(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 October 2016 (06.10.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/156513 A1

(51) International Patent Classification:
G06Q 30/02 (2012.01)

(21) International Application Number:
PCT/EP2016/057109

(22) International Filing Date:
31 March 2016 (31.03.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
1505553.6 31 March 2015 (31.03.2015) GB

(71) Applicant: **IRDETO B.V.** [NL/NL]; 105 Taurus Avenue, 2132 LS Hoofddorp (NL).

(72) Inventors: **GU, Yuan Xiang**; 2500 Solandt Road, Ottawa, Ontario K2K 3G5 (CA). **WAJS, Andrew, Augustine**; Irdeto B.V., 105 Taurus Avenue, 2132 LS Hoofddorp (NL). **GIDLEY, Benjamin, Geoffrey**; 7-8 Eghams Court, Boston Drive, Bourne End SL8 5YS (GB). **MOOIJ, Wim**; Irdeto B.V., 105 Taurus Avenue, 2132 LS Hoofddorp (NL).

(74) Agents: **BOULT WADE TENNANT** et al.; Verulam Gardens, 70 Gray's Inn Road, London WC1X 8BT (GB).

(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: ONLINE ADVERTISEMENTS

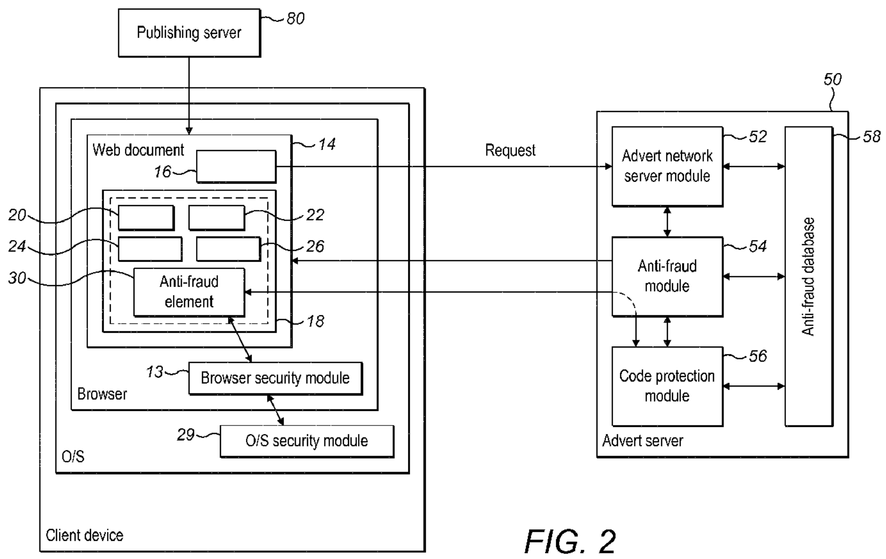


FIG. 2

(57) **Abstract**: There are disclosed techniques for averting advertising fraud, for example a method of operating a server to deliver an online advert to a client device, the method comprising: receiving at the server, from a web document executing in a web browser on the client device, a request for an advert; in response to the request for an advert, preparing, at the server, advert code for execution within the web document at the client device so as to display the advert to a user of the client device, at least a portion of the prepared advert code being in a protected form for averting advertising fraud; and transmitting the advert code to the client device for execution on the browser for execution so as to display the advert to a user of the client device.

WO 2016/156513 A1

ONLINE ADVERTISEMENTS

FIELD OF THE INVENTION

The invention relates to the delivery of online advertisements to client devices, and techniques which can be used to detect and protect against various fraudulent activities relating to such adverts.

BACKGROUND OF THE INVENTION

Online advertising

Web publishers publish online content that attracts an audience. The publishers sell space in their websites to advertisers so that the advertisers can reach their intended audience. The whole process begins when a consumer uses a browser to visit a webpage. The browser opens a connection to the publisher's content server and the server returns the content for the page. If the page contains ad, the browser requests the advert network to serve an ad. The advert network may connect to other advert networks or advert exchanges with necessary client information to better serve the particular client and the advertisers. There may or may not be an automatic bidding process. After an advert is selected, the advert network returns the advert information to the browser. The browser now visits the agency advert server to retrieve the advert creative (image/rich media) and the advert network records the request as an impression. The browser now renders the actual page with the advert creative. The whole process takes less than one second and more systems may be involved in the process. The advert creative contains a link to the advert network. When any user clicks on an ad, the advert network first registers the click for billing purposes and then redirect the browser to the advertiser's page.

Thus, online advertising is a form of marketing which uses websites to deliver promotional messages to consumers. Online advertising includes email marketing, search engine marketing (SEM), social media marketing, many types of display advertising, and mobile advertising. It is the main financial incentive for free web contents and services as well as free mobile apps. The online advertising industry has expanded rapidly and grown into a $121 billion industry with revenue projected to reach $161 billion by 2016. The largest revenue shares within internet advertising are generated by display-based and search based advertising.

Online advertising networks serve advertisements into the web browser display of client devices such as personal computers, tablet computers and smart phones. Typically, an advertisement network operator contractually acquires advertisement space on host web sites, and sells this space on to advertisers. The advertisement network then manages the injection of advert material provided by the advertisers into the presentation of the host web site on the client devices.

Online advertising is readily amenable to measurement of advert delivery and effectiveness, and such measurements may form a basis of the contractual terms between advertisers, advertising networks, and host websites. For example, an advertising network will typically record details such as the number of times a particular advert has been delivered to client devices, and such statistics may be quite sophisticated for example recording demographic details of the client devices and associated users, distinguishing between first and subsequent delivery of an advert to a particular user or device. Advertising networks also typically use sophisticated techniques including tracking browsing activity of particular users, in order to target online adverts more effectively.

Because mere injection of an advert into a web document being displayed on a browser does not guarantee viewing by a user, advertisers are interested in paying advertising networks on more sophisticated terms, for example depending on whether an advert actually appeared on the viewable area of a browser window (rather than being outside the current scroll area or being covered by another window), and whether a user interacted with an advert for example by moving a cursor over the advert, and clicking, tapping or otherwise selecting the advert.

Advertisers and publishers use a wide range of payment calculation methods in online advertising contracts. But they can be broadly categorized into three models, namely, PPM, PPC and PPA. In pay per mille or PPM (also known as cost per mille or CPM), advert networks charge advertisers every thousand displays of their advert to potential customers (mille is the Latin word for thousand). In the online context, advert displays are usually called “impressions”. In PPC (Pay Per Click) or CPC (Cost per click), advertisers pay each time a user clicks on the ad. PPC is the current dominant revenue model for online advertisement. However, such contracts are open to abuse and fraud, for example by the operator of a host website boosting the number of downloads, viewable impressions, or clicks on an advert actually or apparently injected into display of their own website, in order to increase their revenue from the advertising network operator, or a third party carrying out similar activities in order to deplete the advertising budget of a competitor. Thus, not all recorded clicks are valuable to advertisers. There are claims that up to 36% of all clicks are fraudulent.

Click fraud

Such fraudulent activity may be carried out in various ways. The biggest threat to PPC advertisement is click fraud. Click fraud occurs when illegitimate sources click on online adverts with malicious intent. Such clicks are often called “invalid clicks”. Invalid clicks are any clicks that an advert network chooses not to charge for. When clicks are marked invalid, the user agent that issued the click is still directed to an advertiser’s website. Since intent is only in the mind of the person issuing the click or in the mind of the author of software that issues clicks, it is difficult to know with certainty that any click is fraudulent. It is not trivial to detect all invalid clicks due to the server-based detection techniques used by the advert networks. In fact, about 36% of all web traffic is considered fake, the product of computers hijacked by viruses and programmed to visit sites, according to estimates cited by the Interactive Advertising Bureau trade group.

The intentions of click fraud can be different. Different attackers have their own motivations. Enriching click fraud involves generating invalid advert clicks to boost pay per click (PPC) income. Depleting click fraud involves clicking rival's adverts to exhaust their PPC advertising budget. Thus, click fraud includes the practice of deceptively clicking on online adverts with the intention of either increasing third party website revenues or exhausting an advertiser's budget. Website publishers or rival advertisers may impersonate consumers and click search ads, driving up advertising costs without increasing sales, effectively stealing a firm's paid advertising inventory. Another type of click fraud is disbarring click fraud which involves attempting to frame a rival by generating invalid clicks that appear to be associated with the rival, in hopes that this rival will be banned from an advert network or punished in search engine listings.

Click fraud may be accomplished by employing a number of people in a low cost jurisdiction to visit target host websites using normal client devices such as personal computers operating normal web browsers, so that advertisements are downloaded to the client devices to record an impression in the advertising network. Advertisements may also be clicked or otherwise selected to record a click. Such fraud operations are often referred to as click farms. Members of a click farm click on adverts with no intention of buying or performing any actions on the advertisers page.

<u>Automated clickers</u>

Technologically more sophisticated fraud operations may operate a collection of client devices automatically to try and achieve the same effect, simulate such client devices on servers, or operate less complete simulations using non browser software to simulate different parts of the advertising operation such as requesting advert downloads, reporting impressions, reporting advert clicks and so forth.

Software that can automatically click on online adverts is called a clickbot. A clickbot can be independent malware (with/without own browser) that infects internet users' PCs, PCs of users willingly participated in a botnet, or infected browsers (browser helper objects/extensions). Clickbots can be very sophisticated and often equipped with capabilities of a real browser. They crawl to different websites and click on links provided by their user or bot-masters. Some of them can imitate real human browsing behaviour and mouse movement.

Clickbots go to different publishers' pages and click programmatically. Botnets have been extensively used for click fraud during recent years to launch low-noise large-scale attacks to avoid detection. A botnet is a network of malware-infected machines that are controlled by a bot-master. However, advertisers are forced to trust that advert networks detect and prevent click fraud even though they get paid for every undetected fraudulent click.

A clickbot in a botnet performs some common functions including initiating HTTP requests to a webserver, following redirections, and retrieving contents from a web server under the control of a remote bot-master. A bot-master can leverage millions of clickbots to perform automatic, low-noise, and large-scale click fraud attacks. For example, first, the bot-master may use the internet to distribute malware to the victim host. Once compromised, the victim host becomes a bot and receives instructions from a command-and-control (C&C) server controlled by the bot-master. Such instructions may specify the target website, the number of clicks to perform on the website, the referrer to be used in the fabricated HTTP requests, what kinds of adverts to click on, and when or how often to click. After receiving instructions, the clickbot begins traversing the designated publisher website and simulates a click on each selected ads. The advert network logs the click traffic and then returns a HTTP 302 redirect response to the advertiser's page. Every time an advert is clicked by a clickbot, the advertiser pays the advert network if the click is not detected as "invalid", and the involved publisher receives a portion of the revenue from the advert network.

<u>Techniques used in malicious clickbots</u>

A wide range of click fraud attacks have been reported in the literature. Daswani et al., *Proceedings of the first conference on Hot Topics in Understanding Botnets*, USENIX Association, 2007, analysed "Clickbot.A" malware and found that the bot attempts a low-noise click fraud attack against syndicated search engines. Launching low-noise attacks from millions of different infected computers has become popular among bot-masters as it helps them to avoid detection.

Miller et al. in *Proceedings of the Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, 2011, pp. 164–183, examined two other clickbot families and found more advanced techniques used to evade detection. One clickbot introduced indirection between clickbots and advert networks through middlemen (i.e., service providers), while the other bot exhibited human-like browsing behaviours.

Dave et al. in *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, ACM, 2012, pp. 175–186, described a number of characteristics of clickbots. Often, clickbots generate fraudulent clicks periodically and only issue a click in the background when a legitimate user clicks on a link, which makes fraudulent traffic hardly distinguishable from legitimate traffic.

Normal browsers may also be exploited to generate fraudulent traffic. The traffic generated by a normal browser could be hijacked by currently visited malicious publisher and be further converted to fraudulent clicks. Also, many click fraud malware infect browsers or use a browser driver to control real browsers to generate traffic and thus avoiding detection that uses client capability test. In Alrwais et al. *Proceedings of the 28th Annual Computer Security Applications Conference*, ACM, 2012, pp. 21–30, ghost click botnet leveraged DNS changer malware to convert a victim’s local DNS resolver into a malicious one and then launched advert replacement and click hijacking attacks.

<u>Click fraud detection</u>

Metwally et al. *Proceedings of the 31st international conference on Very large data bases*, VLDB Endowment, 2005, pp. 169–180, presented three forms of click fraud and methods to detect them. They found that several websites can cooperate with each other to create fraudulent clicks and thus advance their commercial interests. They developed an algorithm, called streaming-rules, to detect fraud in advertising networks. Immorlica et al., *Proceedings of the Internet and Network Economics*. Springer, 2005, pp. 34–45, studied fraudulent clicks and presented a click fraud resistant method for learning the click through rate of advertisements.

Haddadi, *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 21–25, 2010, suggested that advertisers can use bluff adverts to detect fraudulent clicks on their ads. While bluff adverts may be effective in detecting click fraud, advertisers have to spend extra money on those bluff ads. Also, for the publisher, it is not good to show meaningless advert to the real users.

Dave et al. *Proceedings of the ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 765–776, presented an approach to catch click spam from an advert network's perspective. It is designed around the invariant that click-spam is a business (for click-spammers) that needs to deliver high return on investment (ROI) to offset the risk of getting caught. Schulte et al. [27] detected malware using program interactive challenge (PIC) mechanism. However, in their approach, an intermediate proxy has to be introduced to examine all HTTP traffic between a client and a server, which is practically not feasible.

Xu et al., *Proceedings of the 19th European Symposium on Research in Computer Security*, Springer, 2014, pp. 419–438, proposed a new approach for advertisers to independently detect click fraud activities issued by clickbots and human clickers. Their proposed detection system performs two main tasks of proactive functionality testing and passive browsing behaviour examination. The purpose of the first task is to detect clickbots. It requires a client to actively prove its authenticity of a full-fledged browser by executing a piece of JavaScript code. For more sophisticated clickbots and human clickers, the system observes what a user does on the advertised site. However, as mentioned before, their technique is vulnerable to malware that infect real browsers or perform click fraud using own version of full-fledged browser and imitate human-like behaviour.

Crussell et al. *Proceedings of the 12th annual international conference on*

Mobile systems, applications, and services, ACM, 2014, pp. 123–134. analysed Android malware apps to investigate advert related fraud. They achieved an overall class weighted accuracy of 85.9% in classifying advert requests. Kitts et al. *Real World Data Mining Applications*, Springer, 2015, pp. 181–201. presented a data mining approach to score advert traffic quality and showed some results on the effectiveness of the system.

Other fraud operations may make use of bona fide users and their client devices by installing malware which interferes with the browser operation for example by downloading and displaying adverts preferred by the fraud operator either in place of other adverts or in places where adverts were not intended by the host web site, automatically generating advert clicks, changing browser navigations, adding pop-up windows, and so forth, in ways which affect data recorded by the advertisement network relating to the adverts. Fraud operations may also hijack or interfere with HTML sessions between the client device of a bona fide user and an advert network, through network attacks of various kinds.

Consequently, the prior art includes various proposals for better security and monitoring of the delivery of online advertisements, and techniques to help prevent fraudulent activity such as boosting the number of recorded impressions and user interactions.

The invention seeks to address limitations of the related prior art.

SUMMARY OF THE INVENTION

The invention provides improved detection and/or prevention of online advertising fraud. Aspects of the invention are based on the fundamental components of the online advertising ecosystem: publisher, client environment (webpage and browser) interacting with an advertising network. A client centric model is proposed based on secured interaction between code (for example HTML and JavaScript) at the client, and a trusted publisher and advertising network. Stronger security within the protected client advertising environment can make the anti-fraud solution more effective, accurate and measurable.

In particular, a solution which can be tightly integrated into existing online advertising eco-systems without changing existing online advertising infrastructures is proposed. To this end, advert code and/or content at the client is at least partially protected, for example by applying cloaking technology, integrity verification, diversity techniques and so forth.

Providing a client based protection scheme can provide a much more fine grained solution than more global protection approaches, simplifying detection performance and improving detection results. Unlike existing global based anti-fraud approaches, the proposed client based anti-fraud approach can provide truly trusted user data that can be used to accurately capture fraud characteristics. Therefore, the proposed client based anti-fraud system can work in conjunction with global based anti-fraud systems to achieve a much higher degree of detection accuracy with well-defined metrics.

The invention provides a number of anti-fraud methodologies at both the client and server sides. The invention also provides apparatus arranged to implement the methods and techniques discussed herein, including, both separately and in combination, one or more servers and one or more client devices.

According to a first aspect, the invention provides a method of operating a server to deliver an online advert to a client device, the method comprising: receiving at the server, from a web document executing in a web browser on the client device, a request for an advert; in response to the request for an advert, preparing, at the server, advert code for execution within the web document at the client device so as to display the advert to a user of the client device; and transmitting the advert code to the client device for execution on the browser so as to display the advert to a user of the client device. Preferably, at least a portion of the prepared advert code is in a protected form so as to avert advertising fraud, for example through detection or prevention of such fraud.

The advert code may be in JavaScript (JS) and/or HTML and/or WebAssembly and/or any other language / source code that is executable in a browser environment.

The advert code may comprise an anti-fraud element at least partly within the protected portion of the advert code. The anti-fraud element is arranged for execution within the web document at the client device so as to provide one or more anti-fraud functions relating to the advert. The anti-fraud functions may include one or more anti-fraud verifications arranged to verify integrity of the anti-fraud element and/or integrity of the advert and/or that the anti-fraud element is executing within a browser and/or visibility of the advert in a graphical display of the browser and/or behavioural characteristics of the user of the browser and/or structure of the DOM of at least part of the web document.

The advert code may comprise an interaction action which at least partly defines one or more interactions between a user of the browser and the advert, and one or more actions to be taken by the browser when the one or more interactions occur, for example navigating to an advertiser webpage when an advert link is selected or clicked by a user

The anti-fraud element may be arranged to provide one or more of the anti-fraud functions following occurrence of one or more of the defined user interactions with the advert. The method may further comprise, following said one or more anti-fraud functions, permitting the browser to complete an action which is defined by the interaction action. Permitting an action to be completed by the browser may comprise one of: transmitting to the browser a redirection URL for the browser to navigate to in response to the user interaction with the advert; and permitting the browser to navigate to a redirection URL already comprised in the web document, in response to the user interaction with the advert. The method may further comprise recording the user interaction as a suspect or fraudulent user interaction if one or more of the anti-fraud functions fails.

Preparing the advert code may comprise incorporating an advert serial code within the advert code, and the method may further comprise carrying out an anti-fraud verification that information subsequently received from the advert code executing on the browser corresponds with the advert serial code.

The method may further comprise: storing, on the server, one or more initial attributes of the prepared advert code; receiving a subsequent message from the client device comprising one or more later attributes of the advert code; and verifying the one or more later attributes of the advert code against the corresponding stored initial attributes of the advert code .

The method may further comprise: receiving, from the client device, cursor trajectory data relating to cursor movement associated with an interaction between a user of the browser and the advert; and verifying the cursor trajectory data by comparing the received cursor trajectory data against previous cursor trajectory data stored on the server. The comparing may comprise calculating a geometric distance between the received cursor trajectory data and the previous cursor trajectory data. The verifying may comprise comparing the geometric distance to a threshold.

The server may carry out anti-fraud verification of the request for an advert before transmitting the advert code to the web browser. Such anti-fraud verification of the request for an advert may comprise checking an origin of the request against a database relating origin of a request to fraud risk.

At least some of the prepared advert code in a protected form may be protected by one or more cloaking techniques and/or one or more software obfuscation techniques and/or one or more node locking techniques and/or one or more diversity techniques and/or one or more digital watermarking techniques. The cloaking techniques may include one or more of homomorphic data transformation, control flow transformation, white box cryptography, key hiding, program interlocking and boundary blending.

According to a second aspect, the invention provides a server arranged to deliver an online advert to a client device by carrying out the method of the first aspect.

According to a third aspect, the invention provides a method of operating a client device to display an online advert using a web browser comprising: receiving and executing, using the browser, a web document including advert request code, execution of the advert request code causing the browser to transmit an advert request to an advert server; receiving advert code from the advert server, at least a portion of the advert code being in a protected form for averting advertising fraud; and executing the advert code at the client device so as to present the advert to a user of the client device.

The advert code may comprise an anti-fraud element at least partly within the protected portion of the advert code, and the method may comprise executing the anti-fraud element within the web document on the client device to provide one or more anti-fraud functions relating to the advert. The anti-fraud functions may include one or more anti-fraud verifications arranged to verify integrity of the anti-fraud element and/or integrity of the advert and/or that the anti-fraud element is executing within a browser and/or visibility of the advert in a graphical display of the browser and/or behavioural characteristics of the user of the browser and/or structure of the DOM of at least part of the web document.

The advert code may comprise an interaction action which at least partly predefines one or more interactions between a user of the browser and the advert, and one or more actions to be taken by the browser if and when the one or more interactions occur. The anti-fraud element may be executed to provide one or more of said anti-fraud functions following occurrence of one or more of the defined user interactions with the advert.

The method may further comprise, following said one or more anti-fraud functions, permitting the browser to complete an action which is defined by the interaction action.

The advert code may comprise an advert element including an advert URL associated with the advert.

The anti-fraud element may be arranged to replace the advert URL with a fake URL until one or more conditions are satisfied, the fake URL being different from the advert URL. The one or more conditions may include one or more of: a cursor being present over the advert whilst the advert is being displayed to the user; a fixed time period having elapsed following loading of the web document on the browser; and a frame rate of the browser having stabilised following loading of the web document on the browser.

The anti-fraud element may be arranged to periodically replace the advert URL with an updated advert URL. The anti-fraud element may be arranged to periodically replace the advert URL in response to user interactions with the web document using the browser.

The anti-fraud element is arranged to create copies of the advert element within the DOM of the web document, each copy including a respective fake URL different from the advert URL, such that the advert and copies of the advert are displayed in a stack with only the advert being visible to the user at the top of the stack. The anti-fraud element may be arranged to periodically alter the DOM structure of the web document so as to randomly rearrange the advert element and the copies of the advert element within the DOM structure. The anti-fraud element may be arranged to periodically alter the DOM structure of the web document in response to user interactions with the web document using the browser.

The advert element may further include an advert creative associated with the advert, and the anti-fraud element may be arranged to replace the advert creative with a fake creative until the advert is displayed to the user, the fake creative being different from the advert creative.

The anti-fraud element may be arranged to verify whether the advert is on a viewable portion of the browser window and/or whether the advert is being displayed on a topmost tab within the browser.

The anti-fraud element may be arranged to store on the client device one or more initial attributes of the advert code received from the advert server, and the anti-fraud element may be further arranged to verify one or more later attributes of the advert code against the corresponding stored initial attributes of the advert code.

The anti-fraud element may be arranged to send cursor trajectory data to the advert server, the cursor trajectory data relating to cursor movement associated with an interaction between a user of the browser and the advert.

According to a fourth aspect, the invention provides a client device arranged to display an online advert using a web browser by carrying out the method of the third aspect.

According to a fifth aspect, the invention provides a method of operating a client device. The method comprises monitoring advertising page visits made by one or more processes executing on the client device, and detecting if each advertising page visit is triggered by a user interaction.

The one or more processes may be processes of one or more web browsers executing on the client device, and the advertising page visits may comprise HTTP requests to advertising web sites.

The method may further comprise creating an HTTP request tree for each process and analysing the request trees to identify advertising page visits.

Analysing the request trees may further comprise applying a machine learning classification to the request trees.

The method may further comprise, for each process, monitoring one or more respective hardware input device events, and wherein said detecting comprises, for each process, comparing the one or more respective hardware input device events with the advertising visits associated with the process.

According to a fifth aspect, the invention provides a method of detecting a fraudulent process in a client device, wherein the client device is executing one or more processes. The method comprises, for each process of the one or more processes: monitoring respective requests transmitted across a network interface of the client device; monitoring respective hardware input events associated with the process; identifying one or more respective advert impressions by applying a machine learning algorithm to the respective requests; classifying the one or more respective advert impressions as either fraudulent or non-fraudulent in dependence on the respective hardware input events; and marking the process as fraudulent based on said classification.

The identifying may comprise creating a respective request tree based on the respective requests and applying one or more criteria to the request tree based on the output of the machine learning algorithm.

The identifying may comprise filtering the respective requests using an advert blocking list.

The classifying may comprises mapping the respective hardware events to the one or more respective advert impressions.

The invention also provides aspects in which digital rights management techniques are used to provide verification and security functions in an advertising network. In one such aspect, a method of delivering an online advert to a web browser executing on a client device is provided, the method comprising: transmitting a web document to the client device for execution by the browser; receiving from the web document executing on the browser, at a remote server, a request for a piece of digital rights management (DRM) protected content; delivering the requested DRM protected content to the client device and providing the client device with DRM rights for the browser to replay the content, the content comprising an identifier for the browser to recover from the content when the content is replayed; and receiving the identifier from the browser.

In response to receiving the identifier from the browser, the method may comprise preparing advert code for execution within the web document and transmitting the advert code to the web browser using the methods already set out above.

Providing mechanisms for defending against advertising fraud using a client side solution may be provided in various other ways, for example by operating a client device so as to: monitor advertising page visits made by one or more processes executing on the client device, and detect if each advertising page visit is triggered by a user interaction. The one or more processes may be processes of one or more web browsers executing on the client device, and the advertising page visits may comprise HTTP requests to advertising web sites.

Aspects of the invention also provide an anti-fraud system arranged to receive, from advert request code executing on a browser of a client device, a request for a piece of digital rights management (DRM) protected content; to deliver the requested DRM protected content to the client device and provide the client device with DRM rights for the browser to replay the content, the content comprising an identifier for the browser to recover from the content when the content is replayed; and to receive the identifier from the browser.

The invention also provides computer program elements and computer program code comprising elements for putting into effect some or all of the aspects of the invention, and one or more computer readable media carrying such program elements, and a signal transmitted within a network comprising such program elements. For example, the invention provides advert code, for example in HTML and/or JavaScript and/or WebAssembly, or other suitable languages, as described herein, whether stored at a server, at a client, or in transmission there between.

