Usability Enhancement on the Privacy-Preserving Online Monitoring Framework for E-Health Applications

Youna Jung and Minsoo Kim
Department of Computer and Information Sciences
Virginia Military Institute
Lexington, Virginia, United States
e-mail: {jungy, kimm}@vmi.edu

Abstract— **Many e-health applications are currently using online monitoring services to improve the accuracy and quality of services. Privacy concern is however one of the biggest obstacles in widespread adoption of e-health applications. To address the privacy issue on e-health applications, we have preliminarily developed the privacy-preserving online monitoring framework (PPoM) that enables healthcare providers and patients to specify their own privacy policies without professional knowledge and skills and enforces patients' privacy policies during monitoring in systematic manner. The prototype successfully protects patients' privacy against unwanted data disclosure but its complex user interfaces reduce the performance of the PPoM. In this paper, we describe how we improve the PPoM to address the usability issue and present the enhanced version of the PPoM.**

Keywords - Privacy; policy-based protection; online monitoring; framework; e-health; usability.

I. Introduction

User monitoring on e-health application is a controversial issue. In order to assess and improve the performance of e-health applications, monitoring is one of the essential techniques. It tracks and analyzes patients' online activities (e.g., mouse clicks, frequency of use, time spent in a particular page, media viewed, page navigation sequences, content entered into a textbox, location, whether a mobile device is being used, etc.) on e-health applications.

In case of e-health applications that often deal with sensitive information, however, the protection of user privacy is critical. Indiscriminate monitoring without control over the sharing of patients' sensitive data may cause serious privacy problems (i.e., private health data may be used for unwanted purposes and/or shared with unknown people) [1][2][3][4]. It is therefore urgent and critical to facilitate online monitoring without privacy loss.

To this end, we have preliminarily proposed the PPoM framework [5] and the Privacy Policy Language that complies with the Health Insurance Portability and Accountability Act (HIPAA) [6] for e-health Applications. The PPoM framework enables healthcare providers to collect necessary information without violation of patient's privacy preferences and HIPAA regulations by enforcing patients' privacy preferences on the user side, not application side. To realize the proposed idea, we developed a prototype [1].

The prototype benefits both healthcare providers and patients. For healthcare provides, it offers an intuitive way to describe privacy policies for their e-health applications, monitor patients' activities, and collect patients' data without serious privacy breach. At the same time, for patients, it provides a way to verify an application's compliance with HIPAA and policies that are mutually agreed with patients, and if necessary, rigorously protect patients' private data based on their preference on the user side. However, in the previous development in [1], we focused on the feasibility and the performance of the prototype and did not deeply concern its usability. The usability is a critical issue because the PPoM aims at non-IT patients and healthcare providers. To address the limitation, in this paper, we present an enhanced prototype having improved user interfaces (UIs) and describes Human-Computer Interaction (HCI) techniques that are used in this enhancement.

The rest of this paper is organized as follows. In Section II, our preliminary work on the PPoM is introduced and the enhanced development is described in Section III with details and examples. In Section IV, we present evaluation results and in Section V, describe our conclusion and future work.

II. Preliminary Work

The PPoM framework has been proposed to address the privacy issues on e-heath applications conducting online monitoring [5]. In this section, we briefly introduce our preliminary work on the PPoM. As shown in Fig. 1, the PPoM framework consists of four components: the *HIPAA Profile*, the *PPoM Service*, the *PPoM Browser*, and the *PPoM Tools* (PPoMT).

- *HIPAA Profile* [6] – It is a policy profile that enables both patients and e-health providers to specify a privacy policy related to health data and HIPAA regulations. It has been proposed to address the lack of considerations on health-related data of existing general-purpose policy languages (e.g., P3P [7], APPEL [8], and XPref [9]). All patients by law have a right to know if an e-health application is compliant with HIPAA and Service-Level Agreements (SLAs). To this end, they need to publish their privacy preferences on health data first. However, it is a herculean task for non-IT people to specify and verify privacy policies on health-related data using existing languages because in order to so, they have to create their own data schema for health data. To address the limitations of existing languages, the PPoM employees the *HIPAA Profile* [6] that provides the *Health* data schema and extensions to P3P.

