IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 5, 2016, accepted August 29, 2016, date of publication September 8, 2016, date of current version October 6, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2606478

Improving Home Automation Security; Integrating Device Fingerprinting Into Smart Home

ARUN CYRIL JOSE[1], **REZA MALEKIAN**[1], **(MEMBER, IEEE), AND NING YE**[2,3]
[1]Department of Electrical Electronics and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa
[2]College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
[3]Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Jiangsu Province, Nanjing 210003, China

Corresponding author: R. Malekian (reza.malekian@ieee.org).

This work was supported in part by the National Research Foundation, South Africa, under Grant IFR160118156967, in part by the University of Pretoria's Post Graduate Research Support Bursary, in part by the National Natural Science Foundation, China, under Grant 61572260, Grant 61373017, Grant 61572261, and Grant 61672296, and in part by the Scientific & Technological Support Project of Jiangsu Province under Grant BE2015702, Grant BE2016185, and Grant BE2016777.

ABSTRACT This paper explains the importance of accessing modern smart homes over the Internet, and highlights various security issues associated with it. This paper explains the evolution of device fingerprinting concept over time, and discusses various pitfalls in existing device fingerprinting approaches. In this paper, we propose a two-stage verification process for smart homes, using device fingerprints and login credentials, which verifies the user device as well as the user accessing the home over the Internet. Unlike any other previous approaches, our Device Fingerprinting algorithm considers a device's geographical location while computing its fingerprint. In our device identification experiment, we were able to successfully identify 97.93% of the devices that visited our Webpage using JavaScript, Flash, and Geolocation.

INDEX TERMS Home automation, smart homes, identity management systems, security, access control.

I. INTRODUCTION

The concept of Home Automation was a topic of interest in the Academic arena since the late 1970s, with time and advancement of technology people's expectations about Home Automation and how they should access their home has dramatically changed. The affordability and popularity of electronic devices and internet were contributing factors to this change. The modern Home Automation System [1] is a delicate balance of Ubiquitous Computing Devices [2] and Wireless Sensor/Actor Networks. The added expectations and 'Convenience of Access' has brought new security challenges to the Home Automation front.

Various researchers showed that, there are vulnerabilities in many commonly used devices and technologies in Home Automation. The Wireless Sensor Networks deployed in Home Automation System are vulnerable to various Routing attacks [3] and Wormhole attack [4], communication technologies like ZigBee and 802.15.4 used in Home Automation are vulnerable to Replay Attacks [5]. Various approaches to preserve privacy and security in Wireless Sensor Networks (WSN) were discussed in [6]–[9]. However, the work of Fouladi and Ghanoun [10] shows that some of the security specific products used in homes like the Z-Wave door locks are vulnerable to hacks. The work of Jose and Malekian [11] specifies how the concept of security has changed in modern homes and explains the changing role of a modern Home Automation Security Systems. A modern Home Automation System must identify, alert and prevent an intrusion attempt in a home; it must also try to preserve evidence of the intrusion or attempted intrusion, so the perpetuator can be brought to justice. The 'Convenience of Access' mentioned above is achieved through internet and mobile electronic devices. They allow Home owners to access their home from anywhere in the world at any given time. Connecting Home Automation System to the internet gives an attacker the opportunity to try and gain access to a home from the comforts of their own home. On the contrary, in a non-internet enabled home this could only be done when an attacker is within the proximity of the home's internal network.

Most obvious way to improve security would be to deny access to a home over the internet, but that significantly inconveniences the home inhabitants and the way they access their home and services, this defeats the purpose of Home Automation Systems. So, securing access to a Home over the internet is a vital part in Home Automation Security. This could be established by limiting access to a home over the internet; Access should be limited to a fixed number of trusted people using a fixed number of trusted electronic devices. To achieve this, we have to identify the user as well as the device accessing the home over the internet.

Objectives of our work:

- Successfully identify a device accessing the home over the internet using Device Fingerprinting. Successfully identify a user accessing the home over the internet using his/her login credentials.
- Identify legitimate user even when there are changes in location, browser or other browser specific features, which happens over time.
- Identify malicious devices and create a 'blacklist' consisting of fingerprints of those devices that will not be allowed access to home. Identify legitimate devices and develop a 'whitelist' consisting of fingerprints of devices that are allowed access to the home.

Rest of the paper is organized as follows, Section II discusses the Related Works on device fingerprinting and security issues associated with username and passwords. Section III describes different device fingerprinting parameters used in this paper; Section IV discusses the implementation details along with fingerprinting algorithm and explains the two stage verification process in home automation systems. Section V explains the Mathematical modelling of our work. The Experiment Setup is mentioned in Section VI. Section VII, states our Results and discuss the obtained result and compares them with previous studies. The paper Concludes by indicating future directions our work could take.

II. RELATED WORKS

The concept of cookie was introduced into the context of the web browser in 1994, by Lou Montulli [12], [13]. Cookie allowed webservers to store small amount of data on the visiting user's computer which is sent back to the server upon request. The concept of cookie was quickly embraced by browser manufacturers. Soon after, attackers began to take advantage of cookie's statefull nature. Third-party advertising sites used cookies to track users over multiple websites which encouraged behavioral advertising [14]. This privacy violating behavior caught the attention of the research community [15]–[18], legal community [18] and was a cause of concern among the public [20], [21]. Moreover, cookies are vulnerable to Cross Site Scripting [22] and Cookie Stealing. The concept of cookie was further expanded to Flash cookies [23] and later to 'evercookie' [24] which is almost impossible to remove; this further enhanced the privacy concerns associated with cookies. A cookie-retention study [25] showed that one in three users deleted their first and third-party cookies within a month of visiting a website. The above researches illustrate the privacy, security and unavailability issues associated with using cookies to identify a user. So utilizing cookies in Home Automation to identify a user over the internet doesn't seem like a sensible decision.

The issues associated with cookies prompted researchers and internet advertisers to come up with a new way to tracking internet users. In 2009 Mayer [26] and in 2010 Eckersley [27] demonstrated how features of a web browser can be used to uniquely identify a user without cookies over the internet. Mayer [26] did a study on 1328 web clients. In his study, he hashed the combined contents of navigator, screen, navigator.plugins and navigator.mimeTypes. Using this, he uniquely identified 96% of the web browsers in his study. Eckersley [27] conducted a study on 500,000 users and uniquely identified 94.2% of them. He combined various properties of the web browser and installed plugins, to uniquely identify users. He used Flash and JavaScript to collect the required information from the client machine. In 2012, Yen *et al.* [28] conducted a fingerprint study on month long logs of Bing and Hotmail by using User Agent (UA) string and client's Internet Protocol (IP) address. The authors were able to identify 60 to 70% of users by just using UA string and the accuracy improved to 80% when IP prefix information was combined with the UA string. Eckersley [27] dismisses the use of IP address for fingerprinting as they are "not sufficiently stable." Now a days, internet users try to mask the UA string of their web browser to avoid identification, but the work of Nikiforakis *et al.* [29] illustrated how counterproductive this is and demonstrated spoofing UA string aids in user identification which is contrary to the popular user belief. So, in our work IP prefix was avoided but UA string was utilized for device identification.

Mowery *et al.* [30] proposed a device fingerprinting method exploiting the difference in JavaScript performance profiles among different browser families. Each browser executes a set of predefined JavaScript bench marks and the completion time of each bench mark forms a part of the performance signature of the browser. Using this technique, the authors were able to successfully identify a browser's family 98.2% of the time; the identification process took over 3 minutes to completely execute. A study [31] shows that, an average user views a web page for about 33 seconds. So, fingerprinting based on JavaScript benchmark execution time may not be the solution as it takes too long to identify a client. Moreover, accuracy and detection rate of more specific device fingerprinting attributes such as, operating system, browser version and Central Processing Unit (CPU) architecture is significantly low. The work of Mowery *et al.* [30], also demonstrated how selective enabling/disabling of JavaScript using browser plugins (like 'NoScript' in Firefox) for certain websites could aid in fingerprinting and subsequently helps user identification. Disabling JavaScript completely in the browser would be the way to preserve a user's online privacy but most websites needs JavaScript to function properly. The above research shows, contrary to the popular user belief selectively enabling of JavaScript helps in device identification. This gives a reason for even the most privacy concerned users not to disable java script, so the fingerprinting algorithm discussed in this paper utilizes JavaScript for device identification.