The present application also discloses subject matter in accordance with the following numbered clauses:

Clause A1. A method of operating a server to deliver an online advert to a client device, the method comprising: receiving at the server, from a web document executing in a browser on the client device, a request for an advert; in response to the request for an advert, preparing, at the server, advert code for execution within the web document at the client so as to display the advert to a user of the client device, at least a portion of the advert code prepared at the server being in a protected form for averting advertising fraud; and transmitting the advert code to the client device for execution on the web browser for execution so as to display the advert to a user of the client device.

Clause A2. The method of clause A1 further comprising the server carrying out anti-fraud verification of the request for an advert before transmitting the advert code to the web browser.

Clause A3. The method of clause A2 wherein carrying out anti-fraud verification of the request for an advert comprises checking an origin of the request against a database relating origin of a request to fraud risk.

Clause A4. The method of any preceding clause wherein the advert code comprises an interaction action which at least partly defines one or more interactions between a user of the browser and the advert, and one or more actions to be taken by the browser when the one or more interactions occur.

Clause A5. The method of clause A4 wherein the advert code comprises an anti-fraud element at least partly within the protected portion of the advert code, the anti-fraud element being arranged to provide one or more anti-fraud verifications relating to the advert.

Clause A6. The method of clause A5 wherein the anti-fraud element is arranged to provide an anti-fraud verifications of each of one or more of: integrity of the anti-fraud element; that the anti-fraud element is executing within a browser; visibility of the advert in a graphical display of the browser; behavioural characteristics of the user of the browser; and structure of the document object model of at least part of the web document.

Clause A7. The method of clause A5 or A6, wherein the anti-fraud element is arranged to provide one or more anti-fraud verifications following occurrence of one or more of the defined user interactions with the advert.

Clause A8. The method of clause A7 comprising, following said one or more anti-fraud verifications, permitting the browser to complete an action which is defined by the interaction action.

Clause A9. The method of clause A8 wherein permitting an action to be completed by the browser comprises one of: transmitting to the browser a redirection URL for the browser to navigate to in response to the user interaction with the advert; and permitting the browser to navigate to a redirection URL already comprised in the web document, in response to the user interaction with the advert.

Clause A10. The method of any preceding clause wherein preparing the advert code comprises incorporating an advert serial code within the advert code, and the method further comprises carrying out an anti-fraud verification that information received from the advert code executing on the browser corresponds with the advert serial code.

Clause A11. The method of clause A10 comprising receiving the information from the advert code executing on the browser in response to a user interaction.

Clause A12. The method of any of clauses A7 to A11 comprising recording the user interaction as a suspect or fraudulent user interaction if one or more of the anti-fraud verifications fails.

Clause A13. A method of operating a client device to display an online advert using a web browser comprising: receiving and executing, using the web browser, a web document including advert request code, execution of the advert request code causing the web browser to transmit an advert request to an advert server; receiving advert code from the advert server, at least a portion of the advert code being in a protected form for averting advertising fraud; executing the advert code at the client so as to present the advert to a user of the client device.

Clause A14. The method of clause A13 wherein the advert code comprises an anti-fraud element at least partly within the protected form portion of the advert code, the method comprising executing the anti-fraud element on the client device to provide one or more anti-fraud verifications relating to the advert.

Clause A15. The method of clause A14 wherein the advert code comprises an interaction action which at least partly predefines one or more interactions between a user of the browser and the advert, and one or more actions to be taken by the browser if and when the one or more interactions occur, and the anti-fraud element is executed to provide one or more of said anti-fraud verifications following occurrence of one or more of the defined user interactions with the advert.

Clause A16. The method of clause A14 or A15 comprising, following said one or more anti-fraud verifications, permitting the browser to complete an action which is defined by the interaction action.

Clause A17. The method of any preceding clause wherein the anti-fraud element is arranged to periodically alter a document object model structure of at least part of the web document.

Clause A18. The method of clause A17 wherein the anti-fraud element is arranged to periodically alter the structure of the document object model of at least part of the web document in response to user interactions with the web document using the browser.

Clause A19. The method of clause A17 or A18 wherein the at least part of the web document at least partly defines at least one of: one or more interactions between a user of the browser and the advert; and one or more actions to be taken by the browser when the one or more interactions occur.

Clause A20. The method of any preceding clause wherein at least some of the prepared advert code in a protected form is protected by one or more cloaking techniques.

Clause A21. The method of clause A20 wherein the cloaking techniques include one or more of homomorphic data transformation, control flow transformation, white box cryptography, key hiding, program interlocking and boundary blending.

Clause A22. The method of any preceding clause wherein at least some of the prepared advert code in a protected form is protected by one or more node locking techniques.

Clause A23. The method of any preceding clause wherein at least some of the prepared advert code in a protected form is protected by one or more diversity techniques.

Clause A24. The method of any preceding clause wherein at least some of the prepared advert code in a protected form is protected by one or more digital watermarking techniques.

Clause A25. A method of delivering an online advert to a web browser executing on a client device comprising: transmitting a web document to the client device for execution by the browser; receiving from the web document executing on the browser, at a remote server, a request for a piece of digital rights management (DRM) protected content; delivering the requested DRM protected content to the client device and providing the client device with DRM rights for the browser to replay the content, the content comprising an identifier for the browser to recover from the content when the content is replayed; and receiving the identifier from the browser.

Clause A26. The method of clause A25 further comprising, in response to receiving the identifier from the browser, preparing advert code for execution within the web document and transmitting the advert code to the web browser according to any of clauses A1 to A24.

Clause 27. The method of clause A25 or A26 wherein the web document is arranged to recover the content from a graphics buffer of the client device during replay of the content.

Clause A28. The method of any of clauses A25 to A27 wherein the identifier is specific to one or more of: the client device, the web browser executing on the client device; and the web document executing on the web browser.

Clause A29. A method of operating a client device comprising: monitoring advertising page visits made by one or more processes executing on the client device, and detecting if each advertising page visit is triggered by a user interaction.

Clause A30. The method of clause A29 wherein the one or more processes are processes of one or more web browsers executing on the client device, and the advertising page visits comprise HTML requests to advertising web sites.

Clause A31. The method of clause A29 or A30 further comprising creating an HTML request tree for each process and analysing the request trees to identify advertising page visits.

Clause A32. A server for delivering an online advert to a client device, comprising: an advert network server module arranged to receive a request for an advert from a web document executing in a browser on the client device; and an anti-fraud module arranged to prepare advert code for execution within the web document at the client so as to display the advert to a user of the client device, at least a portion of the advert code being arranged for averting advertising fraud.

Clause A33. The server of clause A32 further arranged to carry out anti-fraud verification of the request for an advert before transmitting the advert code to the web browser.

Clause A34. The server of clause A32 or A33 wherein the advert code comprises an anti-fraud element at least partly within a protected portion of the advert code, the anti-fraud element being arranged to provide one or more anti-fraud verifications relating to the advert.

Clause A35. The server of clause A34 wherein the anti-fraud element is arranged to provide an anti-fraud verifications of each of one or more of: integrity of the anti-fraud element; that the anti-fraud element is executing within a browser; visibility of the advert in a graphical display of the browser; behavioural characteristics of the user of the browser; and structure of the document object model of at least part of the web document.

Clause A36. A client device comprising a web browser, the web browser comprising: a web document including advert request code, execution of which causes the web browser to transmit an advert request to an advert server; advert code received from the advert server in response to the advert request, the advert code being arranged to present an advert to a user of the client device, wherein the advert code is arranged for averting advertisement fraud.

Clause A37. The client device of clause A36 wherein the received advert code is at least partly in a protected form to avert advertising fraud.

Clause A38. The client device of clause A37 wherein at least some of the received advert code in a protected form is protected by one or more cloaking techniques and/or one or more node locking techniques and/or one or more diversity techniques.

Clause A39. An anti-fraud system arranged for: receiving, from advert request code executing on a browser of a client device, a request for a piece of digital rights management (DRM) protected content; delivering the requested DRM protected content to the client device and providing the client device with DRM rights for the browser to replay the content, the content comprising an identifier for the browser to recover from the content when the content is replayed; and receiving the identifier from the browser.

Clause A40. Advert request code for execution within a web browser on a client device, the advert request code comprising: a content play function, the content play function being arranged to send a content request to a remote server for a piece of DRM protected content, and to receive and cause replay of the content at the client device, the content comprising an identifier; and a request generator arranged to receive the identifier from the replayed content and incorporate the identifier within an advert request for sending to an advert server.

Clause A41. Computer program code arranged to put into effect the method of any of clauses A1 to A31.

Clause A42. One or more computer readable media carrying the computer program code of clause A41.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will now be described, by way of example only, with reference to the drawings of which:

Figure 1 schematically illustrates an example of a computer system for delivering online adverts to web browsers executing on client devices;

Figure 2 illustrates aspects of a client device and an advert server according to the invention;

Figure 3 shows how such a client device and advert server can interact to deliver an advert to the client device;

Figure 4 shows some functional aspects of anti-fraud code executing at the client device;

Figure 5 illustrates functional elements of the advert server;

Figure 6 shows how DRM protected content may be used to provide verification of identity of a client device to which an advert is to be transmitted;

Figure 7 illustrates an approach to advert fraud detection;

Figure 8 shows the process of capturing HTTP packets per process and saving information along with advert blocker decision;

Figure 9 is an example of HTTP request trees of a process. The darker shaded nodes are static contents of a webpage and the "Ad request" and "www.adserver.com" nodes are advert request nodes. The node advertiser.com/index.html is an advert click node;

Figure 10 graphs accuracy of the positive class (recall) of different classifiers;

Figure 11 graphs evaluation results for the detection of advert clicks;

Figure 12 gives examples of false advert clicks in the described system;

Figure 13 graphs improvements of precision for identifying advert clicks after applying filtering. The number of false positive advert clicks is reduced from 69 to 39. Precision is increased to 38.1% from 25.8%;

Figure 14 schematically illustrates a fraudulent scenario concerning a hijacked client device with session hijacking;

Figure 15 schematically illustrates a fraudulent scenario concerning adware traffic;

Figure 16 schematically illustrates a fraudulent scenario concerning botnet fraud without using a browser; and

Figure 17 schematically illustrates a fraudulent scenario concerning botnet fraud with a “bad” publisher.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

In the description that follows and in the figures, certain embodiments of the invention are described. However, it will be appreciated that the invention is not limited to the embodiments that are described and that some embodiments may not include all of the features that are described below. It will be evident, however, that various modifications and changes may be made herein without departing from the broader spirit and scope of the invention as set forth in the appended claims.

Online advertising ecosystem

Figure 1 shows schematically a system within which the inventions described herein may be implemented, in which online adverts are delivered to a browser executing on a client. In figure 1, a plurality of client devices 10 are shown connected to the Internet 17. Each such client device 10 could be a smartphone, a personal computer (laptop, desktop etc.), a tablet computer, a smart TV, or any other suitable client device capable of presenting online adverts to a user through a browser. Each client device executes a web browser 12 which is arranged to download web documents 14 for browsing on the client device. Such web documents 14 are provided by publishing servers 80, which are also connected to the Internet 17, and which transmit web documents to client devices in response to corresponding requests for those web documents from the client devices.

Typically, client devices send requests for web documents to publishing servers using HTTP requests, and the web documents delivered in response are typically HTML documents but frequently also include other elements, such as program code written in JavaScript (JS) or other interpreted languages, executable code, Java applets, Flash code, and so forth. The publishing servers may also deliver other data to a requesting client device such as cookies of various kinds.

It is well known to incorporate adverts within a web document 14 delivered to a client device. However, such online adverts are not usually explicitly provided by the publishing servers 80. Instead, a web document 14 may contain one or more elements of advert request code 16. When a web document 14 containing one or more advert request code elements 16 is executed on a browser 12, execution of an advert request causes the web browser 12 to send a request over the Internet to an advert server 50, which in reply returns, to the browser 12, client side advert code 18 for execution by the browser 12 to present the advert to the user. Thus, the advert code 18 is program code or software executable by the browser 12 to present the advert to the user.

The advert code 18 transmitted by the advert server 50 may itself contain media 20 of the advert such as text, image, sound, video and any other required data, or may cause such media 20 to be subsequently downloaded by the browser from the advert server 50 or another source. The media 20 may be referred to as the “ad creative”. The advert code may typically also contain display parameters 22 such as a preferences or requirements for display of the advert within the browser, for example constraining position and size of display of the advert in a browser frame, duration for the display, whether to use a pop-up box, etc.

The advert code 18 will also typically contain one or more interaction actions 24, which define what is to happen when a user interacts with the advert in one or more ways, and one or more reporting actions 26. An interaction action could for example be for a user to click or tap on the advert and cause consequential navigation by the browser to a web document defined by the interaction action 24, such as the web document of a party with an interest in the advert. Other interaction actions could involve movement of a cursor over the advert, perhaps causing a change in the displayed advert, closing of a pop-up box leading to a further display of information, and so forth.

Reporting actions 26 are used for the system to register the effectiveness of the advert in some way, for example by sending a report to the advert server 50 or some other entity when an interaction action 24 such as a user click or tap on the displayed advert media takes place, or by sending a report when some other suitable event such as partial or complete display of the advert in a visible part of an active window or tab of the browser takes place.

By means of the interaction actions 24 a user can conveniently access further information about goods and services advertised using the online advert. By means of the reporting action 26, the parties involved in the commercial aspects of the online advert can detect when a advert has been effected in some way, for example by being displayed in a visible part of a frame, by being actioned by a user, and/or a variety similar circumstances, and the parties can manage the associated commercial advertising contracts using the reported information.

Also shown in figure 1 are a number of attacker entities 8, also connected to the Internet 17. These attacker entities may interfere in the advert delivery and monitoring process in various ways as already described above, for example appearing to the advert server 50 as if they were *bona fide* client devices which request and receive adverts, controlling *bona fide* client devices in various ways to interfere with the advert processes, interfering in sessions between *bona fide* client devices and advert servers, being used by attackers in click farms or botnet based click fraud, and so forth.

Because the client side advert code 18 typically controls how an advert is handled by a browser, including how it is displayed, and how interaction actions 24 and reporting actions 26 are handled, it is vulnerable to compromise. For example, once an attacker 8 has obtained from the client side advert code 18 information about a reporting action 26, the attacker 8 may be able to fraudulently reproduce the reporting action activity in circumstances where the advert has not been presented to a *bona fide* user (e.g. by reporting user clicks automatically without any human interaction).

Averting fraud in online advertising

Figure 2 illustrates how, in general terms, embodiments of the invention may be used to deliver protected client side advert code 18, for example from an advert server 50 to a client device 10, and how further aspects of the client device and browser can also be protected to make fraudulent activity relating to the advert more difficult to carry out.

The client device 10 shown in figure 2 shows the browser 12 executing within an operating system 28 of the client device. As already shown in figure 1, a web document 14 downloaded to the browser 12 from a publishing server 80 contains advert request code 16 which causes the browser 12 to request the client side advert code 18 from the advert server 50. This request is received by an advert network server module 52 of the advert server, which coordinates with an anti-fraud module 54 and a code protection module 56 of the advert server to return to the client device the client side advert code 18 for execution on the browser 12. The advert network server module 52, the anti-fraud module 54 and the code protection module 56 each also communicate with an anti-fraud database 58 of the advert server 50.

Some or all of the advert code 18 delivered to the browser is in a protected form, and ways in which this protection can be achieved are described in more detail later in this document. Generally however, the protection achieves at least one or more of making the advert code tamper resistant, tamper evident, difficult to read or to extract information from, unique in the sense that repeated versions of the advert code generated on request from browsers are distinguishable from each other and can therefore be identified from each other, and diverse in the sense that repeated versions of the advert code generated on repeated requests are differently structured so as to make an fraud attack difficult to replicate.

In figure 2 the broken line within the advert code 18 shows the extent of the protection implemented using the code protection module 56, demonstrating that in this case the protected advert code includes all of the media 20, display parameters 22, interaction action 24 and reporting action 26 already discussed in connection with figure 1 above, although one or more of these elements could either be omitted from the advert code altogether, or could be outside of the protected portion of the advert code. The advert code 18 also includes an anti-fraud element 30 which provides anti-fraud functionality at the browser 12, and preferably communicates with the anti-fraud module 54 of the advert server 50 as part of this provision. This communication, typically via the advert network server module 52, is preferably implemented to be trusted and secure. As shown in figure 2, the anti-fraud element 30 is also preferably wholly or at least partly within the protected portion of the advert code.

The anti-fraud module 54 provides anti-fraud functionality and support to the advert network server module 52, including by managing the initial protection of advert code 18 before it is delivered to the browser. The protection processes, in terms of obfuscation, diversification, and similar processes which can be applied to protect the advert code, may be carried out wholly or in part by the code protection module 56 of the advert server.

The advert code 18 should be in a form executable in a browser environment. Thus, the anti-fraud element 30 (which forms part of the advert code 18, see figure 2) should also be in a form executable in a browser environment. Typically, the advert code 18 and the anti-fraud element 30 may be written in a combination of JavaScript and HTML. Use of WebAssembly rather than JavaScript would also be an option. Thus, any references to JavaScript throughout this application should not be viewed as limiting, and could instead be read to refer to WebAssembly or other suitable languages as appropriate. The code protection module 56 then provides obfuscation, diversification and other protections to the JavaScript and HTML code, and has access to or contains one or more security JavaScript libraries containing code for the anti-fraud element 30, for example stored in the anti-fraud database 58.

The anti-fraud element 30 contained within the advert code can provide a reasonable level of fraud prevention and detection using conventional software interactions with API calls or similar to a conventional executing browser 12. Thus, the anti-fraud element 30 provides anti-fraud protection at the application layer/level. However, if required an additional browser security module 13 may also be integrated within the browser to provide further fraud prevention and detection functionality, whereby the browser becomes a trusted party in this process for example in interacting and communicating with one or both of the anti-fraud element 30 of the advert code and the anti-fraud module 54 of the advert server 50. Similarly, an additional operating system security module 29 may be integrated within the operating system 28 to thereby become a trusted party in interactions and communications with other elements of the anti-fraud scheme. As example of operating system level security is provided in the Annexe of this application. Trusted / secure communication flows between the anti-fraud element 18 and the browser security module 13, and between the browser security module 13 and the operating system security module 29 are shown in figure 2, but clearly other trusted / secure communication flows can be implemented between the various elements.

Figure 3 shows communications between a publishing server 80, a browser 12 on a client device 10, and an advert server 50, these elements having already been depicted in figure 2, and provides some further detail about process steps and communications between the browser 12 and the advert server 50 which may take place when the advert code 18 is executed by the browser 12.

In step 102 the browser 12 navigates from a previous web document to a new web document 14 which is requested by and received at the browser 12 from a publishing server 80 over the Internet 17. During subsequent execution of the web document 14 received by the browser 12, advert request code 16 contained within the web document 14 is executed causing the browser 12 to send an advert request message 105 for a corresponding client side advert code 18 to the advert server 50 at step 104. At step 106, the anti-fraud module 54 of the advert server 50 carries out verification of the advert request message 105. The verification criteria may include one or more of an ID relating to a publisher of the web document 14, an IP address of the client device 10, and user agent information. Such data may be recorded in the anti-fraud database 58 on the advert server 50 so that if the present advert session is later identified as a fraudulent session (e.g. during the steps of impression verification 116 and/or interaction verification 120), then the relevant data may be blacklisted. This enables the anti-fraud database 58 on the advert server 50 to develop a list of resources being used by attackers. If the advert request message 105 fails this step of verification then some consequential reporting or recording action may be taken by the anti-fraud module 54. If the advert request message 105 passes the step of verification then corresponding advert code 18 is prepared at the advert server 50 at step 108. This step of preparing corresponding advert code 18 may include explicit steps of protection such as obfuscation, diversification and so forth carried out using the code protection module 56 (not shown in this figure), or some or all of these steps may have been carried out in advance, for example if a library of advert codes which are already suitably protected is used.

An advert serial code 109 may also be included in the advert code 18. The advert serial code 109 may be generated or assigned by the anti-fraud module 54 in response to the advert request 105. Furthermore, the advert serial code 109 may be assigned as a unique identifier for the advert request 105 so as to provide a unique identifier for the current protected advert session. In this case, the advert serial code 109 will be unique to the advert code 18 currently being prepared. Subsequently, each time the client device 10 sends a message to the advert server 50 during the current advert session, the advert serial code 109 may be included in the message to enable the anti-fraud module 54 of the advert server 50 to verify that the message is legitimate. Thus, the advert serial code 109 is one element which may enable a secured channel of communication between the client device 10 and the advert server 50. The advert serial code 109 may be included in the advert code 18 in a manner which is difficult for an attacker to read, and/or difficult for an attacker to change. When the advert code 18 has been prepared by the anti-fraud module 54, it is transmitted to the browser at step 110.

During subsequent execution of the advert code 18 by the browser a number of different steps and processes occur which are described in more detail later. However, briefly, at step 112 the anti-fraud element 30 of the advert code 18 carries out an initialisation process. This may involve some communication with the anti-fraud module 54 of the advert server 50.

During subsequent and ongoing execution of the web document 14 the anti-fraud element 30 then repeatedly modifies the structure of the document object model (DOM) of the web document 14, making it more difficult for an attacker to quickly or automatically locate particular elements of the web document 14. This is shown in the figure as a particular step 114, but it should be noted that this process is ongoing, triggered periodically for example by user actions in the browser.

At step 116 the client side anti-fraud element 30 carries out a step of impression verification in which it is verified that the advert has been displayed to the user according to required parameters (for example being in the currently active frame or i-frame and being at least 80% visible). This step may occur as part of the initialisation step 112, immediately after the initialisation step 112, or at other times, and may be repeated periodically, for example until an adequate impression of the advert can be reported. The impression verification 116 can be carried out at the browser 12 and reported to the advert server 50. Alternatively, the impression verification 116 may be carried out in conjunction with the advert server 50, for example by gathering impression verification information at the browser 12, and passing this information for processing to the anti-fraud module 54 of the advert server 50. If either a valid impression of the advert is verified during this step, or fraudulent activity is detected, then the advert server 50 may take suitable impression reporting action 118 and may update the anti-fraud database 58 on the advert server 50 as appropriate.

At step 120 the anti-fraud element 30 carries out interaction verification in which it is verified that a user has interacted with the advert in one or more particular ways according to the interaction action 24, for example by clicking or tapping on an active area of the advert or by passing a cursor over the advert. As for impression verification, this step may be carried out at the browser 12 and reported to the advert server 50, or may be carried out in conjunction with the advert server 50 for example by gathering interaction verification information at the browser, and passing this information for processing to the anti-fraud module 54 of the advert server 50.

With regard to interaction actions 24, functions may be defined in the protected JavaScript code of the advert code 18 in order to do different things based on fired events. For example:

- window.onmousemove=irdetoonmousemove()
- Adurl.onlick=irdetoadclicked()

Such functions may also collect information on browsing behaviours and user interactions. For example, such functions may be used to track mouse movements.

If either a valid user interaction with the advert is verified during this step 120, or fraudulent activity is detected, then the advert server 50 may take suitable interaction reporting action 122 and may update the anti-fraud database 58 on the advert server 50 as appropriate. During this process, the advert server 50 also receives from the browser 12 the advert serial code 109 from the advert code 18, or information derived from the advert serial code 109, which can be compared against the advert serial code 109 as originally included in the advert code 18 by the advert server 50. If the original and returned serial codes or related/derived data do not match then the user interaction verification 120 fails and may be reported as such. Again, the anti-fraud database 58 on the advert server 50 may be updated as appropriate.

In the description that follows the term URL is used for ease of discussion. The skilled person would appreciate, however that when a URL is referred to, in practice this may be a URI or combination of one or more URLs and/or one or more URIs.

In some embodiments of the invention, an interaction action 24 defined in the advert code 18 can only be completed when the advert server 50 returns relevant interaction information and/or code to the browser 12 following a positive result of the interaction verification step 122. In figure 3 this is illustrated by step 124 carried out by the anti-fraud module 54, in which a redirection URL 125 to which the browser 12 should now navigate following a user interaction with the advert is returned to the browser 12. The browser 12 then navigates to the redirection URL at step 126. The redirection URL or other interaction information returned to the browser 12 may be in an obfuscated or otherwise protected form which can only be correctly interpreted by the anti-fraud element 30 of the advert code 18.

Figure 4 illustrates in apparatus form functional aspects of the anti-fraud element 30 of the advert code 18, which may be used to implement the steps outlined in discussion of figure 3 above, along with suitable communication with an advert server 50 where appropriate. The anti-fraud element 30 includes one or more of: an initialisation function 32, a DOM diversification function 34, an impression verification function 36, a deferred action loader 35, and an interaction verification function 38.

The interaction verification function 38 in turn includes a number of sub functions including one or more of: a code integrity verification function 40, a browser verification function 42, a visibility verification function 44, a cursor verification function 46, and a DOM/URL verification function 48. Of course, the functionality provided by these functions and sub functions may be provided in different combinations, further functionality may be added, and some of the described functionality may be omitted.

The initialization function 32 unpacks data relating to the advert from the advert code 18 and inserts it into particular locations in the web document 14. For example advert data fields may include advert serial number or advert serial code 109, publisher ID, advert URL info (creative, tracking pixel, original advert URL, modification information (if any information changes unexpectedly), modified advert URL (if malware modifies advert URL), advert URL location (xpath, as it may change), and DOM change information. In order to provide the code integrity verification functionality 40 discussed in more detail below, the initialisation function 32 is also responsible for recording initial attributes of the advert session from the advert server 50 to enable verification of later values of the same attributes during the advert session. The initial advert session attributes may be recorded in a protected form in the user environment of the client device 10. The initial attributes may include one or more of the advert serial code 109, a hash value (or similar) relating to at least a portion of the advert code 18 (e.g. a hash value of the anti-fraud element 30, or a hash value of an ad tag), a copy of at least a portion of the advert code 18 (e.g. a copy of the anti-fraud element 30, or a copy of an ad tag), and any other desired data relating to the advert code 18.

