BrowserCloud: A Personal Cloud for Browser Session Migration and Management

Junjie Feng
Department of Computing and Information Systems
The University of Melbourne
Victoria, Australia
fed@student.unimelb.edu.au

Aaron Harwood
Department of Computing and Information Systems
The University of Melbourne
Victoria, Australia
aharwood@unimelb.edu.au

ABSTRACT
Web browsers are de facto clients for an ever-increasing range of web applications. At the same time, web users are accessing these applications from a wide range of devices. This paper presents a solution for runtime browser session migration and management, called BrowserCloud, which allows a user to securely manage multiple browsers from a personal or third-party Cloud service, migrate snapshots of active browser sessions between browsers over different devices, using a robust security module. The design of BrowserCloud is based on browser extensions/plugins that can preserve and restore browser session state, and a PHP server that stores browser sessions securely. We have tested our implementation over a range of increasingly complex web applications, including WebRTC and HTML5 video. To the best of our knowledge, our implementation is the most robust and secure approach to runtime browser session management to date.

Categories and Subject Descriptors
C.2.4 [**Computer Systems Organization**]: Distributed Systems; D.3.2 [**Software**]: Language Classifications - *JavaScript*; E.3 [**Data**]: Data Encryption;

General Terms
Application

Keywords
Browser Session Migration, Personal Cloud, WebSocket, Browser Extension, Security Module

1. INTRODUCTION
Web browsing is a ubiquitous daily activity for many people. For some time, "bookmarks" were enough to meet a user's expectation of browsing mobility. Nowadays, each user owns multiple browsers running with different browser software (Chrome, Firefox, Safari, and Internet Explorer), different hardware devices (Home PC, Workplace PC, mobile phone) and different operating systems (Windows, Linux, Mac OS, Android, iOS). Emerging web technology (HTML5, CCS3 and Web 3.0) and high speed Internet give web applications similar power and capabilities to conventional desktop applications. In another direction, the browser is being increasingly used as a thin client to a range of Cloud-based applications. Most web applications today, including Cloud-based applications, are session-oriented and as a result, standard bookmarks are no longer sufficient to satisfy the expectation of browsing mobility. To support a much richer user experience within this emerging browser-based paradigm, we propose the capability for a user to remotely manage multiple browser instances and seamlessly migrate an active browser session between browsers, using e.g. a personal or third-party Cloud service. In this paper we present our real-world implementation of such a service, which we call *BrowserCloud*.

Consider, for example, eBay users who are selling items, and that may check order and payment history regularly. Each time they revisit their order history, they may ordinarily click through a number of web pages to arrive at the order history page. Using a bookmark of the web URL in a second browser is insufficient to arrive at the order history page directly, because related cookies are needed to pass through the eBay login authentication. Rather, migration of the entire session state is needed. This is an example of more common problem that arises when people try to share a bookmark to web content that is session state dependent. Consider another case, where a user is enjoying a YouTube video, but has to shut down the PC shortly. She may like to continue the video on her smart phone, but would have to search for the video, and move the video time-line to where she left off. A service that could seamlessly migrate the active YouTube video from the PC to the mobile phone, continuing the video stream from where it left off, would be much more convenient.

1.1 Our Contributions
To address the limitations of bookmarks, and increase browsing mobility, we propose BrowserCloud, which was designed and developed to allow a user to centrally control multiple browsers, and seamlessly migrate active browser sessions in a secure way. We present a detailed description of our system and its operation. We tested our system on a range of websites, with PC and mobile browsers, and varying website complexity, which highlights a number of interesting and problematic cases. We also present measurements of performance of session migration in terms of latency and data transfer. Furthermore, we present our robust implementation of security that allows BrowserCloud to be run by a third-party service provider in such a way that a user's session data is kept confidential.

1.2 Paper Organization
Section 2 provides a high level description of our proposed system. Section 3 provides the details of its implementation. Section 4 shows our testing and performance measurements. Sections 5 and 6 discuss related work and make concluding remarks.

