



- (51) **International Patent Classification:**
G06F 7/00 (2006.01) *G06Q 50/18* (2012.01)
- (21) **International Application Number:**
PCT/AU2015/000424
- (22) **International Filing Date:**
21 July 2015 (21.07.2015)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
2014905255 23 December 2014 (23.12.2014) AU
- (71) **Applicant:** UNIVERSITY OF SOUTH AUSTRALIA
[AU/AU]; North Terrace, Adelaide, S.A. 5000 (AU).
- (72) **Inventors:** MARTINI, Ben; North Terrace, Adelaide, S.A. 5000 (AU). CHOO, Kim, Kwang; North Terrace, Adelaide, S.A. 5000 (AU).
- (74) **Agent:** MADDERNS PATENT & TRADE MARK ATTORNEYS; Level 4, 19 Gouger Street, Adelaide, South Australia 5000 (AU).

(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).

Published:

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

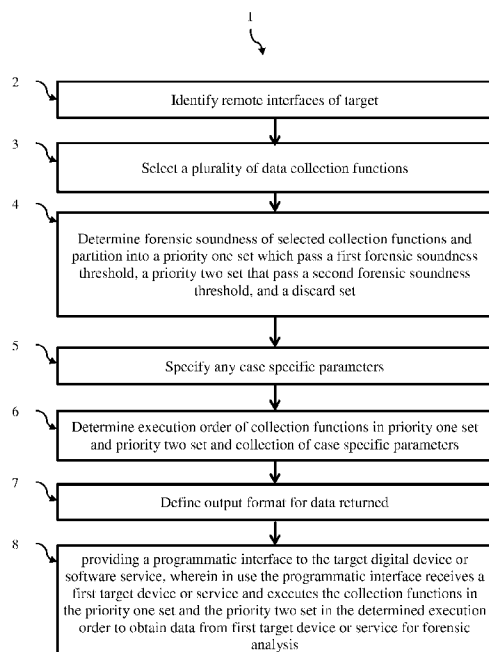
(54) Title: REMOTE PROGRAMMATIC FORENSIC DATA COLLECTION METHOD AND SYSTEM

Figure 1

(57) **Abstract:** A method for collection of digital forensic evidence from a target entity such as a computing device or software service such as a cloud service is described. The method includes identifying one or more remote interfaces for a target digital device or software service and then selecting a plurality of data collection functions provided by the one or more remote interfaces. The forensic soundness of the plurality of collection functions is determined and collection functions are partitioned into a priority one set which pass a first forensic soundness threshold, a priority two set that pass a second forensic soundness threshold, and a discard set. The execution order of the collection functions is determined based on the priority one set and the priority two set. A programmatic interface to the target digital device or software service is then developed such that in use the programmatic interface will execute the collection functions in the priority one set and the priority two set in the determined execution order to obtain data from first target device or service for forensic analysis. Programmatic interfaces can be generated for a range of digital devices or software service to generate a library of programmatic interfaces that can then be used by a wider less skilled user base to collect digital forensic data.

REMOTE PROGRAMMATIC FORENSIC DATA COLLECTION METHOD AND SYSTEM**PRIORITY DOCUMENTS**

[0001] The present application claims priority from Australian Provisional Patent Application No. 2014905255 titled “REMOTE PROGRAMMATIC FORENSIC DATA COLLECTION METHOD AND SYSTEM” and filed on 23 December 2014, the content of which is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The present invention relates to digital forensic analysis.

BACKGROUND

[0003] Digital forensic analysis is the process of identification, preservation, analysis, and presenting digital evidence in a manner that is legally acceptable, for example as defined by Rodney McKemmish in McKemmish, R 1999, 'What is Forensic Computing?', Trends & Issues in Crime and Criminal Justice, vol. 118, pp. 1 - 6. With the growing use of digital devices such as computers and mobile phones, there has been a growing demand for digital forensic investigations and analysis by digital forensic laboratories within Law Enforcement and other investigative agencies. There are guidelines and recommended practices to which practitioners abide to ensure the data and information they examine is done in a legally acceptable manner. One of these practices is the process of undertaking an examination on a forensic copy of media, rather than the original. For example in the case of storage media such as computer hard drives or USB storage media, it is typical to seize or gain authorised access to the device (for example under a Court order) and then make a full bit-for-bit forensic copy of exhibit media. A forensic investigator can then perform further analysis of the forensic copy without tainting the original data.

[0004] More recently there has been a significant increase in the use of new technologies such as cloud storage and cloud service environments, as well as increasing networking of non-traditional devices known as the internet of things (IoT). In these new technological environments it is simply not possible or practical to seize the equipment or media as in many cases the servers or equipment are located in another country and out of the jurisdiction a law enforcement agency (LEA) making it extremely difficult to physically access, let alone seize a device. Without direct physical access to the device, and with the complexity and distributed nature of the digital environment it is not clear what is the most forensically sound approach to collect data.

[0005] There is thus a need to develop new methods and systems for collection of forensic data from cloud environments and remote devices.

SUMMARY

[0006] According to a first aspect of the present invention, there is provided a method for collection of digital forensic evidence from a target entity, the method comprising:

- identifying one or more remote interfaces for a target digital device or software service;
- selecting a plurality of data collection functions provided by the one or more remote interfaces;
- determining the forensic soundness of the plurality of collection functions and partitioning the plurality of collection functions into a priority one set which pass a first forensic soundness threshold, a priority two set that pass a second forensic soundness threshold, and a discard set;
- determining the execution order of the collection functions in the priority one set and the priority two set; and
- providing a programmatic interface to the target digital device or software service, wherein in use the programmatic interface receives a first target device or service and executes the collection functions in the priority one set and the priority two set in the determined execution order to obtain data from first target device or service for forensic analysis.

[0007] In one form, the target entity is a hardware digital device, and identifying one or more remote interfaces comprises identifying one of a local physical (e.g. USB) or remote (e.g. TCP/IP) interface and a corresponding software agent for interfacing with the digital device.

[0008] In one form, identifying one or more remote interfaces further comprises selecting the most feasible remote interface. In a further form, selecting the most feasible remote interface is based at least upon whether authentication is feasible on a remote interface, and if authentication is not feasible then selecting another remote interface from the one or more remote interfaces. In a further form selecting the most feasible remote interface is based upon one or more of the capabilities of the set of available collection functions, the speed of the interface and interference.

[0009] In one form, selecting a plurality of data collection functions comprises

- explicating the available collection functions via the remote interface;
- selecting a set of collection functions from the explicated collection functions based upon one or more constraints or criteria; and
- ordering the selected set of collection functions.

[0010] In a further form, determining the forensic soundness of the plurality of collection functions and partitioning the plurality of collection functions comprises:

Determining, for each collection function whether the collection function will result in modifications to the data source or if it will lead to a loss of data and whether the practitioner has a complete understanding of the results provided by the collection function;

Adding the collection function to a priority one set of collection functions if the collection function will not result in modifications to the data source or if it will lead to a loss of data and is understood; and

If the collection function will result in modifications to the data source, then adding the collection function to a priority two set of collection functions only if the collection function is integral otherwise discarding the collection function.

[0011] In one form, partitioning the plurality of collection functions further comprises:

Determining, for each collection function in the priority one set of collection functions, the forensic soundness, and if the forensic soundness passes a first threshold, keeping the collection function in the priority one set of collection functions, else moving the collection function to the priority two set of collection functions; and

Determining, for each collection function in the priority two set of collection functions, the forensic soundness, and if the forensic soundness passes a second threshold, keeping the collection function in the priority two set of collection functions, else discarding the collection functions.

[0012] In one form, the method further comprises specifying one or more case specific parameters, and determining the execution order comprises determining the execution order of the collection functions in the priority one set and the priority two set and the case specific parameters.

[0013] In one form, the method further comprises designing an output format for the data returned by the functions collection functions in the priority one set and the priority two set to facilitate further analysis of the collected data.

[0014] In one form, the method is repeated for a plurality of target digital device or software service to generate a library of programmatic interfaces.

[0015] According to a second aspect of the present invention, there is provided a computer readable medium comprising instructions for causing a computer to perform the method of the first aspect.

[0016] According to a third aspect of the present invention, there is provided a computing apparatus comprising a communications interface, a memory and a processor, wherein the processor is configured to perform the method of the first aspect.

BRIEF DESCRIPTION OF DRAWINGS

[0017] Embodiments of the present invention will be discussed with reference to the accompanying drawings wherein:

[0018] Figure 1 is a flowchart of a method for collection of digital forensic evidence from a target digital device or software service according to an embodiment;

[0019] Figure 2A is a flowchart of the step of identifying remote interfaces of target in Figure 1 according to an embodiment;

[0020] Figure 2B is a flowchart of the step of selecting the most feasible remote interfaces and selecting associated data collection functions in Figure 1 according to an embodiment;

[0021] Figure 2C is a first partial flowchart of the step of determining forensic soundness of selected collection functions and partitioning collection functions in Figure 1 according to an embodiment;

[0022] Figure 2D is a second partial flowchart of the step of determining forensic soundness of selected collection functions and partitioning collection functions in Figure 1 according to an embodiment;

[0023] Figure 2E is a third partial flowchart of the step of determining forensic soundness of selected collection functions and partitioning collection functions in Figure 1 according to an embodiment;

[0024] Figure 3 is a flowchart illustrating execution of a method for collection of digital forensic evidence from a target digital device or software service according to an embodiment; and

[0025] Figure 4 is a schematic diagram of a computer apparatus according to an embodiment.

[0026] In the following description, like reference characters designate like or corresponding parts throughout the figures.

DESCRIPTION OF EMBODIMENTS

[0027] The present invention provides an improved method and system for collection of forensically sound data from a range of digital devices and environments. The methodology broadly comprises the analysis of the forensic soundness of public interfaces for the digital device in order to select and prioritise the implementation of data collection routines to extract data in a forensically sound manner.

[0028] One definition of forensic soundness was proposed by Rodney McKemmish in McKemmish, R 2008, 'When is Digital Evidence Forensically Sound?', in Ray, I & Sheno, S (eds), Advances in Digital Forensics IV, vol. 285, Springer US, pp. 3-15, in which he proposed a definition of forensically sound as *"[t]he application of a transparent digital forensic process that preserves the original meaning of the data for production in a court of law"*. In the same publication he further specified forensic soundness using four criteria for determining whether a process is forensically sound. The criteria are as follows:

1. Meaning – "Has the meaning and, therefore, the interpretation of the electronic evidence been unaffected by the digital forensic process?" (McKemmish 2008, p. 11).
2. Errors – "Have all errors been reasonably identified and satisfactorily explained so as to remove any doubt over the reliability of the evidence?" (McKemmish 2008, p. 12).
3. Transparency – "Is the digital forensic process capable of being independently examined and verified in its entirety?" (McKemmish 2008, p. 12).
4. Experience – "Has the digital forensic analysis been undertaken by an individual with sufficient and relevant experience?" (McKemmish 2008, p. 13).

[0029] Referring now to Figure 1, there is shown a flowchart 1 of a method for collection of digital forensic evidence from a target entity according to an embodiment. In the context of this specification target entity is defined to be digital device, apparatus or computing hardware, as well as software services and software interfaces including cloud services, virtual machines, virtual data centres, software as a service, infrastructure as a service etc. Further in the context of this specification device is defined as a hardware apparatus that may be a unitary apparatus, or a distributed apparatus in which the components or modules communicate over a wired or wireless interface

[0030] The method begins with identifying one or more remote interfaces for a target entity such as a digital device or software service (2). This step is further illustrated in Figure 2A which is a flowchart 2 of the step of identifying remote interfaces of target in Figure 1 according to an embodiment. The first test involves determining if the target is a software environment (20). If it is (Y) then the method can proceed directly to identifying remote interfaces such as a ReprEsentational State Transfer (REST) web service, Simple Object Access Protocol (SOAP) or any other type of application programming interface (API) or service interface (e.g. HTTP). However if the target is a hardware target then a second test is performed to identify a local physical (e.g. USB) or remote (TCP/IP) interface and if there is a corresponding software agent for interfacing with the digital device exists 22. If no software agent exists, then the practitioner would first establish if it was possible to communicate with the device via lower level interfaces (e.g. JTAG). If such interfaces exist, then a software agent can be developed to exploit this. If development of a software agent is not feasible or there are no available lower level protocols then the method terminates 24. However if a software agent exists, then it can be used as the remote interface 28. In the context of the specification a software agent comprises a software program, module or interface such as a web interface

implementing get and set subroutines, or terminal emulation software (e.g. Telnet) that sends key strokes to the device and returns results.

[0031] For a networked software service the first step would be to review vendor documentation and/or research publications on the product to establish which interfaces exist. If the entity is particularly esoteric or it is expected that there are other interfaces which are not documented, a local copy of the product could be obtained and analysed, or if not possible a remote interrogation of the product may be performed such as via a port scan.

