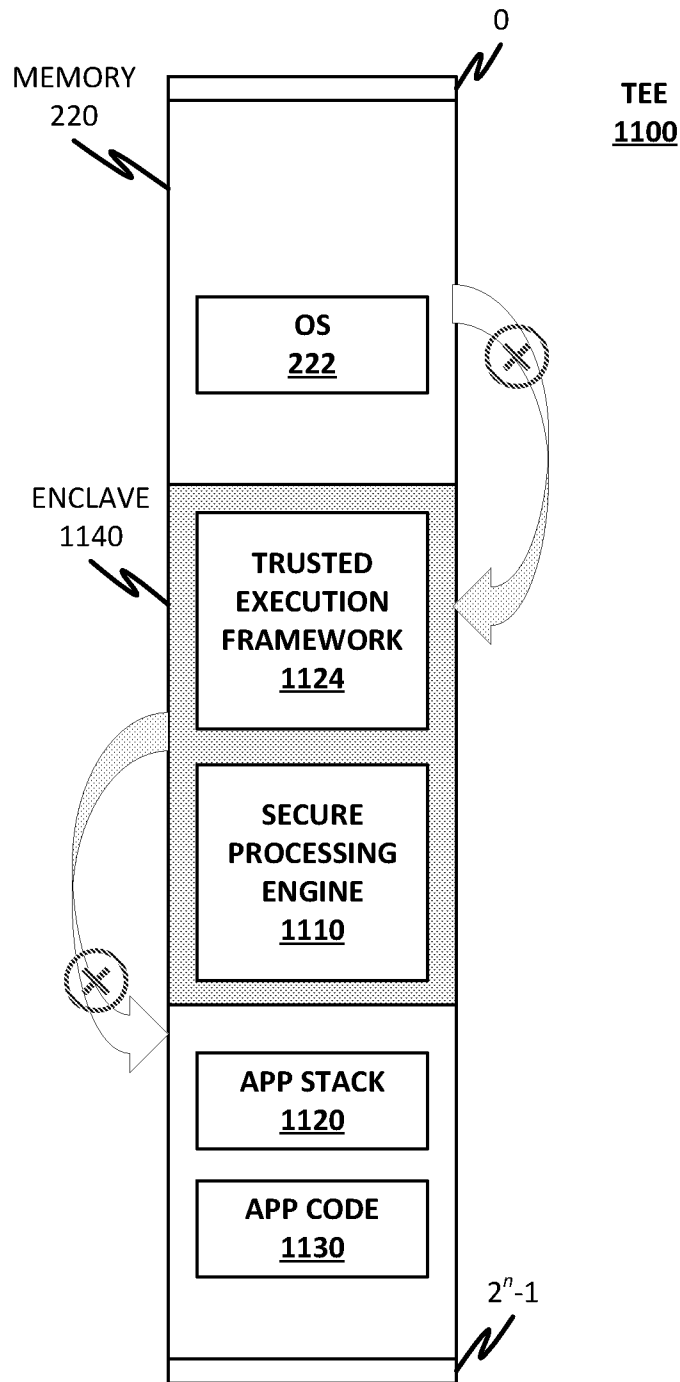


Fig. 11

A. CLASSIFICATION OF SUBJECT MATTER**H04L 29/12(2006.01)i, H04L 29/10(2006.01)i, H04L 29/08(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L 29/12; G06F 15/16; H04W 12/02; G06F 15/173; G06K 1/00; H04L 29/06; H04L 9/00; H04M 3/42; G06F 15/177; H04L 29/10; H04L 29/08

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: name, registration, certificate, secure connection