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2743043

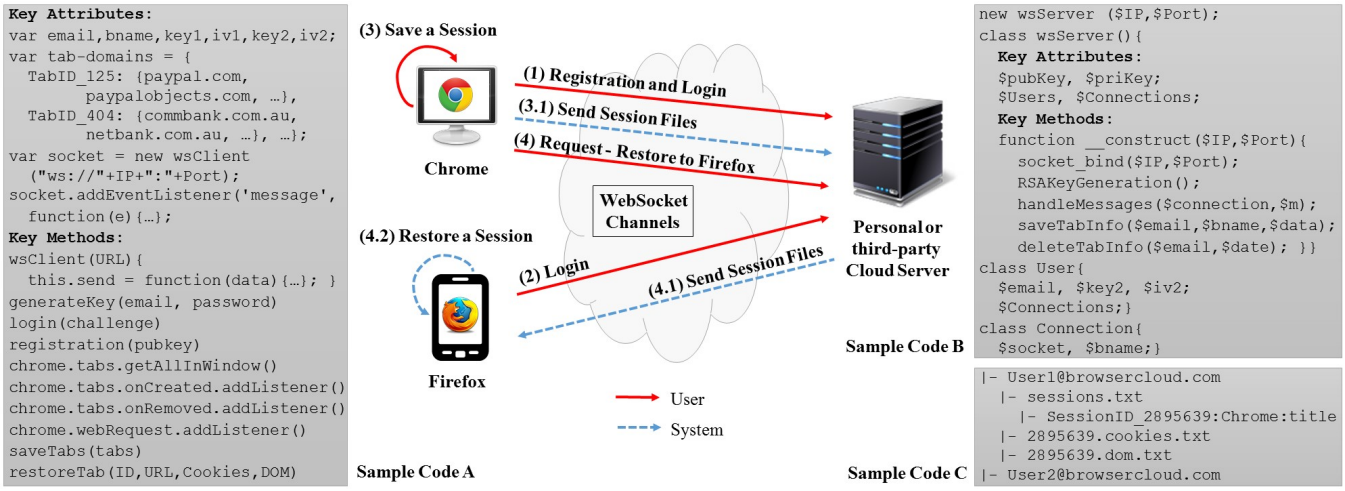

Figure 1. Overview of BrowserCloud

2. PROPOSED SYSTEM

This section begins by presenting the primary challenges and objectives of browser session migration and management, followed by the architecture and design of BrowserCloud.

2.1 Challenges and Objectives

Capturing a browser session is significantly more challenging than saving a web URL as a "bookmark", because of the following reasons:

Cookies - Browsers do not store cookies separately for each browser session and therefore simply migrating all cookies within a browser could include cookies for unrelated sessions and result in unnecessary overhead, security and privacy concerns. How to separate and migrate related cookies is the first challenge.

DOM - Browser session migration should also include the web page DOM elements and attributes, which could be modified by human input and web scripts, after a web page is loaded. Retrieving and migrating these updated DOM elements and attributes is the second challenge. Such DOM elements and attributes include window scroll bar position, web form input fields, audio/video current-time attributes, and others.

Security - Most browser sessions contain confidential information (e.g. ID card, credit card numbers), but the SSL protocol only secures communication channels between clients and servers. These servers or databases can potentially be hacked. Robust protection of this information is the third challenge.

Mobility - With the rapid growth of smart phone users, browsing mobility should be extended to mobile devices [1]. Migration of browser sessions between desktops and mobile devices, is the fourth challenge.

BrowserCloud successfully addresses all of the above challenges transparently to web developers and end-users. Beyond this, BrowserCloud also provides the following properties:

Remote Management - BrowserCloud allows a user to monitor multiple remote browsers' real-time activities, and manage any connected browsers' running sessions, from a central point.

Transparency - BrowserCloud does not require any change to existing websites or technical skills from end-users. It can automatically capture snapshots of running browser sessions, transfer session files to the server, and restore sessions to any target browser.