In 2012, Mowery and Shacham [32] proposed a device fingerprinting technique based on the hypothesis that different browsers display text and graphics in a different way. This difference raise from a combination of configuration differences in software, browser, driver, hardware and GPU. To exploit this, the authors rendered text and Web Graphics Library (WebGL) scenes into a HyperText Markup Language 5 (HTML 5) <canvas> element and measured the difference in the resulting pixel map of the canvas for different users. The proposed method cannot differentiate between two web clients with the exact same software and hardware configurations and will not work on older versions of a web browser.

Kohno *et al.* [33] proposed device fingerprinting using clock skew. The authors observed that, there is distinguishable clock skew difference between any two physical devices, and this unique clock skew difference between two devices will remain relatively stable over time. They exploited this clock skew feature to fingerprint a remote physical device by stealthily recording and analyzing its Internet Control Message Protocol (ICMP) or Transmission Control Protocol (TCP) timestamps. Using ICMP and TCP timestamps has their limitation, ICMP timestamps are blocked by numerous firewalls, and some operating systems by default disable TCP timestamps. Later Zander and Murdoch [34] developed a device identification technique with synchronized sampling which significantly reduces the quantization error. It reduces the heavy network traffic which was necessary for previous identifications, their work was the first to calculate clock skew estimation through Hyper Text Transfer Protocol (HTTP) protocol. However, their approach could not be directly implemented at the server side for device identification. Inspired by this work, Huang *et al.* [35] developed a client device identification in cloud computing scenario, which relay on JavaScript to send periodic timestamp back to the server for device fingerprinting. Nakibly *et al.* [36] proposed a device fingerprinting technique by exploiting the uniqueness of hardware features like, speaker/microphones, motion sensors, Global Positioning System (GPS) accuracy, battery charge and discharge time and GPU clock skew. Most of their proposed techniques remain purely theoretical at the moment. Moreover, their fingerprinting approach requires constant user interactions, which is not ideal.

Other attempts in device fingerprinting include Operating System (OS) fingerprinting using popular tools like Nmap, Xprobe etc.; device fingerprinting approach discussed in this paper did not implement OS fingerprinting as most of the firewalls and network administrators prevent this [37] and it requires manual interpretation [38].

Oluwafemi *et al.* [39] discussed the presence of some well-known vulnerabilities in home automation systems, such as, Cross Site Scripting (XSS) [40] and cookie stealing which could be exploited to gain online access to home; authors also demonstrates, how simple devices such as Fluorescent lamps (CFL) connected to a home automation network or internet could be manipulated to cause physical harm to home's inhabitants.

Passwords are always vulnerable to brute force [41], dictionary [42] and rainbow-table attacks [43]. A study done by Kato and Klyuev [44] among 262 University students revealed that, 80% of the passwords were not strong and 40 % of the passwords were reused for different accounts. This concurs with the work of Hart [45] who concludes that 30% of people reused their passwords 4 or more times. The work of Yan *et al.* [46] demonstrated that, average users have difficulty remembering random passwords and people are reluctant to use special characters in their passwords. Various password policies implemented by the administrator further complicates things [47]. So, people tends use grammatical structures in their passwords, the work of Rao [48] illustrates security issues associated with such passwords. Passwords are set and has to be remembered by humans, whose memory for sequences of items are temporally limited [49], with a short term capacity of around seven plus or minus two items [50]. Moreover, humans are vulnerable to social engineering [51]. These human errors, human memory limitations and social engineering compounds to the security issues associated with passwords [52], [53]. Moreover, well known password hacking tools such as 'John the Ripper' or 'Hashcat' also assists an attacker. So passwords alone are not enough to keep access to our homes secure over the internet. This paper proposes a security system with two stage verification, which utilizes password and device fingerprinting before granting access to home.

III. METHOD

From the Related Works in Section II, it is clear that there are well documented security issues associated with implementing just password based user authentication in the home automation scenario. The system is at its most vulnerable when the home is online. Our work utilizes device fingerprinting and legitimate login credentials as a part of double verification process for authorized user and their device identification. Various approaches for Remote Physical Device Fingerprinting are considered before we settled on fingerprinting using JavaScript, Flash and Geo-Location. Our reliance on JavaScript was justified by a study [54] which showed that, 98% of internet users had their JavaScript enabled when they visited Yahoo's homepage. According to Adobe, more than 1 billion devices were using Flash by the end of 2015. The algorithm implemented in this paper avoided using Java Plugin for device fingerprinting because of their known security vulnerabilities. To the best of our knowledge, this is the first attempt that incorporates HTML 5's Geo-Location capability into device fingerprinting.

A. PARAMETERS CONSIDERED FOR DEVICE FINGERPRINTING

The tables given below, Table 1 and Table 2, shows all the JavaScript parameters used for Device Fingerprinting. JavaScript is used to identify browser specific and device specific parameters for device fingerprinting.

TABLE 1. Browser specific parameters using java script.

No.	Parameter	Obtained From
1	Browser Name	navigator.userAgent
2	Browser Version	navigator.userAgent
3	JavaScript Enabled	navigator.javascriptEnabled
4	Flash Enabled	navigator.flashEnabled
5	Cookie Enabled	By actually setting/retrieving and deleting a cookie
6	Local Storage Enabled	By actually setting/retrieving and deleting an item in local storage
7	Mime Length	navigator.mimeTypes.length
8	Mime Type	[EachMimeObject].type
9	Suffix Associated with each Mime Type	[EachMimeObject].suffixes
10	Number of Plugins Associated with each Mime Type	[EachMimeObject]. enabledPlugin.length
11	Plugin Length	navigator.plugin.length
12	Plugin name	[EachPluginObject].name
13	Each Plugin's Version	[EachPluginObject].description
14	Number of Mime Type Associated with each plugin	[EachPluginObject].length

TABLE 2. Device specific parameters using java script.

No.	Parameter	Obtained From
1	OS Name	navigator.userAgent
2	OS Bits	navigator.userAgent
3	Screen Maximum Width	navigator.screen.maxWidth
4	Screen Maximum Height	navigator.screen.maxHeight
5	Screen Current Width	navigator.screen.availWidth
6	Screen Current Height	navigator.screen.availHeight
7	Screen Color Depth	navigator.screen.colorDepth
8	Screen Pixel Depth	navigator.screen.pixelDepth
9	Taskbar Position	Calculated from Maximum Height, Width and Current Height and Width
10	Taskbar Size	Calculated from Maximum Height, Width and Current Height and Width
11	Time zone	navigator.date
12	Country Name	navigator.date
13	Current time	navigator.date
14	Geographical Location (Latitude, Longitude)	navigator.geolocation. getCurrentPosition()

Browser Specific parameters given in Table 1 can be obtained from "navigator.userAgent," "navigator.javascript Enabled," "navigator.flashEnabled," "navigator.mime Types" and "navigator.plugins." "navigator.userAgent" provides information about OS name, OS Bits, Browser Name and Version which can be utilized for fingerprinting and identifying a device, while lesser bit parameters like "navigator.javascriptEnabled," "navigator.flashEnabled" provides 'true' or 'false' values which provides less identifiable information. The algorithm determines whether 'cookies' are enabled by actually setting/retrieving and then deleting the set cookie. 'Local Storage' enabled is also checked in a similar way.

The browser specific parameters mentioned above OS name, OS Bits, browser name, browser version when combined adds to the uniqueness of the fingerprint thus improving the fingerprint accuracy. Other browser specific parameters like, Multi-Purpose Internet Mail Extensions (MIME) length, MIME type, MIME suffixes and their number of associated plugins provides highly identifiable information corresponding to a browser which are utilized for fingerprinting. Total number of installed plugins, plugin name, version of each installed plugins and number of mime types associated with each plugin also contribute to the high accuracy of our fingerprint. The order of the installed plugins retrieved depends on the installation time of each of the individual plugin, as demonstrated by Mayer [26]. A combination of all these parameters are used to develop a client device's fingerprint.