The *Health* data schema, an addition to the existing P3P data schemas, aims to describe a patient's health status. It contains sixteen health terminologies that can be widely acceptable in a variety of e-health applications: *height*, *weight*, *hearing-acuity*, *visual-acuity*, *blood-type*, *blood-pressure*, *allergies, blood-sugar-level*, *cholesterol-level*, *family-medical-history, disease-history, disabilities*, *immunization-history*, *healthcare-providers, medication*, and *lab-tests*. By using the *Health* data schema, patients and healthcare providers can specify their privacy preferences on health data and HIPAA regulations. In addition, the *HIPAA Profile* also prevents us from having inconsistent schemas across different patients and e-health applications.

In addition to the *Health* data schema, the profile provides several extensions to P3P, which allows specifying HIPAA-friendly privacy policy. It is critical for healthcare providers to ensure full compliance with HIPAA regulations because HIPAA is the most stringent rules for privacy protection against indiscriminate disclosure of health data. To this end, e-health providers need a privacy policy language that deals with terminology and rules in HIPAA. Towards this end, the *HIPAA Profile* provides extensions to the POLICY element of P3P (represented as <POLICY>) [6].

- *PPoM Browser* – It is a user browser having the PPoM plugin. It protects patients' privacy, even if a patient is exposed to untrustworthy e-health applications that conduct indiscriminate monitoring in violation of a patient's privacy policies. To do so, the *PPoM Browser* understands a patient's privacy preferences, presents all user data being monitored, and blocks outgoing messages that contain data he/she does not want to disclose.

- *PPoMT* – It is a toolkit that helps non-IT healthcare providers develop PPoM-enabled applications. Although patient monitoring is essential, it is quite difficult for healthcare professionals to develop e-health applications conducting online monitoring based on application policies. To overcome the difficulty, the *PPoMT* provides several tools that enable them to specify application policies and convert existing applications into PPoM-enabled applications without professional IT knowledge and skills.

- *PPoM Service* – It is an online monitoring service that gathers only authorized user/usage data that users allow to monitor. By specifying user policies, patients can determine which data can be monitored. Then, the *PPoM Service* selectively collects user/usage data based on user policies. Unlike the existing monitoring services where user data are collected based on applications' preferences, the *PPoM Service* provides a way to refer user policies during online monitoring in a systematic manner, rather than simply presenting a written agreement.

In the PPoM, a healthcare provider first needs to upload the source code or enter the URL(s) of his/her e-health application to the PPoMT and then select objects to be monitored and the corresponding privacy policies through the user-friendly interfaces generated by the In-page Selector. The Privacy Policy Generator then creates the application's policies by analyzing selected monitoring data and policies, while the Application Converter produces updated source code by inserting monitoring code generated by the Monitoring Code Generator into the original source code. The provider then needs to deploy the generated application policies and updated source code in the application's server.