[0032] For a physical device the first step would be to analyse the stock software and user accessible connections (e.g. USB, Wi-Fi, etc.) to determine if an appropriate interface is available for our collection. If the primary user interfaces are deemed unsuitable (e.g. not forensically sound or not possible to recover all available data) then other available interfaces will be considered. The interfaces are generally those used for debugging, engineering or during product manufacture such as UART, JTAG and custom interfaces setup via the USB connection.

[0033] If pertinent and feasible then a 'man-in-the-middle' approach may be used between the target entity from which data is being collected and another entity that the target entity is communicating with. This provides a copy of the inbound and outbound communication to the device. While this is commonly undertaken at the network layer it can be achieved at lower physical connection layers also.

[0034] The next step is to select a plurality of data collection functions (3), which we will also refer to as the set of collection functions. These may be from a single remote interface, or from multiple remote interfaces. A further step is to select the most feasible remote interface and obtain a set of data collection functions provided by the selected most feasible remote interface. Figure 2B is a flowchart 3 of the step of selecting the most feasible remote interfaces and selecting associated data collection functions in Figure 1 according to an embodiment. We begin by selecting the most feasible remote interface in the list of one or more identified remote interfaces 30. A first test is whether authentication is feasible on this interface 31. If not then we discard the interface and select the next most feasible interface 32. In other embodiments, determining feasibility is based upon one or more of the capabilities of the set of available functions, the speed of the interface and interference. The capabilities and extent of the set of available functions and thus feasibility can be ascertained by explication and/or practitioner knowledge and experience. For example if the available functions cannot provide log files then they may not be feasible for a forensic investigation. Similarly if the speed or throughput of the interface is very slow (e.g. 1200 or 2400 baud) then it may not be feasible to use. Similarly if the interface is susceptible to environmental or malicious interference then it may not be feasible. However if authentication is feasible then we explicate the available collection functions via the selected most feasible remote interface 33. In general feasibility relates to a range of factors including technical feasibility and forensic soundness. The most feasible

interfaces are those that do not create forensic soundness issues (i.e. provides reliable, error free, meaningful data). A further feasibility consideration is having enough usable data collection functions to make the collection feasible. Whilst using a single interface is preferable collection functions from multiple interfaces may be used to ensure an adequate set of collection functions are available (i.e. to ensure sufficient data is obtained). With reference to logging as an example, one reason logging may be required on an interface is as a measure of forensic soundness (i.e. a method to demonstrate to the court which methods were used and, in turn, that they were forensically sound). Speed as a feasibility factor is case specific – in some cases it may be integral but is less relevant in other cases.

[0035] We then select a set of collection functions from the explicated collection functions based upon one or more constraints or criteria 34. This may be based on situational constraints, such as available time, requirements of other users (in the case of cloud environments), expiration of software leases, quantity of data, data extraction rate or efficiency, etc. Depending on the interface this information may be provided on connection, or it may be queryable after connection. Alternatively or additionally it may be derived from documentation or by reverse engineering approaches. More generally assessment of feasibility and set of collection functions to be used will often be interrelated, and similar considerations such as authentication feasibility, breadth, speed, and potential for interference apply.

[0036] Selection of a collection function, or the set of collection functions, or in some case the remote interface may be based on forensic soundness and in particular the first two criteria identified by McKemmish relating to meaning and errors. Criteria three and four are non-technical and related more to the overall methodology i.e. is the proposed method of collection for a specific device transparent and has it been developed by a suitably experienced practitioner (i.e. so it can be used by less experienced practitioners). Meaning and Error criteria can both be reasonably addressed by ensuring that the methods selected do not modify the evidence source and that the copy of the evidence which is remotely received is a true (unmodified and free of errors) duplicate of the source. Where modification is absolutely necessary it must be discrete, justifiable and clearly evident in reporting.

[0037] Another consideration is completeness or knowledge of the operation and output of a function. For example, when a function returns a dataset (e.g. a copy of all files on the device), it must be known or ascertainable (for example in an error or information message or code) that that dataset is complete, or if not, that the reason for incompleteness is known or provided. For example a filter may have been applied (e.g. to return only the image files, but importantly all of the image files). In another case a function may return the rows in a table, in which case it is important to know if the function returns all rows in a table returned, or just the first X rows. In cases where filters are applied it is important that these are later documented during reporting so that an accurate description of the operation of the function and the data returned is provided to a user. Similarly if incomplete data is returned (e.g. some files or rows are missed), then if this function is used, such limitations must also be reported to the user.

[0038] Similarly when selecting a function, the true meaning of individual data items returned must be ascertainable or understood. For example in the case cloud VM metadata, what is an "interface address" - this can be difficult to ascertain as vendor documentation can be inaccurate. Preferably experiments are conducted with known starting values being entered, so that an understanding of the transformations (if any) being applied by the software can be obtained, and ultimately an accurate determination of the expected result when collected. If our determinations are not correct then we do not completely understand the process and therefore cannot describe the 'meaning' of the data collected. Similarly timestamps are also important to forensics and as such deserve further examination. In particular it is preferable to know what timezone they are recorded in (generally local or UTC) and what the offset from real time (if any) and time zone that the remote system is working in. This may be included in data returned, or may require experimentation. In one embodiment, during a later reporting stage a description of the operation of the function and the data returned is provided for each collection function used.

[0039] The next general step is to determine forensic soundness of selected collection functions and partition into a priority one set which pass a first forensic soundness threshold, a priority two set that pass a second forensic soundness threshold, and a discard set 4. This is further illustrated in Figures 2C to 2E. Additionally the priority one set may comprise collection functions to collect volatile data (or prevent loss of volatile data before collection).

[0040] Figure 2C is a first partial flowchart 4A of the step of determining forensic soundness of selected collection functions and partitioning collection functions in Figure 1 according to an embodiment.

[0041] We begin by ordering the selected set of collection functions 40. We then start a loop and select the next collection function in the ordered list 41 until no more functions are left. We then determine whether the collection function results in modification to the data source 42. If the answer is no, then we add the collection function to a priority one set of collections functions 43, and select the next collection function in the ordered list 41. If the answer is yes, then we check if delaying the execution of the collection function will result in loss of volatile data 45. If yes then we add the function to the set of priority one functions, if the modification required by the collection function is known, discrete and of greater benefit to the collection process than the loss of the volatile data. This necessary modification must be clearly documented in the reporting and particularly highlighted for the data items that have been modified as a result of the collection function. The collection functions may also be ordered in the set of the priority one functions to minimise modification of the data source and possible loss of volatile data. If there is no loss of volatile data, then we ask if collection function that causes modification is integral (i.e. essential) 45. If no we discard the collection function 46 so as not to taint the data source, and select the next collection function in the ordered list 41. If the collection function is integral, then we add the collection function to a priority two set of collections functions 47, and select the next collection function in the ordered list 41. The method may also be varied, including the order of steps, to ensure volatile data

is collected in cases where volatile data is important to the case. Further whilst the above method has concentrated on loss of volatile data, the method may more generally consider the loss of any data including non-volatile data that is at risk of being overwritten or to which access will be lost. In another embodiment the ordering of the functions make take into account the operation of the API (i.e. an interface related constraint). That is the way an API operates may be used to choose the order. Similarly if collection functions from different interfaces are selected then the ordering may be based upon how each API functions, e.g. what data it modifies during operation, or speed of operation, etc.

[0042] Figure 2D is a second partial flowchart 4B of the step of determining forensic soundness of selected collection functions and partitioning collection functions in Figure 1 and follows from Figure 2C according to an embodiment. We first loop through the priority one set of collections functions and select next collection function in priority one set of collection functions 47. We determine the forensic soundness 48, and if the forensic soundness passes a first threshold 49 we keep the collection function in the priority one set of collection functions 51, else we move the collection function to the priority two set of collection functions 50. We then select next collection function in priority one set of collection functions 47.

[0043] Forensic soundness can be determined in several ways such as determination via experiments conducted by the practitioner or based on a review of vendor documentation of the methods to make an assessment as to their effect on existing data and the meaning of the data they return. In this case the documentation should also be reviewed to assess accuracy and completeness.

[0044] In a first embodiment forensic soundness is determined via experiments conducted by the practitioner. In one embodiment this comprises inputting some known data, understanding the transformations which occur (if any) and as such being able to determine the output which is received during collection. For example, in a cloud infrastructure as a service (IaaS) virtual machine (VM) environment there are two main types of data for collection, VM data (e.g. the virtual hard disk) and metadata. In the case of metadata, we would use the standard user interface(s) as comprehensively as possible, creating VMs and auxiliary entities such as users, organisations, groups, permissions, etc., ultimately entering a range of known values and using as many features as possible over an extended time period (i.e. simulating a standard environment). We then use our selected collection interface to retrieve all of this data and ensure that it is a precise match for that which we entered via the user interface. We would also collect further data which may be exposed via the collection interface but isn't visible in the user interface (commonly low level logging for example) and we would confirm that this collected data matches the source (it is originally stored somewhere on the volatile or non-volatile storage of the entity which we are collecting data from). In this case it is important that we understand and convey the true meaning of the data collected which is not visible as part of the standard UI. Further research (e.g. application decompilation and analysis) may be required to document the provenance of this data.

[0045] In the case of data which cannot be simply visually compared, cryptographic hashes (e.g. MD5 and SHA1) can be used to compare the resultant data files. For example, to confirm that a VM disk image is not modified as part of the forensic process or transport we would hash the disk image before we commence any forensic collection process, then undertake the collection and once complete, hash the source image and the collected image (all on a test system). All three hashes should match, if not then the source evidence file has been modified either during the collection or in transit (e.g. due to errors). The final component (hashing the file on the remote system and the collected version) would also be conducted during an actual collection as part of error detection and ensuring that the meaning of the evidence has not been modified.

[0046] These processes are relatively generic regardless of remote entity. For example, in the case of a physical device we may seek to collect a physical image of its flash storage. In that case we would hash the flash storage (e.g. block device) before the forensic process and after the forensic process and hash the received forensic image. Once again all hashes should match.

[0047] Once the priority one set of collections functions have been considered, we turn our attention to the priority two set of collection functions. Figure 2E is a third partial flowchart 4C of the step of determining forensic soundness of selected collection functions and partitioning collection functions in Figure 1 following on from Figure 2D according to an embodiment. We now loop through the priority two set of collections functions and select next collection function in priority two set of collection functions 52. We determine the forensic soundness 53, and if the forensic soundness passes a second threshold 54 we keep the collection function in the priority two set of collection functions 56, else we discard the collection function 55. We then select next collection function in priority one set of collection functions 52.

[0048] In some embodiments, the method further comprises specifying one or more case specific parameters 5, and determining the execution order of the collection functions in the priority one set and the priority two set and the case specific parameters 6. These case specific parameters are any functions (or sub-functions) which are not of general interest but may be useful in specific instances

[0049] For example, in the case of a VM, if we are investigating computer misuse (e.g. hacking) we may collect further data on the network interfaces (e.g. when they were setup, by which user, what unique identifiers they used, their assigned IP details, when these details were assigned, etc.) which would not be necessary when investigating other crimes. The end user might also focus the collection on particular components at this stage, for example in the case of a cloud, whether to collect the data stored by the entire cloud (comprehensive but time consuming) or only for a particular organisation or a particular user. Generally case specific parameters will vary depending on who the investigator is and what the

organisation focuses its investigations on (e.g. state vs federal law enforcement focus on different types of crime, vs a company which might only be interested in focusing on intrusion data).

[0050] We then determine the execution order of the collection functions in the priority one set and the priority two set (6); and any case specific parameters. The software can then be written to provide a programmatic interface to the target digital device or software service. In use this programmatic interface receives specific details of a specific target device or service and executes the collection functions in the priority one set and the priority two set in the determined execution order to obtain data from first target device or service for forensic analysis.

[0051] In some embodiments, the method further comprises designing an output format for the data returned by the functions collection functions in the priority one set and the priority two set to facilitate further analysis and/or reporting of the collected data 7. This may include specifying container formats and data types.

[0052] The format may be constrained by the interface (e.g. an interface may have a most appropriate format for data export), however we may transform the data (e.g. REST data stored as application variables and ultimately output in plain text CSV). The output may also be rich e.g. a formatted report such as a PDF. Data originally stored in a binary format for computer use (e.g. a Virtual Machine Disk file – VMDK) would often be retrieved in that format and examined and analysed by the practitioner on their client machine using other software (e.g. Guidance Software's EnCase). However, as the ultimate aim of forensics is to produce evidence for court, then if the remote system is capable of better translating the data to evidence then this translation may be executed remotely (e.g. remote collection of a key-value pair database in XML format is much more straightforward than attempting to rebuild the database from its binary log on the investigators machine). In the case of multiple options for export (e.g. collecting a machine image or individual files) the practitioner would make the appropriate selection at this point, depending on their ultimate use for this data, in terms of further analysis or reporting of the collected data.

