



- (51) **International Patent Classification:**
G06F 21/56 (2013.01) *G06F 17/22* (2006.01)
G06F 21/60 (2013.01)
- (21) **International Application Number:**
PCT/US2016/052392
- (22) **International Filing Date:**
18 September 2016 (18.09.2016)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
14/929,267 31 October 2015 (31.10.2015) US
- (71) **Applicant:** MCAFEE, INC. [US/US]; 2821 Mission College Boulevard, Santa Clara, CA 95054-1838 (US).
- (72) **Inventors:** ALME, Christoph; Vattmannstrasse 3, 33100 Paderborn, MV (DE). GEHWEILER, Joachim; Alte Brauerei 19, 33098 Paderborn (DE). MARQUARDT, Oliver Helge; Deipe Laake 1, 33106 Paderborn (DE).
- (74) **Agent:** GUPTA, Rishi; Patent Capital Group, c/o CPA Global, 900 Second Avenue South, Suite 600, 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, KW, 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).

[Continued on next page]

- (54) **Title:** DECISION FOREST COMPILATION

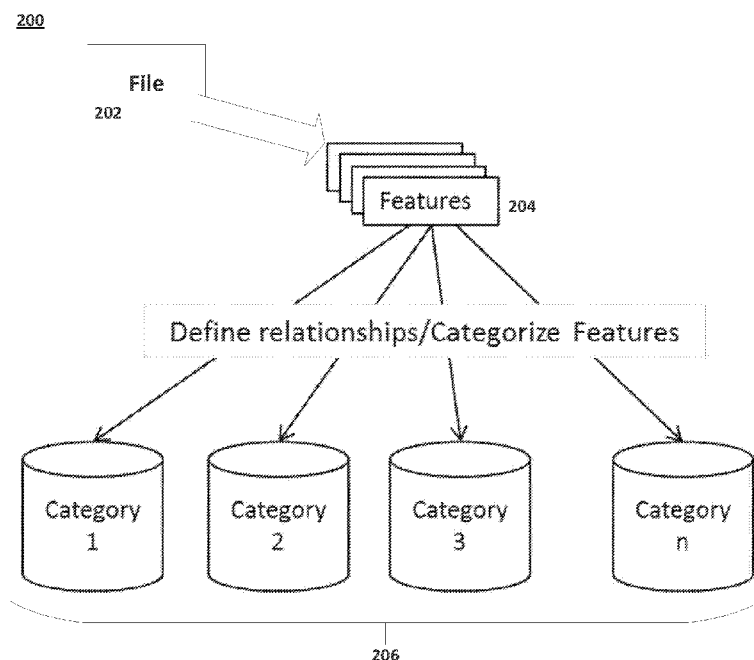


FIG. 2

- (57) **Abstract:** Embodiments of the present disclosure include methods, devices, and computer program products for detecting malware in a file. Embodiments include identifying a plurality of features of the file, categorizing each of the plurality of features to define a plurality of categories of features, building a first decision tree based on a first category from the plurality of categories, the first category comprising a first set of features of the file, and building a second decision tree based on a second category from the plurality of categories, the second decision tree comprising a second set of features of the file, the second set different from the first set. Some embodiments include comparing results from each decision tree to determine the presence or absence of malware.

**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))*

DECISION FOREST COMPILATION

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of priority to U.S. Nonprovisional (Utility) Patent Application No. 14/929,267 filed 31 October 2015 entitled, “DECISION FOREST COMPILATION”, which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present disclosure is directed to the field of computer security, and, more particularly, to defining interrelationships between features in a decision forest.

BACKGROUND

[0003] A random forest can be considered a mechanism by which decision trees are formed. Random forest decision mechanisms have been an effective solution for computer security systems in that random forests are characterized by a combination of being both reasonably generic and accurate. Random forests are additionally fast to compile at runtime, in that the decision trees need to only be interpreted once.

[0004] The problem random forest is the randomness—random forests treat all attributes equally and do not incorporate interrelationships between these attributes. As a result, random forest is a sub-optimum solution for use in malware detection.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. 1 is a schematic diagram of a system for detecting malware in a file in accordance with embodiments of the present disclosure.

[0006] FIG. 2 is a schematic diagram of features of a file categorized into one or more categories in accordance with embodiments of the present disclosure.

[0007] FIG. 3 is a schematic diagram of a decision tree forest in accordance with embodiments of the present disclosure.

[0008] FIG. 4 is a process flow diagram for defining a decision tree in accordance with embodiments of the present disclosure.

[0009] FIG. 5 is a process flow diagram for executing one or more decision trees to determine the presence or absence of malware in a file in accordance with embodiments of the present disclosure.

[0010] FIG. 6 is an example illustration of a processor according to an embodiment of the present disclosure.

[0011] FIG. 7 is a schematic block diagram of a mobile device in accordance with embodiments of the present disclosure.

[0012] FIG. 8 is a schematic block diagram of a computing system according to an embodiment of the present disclosure.

DETAILED DESCRIPTION

[0013] The present disclosure describes defining interrelationships between the features to be used in the decision forest's decision trees. The trees can be configured to incorporate known relationships or dependencies. In addition or in the alternative, known relationships or dependencies can be deliberately and/or selectively ignored if appropriate for a particular use case. Incorporating the relationships between the features considered in the trees allows for an improvement to the overall accuracy and configurability of these decision trees.

[0014] When assessing a given file for malware, one approach is to use a decision tree that incorporates several branches (or decisions), each representing a feature or file characteristic of interest in the malware assessment. Such features may include, as examples, the file size, entropy, presence of special key words, presence of I-Frame tags in HTML code, malicious Javascript code, among a multitude of other examples (including features that could be added to the set in the future). This disclosure describes using multiple different trees, each tree being used to assess the same file and resulting in a vote or score indicating the probability of whether the file is malware or not. The multiple scores from each decision tree are then aggregated and processed (e.g., using a Bayesian sum) to determine an overall decision regarding whether the file is malware.

[0015] The present disclosure describes a decision forest mechanism that can use a multi-tree approach and where each decision tree includes features that are related or interdependent. The present disclosure describes defining a set of features to be used in decision trees and to identify or define relationships between some of these features.

[0016] FIG. 1 is a schematic diagram of a system 100 for detecting malware in a file in accordance with embodiments of the present disclosure. System 100 includes a user device 102, which can be a computer, tablet, smart phone, or other computing or telecommunications device. The user device 102 includes a processor 104, which can be implemented at least partially in hardware. The user device 102 also includes a memory 122. Memory 130 may be a local memory or may be a memory accessible across a network 150. Memory 130 can store features 132 extracted from files, categories 134, decision trees 136, and decision forests 138. Memory

130 can also store decision results 140, which can include results from each decision tree in a decision forest and aggregated results from all of the decision trees in a decision forest.

[0017] The use device 102 can also include one or more modules, which can be hardware logic, software logic, or a combination of hardware and software (e.g., a hardware processor and hardware memory can execute program code). In some cases, the modules can be implemented at least partially in hardware, the hardware including a hardware processor, such as processor 104.

[0018] In some embodiments, user device 102 can include an extraction module 106 (which can include extraction logic) to extract one or more features from a file (e.g., a file received across network 150, such as an e-mail attachment or a download). The extraction module 106 can analyze the incoming file to identify one or more features of the file. In some embodiments, the extraction module 106 can identify features by calling a simulation module 112. Simulation module 112 (which can include simulation logic) can run a simulation of a run-time environment for the file to identify features associated with the file, such as file size, file type, as well as network addresses the file access, programs the file calls, etc. Results from the simulation module 112 are used for random tree/forest creation.

[0019] In some embodiments, the user device 102 can include a categorization module 108 (which can include categorization logic) to associate each of the one or more features with one or more categories. Memory 130 can store features 132 that are extracted from files and categories 134 that can include associations with features 132 based on how the features are related or interdependent. For example, two features can be associated with a category if the two features are related to each other (e.g., each feature pertains to file size or file type). The category the two features is associated with is descriptive of the relationship between the features. Each feature can be associated with one or more categories.

[0020] In some embodiments, the categorization module can also create categories for extracted features based on how the features are not related. In some embodiments, the categorization module can augment categories of related features with additional, unrelated features.

[0021] In some embodiments, the categorization module 108 is operable to identify a relationship between each of the one or more features of the file. Relationships can be identified by comparing each feature with a table or other predetermined basis. The categorization module is also operable to identify one or more categories for each of the one or more features based on the relationship between each feature. A relationship can also be identified by associating a

feature with a category, and each feature associated with the same category can be considered to be related.