In order to provide the DOM diversification functionality 34 discussed in more detail below, the initialisation function 32 is also responsible for inserting suitable sub functions at particular positions in the web document 14, for example at particular DOM nodes. In order to provide the interaction verification functionality 38 discussed in more detail below, the initialisation function 32 is also responsible for configuring the web document 14 so that the interaction verification function 38 is called when a suitable user action such as a click or tap on an active region of the advert occurs.

Following suitable setup of the web document 14 by the initialization function 32, the DOM diversification function 34 may be repeatedly triggered, for example by user interaction with the web document 14 such as cursor movements, scrolling, clicks, taps, and/or other actions. The DOM diversification function 34 may be implemented as a single or as multiple sub functions at different positions or nodes in the DOM, and when triggered these DOM diversification sub functions generate changes in the structure of the DOM, for example moving a DOM subtree containing some or all aspects of the advert to a different node or position within the DOM. This dynamic DOM modification is intended to make it more difficult for an attacker to determine a target element of the advert such as an aspect of the interaction action 24 such as a URL to which the browser 12 should navigate when the user interacts with the advert in a predefined manner for example by clicking on the advert. Thus, the DOM diversification function 34 increases tamper resistance of the advert code 18.

The DOM diversification function 34 may implement what will be referred to herein as “**URL hiding**”. This is a form of dynamic DOM modification that is intended to make it more difficult for an attacker to determine a true URL associated with the advert (e.g. a URL from which the advert creative is to be obtained, or a URL to which the browser 12 should navigate when the user interacts with the advert in a predefined manner). Thus, when using URL hiding, an advert is initialised with random attributes such class-name and id, and the advert URL will be a URL known to be fake by the advert server 50. While the fake URL still is alive, if there is a click to it, we can conclude that the http client is a bot. The malicious behaviors by bots will be recorded in the anti-fraud database 58 at the advert server 50 after detecting the fraud case.

In one embodiment of URL hiding, if there is no mouse cursor over the creative of the advert, the URL related to the advert will be a fake one. This guarantees that machines without human activity capability cannot get the true URL since such machines will be programmed to automatically generate an advert click without using a mouse cursor.

In another embodiment of URL hiding, there is a time dependency associated with whether a fake or real URL is used. For example, a fake URL may be used for a time period after the web document 14 is loaded, regardless of whether a mouse cursor is over the creative. This is because very fast clicking after loading is usually associated with bot or human click fraud. In comparison, it generally takes a certain period of time for a normal human user to review an advert creative, then move a mouse to the creative space before interacting with (e.g. clicking) the advert. After the time period has elapsed, the true URL may be provided if all other verifications are positive (e.g. if the mouse cursor is over the creative, as described above). The time period may be a fixed time period based, for example, on the fastest expected time for a legitimate human user to interact with the advert. Alternatively, the time period may be based on other factors such as the stabilisation of a frame rate of the browser 12 after loading the web document 14.

The DOM diversification function 34 may additionally/alternatively make use of a **dynamic advert URL**. The use of a dynamic advert URL may also be considered to be a type of URL hiding. In existing advert networks, the advert URL is only initialized at most once per ad session. In our approach, either the advert server 50 or the anti-fraud element 30 on the client side will periodically update the advert URL using a new identity of the URL. The periodic updates may happen in response to specific user behaviour/interactions, for instance, per mouse movement. Each time the anti-fraud element 30 detects a specific user interaction, an update request message may be sent to the advert server 50. If such a request is sent, the advert server 50 may return a new identity of the advert URL (in protected form) to the client device 10 that will replace the old one. It is optional how often to issue such a update request based on balance between performance and security. When a user clicks on a URL, the related message to the advert server 50 includes the advert URL identity. The advert server 50 keeps track of the advert URLs sent to the client device so that it can identify if a particular click is against the latest advert URL or an outdated advert URL. A click to an outdated advert URL may result in updating the anti-fraud database 58 to blacklist the client device 10. Thus, effectively, the use of a dynamic advert URL may be considered as a form of real-time token updating. In particular, the anti-fraud element 30 receives a token from the advert module 50 periodically to update its internal variable and part of the advert URL. Therefore, even if the hackers find the advert URL in the DOM tree, they still cannot make valid clicks because the advert server 50 and the anti-fraud element 30 on the client device 10 will be synchronized in real-time for each advert session.

To summarise, in URL hiding the advert code 18 comprises an advert element (i.e. an ad tag – see below) including an advert URL associated with the advert. The anti-fraud element 30 may be arranged to replace the advert URL with a fake URL until one or more conditions are satisfied, the fake URL being different from the advert URL. The fake URL may be a real URL associated with a real webpage, or may be a made-up URL with no associated webpage. The one or more conditions may include one or more of: a cursor being present over the advert whilst the advert is being displayed to the user; a fixed time period having elapsed following loading of the web document on the web browser; and a frame rate of the browser having stabilised following loading of the web document on the web browser. The anti-fraud element 30 may additionally/alternatively be arranged to periodically replace the advert URL with an updated advert URL.

Periodically replacing the advert URL may occur at predefined time intervals, or in response to user interactions.

The DOM diversification function 34 may additionally/alternatively implement what will be referred to herein as "**DOM shuffling**". In general, for each advert to be displayed as part of that web document 14, the DOM comprises a respective piece of code that is often referred to as an advert element or "ad tag". An ad tag usually has two parts: a URL from which the browser 12 will request the advert, and some HTML and/or JavaScript code to tell the browser 12 how to display the advert received via the URL request. In DOM shuffling, the DOM diversification function 34 creates multiple copies of a given ad tag. Only one copy (i.e. the true copy) of the ad tag includes the true advert URL from which the browser 12 will request the advert. Each other copy (i.e. fake copies) of the ad tag includes a respective fake URL which is different from the true URL. The multiple copies are intended to be displayed in the same space on the web document 14 (i.e. they have the same HTML and/or JavaScript code telling the browser 12 the size and position of the advert display). However, since all of the adverts are to be displayed in the same place, only one of the copies will be displayed on the top of the stack such that it is visible to a human user. The DOM diversification function 34 ensures that the top (displayed) copy is always the true copy which includes the true advert URL. Thus, a human user will always interact with the true copy and will always request the advert from the true URL. In contrast, an automated clicker is equally likely to click on any of the multiple copies of the ad tag. On receiving any request for a fake URL, the advert server 50 may blacklist the device requesting the fake URL by updating the anti-fraud database 58.

The DOM structure including the multiple copies of a given ad tag may be "shuffled" when there is activity over it. For example, the DOM shuffling may be repeatedly triggered by user interaction with the web document 14 such as cursor movements, scrolling, clicks, taps, and/or other actions. The shuffling involves randomising the locations of the multiple copies of the ad tag within the DOM structure of the web document 14. So, each time a user moves the mouse, the

DOM shuffling dynamically and randomly selects one copy of the ad tag as the true copy, and ensures that this copy is displayed to a human user.

The DOM shuffling functionality may be accomplished in HTML by formulating each ad tag as a <div> and creating copies of the div code associated with the ad tag. To manipulate the multiple divs associated with a given advert, a unique random ID is created for each div. The divs for a given ad tag are sitting under the same node in the DOM such that there is a subtree with all these divs. To hide the redundant (i.e. fake) divs in the subtree, the stack order of the divs is set such that only the div with the true URL is visible at the top of the stack. The DOM diversification function 34 may accomplish this by setting the CSS property of the z-index of these divs so that only the div with the true URL is located at the top of the stack.

Assume a hijacked browser. An attacker may analyse the DOM tree to find an ad tag. However, let there be M-1 duplicated fake ad tags in the DOM subtree. If the attack is only struck once, the attacker can succeed with a probability of 1/M. However, attackers generally conduct fraud repeatedly. Furthermore, we only need to detect one instance of a fake URL click to identify the attacker device as a fraud because the fake URLs will never be clicked by a real human click. Thus, if an attacker strikes repeatedly n times, the attacker can only succeed with a probability of (1/M)^n, which is very low for reasonable M and big n.

To summarise, in DOM shuffling the advert code 18 comprises an advert element (i.e. an ad tag – see below) including an advert URL associated with the advert. The anti-fraud element 30 may be arranged to create copies of the advert element within the DOM of the web document 14, each copy including a respective fake URL different from the advert URL, such that the advert and copies of the advert are displayed in a stack with only the advert being visible to the user at the top of the stack (the copies of the advert are associated with the copies of the advert element). The anti-fraud element 30 may be arranged to periodically alter the DOM structure of the web document 14 so as to randomly rearrange/reorder the advert element and the copies of the advert element within the DOM structure. Notably, “periodically” may mean at regular time intervals or may mean at irregular time intervals. Thus, “periodically” should be interpreted to mean “now and then”. The anti-fraud element 30 may be arranged to periodically alter the DOM structure of the web document 14 in response to user interactions with the web document 14 using the browser 12.

In one embodiment, the DOM diversification function 34 may combine the DOM shuffling and URL hiding functionalities described above. In this case, the initialization function 32 first replaces the true URL in an original ad tag with a fake URL, and then copies the whole ad tag. For each copy of the ad tag, a globally unique ID is created. The z-index of the ad tag copies is then modified so that the advert displayed at the top of the stack is defined as the true advert. Then, in response to a user interaction which brings the mouse over the advert, the DOM diversification function 34 replaces the fake URL in the true advert with the true URL. Subsequently, the DOM diversification function 34 randomly shuffles the relevant DOM subtree. Preferably, the true advert should not be the first child of the parent node.

The DOM diversification function 34 may additionally/alternatively implement what will be referred to herein as “**creative hiding**”. In creative hiding, the true advert creative in an ad tag may be hidden and substituted with a fake advert creative in certain circumstances. This is intended to prevent hijacking of the advert creative itself, rather than to prevent click fraud. The criteria for displaying a true/fake advert creative may include a time dependency (e.g. as for URL hiding – see above). The decision on whether to display a true/fake creative may additionally/alternatively depend on the visibility of the creative on screen. A human user will only interact with a creative that is being displayed to them on screen (e.g. in a visible portion of a web document 14). Thus, it is possible to hide the true creative from possible hijacks by displaying a fake creative until the creative moves on screen, at which point it is necessary to display the true creative to the user.

To summarise, in creative hiding, the advert element may further include an advert creative associated with the advert, and the anti-fraud element 30 may be arranged to replace the advert creative with a fake creative until the advert is displayed to the user, the fake creative being different from the advert creative.

Following suitable setup of the web document 14 by the initialization function 32, execution of the interaction verification function 38 may be triggered by various user actions such as a user click on an active region of the displayed advert. The executing interaction verification function 38 then executes one or more of the interaction verification sub functions as shown in figure 4 before the interaction action defined for user action on the advert, such as navigation to the advertisers web page, is carried out.

A first such sub function is the code integrity verification function 40 which is arranged to verify the integrity of the anti-fraud element 30 and/or any required combination of parts or all of the advert code 18. This ensures that the functionality of the advert code 18, and in particular the correct functioning of the anti-fraud element 30, has not been compromised. Additionally/alternatively to verifying the correct functioning of the anti-fraud element 30, the code integrity verification function 40 may be used to verify the integrity of any ad tags. Such integrity verification can be carried out in various ways. Simple integrity verification checks involve comparing current attributes of the advert code to the initial attributes of the advert code 18 that were stored on the client device as part of the initialisation function 32. If the integrity verification fails then this may be reported to the anti-fraud module 54 in the advert server 50, and/or further execution of the interaction verification function halted. In one example, any modifications to the advert code 18 (e.g. ad tags) may be reported to the advert server 50 by the code integrity verification function 40 of the anti-fraud element 30. In this case, the code integrity verification function 40 may be triggered by modifications of the advert code 18, if desired, rather than being triggered by the various user actions.

A second such sub function is the browser verification function 42 which is arranged to detect and verify that a genuine browser is being used to execute the advert code. This seeks to stop attackers from executing parts of the advert code in another, non-browser environment, which is therefore not presenting the advert to a *bona fide* end user. The browser verification can be carried out in one or more different ways. One method would be to record in the advert code the "user agent" field in the http header of the advert request sent to the advert server, and to detect from the "user agent" field if the agent is not a browser. However, the "user agent" field can easily be spoofed in the advert request by an attacker.

Other methods would be to check the "session per IP" or "visit count".

The browser verification function 42 may also be implemented by detecting features of the client environment which is executing the function. For example, by detecting aspects of the JavaScript capability of the client. One way of doing this is to create cookies in the JavaScript side to check if the client environment does interpret JavaScript - for example a crawler using Curl would not. Other techniques which could be used by the browser verification function 42 to verify the browser could be to use a third part library providing JavaScript driven feature detection such as Modernizr (see http://moderizr.com/), or a DRM based method as discussed below in connection with figure 6.

The browser verification function 42 may be performed in conjunction with the advert server 50 using a DRM methodology described below in the section entitled “Browser verification using DRM”.

If the browser verification fails then this may be reported to the anti-fraud module 54 in the advert server 50, and/or further execution of the interaction verification function halted.

A third sub function of the interaction verification function 38 is the visibility verification function 44. This may typically check that the window, tab, frame or iframe containing the displayed advert is potentially visible to a user, and that the advert media is at least partially visible to the user. To achieve this, the visibility verification function 44 may use the browser’s API to check whether the advert is on a viewable portion of the browser window (i.e. the “viewport”). Most browsers include native page visibility support (a “Page Visibility API”), which can be used to detect whether a current tab is the topmost tab in the browser. However, this support may be limited in working only for multiple tabs within a particular browser window, and may fail to detect if there are other windows covering the browser window, or if the browser is otherwise hidden or not shown by the operating system. Different or additional functionality may therefore be provided to determine visibility, for example a measure of the fraction of the advert that is visible.

If the visibility verification fails then this may be reported to the anti-fraud module 54 in the advert server 50, and/or further execution of the interaction verification function halted.

A fourth sub function is the cursor verification function 46. This seeks to verify that the user of the browser is a *bona fide* human user, for example by making an assessment of cursor (or mouse) movements (or trajectories). These cursor movements could be only movements within the graphical or active area of the advert itself, or more widely within the browser frame, window, or whole display. Thus, the cursor verification function 46 is arranged to track cursor movements and to send to the advert server 50 data relating to cursor movements. The data may first be stored on the client device 10 if desired. The data may include, for example, a time series of cursor coordinates, and optionally the associated time stamps. In one embodiment, the data includes a time series of cursor coordinates ending with a mouse click (or touchscreen press). Thus, in response to a mouse click, a cursor trajectory (in the form of a time series of cursor coordinates) may be sent to the advert server. Determinations of whether the user is human or not can be made using pattern recognition and other techniques. Such determinations will generally be made at the advert server 50, but one or more simple initial determinations could be performed at the client device 10.

If the cursor verification fails then this may be reported to the anti-fraud module 54 in the advert server 50, and/or further execution of the interaction verification function halted.

A fifth sub function is the DOM/URL verification function 48. This sub function may seek to verify that the document object model (DOM) of the web document 14 has not been altered in a manner which suggests fraudulent activity. For example, the sub function may seek to verify that the subtree of the

DOM relating to the advert has not been unexpectedly modified, which could indicate compromise of the advert, or that other parts of the DOM have not been unexpectedly modified, which could indicate malware inserted adverts elsewhere within the web document. The DOM/URL verification function 48 may also seek to verify that the interaction action 24 of the advert code 18 has not been modified, for example that a redirection link for use when the advert is clicked has not been modified.

If the verification fails due to detection of unexpected modification of the DOM or of the interaction action then this may be reported to the anti-fraud module 54 in the advert server 50, and/or further execution of the interaction verification function halted.

Notably, the system can be set up such that failure of only a single verification sub function will lead to detection of fraud (i.e. an invalid click), whereas passing all verification sub functions is necessary to register a legitimate click. This makes it very difficult for an attacker to circumvent the security procedures in place.

In a similar manner to the interaction verification function 38 discussed above, the impression verification function 36 may provide various verification functions which need to be successfully completed before an advert is considered to have been delivered to the user of the browser 12 by way of an impression, whether or not the user subsequently interacts with the advert. Thus, the impression verification function 36 may use some of the same sub functions as the interaction verification function 38. For example, current online advertising practices tend to count an impression once the JavaScript that creates the advert (i.e. the advert code 18) is loaded by the browser 12, no matter whether the advert creative can be seen by human beings. However, it is desirable for the impression verification function 36 to check the viewability of the advert creative in real-time. This may be done using the functionality of the visibility verification function 44. If the advert creative is viewable, the verification performed by the impression verification function 36 will be positive and an impression will be counted. Otherwise, it will not be considered as a valid impression.

Alternatively/additionally, the impression verification function 36 may check whether the advert creative is being displayed on a topmost tab within the browser 12. If the advert creative is detected to be on the topmost tab, the verification performed by the impression verification function 36 will be positive and an impression will be counted. Otherwise, it will not be considered as a valid impression. If both viewability and topmost tab are checked, then a valid impression will only be counted when the advert creative is both viewable and on the topmost tab.

The anti-fraud element 30 may include a deferred action loader 35. In this option, the interaction action 24 of the advert code 18 remains incomplete and therefore not fully functional until particular conditions are satisfied. For example, the deferred action loader 35 may wait until the anti-fraud element 30 detects visibility of the advert on the user device, or detects that the frame of the advert is the currently active frame, or detects that a cursor is over or proximal to the advert, and only then modify the interaction action 24 to render it functional for example to include a URL to which the browser should navigate when the user interacts with the advert in a predefined manner, for example by clicking on the advert. The deferred action loader 35 may then render the interaction action 24 again incomplete and not fully functional when the particular conditions are no longer satisfied.

The conditions to be satisfied before the deferred action loader 35 renders the interaction action functional could instead be one or more of: that a particular user action has occurred such as a click on or selection of the advert; that such an action has occurred and some or all of the verification sub functions of the interaction verification function 38 have been successfully completed; that data required to complete the interaction action, such as a redirection URL, has been received from the advert server 50 following such verification, that a verification conformation has been received from the advert server 50,and so forth, in various combinations.

Figure 5 illustrates in more detail functional parts of the advert server 50 of figures 2 and 3. The anti-fraud module 54 is shown as comprising a request handler 60, an interaction handler 62, an impression handler 64, and a security manager 66. The anti-fraud database 58 is shown as comprising diversified anti-fraud elements 68, anti-fraud policy 70, publisher data 72, user data 74, track data 76, and base advert code 78.

The request handler 60 carries out the step 106 of verifying an advert request from a browser, already discussed in connection with figure 3. If the advert request is suitably verified then the security manager 66 carries out the step 108 of preparing a suitable advert code 18, including use of the code protection module 56 to provide protection of the advert code 18, and arranging transmission of the advert code 18 to the requesting browser 12.

The request verification can include various different verification processes and steps. For example, the request handler 60 may check whether the advert request has been received from a client device or other origin corresponds to an origin which is recorded in the anti-fraud database 58 as being, or not being associated with a fraud risk. In particular embodiments, the origin may be an IP address or domain from which the advert request 105 appears to have been transmitted. If the IP address or domain is that of a publishing server 80, then this may indicate a server based fraud scenario and the request handler 60 then causes this IP address to be marked as a fraud device, for example in the database element shown in figure 5 as publisher data 72, and may also cause the current advert request and future advert requests from this IP address to be rejected.

The request handler 60 may also use various statistical techniques to decide whether an advert request should be rejected.

In carrying out the step 108 of preparing an advert code 18, the security manager 66 may receive a base advert code 78 from the anti-fraud database 58 or elsewhere. The base advert code 78 will typically have been provided by an advertiser, defining the media 20, display parameters 22, interaction action(s) 24 and reporting action(s) 26. Typically, the base advert code 78 will have been constructed in a combination of HTML and JavaScript. The security manager 66 then causes the base advert code 78 to be protected by the code protection module 56 using techniques described in more detail below, and integrates the anti-fraud element 30 with the base advert code to thereby provide the client side advert code 18. The protection may be carried either one or both of before and after integration with the anti-fraud element 30.

The security manager 66 may also generate the advert serial code 109 (depicted in figure 3, which may be unique for the particular advert request 105), and combine this advert serial code 109 into the advert code 18. The advert serial code 109 may be combined with the advert code 18 in a protected or obfuscated manner, so that it is at least difficult for an attacker to tamper with, read or otherwise compromise.

The security manager 66 also provides the advert code 18 with a suitable entry point for invocation of the initialisation function 32 for execution at step 112 when the advert code 18 has been downloaded to the browser 12.

The anti-fraud element 30 may be obtained by the security manager 66 in a form ready to integrate into the advert code 18 from a store of pre-prepared diversified anti-fraud elements 68 in the database 58, or diversification of the advert code 18 may be carried out during the step 108 of preparing the advert code 18.

The interaction handler 62 interacts with the interaction verification function 38 of the client side advert code anti-fraud element in order to support completion of the interaction verification step 120 of figure 3 in which a user interaction with an advert (such as clicking on an advert) is subject to various verification processes before being accepted. The interaction handler 62 also carries out the step of interaction reporting 122 shown in figure 3 which may include reporting a failure of the interaction verification or a detection of fraud, for example by writing corresponding data into the anti-fraud database 58, or the step 124 of returning suitable data to the browser 12 to enable an interaction action to be completed, for example by returning a suitable advert URL for the browser to navigate to. Similarly, the impression handler 64 of figure 5 interacts with the impression verification function 36 of the client side advert code anti-fraud element 30 in order to support completion of the impression verification step 116 of figure 3, and completes the step of impression reporting 118 if required, which may include reporting a failure of the impression verification or a detection of fraud, for example for recording in the database 58.

In more detail, the interaction handler 62 may receive, from the anti-fraud element 30 of the browser 12, the advert serial code 109 which was included in the advert code 18 before sending to the browser 12, or information derived from the advert serial code 109. If the received and original codes do not match, then the interaction verification step 120 fails, and the interaction reporting step 122 may carry out suitable reporting of this failure. Furthermore, one or more details of the current advert session (e.g. an IP address of the client device 10) may be recorded in the anti-fraud database 58 and blacklisted to indicate fraudulent activity for future reference.

In some embodiments, the advert code 18 delivered to the browser 12 already includes a redirection URL (forming part of the interaction action 24) to which the browser 12 should navigate if the user interacts with the advert in a defined way for example by clicking on the advert. This redirection URL may be passed to the interaction handler 62 as part of the step 120 of interaction verification. The interaction handler 62 may then check if the redirection URL received in step 120 matches the redirection URL originally included in the advert code in step 108. If the two do not match then the verification fails, and the returned redirection URL or related data identifying the URL such as a corresponding IP address may be recorded in the database 58, for example in the publisher data 72, for future reference.

The interaction handler 62 may also receive, from the anti-fraud element 30 executing in the browser 12, a result of fraud detection carried out by the anti-fraud element 30 in the browser 12 as part of the interaction verification step 120. If this result indicates fraud then the verification step 120 fails and the interaction reporting step 122 may carry out suitable reporting of this failure. For example, the fraud detection carried out in the browser 12 may include executing one or more of the sub functions shown in figure 4 as code integrity verification 40, browser verification 42, visibility verification 44, cursor verification 46 and DOM/URL verification 48. However, one of more of these functions may be at least partly carried out by the interaction handler 62, for example on the basis of data provided by the relevant sub functions executing in the browser 12. Elements of such client return data, such as mouse movement and track data, may also be stored in the anti-fraud database 58, for example as user data 74 and track data 76.

As described previously with respect to figure 4, the client-side cursor verification function 46 sends data relating to cursor movements to the advert server 50. This data is received and processed by the interaction handler 62. In one embodiment, the data comprises a cursor trajectory (i.e. a time series of cursor coordinates). In one embodiment, received cursor trajectory ends with a mouse click (or touchscreen press). In general, when a human user moves a mouse from a start point to the point of clicking a clickable webpage object, the resulting cursor trajectory should be unique due to the randomness of the start point and due to an individual user's own way of moving the mouse. In other words, detection of the same or similar cursor trajectories at different times while associated to a different click on the same user environment is very likely to be an indication of automated malware (i.e. an invalid click). On receiving data relating to cursor movements (e.g. a cursor trajectory relating to a mouse click) from the client-side cursor verification function 46, the current trajectory is stored as track data 76 in the anti-fraud database 58. In addition, the current trajectory is compared against previous trajectories in the anti-fraud database 58. If there is an identical or similar previous trajectory, then this is an indication of fraud. In one embodiment, not all previous trajectories in the anti-fraud database 58 are used in the comparison. Instead, a subset of previous trajectories is used for comparison to the current trajectory. In a simple embodiment, the subset of trajectories used for the comparison comprises the trajectories in which the click pixel (i.e. the pixel in which the mouse click occurred) is the same pixel as the click pixel for the current trajectory. The comparison may include any mathematical method of determining the similarity between two trajectories. For example, each time series of cursor coordinates may first be interpolated to provide a continuous trajectory in space and time. Then, the trajectories may be compared by calculating the geometric distances between the current trajectory and each of the subset of trajectories. If the average geometric distance is greater than a threshold, then the click may be considered to be valid, else the click may be considered to be invalid and a fraud may be reported. However, it will be understood that alternative “comparison” methodologies may be used with or without calculation of geometric distances between trajectories.

The interaction handler 62 may be adapted to detect if the anti-fraud element 30 and/or other parts of an advert code 18 have been compromised, for example through circumvention of the protection applied by the code protection module 56 and so forth. If compromise is detected then the security manager 66 and request handler 60 may then be operated to take suitable action.

Operation of the arrangements described above under various different scenarios of fraud and attempted compromise by one or more attackers will now be described.

Scenario 1 - hijacked client device with session hijacking

This scenario is schematically illustrated in figure 14. Under this scenario, an otherwise valid browser session between the browser of a client device and a server such as a publishing server 80 is hijacked, for example by malware running on the same client device which obtains one or more session keys to thereby participate in the browser session. This enables the hijacker malware to make additional advert or other HTTP requests which have then been made by the legitimate browser 12, but are actually independent of content genuinely requested by the user.