Device specific parameters given in Table 2 can be obtained from "navigator.screen" and "navigator.date" object in JavaScript. Parameters like screen maximum width, screen maximum height, screen current width, screen current height, screen color depth, screen pixel depth does not change even if a user changes their web browser. Position of the taskbar (top/bottom OR left/right) and taskbar size can be deduced from the screen parameters, these two parameters almost never changes. A device's current time, time zone and country name can be obtained from the "navigator.date" object in JavaScript. The OS Name and OS Bits obtained from UA string of the browser are also device specific parameters. Many other device fingerprinting parameters such as, "navigator.language, navigator.product, navigator.appVersion, navigator.appName" etc. can be obtained using JavaScript but these were ignored because they were mostly unreliable and gave inconsistent or false values across different browsers. Moreover, some parameters like "screen.updateInterval, screen.buffer" were browser specific.

We utilized the geo-location feature available in the HTML 5 to improve the accuracy of our fingerprinting algorithm. A lot of finger printable parameters can be gathered from "navigator.geoLocation.getCurrentPosition ()"; they include, latitude, longitude, altitude, accuracy, altitude accuracy, heading, speed. We only utilized two of those parameters, namely latitude and longitude in our fingerprinting algorithm. During the course of our work, it was found that these two parameters are readily available in almost all machines which supports HTML5 as compared to other parameters, which requires constant monitoring and in some cases specific equipment at the client's side. After accurately determining the location, Google Application Program Interface (GoogleAPI) is used to identify the actual country name based on latitude and longitude. The country name from GoogleAPI is compared with that obtained from the date object. The country names must be same, but if there is a mismatch it means the client's "navigator.date" object is intentionally giving misinformation or the client's device is in another time zone, either way in case of country name mismatch, the date parameter from our fingerprinting algorithm is ignored.

A client's device specific screen parameters will remain constant over time, Moreover, a client device's time zone and country name is unlikely to change unless they travel outside the country or changes time zones. Even when that happens, by analyzing their geo-location and date object the real country name and time-zone can be obtained and compared. So when device specific parameters are unavailable the security and device identification capability of the device fingerprinting algorithm decreases.

The table given below, Table 3, shows the Flash parameters used in our fingerprinting algorithm. There are about 38 Flash parameters considered excluding Regular and Non-Regular device fonts. All of the parameters except system fonts are obtained from the 'Capabilities' class in flash. Even though, most of these parameters returned Boolean values with less identifiable qualities, lion share of them were device specific parameters, which when considered as a whole provides reliable information about a device's configuration.

The OS name obtained from flash is compared with those obtained from the "navigator.userAgent." Ideally, the two OS names should match but if they are different it implies the user is using some user agent spoofing techniques even though it is counterproductive in protecting user identity as demonstrated by Nikiforakis *et al.* [29]. So in such a case, user agent and its associated parameters are ignored from the device fingerprinting algorithm. Similarly screen maximum width and screen maximum height obtained from the "navigator.screen" object are compared with the screenResolutionX and screenResolutionY to determine the validity of the "navigator.screen" object. If they are mismatched, it means screen parameters available from JavaScript are not reliable, so all the screen parameters obtained from "navigator.screen" object can be ignored in the algorithm.

The system fonts installed in a device is mostly unique, it depends on user preferences and the presence of different browser plug-ins and software; both regular and irregular device fonts were considered for our fingerprinting algorithm. These fonts and the order in which these font names are retrieved in flash provide highly identifiable information which aids our fingerprinting algorithm as demonstrated by Eckersley [27]. Another method for extracting device fonts is by using JavaScript side channel font detection, this technique requires the names of the fonts to be checked be included in the fingerprint script, this limits font checking to only well-known system fonts. Moreover, it will not allow us to determine the order of fonts in a client's device. So, side channel font detection is not implemented in our algorithm. When some of these parameters mentioned in Table 1, 2 or 3 are unavailable the ability of our algorithm to distinguish between similar machines decreases and the entropy of the generated fingerprint goes down.

Our work tried to detect a client's history to see if he/she has visited a particular Universal Resource Locator (URL) through the vulnerability in Cascading Style Sheets (CSS) as exploited by Jang [55]. The URL checked was not indexed in any search engines and will only be visited by a legitimate user, as he visits his home from a device; it is used as a mechanism to identify a legitimate returning user. This attempt failed and confirms the researcher's notion that, history detection using CSS vulnerability in modern web browsers is not possible. Various attempts to exploit this vulnerability in Mozilla Firefox version 44 and 45, Google Chrome version 48 and 49 and Microsoft Internet Explorer version 11 failed and did not produce the desired result.

TABLE 3. Device fingerprinting parameters using flash.

No.	Parameter	Obtained from flash.system.Capabilities	Return Type
1	AV Hardware Disabled	avHardwareDisable()	Boolean
2	CPU Architecture	cpuArchitecture()	String
3	Has Accessibility	hasAccessibility()	Boolean
4	Has Audio	hasAudio()	Boolean
5	Has Audio Encoder	hasAudioEncoder()	Boolean
6	Has Embedded Video	hasEmbeddedVideo()	Boolean
7	Has IME	hasIME()	Boolean
8	Has MP3	hasMP3()	Boolean
9	Has Printing	HasPrinting()	Boolean
10	Has Screen Broadcast	hasScreenBroadcast()	Boolean
11	Has Screen Playback	hasScreenPlayback()	Boolean
12	Has Streaming Audio	hasStreamingAudio()	Boolean
13	Has Streaming Video	hasStreamingVideo()	Boolean
14	Has TLS	hasTLS()	Boolean
15	Has Video Encoder	hasVideoEncoder()	Boolean
16	Is Debugger	isDebugger()	Boolean
17	Is Embedded in Acrobat	isEmbeddedInAcrobat()	Boolean
18	Language	Language()	String
19	Local File Read Disabled	localFileReadDisabled()	Boolean
20	Manufacturer	Manufacturer	String
21	Max Level IDC	maxLevelIDC()	String
22	Operating System	os()	String
23	Pixel Aspect Ratio	pixelAspectRatio()	Number
24	Player Type	playerType()	String
25	Screen Color	screenColor()	String
26	Screen DPI	screenDPI()	Number
27	Screen Resolution X	screenResolutionX	Number
28	Screen Resolution Y	screenResolutionY	Number
29	Support 32 Bit Processes	Support32BitProcesses()	Boolean
30	Support 64 Bit Processes	Support64BitProcesses()	Boolean
31	Touch Screen Type	touchScreenType()	String
32	Flash Player Version	version()	String
33	Dolby Digital Audio Enabled	hasMultiChannelAudio (flash.media.AudioDecoder.DOLBY_DIGITAL)	Boolean
34	Dolby Digital Plus Audio Enabled	hasMultiChannelAudio (flash.media. AudioDecoder. DOLBY_DIGITAL_PLUS)	Boolean
35	DTS Audio Enabled	hasMultiChannelAudio (flash.media. AudioDecoder.DTS)	Boolean
36	DTS Express Audio Enabled	hasMultiChannelAudio (flash.media. AudioDecoder.DTS_EXPRESS)	Boolean
37	DTS HD High Resolution Audio Enabled	hasMultiChannelAudio (flash.media.AudioDecoder.DTS_HD_HIGH_RESOLUTION_AUDIO)	Boolean
38	DTS HD Master Audio Enabled	hasMultiChannelAudio (flash.media. AudioDecoderDTS_HD_MASTER_AUDIO)	Boolean
39	Regular Device Fonts	Using Action Script in Flash	Array
40	Non-Regular Device Fonts	Using Action Script in Flash	Array

The above mentioned fingerprinting parameters are mainly classified into 9 categories; Parameters from User Agent String, Screen Parameters, Lesser Bit Parameters (cookie enabled, java script enabled, local storage enabled, flash enabled), MIME Parameters, Plugin Parameters, Parameters from Date object, Geo-Location Parameters [56], Flash Parameters and System Fonts. Our device identification proceeds by identifying, verifying, comparing and analyzing various device specific features associated with each of these 9 parameters.