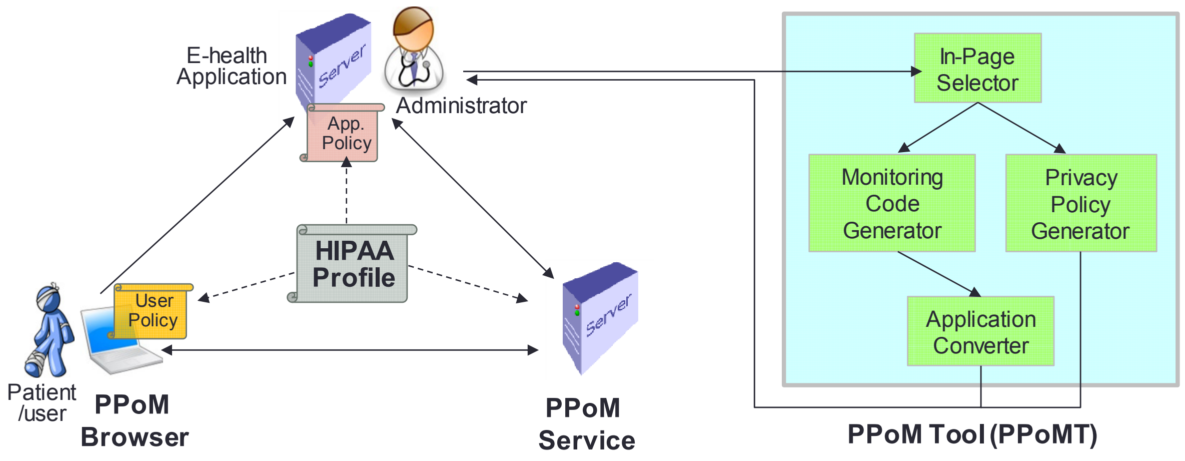


Figure 1. Overall Architecture of the PPoM Framework

When a patient enters a URL of e-health applications, his/her *PPoM Browser* compares user policies and application policies. If they match, the application server sends PPoM-enabled pages, which privacy-aware monitoring code is embed in. While the patient interacts with the application, the *PPoM Browser* displays all user/usage data being monitored so that the patient can verify privacy protection during online monitoring. The monitoring code inserted in webpages checks user policies prior to monitoring and collects only authorized user/usage. If any violation is suspected, the *PPoM Browser* will block outgoing messages to a monitoring server.

III. Enhancement on the PPoM Framework

To realize the proposed idea described above, we developed a prototype [1]. During our tests on the prototype, we found some usability issues. The usability is indeed a critical issue because the PPoM platform targets at non-IT users. In this section, we describe our improvement on each component of the PPoM in detail. Note that we developed the PPoM using PHP and JavaScript. We use PHP to develop the backend of the PPoM Browser and the *PPoMT* and JavaScript, HTML5, and CSS3 to develop the user interfaces (UIs). MySQL is used for a database of the *PPoM Service*.

A. HIPAA Profile

As mentioned above, the PPoM uses the *HIPAA Profile* to allow specifying privacy preference on data related to health and HIPAA regulations. To describe operations of each component in the PPoM, we use two examples of *HIPAA Profile* policy, a user policy shown in Fig. 2 and an application policy shown in Fig. 3. The examples are upgraded version of the previous examples in [6].

Fig. 2 shows an example of a patient's XPref policy specified using the *HIPAA Profile*. The user policy indicates that a patient allows a first party clinic to use his/her health data, except his/her *disability status* and *family history*, for a HIPAA-regulated retention period if the data collection is not for *telemarketing* purpose. This agreement is subject to an application's compliance with HIPAA regulations that stipulate healthcare providers must guarantee patients' access right.

```
<RULESET>
  <RULE behavior="block" condition="/POLICY[ACCESS/*
    [name(.) != "HIPAA-compliant-access"]
  <RULE behavior="block" condition="/POLICY/STATEMENT
    [PURPOSE/*[name(.) = "telemarketing"] or
    RECIPIENT/*[name(.) !="ours"] or
    RETENTION/*[name(.) !="HIPAA-compliant-retention"]]"/>
  <RULE behavior="block" condition="/POLICY/STATEMENT
    /DATA-GROUP/DATA [@ref="#health.disability" or
    @ref="#health.family-medical-history"/]
  <RULE behavior="request" condition= "true"/>
</RULESET>
```