[0022] The categorization module 108 can also categorize unrelated features into the same category to define a category of unrelated features.

[0023] The categorization module 108 can also augment categories of related features with unrelated features or features from other categories. In some implementations, the tree generator module can augment trees generated based on categories with additional branches using unrelated features (i.e., features not present in the category used to generate the decision tree).

[0024] In some embodiments, user device 102 can include a tree generator module 110 (which may include tree generator logic) that uses the categories and features to build decision trees. In some embodiments, the tree generator module 110 uses a category as the basis behind a decision tree. In that case, the features that make up the decision branches of the decision tree are all associated with the corresponding category. The decision tree, therefore, can include branches that are interrelated or interdependent.

[0025] In some embodiments, the decision tree can be populated with features from a single category, and augmented with features from another category (i.e., either a similar or dissimilar category). The decision tree can also include branches having varying degrees of interdependency, from “closely related” to “somewhat related” to “unrelated.”

[0026] The user device 102 can include an execution module 114 (which may include execution logic) to execute the decision trees 136 (e.g., or trees that make up a decision forest 138) to generate a first decision result 140. For example, the execution module 114 can execute a first and second decision tree to generate a first and second decision result. A classification module 116 (which can include classification logic) can determine whether the file has malware based on the first decision result and the second decision result. For example, each decision tree can result in a vote or score indicating the probability of whether the file is malware or not. The multiple scores from each decision tree can then be aggregated and processed (e.g., using a Bayesian sum) to determine an overall decision regarding whether the file is malware.

[0027] The system 100 may also include a server 120 that can include the same or similar modules as the user device 102 that operate in the same or similar fashion as described above. The user device 102 can access modules running on server 120 across network 150 to test files for malware. The server 120 can include a memory 130 that is local or accessible across a network 150. The user device 102 can access remote applications or cloud based applications across network 150 that are run on server 120. Server 120 can include a processor

122 that is implemented at least partially in hardware. The processor 122 can execute instructions received from user device 102 across network 150. Processor 122 run modules 124, which can include any combination of the extraction module 106, categorization module 108, tree generator module 11, simulation module 112, execution module 114, and classification module 116.

[0028] FIG. 2 is a schematic diagram 200 of features of a file categorized into one or more categories in accordance with embodiments of the present disclosure. A file 202 can be received by a user device across a network, such as a downloaded file or an e-mail attachment. Features 204 of the file can be extracted. Each feature can be associated with or assigned to a category 206, such as category 1 through category n for “n” number of categories.

[0029] The features 202 of the file can be categorized based on their similarity. For instance, categories of features can be defined, such as semantic, geometric, behavioral features, etc. Data structures can be defined to map attributes according to these relationships (e.g., using a Markov chain). This definition of relationships can then be used during compilation of the decision trees to be included in a forest of decision trees for use during malware assessment based on the relative degree of relationships between the parent’s (or previously selected branch’s) feature and the feature(s) that is to be used in the subsequent (or child) branches.

[0030] For example, a first feature can be the number of readable strings in the file and a second feature can be the entropy of a file. These example features are related, so both would be associated with the same category.

[0031] FIG. 3 is a schematic diagram of a decision forest 300 in accordance with embodiments of the present disclosure. Decision forest 300 includes decision trees associated with one or more categories. For example, decision forest 300 includes decision tree 1 302 associated with category 1, decision tree 2 304 associated with category 2, and decision tree n 306 associated with category n, e.g., for n categories.

[0032] Decision tree 1 302 includes a parent branch A that is representative of a feature and child branches B and E. The features A and B and E can be selected from a set of features that make up the category 1. In some embodiments, feature A is selected such that the first decision reduces the number of child branches by a relatively large number. For example, a file size can be selected at a half-way point between typical file sizes. If feature A is “file size,” a file size decision can be “below X bytes” and “equal to or above X bytes,” where X is the half-way point previously determined. Such a feature would reduce the potential results by approximately half, which can reduce the total number of branches required to traverse before reaching the end of the decision tree.

[0033] In some embodiments, each branch of a decision tree is related to other branches. For example, in decision tree 1 302, feature A that makes up a parent branch is related to feature B and feature E, which make up child branches to feature A. Features B and E may also be related. The degree to which a child branch is related to its parent is configurable prior to compilation. An administrator or developer can select that one or more (or all) of the trees in a given forest consider the defined degrees of relation between features when building the tree (i.e., determining the next child branch for each parent).

[0034] If a more generic tree is desired, the tree generator selects child features with a closer relation to the feature of the parent. For example, feature N of decision tree n can be from the same category as features B and C.

[0035] If more accurate trees are desired, the tree generator is to select child features (branching from a given parent feature) with a greater distance of relation (e.g., from a different category of features) of the parent. For example, feature A of decision tree 2 304 maybe associated with a different category than feature F.

[0036] A decision tree can also include a mix of related and unrelated features. For example, decision tree 2 can include feature A that is related to feature D (i.e., A and D are both associated with category 2), and decision tree 2 can include features F and G, both of which are associated with a different category than features A and D.

[0037] The number of decision trees generated is defined by an administrator. As an example, the number of categories can equal the number of decision trees; or, a specific number of decision trees can be generated based on preferences, accuracy targets, etc. Using this platform, optimal tree formats can be determined for delivering accuracy in malware detection. This optimization can be accomplished through a separate process (e.g., by the simulation module 112) that iterates through each possible tree permutation of features to determine which tree configuration(s) deliver the most accurate results (e.g., based on assessments of test “clean” and “dirty” files). This iterative process would be a resource-intensive exercise as the number of permutations can be very large, but because these relationships do not generally change over the short term, such an exercise would be rare (e.g., yearly, bimonthly, etc.). A heuristic analysis could be performed to determine a subset of the universe of potential forest permutations that are most likely to be the most accurate, and the test would be run against this subset to determine which of the trees (and tree combinations) yields the most accurate results with the set of features we utilize. Regardless of the method, the optimized forest could then be used as a baseline of the most accurate forest. In embodiments where accuracy is desired, this baseline can be applied in compiling the forest to be used.

[0038] In other cases, where generic results are desirable, a baseline can be determined and used that attempts to develop trees with less “distance” between the parent and child branches.

[0039] FIG. 4 is a process flow diagram 400 for defining a decision tree in accordance with embodiments of the present disclosure. A user device can receive a file (e.g., from a network location, via a download or e-mail) (402). The user device can identify and extract features of the file (404). The features can be any type of features that provide information about the file, such as file size, file type, file structure, compression, file source (where the file came from), who sent the file, types of programs that can run the file, behavior while simulating the file content, entropy, number of bit strings, what resources the file will call upon, what networked locations that file accesses, etc.

[0040] In some embodiments, the features are associated with one or more categories (408). Each feature can be predetermined to be associated with one or more categories. And features associated with the same category can be said to be related. In some embodiments, a category can be defined so that features associated with that category are unrelated.

[0041] In some embodiments, the user device can identify relationships between features (406). This identification can be based on a predetermined relationship between known features. For example, if features include file entropy and the number of bit strings, the user device can cross reference a predetermined table or database or use other identifiers to relate the features.

[0042] Table 1 shows how identified features can be associated with categories:

Table 1. Feature associations with categories.

Feature A	Cat 1	Cat 2	Cat 6	Cat 10	→	Cat 20
Feature B	Cat 1	Cat 3	Cat 8	Cat 11		
Feature C	Cat 4				→	Cat 20
Feature D	Cat 1	Cat 2	Cat 10			

[0043] Table 1 shows how features A–D can be associated with categories. For example, when user device identifies feature A, the user device can consult a predefined association matrix to identify categories for feature A. In this case, Feature A can be associated with Categories 1, 2, 6, 10, and 20. Likewise, Feature B can be associated with Categories 1, 3, 8, and 11. Feature A and Feature B, therefor can be considered to be related (because they are both associated with Category 1; however, Feature A and Feature B may be considered only “somewhat” related because Features A and B are only associated with one overlapping category (Category 1). By contrast, Features A and D may be considered more related because Features A and D are associated with three overlapping categories (Categories 1, 2, and 10).

[0044] In some embodiments, unrelated features can be associated with a category. For example, Feature A and Feature C do not overlap categories, so the user device can consider Features A and C to be unrelated. The user device can then associate Features A and C with Category 20, which is reserved for unrelated features.