IV. IMPLEMENTATION

A. DEVICE FINGERPRINTING PROCESS

The figure given below, Fig. 1 shows the Device Fingerprinting process in the proposed system. When a user wishes to access the home over the internet, he requests the login page from the server, the server then returns the login page along with the fingerprint java script. The user provides the login credentials along with the fingerprint of the device he is using. The login credentials are verified, if the verification is passed, then the gathered device fingerprint is analyzed to see if there are enough device fingerprinting parameters available to provide a comprehensive fingerprint of the user device. If not, the client is requested to enable his Flash, JavaScript and Geo-location for accurate fingerprinting at the login page again.

There are two fingerprint lists in our database, whose entries are accumulated over time. The 'whitelist' is a list of approved or authorized device fingerprints belonging to legitimate users. Client devices with fingerprints in the whitelist are allowed access to the home after login credential verification. The 'blacklist' is a list of unauthorized or malicious device fingerprints belonging to potential attackers who tried to gain access to the home. Client devices with fingerprints in the blacklist are denied access to the home even if their login credentials are correct.

If the login credential are matched and there are sufficient fingerprinting parameters and the Device Fingerprint is not in our '*whitelist*' and '*blacklist*', then the client should be verified by some other more direct method in order to assure legitimacy. A simple and safe method would be make contact with the client using a phone call to the registered mobile number of the client and verify it is him trying to login to his home. Another alternative is, the server generates a One Time Password (OTP) and sent it to the legitimate user's registered mobile number via Short Message Service (SMS), which the user enters in the website and thus the legitimacy of the user is verified. When a new device's legitimacy is verified, user is asked, if he wants to add the device's fingerprint into the whitelist. A user adds a device's fingerprint to the whitelist, if that device is his own or it is a trusted third party device which is often used to access the home like the clients office computer. Irrespective of the user's choice to add/not add a fingerprint into the database, he is allowed access to the home after direct verification.

During direct verification via phone or OTP, if the verification fails it means the user trying to access the home is not a legitimate user. So that device's fingerprint is added to the blacklist. It also means the login credentials of a legitimate user is compromised, so he is also asked to change them. User devices with continuous and repeated failed login attempts are also added to the blacklist as they are trying to guess the login credentials. When a device tries to access the home whose fingerprint is in the blacklist, it is immediately denied access to the home even if the login credential is correct. This way our proposed system identifies an attacker's device and denies access to the home without bothering the user. The flow chart of the two stage verification process is given in Fig. 3.

OTPs can be easily generated by the server and are short lived. OTPs can only be used once, so even if an attacker managed to record the OTP which is already used he will not be able to abuse it, thus defending against password replay attacks. If a legitimate user makes a mistake when entering the OTP the device's fingerprint is blacklisted and he is denied access to the home from that device for the time being. If the user wants to regain access to the home from a blacklisted device, he has to delete a device's fingerprint from the blacklist of the webserver's database. In order to prevent malicious behavior and to improve home security this can only be done from inside the home where the webserver is physically located.

B. DEVICE FINGERPRINT ALGORITHM

As discussed in Section III. A, the 9 device fingerprinting parameters mainly considered in the device identification algorithm are —-:

User Agent Parameters: These are parameters obtained from 'navigator.userAgent', they are namely Browser name, Browser version, OS name, OS Bits.

Screen Parameters: These are parameters obtained from 'navigator.screen' object, they are namely Screen maximum width, Screen maximum height, Screen available width, Screen available height, Screen color depth, Screen pixel depth, Taskbar position, Taskbar size.

Lesser Bit Parameters: Lesser bit parameters provide very little identifiable information about a client's device. They are namely Cookie enabled, Local storage enabled, Flash enabled, JavaScript enabled.

MIME Parameters: These parameters are obtained from 'navigator.mimeTypes', they are namely Mime length, Mime type, Suffixes associated with each mime type and Plugins associated with each mime type.

Plugin Parameters: Plugin parameters are obtained from 'navigator.plugin', they are namely Plugin length, Plugin name, Version of each plugin, Number of mime types associated with each plugin.

Date object Parameters: A client device's time zone, country name and current time can be obtained from the 'navigator.date' object.

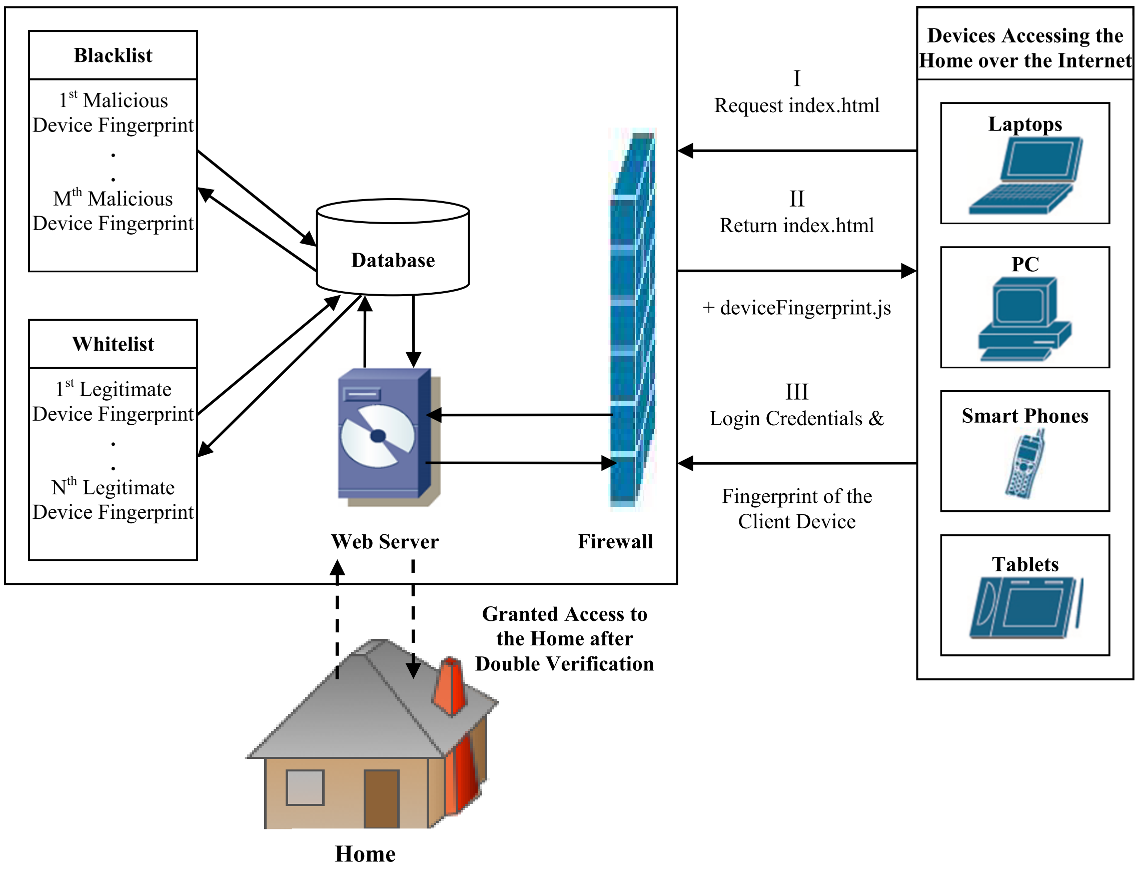
Blacklist
1st Malicious Device Fingerprint
Mth Malicious Device Fingerprint
Whitelist
1st Legitimate Device Fingerprint
Nth Legitimate Device Fingerprint
Database
Web Server
Firewall
Devices Accessing the Home over the Internet
Laptops
PC
Smart Phones
Tablets
I Request index.html
II Return index.html + deviceFingerprint.js
III Login Credentials & Fingerprint of the Client Device
Granted Access to the Home after Double Verification
Home

FIGURE 1. Logical diagram of the proposed system.

Geo-Location Parameters: Geo-Location parameters can be obtained from 'navigator.geolocation.getCurrent Position()', they include the client's current latitude and longitude and the country name corresponding to the latitude and longitude obtained from the Google API.

Flash Parameters: All the flash parameters are obtained from the 'flash.system.Capabilities' class when a client has Flash installed on their browser. The first 38 Parameters mentioned (excluding regular and non-regular device fonts) in Table 3 are the Flash parameters considered in the algorithm.