Fig. 2. An example of a user policy specified using the HIPAA Profile

A sample application policy shown in Fig. 3 is generated for the *Daily Weight Tracker*, an e-health application that we developed as a testbed. The policy indicates that the tracker collects *weight*, *disability status*, *blood sugar level*, and *family medical history* of obese patients for the application's *healthcare operations* and *telemarketing* purposes. The collected health data will be disclosed to only the first party, the obese patient clinic that owns the online tracker. The clinic guarantees patients' right to access their health data and will retain monitoring data according to HIPAA regulations.

```
<POLICIES xmlns="http://www.w3.org/2002/01/P3Pv1">
 <POLICY discuri=http://dailyweighttracker.com/privacy.html
   name="policy">
  <ENTITY>
   <DATA-GROUP>
    <DATA ref="#business.contact-info.online.uri">
    http://dailyweighttracker.com/ </DATA>
    <DATA ref="#business.name"> Daily Weight Tracker
    </DATA>
   </DATA-GROUP>
  </ENTITY>
  <ACCESS> <HIPAA-compliant-access/> </ACCESS>
  <STATEMENT>
   <CONSEQUENCE>We collect health data of obese patients.
   </CONSEQUENCE>
   <PURPOSE>  <healthcare-operation/>  <telemarketing/>
   </PURPOSE>
   <RECIPIENT> <ours/> </RECIPIENT>
   <RETENTION> <HIPAA-compliant-retention/>
   </RETENTION>
   <DATA-GROUP>
    <DATA ref="#health.weight">
      <CATEGORIES> <health/> </CATEGORIES> </DATA>
    <DATA ref="#health.disabilities">
      <CATEGORIES> <health/> </CATEGORIES> </DATA>
    <DATA ref="#health.blood-sugar-level">
      <CATEGORIES> <health/> </CATEGORIES> </DATA>
    <DATA ref="#health.family-medical-history">
      <CATEGORIES> <health/> </CATEGORIES> </DATA>
   </DATA-GROUP>
  </STATEMENT>
 </POLICY>
</POLICIES>
```

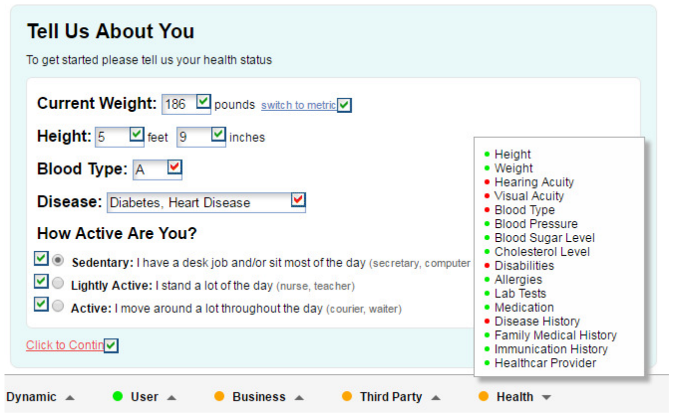
Fig. 3. An example application policy specified using the HIPAA Profile

B. PPoM Browser

To protect user privacy, the *PPoM Browser* provides three ways to enable non-IT patients to: 1) specify users' privacy preferences on health data without knowledge about policy language, 2) check all usage and user data being monitored, and 3) block monitoring if a patient finds unwanted data disclosure.

First, a patient can generate user policies through user interfaces. The PPoM supports three levels of user policy: the *General Policies* (GP), the *Application-specific Policies* (AP), and the *Page-specific Policies* (PP). A *GP* describes a patient's general preference regarding data sharing and it must be applicable to all online applications. To define a *GP*, a patient needs to specify data types that a patient allows or disallows to be monitored across different applications. An *AP* is generated for a particular application and it affects all webpages in an application. Unlike *GP* and *AP* that describe preferences on data types, in a *PP*, we can describe privacy policies for designated web objects and values. A *PP* is applied to only a particular webpage of an application. To specify an *AP* or a *PP*, a patient needs to enter an application's url in a *PPoM Browser* as shown in Fig. 5. If two or more policies conflict, then the most specific policy takes precedence. Note that the prototype of the current version of the *PPoM Browser* only supports the *AP*s and the *PP*s.