[0045] The user device can build decision trees based on the features from the categories (410). A plurality of decision trees constitutes a decision forest. Each tree can be representative of a category of related features, so that the branches of each tree are related or interdependent. In some embodiments, a decision tree can be populated by unrelated features. In some embodiments, a decision tree that includes related features can be augmented with one or more unrelated features (i.e., features that are associated with a different category than that used to generate the decision tree) (412).

[0046] In some embodiments, one or more features can be extracted and identified that have no known relation to other features (e.g., unrelated features) (414). A decision tree can be built using features that have no known relation to other features (416). The decision forest can be compiled using decision trees built from categories of related features (and, in some cases augmented with unrelated features) and, in some cases, from decision trees built from features that have no known relation.

[0047] FIG. 5 is a process flow diagram 500 for executing one or more decision trees to determine the presence or absence of malware in a file in accordance with embodiments of the present disclosure. A user device can execute each decision tree of a decision forest (502). The user device can ascertain a result for each decision tree (504). The user device can determine the presence or absence of malware based on a combined result from each decision tree in the decision forest (506).

[0048] For example, each decision tree can result in a vote or score indicating the probability of whether the file is malware or not. The multiple scores from each decision tree can then be aggregated and processed (e.g., using a Bayesian sum) to determine an overall decision regarding whether the file is malware.

[0049] Although this disclosure has been described in terms of certain implementations and generally associated methods, alterations and permutations of these implementations and methods will be apparent to those skilled in the art. For example, the actions described herein can be performed in a different order than as described and still achieve the desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve the desired results. In certain implementations, multitasking and parallel processing may be advantageous. Additionally,

other user interface layouts and functionality can be supported. Other variations are within the scope of the following claims.

[0050] In example 1, aspects of the embodiments are directed to identifying, by extraction logic implemented at least partially in hardware, a plurality of features of the file, categorizing, by categorization logic implemented at least partially in hardware, each of the plurality of features to define a plurality of categories of features, building, by tree generator logic implemented at least partially in hardware, a first decision tree based on a first category from the plurality of categories, the first category comprising a first set of features of the file, and building, by the tree generator logic, a second decision tree based on a second category from the plurality of categories, the second decision tree comprising a second set of features of the file, the second set different from the first set. Further decision trees can be built using other categories. The decision trees can be compiled to form a decision forest.

[0051] In example 2, the subject matter of example 1 may also include identifying, by the categorization logic, a relationship between each feature of the file, and identifying one or more categories for the features based on the relationship between each feature.

[0052] In example 3, the subject matter of any of examples 1 or 2 may also include identifying, by the categorization logic, one or more categories for the features based on the relationship between each feature comprises categorizing related features into a same category to define a category of related features.

[0053] In example 4, the subject matter of any of examples 1 or 2 or 3 may also include identifying, by the categorization logic, one or more categories for the features based on the relationship between each feature by categorizing unrelated features into the same category to define a category of features with no known relationship.

[0054] In example 5, the subject matter of any of examples 1 or 2 may also include categorizing, by the categorization logic, each of the plurality of features to define a plurality of categories of features by associating for each feature of the plurality of features of the file with one or more category.

[0055] In example 6, the subject matter of any of examples 1 or 2 or 3 or 4 or 5 can also include augmenting, by the tree generator logic, the first decision tree with a feature that is not related to a feature in the first set of features.

[0056] In example 7, the subject matter of any of examples 1 or 2 or 3 or 4 or 5 or 6 can also include executing, by execution logic implemented at least partially in hardware, the first decision tree to generate a first decision result, executing the second decision tree to generate a second decision result, and determining, by classification logic implemented at least partially in

hardware, that the file has malware based on the first decision result and the second decision result.

[0057] In example 8, aspects of the embodiments are directed to computing device. The computing device may include an extraction means for extracting one or more features from a file; a categorization means for associating each of the one or more features with one or more categories; and a tree generator means for generating a decision tree based on the one or more categories.

[0058] In example 9, the subject matter of example 8 may also include an execution means for executing a first decision tree to generate a first decision result, the first decision tree comprising a first set of features, and executing the second decision tree to generate a second decision result, the second decision tree comprising a second set of features, the first set different from the second set. The subject matter of example 8 may also include a classification means for determining whether the file has malware based on the first decision result and the second decision result.

[0059] In example 10, the subject matter of any of examples 8 or 9 may also include categorization means for identifying a relationship between each of the one or more feature of the file; and identifying one or more categories for each of the one or more features based on the relationship between each feature.

[0060] In example 11, aspects of the embodiments are directed to a computer program product tangibly embodied on non-transient computer readable media. The computer program product includes instructions operable when executed to extract, by an extraction module implemented at least partially in hardware, a plurality of features of the file; categorize by an categorization module implemented at least partially in hardware each of the plurality of features to define a plurality of categories of features; build by an tree generator module implemented at least partially in hardware a first decision tree based on a first category from the plurality of categories, the first category comprising a first set of features of the file; and build a second decision tree based on a second category from the plurality of categories, the second decision tree comprising a second set of features of the file, the second set of features of the file different from the first set of features of the file.

[0061] In example 12, the subject matter of example 11 may also include categorizing by the categorization module each of the plurality of features may include identifying a relationship between each feature of the file; and identifying one or more categories for the features based on the relationship between each feature.

[0062] In example 13, the subject matter of any of examples 11 or 12 may also include identifying by the categorization module one or more categories for the features based on the relationship between each feature includes categorizing related features into a same category to define a category of related features.

[0063] In example 14, the subject matter of any of examples 11 or 12 or 13 may also include identifying by the categorization module one or more categories for the features based on the relationship between each feature includes categorizing unrelated features into the same category to define a category of unrelated features.

[0064] In example 15, the subject matter of any of examples 11 or 12 or 13 or 14 may also include categorizing by the categorization module each of the plurality of features to define a plurality of categories of features includes associating for each feature of the plurality of features of the file with one or more category.

[0065] In example 16, the subject matter of any of examples 11 or 12 or 13 or 14 or 15 may also include instructions to augment by the tree generator module the first decision tree with a feature that is not related to a feature in the first set of features.

[0066] In example 17, the subject matter of any of examples 11 or 12 or 13 or 14 or 15 or 16 may also include instructions to execute, by an execution module, the first decision tree to generate a first decision result; execute by the execution module the second decision tree to generate a second decision result; and determine, by a classification module, whether the file has malware based on the first decision result and the second decision result.

[0067] In example 18, the subject matter of any of examples 11 or 12 or 13 or 14 or 15 or 16 or 17 may also include instructions to augment, by the categorization module, one or more predefined categories with one or more features of the file.

[0068] In example 19, aspects of the embodiments are directed to a computing device that includes an extraction logic implemented at least partially in hardware to extract one or more features from a file; a categorization logic to associate each of the one or more features with one or more categories; and a tree generator logic to generate a decision tree based on the one or more categories.

[0069] In example 20, the subject matter of example 19 may also include an execution logic to execute a first decision tree to generate a first decision result, the first decision tree comprising a first set of features, and execute by the execution logic the second decision tree to generate a second decision result, the second decision tree comprising a second set of features, the first set different from the second set. The computing device can also include a classification

logic to determine whether the file has malware based on the first decision result and the second decision result.

[0070] In example 21, the subject matter of any of claims 19 or 20 may also include that the categorization logic is further operable to identify a relationship between each of the one or more feature of the file; and identify one or more categories for each of the one or more features based on the relationship between each feature.

[0071] In example 22, the subject matter of claim 21 may also include that identifying one or more categories for the features based on the relationship between each feature may include categorizing related features into a same category to define a category of related features.

[0072] In example 23, the subject matter of claim 21 may also include identifying one or more categories for the features based on the relationship between each feature may include categorizing unrelated features into the same category to define a category of unrelated features.

[0073] In example 24, the subject matter of any of claims 19 or 20 or 21 or 22 or 23 may also include that categorizing each of the plurality of features to define a plurality of categories of features may include associating for each feature of the plurality of features of the file with one or more category.

[0074] In example 25, the subject matter of any of claims 19 or 20 or 21 or 22 or 23 or 24 may also include instructions to augment by the tree generator logic the first decision tree with a feature that is not related to a feature in the first set of features.

[0075] In example 26, aspects of the embodiments are directed to identifying, by extraction logic implemented at least partially in hardware, a plurality of features of the file, categorizing, by categorization logic implemented at least partially in hardware, at least some of the plurality of features to define a plurality of categories of features, building, by tree generator logic implemented at least partially in hardware, a first decision tree based on a first category from the plurality of categories, the first category comprising a first set of features of the file, and building, by the tree generator logic, a second decision tree based on a plurality of features selected from different categories, wherein each feature of the second decision tree is.