In this scenario, the ongoing step 114 of figure 3 in which the document object model (DOM) is periodically modified makes it much more difficult for the hijacker to make effective changes to the web document 14, since the effective target location of such a change will not be predictable. If the user of the client device 10 clicks on an advert, or otherwise interacts with an advert, then as part of the interaction verification step 120 of figure 3, the DOM/URL verification function 48 of the client side code anti-fraud element 30 can detect any fraudulent changes to the DOM or modifications to the interaction action 24 of the advert code 18. Such changes or modifications could cause the interaction verification function to fail verification, so that the changes made by the malware code are blocked, or could cause such changes to be reversed or removed.

During operation of the interaction handler 62 of the advert server 50 as part of the interaction verification step 120, the interaction handler 62 may check any redirection URL received as part of the step from the browser 12. If this does not match the correct redirection URL then this may cause the verification step to fail. Changes to the redirection URL made at the browser 12 by the malware are therefore blocked, or the interaction handler 62 could cause such changes to be reversed or removed. A redirection URL detected as added by a hijacker or malware could also be marked as the URL or domain of a "bad" publisher or similar in the publisher data 72 of the anti-fraud database 58 or elsewhere.

The interaction verification may also fail if the advert serial code 109 or related data returned as part of the interaction verification fails to match the advert serial code 109 originally added to the advert code 18 by the advert server 50, or inconsistencies or changes in other data relating to the advert are detected.

Scenario 2 - adware traffic

This scenario is schematically illustrated in figure 15. Under this scenario a normal user is operating the browser on a client device, but additional html or advert requests are made by elements of advert request code 16 included in a web document by a publisher 80 (or other adware), independently of content genuinely requested by the user. In this way, the advert request code elements 16 of the web document 14 can be used by an attacker / bad publisher to add or change adverts in the web documents viewed by the user, and to add or modify redirections and browser navigations to web documents which are preferred by the publisher 80 or other party modifying the adware code.

Additionally to the measures described in scenario 1 above, the request handler 60 of the anti-fraud module 54 at the advert server may detect if the publisher server 80 which supplied the current web document 14 is recorded as a bad publisher in the anti-fraud database 58, and if so then can decline to process or respond to an advert request 105.

Scenario 3 - botnet fraud without using browser

This scenario is schematically illustrated in figure 16. Under this scenario, a third party process, acting as a botnet but not from within a browser context, sends messages to the advert server 50 which are intended to be interpreted by the advert server as genuine user advert requests 105, and then sends further messages to the advert server 50 which are intended to be interpreted by the advert server as resulting from genuine user interactions with an advert.

In this scenario, various sub functions of the interaction verification function 38 will detect failures in verification during the interaction verification step 120. For example, the browser verification function 42 will detect a lack of browser characteristics in the process sending messages to the advert server 50.

Scenario 4 - botnet fraud with bad publisher

This scenario is schematically illustrated in figure 17. Under this scenario, a third party process, acting as a botnet operating within a browser context, sends messages to an advert server 50 which are intended to be interpreted as genuine user advert requests 105, and then also sends messages which are intended to be interpreted as arising from genuine user interactions with an advert. The fraud might be initiated by a publisher 80, for example by including compromised advert request code 16 in the web document 12, or the publisher 80 might not be involved in the fraud.

As for scenario 2, the request handler 60 of the anti-fraud module 54 at the advert server may detect if the publisher server 80 which supplied the current web document 14 is recorded as a bad publisher in the anti-fraud database 58, and if so then can decline to answer an advert request 105, and as for scenarios 1 and 2, the ongoing step 114 of figure 3 in which the document object model (DOM) is modified makes it much more difficult for the attacking party to make effective changes to the web document. Additionally, the various parts of the interaction verification process 120 act as already described in respect of scenarios 1 and 2.

In particular in this scenario, however, the cursor verification subfunction 46 of the verification function 38 aims to detect if the interaction action was not accompanied or preceded by cursor movements or similar user behaviour indicating a human user of the browser, leading to a failure of the verification step.

Three categories of anti-fraud functions

In summary, we envisage three categories of anti-fraud functions which may be referred to as (a) front-end anti-fraud functions, (b) just-in-time (JIT) anti-fraud and tracking functions, and (c) back-end anti-fraud functions.

The advert server 50 performs some front-end anti-fraud functions. In particular, in the step 106, the request handler 60 uses historical data stored in the anti-fraud database 58 to verify whether an advert request 105 is valid. This also includes verifying the user and publisher associated with the advert request 105. Notably, the historical data have been collected by running just-in-time and back-end anti-fraud functions during the processing of previous advert requests. In addition, the security manager 66 of the advert server 50 performs front-end anti-fraud functions during the step 108 by applying HTML/JavaScript protections to the advert content, and by creating protected advert code 18 for delivery to a client device10 in response to the advert request. In particular, as previously described, the advert server 50 combines the advert content with a client-side anti-fraud element 30 for delivery in real time to the user environment.

The client device 10 performs some front-end anti-fraud functions too. In particular, the client device 10 uses the received anti-fraud element 30 to initialise and set-up a protected advert session with an initial state of anti-fraud protection of the advert content (see the step 112 in figure 3 and the initialisation function 32 in figure 4). The protected advert session is established and the anti-fraud user environment is up before there are any user actions to the advert content.

Just-in-time (JIT) anti-fraud and tracking functions refer to functions performed in response to user interactions with the browser. For example, JIT anti-fraud and tracking functions may be performed in response to movement of a cursor over the advert creative, or clicking on the advert creative.

The client device 10 performs some just-in-time anti-fraud and tracking functions. In particular, the client device 10 performs data collection and security enhancements just-in-time when the user makes any action (e.g. mouse movement and click) in response to currently displayed advert content. This is best illustrated by the interaction verification step 120 of figure 3 which performs the various interaction verification functions 38 shown in figure 4. The JIT anti-fraud functions on client side collect the latest values of essential attributes of the protected advert code 18 and information on user behaviour and user environment (such as browser, window and webpage). This JIT data is then transferred to the advert server 50 for its peer function to process them.

Some just-in-time anti-fraud and tracking functions are also performed by the advert server 50. In particular, the advert server 50 performs a series of just-in-time verifications based on JIT data collected from the client environment. This is best illustrated by the interaction handler 62 of figure 5. The advert server 50 checks JIT data received from the client device 10 against original values of the corresponding attributes pre-recorded for the current advert content and session, user behaviour, and user environment. If there is any difference, this may imply that these attributes have been modified without authorisation, which leads to the conclusion that there has been possible hijacking of the user environment or advert session, or possible tampering with the advert content. The advert server 50 also handles URL dynamics and updates JIT data to the anti-fraud database 58 for the current advert session. Because such just-in-time checks are performed on the server side, it is impossible for malware on the client side to understand and attack such verification and handling. Also, doing these just-in-time checks on the server side can reduce the security computation load from the client side.

The advert server 50 also performs some back-end anti-fraud functions. In particular, the advert server 50 combines both this advert session data and historical data, performs click verification and accordingly updates the anti-fraud database 58 for future verification.

Protecting the advert code

Various ways in which the advert code 18 may be protected before delivery from the advert server 50 to the browser 12 will now be described. As mentioned above, the protection achieves at least one or more of making the advert code 18 tamper resistant, tamper evident, difficult to read or to extract information from, unique in the sense that repeated versions of the advert code generated on request from browsers are distinguishable from each other and can therefore be identified, and diverse in the sense that repeated versions of the advert code generated on repeated requests are different so as to make an fraud attack difficult to replicate.

Some or all of the advert code may also be optimized in various ways, and the parts which are protected may or may not overlap with or be the same as the parts which are protected. Some ways in which items of software such as the presently discussed advert code may be protected and/or optimized are described in WO2015/150376, for example using intermediate representation techniques and/or unified security framework technology described therein. This document is therefore incorporated by reference herein in its entirety for all purposes, including to illustrate ways in which protection may be provided to the advert code.

Some or all of the advert code may also or instead be protected and/or provided using techniques described in WO2015/150391, which document is incorporated by reference herein in its entirety for all purposes. These documents illustrate ways in which protection described herein may be provided to the advert code or other software elements, for example comprising applying one or more white-box, node-locking or other protection techniques to at least a portion of the advert code or other software elements when these are provided in a scripted language, interpreted language, or in the form of source code.

Protection of the advert code, however so achieved, may be implemented wholly or in part by the code protection module 56 of figure 5, or may be implemented partly before the action of the code protection module 56, for example by storing diversified and/or partly protected anti-fraud elements in the anti-fraud database 58 of the advert server 50.

The aim of applying protection to the advert code 18 is to protect the functionality or data processing aspects of the advert code 18, and/or to protect data used or processed by the advert code 18. This can be achieved by applying cloaking techniques such as homomorphic data transformation, control flow transformation, white box cryptography, key hiding, program interlocking and boundary blending.

In particular, the advert code 18 after processing by the code protection module 56 will provide the same functionality or data processing as before such processing – however, this functionality or data processing is typically implemented in the advert code 18 in a manner such that a compromised client device 10, a botnet, or a hijacker of an HTML session between an advert server 50 and a client device 10 cannot access or use this functionality or data processing from the advert code 18 in an unintended or unauthorised manner (whereas if the client device 10 were provided with the advert code 18 in an unprotected form, then the a session hijacker or operator of the client device might have been able to access or use the functionality or data processing in an unintended or unauthorised manner).

Similarly, the advert code 18, after processing by the code protection module 56, may store secret information (such as a cryptographic key or an advert serial code 109) in a protected or obfuscated manner to thereby make it more difficult (if not impossible) for an attacker to deduce or access that secret information (whereas if a client device 10 were provided with the advert code 18 in an unprotected form, then the operator of the client device 10 might have been able to deduce or access that secret information).

For example:

The advert code 18 may comprise a decision (or a decision block or a branch point) that is based, at least in part, on one or more items of data to be processed by the advert code 18. If the advert code 18 were provided to a client device 10 in an unprotected form, then an attacker may be able to force the client device 10 to execute so that a path of execution is followed after processing the decision even though that path of execution were not meant to have been followed. For example, the decision may comprise testing whether a program variable B is TRUE or FALSE, and the advert code 18 may be arranged so that, if the decision identifies that B is TRUE then execution path PT is followed/executed whereas if the decision identifies that B is FALSE then execution path PF is followed/executed. In this case, the attacker could (for example by using a debugger) force the advert code 18 to follow path PF if the decision identified that B is TRUE and/or force the advert code 18 to follow path PT if the decision identified that B is FALSE. Therefore, in some embodiments, the code protection module 56 aims to prevent (or at least make it more difficult) for the attacker to do this by applying one or more software protection techniques to the decision within the advert code 18.

The advert code 18 may comprise one or more of a security-related function; an access-control function; a cryptographic function; and a rights-management function. Such functions often involve the use of secret data, such as one or more cryptographic keys or serial codes. The processing may involve using and/or operating on or with one or more cryptographic keys. If an attacker were able to identify or determine the secret data, then a security breach has occurred and control or management of data (such as audio and/or video content) that is protected by the secret data may be circumvented. Therefore, in some embodiments, the code protection module 56 aims to prevent (or at least make it more difficult) for the attacker to identify or determine the one or more pieces of secret data by applying one or more software protection techniques to such functions within the advert code 18.

A “white-box” environment is an execution environment for an item of software in which an attacker of the item of software is assumed to have full access to, and visibility of, the data being operated on (including intermediate values), memory contents and execution/process flow of the item of software. Moreover, in the white-box environment, the attacker is assumed to be able to modify the data being operated on, the memory contents and the execution/process flow of the item of software, for example by using a debugger – in this way, the attacker can experiment on, and try to manipulate the operation of, the item of software, with the aim of circumventing initially intended functionality and/or identifying secret information and/or for other purposes. Indeed, one may even assume that the attacker is aware of the underlying algorithm being performed by the item of software. However, the item of software may need to use secret information (e.g. one or more cryptographic keys or serial codes), where this information needs to remain hidden from the attacker. Similarly, it would be desirable to prevent the attacker from modifying the execution/control flow of the item of software, for example preventing the attacker forcing the item of software to take one execution path after a decision block instead of a legitimate execution path. There are numerous techniques, referred to herein as “white-box obfuscation techniques”, for transforming the advert code 18 so that it is resistant to white-box attacks. Examples of such white-box obfuscation techniques can be found, in *“White-Box Cryptography and an AES Implementation”*, S. Chow et al, Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, Lecture Notes in Computer Science 2595 (2003), p250-270 and *“A White-box DES Implementation for DRM Applications”*, S. Chow et al, Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Lecture Notes in Computer Science 2696 (2003), p1-15, the entire disclosures of which are incorporated herein by reference. Additional examples can be found in US61/055,694 and WO2009/140774, the entire disclosures of which are incorporated herein by reference. Some white-box obfuscation techniques implement data flow obfuscation – see, for example, US7,350,085, US7,397,916, US6,594,761 and US6,842,862, the entire disclosures of which are incorporated herein by reference . Some white-box obfuscation techniques implement control flow obfuscation – see, for example, US6,779,114, US6,594,761 and US6,842,862 the entire disclosures of which are incorporated herein by reference. However, it will be appreciated that other white-box obfuscation techniques exist and that embodiments of the invention may use any white-box obfuscation techniques.

As another example, it is possible that the advert code 18 may be intended to be provided (or distributed) to, and used by, a particular client device 10 (or a particular set of client devices 10) and that it is, therefore, desirable to “lock” the advert code 18 to the particular client device(s) 10, i.e. to prevent the advert code 18 from executing on another client device 10. Consequently, there are numerous techniques, referred to herein as “node-locking” protection techniques, for transforming the advert code 18 so that the protected advert code 18 can execute on (or be executed by) one or more predetermined/specific client devices 10 but will not execute on other client devices. Examples of such node-locking techniques can be found in WO2012/126077, the entire disclosure of which are incorporated herein by reference. However, it will be appreciated that other node-locking techniques exist and that embodiments of the invention may use any node-locking techniques.

Digital watermarking is a well-known technology. In particular, digital watermarking involves modifying an initial digital object to produce a watermarked digital object. The modifications are made so as to embed or hide particular data (referred to as payload data) into the initial digital object. The payload data may, for example, comprise data identifying ownership rights or other rights information for the digital object. The payload data may identify the (intended) recipient of the watermarked digital object, in which case the payload data is referred to as a digital fingerprint – such digital watermarking can be used to help trace the origin of unauthorised copies of the digital object. Digital watermarking can be applied to items of software. Examples of such software watermarking techniques can be found in US7,395,433, the entire disclosure of which are incorporated herein by reference. However, it will be appreciated that other software watermarking techniques exist and that embodiments of the invention may use any software watermarking techniques.

It may be desirable to provide different versions of the advert code 18 to different client devices 10. The different versions of the advert code 18 provide the different client devices 10 with the same functionality – however, the different versions of the protected advert code 18 are programmed or implemented differently. This helps limit the impact of an attacker successfully attacking the protected advert code 18. In particular, if an attacker successfully attacks his version of the protected advert code 18, then that attack (or data, such as cryptographic keys, discovered or accessed by that attack) may not be suitable for use with different versions of the protected advert code 18. Consequently, there are numerous techniques, referred to herein as “diversity” techniques, for transforming the advert code 18 so that different, protected versions of the advert code 18 are generated (i.e. so that “diversity” is introduced). Examples of such diversity techniques can be found in WO2011/120123, the entire disclosure of which are incorporated herein by reference. However, it will be appreciated that other diversity techniques exist and that embodiments of the invention may use any diversity techniques. Furthermore, as described above in connection with figure 5, such diversity techniques may be implemented in advance to provide a set of diversified copies of whole or parts of the advert code 18 (for example diversified copies of the anti-fraud element 30 as shown stored in anti-fraud database 58), before further protection techniques are applied for example by the code protection module 56.

The above-mentioned white-box obfuscation techniques, node-locking techniques, software watermarking techniques and diversity techniques are examples of software protection techniques. It will be appreciated that there are other methods of applying protection to an advert code 18. Thus, the expression "protection" as used herein shall be taken to mean any method of applying protection to an advert code 18 or other piece of software (with the aim of thwarting attacks by an attacker, or at least making it more difficult for an attacker to be successful with his attacks), such as any one of the above-mentioned white-box obfuscation techniques and/or any one of the above-mentioned node-locking techniques and/or any one of the above-mentioned software watermarking techniques and/or any one of the above-mentioned diversity techniques.

There are numerous ways in which the code protection module 56 and/or other elements may implement the above-mentioned software protection techniques within the advert code 18. For example, to protect the advert code 18, the code protection module 56 may modify one or more portions of code within the advert code 18 and/or may add or introduce one or more new portions of code into the advert code 18. The actual way in which these modifications are made or the actual way in which the new portions of code are written can, of course, vary – there are, after all, numerous ways of writing software to achieve the same functionality.

Browser verification using DRM

Referring now to figure 6 there are shown further aspects of ways in which online adverts can be delivered to a browser 12 executing on a client device 10 in a system such as that shown in figure 1. These further techniques can beneficially be used in combination with the other aspects already described above. As in figure 2, advert request code 16 is included in a web document 14 downloaded from a publisher server 80. However, in this case the advert request code 16 includes extra functionality to assist the advert server 50 or another external party in identifying the browser 12 executing the advert request code, and/or the client device 10 on which the browser is executing.

In particular, the advert request code 16 of figure 6 includes a content play function 210 which is arranged to send a content request 212 to a remote server for a piece of content 214 such as a video fragment or still image, and to replay the content at the browser 12 when received. The content request 212 may, for example, be an HTTP request made to a remote media server 220. The media server 220 could form a part of the advert server 50 of figure 2, but could instead be a separate entity.

The media server 220 obtains the requested content, for example from a media database 222, and returns the requested content 214 to the browser 12, in a DRM (digital rights management) protected form. This DRM protection may be implemented in a variety of ways familiar to the skilled person, but may typically involve the media server 220 negotiating DRM rights protection for the content with a collocated or separate DRM server 224. Typically, the returned content 214 will be transmitted in an encrypted or otherwise obfuscated form, requiring one or more keys under the DRM scheme in order for the content to be replayed by the browser 12.

The browser 12 includes or has access to a browser DRM function 230 which enables the browser to replay the DRM protected content, for example by negotiating access to the one or more keys from the DRM server 224. The content 214 is then replayed by the browser 12 on a display 15 of the client device 10.

The content 214 delivered in a DRM protected form to the browser 12 further includes an identifier 216 which can be extracted from the content when replayed on the display 15. The identifier 216 could, for example, be provided to the media server for incorporation in the content 214 by the security manager 66 of the advert server 50 discussed in connection with figure 5.

The advert request code 16 then further includes a request generator 218 which recovers the identifier 216 from the content 214 as displayed by the browser 12 on a display 15 of the client device, and incorporates the identifier 216 or other relevant data derived from the identifier 216 into an advert request 105 sent to the request handler 60 of the advert server 50 as already described above. The request handler 60 also has access to the identifier 216 or data derived form the identifier, for example by communication from the security manager 66, and so is able to determine or verify that the advert request 105 was indeed generated by a particular browser 12. Thus, this process enables the advert server 50 to perform browser verification.

Optionally depending on other factors, such as whether or not the request handler 60 of figure 5 determines that the advert request is to be fulfilled, advert code 18 such as that described elsewhere herein may then be transmitted to the browser 12 for incorporation and execution within the current web document 14.

The request generator 218 need not include the identifier in the advert request 105 in plain text or in a readable form, but could obfuscate the identifier in various ways, for example by encryption, hashing, etc.. The identifier 216 could be incorporated in the content 214 in various ways, for example by means of a watermark, or specific pixels distributed in a predetermined way in time and/or space, and could be recovered from the displayed content by the request generator 218 in various ways for example by reading particular pixel values from a display buffer of the client device and hashing together values of those pixels to regenerate the identifier 216 or data derived from the identifier.

It may be desirable for the display of the content 214 to be carried out on the browser in a manner which does not interfere with the user experience. For example, the content could be displayed briefly in a small and relatively unimportant part of the display 15, or could be displayed briefly within the viewport or other display area in which a subsequent advert is to be displayed shortly thereafter.

The identifier 216 could be unique to a particular client device, or to a particular group of client devices. It can be seen that by using the scheme of figure 6, return of the correct identifier 216 or related data makes use of the secure nature of the browser DRM function 230 and its secure communication with the DRM server 222. The scheme may be further secured by obfuscated inclusion of the identifier 216 in the content, for example through watermarking. By reading the identifier 216 or related data from the screen buffer or another low level display aspect of the client device 10, the scheme also protects against advert fraud by requiring the content to actually be displayed on the client device 10 before the identifier 216 can be recovered and used to form a valid advert request 105. Fraudulent advert requests could be identified by noting that advert requests including the same identifier 216 were apparently being made a large number of times from a particular client device, for example as identified by the IP address and/or DRM identity of the device.

In figure 6 the media server 220, the media database 222 and the DRM server 222 are illustrated as being located within the advert server 50, but of course they could be separately implemented in a common or distant locations. Similarly, the scheme of figure 6 has been described as making use of the security manager 66 and request handler 60 already discussed above in connection with figure 5 and elsewhere, but could of course be implemented in other contexts and using different server elements.

O/S level fraud detection

Ad networks mostly use server-based techniques to detect advertising frauds as they do not have enough control over the client machines. They gather information from different sources about the user, the machine and the behaviour of the user. Then, they apply machine learning or pattern recognition techniques to identify suspicious impressions or clicks. However, botnets that control real PCs avoid detection by imitating human behaviour and limiting the number of click per day per bot. If a botnet controls thousands of computers, then it manages to click thousands of adverts even if one bot clicks only one advert per day. Novel approaches are presented herein to address this problem from the client-side. The conjecture is that a better performance can be achieved if we leverage the information that is only available to the operating system of a client machine.

In one example, we capture HTTP packets from all the running processes for a specific time-frame each day along with mouse events from all hardware mouse devices. In order to identify advert clicks from the captured raw HTTP packets the following may be done in this example. We first construct HTTP request trees to represent a causal relationship between different HTTP requests and responses. Then, we create features from HTTP headers, query parameters and properties of the subtree of a node to classify advert request nodes using machine learning classification. We may also use the decision of a popular advert block filter as one of our features. Finally, based on our observations, we create rules to identify advert clicks in the request tree. We also populate the tree with real mouse click information. We detect all fraudulent processes that programmatically click on adverts as they cannot generate real mouse click. They either make a new HTTP request to simulate click on an advert or generate software simulated clicks which is distinguishable from real mouse clicks. A potential limitation though, is that our approach does not detect fraudulent processes if they somehow interact with a real human using the mouse. In this particular case, the same process is shared by the malware and the human and the process receives both real and fraudulent clicks.

Over the last decade, researchers from big companies like Google and Yahoo! are doing much work to detect click fraud. Since advertising revenue is the main source of income of these companies and many independent publishers, click fraud became a major threat to the survival of free content on the web. Advertisers also have strong interest to combat all sorts of advertising frauds that are draining their money. Examples of techniques used in malicious clickbots are set out in the “Background of the invention”. One major limitation of all these clickbots is, they either generate software simulated mouse clicks or generate HTTP request to the advert link without any mouse click. Our approach uses this limitation of the bots to identify them.

Examples of click fraud detection are set out in the “Background of the invention”. In the field of click fraud, most research focuses on the detection from the server-side. In industry, either the advert networks or the advertisers use global advertising web traffic and their own proprietary technology to detect fraudulent clicks. Most of these techniques involve data mining or pattern recognition which are not very effective against large-scale low-noise botnets that imitate human browsing behaviour. In our understanding, click fraud is a client-side behaviour. Therefore, our approach focuses on client-side click fraud detection.

There is provided a method of operating a client device, the method comprising: monitoring advertising page visits by running processes and identifying whether said visits are preceded by real mouse clicks. In this way a process attempting to make automated clicks on online advertisements may be detected. Typically such a process will then be reported to the user and/or marked for further action. As shown in figure 7, in one example HTTP packets and mouse events (such as clicks) are extracted per process. Based on the HTTP packets and/or mouse events create respective HTTP request tree is created for each process. The request trees may then be analysed to identify advert requests and/or advert clicks. Based on the analysis advert clicks may then be identified as fraudulent if they are not preceded by a mouse click.

In the description that follows the term URL is used for ease of discussion. The skilled person would appreciate, however that when a URL is referred to, in practice this may be a URI or combination of one or more URLs and/or one or more URIs.

The traffic across a network interface is monitored. In particular, requests, such as HTTP requests, are captured (or logged or otherwise inspected). Typically, said monitoring comprises capturing (or recording) packets from the network interface. These packets are usually all (or substantially all) of the HTTP packets from the network interface. An HTTP packet usually comprises an HTTP request or an HTTP response. Whilst the discussion below relates to HTTP requests it will be appreciated that a similar approach could be used with HTTP responses or a combination of HTTP requests and HTTP responses. There are many ways to identify packets as HTTP packets, such as by using a tcp port filter. For example, the port filter might use ports 80 and or 8080, which are the standard ports for HTTP traffic. In this way a port filter may enable the capture of only those packets that have a specified port as that packet’s destination port. Said capturing may be performed using a packet capture library, such as libcap (described further in “*The libpcap project*,” http://sourceforge.net/projects/libpcap/, accessed: 2015-02-06)

Each HTTP request has a respective source port. The respective source port is typically the port of the client device from which the packet originates (or was sent). For each captured HTTP request, the respective process associated with said packet is identified. The respective process is typically the process that sent the packet. Typically, for an HTTP request the respective process is identified based on the source port of the request. In particular, the operating system may be queried, substantially at the time of capturing the request, to identify the process using the source port. It will be appreciated that there are many ways to query the operating system to identify such a process. For example, in a Linux (or Unix-like) system the the "/proc" file system can be interrogated to find out the process which is currently using the source port. It will also be appreciated that the query typically must be carried out during said monitoring of the network interface. Usually processes use random ports for making HTTP requests and same port may be used by different processes at different times.