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2006-0168161 A1 (WARREN ADELMAN et al.) 27 July 2006 See paragraphs [0007], [0021], [0078]-[0079], claim 1 and figures 6-7.	1-2,5-15,18-23
A		3-4,16-17,24-25
Y	US 2005-0086377 A1 (TAKAHIRO ASO) 21 April 2005 See paragraphs [0058]-[0062], claims 6, 9 and figures 2, 6.	1-2,5-15,18-23
Y	US 2007-0150737 A1 (GOPAL PARUPUDI et al.) 28 June 2007 See paragraphs [0016], [0031] and figure 4.	5-7,18-19
Y	WO 2014-194858 A1 (QUANTUM COMMUNICATION TECHNOLOGY CO., LTD., ANHUI) 11 December 2014 See claims 1, 14 and figure 2.	24-25
Y	US 2013-0311774 A1 (VIRNETX, INC.) 21 November 2013 See paragraphs [0156], [0258], claim 29 and figure 26.	24-25
A	US 2008-0043973 A1 (YONGHONG LAI et al.) 21 February 2008 See claims 1-10 and figure 1.	1-25

 Further documents are listed in the continuation of Box C. See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

08 August 2016 (08.08.2016)

Date of mailing of the international search report

08 August 2016 (08.08.2016)

Name and mailing address of the ISA/KR

International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

LEE, EUN KYU

Telephone No. +82-42-481-3580



Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Group I , claims 1-23, drawn to a computing apparatus for registering a name of a device.

Group II , claims 24-25, drawn to a computing apparatus for establishing a secure channel and issuing a key management ticket.

The inventions listed as Groups I - II do not relate to a single general inventive concept under PCT Rule 13.1, because under PCT Rule 13.2 they lack the same or corresponding special technical features for the following reason; they are separate inventions with distinct fields of search.

1. As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of any additional fees.
3. As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

- The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/031590

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006-0168161 A1	27/07/2006	AU 2004-233002 A1	04/11/2004
		AU 2004-236260 A1	18/11/2004
		AU 2004-256392 A1	20/01/2005
		AU 2005-218592 A1	15/09/2005
		AU 233002 B2	15/04/2010
		AU 236260 B2	01/04/2010
		AU 256392 B2	01/10/2009
		CA 2522678 A1	04/11/2004
		CA 2522678 C	19/06/2012
		CA 2522866 A1	20/01/2005
		CA 2522872 A1	18/11/2004
		CA 2522872 C	29/04/2014
		CA 2557757 A1	15/09/2005
		CA 2557757 C	22/01/2013
		EP 1617769 A2	25/01/2006
		EP 1617769 B1	04/11/2015
		EP 1618223 A2	25/01/2006
		EP 1620256 A2	01/02/2006
		EP 1725189 A1	29/11/2006
		EP 1869237 A2	26/12/2007
		EP 2114299 A1	11/11/2009
		JP 2006-526059 A	16/11/2006
		JP 2007-512211 A	17/05/2007
		JP 2007-526439 A	13/09/2007
		JP 2007-533371 A	22/11/2007
		JP 4741477 B2	03/08/2011
		JP 4871726 B2	08/02/2012
		JP 4889484 B2	07/03/2012
		JP 5039539 B2	03/10/2012
		TW I427709 B	21/02/2014
		US 2004-0206448 A1	21/10/2004
		US 2004-0250950 A1	16/12/2004
		US 2005-0038498 A1	17/02/2005
		US 2005-0181195 A1	18/08/2005
		US 2005-0221072 A1	06/10/2005
		US 2005-0289084 A1	29/12/2005
		US 2006-0031492 A1	09/02/2006
		US 2006-0122596 A1	08/06/2006
		US 2006-0159916 A1	20/07/2006
		US 2006-0161644 A1	20/07/2006
		US 2006-0165952 A1	27/07/2006
		US 2006-0168116 A1	27/07/2006
		US 2006-0204738 A1	14/09/2006
		US 2007-0275232 A1	29/11/2007
		US 2007-0282247 A1	06/12/2007
		US 2009-0162643 A1	25/06/2009
		US 2010-0140160 A1	10/06/2010
		US 2010-0161971 A1	24/06/2010
		US 2010-0285972 A1	11/11/2010