[0076] In example 27, aspects of the embodiments are directed to a computer program product tangibly embodied on non-transient computer readable media. The computer program product includes instructions operable when executed to extract, by an extraction module implemented at least partially in hardware, a plurality of features of the file; categorize by an categorization module implemented at least partially in hardware each of the plurality of features to define a plurality of categories of features; build by an tree generator module implemented at

least partially in hardware a first decision tree based on a first category from the plurality of categories, the first category comprising a first set of features of the file; and build a second decision tree based on a set of features with no known relation to other features.

[0077] FIGS. 6–8 are block diagrams of exemplary computer architectures that may be used in accordance with embodiments disclosed herein. Other computer architecture designs known in the art for processors, mobile devices, and computing systems may also be used. Generally, suitable computer architectures for embodiments disclosed herein can include, but are not limited to, configurations illustrated in FIGS. 6–8.

[0078] FIG. 6 is an example illustration of a processor according to an embodiment. Processor 600 is an example of a type of hardware device that can be used in connection with the implementations above.

[0079] Processor 600 may be any type of processor, such as a microprocessor, an embedded processor, a digital signal processor (DSP), a network processor, a multi-core processor, a single core processor, or other device to execute code. Although only one processor 600 is illustrated in FIG. 6, a processing element may alternatively include more than one of processor 600 illustrated in FIG. 6. Processor 600 may be a single-threaded core or, for at least one embodiment, the processor 600 may be multi-threaded in that it may include more than one hardware thread context (or “logical processor”) per core.

[0080] FIG. 6 also illustrates a memory 602 coupled to processor 600 in accordance with an embodiment. Memory 602 may be any of a wide variety of memories (including various layers of memory hierarchy) as are known or otherwise available to those of skill in the art. Such memory elements can include, but are not limited to, random access memory (RAM), read only memory (ROM), logic blocks of a field programmable gate array (FPGA), erasable programmable read only memory (EPROM), and electrically erasable programmable ROM (EEPROM).

[0081] Processor 600 can execute any type of instructions associated with algorithms, processes, or operations detailed herein. Generally, processor 600 can transform an element or an article (e.g., data) from one state or thing to another state or thing.

[0082] Code 604, which may be one or more instructions to be executed by processor 600, may be stored in memory 602, or may be stored in software, hardware, firmware, or any suitable combination thereof, or in any other internal or external component, device, element, or object where appropriate and based on particular needs. In one example, processor 600 can follow a program sequence of instructions indicated by code 604. Each instruction enters a front-end logic 606 and is processed by one or more decoders 608. The decoder may generate,

as its output, a micro operation such as a fixed width micro operation in a predefined format, or may generate other instructions, microinstructions, or control signals that reflect the original code instruction. Front-end logic 606 also includes register renaming logic 610 and scheduling logic 612, which generally allocate resources and queue the operation corresponding to the instruction for execution.

[0083] Processor 600 can also include execution logic 614 having a set of execution units 616a, 616b, 616n, etc. Some embodiments may include a number of execution units dedicated to specific functions or sets of functions. Other embodiments may include only one execution unit or one execution unit that can perform a particular function. Execution logic 614 performs the operations specified by code instructions.

[0084] After completion of execution of the operations specified by the code instructions, back-end logic 618 can retire the instructions of code 604. In one embodiment, processor 600 allows out of order execution but requires in order retirement of instructions. Retirement logic 620 may take a variety of known forms (e.g., re-order buffers or the like). In this manner, processor 600 is transformed during execution of code 604, at least in terms of the output generated by the decoder, hardware registers and tables utilized by register renaming logic 610, and any registers (not shown) modified by execution logic 614.

[0085] Although not shown in FIG. 6, a processing element may include other elements on a chip with processor 600. For example, a processing element may include memory control logic along with processor 600. The processing element may include I/O control logic and/or may include I/O control logic integrated with memory control logic. The processing element may also include one or more caches. In some embodiments, non-volatile memory (such as flash memory or fuses) may also be included on the chip with processor 600.

[0086] Referring now to FIG. 7, a block diagram is illustrated of an example mobile device 700. Mobile device 700 is an example of a possible computing system (e.g., a host or endpoint device) of the examples and implementations described herein. In an embodiment, mobile device 700 operates as a transmitter and a receiver of wireless communications signals. Specifically, in one example, mobile device 700 may be capable of both transmitting and receiving cellular network voice and data mobile services. Mobile services include such functionality as full Internet access, downloadable and streaming video content, as well as voice telephone communications.

[0087] Mobile device 700 may correspond to a conventional wireless or cellular portable telephone, such as a handset that is capable of receiving "3G", or "third generation" cellular

services. In another example, mobile device 700 may be capable of transmitting and receiving “4G” mobile services as well, or any other mobile service.

[0088] Examples of devices that can correspond to mobile device 700 include cellular telephone handsets and smartphones, such as those capable of Internet access, email, and instant messaging communications, and portable video receiving and display devices, along with the capability of supporting telephone services. It is contemplated that those skilled in the art having reference to this specification will readily comprehend the nature of modern smartphones and telephone handset devices and systems suitable for implementation of the different aspects of this disclosure as described herein. As such, the architecture of mobile device 700 illustrated in FIG. 7 is presented at a relatively high level. Nevertheless, it is contemplated that modifications and alternatives to this architecture may be made and will be apparent to the reader, such modifications and alternatives contemplated to be within the scope of this description.

[0089] In an aspect of this disclosure, mobile device 700 includes a transceiver 702, which is connected to and in communication with an antenna. Transceiver 702 may be a radio frequency transceiver. Also, wireless signals may be transmitted and received via transceiver 702. Transceiver 702 may be constructed, for example, to include analog and digital radio frequency (RF) 'front end' functionality, circuitry for converting RF signals to a baseband frequency, via an intermediate frequency (IF) if desired, analog and digital filtering, and other conventional circuitry useful for carrying out wireless communications over modern cellular frequencies, for example, those suited for 3G or 4G communications. Transceiver 702 is connected to a processor 704, which may perform the bulk of the digital signal processing of signals to be communicated and signals received, at the baseband frequency. Processor 704 can provide a graphics interface to a display element 708, for the display of text, graphics, and video to a user, as well as an input element 710 for accepting inputs from users, such as a touchpad, keypad, roller mouse, and other examples. Processor 704 may include an embodiment such as shown and described with reference to processor 600 of FIG. 6.

[0090] In an aspect of this disclosure, processor 704 may be a processor that can execute any type of instructions to achieve the functionality and operations as detailed herein. Processor 704 may also be coupled to a memory element 706 for storing information and data used in operations performed using the processor 704. Additional details of an example processor 704 and memory element 706 are subsequently described herein. In an example embodiment, mobile device 700 may be designed with a system-on-a-chip (SoC) architecture, which

integrates many or all components of the mobile device into a single chip, in at least some embodiments.

[0091] FIG. 8 is a schematic block diagram of a computing system 800 according to an embodiment. In particular, FIG. 8 shows a system where processors, memory, and input/output devices are interconnected by a number of point-to-point interfaces. Generally, one or more of the computing systems described herein may be configured in the same or similar manner as computing system 800.

[0092] Processors 870 and 880 may also each include integrated memory controller logic (MC) 872 and 882 to communicate with memory elements 832 and 834. In alternative embodiments, memory controller logic 872 and 882 may be discrete logic separate from processors 870 and 880. Memory elements 832 and/or 834 may store various data to be used by processors 870 and 880 in achieving operations and functionality outlined herein.

[0093] Processors 870 and 880 may be any type of processor, such as those discussed in connection with other figures. Processors 870 and 880 may exchange data via a point-to-point (PtP) interface 850 using point-to-point interface circuits 878 and 888, respectively. Processors 870 and 880 may each exchange data with a chipset 890 via individual point-to-point interfaces 852 and 854 using point-to-point interface circuits 876, 886, 894, and 898. Chipset 890 may also exchange data with a high-performance graphics circuit 838 via a high-performance graphics interface 839, using an interface circuit 892, which could be a PtP interface circuit. In alternative embodiments, any or all of the PtP links illustrated in FIG. 8 could be implemented as a multi-drop bus rather than a PtP link.