Additionally other fields from the HTTP request (and/or any respective response) may be recorded. To find the respective response of a request, the respective source port of the request may be matched with the destination port of the response. From the request the other fields recorded may comprise any combination of: timestamp, source IP, source port, destination IP, destination port, referrer, the host URL, the request URL, and so on. From the response the other fields recorded may comprise any combination of: location, content type, content length, content encoding, http version, status code, reason phrase, and so on. This information may be stored in a database (such as a SQLite database). Typically, information is not extracted from the body of the requests, and the packets themselves are not saved as to do so may require gigabytes of storage space.

Optionally, the HTTP requests may be further processed based on the content requested. Such further processing may be based on any one of or any combination of: request URL, host URL, timestamp, and so forth. Typically, the further processing comprises identifying HTTP requests for advert related content. In particular, a set of rules for identifying HTTP requests for advert related content may be used. For example, advert blocking filters may be used to identify HTTP requests for advert related content. Such advert blocking filters are usually used in web browsers to prevent adverts being downloaded and/or displayed and are well known in the art, An example advert block filter that may be used is EasyList ("Easylist advert block filter," https://easylist.adblockplus.org/en/, accessed: 2015-02-06). Easylist is a popular advert block filter maintained by the online community and used in advert blocker add-ons of web browsers. It can filter advert related content via any of: URL filters, DOM element filters, and third-party advertisement domain filters.

Examples of the further processing steps above are shown in figure 8 use the Easylist filter which contains a total of 45423 rules. In these example steps a respective URL (for example the host URL and/or the request URL) from each HTTP request is checked against these rules. A decision as to whether the URL is blocked or not may then be saved in a database. These decisions may then be used to determine whether the request is classified as a request for advert related content as described in more detail shortly below.

The HTTP requests are logically grouped together. In particular, an HTTP request tree may be generated using the HTTP requests. Typically, a respective request tree is generated for each process. In this way it will be understood that a request tree usually comprises some or all of the HTTP requests for a given process.

Multiple HTTP requests may be generated when a web browser loads a given web page. For example, a browser loading a web page may fetch many other static resources, such as CSS, JavaScript or images, to embed in the web page. Each of these static resources may be fetched using a respective HTTP request. It will be appreciated that the respective HTTP requests for each static resource corresponding to a given web page may be grouped with the HTTP request for the given web page. One such example of grouping HTTP requests is presented in Crussel et al. (J. Crussell, R. Stevens, and H. Chen, "*Madfraud: investigating ad fraud in android applications*," in Proceedings of the 12th annual international conference on Mobile systems, applications, and services. ACM, 2014, pp. 123–134). In this case, we can group the HTTP requests using the

HTTP referrer header to form a tree of requests where the request to the HTML page is the root and requests to the static resources are the children.

By constructing a request tree corresponding to a given scenario of browsing, then it is easier to analyse and automatically detect advert clicks. Typically, each HTTP request (and/or the corresponding response) is represented as a single node in the request tree. A request tree may comprise a plurality of nodes (each node usually corresponding to a single HTTP request and/or the corresponding HTTP responses) and a plurality of connections (or edges or links) between nodes. Each connection usually connects two nodes. A node may have more than one connection to it. Nodes may be connected based on pre-determined criteria. For example, two nodes may be connected if:

1) The HTTP request of the latter node contains the request referrer field set to the host of the HTTP request of the former node; and/or

2) The HTTP request of the former node contains a location header along with a redirection status code to redirect the client to the latter node (such as to a URL of the HTTP request to the latter node); and/or

3) The HTTP request of the latter node contains an identifier (such as a client ID) of the former node in a URL of the HTTP request of the latter node.

Additionally, in the case of (1) above the former node may be considered as the parent of the latter node. As such, the connection may be marked as a “referrer” (e.g. see figure 9). In the case of (2) above the former node may be considered as the parent of the redirected node. As such, the connection may be marked as a “location” (e.g. see figure 9). In the case of (3) above, the former node may be considered as the parent node of the latter node. As such, the connection may be marked as a “client ID”. In this case, the client ID of a publisher (typically assigned by the advert network at the time of registration) may be used.

A given HTTP request tree may be processed to identify advert visits (or advert clicks or advert impressions). In particular, nodes corresponding to requests for advert related content may be identified (or classified). This is done using machine learning classification (or a machine learning algorithm). Machine learning typically involves the automatic use of past observations to make accurate predictions and would be well known to the skilled person. As such machine learning itself is not discussed any further herein. The classification may be based on one or more common characteristics of requests for advert related content. For example, requests for advert related content may comprise large number of query parameters. Additionally, or alternatively, responses corresponding to said requests normally comprise image or JavaScript content. Typically, the features (or data) extracted from the requests include any of: query parameters HTTP headers of the request (and/or the response); the constructed HTTP request trees, and so forth. The machine learning classification may then be based on any of these features. Optionally, so as to increase the accuracy of the classification decisions from the further processing, described previously, may be used.

It will be appreciated that there are many kinds of advert provider in the web space and it would be very difficult to identify all kinds of advert requests entirely by hard coded rules. The use of machine learning classification to automatically identify advert requests helps address this problem and often results in more accurate identification. Typically, web advertising links show much more diversification over time. As such, machine learning allows the classifiers to be retrained automatically. Typically, a machine learning library is used to implement machine learning classification. One such machine learning library is the weka machine learning library (M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "*The weka data mining software: an update*," in ACM SIGKDD explorations newsletter, vol. 11, no. 1, pp. 10–18, 2009).

It will be appreciated that the classification problem, outlined above, may be considered to be a binary classification problem on URLs. Typically, positive results are advert related and negative results are not advert related. In other words, advert related URLs may be considered requests to serve adverts to an advert network and negative examples may be considered all other requests (i.e. requests that are not related to serving ads).

As outlined above, the machine learning classification may be based on features related to the HTTP requests. These may include any of: characteristics of the structure of the URL, other URL (or URLs) present in the URL, and other page properties. Typically, features are encoded as either binary or integer values. This enables all features to be given equal weight at training time. Examples of such features are described in more detail below. It will be appreciated that embodiments of the invention may use any of, or any combination of, such features.

Features related to HTTP packet headers. Features in this set are typically extracted from the captured packet headers. These features can be based on any of: destination IP, content type, content length, status code, location URL, etc. For example, based on the location URL, the number of subdomains in the URL may be calculated as a feature. Additionally, or alternatively, the length (such as the number of characters) of the URL may be calculated as a feature. Features may be calculated from the host and/or request URL. These feature may include any of: the length of the host and/or request URL, number of subdomains in the host and/or request URL, whether a referrer is present or not, the length of the referrer URL, the number of subdomains in referrer URL, etc.

Features related to query parameters. A typical advert related HTTP request comprises a large number of query parameters. These query parameters are often used to relay information about the particular client machine and browser to find a matching advert for the request. Examples of features may include any of number of query parameters, average length of query parameters, the presence of pre-determined query parameters, etc.

Features related to HTTP request trees. Features may be calculated based on an HTTP request tree itself. Additionally, or alternatively, features may be calculated based on sub-trees of the HTTP request tree. A sub tree may be a part of the tree rooted at a given node (typically corresponding to a parent HTTP request). For example, features may include any of: the height of the tree or subtree (such as the height rooted at each node), the number of blocked URLs in the tree or subtree, the number of image and/or JavaScript requests in the tree or subtree, the number of different domains in the tree or subtree, etc.

The machine learning classification uses a classification model. The classification model is typically implemented using a machine learning library, such as that described previously. Five example classification models are described in more detail below, but it will be appreciated that the invention is not limited to any of these models. In these examples a binary output ids given for all instances. This can be thought of as returning either “ad” (i.e. the request is related to advert content) or “notad” (i.e. the request is not related to advert content) for every input instance. Of course it will be appreciated that other models may instead provide relative likelihoods, for example using percentages, and so on.

Naive Bayes: Naive Bayes is a simple probabilistic classifier which relies on Bayes Theorem with the assumption of independence between individual features of an example. Naïve Bayes has been shown to be effective on low-dimensional data in certain situations (O. Aydemir, M. Ozturk, and T. Kayikcioglu, “*Performance evaluation of five classification algorithms in low-dimensional feature vectors extracted from eeg signals*,” in Proceedings of the 34th International Conference on Telecommunications and Signal Processing (TSP). IEEE, 2011, pp. 403–407). This model may be implemented using weka as outlined above. Typically, a Gaussian distribution is used for the likelihood of each feature $P(x_i | y)$. This may be particularly advantageous due to the numeric nature of the features. The class is computed by the following equation:

$$y = arg\ max_y P(y) \Pi_{i=1}^{n} P(x_i|y)$$

For Naive Bayes classifier, other parameters are usually left at the default values.

Support Vector Machines: Support Vector Machines or SVMs are non-probabilistic classifiers that typically work in high-dimensional feature spaces. SVMs for binary classification are trained to construct hyperplanes with the widest possible margins to the nearest training examples. The decision rule during testing is made by determining which side of the hyperplane the point lies on and is given by the sign of the result of the following formula:

$$\sum_{i=1}^{n} y_i \alpha_i K(x_i, x)$$

Here, K(x_i, x) is the kernel function that implicitly maps the feature vectors into higher-dimensional space. x_i refers to each example in the training set, y_i to each corresponding class and i to the weight of each training example.

This model may also be implemented using weka. Typically, Sequential Minimal Optimizaiton (SMO) is used with polykernel as the kernel function. The penalty parameter C may take the value of 1.0.

k-Nearest Neighbours: k-Nearest Neighbours (sometimes referred to as iBK) is a popular non-parametric estimator for classification problems and can be efficient on low-dimensional data. The decision function is computed implicitly by taking the majority vote of a data points k-nearest neighbours classes. The kNN classifier may use the distance-weighted k-nearest neighbours rule. Usually weights are assigned to the k-nearest points proportional to the inverse of the Euclidian distance between each neighbour and the point in question. In some examples, the weight w_i of each point x_i in the vote may be given by:

$$w_i \infty \frac{1}{d(x, x_i)}$$

, or

$$w_i \propto \frac{1}{d(x, x_i)}$$

Different values of k that may be include, k=2 and k=5 (often denoted in the evaluation as 2kNN and 5kNN). This model may be implemented using weka.

C4.5: C4.5 is an algorithm developed by Quinlan (J. R. Quinlan, C4. 5: programs for machine learning. Elsevier, 2014) that builds decision trees from a set of training data using the concept of information entropy. A leaner may be constructed using J48 in weka which is an implementation of C4.5. For an implementation using weka the default values are sufficient in this classifier.

Random Forest: The random forest (RF) classifier (L. Breiman, "*Random forests*," in Journal of Machine Learning, vol. 45, no. 1, pp. 5–32, 2001) utilizes bagging and the "random subspace method" to construct randomized decision trees. The outputs of ensembles of these randomized, unpruned decision trees are then typically combined. A maximum depth value of 10 may be used.

Typically, when a user clicks on an advert, the browser generates a HTTP request to the advert network and the browser is redirected to the advertiser's page. The advert network may also record the event. The URL of the advertiser's page is typically provided in the response (for example, as part of the location header). This whole process may be thought of as an advert visit, initiated by an advert click. In this case this is a real advert click as the user initiated the visit.

Advert clicks (or advert visits) are identified in a request tree. Typically, a node is identified as an advert click based on the classification of the HTTP request corresponding to the node, as set out above. In particular, criteria are used to identify nodes as advert clicks in the request tree. For example, a node may be marked as an advert click if the node is a child of an advert request, the edge is marked as location, and it is the root of a subtree whose nodes represents a separate website. Additionally, or alternatively, a node may be marked as an advert click where the node is a root of a subtree (or subtrees) representing visits of separate websites. In particular, it may be considered that if the parent of such a node contains a pre-specified number of advert request nodes, then the node is an advert click. Such criteria may be especially useful where there is no location header.

In some cases it may be assumed that the parent node contains a number of adverts and the subtree is formed because of the user click on one of the adverts. Typically such nodes use the former pages as referrer. Both scenarios are illustrated in figure 9. The criteria may be adjusted to try to maximize the recall of advert clicks. Recall may be calculated by the following formula:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

This allows the detection of as many real advert clicks as possible so as to find fraudulent clicks.

Hardware mouse events are logged/monitored during the monitoring of the network interface. In particular, a timestamp is typically recorded for each hardware mouse click. Such logging is usually performed on a per process basis. The logging of the hardware mouse events may comprise executing a separate thread for each mouse device present in the system.

In order to perform the logging per process, the active process on the system for each hardware mouse event is typically recorded. For example, at the time of recording a mouse event, the top window ID of the GUI system is determined. The process IDs associated with said window (and, optionally, its parent and/or child windows) is recorded. This may be recorded as “event— pid1, pid2, pid3, ...—time” in a database.

The above discussion refers to hardware mouse events which may be thought of as events triggered by a physical device in communication with the system. Usually, events from such devices trigger interrupts on the system via their device drivers. It will be appreciated that the above discussion is equally applicable to virtual devices, such as where interrupts are triggered by virtual device drivers. In particular, the invention may be used with systems running on virtual machines where “hardware” devices seen by the system correspond to virtual devices presented to the “hardware” interface by the host system. In other words, the “hardware” events outlined above may be thought of as any device level event, rather than say an API level event of the operating system of the system. These API level events are typically the mechanism by which “ghost” events, or fraudulent clicks, are triggered.

Based on the advert clicks identified in a request tree the process corresponding to said request tree may be marked as fraudulent or non-fraudulent. Typically, if at least one click which is real, then we consider the process as non-fraudulent. We assume that the process is communicating with real user and fraudulent clicks are false positives.

Individual advert clicks (or impressions) are classified as based on the hardware mouse clicks logged previously. In particular, the advert clicks in a request tree (or trees) and the hardware mouse events for the corresponding process are compared (or mapped). Such a comparison allows it to be determined if a process clicked on an advert without a corresponding hardware mouse event (or click). In this way it may be determined that the process used a software simulated click or independent HTTP request (without a mouse click). For example, if the HTTP request trees of a process contain nodes detected as advert click, however they do not receive any hardware mouse click, then the click is marked as fraudulent. In this way it will be appreciated that the mapping is typically one-to-one, however there may be advert clicks which do not map to any hardware mouse click. Such advert clicks are considered fraudulent.

As describe above, typically if all the detected advert clicks are fraudulent for a given process, then the process is marked as fraudulent. On the other hand, if some of the mouse clicks are real, then the process is marked as non-fraudulent. However, it will be appreciated that this behaviour may be modified to alter the false positive/false negative rate. In particular, in some examples, if the number real mouse clicks exceed a certain threshold the process may be marked as non-fraudulent. Equally, the marking of the process may be based upon a ratio of real to fraudulent clicks, and so forth.

It will be appreciated that information regarding fraudulent processes may be transmitted to the advertising network. For example, once a process has been marked as fraudulent then subsequent advertising visits (or impressions) related to that process may be discarded by the advertising network. Additionally, or alternatively, an advertising network may disregard all (or some) previous impressions related to that process. In some cases an advertising network may extend any of these measures to include instances of the same and/or processes running on different client devices.

Example implementations and testing

Described below are various example implementations of the systems and methods outlined above. Accompanying these example implementations is test data derived from testing performed on these example implementations. It will be appreciated that these examples below and the accompanying testing is merely provided to aid understanding of the invention and to highlight some of the advantages it may provide. The skilled person would recognise that these examples are not in any way limiting to the invention described herein and the presence and/or lack of specific features from these examples does not in any way limit how the skilled person may work the invention.

To evaluate the average classification accuracy of each classifier, we use k-fold cross validation (CV) with k=3. By using k-fold CV, we ensure that the classifier is trained on every example at one point as opposed to random splits where some examples might be excluded all together. In comparing the overall effectiveness of the classifiers, we look at the overall average accuracy on the testing data as well as the precision and false positive rate. The formulas to calculate precision and false positive rate are given below:

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$false\,positive\,rate = \frac{false\ positives}{false\ positives + true\ negatives}$$

In our case, we like to minimize false positive rate and maximize the overall positive classification accuracy.

We select 25 high ranked websites containing advertisements for our experiments. We use firefox developer edition and four bots to browse these websites and generate a total of 7708 HTTP requests. The table below lists the category of the websites and average global rank of the websites in www.alexa.com:

Category	# of pages	Avg global rank
articles	2	84.5
auto	1	1880
blog	3	219.6667
computer games	1	91412
education	1	534
entertainment	5	649.8
health	1	31262
news	2	652.5
science	2	2322.5
technology	2	325.5
travel	3	1947
weather	2	1011
	Total: 25	

TABLE I: Category and Average rank of websites in Alexa.com

A. Execution of our approach

We execute our approach as a service of the operating system in client machine. The OS select a time frame each day when it detects heavy network use and run our service to capture the HTTP requests. Then, in its idle time, it analyses the requests and alarms the user accordingly. In our experiment, it took 429.336 seconds to analyse 7708 HTTP requests in a Pentium Core i7-4770 3.4GHz machine with 20GB of RAM and UBUNTU 14.04 64-bit operating system.

B. Click bots

We use four kinds of bots to generate fraudulent traffic. Two of them use browser-driver to control two browsers (Google Chrome and PhantomJS). Browser drivers are application programming interfaces (API) that can communicate with a browser and can do all the interactions programmatically. We use selenium browser driver available from "*Selenium: Browser automation*," http://www.seleniumhq.org/, accessed: 2015-01-09. Google chrome creates visible window and thus noticeable by the user. So, in case of google chrome, we make the window hidden. PhantomJS is a headless browser and does not create any human viewable window. The third one is a firefox extension that visits instructed webpages and click on links. The fourth bot is an independent program, which generates HTTP traffic and parses the response. We use web browser control to implement the feature.

Number of HTTP request	Number of ad request	Number of ad related domain
7708	809	13

TABLE II: Ground truth for classifiers

In all the bots, we supply website addresses and XPaths of the web elements containing ads. XPath is a query language for selecting nodes in a XML document. The bots first visit the websites given to them and randomly scroll up and down. Then they generate clicks on the web elements containing adverts which redirect the browser to the advertisers' pages. Occasionally our bots generate multiple clicks on the same page, if multiple xpaths are supplied to them.

C. Ground truth for advert request classifier

To build the ground truth dataset for identifying advert requests, we visit all the websites and inspect the advert elements by using firebug extension of firefox. We, then manually classify 809 advert requests from 7708 HTTP requests generated by our bots and firefox browser. We do not classify tracking and conversion requests to the advert networks and advertisers as they are not requests for serving ads. Total number of advert related domain is 13. We present the statistics in Table II.

Classification Algorithm	Avg. Accuracy (%)	Precision (%)	FP Rate (%)
NaiveBayes	92.46	59.05	7.48
SVM	96.85	100	0
C4.5	99.27	95.09	0.59
2kNN	99.27	95.64	0.52
5kNN	99.29	95.64	0.52
RandomForest	99.57	97.43	0.30

TABLE III: Performance of different machine learning classifiers to classify ad requests over 3-fold cross-validation. They classify a web request as either "ad" or "notad". We consider the instances from the class "ad" as positive.

D. Identifying advert requests

After we build the training dataset, we use our classifiers to classify them. Table III shows the comparison of the average accuracy, precision and false positive rate of various classification algorithms. In figure 10, we present the accuracy of the positive class (recall) of each classifier. Other than Naïve Bayes and SVM, all other algorithms perform acceptably. In our experiment, RandomForest classifier do the best job in classifying advert requests (highest recall and lowest false positive rate). In fact, SVM and k-neighbours do poorly on categorical (i.e., non-Euclidean) feature space. We believe that for our feature space decision trees would be an appropriate choice as we include categorical features. Nonetheless, the acceptable performance of the classification algorithms come from the choice of features and the use of the decision of an advert blocker. In conclusion, we decide to use RandomForest as our classification algorithm. The RandomForest classifier creates a forest of decision trees by randomly selecting subsets of the feature space and training a decision tree using each subset. The most predictive decision trees are weighted appropriately, and in turn indicate which features are the most predictive.

To compare, in Table IV and Table V, we present the confusion matrix of our RandomForest classifier and from a similar work done by Crussell et. al. (“*Madfraud: investigating ad fraud in android applications*,” in Proceedings of the 12th annual international conference on Mobile systems, applications, and services. ACM, 2014, pp. 123–134). They analysed HTTP requests from Android apps and used RandomForest classifier to find advert requests.

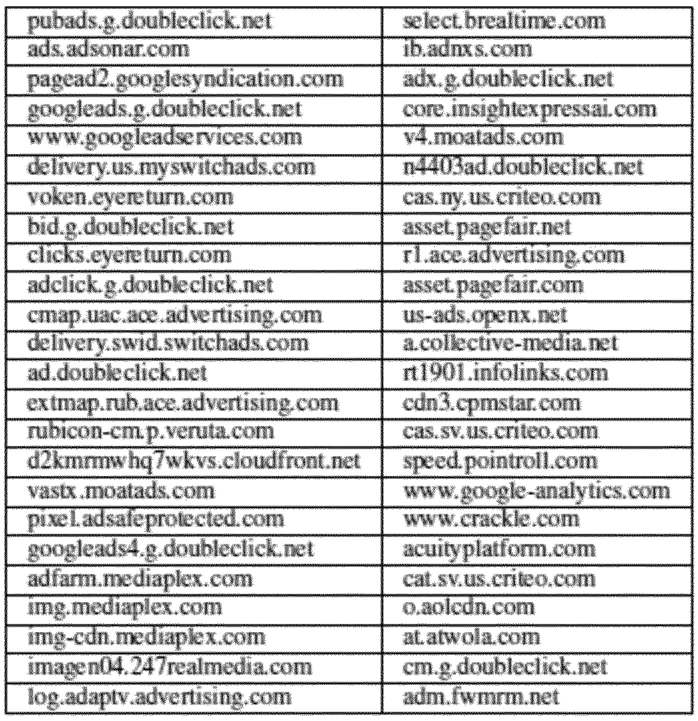
	NOTAD	AD	Recall
NOTAD	6879	20	99.7%
AD	17	792	97.9%
Precision	99.8%	99.7%	

TABLE IV: Confusion matrix of our RandomForest classifier, computed using 3-fold cross validation. The class-weighted accuracy is 98.8%

	ARQ	NARQ	Recall
ARQ	28	11	71.8%
NARQ	9	11475	99.9%
Precision	75.7%	99.9%	

TABLE V: Confusion matrix of RandomForest classifier from [28]. They analyzed HTTP requests from Android apps. the class-weighted accuracy is 85.9%. Here, ARQ is the positive class of ad requests and NARQ is the negative class of not ad requests.

As we can clearly see, our algorithm outperforms their with a large margin in detecting positive instances. We achieve 97.9% recall and 99.7% precision for advert request class, while they reported 71.8% recall and 75.7% precision for the same class. We believe the use of an advert block filter may have impacted our performance positively. Table VI shows some example of the ad-related domains present in detected advert requests.



pubads.g.doubleclick.net	select.brealtime.com
ads.adsonar.com	ib.adnxs.com
pagead2.googlesyndication.com	adx.g.doubleclick.net
googleads.g.doubleclick.net	core.insightexpressai.com
www.googleadservices.com	v4.moatads.com
delivery.us.myswitchads.com	n4403ad.doubleclick.net
voken.eyereturn.com	cas.ny.us.criteo.com
bid.g.doubleclick.net	asset.pagefair.net
clicks.eyereturn.com	r1.ace.advertising.com
adclick.g.doubleclick.net	asset.pagefair.com
cmap.uac.ace.advertising.com	us-ads.openx.net
delivery.swid.switchads.com	a.collective-media.net
ad.doubleclick.net	rt1901.infolinks.com
extmap.rub.ace.advertising.com	cdn3.cpmstar.com
rubicon-cm.p.veruta.com	cas.sv.us.criteo.com
d2kmrmwhq7wkvs.cloudfront.net	speed.pointroll.com
vastx.moatads.com	www.google-analytics.com
pixel.adsafeprotected.com	www.crackle.com
googleads4.g.doubleclick.net	acuityplatform.com
adfarm.mediaplex.com	cat.sv.us.criteo.com
img.mediaplex.com	o.aolcdn.com
img-cdn.mediaplex.com	at.atwola.com
imagen04.247realmedia.com	cm.g.doubleclick.net
log.adaptv.advertising.com	adm.fwmrm.net

TABLE VI: Some examples of domains found in detected ad requests

E. Finding advert clicks in request tree

Finding advert click is the trickiest part in our approach. In figure 11, we report the result of our detection. We detect 24 advert clicks among a total of 28. Our system could not detect 4 advert clicks because those requests do not contain a referrer or location header in the HTTP packet and they are likely generated by dynamic javascript code with those information hidden in different url parameters. Here, we have a total 69 false positives. However, in this step, the most important thing is to detect real advert clicks. False positives do not impact the result of the next step of detecting fraudulent processes. We only fail in the event of all the detected advert clicks are false positives but there exist some real advert clicks. Fortunately, we find no such case in our experiment. Crussell et. al. did not report the performance of their advert click detection, so we are unable to compare in this case. Table VII shows the recall and precision of our advert click detection system.

	Recall	Precision
Ad click detection	85.71%	25.8%

TABLE VII: Evaluation of ad click detection

F. Finding fraudulent processes

We successfully detect all our bots as fraudulent. We find all the advert clicks are detected as fraudulent in them. In contrast, we mark the firefox developer edition non-fraudulent because we find 26.98% of the clicks are fraudulent. Since it contains 73.02% real advert clicks, it must have interacted with a real human. We assume that the fraudulent advert clicks are the result of the false positives that are not preceded by any real mouse click.

G. False positive advert clicks and improvement

We have a large number of false fraudulent clicks in our result. We begin to investigate the reason and observe that websites load contents from different domain which is falsely detected as advert clicks in our system. We also observe that many requests to the advert networks, web analytics, and web tracking sites generate further communications with them and those later requests use the first one as referrer, which is falsely detected as advert click by the second rule of our detection algorithm. Unfortunately, we have to use our second rule to improve the recall of our system as detecting real clicks are very important for identifying fraudulent processes. Figure 12 gives some examples of false advert clicks. We can see that, some of the requests are made to static resources, such as css files.