[0053] At step 8, a programmatic interface to the target digital device or software service is provided. This may be as a computer program with a user interface, or as a library function or functions or in some other electronically executable form. In use the programmatic interface receives a first target device or service and executes the collection functions in the priority one set and the priority two set in the determined execution order to obtain data from first target device or service for forensic analysis. In one embodiment this is provided according to the format selected at step 7. Use of the programmatic interface is further illustrated in Figure 3. Figure 3 is a flowchart illustrating execution of a method for collection of digital forensic evidence from a target digital device or software service 8 according to an embodiment. This comprises selecting an appropriate collection specification 80 for the product or technology to be investigated. The user then selects the relevant remote target 81, such as the specific device, or virtual

machine in a cloud service. Case specific parameters are then configured 82. Note that parameters may differ between events for the same device due to the nature of the individual collection requirements. Execution of the collection specification on the target is then performed 83 and returned results are recorded or stored for further analysis 84. These inputs can be provided through an appropriately configured user interface, or configuration file, and does not require the user to have in-depth knowledge of the device or service, or in particular the most forensically sound access methods. This allows a larger population of staff to undertake the collection process.

[0054] In one form, the method is described above is repeated for a plurality of target digital device or software service to generate a library of programmatic interfaces, and a user interface is provided to allow a user to specify which programmatic interface to use for the first target device or service. This provides a mechanism to build up a library of approved procedures for use by operators. The method can be used to obtain a data collection interface or specification for each type of digital device or service. This approach transfers the primary responsibility for experience from the person conducting the collection to the person designing the collection process. This allows a process to be designed once, by a specialist, per application or organisation or device, and allows a larger population of staff to undertake the collection. That is the system can be configured with a user interface that allows a user (eg non specialist or less qualified staff) to simply select the target entity (ie digital device or software service) they wish to extract forensic data from. The system then determines the specific type of the target entity and then looks up the programmatic interface designed for that specific type in the library of programmatic interfaces, and proceeds to execute the interface. Determination of the specific type can be performed by a query function or it may be specified by the user (if known). The query function may send an identification request to the target entity, or send a request that will generate a response containing the type such as a HTTP query to the device from which the HTTP header will list the server type. Alternatively another device with knowledge of the type of target entity (eg network controller) could be used, or the query function may send a series of requests and based on the response determine the type of the target entity. The design specification can be subject to a review process and independently verified. This allows library entries for a range of target entities to be created by a range of experts and each of which can be peer reviewed and then distributed to a wider community of users.

[0055] An embodiment will now be described in the context of collecting forensic evidence from a cloud environment. Cloud forensics is an emerging and relatively under-researched area compared with traditional digital forensics such as hard-drive forensics. One obvious reason for this lack of research is the difficulties for law enforcement agencies (LEAs) to physically access servers for analysis as for the majority of LEAs worldwide, these servers would be outside of their jurisdiction. In this embodiment the cloud service analysed is VMware Inc's vCloud™ which is a popular cloud service.

[0056] The process begins with analysing the vCloud environment which comprises a number of software defined logical components in order to explicate the potential evidence sources and determine the available, and then the most feasible remote interface for collection. This stage assists practitioners in understanding the capabilities and extent of the set of functions that may be available from one or more interfaces. From a forensic perspective, these components need to be well understood to ensure that identification of all potential evidence is undertaken and the meaning of the individual data items is known. We thus commence this stage by undertaking identification of the logical architecture.

[0057] The vCloud environment consists of a number of logical components categorized into three major groupings, namely: Organizations, cloud resources and vSphere resources.

[0058] Customers and their associated virtual infrastructure are represented in the vCloud environment as Organizations (in this embodiment, Organization with a capital 'O' represents the vCloud object). Organizations can be considered individual (software isolated) Infrastructure as a Service (IaaS) clouds which contain vApps, virtual machines (VMs), Organization members, Organization Virtual Datacenters (VDCs) and Catalogs. Organizations are likely to be the focus for the majority of forensic investigations in the vCloud environment as most forensic investigations in the cloud environment are likely to focus upon an individual customer of a cloud service provider (CSP) rather than the entire cloud environment hosted by the CSP.

[0059] vApps are collections of VMs (and supporting services such as internal networks) designed to cooperate to provide a particular service (e.g. one VM to host a webserver and another to host a database working together to provide a particular web application). In most circumstances, vApps would likely be pre-provisioned as an Open Virtualization Format (OVF) and uploaded from the client's computer or downloaded from a public source. However, vCloud also supports the building (specifying) of vApps directly from the web interface. vApps can also be provisioned from "public" and organizational catalogs which consist of vApps uploaded by organizational and vCloud administrative users. vApps are the central forensic artefact in terms of the compute and storage capabilities in the vCloud environment.

[0060] VMs can be considered child objects of vApps within an Organization. They are specified within vApps; however, they can be individually controlled and customized to an extent. As with other VMware hypervisor products, CPU, memory and network interfaces are customizable. There is also built-in customization for guests, such as local administrator password reset and domain join. In many forensic cases, a practitioner's focus will be on obtaining copies of the VMs (especially the virtual hard disk) created by a particular user or Organization hosted in a vCloud environment. This allows the practitioner to conduct a standard digital forensic analysis of the VM using techniques similar to those used on the disk image of a personal computer or other devices.

[0061] Organization members are a relatively minor component of the vCloud system. However, they are a particularly important concept for forensic practitioners especially in terms of gaining an understanding of data provenance. Within a vCloud environment, there are two types of users, namely those who manage the entire vCloud environment, known as “system administrators”, and users who manage or use an individual Organization (also known as “members” of a particular Organization).

[0062] The vCloud environment segments the various Organizations provisioned in the environment and as such members who are created within one Organization will not be visible or accessible from within another Organization. This also applies to system administrators who may have control over the entire vCloud environment but cannot logon directly to individual vCloud Organizations. It should, however, be noted that system administrators can administer Organizations via the central vCloud web administration interface. When a practitioner collects log data from a cloud environment, authenticated users and organizations can both be used to filter the logs to only the relevant events.

[0063] Organization Virtual Datacenters (which are commonly referred to as VDCs in vCloud) is a means for the vCloud environment administrator to control the amount of resources a particular Organization can access and the terms on which the resources can be accessed. The three types of VDC allow compute and storage resources to be assigned to an organization (known as an “Allocation Model”). Resources can be assigned at the time of VDC creation (with options for committing all – Reservation Pool - or only a percentage of the total resources assigned at VDC creation – Allocation Pool) or only while a vApp remains provisioned (Pay-As-You-Go).

[0064] Organization administrators can only view the VDCs which have been assigned to their Organization, their ability to modify the VDCs is minimal. However, Organization administrators must assign a VDC to a vApp when it is provisioned. With these factors in mind, it is not likely that VDCs will be of particular interest to a forensic practitioner in most general forensic cases. However, if a practitioner is interested in the amount of compute and storage resources available to the Organization being investigated, the Organization VDCs can provide guidance in this area.

[0065] Catalogs are listings of published vApps, which can be downloaded from the catalog and added to an Organization. The two major types of catalogs are Organization catalogs and public catalogs. vApps listed in the former catalog are only accessible by users in the same Organization while vApps listed in the latter catalog are accessible by every organization hosted in the vCloud environment.

[0066] The ability to publish vApps in catalogs can be limited by the vCloud environment system administrator to restrict particular Organizations to publishing only within their own catalog or only within the vCloud environment and not to external cloud environments (via URL). From a forensic perspective, catalogs can potentially be used to share VMs between Organizations and clouds. Where a

specific vApp is being used as a base image for cybercrime in multiple organizations and clouds, the originating organization and member would be of interest to a practitioner.

[0067] Logs include both tasks and events. While logs can be represented system-wide, they fit best under an Organization from a forensic perspective. Tasks represent potentially long running actions on an object in the vCloud environment. For example, starting or updating a vApp are both examples of a task. Events represent actions which have occurred in the vCloud environment. For example, tasks often result in one or more events being logged. However, events which are not tasks such as a user login will be logged as events. There are a large number of different events in the vCloud environment and they are likely to change and increase with time. Events are generally more detailed than tasks (e.g. one modify vApp task we initiated resulted in five events including information on the specific vApp parameter we changed) and, as such, would likely be the focus for a forensic practitioner. Log data is a critical component of a forensic investigation in the cloud environment as the physical separation between the cloud environment and the suspect means that evidence sources other than physical proximity must be used to indicate a suspect's potential involvement or exonerate them.

[0068] Cloud resources are provisioned and administered by the vCloud environment system administrator. For a forensic practitioner to collect data on cloud resources, they will require system administrator privileges, as the underlying cloud configuration is almost entirely abstracted from Organization members. The requirement to collect data on cloud resources (and vSphere resources) will depend on the type of investigation and the level at which the investigation is to be undertaken. For example, a practitioner will need to investigate the cloud resources configuration to begin to determine which physical devices VMs are on hosted both for compute and storage.

[0069] The following elements are classified as cloud resources in vCloud: cloud cells, provider and organization VDCs, edge gateways, external networks and network pools.

[0070] Cloud cells are the instances of vCloud Director operating in the vCloud environment. The IP address of each cell is available which could be used by the practitioner as a means to physically locate cloud cells.

[0071] Provider VDCs are allocations of resource pools within vCenter instances, which are generally supported by ESXi hosted on physical hardware and storage systems on physical hardware. In essence, provider VDCs are the bridge between individual vSphere instances which support cloud VMs and the vCloud environment.

[0072] External networks, network pools and edge gateways all relate to the network configuration of the vCloud environment. External networks are used to provision IP connectivity to networks outside of the

cloud environment (most commonly the Internet). They could be useful if a practitioner is attempting to determine if an IP address is assigned to a particular organization in the vCloud environment. Network pools are a means of provisioning private networks for use in Organizations and individual vApps which are logically segmented from all other networks. As such they are unlikely to be of significant interest in most forensic cases. Edge gateways provide interfaces between internal and external networks, and can provide network services.

[0073] vSphere is a VMware virtualization management suite and is one of the major underlying components that support the VMs which are hosted in the vCloud environment. vSphere resources are primarily represented by vCenter which administers the compute, storage and network resources available to Organizations in the vCloud environment.

[0074] vSphere resources include vCenter hosts and within those vCenters, resource pools, ESXi hosts, datastores, storage policies and switches & port groups. It should be noted that vCloud does not permit any significant configuration of the vSphere/vCenter environment from the vCloud administration interface. vSphere has a separate web administration interface used to provision hosts and configure base services for the hosting of cloud VMs.

[0075] vCenter configuration information is stored to allow vCloud to connect to and manage the vCenter instances which, in turn, manage the clouds resources. These details include the vCenter name, IP addresses, ports, versions, administrative credentials, and the web client URL.

[0076] Within these vCenters, resource pools are configured and linked to the vCloud instance. vCenters also log tasks and events, which they process similarly to vCloud. However, as important metadata such as links between VMs and organizations/members is not stored in the vCenter environment (instead vCloud GUIDs – Globally Unique Identifiers - are used to identify vCenter objects), the use of this metadata is somewhat limited.

[0077] Resource pools are software defined resource containers which determine which physical resources within a vCenter can be accessed and used by the vCloud. The use of resource pools allows a vCenter/vCloud administrator to assign a subset or different subsets of the CPU/Memory resources of a particular host or cluster in the vCenter environment to the vCloud. Resource pools are a distinct concept within the vCenter environment. The vCloud environment permits more than vCenter resource pools to be assigned as vCloud resource pools. For example, a vSphere Distributed Resource Scheduler (DRS) – a means for load balancing virtual machines between VM hosts and/or datastores – cluster can be configured in vCloud as a resource pool (which would seem likely to be a common configuration).

[0078] Details for the vCenter hosts which support vCloud VMs are stored within vCloud. Commonly hosts would be running VMware's ESXi hypervisor product. Hosts need to be 'prepared' for use with vCloud before they are able to host cloud VMs. Host names, vCenters and the number of cloud VMs hosted on each host (along with other status data) is available in the vCloud hosts interface. However, it is not possible from the web client hosts interface to determine which Host is running a particular VM.

[0079] Datastores (and datastore clusters) which support the vCloud are listed as vSphere resources. Storage capacity information is available along with the number of provider VDCs which make use of the datastores. Storage policies can be configured in a vCloud environment to distinguish across the environment between the various types of storage available to organizations. For example, in a commercial public cloud, fast storage (e.g. SSD backed) may be available alongside slow storage (e.g. HDD backed) with different fees associated for each option.

[0080] Switches & Port Groups which have been configured in the vSphere environment are visible from the web client interface. The cloud network and type configured for each port group can be viewed in summary from this view. Other than the vCenter which hosts the switches & port groups, little other data of forensic interest appears in the web client interface.

[0081] Unfortunately, vCloud does not have any obvious means to preserve or freeze the state of an Organization or vApp/VM for the purposes of digital forensics. However, according to the method described herein a standard collection approach can be developed.