Scalability/Openness - BrowserCloud was developed based on open source and widely used programing languages – JavaScript and PHP, WebSocket protocol and cryptographic algorithms – RSA, SHA3, SHA236 and AES. By developing browser extensions/plugins, any browsers can join BrowserCloud.

Multi-user - A single BrowserCloud server can be shared by multiple users, each of which can connect and manage multiple browsers at the same time. Each user owns a private space in the BrowserCloud server to store browser sessions.

2.2 Architecture and Design

The architecture of BrowserCloud includes two main elements: BrowserCloud client and BrowserCloud server. The BrowserCloud client is a JavaScript-based browser extension/plugin, which consists of a background process that provides access to browser APIs, and a popup window that interacts with end-users. The BrowserCloud server is a PHP script, which is responsible for user registration, login authentication, and browser session storage.

Please see Figure 1 for the process of a runtime browser session migration. In our discussion we use an example case, that a user would like to migrate a running Commonwealth Bank Netbank login session as a demo.

Firstly, by installing the BrowserCloud client extensions/plugins on both the Desktop Chrome and Android Firefox browsers, a BrowserCloud icon/menu is added to the browsers' toolbar section. The user registers an account at the Chrome browser (1), and logs in at the Firefox browser (2). Secondly, the user logs in to the Commonwealth Bank Netbank at the Chrome browser, and submits a request to save the Netbank login session (3); the equivalent of bookmarking. The BrowserCloud client takes a snapshot of the Netbank login session and transfers it to the BrowserCloud server (3.1). Following, the user sends a request to restore the Netbank login session to the Firefox browser (4). The BrowserCloud server receives the request, locates the session files and sends them to the Firefox browser (4.1). Finally, the BrowserCloud client of the Firefox browser loads the Netbank login session into a new tab (4.2), and the Netbank login session continues exactly the same as where it was saved.

3. IMPLEMENTATION

The implementation of BrowserCloud is fundamentally based on a **communication** module, which provides persistent connections between BrowserCloud clients and the server. The **browser session migration** module allows each BrowserCloud client to capture and transfer browser sessions over the established connections. Additionally, a robust **security** module provides security assurance to the whole session migration process. See Figure 1 Sample Code A and B for the key attributes and methods that BrowserCloud implements.

3.1 Communication Establishment

BrowserCloud implements WebSocket as the communication protocol between the BrowserCloud server and BrowserCloud clients (see Figure 1), primarily because the WebSocket protocol provides a persistent connection between two parties, and either of them can initiate data transfer at any time. The BrowserCloud server runs as a WebSocket server, and listens on a specified port for WebSocket connection requests at all times. To establish a WebSocket connection, each BrowserCloud client starts by sending a regular HTTP request, which includes a WebSocket handshake request, to the BrowserCloud server. The BrowserCloud server accepts the WebSocket request by returning a WebSocket handshake response. Once the WebSocket handshake is completed, the initial HTTP connection is upgraded to a WebSocket connection, which utilizes the same underlying TCP/IP protocol. A WebSocket connection allows the client and server to send data back and forth in full-duplex mode. This feature allows BrowserCloud clients to exchange instant update messages containing the browsers' real-time activity information. It also allows BrowserCloud clients to send command messages for browser session save and restore processes at any remote browsers.

Additionally, WebSocket allows the BrowserCloud server and clients to transfer unlimited data over the connection channel. Data is minimally framed, with a small header (4-12 bytes), followed by a payload. A single message could optionally be split across several data frames. This feature allows the BrowserCloud server and clients to transfer browser session files, which generally consist of large Strings.

3.2 Browser Session Migration

BrowserCloud allows a user to save and restore browser sessions between any connected browsers. In the **browser session save process**, the BrowserCloud client completes the following tasks:

1. Maintain relationship between sessions/tabs and domains.
2. Identify the browser where sessions/tabs are running.
3. Save cookies for each session/tab.
4. Save modified and hidden DOM attributes.
5. Save the tab object.
6. Take a timestamp as the unique ID.
7. Transfer session files to the BrowserCloud server.
8. Close saved sessions/tabs.