As we can see in Fig. 4, the interface of the previous prototype displayed many buttons and colorful checkboxes in one page in order to receive a user's selection of data and policy. However, such complicated interface increases task complexity, and in turn, reduced the usability of the *PPoM Browser*. To address the usability issue, we improve the browser's interface using dynamic menus. According to the study of Stefan Leuthold et. al., dynamic menus significantly reduce task complexity and increase retention rates [10]. By leveraging the results of the study, the enhanced *PPoM Browser* displays a main menu and limited number of sub-menus, as shown in Fig. 5.


Tell Us About You

To get started please tell us your health status

Current Weight: 186 pounds switch to metric

Height: 5 feet 9 inches

Blood Type: A

Disease: Diabetes, Heart Disease

How Active Are You?

Sedentary: I have a desk job and/or sit most of the day (secretary, computer

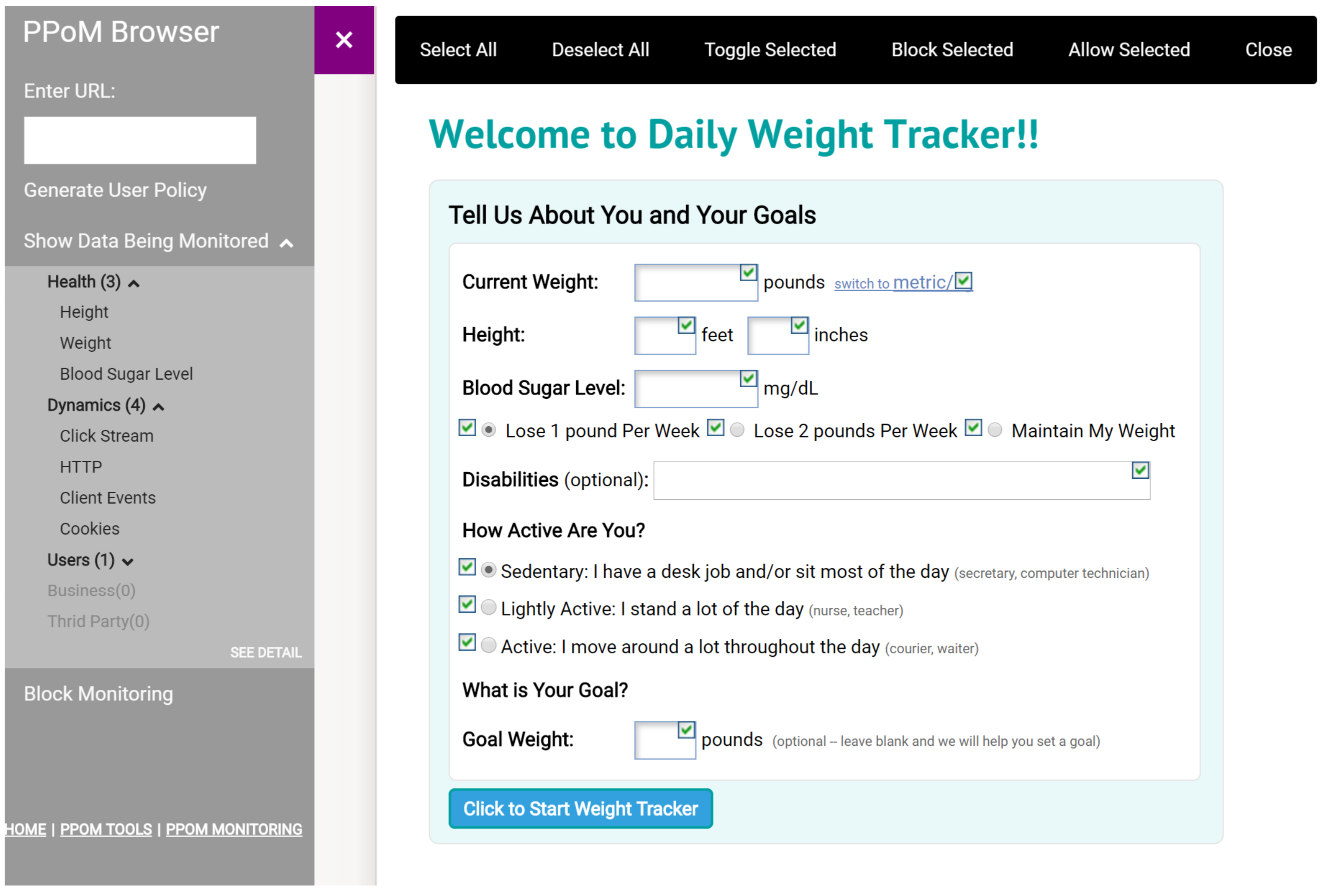
Lightly Active: I stand a lot of the day (nurse, teacher)