[0094] Chipset 890 may be in communication with a bus 820 via an interface circuit 896. Bus 820 may have one or more devices that communicate over it, such as a bus bridge 818 and I/O devices 816. Via a bus 810, bus bridge 818 may be in communication with other devices such as a keyboard/mouse 812 (or other input devices such as a touch screen, trackball, etc.), communication devices 826 (such as modems, network interface devices, or other types of communication devices that may communicate through a computer network 860), audio I/O devices 814, and/or a data storage device 828. Data storage device 828 may store code 830, which may be executed by processors 870 and/or 880. In alternative embodiments, any portions of the bus architectures could be implemented with one or more PtP links.

[0095] The computer system depicted in FIG. 8 is a schematic illustration of an embodiment of a computing system that may be utilized to implement various embodiments discussed herein. It will be appreciated that various components of the system depicted in FIG. 8 may be combined in a system-on-a-chip (SoC) architecture or in any other suitable configuration

capable of achieving the functionality and features of examples and implementations provided herein.

[0096] Although this disclosure has been described in terms of certain implementations and generally associated methods, alterations and permutations of these implementations and methods will be apparent to those skilled in the art. For example, the actions described herein can be performed in a different order than as described and still achieve the desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve the desired results. In certain implementations, multitasking and parallel processing may be advantageous. Additionally, other user interface layouts and functionality can be supported. Other variations are within the scope of the claims.

[0097] While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any inventions or of what may be claimed, but rather as descriptions of features specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0098] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

[0099] Thus, particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. In some cases, the actions recited in the claims can be performed in a different order and still achieve desirable results. In addition,

the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results.

Claims:

1. A computer program product tangibly embodied on non-transient computer readable media, the computer program product comprising instructions operable when executed to:
 - receive a file from a network location;
 - extracting, by an extraction module implemented at least partially in hardware, a plurality of features of a file;
 - categorize, by a categorization module implemented at least partially in hardware, each of the plurality of features to define a plurality of categories of features;
 - build, by a tree generator module implemented at least partially in hardware, a first decision tree based on a first category from the plurality of categories, the first category comprising a first set of features of the file; and
 - build, by the tree generator module, a second decision tree based on a second category from the plurality of categories, the second decision tree comprising a second set of features of the file, the second set of features of the file different from the first set of features of the file.
2. The computer program product of claim 1, wherein categorizing each of the plurality of features comprises:
 - identifying a relationship between each feature of the file; and
 - identifying one or more categories for the features based on the relationship between each feature.
3. The computer program product of claim 2, wherein identifying one or more categories for the features based on the relationship between each feature comprises categorizing related features into a same category to define a category of related features.
4. The computer program product of claim 2, wherein identifying one or more categories for the features based on the relationship between each feature comprises categorizing unrelated features into the same category to define a category of unrelated features.
5. The computer program product of claim 1, wherein categorizing each of the plurality of features to define a plurality of categories of features comprises associating for each feature of the plurality of features of the file with one or more category.

6. The computer program product of claim 1, the instructions further operable to augment, by the tree generator module, the first decision tree with a feature that is not related to a feature in the first set of features.

7. The computer program product of claim 1 (or any claim), the instructions further operable to:

execute, by an execution module implemented at least partially in hardware, the first decision tree to generate a first decision result;

execute, by the execution module, the second decision tree to generate a second decision result; and

determine, by a classification module implemented at least partially in hardware, whether the file has malware based on the first decision result and the second decision result.

8. The computer program product of claim 1, the instructions further operable to augment, by the categorization module, one or more predefined categories with one or more features of the file.

9. A computer implemented method for assessing a file for malware, the method comprising:

receiving the file from a network location;

extracting, by extraction logic implemented at least partially in hardware, a plurality of features of the file;

categorizing, by categorization logic implemented at least partially in hardware, each of the plurality of features to define a plurality of categories of features;

building, by a tree generator logic implemented at least partially in hardware, a first decision tree based on a first category from the plurality of categories, the first category comprising a first set of features of the file; and

building, by the tree generator logic implemented at least partially in hardware, a second decision tree based on a second category from the plurality of categories, the second decision tree comprising a second set of features of the file, the second set different from the first set.

10. The computer implemented method of claim 9, wherein categorizing each of the plurality of features comprises:

identifying a relationship between each feature of the file; and

identifying one or more categories for the features based on the relationship between each feature.

11. The computer implemented method of claim 10, wherein identifying one or more categories for the features based on the relationship between each feature comprises categorizing related features into a same category to define a category of related features.

12. The computer implemented method of claim 10 or 11, wherein identifying one or more categories for the features based on the relationship between each feature comprises categorizing unrelated features into the same category to define a category of unrelated features.

13. The computer implemented method of claim 9, wherein categorizing each of the plurality of features to define a plurality of categories of features comprises associating for each feature of the plurality of features of the file with one or more category.

14. The computer implemented method of claim 9, further comprising augmenting, by the tree generator logic, the first decision tree with a feature that is not related to a feature in the first set of features.

15. The computer implemented method of claim 9 (any claim), further comprising:
executing, by an execution logic implemented at least partially in hardware, the first decision tree to generate a first decision result;

executing, by the execution logic, the second decision tree to generate a second decision result;

determining, by a classification logic implemented at least partially in hardware, that the file has malware based on the first decision result and the second decision result.

16. The computer implemented method of claim 9, further comprising augmenting, by the tree generator logic, one or more predefined categories with one or more features of the file.

17. A computing device comprising:
extraction logic implemented at least partially in hardware to extract one or more features from a file;

a categorization logic implemented at least partially in hardware to associate each of the one or more features with one or more categories; and

a tree generator logic implemented at least partially in hardware to generate a decision tree based on the one or more categories.

18. The computing device of claim 17, further comprising:

an execution logic implemented at least partially in hardware to:

execute a first decision tree to generate a first decision result, the first decision tree comprising a first set of features, and

execute the second decision tree to generate a second decision result, the second decision tree comprising a second set of features, the first set different from the second set; and

a classification logic implemented at least partially in hardware to determine whether the file has malware based on the first decision result and the second decision result.

19. The computing device of claim 17, wherein the categorization logic is further operable to:

identify a relationship between each of the one or more feature of the file; and

identify one or more categories for each of the one or more features based on the relationship between each feature.

20. The computing device of claim 19, wherein identifying one or more categories for the features based on the relationship between each feature comprises categorizing related features into a same category to define a category of related features.

21. The computing device of claim 19 or 20, wherein identifying one or more categories for the features based on the relationship between each feature comprises categorizing unrelated features into the same category to define a category of unrelated features.

22. The computing device of claim 17, wherein categorizing each of the plurality of features to define a plurality of categories of features comprises associating for each feature of the plurality of features of the file with one or more category.

23. The computing device of claim 17, wherein the tree generator logic is further operable to augment the first decision tree with a feature that is not related to a feature in the first set of features.

24. A computing device comprising:
an extraction means for extracting one or more features from a file;
a categorization means for associating each of the one or more features with one or more categories; and
a tree generator means for generating a decision tree based on the one or more categories.

25. The computing device of claim 24, further comprising:
an execution means for:
 executing a first decision tree to generate a first decision result, the first decision tree comprising a first set of features, and
 executing the second decision tree to generate a second decision result, the second decision tree comprising a second set of features, the first set different from the second set; and
a compilation means for determining whether the file has malware based on the first decision result and the second decision result.

26. The computing device of any of claims 24 or 25, further comprising categorization means for:
 identifying a relationship between each of the one or more feature of the file; and
 identifying one or more categories for each of the one or more features based on the relationship between each feature.

27. The computing device of any of claims 24 or 25 or 26, wherein identifying one or more categories for the features based on the relationship between each feature comprises categorizing related features into a same category to define a category of related features.

28. The computing device of any of claims 24 or 25, wherein identifying one or more categories for the features based on the relationship between each feature comprises categorizing unrelated features into the same category to define a category of unrelated features.

29. The computing device of claim 24, wherein categorizing each of the plurality of features to define a plurality of categories of features comprises associating for each feature of the plurality of features of the file with one or more category.

30. The computing device of claim 24, the instructions further operable to augment the first decision tree with a feature that is not related to a feature in the first set of features.