In the **browser session restore process**, the BrowserCloud client completes the following tasks:

1. Receive session data from the BrowserCloud server.
2. Restore cookies.
3. Create a new tab with the URL.
4. Restore DOM elements and attributes.
5. Delete restored session files.

3.2.1 Maintaining the relationship between a browser session/tab and its domains

Within a browser, cookies are not stored separately for each browser session/tab. Simply migrating all cookies within a browser results in unnecessary data transfer and security issues. For example, with both PayPal and Hotmail login sessions running in a browser, migrating all cookies to the target browser for only Hotmail session migration will also provide the target browser with the logined PayPal session.

However, only migrating cookies that share the same domain as the URL of the session/tab has problems as well. For example, in the event of Commonwealth Netbank login authentication, it utilizes a server with a domain, which is different to the primary `commbank.com.au`, for login authentication purposes. In this case, only migrating cookies with domain `commbank.com.au` will omit the actual authentication cookies.

In order to collect all related cookies for a specified browser session/tab, ideally browsers should provide an API that returns a collection of cookies for each browser session/tab. However, such API does not exist for Chrome and Firefox at this time of writing. BrowserCloud overcomes this challenge by implementing three Chrome APIs: (1) `chrome.tab.getAllInWindow()` initializes a collection of domains for each browser session; (2) `chrome.webRequest.onResponseStarted.addListener()` updates a domain collection when it receives a web request; (3) `chrome.tabs.onRemoved.addListener()` removes a domain collection when a session/tab closes.

3.2.2 Save cookies for each session/tab

Cookies are client-side files that contain user information, and are significant components of a browser session. In the process of cookie preservation, the BrowserCloud client implements the following tasks: (1) **filtering** which returns only needed cookies, (2) **conversion** which makes cookies ready for transfer and storage, and (3) **encryption** which ensures information security.

3.2.3 Save modified and hidden DOM attributes

Except for cookies, there is also a need to capture the runtime DOM elements and attributes, since human inputs and JavaScript/VB scripts could add/edit/delete DOM elements and attributes, after a web page is loaded. However, saving the whole web page source code does not solve this problem, because some DOM elements and attributes are hidden from the source code. For example, HTML5 video/audio's currenttime attribute and web form post-filled values cannot be found within the source code. BrowserCloud overcomes this challenge by injecting and executing JavaScript in web pages.

In the process of DOM preservation, the BrowserCloud client implements three tasks: (1) **extraction** which retrieves the modified and hidden DOM elements, (2) **encoding** which transforms special characters into a valid ASCII format, and (3) **encryption** which ensures information security.

3.2.4 Save the Tab object

Each session/tab within a browser is a Tab object, which contains a browser-wide unique TabID, index, URL, title, and other data. A Tab object's TabID is unique within a browser, index represents its order in the browser, URL represents the link of the session/tab, and title represents the name of the web page. In the process of Tab preservation, BrowserCloud client accesses the Tab object through the Chrome API `chrome.tabs.get()` with a valid TabID as the parameter.

3.2.5 Send to the BrowserCloud server

On completion of the above steps, the background process of the BrowserCloud client has three encrypted String objects, which are Cookies String, DOM String and Tab String. A time stamp is taken at this time, and used as the unique ID for the session/tab. The ID and three String objects are joined together in the structure of "Session ID # Tab String # Cookies String # DOM String". This single, large String is sent to the BrowserCloud server through the established WebSocket channel. The BrowserCloud client closes the saved session/tab at the end of the session save process, in order to prevent conflicts when the session/tab is restored in another browser.