So, we write a filter to remove those clicks and ultimately we have 39 false positives. Figure 13 shows the improvement. The precision goes up from 25.8% to 38.1%.

Final comments

Although aspects and embodiments of the invention have been described herein as operating and executing on particular computer equipment such as portable and hand help computing devices and servers, the skilled person will be aware that aspects of the invention can be implemented on a wide variety of data processing equipment. Generally, such equipment will be provided with at least one processor element for execution of the software, and suitable memory operatively coupled to the at least one processor for storing the software and data relevant to execution of the software. For example, a smart phone or other hand held device will typically comprise one or more microprocessor cores, usually one or more graphical processor cores, suitable memory, network connectivity typically through Wi-Fi, Bluetooth and other arrangements, and so forth. Such computer equipment will also typically be provided with suitable in/out functionality such as a display screen, touch screen, keyboard, pointer device, data storage elements and so forth as desired or needed.

However, it should be understood that where particular functionality or software components are described herein as being provided on or executing at a particular device, such functionality or software components may also be distributed across multiple devices, multiple processors and so forth. For example, functions which are described as being executed at a client device may be partly or wholly executed at a server and vice versa.

Similarly, where functionality or software components are described as being located at a server, such functionality or software components may be implemented on a single server computer or processor or distributed across multiple servers or processors, which may be collocated in a single server unit, or distributed across multiple units which may be adjacent or far apart. Such servers are also provided with suitable processors, memory, network connectivity and other functions as required for their operation.

Although particular embodiments have been described, the skilled person will be aware of modifications and alterations to these which remain within the spirit and scope of the invention. For example, although various ways in which the anti-fraud element 30 may contribute to aversion of fraud through detection and/or prevention have already been described, other techniques may be used by the anti-fraud element 30, by other aspects of the advert code 18, or by other processes or software elements executing on the client device. Some examples of such techniques are described in detail in the annexe below.

Therefore, further background, examples, embodiments and discussion of the invention are set out in the following annexe. Although the material in this annexe refers specifically to various aspects such as mouse clicks, platforms and operating systems, of course it is equally applicable to other situations such as any suitable user selection action such as a suitable interaction with a touch screen or a selection using a keyboard or other switch element.

CLAIMS:

1. A method of operating a server to deliver an online advert to a client device, the method comprising:

receiving at the server, from a web document executing in a web browser on the client device, a request for an advert;

in response to the request for an advert, preparing, at the server, advert code for execution within the web document at the client device so as to display the advert to a user of the client device, at least a portion of the prepared advert code being in a protected form for averting advertising fraud; and

transmitting the advert code to the client device for execution on the browser for execution so as to display the advert to a user of the client device.

2. The method of claim 1 wherein the advert code comprises an anti-fraud element at least partly within the protected portion of the advert code, the anti-fraud element being arranged for execution within the web document at the client device so as to provide one or more anti-fraud functions relating to the advert.

3. The method of any preceding claim wherein the advert code comprises an interaction action which at least partly defines one or more interactions between a user of the browser and the advert, and one or more actions to be taken by the browser when the one or more interactions occur.

4. The method of claim 3 when dependent on claim 2 wherein the anti-fraud element is arranged to provide one or more of the anti-fraud functions following occurrence of one or more of the defined user interactions with the advert.

5. The method of claim 4 comprising, following said one or more anti-fraud functions, permitting the browser to complete an action which is defined by the interaction action.

6. The method of claim 5 wherein permitting an action to be completed by the browser comprises one of: transmitting to the browser a redirection URL for the browser to navigate to in response to the user interaction with the advert; and permitting the browser to navigate to a redirection URL already comprised in the web document, in response to the user interaction with the advert.

7. The method of any of claims 4 to 6 comprising recording the user interaction as a suspect or fraudulent user interaction if one or more of the anti-fraud functions fails.

8. The method of any preceding claim wherein preparing the advert code comprises incorporating an advert serial code within the advert code, and the method further comprises carrying out an anti-fraud verification that information subsequently received from the advert code executing on the browser corresponds with the advert serial code.

9. The method of any preceding claim further comprising:

storing, on the server, one or more initial attributes of the prepared advert code;

receiving a subsequent message from the client device comprising one or more later attributes of the advert code; and

verifying the one or more later attributes of the advert code against the corresponding stored initial attributes of the advert code.

10. The method of any preceding claim further comprising:

receiving, from the client device, cursor trajectory data relating to cursor movement associated with an interaction between a user of the browser and the advert; and

verifying the cursor trajectory data by comparing the received cursor trajectory data against previous cursor trajectory data stored on the server.

11. The method of claim 10 wherein the comparing comprises calculating a geometric distance between the received cursor trajectory data and the previous cursor trajectory data.

12. The method of claim 11 wherein the verifying comprises comparing the geometric distance to a threshold.

13. The method of any preceding claim further comprising the server carrying out anti-fraud verification of the request for an advert before transmitting the advert code to the browser.

14. The method of claim 13 wherein carrying out anti-fraud verification of the request for an advert comprises checking an origin of the request against a database relating origin of a request to fraud risk.

15. The method of any preceding claim wherein at least some of the prepared advert code in a protected form is protected by at least one of:

one or more cloaking techniques;
one or more software obfuscation techniques;
one or more node locking techniques;
one or more diversity techniques; and
one or more digital watermarking techniques.

16. The method of claim 15 wherein the cloaking techniques include one or more of homomorphic data transformation, control flow transformation, white box cryptography, key hiding, program interlocking and boundary blending.

17. A method of operating a client device to display an online advert using a web browser comprising:

receiving and executing, using the browser, a web document including advert request code, execution of the advert request code causing the browser to transmit an advert request to an advert server;

receiving advert code from the advert server, at least a portion of the advert code being in a protected form for averting advertising fraud; and

executing the advert code at the client device so as to present the advert to a user of the client device.

18. The method of claim 17 wherein the advert code comprises an anti-fraud element at least partly within the protected portion of the advert code, the method comprising executing the anti-fraud element within the web document on the client device to provide one or more anti-fraud functions relating to the advert.

19. The method of claim 18 wherein the advert code comprises an interaction action which at least partly predefines one or more interactions between a user of the browser and the advert, and one or more actions to be taken by the browser if and when the one or more interactions occur, and the anti-fraud element is executed to provide one or more of said anti-fraud functions following occurrence of one or more of the defined user interactions with the advert.

20. The method of claim 18 or claim 19 comprising, following said one or more anti-fraud functions, permitting the browser to complete an action which is defined by the interaction action.

21. The method of any of claims 17 to 20 wherein the advert code comprises an advert element including an advert URL associated with the advert.

22. The method of claim 21 wherein the anti-fraud element is arranged to replace the advert URL with a fake URL until one or more conditions are satisfied, the fake URL being different from the advert URL.

23. The method of claim 22 wherein the one or more conditions include one or more of:

a cursor being present over the advert whilst the advert is being displayed to the user;

a fixed time period having elapsed following loading of the web document on the browser; and

a frame rate of the browser having stabilised following loading of the web document on the browser.

24. The method of any of claims 21 to 23 wherein the anti-fraud element is arranged to periodically replace the advert URL with an updated advert URL.

25. The method of claim 24 wherein the anti-fraud element is arranged to periodically replace the advert URL in response to user interactions with the web document using the browser.

26. The method of any of claims 21 to 25 wherein the anti-fraud element is arranged to create copies of the advert element within the DOM of the web document, each copy including a respective fake URL different from the advert URL, such that the advert and copies of the advert are displayed in a stack with only the advert being visible to the user at the top of the stack.

27. The method of claim 26 wherein the anti-fraud element is arranged to periodically alter the DOM structure of the web document so as to randomly rearrange the advert element and the copies of the advert element within the DOM structure.

28. The method of claim 27 wherein the anti-fraud element is arranged to periodically alter the DOM structure of the web document in response to user interactions with the web document using the browser.

29. The method of any of claims 21 to 28 wherein the advert element further includes an advert creative associated with the advert, and wherein the anti-fraud element is arranged to replace the advert creative with a fake creative until the advert is displayed to the user, the fake creative being different from the advert creative.

30. The method of any of claims 17 to 29 wherein the anti-fraud element is arranged to verify whether the advert is on a viewable portion of the browser window and/or whether the advert is being displayed on a topmost tab within the browser.

31. The method of any of claims 17 to 30 wherein the anti-fraud element is arranged to store on the client device one or more initial attributes of the advert code received from the advert server, and wherein the anti-fraud element is further arranged to verify one or more later attributes of the advert code against the corresponding stored initial attributes of the advert code.

32. The method of any of claims 17 to 31 wherein the anti-fraud element is arranged to send cursor trajectory data to the advert server, the cursor trajectory data relating to cursor movement associated with an interaction between a user of the browser and the advert.

33. The method of any preceding claim when dependent on claim 2 or claim 18 wherein the anti-fraud functions include one or more anti-fraud verifications arranged to verify one or more of:

integrity of the anti-fraud element;
integrity of the advert;
that the anti-fraud element is executing within a browser;
visibility of the advert in a graphical display of the browser;
behavioural characteristics of the user of the browser; and
structure of the DOM of at least part of the web document.

34. A server arranged to deliver an online advert to a client device by carrying out the method of any of claims 1 to 16.

35. A client device arranged to display an online advert using a web browser by carrying out the method of any of claims 17 to 33.

36. A method of delivering an online advert to a web browser executing on a client device comprising:

transmitting a web document to the client device for execution by the browser;

receiving from the web document executing on the browser, at a remote server, a request for a piece of digital rights management (DRM) protected content;

delivering the requested DRM protected content to the client device and providing the client device with DRM rights for the browser to replay the content, the content comprising an identifier for the browser to recover from the content when the content is replayed; and

receiving the identifier from the browser.

37. The method of claim 36 further comprising, in response to receiving the identifier from the browser, preparing advert code for execution within the web document and transmitting the advert code to the web browser according to any of claims 1 to 24.

38. The method of claim 36 or 37 wherein the web document is arranged to recover the content from a graphics buffer of the client device during replay of the content.

39. The method of any of claims 36 to 38 wherein the identifier is specific to one or more of: the client device, the web browser executing on the client device; and the web document executing on the web browser.

40. An anti-fraud system arranged for:

receiving, from advert request code executing on a web browser of a client device, a request for a piece of digital rights management (DRM) protected content;

delivering the requested DRM protected content to the client device and providing the client device with DRM rights for the browser to replay the content, the content comprising an identifier for the browser to recover from the content when the content is replayed; and

receiving the identifier from the browser.

41. Advert request code for execution within a web browser on a client device, the advert request code comprising:

a content play function, the content play function being arranged to send a content request to a remote server for a piece of DRM protected content, and to receive and cause replay of the content at the client device, the content comprising an identifier; and

a request generator arranged to receive the identifier from the replayed content and incorporate the identifier within an advert request for sending to an advert server.

42. A method of operating a client device comprising:

monitoring advertising page visits made by one or more processes executing on the client device, and detecting if each advertising page visit is triggered by a user interaction.

43. The method of claim 42 wherein the one or more processes are processes of one or more web browsers executing on the client device, and the advertising page visits comprise HTTP requests to advertising web sites.

44. The method of claim 42 or claim 43 further comprising creating an HTTP request tree for each process and analysing the request trees to identify advertising page visits.

45. The method of claim 44, wherein said analysing comprises applying a machine learning classification to the request trees.

46. The method of any of claims 42 to 45 further comprising, for each process, monitoring one or more respective hardware input device events, and wherein said detecting comprises, for each process, comparing the one or more respective hardware input device events with the advertising visits associated with the process.

47. A method of detecting a fraudulent process in a client device, wherein the client device is executing one or more processes, the method comprising, for each process of the one or more processes:

monitoring respective requests transmitted across a network interface of the client device;

monitoring respective hardware input events associated with the process;

identifying one or more respective advert impressions by applying a machine learning algorithm to the respective requests;

classifying the one or more respective advert impressions as either fraudulent or non-fraudulent in dependence on the respective hardware input events; and

marking the process as fraudulent based on said classification.

48. The method of claim 47 wherein said identifying comprises creating a respective request tree based on the respective requests and applying one or more criteria to the request tree based on the output of the machine learning algorithm.

49. The method of claim 47 or claim 48 wherein said identifying comprises filtering the respective requests using an advert blocking list.

50. The method of any one of claims 47 to 49 wherein said classifying comprises mapping the respective hardware events to the one or more respective advert impressions.

51. Computer program code arranged to put into effect the method of any one of claims 1 to 33, 36 to 39, or 42 to 50.

52. One or more computer readable media carrying the computer program code of claim 51.

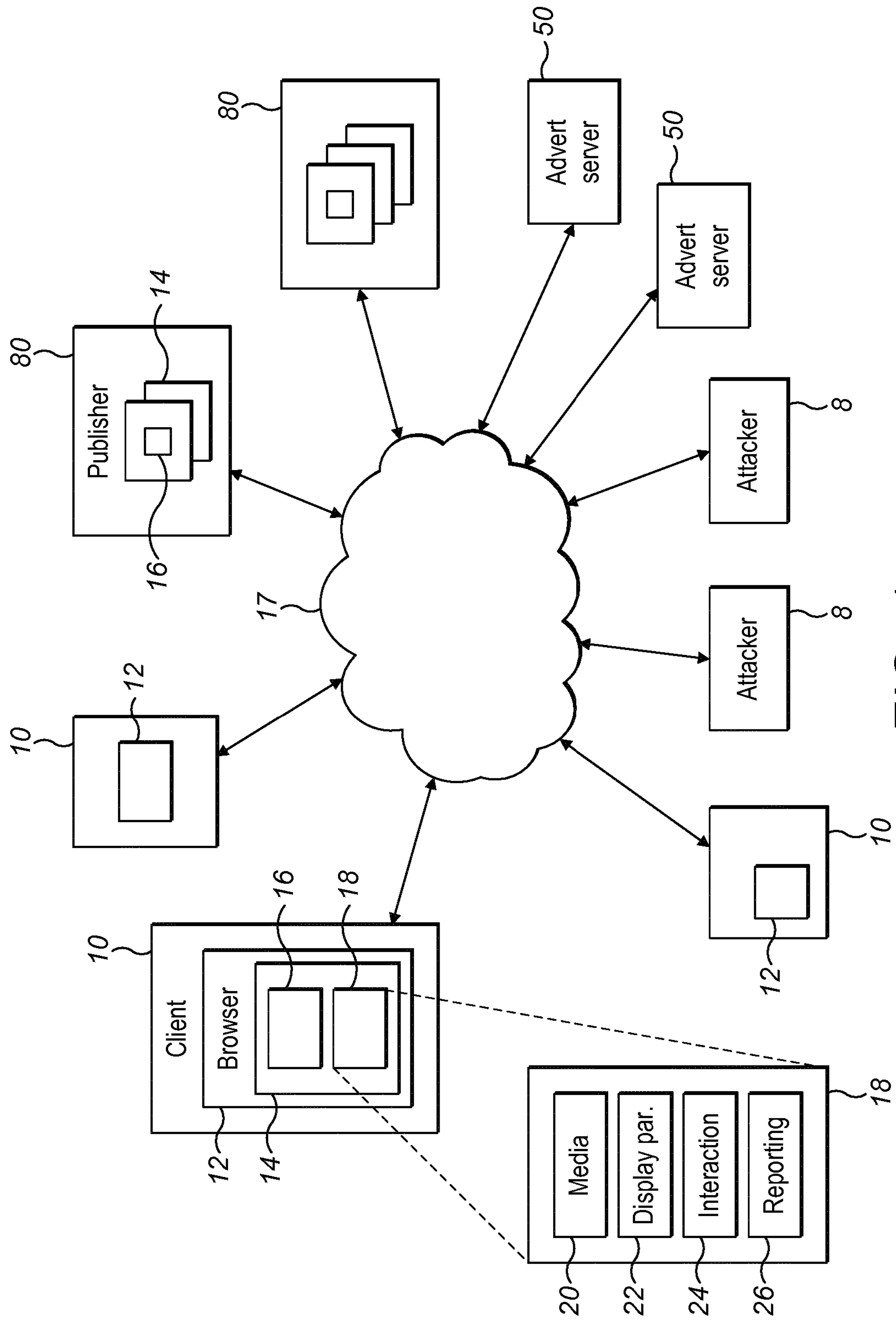

80
Publisher
14
16
80
50
Advert server
50
Advert server
10
12
17
Attacker
8
Attacker
8
10
Client
Browser
16
18
12
14
10
12
18
Media
Display par.
Interaction
Reporting
20
22
24
26

FIG. 1

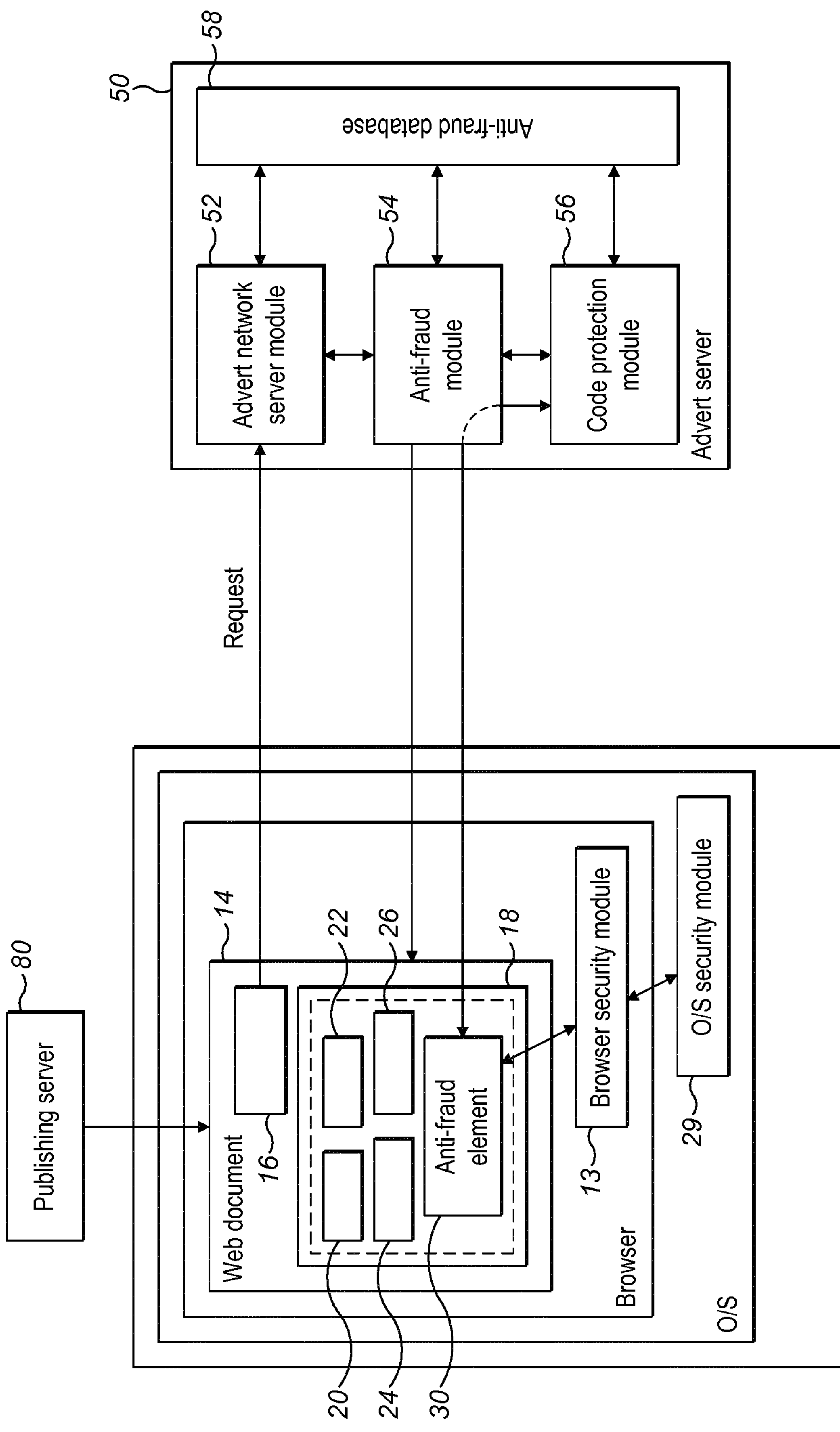

Publishing server
80
Client device
O/S
Browser
Web document
14
16
18
22
26
Anti-fraud element
20
24
30
Browser security module
13
O/S security module
29
Request
Advert server
50
58
Anti-fraud database
Advert network server module
52
Anti-fraud module
54
Code protection module
56

FIG. 2

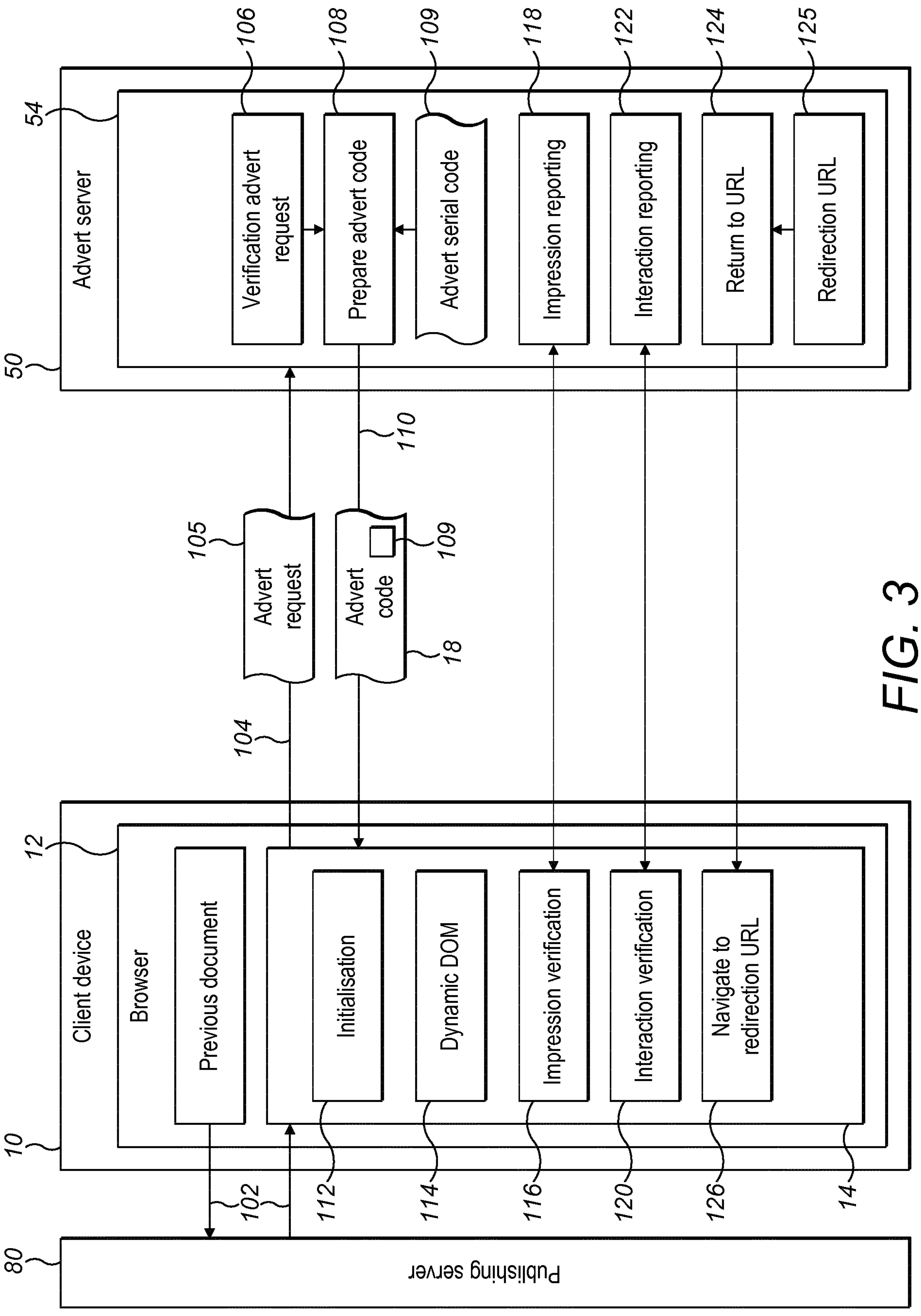

Advert server
54
50
Verification advert request
106
Prepare advert code
108
Advert serial code
109
Impression reporting
118
Interaction reporting
122
Return to URL
124
Redirection URL
125
110
Advert request
105
Advert code
109
18
104
Client device
10
Browser
12
Previous document
Initialisation
112
Dynamic DOM
114
Impression verification
116
Interaction verification
120
Navigate to redirection URL
126
14
102
Publishing server
80

FIG. 3

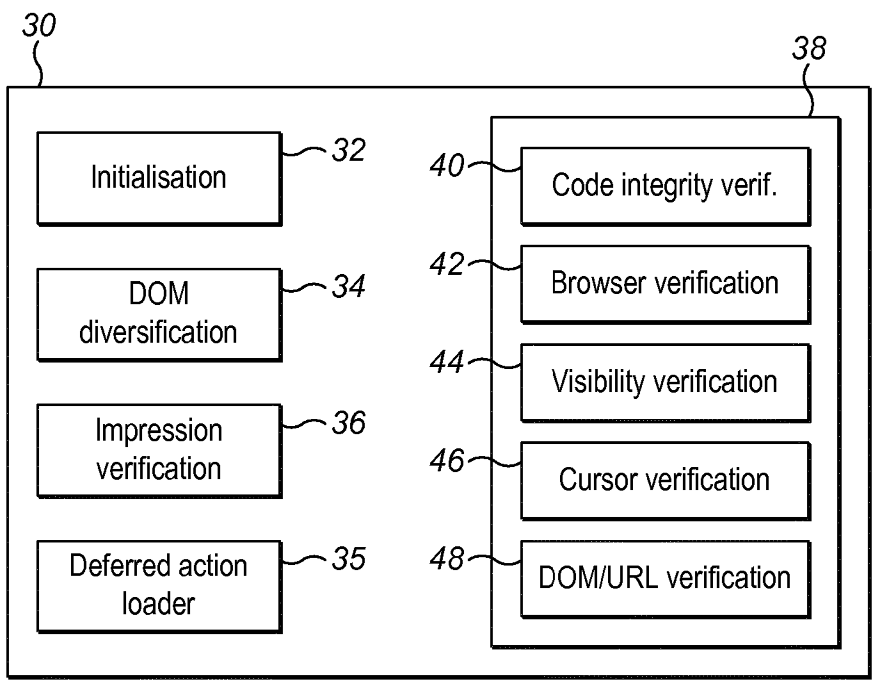

30
38
Initialisation
32
DOM diversification
34
Impression verification
36
Deferred action loader
35
40
Code integrity verif.
42
Browser verification
44
Visibility verification
46
Cursor verification
48
DOM/URL verification