[0082] Case specific parameters will guide the execution order. For example one order may be used for general investigations, whilst a different order may be used in a case where a practitioner is unable to immediately proceed with the collection process for a vCloud environment and they believe there is a high risk that data will be deleted or modified before they are able to undertake a collection. For example whilst collection of metadata should be a relatively quick process, collection of data (such as VMDK files) will likely be time consuming, with the exact time depending on a number of factors including the quantity of data to be collected (e.g. number and size of the VMs), the overall speed and latency of the connection to the vCloud server and the load (e.g. of other customers) on the vCloud environment. Thus ordering of collection of metadata could be prioritized over VMDK files.

[0083] It should be noted that all the preservation techniques discussed in this embodiment require some modification of the metadata in the vCloud environment. In principle, this modification should be limited to discrete items of metadata, such as the enabled flag for an Organization member in a database. In practice, we would note that such innocuous changes may have undesirable side effects as they are not designed to maintain forensic integrity but rather to be used for environment administration purposes. In general a practitioner is guided by the principle that the change is acceptable if the changes required

outweigh the potential for modification of evidential data. It should be noted that a complete implementation of this process would further comprise conducting scientific experiments that are repeatable and reproducible to determine and document the effect of any change.

[0084] The practitioner should assess the need to disable the Organization members to prevent them from potentially modifying or deleting evidential data from the cloud Organization. Depending on the nature of the organization, it may be feasible to disable only individual users who are suspected in being involved in a crime (especially for example in the case of a cloud organization used by to run a large business). In other organizations, the most appropriate solution may be to disable all users (except the user which the investigator is using to connect to the environment). Another issue which the practitioner must address is the potential for some vApps (depending on the organization VDC they are provisioned against) to be automatically archived or deleted when their leases expire. In this case, leases should be extended for any vApps which are likely to be of interest to the practitioner. A practitioner should also consider powering off vApps of evidential interest (after collecting a memory image, if feasible). This prevents any predefined scripts from deleting data from inside the VMs. Powering off is preferred to shutting down the guest as this may also trigger predefined destructive scripts. If the VMs must be kept running, network access to the VMs may also need to be restricted to prevent users disabled in the vCloud environment from connecting to the VMs directly.

[0085] Thus in one embodiment the collection method includes a preservation method that automatically disables users if they have accessed the environment within the last seven days (although the period of activity is entirely configurable by the user as a case specific parameter). This is possible as the application briefly examines the events' data and the members registered in the focus Organization before commencement of the vApp downloads. This preservation process could also be separated into an application which is run immediately when cloud use is detected (perhaps with the assistance of the cloud service provider) but before collection commences. As we collect vApp metadata before we download the vApp it is also possible to extend any storage leases which have an imminent expiration date (preventing their possible deletion). This programmatic approach to disable users and extend leases ensures that mistakes are avoided and potential evidential data is not lost due to human error.

[0086] We now turn to the selection of the most feasible remote interface. In the case of vCloud, one method generally available to a practitioner is preservation and collection from the client web interface. This is the interface which organizations use to provision and administer the vApps and VMs which they have hosted in the vCloud environment. Another available interface is the vCloud API that is exposed via the open REST web communications standard. This API is made available for the purposes of communication with third-party applications and is also suitable for remote collection of potential evidential data from the vCloud environment.

[0087] In this case the vCloud API is selected as the most feasible interface for programmatic collection. Whilst the web interface may have ubiquitous availability, the interface does not layout metadata in a form which can be easily extracted and interpreted as electronic evidence (as it is not designed for this purpose). Further in a cloud environment there may be hundreds or even thousands of vApps and VMs to be provisioned, which would make it very difficult for a practitioner to refine the available data to only the vApps and VMs of interest. For example, in the vCloud environment, it is not possible from the organization's web interface to quickly collect the metadata from VMs in vApps owned by a particular user for reporting or download a large number of vApps without undertaking a slow and tedious multi-screen process. Using the web interface for these purposes would be very time consuming and error prone.

[0088] This significant quantity of metadata makes it difficult for a practitioner to manually identify, preserve and collect the relevant data and considerably increases the potential for evidence to be missed in a volatile environment where deferred collection may not be possible. For example, most vApps are provisioned with 'leases' (both runtime and storage) in the vCloud. After this lease time has expired, they may be stopped and archived or deleted from storage (depending on configuration). It is for this reason and the potential in a live cloud environment for the vApp owner, Organization administrator or system administrator to intentionally or inadvertently delete or modify the vApp, that early identification, preservation and collection is critical.

[0089] In contrast the vCloud API that is exposed via the open REST web communications standard allows programmatic interrogation of the entire vCloud environment, and will execute identically on all vClouds from which we are collecting evidence.

[0090] We next consider whether authentication is feasible on this remote interface. In this embodiment we assume that a forensic practitioner is focusing their evidence collection on a particular individual or organization, in which case the investigation will be conducted on a cloud Organization rather than the entire cloud environment. Therefore, this evidence collection process assumes a focus on a vCloud Organization. The software segmentation which vCloud uses to ensure that organizations cannot interfere with each other also means that most evidential data created by the suspect would be stored at the Organization level. However it will be understood that this approach could be scaled up to multiple vCloud Organizations, which may include all organisations in the cloud environment.

[0091] At the vCloud Organization level any request made via the vCloud API will require authentication. If the practitioner is focusing on an organization, these credentials can be "Organization Administrator" level credentials or "System Administrator" credentials. The former could be obtained from the customer of the cloud environment while the latter can only be obtained from the CSP and would provide the practitioner with access to the entire vCloud environment (including all other

organizations). The most likely identification point for cloud use is the client device. Therefore, the most obvious source of organization administrator credentials would be the device which the administrator uses to connect to the cloud environment. The credentials may be stored in a (browser-based) password manager or if the suspect still has the device, employing a key logger may be feasible. As vCloud sessions are conducted over HTTPS, collection of the credentials from the network is unlikely to be feasible in most circumstances (especially without raising suspicion).

[0092] Organization administrator credentials would seem to be appropriate in most situations. It should be noted that system administrator credentials are required if the practitioner is seeking to conduct an in-depth forensic analysis of the entire cloud environment. This is required as Organization administrator credentials will not permit a practitioner to request detailed information on Cloud and vSphere resources. Thus in most cases these will be obtainable in a forensic investigation and thus authentication is considered feasible with the vCloud API.

[0093] The data collection functions were then analysed for forensic soundness and an evaluation was made of whether the collection functions would modify data or whether there is a potential loss of volatile data in order to generate and order the priority sets. Further an application was built to implement the proposed collection method to collect forensic evidence from a vCloud environment using the REST API. The application was developed using the vCloud Java SDK made available by VMware. However any other suitable development language that can communicate with the vCloud API could be used. In this embodiment the collection order was:

Priority 1:

Collect available events;

Priority 2:

Collect organizational metadata;

Collect VDC metadata;

Collect vApp metadata and VM data; and

Collect VM metadata.

The priority split is conducted on both the high level stages of the collection process and the lower level functions. In this embodiment only the high level stages are represented. It should be noted that the ordering of the collection functions might be constrained by the operation of the system being collected, as it is in this example embodiment.

[0094] In terms of priority split, that would be mostly undertaken on the lower level functions that comprise the high level steps. e.g. collect vApp metadata has multiple functions to collect various types of metadata. In this embodiment collect events has been assigned to P1 and the remaining functions as P2 as many of the P2 functions will log events (access/read events, etc.) so it is appropriate to get a copy of the organisation events before we start adding to the list, as far as possible. It is notable that some

modification to the events “data” is necessary as part of the collection as our initial login is logged as an event and reading the events may be also be logged. However this is an example of a discrete, justifiable and documented modification to the evidence source. Further the P2 ordering is constrained here by the operation of the API.

[0095] Collecting available events (in this case, logs) is prioritised first as changes to the events will be unavoidable in remote cloud forensics, so modification of this evidence source should be kept to a minimum by collecting the logs first. This proceeds by obtaining a copy of all of the events available to the user (Organization administrator) which we have logged in as. This log data may allow a practitioner to rebuild a timeline of events around a suspect (e.g. a chronology of login events by a particular member). Via experimentation it has been found that use of the vCloud API for remote forensic analysis, inevitably creates additional event data that will be added to the Organization’s logs during the collection process. However, by collecting the log data at the beginning of the session (along with accurate documentation of the time the operation commenced), we can ensure that additions made by the forensic process rather than the suspect are kept to a minimum and can be easily accounted for.

[0096] The collection of event logs was implemented as follows. First, before any requests can be made of the vCloud environment via the API, a logged in client session must be created. This is achieved by instantiating a VcloudClient object, and passing it (using various functions) the host address of the vCloud Director instance, the SSL scheme and the username (in the format of username@organization) and password of an Organization administrator.

[0097] Once we have successfully connected and authenticated with the environment, the first function we call in our code to collect the events data from the vCloud environment. The most forensically sound and feasible method to access this events data was considered to be via the “QueryService”. This is a special function offered by the API which allows a developer to query various parts of the vCloud environment primarily to collect metadata information. We can access the QueryService object via the vCloud client which we have instantiated. We must pass the query service a “QueryRecordType”, this record type is defined by a set of constants and, in this case, we select “QueryRecordType.EVENT”. This returns a RecordResult which can be considered similar to a list of objects of the type “QueryResultEventRecordType”. Iterating through this list (and using the getRecords method) for each of the records, we can review each events type (which are defined using constants e.g. EventType.SESSION_LOGIN) and its associated timestamp. We can also use IF statements (low level) or server side filtering via QueryParams (predefined fields only) to limit the results returned to, for example, only particular events or events concerning particular users or time ranges. If this is done this is printed in collection documentation or reports.

[0098] In this embodiment all available events were collected and output the results to CSV on the local machine. We also simultaneously search for user logins for a particular suspect user and can highlight in the application output if the user had logged in during the last 48 hours or other pre-determined time duration (which may indicate that the user is active and further preservation steps should be undertaken).

[0099] While the QueryService may seem like an appropriate method to collect all of the potential evidence, we found that the query service cannot return all data which can be collected from the objects individually. As such, it was only used for the purposes of event collection. More generally, if the case specific parameters only require metadata then the QueryService may be more efficient collection function than iterating through all of the levels of objects as would otherwise be required.

[00100] The next stage is to collect Organizational metadata: This metadata includes basic details such as Organization name, description, quotas, records of the members in the Organization and references to the Organization VDCs. This comprises iterating through the organization(s) which the user has access to and collecting relevant organizational metadata. While most users can only exist within one Organization, this may not apply if an external authentication scheme is used or a system administrator user (in the 'system' context) is used to connect to the environment. This metadata includes organization IDs, full names and descriptions for reporting. It also includes member details for the Organization (e.g. usernames, full names, email, phone and IM). This information can be useful in reporting and presentation and the practitioner may choose to collect additional data where it is relevant to case specific parameters. For example, if the maximum number of VMs a particular user (or the organization) is permitted to provision is of interest, this data would also be collected at this time.

[00101] Using the vCloud client, we obtain a collection of references to the Organizations which the user has access to. Iterating through that list of Organization references, we can use each reference with the Organization class's getOrganizationByReference method to return an Organization object. The Organization object exposes limited information from a forensic perspective; however, it does provide a reference to the organizations "resource". This resource is an OrgType object (described in the SDK Javadoc as a representation of the "user view of a vCloud Director organization") and provides us with the ability to collect the metadata which is stored for the organization. From the available choices (all represented by "get" methods in the OrgType object), we selected to display Name, FullName and Description.

[00102] In addition to the Organization resources, the API also uses "AdminOrganization" objects. These objects represent the configuration of a particular Organization. To access the AdminOrganization (AO) for our Organization, we used AO's getAdminOrgByReference static method and we retrieved the AO reference by name using the name provided by the Organization record. This

allows us to access the configuration settings of the Organization and we used the AO to retrieve vAppLeaseSettings and to ultimately display the default StorageLease in seconds.

[00103] Using the AdminOrganization reference which we have created above, we can now get a list of user references. Iterating through the collection of references, we can use the User class's getUserByReference method to return UserType objects which allows us to access all of the stored metadata for users. We display the username and full name as well as the user email address and IM address (if stored). At the same time, a practitioner can decide if they would like to disable all or particular members within the Organization (perhaps based upon the earlier collection of login events data). It should be noted (as discussed in Section II.B) that while accounts can be disabled at this stage, there are consequences in terms of forensic soundness which requires that modifications to evidence be kept to a minimum.

[00104] The next stage is to collect VDC metadata: Using the Organization VDC references, we iterate through the VDC objects extracting relevant metadata including name, description, allocation model and capacity information, as well as references to the VDCs provisioned vApps. This comprises requesting a list of the VDCs within the Organization(s). Querying VDCs is the next logical choice as vApps and, in turn, VMs will be grouped within the VDCs which they were provisioned against. While the primary dataset we are seeking to collect from the VDCs is a list of references to vApps similarly to organizations, there is a selection of VDC metadata available. This includes VDC name, description, VDC Allocation Model and a range of metadata on allocated compute/storage/networking capacity.