3.2.6 Restore sessions/tabs

When a user submits a browser session/tab restore request through the popup interface, the selected sessions/tabs' IDs and the target browser name are sent to the BrowserCloud server. The server locates the session/tab files, and sends them to the target browser through the established WebSocket channel. The background process of the target BrowserCloud client decrypts the session/tab files, and restores cookies, the tab object, DOM elements and attributes in order. On completion of browser session restoration, the restored session files are deleted from the server, in order to prevent conflicts when a user tries to restore the same browser session/tab in multiple browsers.

3.3 Security

The security and privacy of a user' browser sessions/tabs is essential, because most browser sessions/tabs contain a user' sensitive information, such as login sessions, personal information, credit card details, and much more. BrowserCloud implements a robust security module, which shares the same idea as Apple's recently announced iOS 8 security [2]. A user's secret key for browser session file encryption is never known by the server. That means even the server will not be able to read the user's session data.

BrowserCloud's security module is designed, based on cryptographic hash functions, secret-key encryption and public-key encryption, in order to prevent sniffing attacks, session hijacking attacks, replay attacks, server-side attacks, and similar kinds of attacks at the server.

3.3.1 Key generation

The BrowserCloud server generates an RSA (cryptosystem) key pair consisting of an RSA public key and an RSA private key, when initialized. The plain text encrypted by the RSA public key can only be decrypted by the corresponding RSA private key. BrowserCloud utilizes public-key encryption to transfer a user's secret keys over the WebSocket channels.

BrowserCloud client generates two secret keys and initialization vectors (IV) in the registration and login processes, based on the user's email address and password. The plain text encrypted by a secret key can only be decrypted by the same secret key. An IV is an arbitrary string/number that is utilized with a secret key for data encryption, and makes it more difficult for a hacker using a dictionary attack to find patterns and break a cipher. BrowserCloud implements secret-key encryption for data transfer and browser session storage.

Each BrowserCloud client has two secret keys and IVs. The first secret key consists of the password's SHA3 output as the secret key 1 and the email's SHA3 output as the IV 1. The second secret key consists of the password's SHA256 output as the secret key 2 and the first 16 characters of IV 1 as the IV 2. BrowserCloud implements both SHA3 and SHA256 hash functions in secret key generation to protect the user's password, since hash functions are one-way processes, having hash outputs would not be able to determine the original string (password). Additionally, because there is no relationship between SHA3 and SHA256 outputs, only having one hash output would not be able to retrieve the other one. [3]

3.3.2 Security in the registration process

In the registration process, the BrowserCloud client authenticates the BrowserCloud server, and transfers the secret key 2 and IV 2 to the server through public-key encryption. At the beginning, a BrowserCloud client requests the BrowserCloud server's RSA public key. Following, the user encrypts its secret key 2 and IV 2 with the RSA public key, and sends the cipher text to the server. Finally, the server decrypts the cipher text with its RSA private key, and retrieves the user's secret key 2 and IV 2. If there is no existing user, a new account is created for the user. The registration process prevents unauthorized servers, since the RSA public-key encryption makes sure that only the authorized server with the RSA private key will be able to decrypt and retrieve the user's secret key 2 and IV 2.

3.3.3 Security in the login process

In the login process, the BrowserCloud server and client authenticate each other through a challenge message. At first, the BrowserCloud client generates the same two secret keys and IVs, which are the same as what it gets in the registration process, because SHA3 and SHA256 hash functions are applied to the same user's inputs (email address and password). Secondly, the BrowserCloud client sends a login request with its email address to the BrowserCloud sever. Thirdly, the server retrieves the user's secret key 2 and IV 2 from its array of Users, and creates a random string. Following, the server generates a challenge message by encrypting the random string with user's key 2 and IV 2, and sends the challenge message to the BrowserCloud client. In the end, the client decrypts the challenge message, and responses to the server with the decrypted string. If the decrypted string matches the random string, the login authentication is passed. The utilization of challenge–response authentication prevents the login process from unauthorized servers, unauthorized clients, and replay attacks.