100

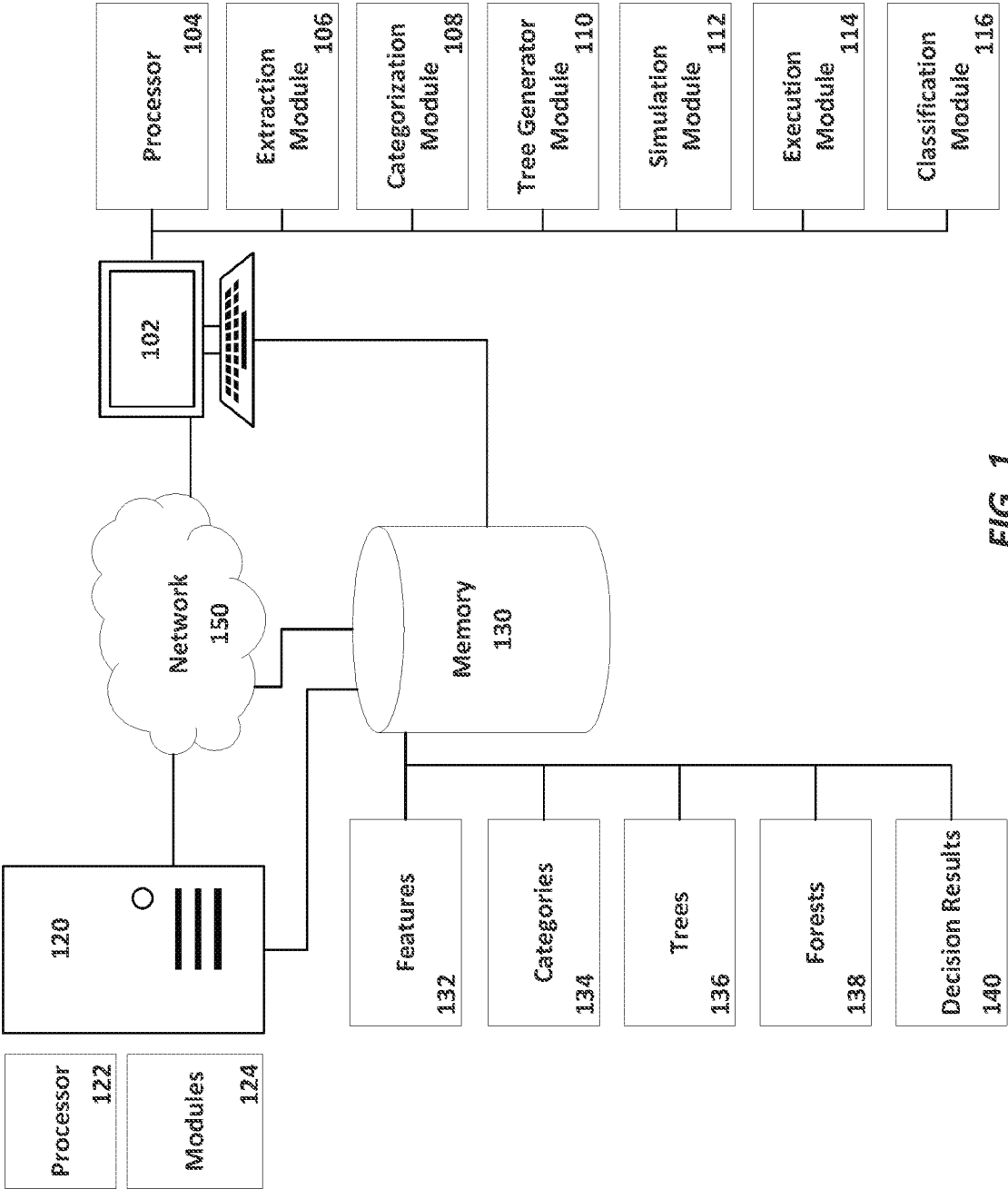


FIG. 1

200

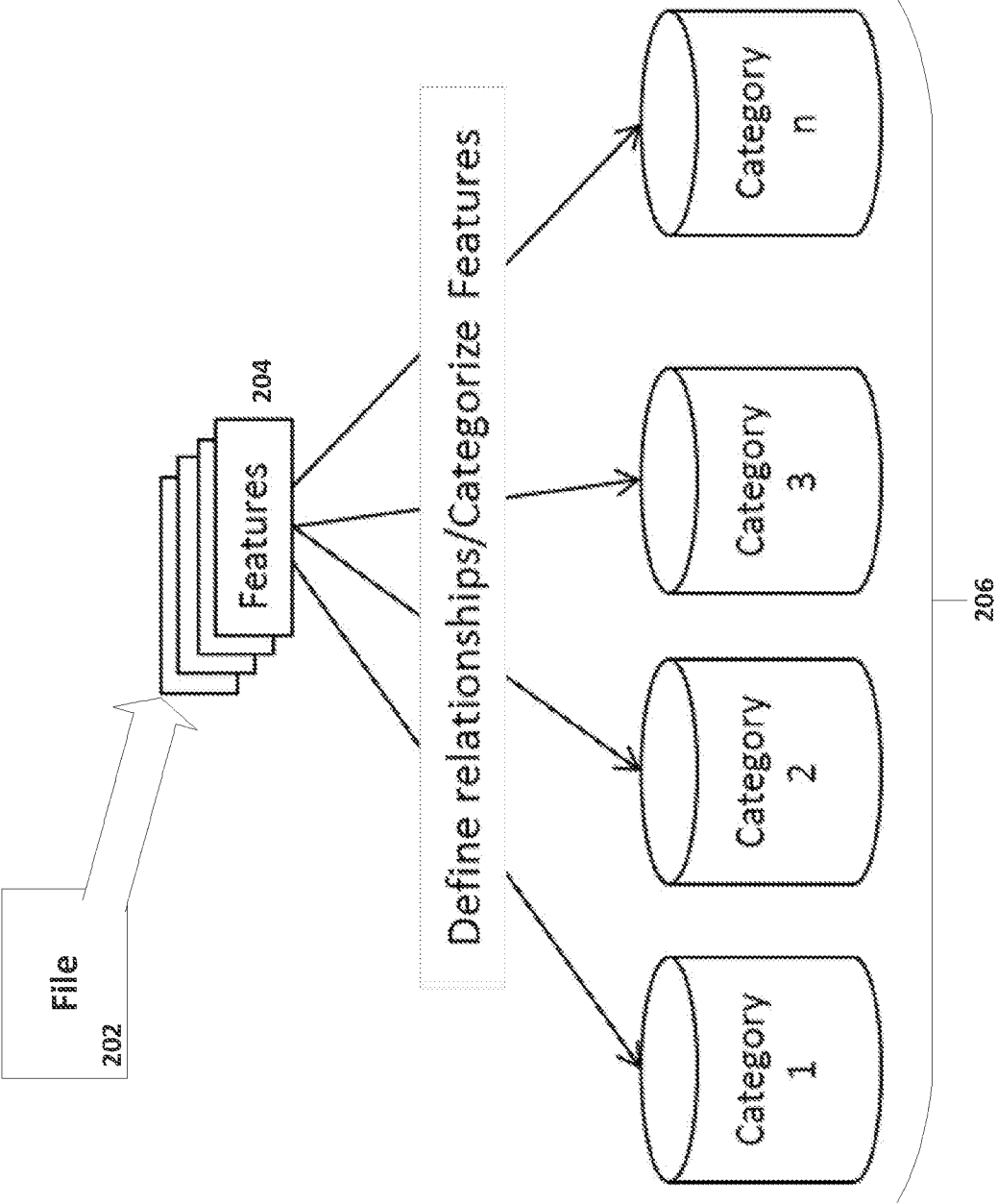


FIG. 2