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/031590

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 2011-0201984 A1	18/08/2011
		US 7056409 B2	06/06/2006
		US 7074294 B2	11/07/2006
		US 7344617 B2	18/03/2008
		US 7579077 B2	25/08/2009
		US 7651769 B2	26/01/2010
		US 7702902 B2	20/04/2010
		US 7707404 B2	27/04/2010
		US 7803574 B2	28/09/2010
		US 7972616 B2	05/07/2011
		US 7985475 B2	26/07/2011
		US 8086848 B2	27/12/2011
		US 8103761 B2	24/01/2012
		US 8285816 B2	09/10/2012
		US 8956637 B2	17/02/2015
		WO 2004-094303 A2	04/11/2004
		WO 2004-094303 A3	04/11/2004
		WO 2004-099068 A2	18/11/2004
		WO 2004-099068 A3	18/11/2004
		WO 2005-005679 A2	20/01/2005
		WO 2005-005679 A3	20/01/2005
		WO 2005-084582 A1	15/09/2005
		WO 2007-078304 A2	12/07/2007
		WO 2007-078304 A3	12/07/2007
		WO 2008-103464 A1	28/08/2008
US 2005-0086377 A1	21/04/2005	DE 602004031910 D1	05/05/2011
		EP 1517519 A1	23/03/2005
		EP 1517519 B1	23/03/2011
		JP 2005-094252 A	07/04/2005
		JP 4101140 B2	18/06/2008
		US 7979581 B2	12/07/2011
US 2007-0150737 A1	28/06/2007	US 7600123 B2	06/10/2009
WO 2014-194858 A1	11/12/2014	CN 104243143 A	24/12/2014
		EP 3007478 A1	13/04/2016
		US 2016-0127127 A1	05/05/2016
US 2013-0311774 A1	21/11/2013	US 2004-0098485 A1	20/05/2004
		US 2008-0040783 A1	14/02/2008
		US 2008-0040792 A1	14/02/2008
		US 2011-0167087 A1	07/07/2011
		US 2011-0185053 A1	28/07/2011
		US 2012-0096541 A1	19/04/2012
		US 2012-0102204 A1	26/04/2012
		US 2012-0102206 A1	26/04/2012
		US 2012-0110103 A1	03/05/2012
		US 2012-0117237 A1	10/05/2012
		US 2012-0117382 A1	10/05/2012

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/031590

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 2012-0226795 A1	06/09/2012
		US 2013-0014226 A1	10/01/2013
		US 2013-0014227 A1	10/01/2013
		US 2013-0064241 A1	14/03/2013
		US 2013-0067103 A1	14/03/2013
		US 2013-0067224 A1	14/03/2013
		US 2013-0263220 A1	03/10/2013
		US 2013-0268683 A1	10/10/2013
		US 2013-0275534 A1	17/10/2013
		US 2013-0311607 A1	21/11/2013
		US 2013-0311767 A1	21/11/2013
		US 2014-0380039 A1	25/12/2014
		US 2015-0237022 A1	20/08/2015
		US 2015-0244680 A1	27/08/2015
		US 2015-0341319 A1	26/11/2015
		US 2016-0021077 A1	21/01/2016
		US 7418504 B2	26/08/2008
		US 7921211 B2	05/04/2011
		US 7945654 B2	17/05/2011
		US 8458341 B2	04/06/2013
		US 8504696 B2	06/08/2013
		US 8504697 B2	06/08/2013
		US 8516131 B2	20/08/2013
		US 8521888 B2	27/08/2013
		US 8554899 B2	08/10/2013
		US 8560705 B2	15/10/2013
		US 8572247 B2	29/10/2013
		US 8843643 B2	23/09/2014
		US 8850009 B2	30/09/2014
		US 8868705 B2	21/10/2014
		US 8904516 B2	02/12/2014
		US 9027115 B2	05/05/2015
		US 9037713 B2	19/05/2015
		US 9038163 B2	19/05/2015
		US 9077694 B2	07/07/2015
		US 9077695 B2	07/07/2015
		US 9100375 B2	04/08/2015
US 2008-0043973 A1	21/02/2008	US 7925872 B2	12/04/2011