3.3.4 Security in communication and storage

After a BrowserCloud client register and login successfully, it utilizes secret key 2 and IV 2 to encrypt command messages between the BrowserCloud server and itself, but utilizes secret key 1 and IV 1 to encrypt its browser sessions/tabs files. Since the secret key 1 and IV 1 are never sent to the server, the server will not be able to decrypt and read the user's session data. In the case that the secret key 2 is stolen, the secret key 1 is still secure, and hackers are still unable to determine the secret key 1 to decrypt sessions/tabs files.

4. TESTING

The performance of BrowserCloud can be measured and evaluated from different perspectives – the kinds of browser sessions it can migrate, the size of browser session files, and the process time of migration. We have tested BrowserCloud on various popular websites and we report our results in the following two sections.

4.1 Feasibility Testing

BrowserCloud has been tested against the following test cases, to identify the kinds and complexities of browser sessions that can

and cannot be migrated between browsers with different public IP addresses. Possible failure reasons, barriers and solutions are discussed in detail.

Table 1. Test Cases of Browser Sessions Migration

No	S	Testing Web Pages	D
1	L	http://ptv.vic.gov.au/projects/	L
	L	http://ptv.vic.gov.au/projects/ http://ptv.vic.gov.au/about-ptv/	R_1
	L	http://ptv.vic.gov.au/projects/	R_2
	R_2	http://ptv.vic.gov.au/projects/	R_1
	R_1 R_2	http://ptv.vic.gov.au/projects/ http://ptv.vic.gov.au/about-ptv/	L
	L	http://www.boc.cn/sourcedb/whpj/	R_1
2	L	https://login.live.com/	R_1
		https://www.my.commbank.com.au/	
		https://www.paypal.com/	
		https://www.facebook.com/	
3		**Migrate playing YouTube video:** https://www.youtube.com/watch?v=UDXUDehUgIQ	
		Migrate playing HTML5 video: http://conference.aifs.gov.au/	
		Migrate filled web form: https://www.mymyki.com.au/NTSWebPortal/Common/Auxillary/Contactus.aspx?menu=Feedback	
		Migrate windows scroll bar position: http://www.www2015.it/	
4		https://docs.google.com/	
		https://apprtc.appspot.com/	
5		https://www.my.commbank.com.au/	
6		http://janeriddellarchitects.com.au/contact	
7		http://www.inmensia.com/files/minesweeper1.0.html	

S – Source Browser **D** – Destination Browser
L – Local Desktop Chrome with IP address 58.162.192.91
R_1 – Remote Desktop Chrome with IP address 122.151.142.44
R_2 – Android Firefox

BrowserCloud successfully migrates browser sessions in Test Cases 1 – 4. **Test Case 1** demonstrates that BrowserCloud can monitor remote browser activities, and control the migration of static web pages between different browsers from a central point, which could be the local browser's popup window. **Test Case 2** confirms that BrowserCloud can migrate login sessions between browsers with different IP addresses. **Test Case 3** shows that BrowserCloud can migrate hidden and modified DOM elements and attributes between different browsers. **Test Case 4** demonstrates that BrowserCloud can migrate some interesting and complex browser sessions, such as Google Docs and WebRTC.

However, BrowserCloud fails to migrate browser sessions in the following test cases.