[00105] Using a similar technique to iterating through the collection of organizations, we iterate through a list of VDCs that can be obtained from the Organization object which we created above and then get Vdc objects from the reference before getting VdcType objects from the Vdc getResource method. Using the VdcType object in our application, we collect and display the VDC's name, description, allocation model and compute capacity.

[00106] The next stage is to collect vApp metadata and VM data. Using the vApp references which we collected from the VDCs, we can begin to collect both the vApp metadata and copies of the OVF and Virtual Machine Disk (VMDK) files that are used to support the Organizations VMs. This vApp metadata includes name, description, compute capacity, creation date, owner, deployment and storage lease expiry dates and sharing permissions metadata which is typically of greatest interest in many forensic investigations. Assuming that a practitioner is looking to investigate a matter which has occurred on a VM, metadata (e.g. vApp owner, vApp creation date, vApp name and description) can be of particular interest. In addition, in the vCloud REST API, VM downloads are undertaken via their parent vApp reference. If a practitioner has not yet undertaken preservation, the practitioner may also wish to look at disabling the user account of users which own the vApp or have shared access to the vApp and

confirm that deployment/storage leases are not going to expire in the immediate future (and extend the leases if necessary).

[00107] Using the technique of collecting the vApp references from the VDC(s) and instantiating vApp and VAppType (VAT) objects, we can both collect a range of metadata about the vApps provisioned in the Organization and commence the download of the vApp. Using the get methods from the VAppType object, our application displays the vApp's name, description and creation date. Using VAT's getOwner function, we are able to display the username of the vApps owner, and using the reference returned by getOwner's getUser method, we display the users email address.

[00108] The vApp object has a number of get functions which return "sections". These sections correlate to the settings sections used in the main vCloud web interface. We use the vApp getLeaseSettingsSection method to return a LeaseSettingsSectionType, which allows us to get and display both the deployment and storage lease expiration dates. The vApp getControlAccess method allows us to explore and display the "sharing" settings for a vApp. Sharing allows an owner or administrator to permit other users varying levels of access (e.g. read only, read/write and full control) to their vApp. This could be useful information to a forensic practitioner in terms of understanding who had access to the VM. We iterate through a list of access settings to display the users which have access to the vApp and their level of access. We also indicate if everyone in the organization has access to the vApp and the level of access granted to everyone.

[00109] Now that we have collected information on the lease expiries for the vApps, a practitioner can decide if they need to extend the expiries to keep the VMs running or provisioned on storage beyond their current lifetimes for further forensic analysis. Similarly to disabling user accounts, modifying the lease expiry dates should only be done where absolutely necessary as it requires modification of the evidence source.

[00110] After the metadata has been collected, we commence the download of the vApps. This is achieved using the vApp object's downloadVapp function which requires a path to save the downloaded OVF file and VMDK file. Before it is possible to download a vApp, its enableDownload method must be run. Unlike the vast majority of methods which we access to collect evidence, enableDownload was a slow operation. It returns a Task object. We used the Tasks waitForTask function to have the application wait until the vApp download has been enabled. Notably, the enableDownload function will fail if the vApp is running and throw an exception which notes that the vApp will need to be stopped before the download can be completed. The OVF file which is downloaded may have its unique identifiers changed. If this is an issue for the practitioner, they should download the OVF separately using the downloadLosslessOVF function or gather unique IDs from the VM objects directly (recommended).

[00111] In our implementation experiments, we had mixed results using the downloadVapp function. We noted that the web client appears to use the VMWare client support plugin to trigger VMware's "ovftool" to download the OVF and VMDK files when a download is requested from the web interface. The proof of concept can also be programmed to output a batch or shell script with the various parameters required by ovftool provisioned to download the vApps using this method if preferred. This has the advantage of including a manifest file with each downloaded vApp which lists the SHA hash for both downloaded files. However, it appears, this hash is calculated on the client machine and cannot be used to show that the download is a bit-for-bit copy of the files on the server. Unfortunately, vCloud does not appear to support such a server side hashing function at this time.

[00112] After collecting the vApp metadata and the VMDK files from the vApps constituent VMs, we can use the VM references in the vApp to collect VM metadata. This metadata includes name, description, creation date, VM hardware (e.g. virtual CPU/memory allocation information), capacity information, IP addresses and potentially login credentials. While information regarding the physical host and storage allocations of the VMs is available via the REST API, it cannot be accessed using Organization administrator credentials as this information is abstracted away from cloud customers and is only available to system administrators. Notably, the guest customization entries may prove useful to a forensic practitioner including both the domain credentials used to add the VM to the organizations domain and the administrator password for the client. Both values can be queried from the vCloud if their guest customizations have been enabled.

[00113] Turning to the implementation, vApps return a list of VM objects which we can iterate through to access VMType objects. These VMType objects return a range of metadata. We display basic metadata (name, description, creation date, etc.) in our application and then iterate through the device disks, returning their size. This allows a practitioner to correlate the disks which the VMs are reporting with the disks which they have downloaded from the vApps as part of the forensic examination stage. From the VM object, we display the IP addresses assigned to the VM (if available) and if guest customization is enabled, we can display the admin password and the domain join account credentials (if these details have been provided). Once this data has been collected and displayed and any remaining actions have been completed, the application exits. Table 5 provides example output of the implementation of the above described collection application according to an embodiment.

TABLE 1

Example output of the implementation of a collection application according to an embodiment.

```

02:09:09 - Collection commenced at 2014\06\14
02:09:09 UTC
02:09:09 - Connecting to: vcloudhost
02:09:17 - Connected and logged in as: cloudorgadmin

Event Collection;
02:09:17 - Total available events to collect: 20
02:09:17 - Greater than 10 events - output to screen
has been suppressed. Outputting to
.\vCollect_vcloudhost_events.txt
02:09:17 - Collection complete. Total events
collected 20 matches expected 20 events.

Organization Collection:
02:09:17 - Organization 1: cloudorg
    ID: cloudorg
    Full Name: A cloud Organization
    Description: Customer#: 123456; Address: 123
    Cloud St, Cloudvillle; Phone: 555-5555
    Default vApp Storage Lease Time: 720 hours

02:09:17 - Organization 1 Users: 2 users found.
Outputting details to
.\vCollect_vcloudhost_cloudorg_users.txt

User 1:
    Username: clouduser
    Full Name: Mr. Cloud User
    Email: cloud@user.cloud
    IM Address: cloudim@user.cloud
User 2:
    Username: cloudorgadmin
    Full Name: Mr. Cloud Admin
    Email: cloud@admin.cloud
    IM Address: cloudim@admin.cloud

```

[00114] The above example illustrates the development of a collection specification for a specific cloud environment (the vCloud environment provided by VMware). This collection specification can be provided as a software library or as part of a software package with multiple collection specification for a

range of hardware and software targets. This allows the collection specification to be used by practitioners with only basic knowledge of the vCloud environment. Rather than having to be concerned with details of the vCloud environment, they need only provide specific inputs such as the remote target entity, and case specific parameters. The more technical questions of how to extract information in a forensically sound manner is handled by the collection specification which returns the results to the practitioner.

[00115] A suitable computing system for implementing the method comprises a display device, a processor and a memory and an input device. The memory may comprise instructions to cause the processor to execute a method described herein. The processor memory and display device may be included in a standard computing device, such as a desktop computer, a portable computing device such as a laptop computer, tablet or smart phone, or they may be included in a customised device or system. The computing device may be a unitary computing or programmable device, or a distributed device comprising several components operatively (or functionally) connected via wired or wireless connections. An embodiment of a computing apparatus 100 is illustrated in Figure 3 and comprises a central processing unit (CPU) 110, a memory 120, a communications module 130, a display apparatus 142, and an input device 144 such as keyboard, mouse, touch screen, etc. The CPU 110 comprises an Input/Output Interface 112, an Arithmetic and Logic Unit (ALU) 114 and a Control Unit and Program Counter element 116 which is in communication with input and output devices (eg input device 140 and display apparatus 130) through the Input/Output Interface 112. The Input/Output Interface 112 comprises a communications module 130, which may include a network interface, for communicating with an equivalent communications module in another device using a predefined communications protocol (e.g. Bluetooth, Zigbee, IEEE 802.15, IEEE 802.11, TCP/IP, UDP, etc). A graphical processing unit (GPU) may also be included. The display apparatus may comprise a flat screen display (eg LCD, LED, plasma, touch screen, etc), a projector, CRT, etc. The computing device may comprise a single CPU (core) or multiple CPU's (multiple core), or multiple processors. The computing device may use a parallel processor, a vector processor, or be a distributed computing device. The memory is operatively coupled to the processor(s) and may comprise RAM and ROM components, and may be provided within or external to the device. The memory may be used to store the operating system and additional software modules or instructions. The processor(s) may be configured to load and executed the software modules or instructions stored in the memory. In use the computing apparatus is connected to a digital device or service 150 over a network connection 152 (ie over the internet or via an intranet). In some embodiments the computing apparatus is connected directed to the device, for example over a wired connection. The method can be implemented in software using a variety of programming languages and operating system utilities such as JAVA, .NET, C++, PERL, PYTHON, etc. The library of programmatic interfaces may be container or a database or binary file such as a DLL file or similar.

[00116] The programmatic digital evidence collection method described herein can be used for a wide range of digital devices and services. These include cloud services as well as a wide range of devices, including Internet of Things devices, mobile devices, automation devices, wearable technologies, etc. For example wearable technologies such as fitness trackers, smartwatches and smartglasses are becoming increasingly prevalent. Due to their 'always on' nature and the number of sensors which are constantly recording, they may prove to be significant evidence sources in the future. Supervisory Control And Data Acquisition (SCADA) devices are another common example as they are increasingly being provided with network connectivity and interfaces to facilitate remote monitoring of industrial process equipment, HVAC systems and water and energy infrastructure.

[00117] The programmatic digital evidence collection method described herein provides numerous advantages of existing methodologies. The use of a designed collection specification ensures that evidence collection is reliable and verifiable and ensures that all of the required collateral data is collected and that the meaning of evidence isn't lost. For example when collecting files from a cloud service, collecting all of the suspects files may be integral, however collecting all (collateral) access/modify log data relating to those files assists in determining provenance and in turn the meaning of those files to the case. In particularly programmatic collection is particularly adept at error management as a correctly designed collection process will not be subject to human error and will detect and report on any errors which occur during an individual collection.

[00118] Further as all designed collection processes run identically they can be independently verified, in complete detail, before and after a collection has occurred. This verification could be undertaken by a trusted authority, the vendor of the product being collected or via case law (as is common with traditional forensic tools).

[00119] Importantly remote programmatic forensics transfers the primary responsibility for experience from the person conducting the collection to the person designing the collection process. This allows a process to be designed once, by a specialist, per application or organisation (ie per type of target entity), and these can be consolidated into a library of programmatic interfaces. This then allows a larger population of staff with less knowledge or skills than the designer of the programmatic interface to undertake the collection. That is they simply need to specify the target entity and the relevant programmatic interface for that target entity can be extracted from the library and used to obtain forensic data. Further the design specification can be subject to a review process and independently verified (ie peer reviewed) to provide confidence to users (and others such as the Courts) that they are using verified interfaces, eg best practice or at least a forensically sound interface.

[00120] Those of skill in the art would understand that information and signals may be represented using any of a variety of technologies and techniques. For example, data, instructions,

commands, information, signals, bits, symbols, and chips may be referenced throughout the above description may be represented by voltages, currents, electromagnetic waves, magnetic fields or particles, optical fields or particles, or any combination thereof.

[00121] Those of skill in the art would further appreciate that the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software or instructions, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

[00122] The steps of a method or algorithm described in connection with the embodiments disclosed herein may be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. For a hardware implementation, processing may be implemented within one or more application specific integrated circuits (ASICs), digital signal processors (DSPs), digital signal processing devices (DSPDs), programmable logic devices (PLDs), field programmable gate arrays (FPGAs), processors, controllers, micro-controllers, microprocessors, other electronic units designed to perform the functions described herein, or a combination thereof. Software modules, also known as computer programs, computer codes, or instructions, may contain a number a number of source code or object code segments or instructions, and may reside in any computer readable medium such as a RAM memory, flash memory, ROM memory, EPROM memory, registers, hard disk, a removable disk, a CD-ROM, a DVD-ROM, a Blu-ray disc, or any other form of computer readable medium. In some aspects the computer-readable media may comprise non-transitory computer-readable media (e.g., tangible media). In addition, for other aspects computer-readable media may comprise transitory computer-readable media (e.g., a signal). Combinations of the above should also be included within the scope of computer-readable media. In another aspect, the computer readable medium may be integral to the processor. The processor and the computer readable medium may reside in an ASIC or related device. The software codes may be stored in a memory unit and the processor may be configured to execute them. The memory unit may be implemented within the processor or external to the processor, in which case it can be communicatively coupled to the processor via various means as is known in the art.