System Fonts: The name and number of fonts installed at a client's machine can be obtained from Flash, using 'flash.system.Capabilities' class. This includes Regular device font length, Non-Regular device font length, names of Regular device fonts installed and names of Non-Regular device fonts installed.

These parameters have different significance depending on the amount of information they have about a client device. Each of these 9 parameters are assigned scores depending on the varying degree of similarity with existing device fingerprints in our database. These computed scores will determine the probability match corresponding to each of these parameters. The algorithm determines if a device's fingerprint is present in the database if the total probability score is greater than or equal to the threshold probability score. Our Device Fingerprinting Algorithm is given in Fig. 2.

C. FEATURES OF OUR WORK

The device fingerprinting technique discussed in this paper was designed for home automation systems with security as the primary objective. Our work attempts to identify the person operating the device as well as the device used to access the home; this is achieved through a two stage verification process. Comparing and verifying different parameters like OS name in UA string, and screen maximum width and height in Screen parameter, with those from flash, helps us to establish the legitimacy of UA and Screen parameters. Moreover, getting the country name from Google API by utilizing the latitude and longitude obtained from Geo-Location and comparing the country name with the country name in the date object helps us to determine the validity of the date object and time zone. These validations safeguard against parameter spoofing and enhances security.

Hash function can be used to encrypt the device fingerprinting parameters (with a few exception where version number has to be checked) to protect against eavesdropping attack or man in the middle attack attempts. The authenticity of the JavaScript used for fingerprinting can be ver-

```
Step 1: Begin
Step 2: Obtain the device fingerprint from the client using java
        script, flash and geo-location.
Step 3: If (at least 7 out of the 9 device fingerprinting parameters
        are available), then, Step 4 else, Step 10.
Step 4: Analyse and compare each device fingerprinting
        parameter with the fingerprints in the whitelist and
        generate the parameter score corresponding to each of the
        available parameters.
Step 5: Compute the probabilities corresponding to each of the
        parameters based on the parameter score from Step 4.
Step 6: Analyse each probability score and compute the total
        probability score corresponding to the client's fingerprint.
Step 7: If (total probability score >= threshold probability) then,
        Step 8 else, Step 9.
Step 8: Device Fingerprint match found. Do Step 11.
Step 9: No Device Fingerprint match found, check the blacklist for
        malicious device's fingerprint match. Contact the user if
        fingerprint not in the blacklist. Do Step 11.
Step 10: Ask user to enabled JavaScript, Flash and Geo-Location
         so that parameters for device fingerprinting can be
         gathered and return to login page.
Step 11: End
```

FIGURE 2. Device fingerprinting algorithm.

ified by java script self-evaluation using Message-Digest algorithm 5 (MD5) checksum.

The proposed system gives clients accessing the home a reason to enable Flash, JavaScript and Geo-Location and encourage device fingerprinting, as it improves the security of their home over the internet. Using blacklist and whitelist the proposed home security system could identify and distinguish between legitimate and attacker device fingerprints, which significantly reduce the need to contact the user every time a fingerprint mismatch occurs, this improves user convenience.

V. MATHEMATICAL MODELLING

The information contained in a device fingerprint can be calculated using Entropy; Shannon entropy H (X) of a discrete variable X with possible values $\{x_1, x_2 \ldots\ldots x_n\}$ is given by the equation:

$$\mathrm{H}\,(\mathrm{X}) = \sum_{i=1}^{n} \mathcal{P}\,(x_i)\, I\,(x_i) = -\sum_{i=0}^{n} \mathcal{P}\,(x_i) \log_b \mathcal{P}\,(x_i)$$

where $\mathcal{P}\,(x_i)$ is the probability of each value and $I(x_i)$ is the information content and 'b' is the base of the logarithm; Shannon Entropy is calculated with logarithm to the base 2.

When all parameters are available the fingerprint algorithm mentioned in this paper gave an entropy of around 22.57 bits; it drops down to around 22.47 bits when the geographical location parameters are unavailable. It further drops to 21.25 bits when screen parameters are inaccessible and to 21.25 bits when user agent parameters are unavailable. Fig. 4 shows the Shannon Entropy of our proposed fingerprinting algorithm when all fingerprinting parameters are available and various parameters are unavailable.

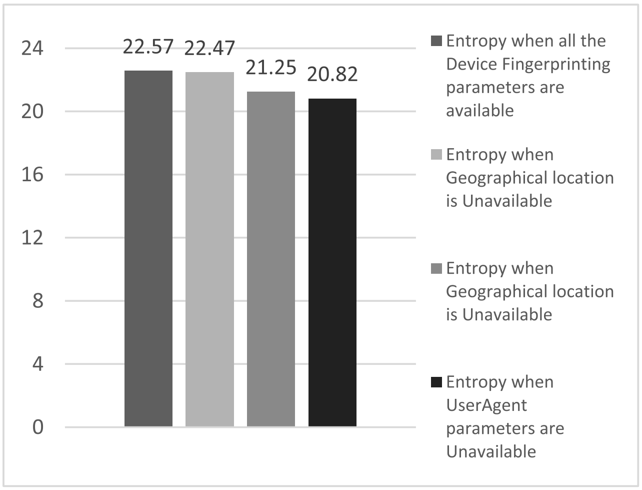

FIGURE 4. Shows the entropy of the fingerprinting algorithm when all fingerprinting parameters are available and various parameters are unavailable.

VI. EXPERIMENT

The experiment discussed in this paper tried to identify a physical device accessing the test website www.fingerprintmydevice.com uniquely using the Device Fingerprinting algorithm developed and mentioned in section IV.B.

Java is used to provide the programming language platform for our work, Java Servlet was utilized for proper server implementation. The Servlet was implemented using Apache Tomcat version 7.0. The work was hosted on a shared server running Linux Centos operating system version 6.8. The server has 18GB RAM and 1TB Hard disk with Intel(R) Xeon(R) CPU E5-2620 0 processor operating at 2.00GHz. Our work was implemented using a database consisting of four tables. The database implementation at the server was done using MySQL 5.1.73.

For simplicity our device identification attempt was restricted to PCs and laptops. Our algorithm was developed to work with and identify devices that uses the 4 most popular web browsers, namely, Mozilla Firefox, Google Chrome, Microsoft Internet Explorer or Edge and Apple Safari. Since 2008, over 95% of internet users use either one of these four web browsers [57]. Access to the website was also restricted to trusted people in order to maintain the accuracy of the collected data. The data was collected between May 2016 and July 2016.

VII. RESULT AND ANALYSIS

The test website www.fingerprintmydevice.com received 283 hits between May 2016 and July 2016, out of them 97 were unique devices. Devices that visited our test website belong to different countries like South Africa, India, Canada, United States etc. The test website saw visits from devices with a wide range of operating systems like, Windows XP,