Active: I move around a lot throughout the day (courier, waiter)

Click to Contin

Figure 4. A complex UI of the previous PPoM Browser

The sub-menus displayed in a screen are dynamically determined depending on a user's current task and context. Fig. 5 shows a UI with five sub-menus (e.g., *Select All*, *Deselect All*, *Block Selected*, and *Allow Selected*), when a user clicks the *Block Selected* button on the main menu to select data to be protected. By using dynamic menus, the enhanced *PPoM browser* can provide more simple and intuitive interface.



Welcome to Daily Weight Tracker!!

Tell Us About You and Your Goals

Current Weight: pounds switch to metric/

Height: feet inches

Blood Sugar Level: mg/dL

Lose 1 pound Per Week Lose 2 pounds Per Week Maintain My Weight

Disabilities (optional):

How Active Are You?

Sedentary: I have a desk job and/or sit most of the day (secretary, computer technician)

Lightly Active: I stand a lot of the day (nurse, teacher)

Active: I move around a lot throughout the day (courier, waiter)

What is Your Goal?

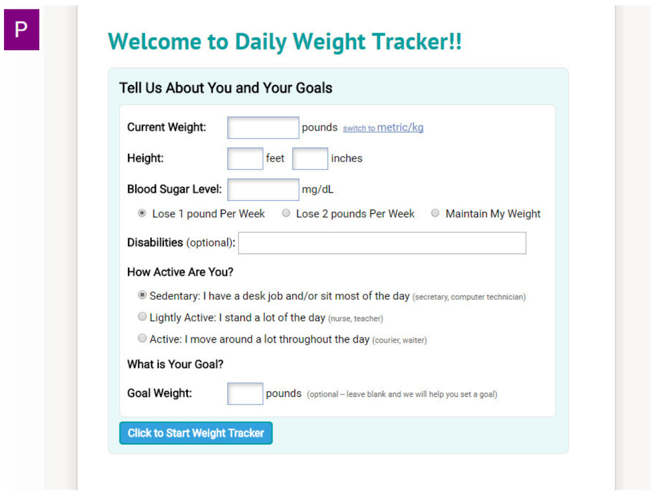
Goal Weight: pounds (optional – leave blank and we will help you set a goal)

Click to Start Weight Tracker

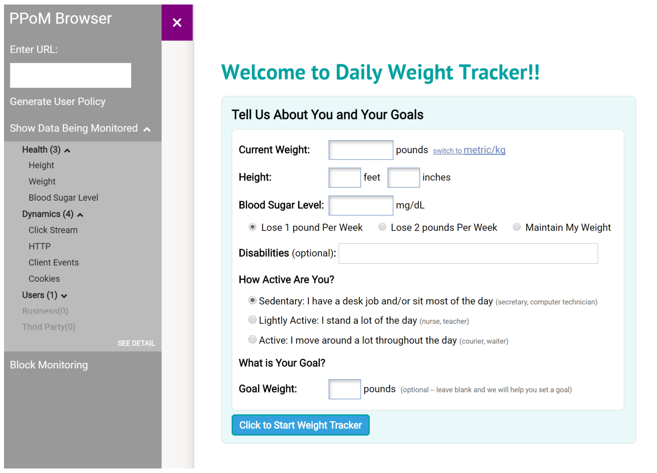
Figure 5. An improved user interface of the PPoM browser that uses dynamic menus

Second, the *PPoM Browser* displays all data being monitored when a patient uses an e-health application. Although a patient agrees to an application's policies, it is critical to verify the application's compliance with the mutually agreed SLA. To do so, a user needs to turn on the privacy-preserving mode by clicking the purple icon on the left top corner and select the *Show Data Being Monitored* menu. Then, a patient can easily figure out that what usage/user data are being monitored.