[00123] Further, it should be appreciated that modules and/or other appropriate means for performing the methods and techniques described herein can be downloaded and/or otherwise obtained by computing device. For example, such a device can be coupled to a server to facilitate the transfer of

means for performing the methods described herein. Alternatively, various methods described herein can be provided via storage means (e.g., RAM, ROM, a physical storage medium such as a compact disc (CD) or floppy disk, etc.), such that a computing device can obtain the various methods upon coupling or providing the storage means to the device. Moreover, any other suitable technique for providing the methods and techniques described herein to a device can be utilized.

[00124] In one form the invention may comprise a computer program product for performing the method or operations presented herein. For example, such a computer program product may comprise a computer (or processor) readable medium having instructions stored (and/or encoded) thereon, the instructions being executable by one or more processors to perform the operations described herein. For certain aspects, the computer program product may include packaging material.

[00125] The methods disclosed herein comprise one or more steps or actions for achieving the described method. The method steps and/or actions may be interchanged with one another without departing from the scope of the claims. In other words, unless a specific order of steps or actions is specified, the order and/or use of specific steps and/or actions may be modified without departing from the scope of the claims.

[00126] As used herein, the term “determining” encompasses a wide variety of actions. For example, “determining” may include calculating, computing, processing, deriving, investigating, looking up (e.g., looking up in a table, a database or another data structure), ascertaining and the like. Also, “determining” may include receiving (e.g., receiving information), accessing (e.g., accessing data in a memory) and the like. Also, “determining” may include resolving, selecting, choosing, establishing and the like.

[00127] Throughout the specification and the claims that follow, unless the context requires otherwise, the words “comprise” and “include” and variations such as “comprising” and “including” will be understood to imply the inclusion of a stated integer or group of integers, but not the exclusion of any other integer or group of integers. The reference to any prior art in this specification is not, and should not be taken as, an acknowledgement of any form of suggestion that such prior art forms part of the common general knowledge.

[00128] It will be appreciated by those skilled in the art that the invention is not restricted in its use to the particular application described. Neither is the present invention restricted in its preferred embodiment with regard to the particular elements and/or features described or depicted herein. It will be appreciated that the invention is not limited to the embodiment or embodiments disclosed, but is capable of numerous rearrangements, modifications and substitutions without departing from the scope of the invention as set forth and defined by the following claims.

CLAIMS

1. A method for collection of digital forensic evidence from a target entity, the method comprising:
identifying one or more remote interfaces for a target digital device or software service;
selecting a plurality of data collection functions provided by the one or more remote interfaces;
determining the forensic soundness of the plurality of collection functions and partitioning the plurality of collection functions into a priority one set which pass a first forensic soundness threshold, a priority two set that pass a second forensic soundness threshold, and a discard set;
determining the execution order of the collection functions in the priority one set and the priority two set; and
providing a programmatic interface to the target digital device or software service, wherein in use the programmatic interface receives a first target device or service and executes the collection functions in the priority one set and the priority two set in the determined execution order to obtain data from first target device or service for forensic analysis.
2. The method as claimed in claim 1, wherein the target entity is a hardware digital device, and identifying one or more remote interfaces comprises identifying one or more of a local physical interface or a remote interface and a corresponding software agent for interfacing with the hardware digital device.
3. The method as claimed in claim 1 or 2, further comprising selecting the most feasible remote interface wherein selecting the most feasible remote interface is based at least upon whether authentication is feasible on a remote interface, and if authentication is not feasible then selecting another remote interface from the one or more remote interfaces.
4. The method as claimed in claim 3 wherein selecting a plurality of data collection functions comprises:
explicating the available collection functions via the remote interface;
selecting a set of collection functions from the explicated collection functions based upon one or more constraints or criteria; and
ordering the selected set of collection functions.
5. The method as claimed in any one of claims 1 to 4, wherein determining the forensic soundness of the plurality of collection functions and partitioning the plurality of collection functions comprises:
determining, for each collection function whether the collection function will result in modifications to the data source or if it will lead to a loss of data and whether the practitioner has a complete understanding of the results provided by the collection function;

adding the collection function to a priority one set of collection functions if the collection function will not result in modifications to the data source or lead to a loss of data and is understood; and

if the collection function will result in modifications to the data source or data, then adding the collection function to a priority two set of collection functions only if the collection function is integral, otherwise discarding the collection function.

6. The method as claimed in claim 5, wherein partitioning the plurality of collection functions further comprises:

determining, for each collection function in the priority one set of collection functions, the forensic soundness, and if the forensic soundness passes a first threshold, keeping the collection function in the priority one set of collection functions, else moving the collection function to the priority two set of collection functions; and

determining, for each collection function in the priority two set of collection functions, the forensic soundness, and if the forensic soundness passes a second threshold, keeping the collection function in the priority two set of collection functions, else discarding the collection functions;

7. The method as claimed in any one of claims 1 to 6, further comprising specifying one or more case specific parameters, and determining the execution order comprises determining the execution order of the collection functions in the priority one set and the priority two set and the case specific parameters.

8. The method as claimed in any one of claims 1 to 7, further comprising designing an output format for the data returned by the functions collection functions in the priority one set and the priority two set to facilitate further analysis and/or reporting of the collected data.

9. The method as claimed in claim 1, wherein the method is repeated for a plurality of target digital device or software service to generate a library of programmatic interfaces.

10. A computer readable medium comprising instructions for causing a computer to perform the method of any one of claims 1 to 9.

11. A computing apparatus comprising a communications interface, a memory and a processor, wherein the processor is configured to perform the method of any one of claims 1 to 9.