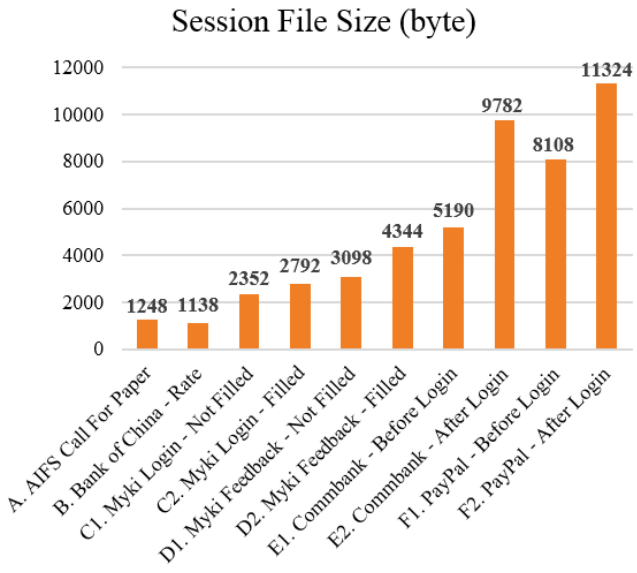

Figure 2. File Size of Browser Sessions

Test Case 5: Save a Commonwealth Bank Netbank login session; but it returns a session timeout error when the user restores it after 15 minutes. When the session has been inactive for more than 15 minutes, Netbank closes the session for security reasons. A possible solution could be that, the BrowserCloud server sends keep-alive requests to the web server on behalf of the user, while the user's session is saved. However, this solution involves security risks, since a hacked server may send unauthorized requests to the web server on behalf of the user.

Test Case 6: Save a contact us web page; but it fails to restore the same CAPTCHA image as the one in the original browser, since the web server creates a new CAPTCHA image for the new web request from the destination browser. To overcome this issue, we could implement a proxy server between the user's browsers and the web server. Since the proxy acts on behalf of the client in handling web requests, the web server only communicates with the proxy even when the client moves. Therefore, the web server uses the same CAPTCHA image. Nevertheless, a proxy server may significantly increase network delay.

Test Case 7: Save a website with a JavaScript game; but it fails to restore the process of the game, since BrowserCloud has not implemented the code to capture the status of JavaScript-based web applications. There are a few existing works regarding JavaScript status migration, such as `IMAGEN` [4] which provides a solution of migrating JavaScript-based online games. We are considering integrating BrowserCloud with `IMAGEN` or similar programs in the future.

4.2 Performance Testing

To measure the size of session files and the process time of session migration, we ran the browser session migration processes for several popular websites.

The size of a browser session is measured by the length of data that sends from the BrowserCloud client to the server. It includes the timestamp ID, encrypted cookies, encrypted DOM and encrypted tab object.

See Figure 2 for the size of browser sessions. Please note that both A and B have a smaller file size than others, mainly because they are static web pages, without many cookies and any filled web forms. The sizes of C1 and D1, whose web forms are unfilled, are

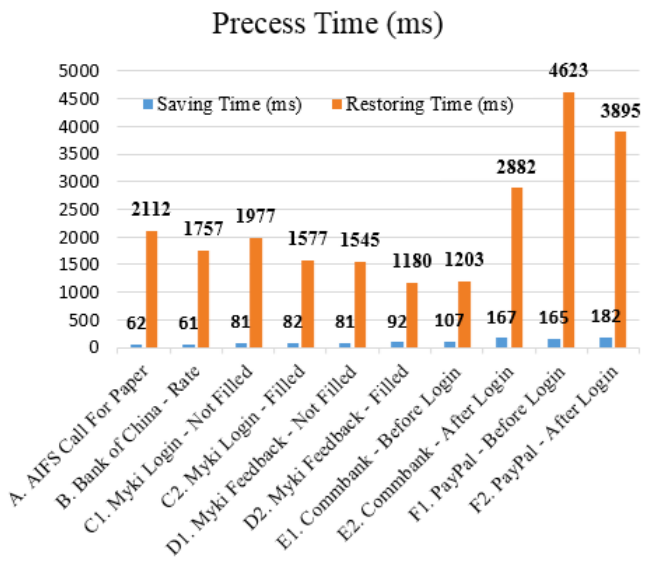

Figure 3. Process Time of Browser Sessions Migration

slightly smaller than C2 and D2 resp., whose web forms are filled. The difference between C1 and C2 is 440 bytes, but there are 1246 bytes difference between D1 and D2. It is because the myki Feedback form in D has more input fields than the myki login form in C. After-login sessions (E2 and F2) consume more space than before-login sessions (E1 and F1), since after-login sessions involve more cookies.