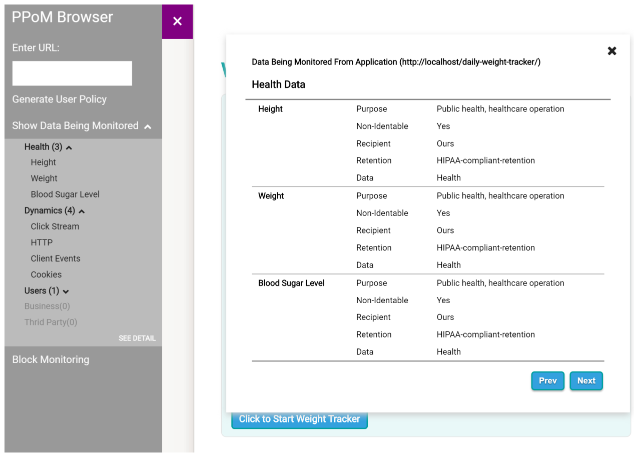
In the previous prototype, all monitoring data and recipients are displayed using different-colored checkmarks in one page as shown in Fig. 4. The *red* checkmarks mean that the data are protected and there is no recipient. The *orange* checkmarks mean that only the first party (for example, an e-health application that a patient is using) is receiving the data. The *green* checkmarks mean that third parties (for example, advertisement companies, payment companies, and other healthcare providers referred by the first party) are also receiving monitoring data. A summary of monitoring data, including not only general usage and user data defined in P3P and but also health-related data defined in the *HIPAA profile*, is displayed in the status bar.


P
Welcome to Daily Weight Tracker!!
Tell Us About You and Your Goals
Current Weight: pounds switch to metric/kg
Height: feet inches
Blood Sugar Level: mg/dL
Lose 1 pound Per Week
Lose 2 pounds Per Week
Maintain My Weight
Disabilities (optional):
How Active Are You?
Sedentary: I have a desk job and/or sit most of the day (secretary, computer technician)
Lightly Active: I stand a lot of the day (nurse, teacher)
Active: I move around a lot throughout the day (courier, waiter)
What is Your Goal?
Goal Weight: pounds (optional – leave blank and we will help you set a goal)
Click to Start Weight Tracker

a) PPoM Browser displaying no data


PPoM Browser
Enter URL:
Generate User Policy
Show Data Being Monitored
Health (3)
Height
Weight
Blood Sugar Level
Dynamics (4)
Click Stream
HTTP
Client Events
Cookies
Users (1)
Business(0)
Thrid Party(0)
SEE DETAIL
Block Monitoring
Welcome to Daily Weight Tracker!!
Tell Us About You and Your Goals
Current Weight: pounds switch to metric/kg
Height: feet inches
Blood Sugar Level: mg/dL
Lose 1 pound Per Week
Lose 2 pounds Per Week
Maintain My Weight
Disabilities (optional):
How Active Are You?
Sedentary: I have a desk job and/or sit most of the day (secretary, computer technician)
Lightly Active: I stand a lot of the day (nurse, teacher)
Active: I move around a lot throughout the day (courier, waiter)
What is Your Goal?
Goal Weight: pounds (optional – leave blank and we will help you set a goal)
Click to Start Weight Tracker

b) PPoM Browser displaying the numbers of monitoring data for each data schema (Health, Dynamics, User, Business, and Third Party)



Data Being Monitored From Application (http://localhost/daily-weight-tracker/)