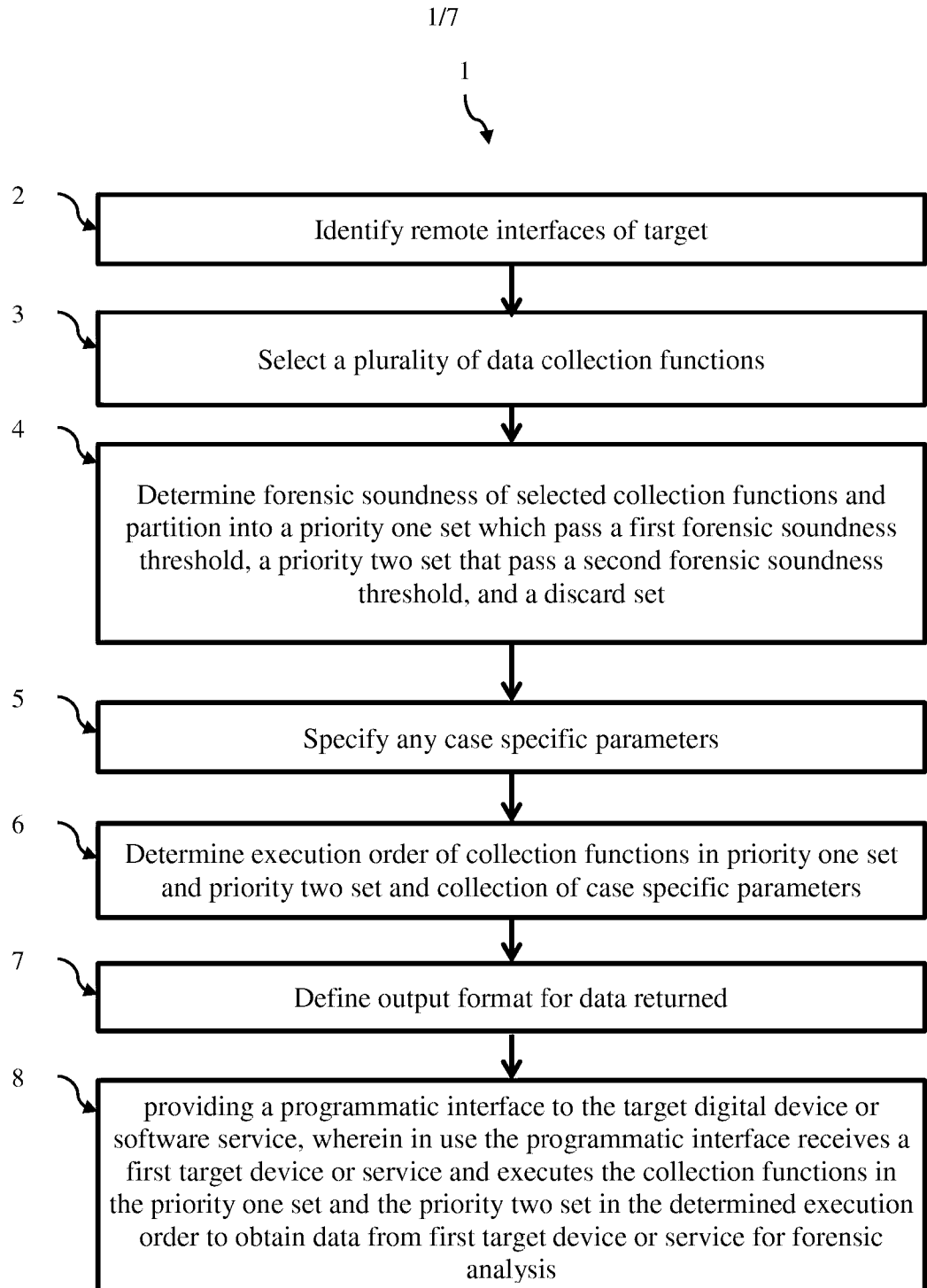


Figure 1

2/7

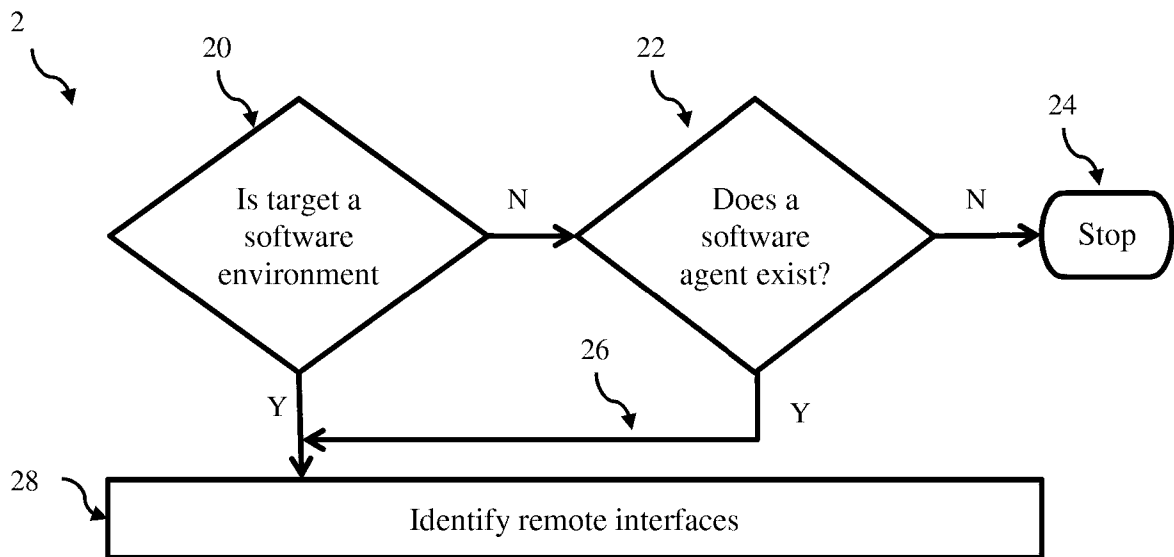


Figure 2A

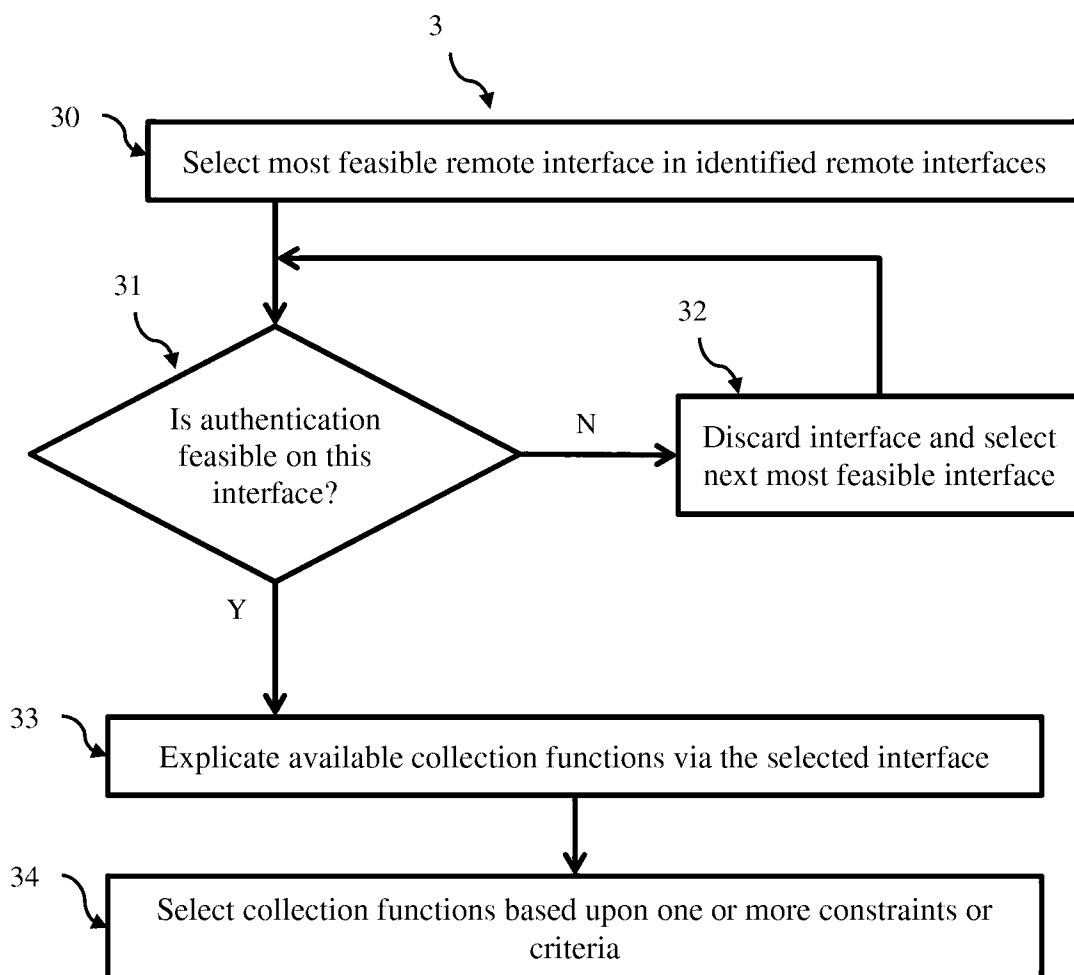


Figure 2B

3/7

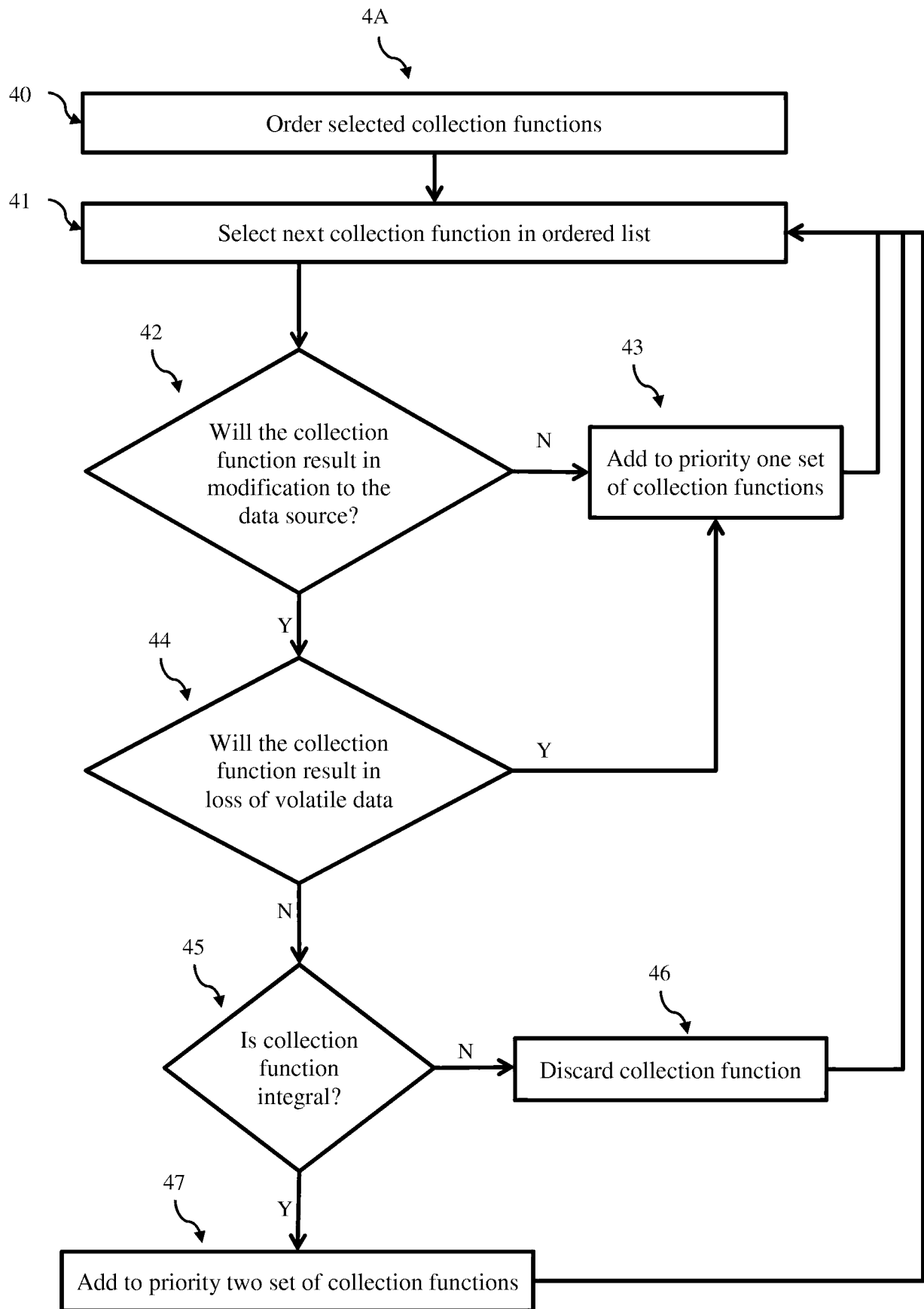


Figure 2C

4/7

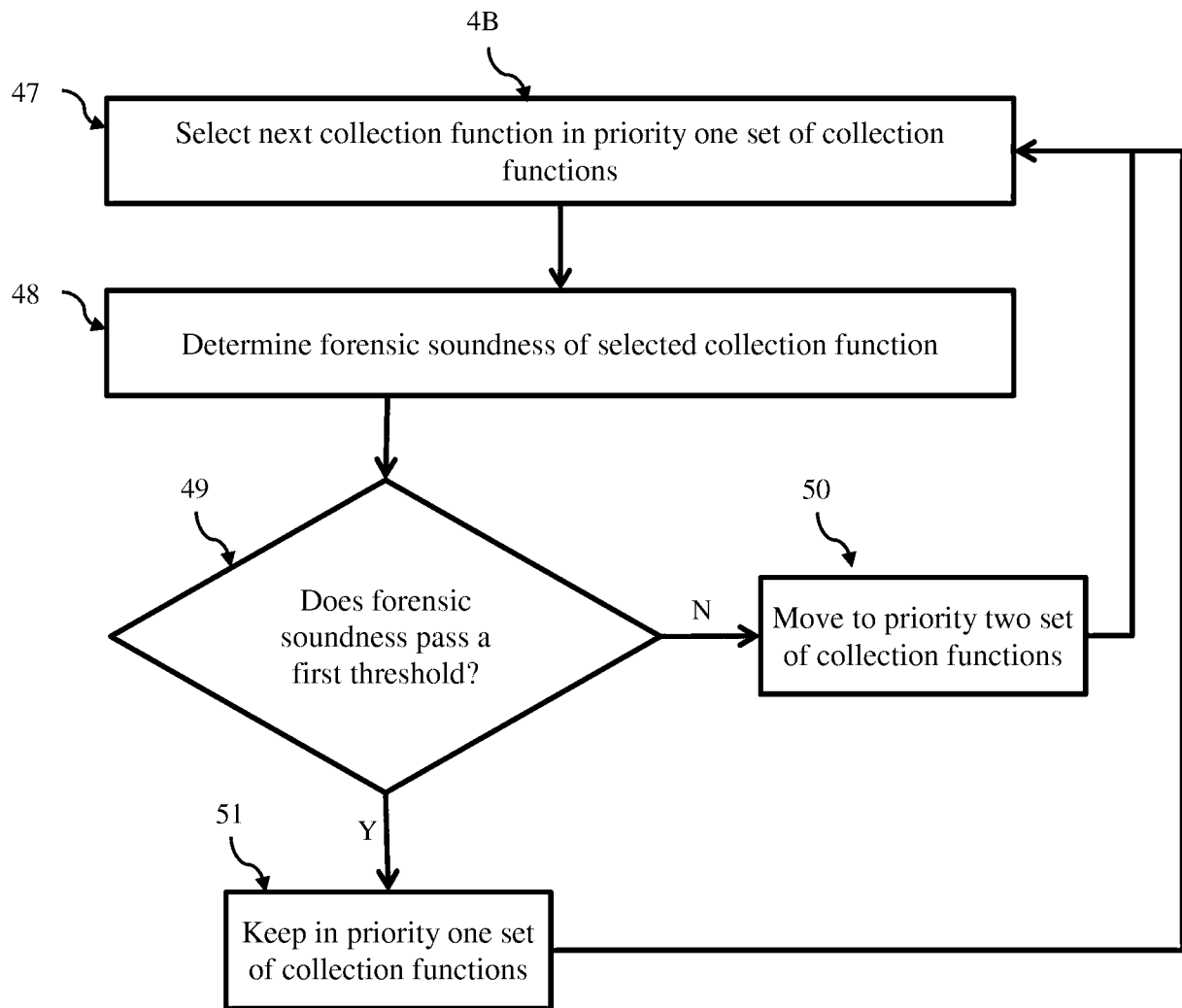


Figure 2D

5/7

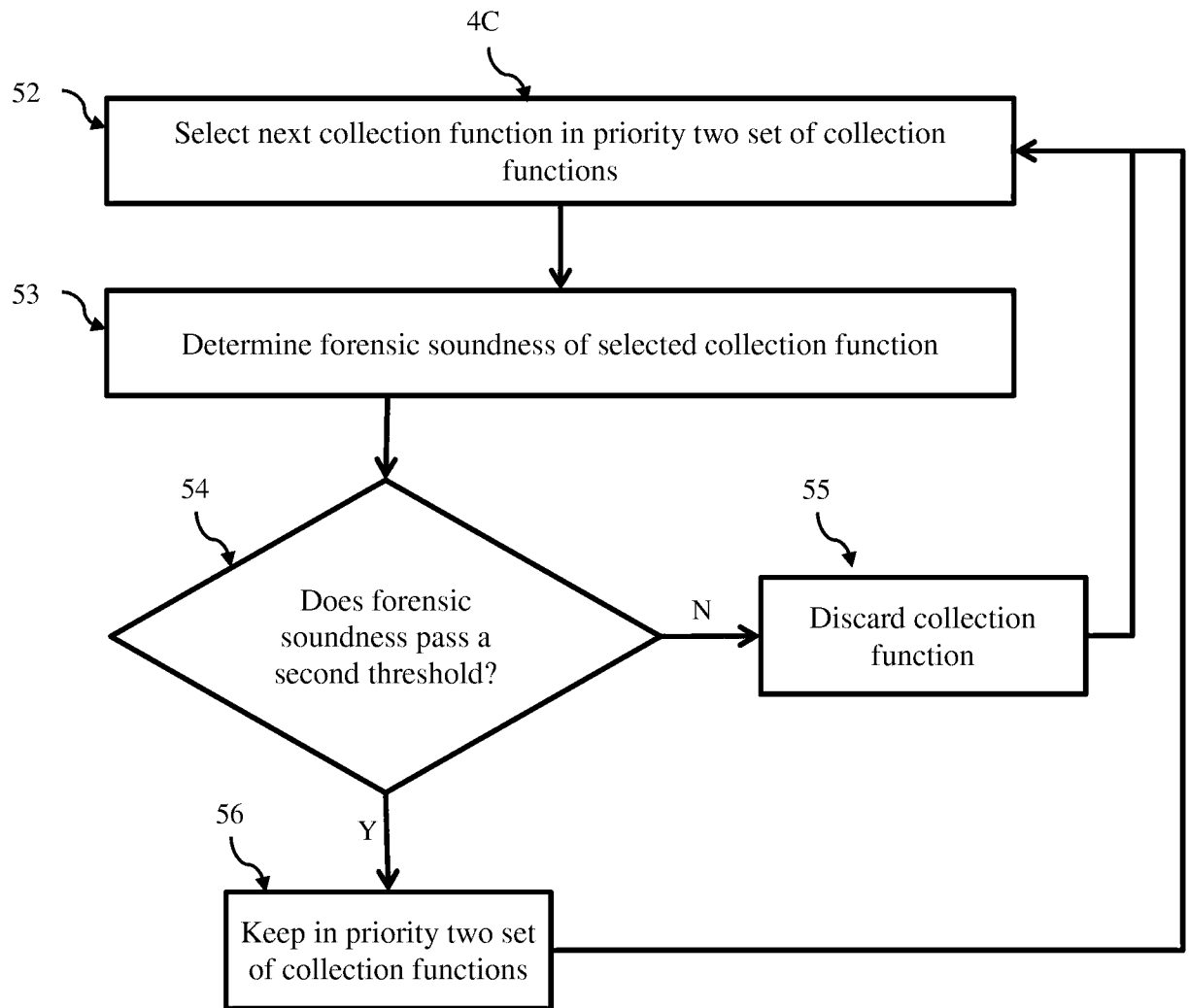


Figure 2E

6/7

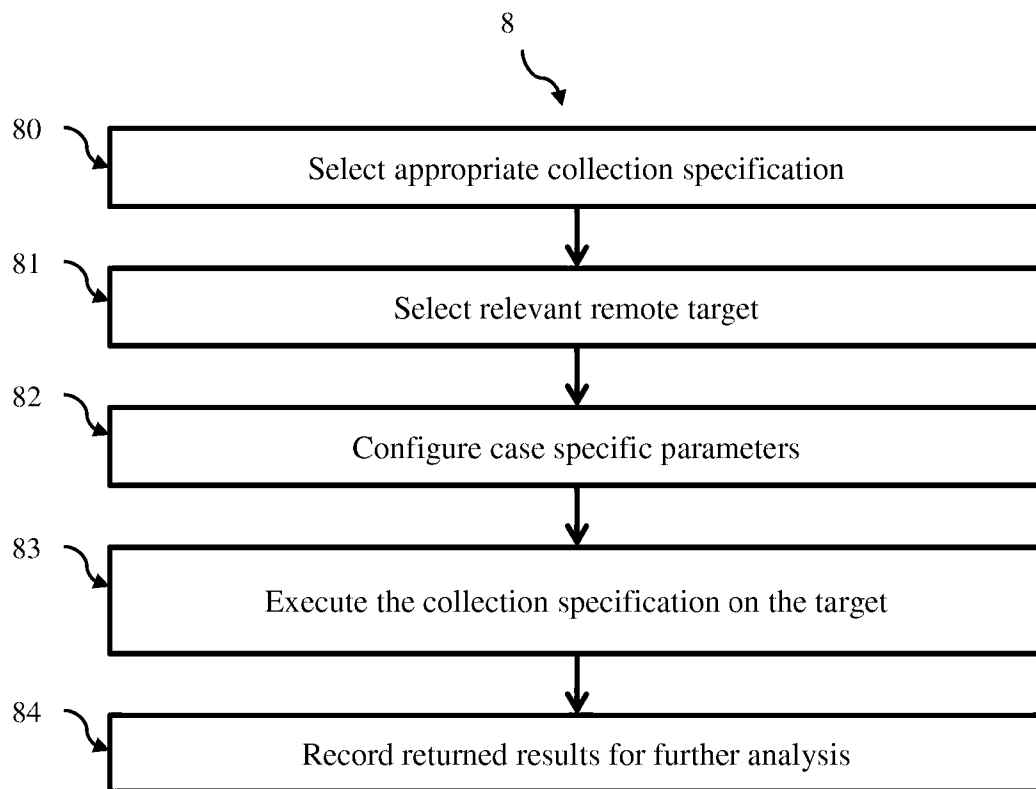


Figure 3

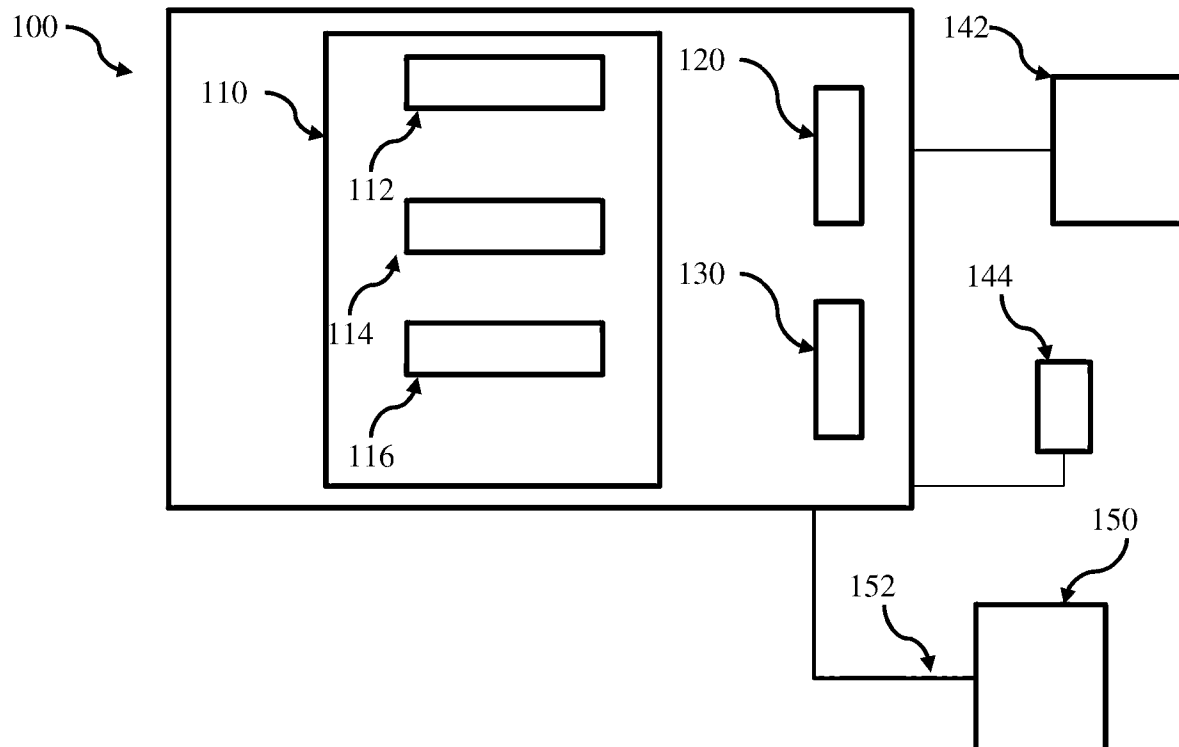


Figure 4

500 → 02:09:09 - Collection commenced at 2014\06\14
02:09:09 UTC
02:09:09 - Connecting to: vcloudhost
02:09:17 - Connected and logged in as: clouddongad-in

Event Collection:
02:09:17 - Total available events to collect: 20
02:09:17 - Greater than 10 events - output to screen
has been suppressed. Outputting to
.vCollect_vcloudhost_events.txt
02:09:17 - Collection complete. Total events
collected 20 matches expected 20 events.

Organization Collection:
02:09:17 - Organization 1: clouddong
ID: clouddong
Full Name: A Cloud Organization
Description: Customer#: 123456; Address: 123
Cloud St, Cloudville; Phone: 555-5555
Default vApp Storage Lease Time: 720 hours

02:09:17 - Organization 1 Users: 2 users found.
Outputting details to
.vCollect_vcloudhost_clouddong_users.txt

User 1:
Username: clouduser
Full Name: Mr. Cloud User
Email: cloud@user.cloud
IM Address: cloudin@user.cloud

User 2:
Username: clouddongadmin
Full Name: Mr. Cloud Admin
Email: cloud@admin.cloud
IM Address: cloudin@admin.cloud

Figure 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU2015/000424

A. CLASSIFICATION OF SUBJECT MATTER

G06F 7/00 (2006.01) G06Q 50/18 (2012.01)

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)

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

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

WPIAP, EPODOC: keywords: digital, computer, electronic, forensic, evidence, collect, gather, obtain, recover, remote, network, interface, target, device, software, material, data, artefact, environment, soundness, credible, reliable, priority, triage, order, rank, weigh, classify, partition, group, separate, split, analyse, investigate and like keywords

Google Patents: keywords: collect, digital, forensic, evidence and like keywords

Espacenet, AusPat, AU Internal Databases: Applicant / Inventor

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	Documents are listed in the continuation of Box C	

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

* "A"	Special categories of cited documents: document defining the general state of the art which is not considered to be of particular relevance	"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
"E"	earlier application or patent but published on or after the international filing date	"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
"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)	"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
"O"	document referring to an oral disclosure, use, exhibition or other means	"&"	document member of the same patent family
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search
03 September 2015

Date of mailing of the international search report
03 September 2015

Name and mailing address of the ISA/AU

AUSTRALIAN PATENT OFFICE
PO BOX 200, WODEN ACT 2606, AUSTRALIA
Email address: pct@ipaustalia.gov.au

Authorised officer

Lili Eng
AUSTRALIAN PATENT OFFICE