FIG. 4

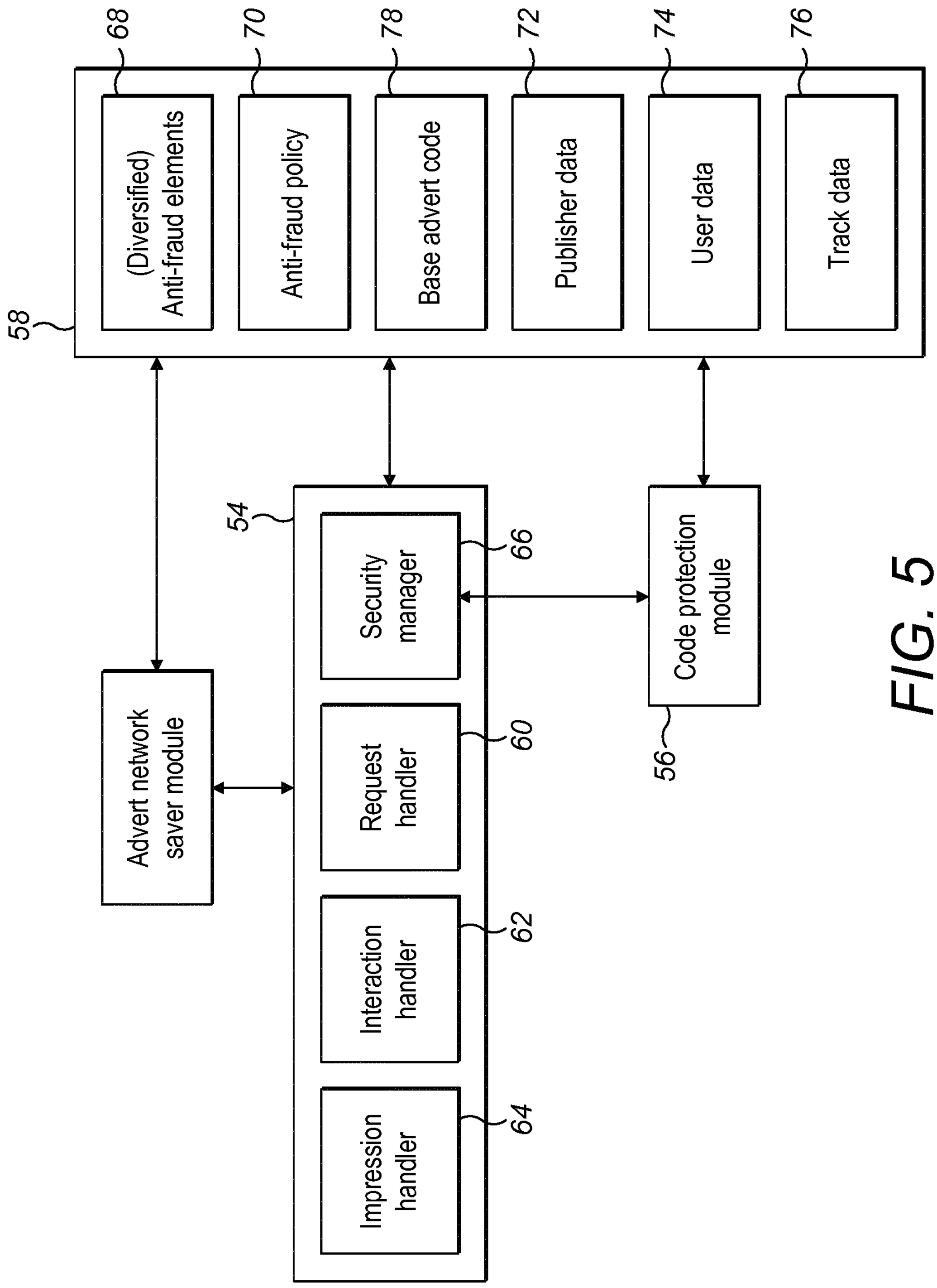

58
68
(Diversified) Anti-fraud elements
70
Anti-fraud policy
78
Base advert code
72
Publisher data
74
User data
76
Track data
Advert network saver module
54
Security manager
66
Request handler
60
Interaction handler
62
Impression handler
64
Code protection module
56

FIG. 5

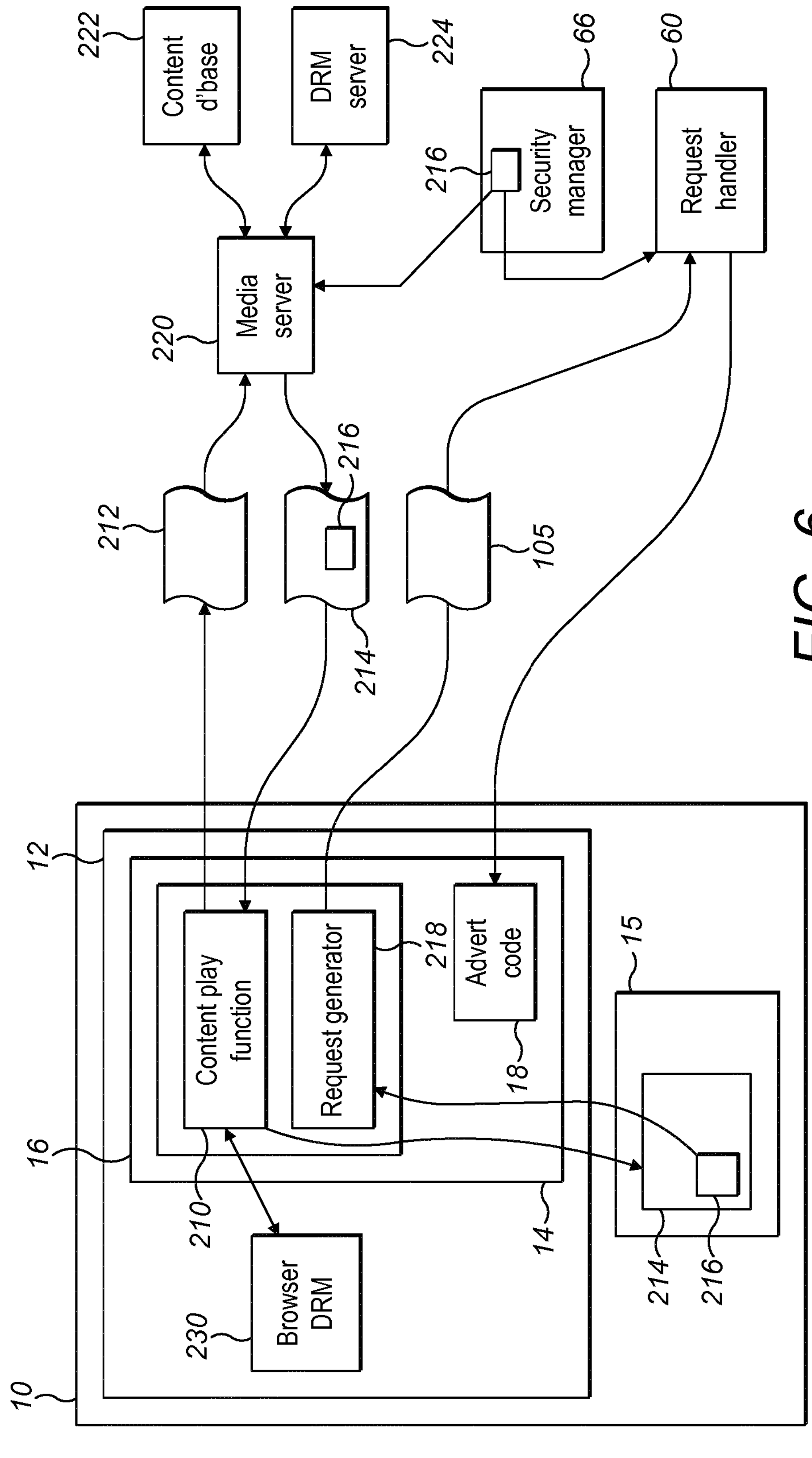

222
Content d'base
DRM server
224
66
216
Security manager
60
Request handler
220
Media server
212
216
214
105
12
16
Content play function
Request generator
218
Advert code
18
15
210
14
Browser DRM
230
10
214
216

FIG. 6

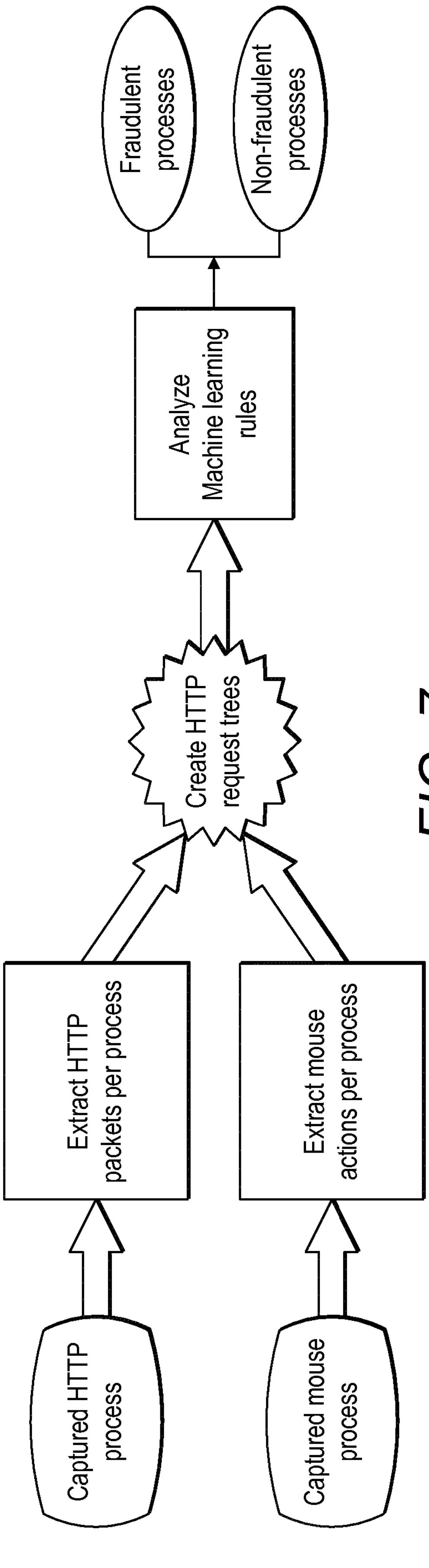

Captured HTTP process
Captured mouse process
Extract HTTP packets per process
Extract mouse actions per process
Create HTTP request trees
Analyze Machine learning rules
Fraudulent processes
Non-fraudulent processes

FIG. 7

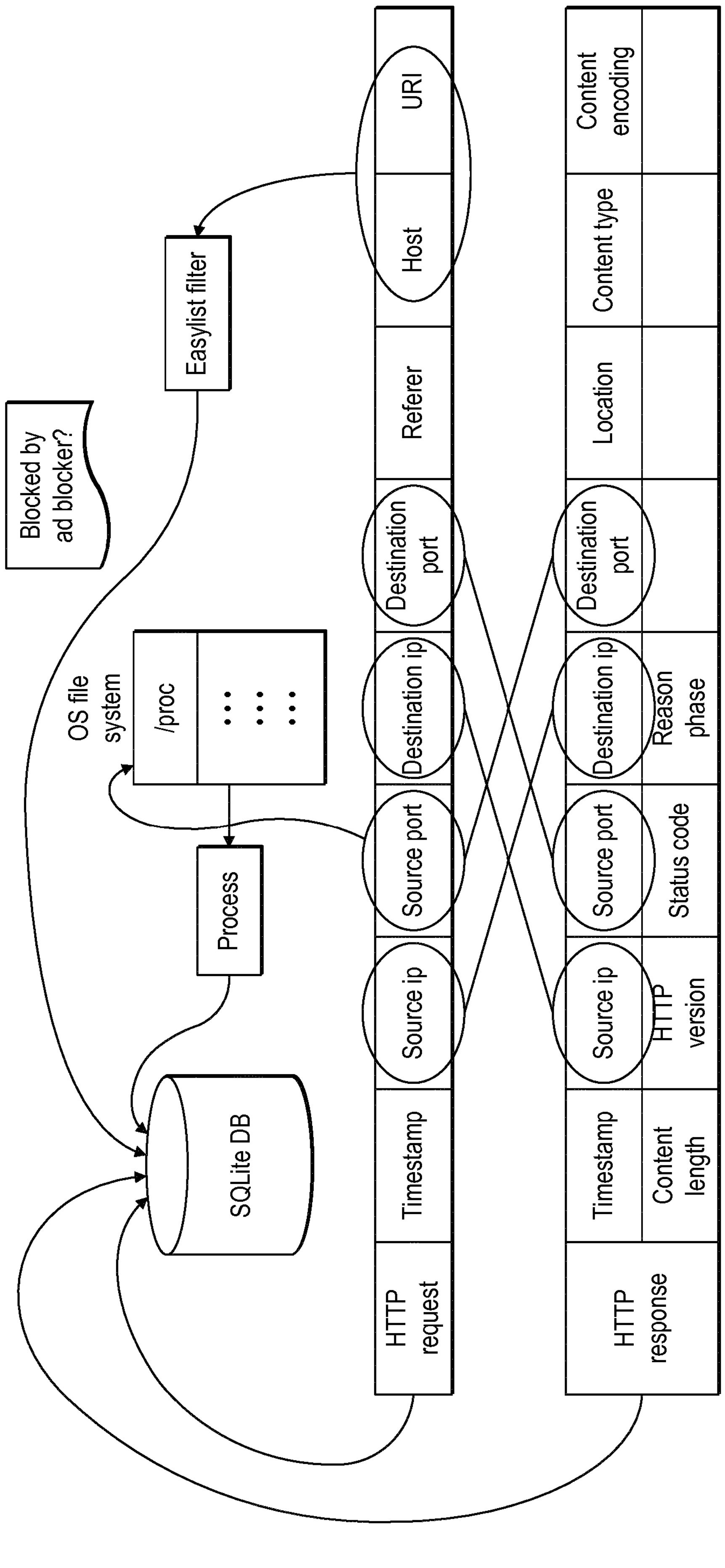

Blocked by ad blocker?
Easylist filter
OS file system
/proc
Process
SQLite DB
HTTP request
Timestamp
Source ip
Source port
Destination ip
Destination port
Referer
Host
URI
HTTP response
Timestamp
Source ip
Source port
Destination ip
Destination port
Location
Content type
Content encoding
Content length
HTTP version
Status code
Reason phase

FIG. 8

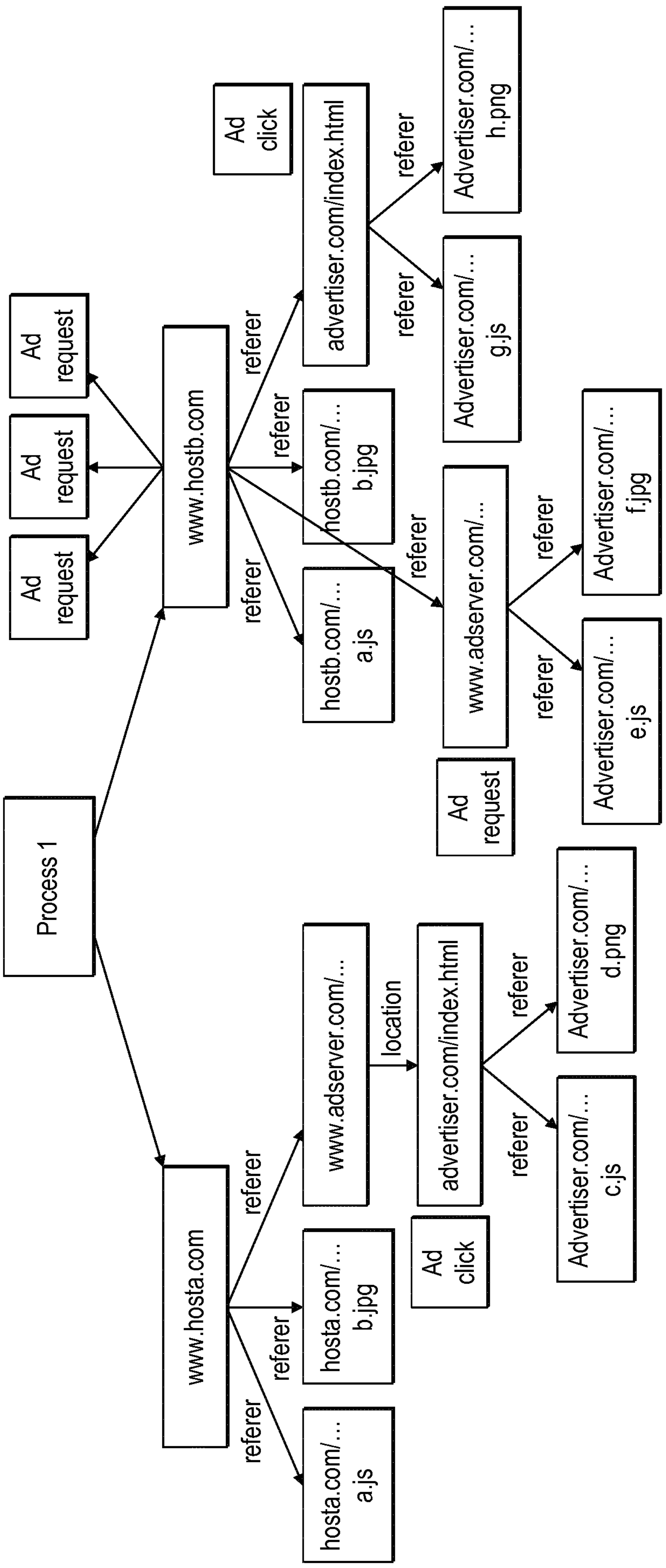

Process 1
www.hostb.com
Ad request
Ad request
Ad request
referer
hostb.com/... a.js
hostb.com/... b.jpg
advertiser.com/index.html
www.adserver.com/...
Ad request
Ad click
Advertiser.com/... g.js
Advertiser.com/... h.png
Advertiser.com/... e.js
Advertiser.com/... f.jpg
www.hosta.com
hosta.com/... a.js
hosta.com/... b.jpg
www.adserver.com/...
location
advertiser.com/index.html
Ad click
Advertiser.com/... c.js
Advertiser.com/... d.png

FIG. 9

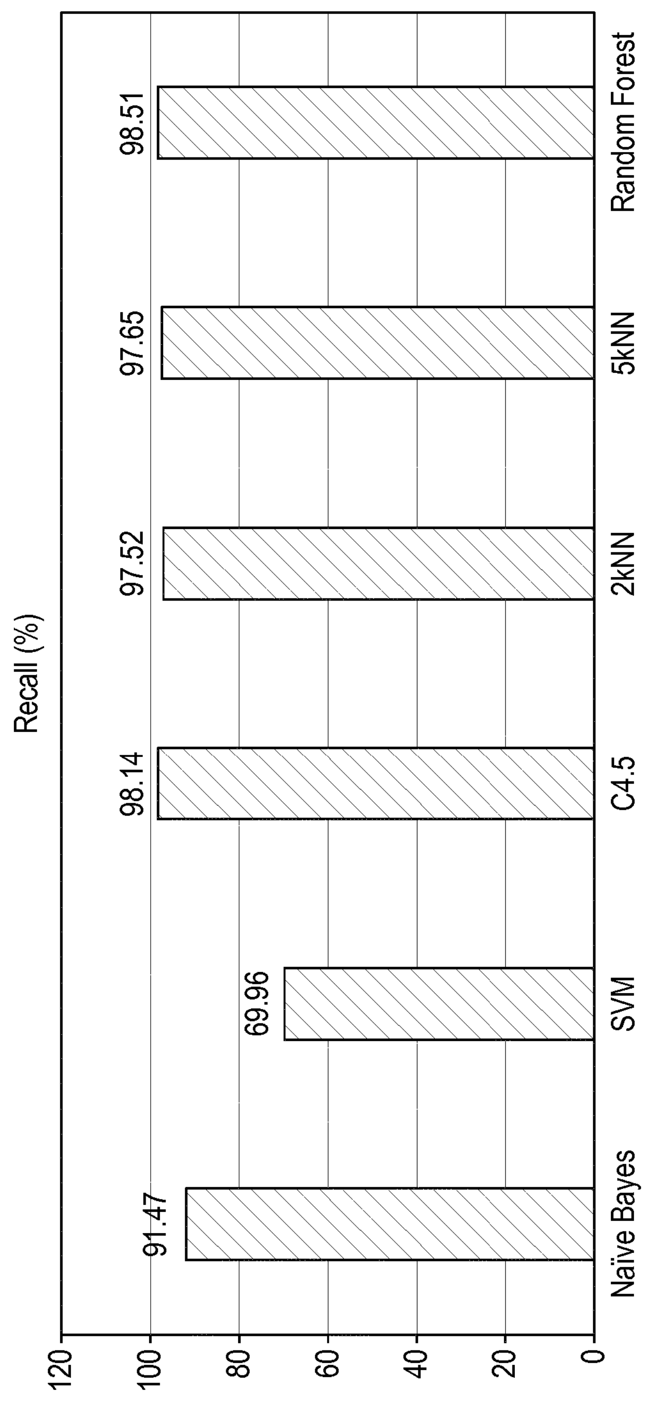

Recall (%)
120
100
80
60
40
20
0
91.47
69.96
98.14
97.52
97.65
98.51
Naïve Bayes
SVM
C4.5
2kNN
5kNN
Random Forest

FIG. 10

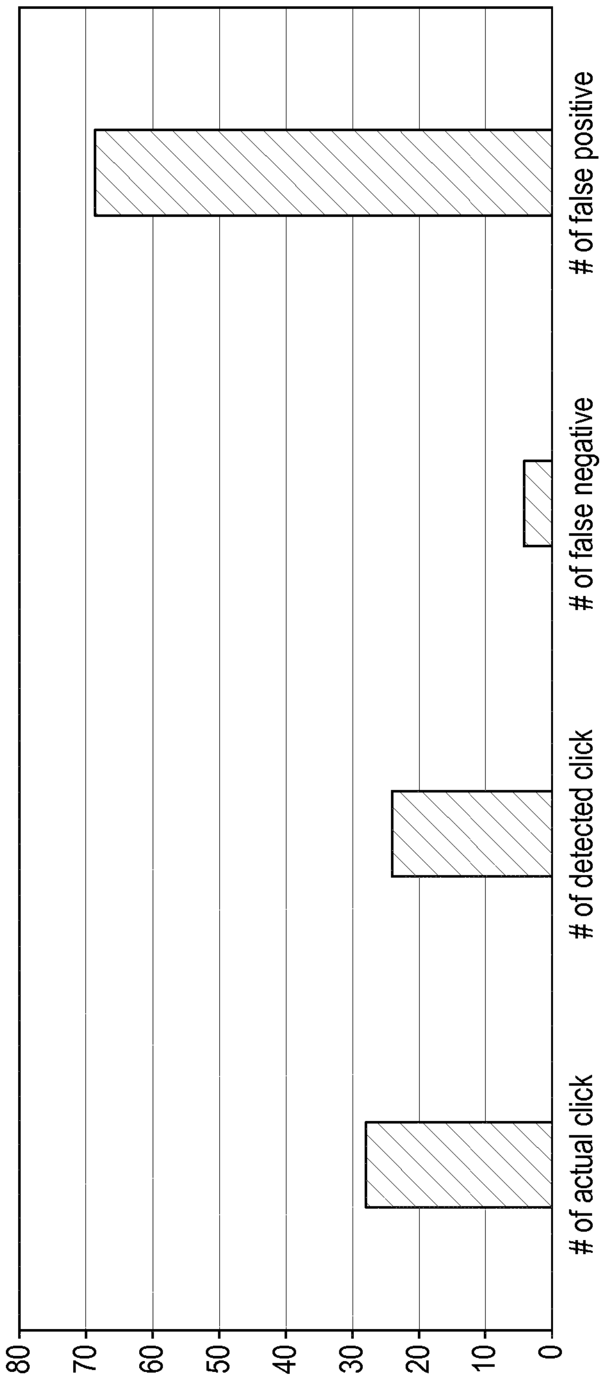

80
70
60
50
40
30
20
10
0
# of actual click
# of detected click
# of false negative
# of false positive

FIG. 11

```
Classifier: notad    Suspicious: yes isAdClick: yes  NumberOfAdinChild: -1 SrcP: 43642
     GET: global.fncstatic.com/static/v/fn-hp/css/ag.home.css?20141021T1229
     Ref:www.foxnews.com/  ResponseLocationHeader:
http://global.fncstatic.com/static/v/fn-hp/css/ag.home.css?20141021T1229    EdgeType:
LOCATION

Classifier: notad    Suspicious: yes isAdClick: yes  NumberOfAdinChild: -1 SrcP: 48773
     GET: www.viarail.ca/sites/all/files/css/css_fd24a8a78a15b3f0afb2d5b475550379.css
     Ref:www.viarail.ca/en/train-
advantages/?utm_campaign=_14q1nonb&utm_medium=_ban&utm_source=_accutab&utm_term=_ros_300
x250_en_150126&utm_content=_kin-tor$41ResponseLocationHeader: -  EdgeType: REFERER

Classifier: notad    Suspicious: yes isAdClick: yes  NumberOfAdinChild: -1 SrcP: 53841
     GET: player.cnevids.com/embed/5480d98f61646d5d5d040000/51799c3f68f9da5d48000002
     Ref:www.wired.com/   ResponseLocationHeader: -  EdgeType: REFERER
```