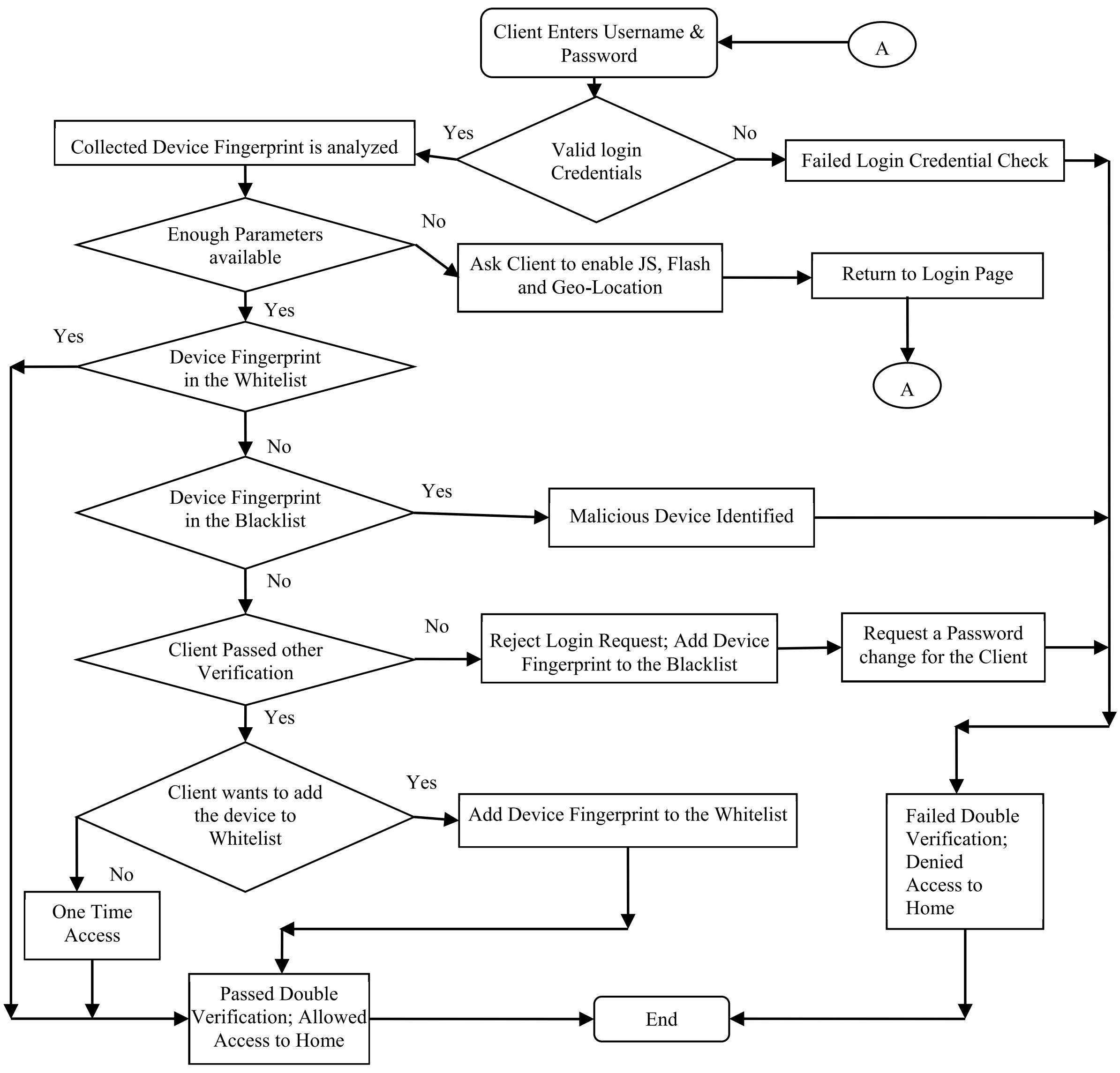
Client Enters Username & Password
A
Valid login Credentials
Yes
No
Collected Device Fingerprint is analyzed
Failed Login Credential Check
Enough Parameters available
No
Ask Client to enable JS, Flash and Geo-Location
Return to Login Page
A
Yes
Yes
Device Fingerprint in the Whitelist
No
Device Fingerprint in the Blacklist
Yes
Malicious Device Identified
No
Client Passed other Verification
No
Reject Login Request; Add Device Fingerprint to the Blacklist
Request a Password change for the Client
Yes
Client wants to add the device to Whitelist
Yes
Add Device Fingerprint to the Whitelist
Failed Double Verification; Denied Access to Home
No
One Time Access
Passed Double Verification; Allowed Access to Home
End

FIGURE 3. Flow chart of double verification process.

Windows 7, Windows 8, Windows 10, Mac OS and Linux in our test dataset. The machines that visited our test website used either Internet explorer, Firefox, Chrome or Safari.

Out of the 97 unique devices visited, the algorithm was able to uniquely identify 95 devices, which accounted for 97.93% of the total devices visited. Our algorithm failed to uniquely identify 2 machines. The 2 machines that our algorithm failed to identify had both flash and geolocation enabled and were identical in almost every aspect. They were machines from the University library with cloned hard disks so their OS, browsers, plugins, mime and all other browser specific and device specific information were identical. Moreover, their screen sizes, pixel depth and resolution were identical as well. Since both these machines were located very close in the University library their geolocation parameters were also indistinguishable. The graph given in Fig. 5, shows our result.

Out of the 95 uniquely identified machines, 58 of them had their Geolocation enabled and our script was able to precisely determine their latitude and longitude. 71 out of the 95 machines had their flash enabled and we successfully collected the 40 flash parameters including system fonts from these devices. 34 machines had both Flash and Geolocation enabled. Fig. 6, shows the number of devices that has enabled flash, geolocation and both.

The algorithm was able to successfully identify 97.93% of the machines that visited our test website using our device fingerprinting algorithm. A total of 14 browser specific parameters, 14 device specific parameters and 40 flash parameters were used to develop a device's fingerprint in an ideal case when all the parameters are available.

Our algorithm allows a 12 degree of variation in both latitude and longitude, so it was able to successfully identify a

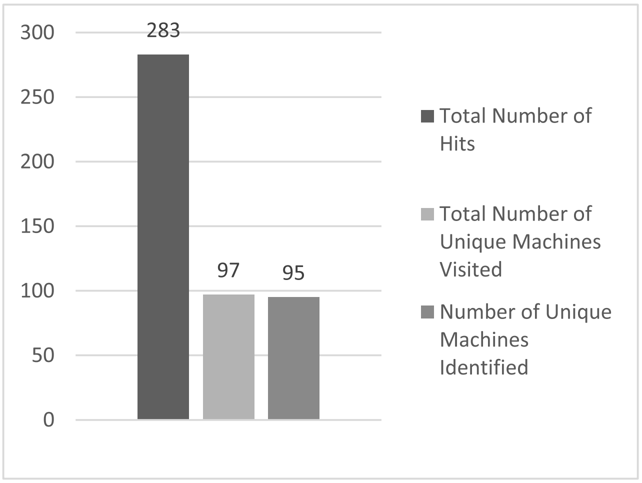

FIGURE 5. Shows the number of total hits, number of unique device's visited our website along with number of devices uniquely fingerprinted and identified.

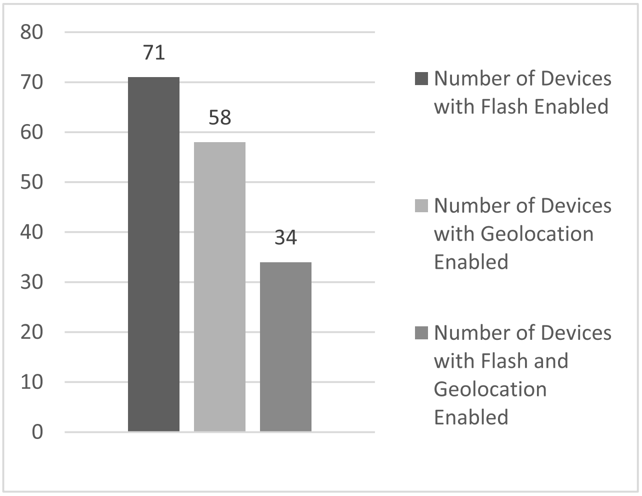

FIGURE 6. Shows the number of devices that has enabled flash, geolocation and both.

machine's fingerprint even when their geographical location was changed. The proposed algorithm is employed for device identification in smart homes, so a significant change in physical location or a change in time zone of a user device is unlikely. When a legitimate user is going abroad, it will result in a significant change in his geographical location and time zone, he has to make the necessary adjustments in his device identification system to still have access to his home via internet. Other modifications in the algorithm can be made specifically for legitimate users who frequently travels abroad.

Even though the version number of a particular plugin or the browser changed with a software upgrade, our algorithm was able to account for those changes and match the information with the corresponding fingerprint. The system fonts also changed over time due to the installation or uninstallation of a plugin or software but a dramatic change in the number or names of the fonts were not seen in the data.

The 97.93% device identification accuracy when combined with legitimate login credentials provide substantial security to modern smart homes when they are accessed over the internet.

Mayer's [26] study provided an accuracy of 96% while our proposed device fingerprinting algorithm provided an accuracy of 97.93%. Mayer utilized screen parameters, mime parameters, plugin parameters and other browser specific parameters for his identification. In our work, in addition to these parameters we considered flash and geolocation parameters for device identification and received a better device identification accuracy.