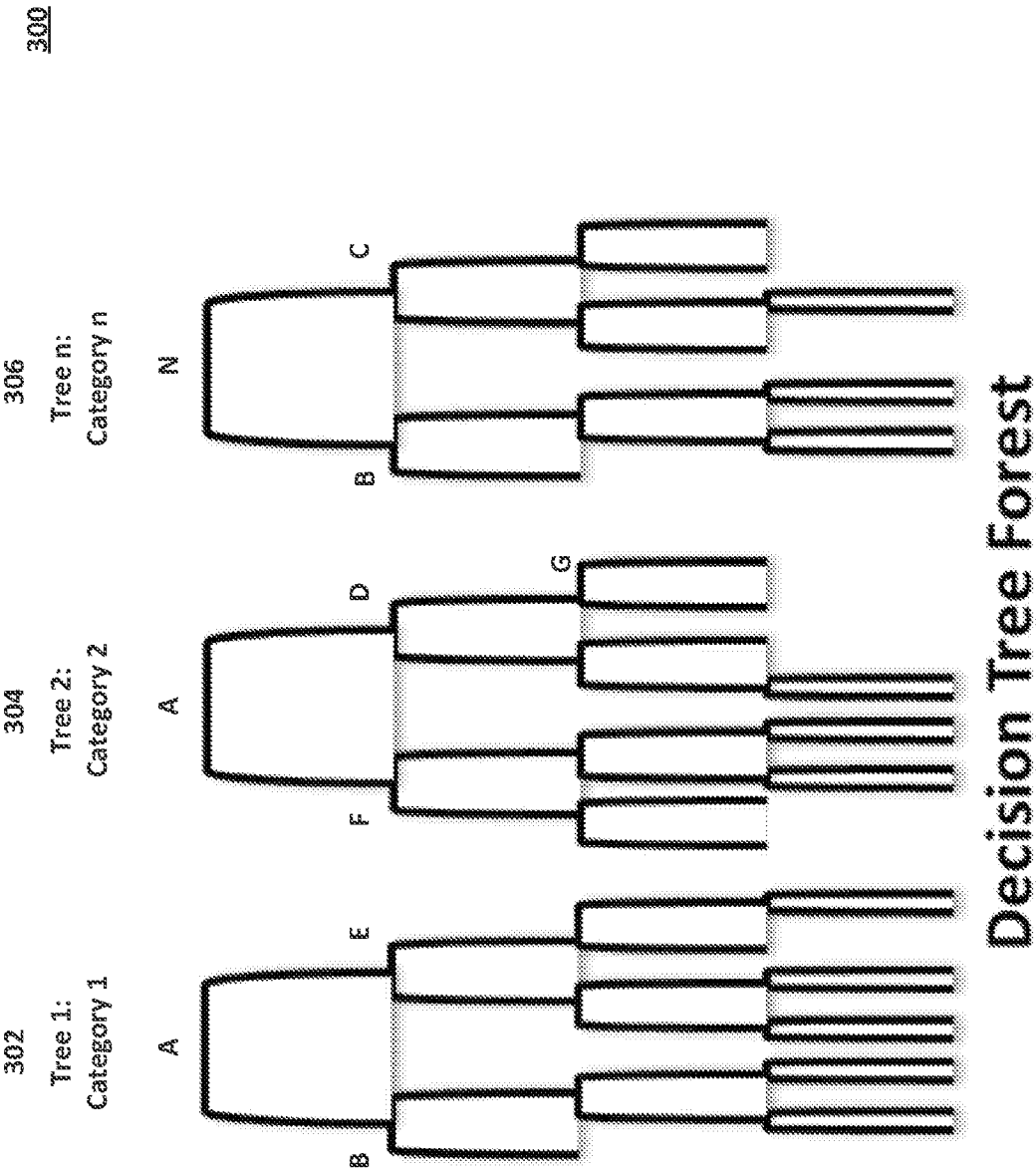
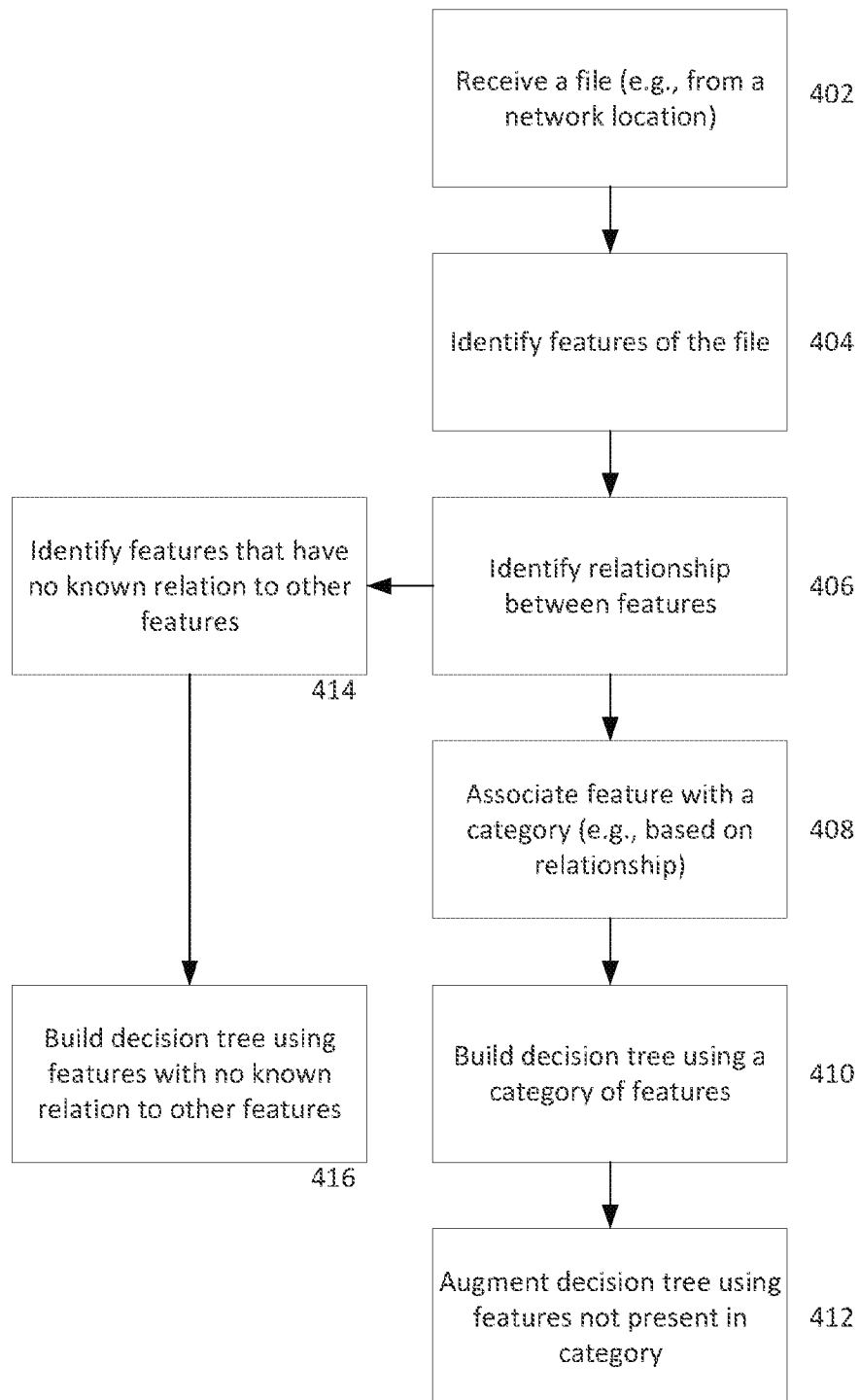
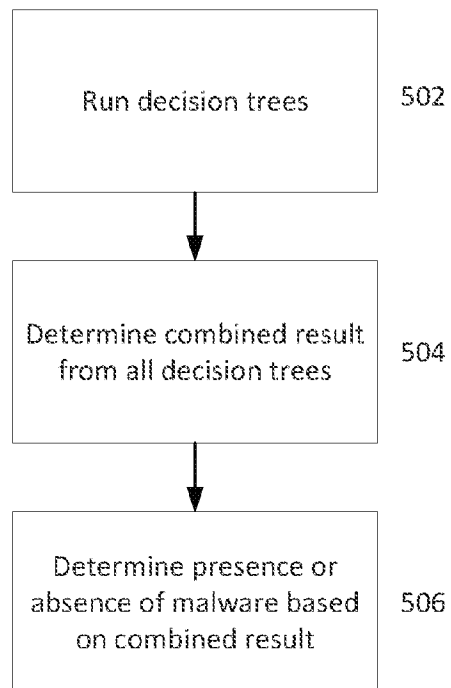