(ISO 9001 Quality Certified Service)
Telephone No. (02) 6283 2166

INTERNATIONAL SEARCH REPORT		International application No.
C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		PCT/AU2015/000424
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 8458805 B2 (ADELSTEIN et al.) 04 June 2013 Abstract, col 1 ln 19-20, col 1 ln 66 to col 2 ln 7, col 2 ln 45-54, col 3 ln 1-20, col 4 ln 16-21, col 7 ln 15-42, col 7 ln 64 to col 8 ln 4, col 8 ln 19-23, col 8 ln 25-32, col 12 ln 21-29, col 13 ln 55-57, col 14 ln 25-29, col 16 ln 11-17, col 18 ln 39-44, Figures [1-4, 8-9, 11-13B]	1-11
X	US 2012/0102571 A1 (SHELDON) 26 April 2012 Abstract, Paragraphs [0019, 0040, 0067], Figures [3a, 3b]	1-11
X	US 7370072 B2 (JESSEN) 06 May 2008 Abstract, col 1 ln 7-9, col 6 ln 2-4, Figures [1, 2, 4]	1-11
X	US 8745100 B2 (KIM et al.) 03 June 2014 Abstract, col 2 ln 40-42, Figure [1]	1-11
A	US 7603344 B2 (BOUSQUET et al.) 13 October 2009	
A	US 2011/0058048 A1 (ELAZAR et al.) 10 March 2011	

Form PCT/ISA/210 (fifth sheet) (July 2009)

INTERNATIONAL SEARCH REPORT		International application No.	
Information on patent family members		PCT/AU2015/000424	
This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.			
Patent Document/s Cited in Search Report		Patent Family Member/s	
Publication Number	Publication Date	Publication Number	Publication Date
US 8458805 B2	04 June 2013	US 2009288164 A1	19 Nov 2009
		US 8458805 B2	04 Jun 2013
		US 2004260733 A1	23 Dec 2004
		US 7496959 B2	24 Feb 2009
		US 2008047010 A1	21 Feb 2008
		US 7818804 B2	19 Oct 2010
		US 2009150998 A1	11 Jun 2009
		US 8176557 B2	08 May 2012
		US 2012221633 A1	30 Aug 2012
		US 8474047 B2	25 Jun 2013
US 2012/0102571 A1	26 April 2012	US 2012102571 A1	26 Apr 2012
		EP 2430580 A1	21 Mar 2012
		GB 2470198 A	17 Nov 2010
		GB 2482840 A	15 Feb 2012
		WO 2010142938 A1	16 Dec 2010
US 7370072 B2	06 May 2008	US 2004006588 A1	08 Jan 2004
		US 7370072 B2	06 May 2008
		AU 2003248875 A1	23 Jan 2004
		CA 2490984 A1	15 Jan 2004
		CA 2514661 A1	26 Aug 2004
		EP 1552430 A1	13 Jul 2005
		EP 1590734 A1	02 Nov 2005
		US 2004006589 A1	08 Jan 2004
		US 7475107 B2	06 Jan 2009
		WO 2004006136 A1	15 Jan 2004
		WO 2004072851 A1	26 Aug 2004
US 8745100 B2	03 June 2014	US 2011047130 A1	24 Feb 2011
		US 8745100 B2	03 Jun 2014
		KR 20110020051 A	02 Mar 2011
		KR 101078288 B1	31 Oct 2011
US 7603344 B2	13 October 2009	US 2007085710 A1	19 Apr 2007
		US 7603344 B2	13 Oct 2009
		US 2007085711 A1	19 Apr 2007
		US 7941386 B2	10 May 2011

Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.

Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.

Form PCT/ISA/210 (Family Annex)(July 2009)

INTERNATIONAL SEARCH REPORT Information on patent family members		International application No. PCT/AU2015/000424	
This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.			
Patent Document/s Cited in Search Report		Patent Family Member/s	
Publication Number	Publication Date	Publication Number	Publication Date
		US 2011295886 A1	01 Dec 2011
		US 8219588 B2	10 Jul 2012
		US 2007139231 A1	21 Jun 2007
		US 2010005073 A1	07 Jan 2010
		WO 2007075813 A2	05 Jul 2007
		WO 2007081960 A2	19 Jul 2007
US 2011/0058048 A1	10 March 2011	US 2011058048 A1	10 Mar 2011
		WO 2010097708 A2	02 Sep 2010
End of Annex			
Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001. Form PCT/ISA/210 (Family Annex)(July 2009)			