Compared to our algorithm Eckersley's [27] study provided a lower device identification accuracy of 94.2%. He utilized java script and Flash for device identification in his study. The proposed work in this paper uses geolocation fingerprinting in addition to flash and java script.

The approach of Yen *et al.* [28] provided a device identification accuracy of 80% at best when they combine user agent parameter with the IP prefix information, which is less than the identification capability of our proposed work. The author only utilized user agent parameters along with IP prefix information to develop their fingerprint, while our proposed work used 68 device fingerprinting parameters (including user agent) to develop our device fingerprint. The paper did not consider any IP prefix information as they are not reliable and can be easily spoofed. Fig. 7 compares the identification accuracy of the fingerprinting techniques proposed by Mayer [26], Eckersley [27], Yen *et al.* [28] and our proposed work.

The work of Eckersley [27] had 18.1 bits of entropy while our proposed work has a better entropy of 22.57 bits. The work of Yen *et al.* [28] had an entropy of 20.29 bits at best when IP addresses were combined with UA information which is lower in comparison with our obtained

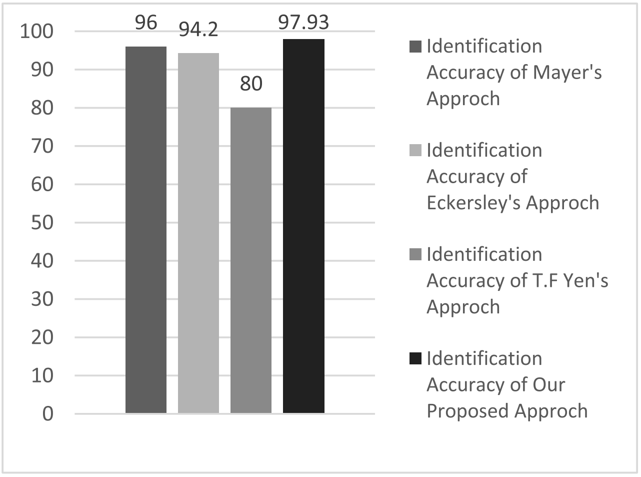

FIGURE 7. Shows the identification accuracy of the fingerprinting techniques proposed by Mayer [26], Eckersley [27], Yen *et al.* [28] and our proposed work.

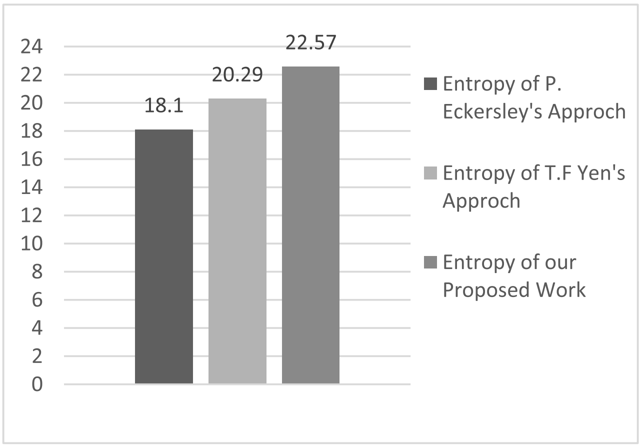

FIGURE 8. **Shows the entropy values of Eckersley [27] and Yen *et al.* [28] and our work.**

entropy of 22.57 bits. Fig. 8 compares the entropy values of Eckersley [27] and Yen *et al.* [28] with our work.

Moreover, none of the previous device fingerprinting approaches include any form of device fingerprinting parameter validation to defend against parameter spoofing. In the proposed work, user agent, screen parameters, date object and flash parameters are validated, which increases the legitimacy of the generated fingerprint.

VIII. CONCLUSION

The device fingerprint along with username/password based security proposed in this paper, enables the verification of user as well as the device used to access the home, which significantly improves home security when they are accessed over the internet. In our work, the device fingerprinting algorithm was able to uniquely identify 97.93% of devices accessing our test website with an entropy of over 22 bits. Unlike any previous approaches to device fingerprinting, we use geolocation data in our algorithm which improves the fingerprint accuracy. The UA verification, screen parameter verification and client's date object verification proposed in our work drastically improves the legitimacy of the fingerprints generated.

For future works, we plan to improve the identification accuracy of our fingerprints when our website is visited using different web clients from the same machine; Adding more fingerprint parameters such as clock skew and other hardware specific parameters for improving the accuracy of the device fingerprint is also an idea worth pursuing.

ACKNOWLEDGMENT

The author would like to thank Ms. Minna Roselin Cherian for proof reading the paper.

REFERENCES

[1] R. Malekian, D. C. Bogatinoska, A. Karadimce, N. Ye, J. Trengoska, and W. A. Nyako, "A novel smart ECO model for energy consumption optimization," *Elektronika Elektrotechnika*, vol. 21, no. 6, pp. 75–80, 2015.

[2] S. Jie, T. Peng, X. Yuan, J. Xiangjun, and R. Malekian, "An improved synchronous control strategy based on fuzzy controller for PMSM," *Elektronika Elektrotechnika*, vol. 20, no. 6, pp. 7–23, 2014.

[3] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Netw.*, vol. 1, nos. 2–3, pp. 293–315, Sep. 2003.

[4] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 370–380, Feb. 2006.

[5] J. Wright, "Practical ZigBee exploitation framework," ToorCon, San Diego, CA, USA, Tech. Rep., Oct. 2011.

[6] T. Gong, H. Huang, P. Chen, R. Malekian, and T. Chen, "Secure two-party distance computation protocol based on privacy homomorphism and scalar product in wireless sensor networks," *Tsinghua Sci. Technol.*, vol. 21, no. 4, pp. 385–396, 2016.

[7] S. Xie and Y. Wang, "Construction of tree network with limited delivery latency in homogeneous wireless sensor networks," *Wireless Pers. Commun.*, vol. 78, no. 1, pp. 231–246, 2014.

[8] P. Guo, J. Wang, X. H. Geng, C. S. Kim, and J.-U. Kim, "A variable threshold-value authentication architecture for wireless mesh networks," *J. Internet Technol.*, vol. 15, no. 6, pp. 929–936, 2014.

[9] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2015.

[10] B. Fouladi and S. Ghanoun, "Security evaluation of the Z-wave wireless protocol," *Black Hat USA*, vol.1 pp. 1–6. Aug. 2013.

[11] A. C. Jose and R. Malekian, "Smart home automation security: A literature review," *Smart Comput. Rev.*, vol. 5, no. 4, pp. 269–285, Aug. 2015.

[12] The New York Times-John Schwartz. (Sep. 4, 2001). *Giving the Web a Memory Cost Its Users Privacy*. [Online]. Available: http://www.nytimes.com/2001/09/04/technology/04COOK.html

[13] B. Thomas, "Burnt offerings [Internet]," *IEEE Internet Comput.*, vol. 2, no. 6, pp. 84–86, Nov. 1998.

[14] L. F. Cranor, "Can users control online behavioral advertising effectively?" *IEEE Security Privacy*, vol. 10, no. 2, pp. 93–96, Mar./Apr. 2012.

[15] B. Krishnamurthy, "Privacy leakage on the Internet," in *Proc. 77th Internet Eng. Task Force*, Mar. 2010, pp. 1–53.

[16] B. Krishnamurthy and C. E. Wills, "Generating a privacy footprint on the Internet," in *Proc. 6th ACM SIGCOMM Conf. Internet Meas. (IMC)*, New York, NY, USA, 2006, pp. 65–70.

[17] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the Web," in *Proc. 9th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, Berkeley, CA, USA, 2012, p. 12.

[18] J. R. Mayer and J. C. Mitchell, "Third-party Web tracking: Policy and technology," in *Proc. IEEE Symp. Secur. Privacy*, San Francisco, CA, USA, May 2012, pp. 413–427.

[19] F. Z. Borgesius, "Behavioral targeting: A European legal perspective," *IEEE Security Privacy*, vol. 11, no. 1, pp. 82–85, Jan. 2013.

[20] J. Turow, J. King, C. J. Hoofnagle, A. Bleakley, and M. Hennessy, "Americans reject tailored advertising and three activities that enable it," Tech. Rep., Sep. 2009.