FIG. 3

4/7

**FIG. 4**

5/7

**FIG. 5**

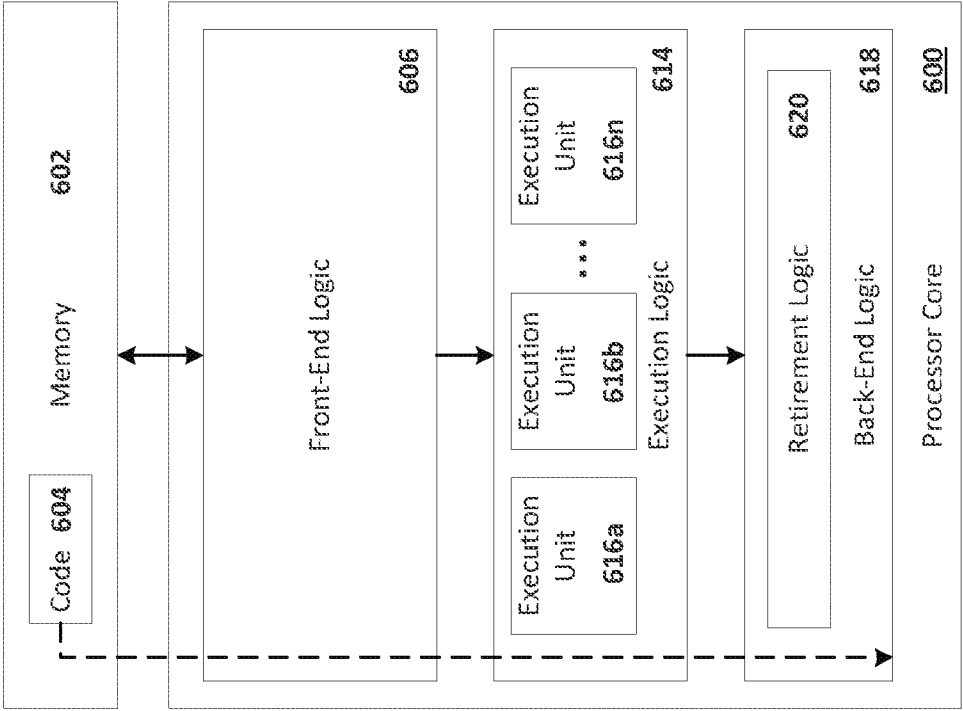


FIG. 6

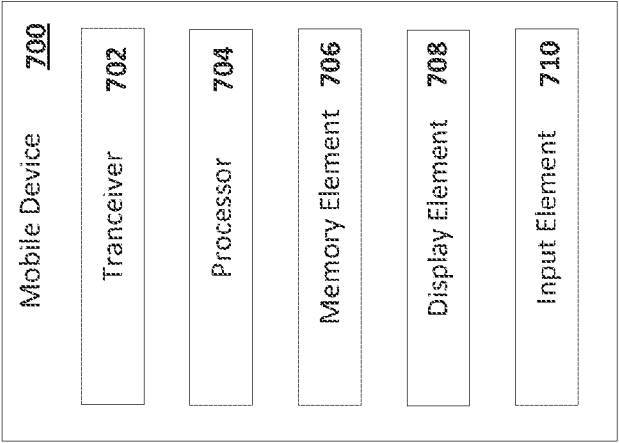
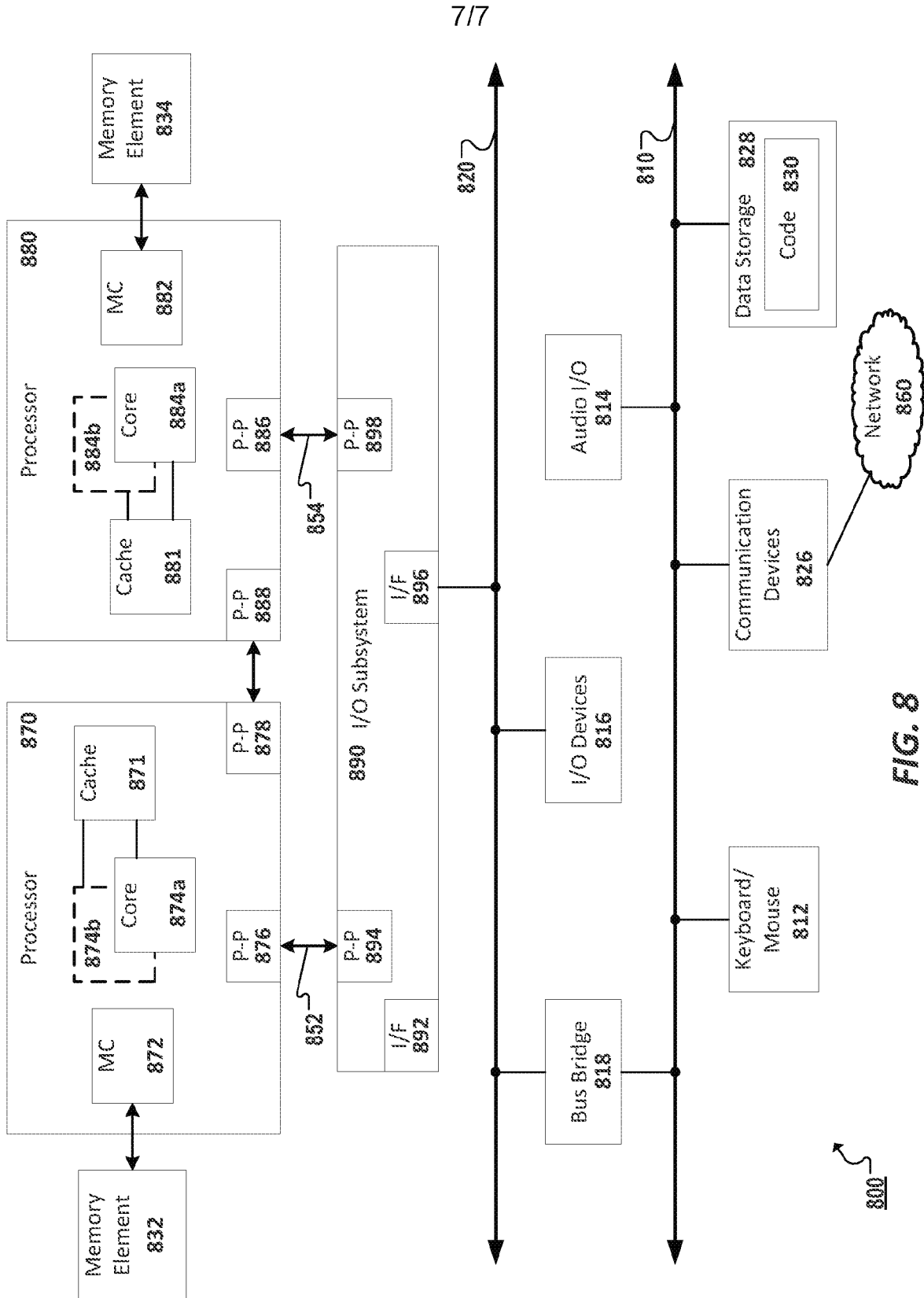


FIG. 7



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/052392**A. CLASSIFICATION OF SUBJECT MATTER****G06F 21/56(2013.01)i, G06F 21/60(2013.01)i, G06F 17/22(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)

G06F 21/56; G06F 9/445; G06N 5/04; G06F 9/45; G06F 15/18; G06F 21/00; G06F 21/60; G06F 17/22

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: decision tree, categorization, relationship, features of a file, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 7,873,947 B1 (ARUN LAKHOTIA et al.) 18 January 2011 See column 7, lines 18-22; column 8, line 59 - column 11, line 9; claim 1; and figures 2 and 4.	1-30
Y	US 2014-0090061 A1 (NORTHROP GRUMMAN SYSTEMS CORPORATION) 27 March 2014 See paragraphs [0040], [0042], and [0049]-[0052].	1-30
A	MOHD NAJWADI YUSOFF et al., 'Optimizing Decision Tree in Malware Classification System by using Genetic Algorithm', International Journal on New Computer Architectures and Their Applications (IJNCAA), Vol. 1, No. 3, pp. 694-713, The Society of iDigital Information and Wireless Communications, October 2011 (ISSN: 2220-9085) See pages 704-705; and figures 4, 7.	1-30
A	US 2012-0323829 A1 (JACK W. STOKES et al.) 20 December 2012 See paragraphs [0067]-[0071]; claim 1; and figures 7-8.	1-30
A	US 8,528,087 B2 (STEPHEN DAO HUI HSU et al.) 03 September 2013 See column 11, line 54 - column 12, line 22; and claim 1.	1-30



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

28 December 2016 (28.12.2016)

Date of mailing of the international search report

28 December 2016 (28.12.2016)

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

NHO, Ji Myong

Telephone No. +82-42-481-8528



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/052392

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 7873947 B1	18/01/2011	None	
US 2014-0090061 A1	27/03/2014	US 2016-203318 A1 US 9292688 B2	14/07/2016 22/03/2016
US 2012-0323829 A1	20/12/2012	US 8799190 B2	05/08/2014
US 8528087 B2	03/09/2013	US 2009-077664 A1	19/03/2009