The saving process time includes the time spent on saving cookies, inserting JavaScript, saving DOM, saving Tab and data encryption. The restoring process time includes the time spent on data decryption, restoring cookies, loading web page, and running JavaScript to restore DOM elements. To improve accuracy, we take an average of five tests as the final result.

See Figure 3 for the process time of browser sessions save and restore. It distinctly shows that the restoring time is much longer than the saving time, because the process of restoring DOM elements happens after a website is fully loaded from its web server, which takes a long time. Since web server response times varies, and this directly effects the restore time, it will be meaningless to compare browser session restore times. By comparing the saving time, it shows that saving login sessions takes longer than saving filled web forms and static web pages, since it involves more cookies. In spite of this, it takes less than 0.2 second to save PayPal and Netbank login sessions, which should be acceptable for end-users. Combining both session file sizes in Figure 2 and session save times in Figure 3, it shows that it takes longer to save larger browser sessions files, partly because it takes longer to encrypt large Strings.

5. RELATED WORK AND DISCUSSION

There are existing researches on browsing mobility and browser session migration. Most of them focus on the integrity and seamlessness of browser session migration, but did not pay much attention to providing central management and addressing privacy concerns. Proxy-based session hand-off in Web applications [5] involves the utilization of a proxy system, called MuffinSH, running between clients and servers. Since the proxy acts on behalf of the client, servers see a surrogate of the client even when the client migrates from a device to another. However, the utilization of a proxy requires extra efforts in system setup and it increases network delay. Browser Mirror [6] enables a user to share the browser screen with anyone else. It differs to typical screen-sharing applications by sending DOM elements in real-time instead of images, but a mirrored user will not be able to interact with the web pages. `IMAGEN` [4] implements a platform, which is specialized for the live migration of JavaScript-based web apps, such as online games. In comparison, BrowserCloud is focused on generalized browse session migration.

6. SUMMARY AND FUTURE WORK

In summary, we describe BrowserCloud, which brings browsing mobility to users, using a communication module that enables a user to access remote browsers, a browser session migration module that captures browser sessions, and a security module that protects a user's information in many common attacks.

For future work, there are a number of aspects that we can address: (1) Websites generally set a timeout attribute for browser sessions. If a user does not restore and reactive the browser session within a certain time, the browser session will be closed automatically. A possible solution may be that the BrowserCloud server acts as a proxy server and regularly sends web requests to keep browser sessions active. (2) BrowserCloud may be able to integrate with `IMAGEN` to capture the status of JavaScript-based web applications. (3) CAPTCHA images are widely used in web forms submission to determine whether or not the user is human. How to migrate and continue to use the same CAPTCHA image at the destination browser is an interesting topic. (4) BrowserCloud clients specialized for Safari, including its mobile phone browser, could be developed. We have successfully developed BrowserCloud clients for desktop Chrome browser and Android Firefox browser, to demonstrate that browser sessions can be migrated between different browser software (Chrome and Firefox), different hardware devices (PC and mobile phone) and different operating systems (Windows and Linux).

7. REFERENCES

[1] Potter, S., & Nieh, J. 2005, May. WebPod: persistent Web browsing sessions with pocketable storage devices. In *Proceedings of the 14th international conference on World Wide Web* (pp. 603-612). ACM.

[2] A message from Tim Cook about Apple's commitment to your privacy. [ONLINE] Available at: https://www.apple.com/privacy/.

[3] Madhuravani, B., & Murthy, D. S. R. Cryptographic Hash Functions: SHA Family.

[4] Lo, J. T. K., Wohlstadter, E., & Mesbah, A. 2013, May. Imagen: Runtime migration of browser sessions for javascript web applications. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 815-826). International World Wide Web Conferences Steering Committee.

[5] Canfora, G., Di Santo, G., Venturi, G., Zimeo, E., & Zito, M. V. (2005, July). Proxy-based hand-off of Web sessions for user mobility. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on* (pp. 363-372). IEEE.

[6] Browser Mirror. 2014. [ONLINE] Available at: https://browsermirror.ianbicking.org/.