[21] B. Ur, P. G. Leon, L. F. Cranor, R. Shay, and Y. Wang, "Smart, useful, scary, creepy: Perceptions of online behavioral advertising," in *Proc. 8th Symp. Usable Privacy Secur. (SOUPS)*, New York, NY, USA, 2012, pp. 4:1–4:15.

[22] S. Saha, "Consideration points: Detecting cross-site scripting," *Int. J. Comput. Sci. Inf. Secur.*, vol. 4, nos. 1–2, pp. 1–8, 2009.

[23] M. D. Ayenson, D. Wambach, A. Soltani, N. Good, and C. J. Hoofnagle, "Flash cookies and privacy II: Now with HTML5 and ETag respawning," SSRN Electronic Journal, Team Res. Ubiquitous Secure Technol., Tech. Rep. 1898390, 2011.

[24] S. Kamkar. (Sep. 2010). *Evercookie—Never Forget*. [Online]. Available: http://samy.pl/evercookie/

[25] *The Impact of Cookie Deletion on Site-Server and Ad-Server Metrics in Australia*, comScore, Reston, VA, USA, Jan. 2011.

[26] J. R. Mayer, "Any person... a pamphleteer: Internet anonymity in the age of Web 2.0," M.S. thesis, Dept. School Public Int. Affairs, Princeton Univ., Princeton, NJ, USA, 2009.

[27] P. Eckersley, "How Unique Is Your Browser?" in *Proc. 10th Privacy Enhancing Technol. Symp. (PETS)*, Berlin, Germany, Jul. 2010, pp. 1–35.

[28] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host fingerprinting and tracking on the Web: Privacy and security implications," in *Proc. 19th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2012.

[29] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of Web-based device fingerprinting," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 541–555.

[30] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in JavaScript implementations," in *Proc. W2SP*, May 2011, pp. 1–11.

[31] J.-L. Gassée and F. Filloux. (May 2009). *Measuring Time Spent on a Web Page*. [Online]. Available: http://www.cbsnews.com/news/measuring-time-spent-on-a-web-page/

[32] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proc. W2SP*, May 2012, pp. 1–12.

[33] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 2, pp. 93–108, Apr./Jun. 2005.

[34] S. Zander and S. J. Murdoch, "An improved clock-skew measurement technique for revealing hidden services," in *Proc. 17th Conf. Secur. Symp.*, Berkeley, CA, USA, 2008, pp. 211–225.

[35] D.-J. Huang, K.-T. Yang, C.-C. Ni, W.-C. Teng, T.-R. Hsiang, and Y.-J. Lee, "Clock skew based client device identification in cloud environments," in *Proc. IEEE AINA*, Mar. 2012, pp. 526–533.

[36] G. Nakibly, G. Shelef, and S. Yudilevich, "Hardware fingerprinting using HTML5," *CoRR*, vol. 3, pp. 1–5 Mar. 2015.

[37] M. Smart, G. R. Malan, and F. Jahanian, "Defeating TCP/IP stack fingerprinting," in *Proc. 9th Usenix Secur. Symp. (USENIX)*, Denver, CO, USA, 2000, p. 17.

[38] D. W. Richardson, S. D. Gribble, and T. Kohno, "The limits of automatic OS fingerprint generation," in *Proc. 3rd ACM Workshop Artif. Intell. Secur. (AISec)*, Chicago, IL, USA, Oct. 2010, pp. 24–34.

[39] T. Oluwafemi, S. Gupta, S. Patel, and T. Kohno, "Experimental security analyses of non-networked compact fluorescent lamps: A case study of home automation security," in *Proc. Workshop Learn. Authoritative Secur. Experim. Results (LASER)*, Arlington, VA, USA, Oct. 2013, pp. 13–24.

[40] M. R. Faghani and U. T. Nguyen, "A study of XSS worm propagation and detection mechanisms in online social networks," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1815–1826, Nov. 2013.

[41] A. De Santis, A. G. Gaggia, and U. Vaccaro, "Bounds on entropy in a guessing game," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 468–473, Jan. 2001.

[42] D. C. Feldmeier and P. R. Karn, "UNIX password security—Ten years later," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 435. New York, NY, USA: Springer, 1990, pp. 44–63.

[43] E. I. Tatlı, "Cracking more password hashes with patterns," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1656–1665, Apr. 2015.

[44] K. Kato and V. Klyuev, "Strong passwords: Practical issues," in *Proc. IEEE 7th Int. Conf. Intell. Data Acquisition Adv. Comput. Syst. (IDAACS)*, Berlin, Germany, Sep. 2013, pp. 608–613.

[45] D. Hart, "Attitudes and practices of students towards password security," *J. Comput. Sci. Colleges*, vol. 23, no. 5, pp. 169–174, 2008.

[46] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: Empirical results," *IEEE Security Privacy*, vol. 2, no. 5, pp. 25–31, Sep./Oct. 2004.

[47] S. Farrell, "Password policy purgatory," *IEEE Internet Comput.*, vol. 12, no. 5, pp. 84–87, Sep./Oct. 2008.

[48] S. Rao, B. Jha, and G. Kini, "Effect of grammar on security of long passwords," in *Proc. 3rd ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, 2013, pp. 317–324.

[49] G. J. Johnson, "A distinctiveness model of serial learning," *Psychol. Rev.*, vol. 98, no. 2, pp. 204–217, 1991.

[50] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychol. Rev.*, vol. 63, no. 2, pp. 81–87, 1956.

[51] E. Rabinovitch, "Staying protected from 'social engineering,'" *IEEE Commun. Mag.*, vol. 45, no. 9, pp. 20–21, Sep. 2007.

[52] M. A. Sasse, S. Brostoff, and D. Weirich, "Transforming the 'weakest link'—A human/computer interaction approach to usable and effective security," *BT Technol. J.*, vol. 19, no. 3, pp. 122–131, Jul. 2001.

[53] H. Thompson, "The human element of information security," *IEEE Security Privacy*, vol. 11, no. 1, pp. 32–35, Jan./Feb. 2013.

[54] N. C. Zakas. (Oct. 2010). *How Many Users Have JavaScript Disabled?* [Online]. Available: https://developer.yahoo.com/blogs/author/nicholas-c–zakas/

[55] D. Jang, R. Jhala, S. Lerner, and H. Shacham, "An empirical study of privacy-violating information flows in JavaScript Web applications," in *Proc. CCS*, Oct. 2010, pp. 270–283.

[56] N. Ye, Z.-Q. Wang, R. Malekian, Q. Lin, and R.-C. Wang, "A method for driving route predictions based on hidden Markov model," *Math. Problems Eng.*, vol. 2015, Jan. 2015, Art. no. 824532.

[57] (Mar. 2016). *Browser Statistics*. [Online]. Available: http://www.w3schools.com/browsers/browsers_stats.asp

ARUN CYRIL JOSE received the undergraduate Degree in engineering in computer science from Mahatma Gandhi University, India, in 2009, and the master's degree in forensic computing from Coventry University, U.K., in 2011. He is currently pursuing the Ph.D. degree with the Department of Electrical Electronic and Computer Engineering, University of Pretoria, South Africa. His research interests include home automation security, distributed denial of service attacks, different online surveillance techniques, security-based social engineering, other implementations of P2P protocols, network vulnerability detection and intrusion prevention, network traffic analysis, and completely anonymous communication over the Internet.

REZA MALEKIAN (M'12) is currently a Senior Lecturer with the Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Pretoria, South Africa. His current research interests include advanced sensor networks, Internet of Things, and mobile communications. He is also a Chartered Engineer and a Professional Member of the British Computer Society. He is an Associate Editor of the *IEEE Internet of Things Journal* and the Area Editor of *Ad Hoc Networks* (Elsevier).

NING YE received Ph.D. degree from the Institute of Computer Science from Nanjing University of Post and Telecommunications, China, in 2009. In 2010, Dr. Ning worked as Visiting Scholar and Research Assistant in the Department of Computer Science, University of Victoria, Canada. She is currently a professor at Nanjing University of Post and Telecommunications, China. She is also a senior membership of CCF and a member of pervasive computing specialty committee in China. Her research focuses on the security and information processing, Advanced Sensor Technologies and intelligent transportation.