FIG. 12

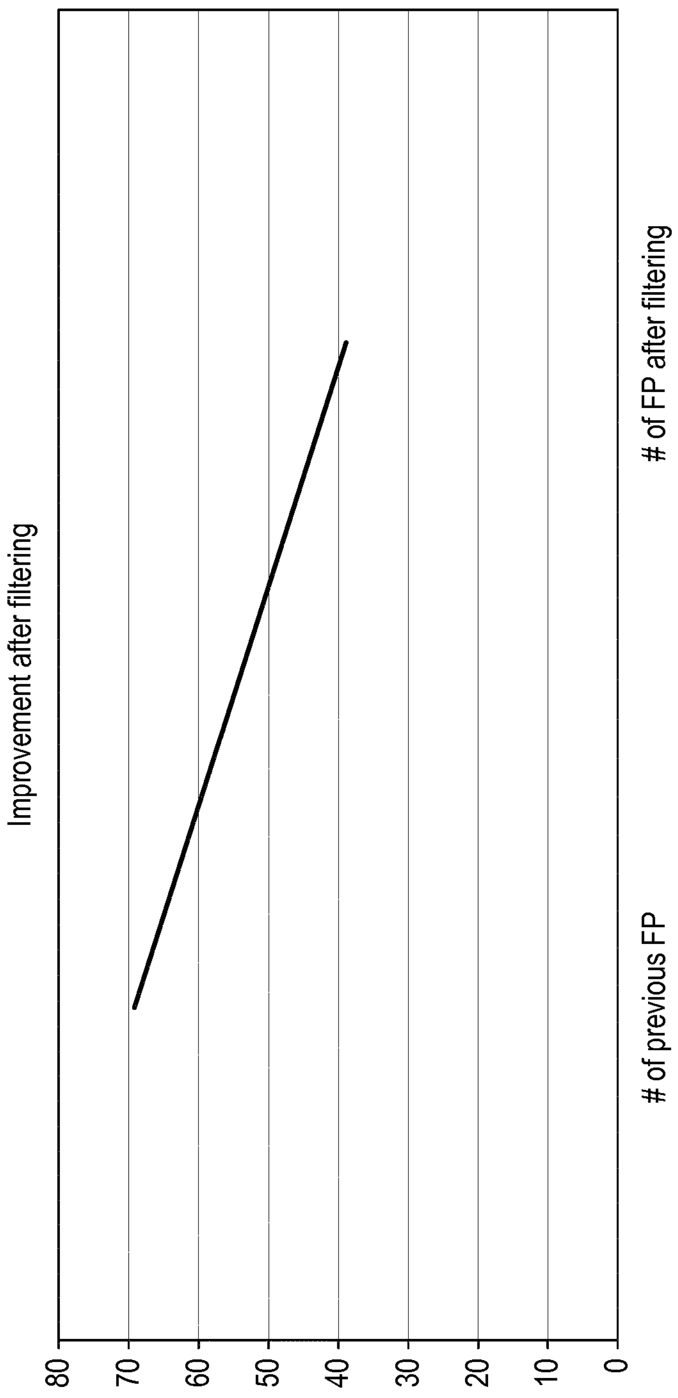

Improvement after filtering
80
70
60
50
40
30
20
10
0
# of previous FP
# of FP after filtering

FIG. 13

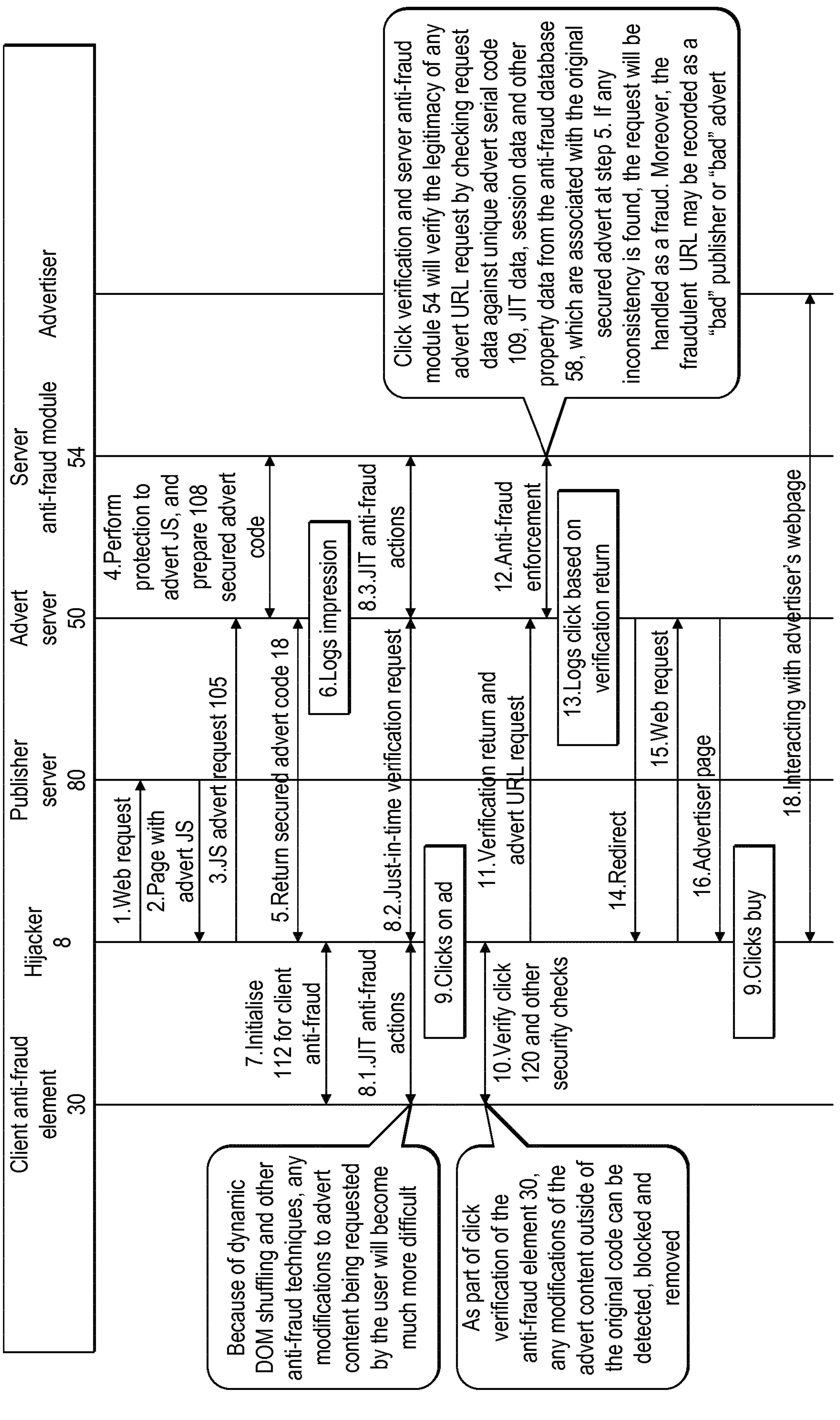

Client anti-fraud element 30
Hijacker 8
Publisher server 80
Advert server 50
Server anti-fraud module 54
Advertiser
1.Web request
2.Page with advert JS
3.JS advert request 105
4.Perform protection to advert JS, and prepare 108 secured advert code
5.Return secured advert code 18
6.Logs impression
7.Initialise 112 for client anti-fraud
8.1.JIT anti-fraud actions
8.2.Just-in-time verification request
8.3.JIT anti-fraud actions
9.Clicks on ad
10.Verify click 120 and other security checks
11.Verification return and advert URL request
12.Anti-fraud enforcement
13.Logs click based on verification return
14.Redirect
15.Web request
16.Advertiser page
9.Clicks buy
18.Interacting with advertiser's webpage
Because of dynamic DOM shuffling and other anti-fraud techniques, any modifications to advert content being requested by the user will become much more difficult
As part of click verification of the anti-fraud element 30, any modifications of the advert content outside of the original code can be detected, blocked and removed
Click verification and server anti-fraud module 54 will verify the legitimacy of any advert URL request by checking request data against unique advert serial code 109, JIT data, session data and other property data from the anti-fraud database 58, which are associated with the original secured advert at step 5. If any inconsistency is found, the request will be handled as a fraud. Moreover, the fraudulent URL may be recorded as a "bad" publisher or "bad" advert

FIG. 14

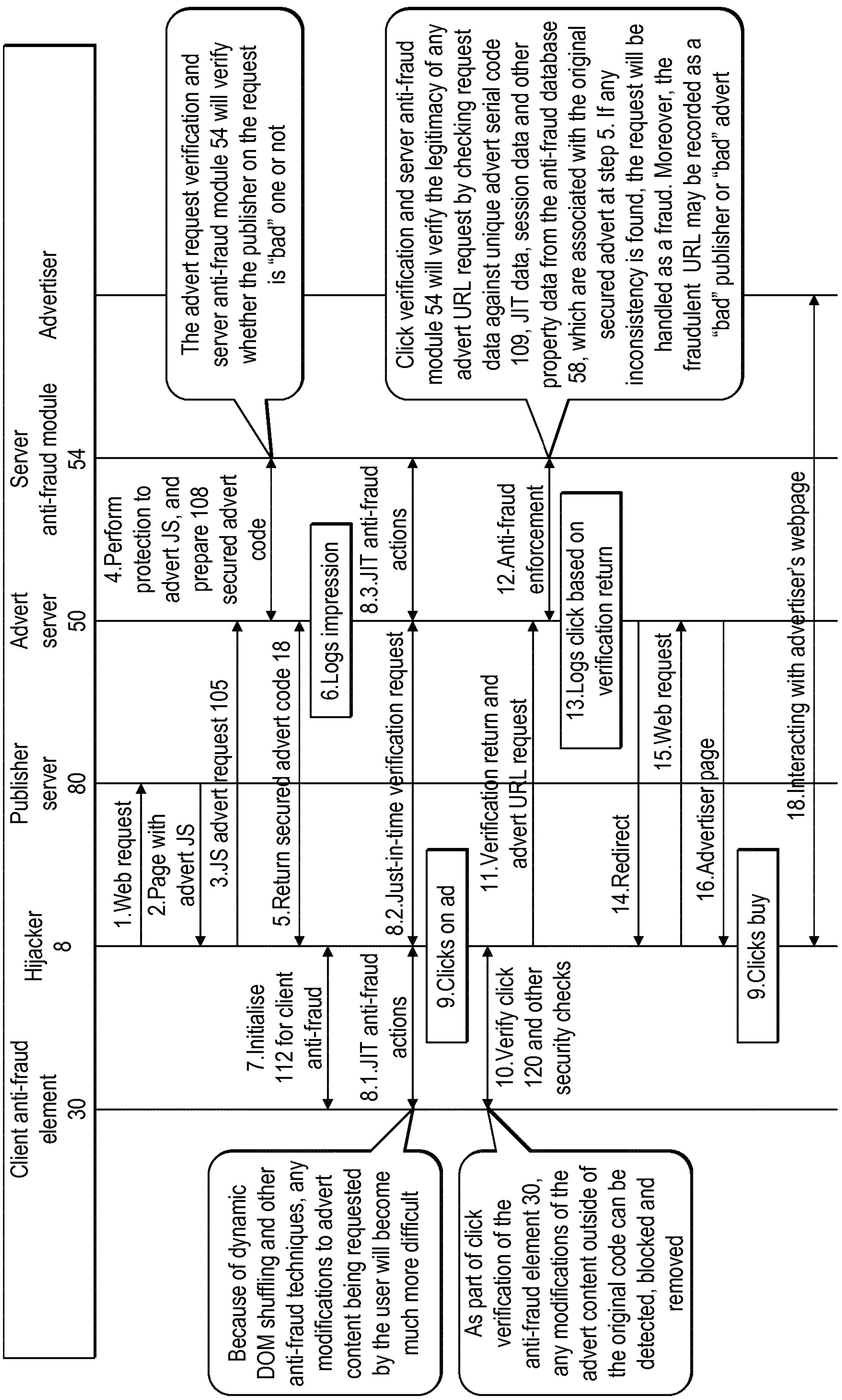

Client anti-fraud element 30
Hijacker 8
Publisher server 80
Advert server 50
Server anti-fraud module 54
Advertiser
1.Web request
2.Page with advert JS
3.JS advert request 105
4.Perform protection to advert JS, and prepare 108 secured advert code
5.Return secured advert code 18
6.Logs impression
7.Initialise 112 for client anti-fraud
8.1.JIT anti-fraud actions
8.2.Just-in-time verification request
8.3.JIT anti-fraud actions
9.Clicks on ad
10.Verify click 120 and other security checks
11.Verification return and advert URL request
12.Anti-fraud enforcement
13.Logs click based on verification return
14.Redirect
15.Web request
16.Advertiser page
9.Clicks buy
18.Interacting with advertiser's webpage
The advert request verification and server anti-fraud module 54 will verify whether the publisher on the request is "bad" one or not
Click verification and server anti-fraud module 54 will verify the legitimacy of any advert URL request by checking request data against unique advert serial code 109, JIT data, session data and other property data from the anti-fraud database 58, which are associated with the original secured advert at step 5. If any inconsistency is found, the request will be handled as a fraud. Moreover, the fraudulent URL may be recorded as a "bad" publisher or "bad" advert
Because of dynamic DOM shuffling and other anti-fraud techniques, any modifications to advert content being requested by the user will become much more difficult
As part of click verification of the anti-fraud element 30, any modifications of the advert content outside of the original code can be detected, blocked and removed

FIG. 15

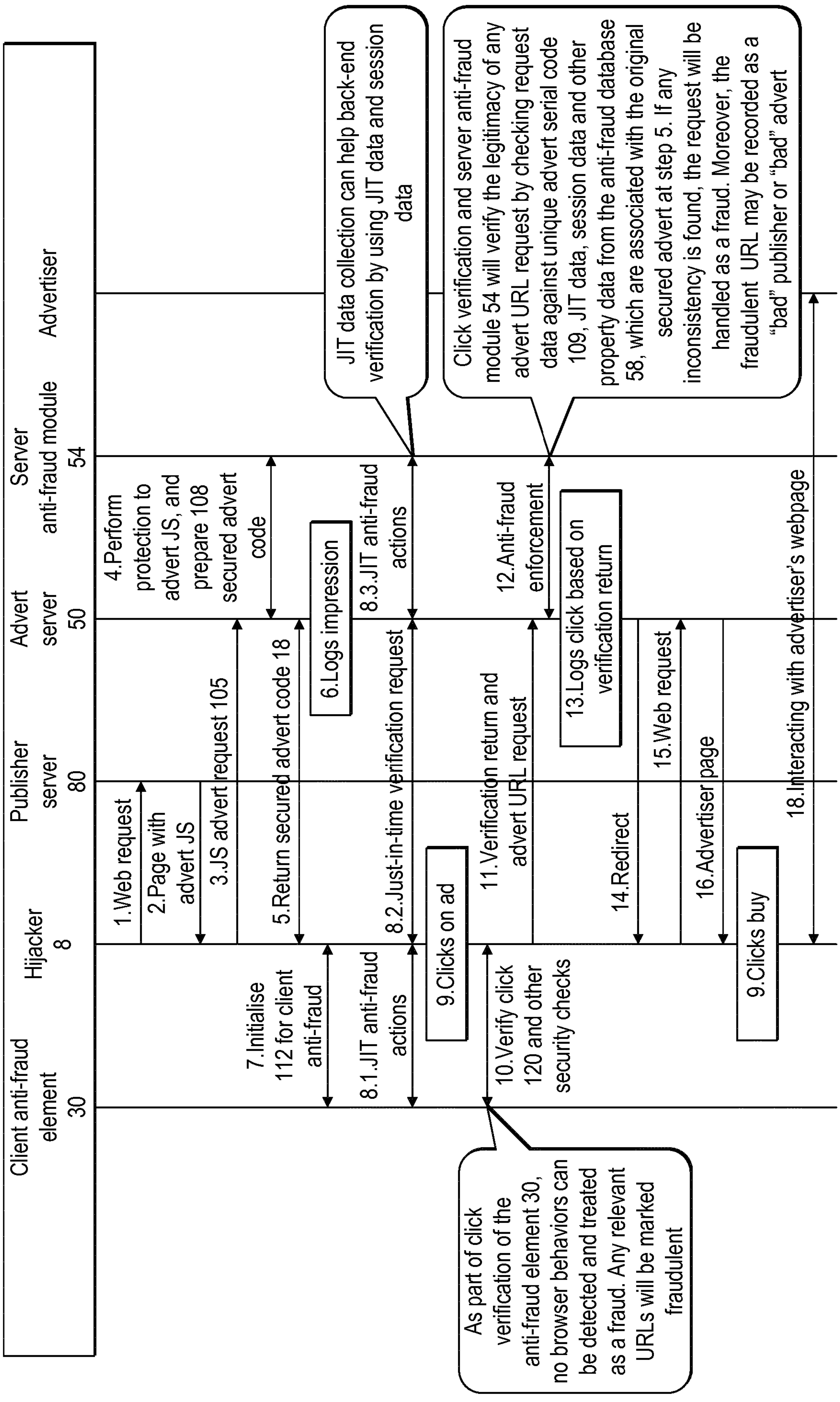

Client anti-fraud element 30
Hijacker 8
Publisher server 80
Advert server 50
Server anti-fraud module 54
Advertiser
1.Web request
2.Page with advert JS
3.JS advert request 105
4.Perform protection to advert JS, and prepare 108 secured advert code
5.Return secured advert code 18
6.Logs impression
7.Initialise 112 for client anti-fraud
8.1.JIT anti-fraud actions
8.2.Just-in-time verification request
8.3.JIT anti-fraud actions
9.Clicks on ad
10.Verify click 120 and other security checks
11.Verification return and advert URL request
12.Anti-fraud enforcement
13.Logs click based on verification return
14.Redirect
15.Web request
16.Advertiser page
9.Clicks buy
18.Interacting with advertiser's webpage
JIT data collection can help back-end verification by using JIT data and session data
Click verification and server anti-fraud module 54 will verify the legitimacy of any advert URL request by checking request data against unique advert serial code 109, JIT data, session data and other property data from the anti-fraud database 58, which are associated with the original secured advert at step 5. If any inconsistency is found, the request will be handled as a fraud. Moreover, the fraudulent URL may be recorded as a "bad" publisher or "bad" advert
As part of click verification of the anti-fraud element 30, no browser behaviors can be detected and treated as a fraud. Any relevant URLs will be marked fraudulent

FIG. 16

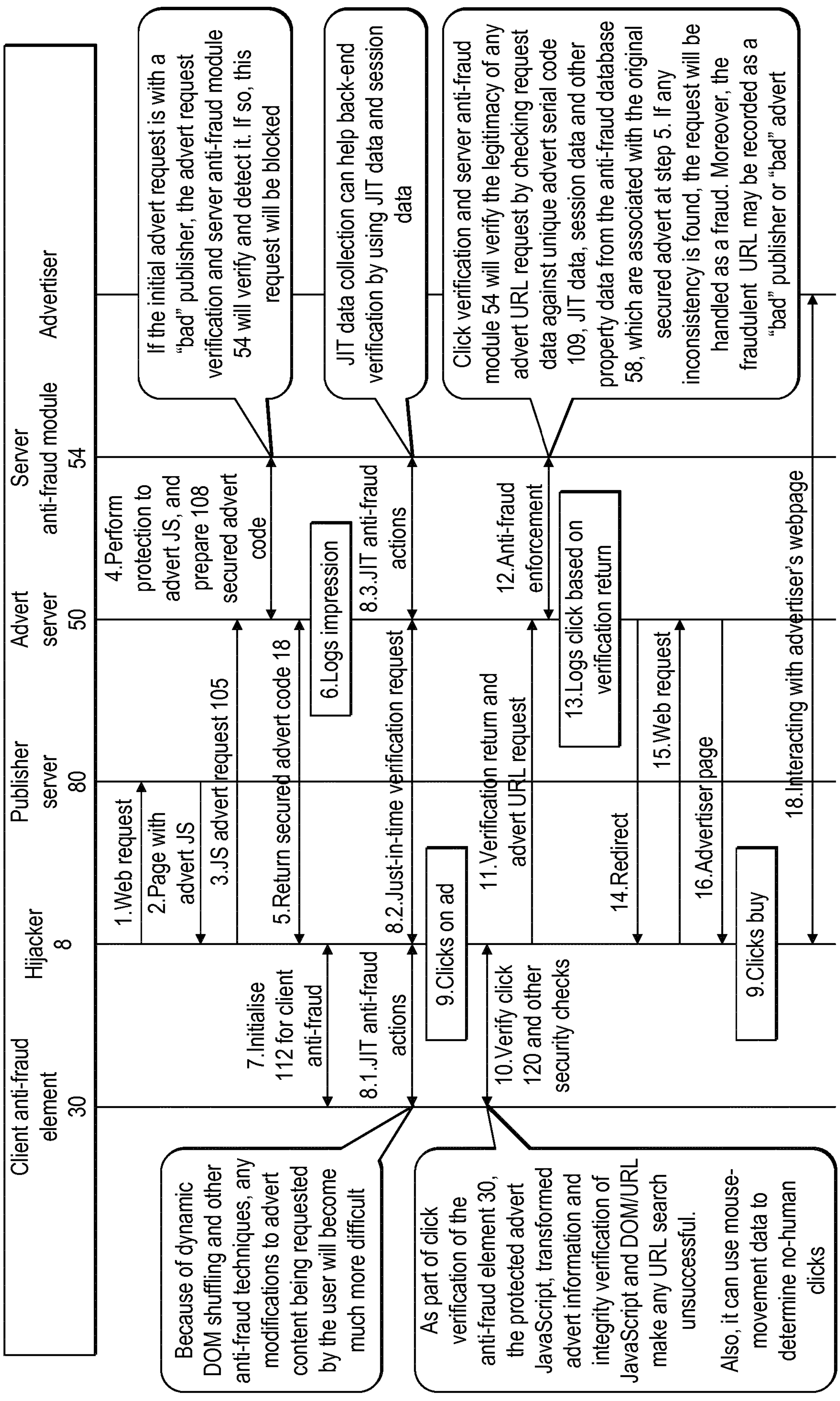

Client anti-fraud element 30
Hijacker 8
Publisher server 80
Advert server 50
Server anti-fraud module 54
Advertiser
1.Web request
2.Page with advert JS
3.JS advert request 105
4.Perform protection to advert JS, and prepare 108 secured advert code
5.Return secured advert code 18
6.Logs impression
7.Initialise 112 for client anti-fraud
8.1.JIT anti-fraud actions
8.2.Just-in-time verification request
8.3.JIT anti-fraud actions
9.Clicks on ad
10.Verify click 120 and other security checks
11.Verification return and advert URL request
12.Anti-fraud enforcement
13.Logs click based on verification return
14.Redirect
15.Web request
16.Advertiser page
9.Clicks buy
18.Interacting with advertiser's webpage
If the initial advert request is with a "bad" publisher, the advert request verification and server anti-fraud module 54 will verify and detect it. If so, this request will be blocked
JIT data collection can help back-end verification by using JIT data and session data
Click verification and server anti-fraud module 54 will verify the legitimacy of any advert URL request by checking request data against unique advert serial code 109, JIT data, session data and other property data from the anti-fraud database 58, which are associated with the original secured advert at step 5. If any inconsistency is found, the request will be handled as a fraud. Moreover, the fraudulent URL may be recorded as a "bad" publisher or "bad" advert
Because of dynamic DOM shuffling and other anti-fraud techniques, any modifications to advert content being requested by the user will become much more difficult
As part of click verification of the anti-fraud element 30, the protected advert JavaScript, transformed advert information and integrity verification of JavaScript and DOM/URL make any URL search unsuccessful.
Also, it can use mouse-movement data to determine no-human clicks

FIG. 17

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2016/057109

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06Q30/02
ADD.

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)
G06Q

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)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2006/265493 A1 (BRINDLEY RICHARD [GB] ET AL) 23 November 2006 (2006-11-23) abstract; figure 2 paragraph [0028] - paragraph [0046]	1-52
X	US 2009/024461 A1 (WILLNER BARRY E [US] ET AL) 22 January 2009 (2009-01-22) abstract; figures 4-8 paragraph [0033] - paragraph [0038] paragraph [0010]	1-52
X	US 2012/173315 A1 (MARTINI EDUARD ERWIN [RO] ET AL) 5 July 2012 (2012-07-05) abstract; figures 6,7,8 paragraph [0086] - paragraph [0105]	1-52
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 April 2016	09/05/2016

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Lopes Margarido, C

1

International application No
PCT/EP2016/057109

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2006/136294 A1 (LINDEN JOHN [US] ET AL) 22 June 2006 (2006-06-22) abstract paragraph [0012] - paragraph [0027] paragraph [0042] - paragraph [0048] -----	1-52

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2016/057109

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006265493 A1	23-11-2006	AU 2006341536 A1	10-01-2008
		BR PI0610014 A2	11-10-2011
		CA 2610438 A1	20-11-2006
		EP 1894158 A2	05-03-2008
		EP 2219149 A1	18-08-2010
		JP 2008541318 A	20-11-2008
		US 2006265493 A1	23-11-2006
		US 2014244382 A1	28-08-2014
		WO 2008004027 A2	10-01-2008
US 2009024461 A1	22-01-2009	NONE	
US 2012173315 A1	05-07-2012	EP 2659418 A1	06-11-2013
		US 2012173315 A1	05-07-2012
		WO 2012089915 A1	05-07-2012
US 2006136294 A1	22-06-2006	US 2006136294 A1	22-06-2006
		US 2013080248 A1	28-03-2013
		US 2015178771 A1	25-06-2015