Health Data

Height	Purpose	Public health, healthcare operation
	Non-Identable	Yes
	Recipient	Ours
	Retention	HIPAA-compliant-retention
	Data	Health
Weight	Purpose	Public health, healthcare operation
	Non-Identable	Yes
	Recipient	Ours
	Retention	HIPAA-compliant-retention
	Data	Health
Blood Sugar Level	Purpose	Public health, healthcare operation
	Non-Identable	Yes
	Recipient	Ours
	Retention	HIPAA-compliant-retention
	Data	Health

c) PPoM Browser disaplying all the data being monitored and its recipients

Figure 6. Three different levels of data display on the enhanced PPoM Browser

As pointed out above, however, displaying different types of information within a page may increase the task complexity and cause confusion to non-IT patients. Therefore, the enhanced browser only shows data that a user is interested in at that moment, rather than all the monitoring data. Fig. 6 (a) shows the enhanced browser's UI that does not display anything except the PPoM icon on the left top corner. If a patient does not want to see monitoring data, he/she can use the browser as an ordinary browser.

When a user needs to check data being monitored, clicks the icon and then the *Show Data being Monitored*. Then, the user can see the number of monitoring data for each data schema. If a user clicks on a particular schema, then specific data types are listed. Fig. 6 (b) indicates that three types of health data (*Height*, *Weight*, and *Blood Sugar Level*) are being monitored on that page. To check all monitoring data and recipients regardless of data schema at once, a user needs to click the *See Details*, and then a summary will be popped up, as shown in Fig. 6 (c).

Third, the *PPoM Browser* enables a patient to stop monitoring if he/she finds out fraud activities that are against the patient's preferences. To this end, a user needs to click the *Block Monitoring* in the main menu. Then, the browser renders all clickable web objects (e.g., buttons and objects handled by JavaScript click-event handler) and input HTML elements (e.g., textbox and checkbox), and creates checkboxes for each of them. By checking or unchecking the checkmarks, a patient can easily select data. At this time, a patient has multiple options for selecting objects: 1) select all clickable and input elements, 2) select all clickable elements, 3) select all input elements, 4) select an individual (clickable or input) element, or 5) select *None*. After selecting objects, the patient must decide whether or not to allow monitoring on those objects by clicking the *Allow Selected* button or the *Block Selected* button on the sub-menu.

To block monitoring, a *PPoM Browser* needs to generate and run JavaScript codes based on a patient's selection in real-time. Before explaining the blocking process further, let us assume that the main functionalities of an e-health application do not depend on JavaScript. To block online monitoring on a particular web element, a *PPoM Browser* disables JavaScript event handlers that are associated with the selected web elements, and in turn, a monitoring JavaScript using those handlers will be disabled. For example, if a patient selects the *Disabilities* textbox and clicks the *Block Selected* sub-menu. Then, the following JavaScript code is generated to disable the PPoM monitoring JavaScript on the *Disabilities* textbox (The ID of the textbox element is "DISABILITIESTXT"): `$("body").off("keyup keypress change click blur", "#DISABILITIESTXT").` The generated code then removes five event handlers from 'DISABILITIESTXT' element by invoking jQuery `off` function. When the blocking code runs, none of monitoring services obtains data from that textbox.

C. *PPoM Tools (PPoMT)*

The *PPoMT* is a server-side toolkit that enables non-IT healthcare providers to generate application policies and monitoring codes for their own e-health applications, and in turn, upgrade their existing applications into a PPoM-enabled application, without professional skills on programming and policy languages.

As a feasibility study, we developed a prototype of *PPoMT* [1] but there was a usability issue due to high task complexity. In order to simplify user tasks, we adopt the UI design rules proposed by Shneiderman [11]. According to the rules, we use dynamic menus and breaks complicated tasks (e.g., data selection and policy specification) into several simple steps using sliders as shown in Fig. 7. The enhanced *PPoMT* shows a main menu on the left side and only if necessary, it presents sub-menus on the top. The detailed explanation for each component is below.

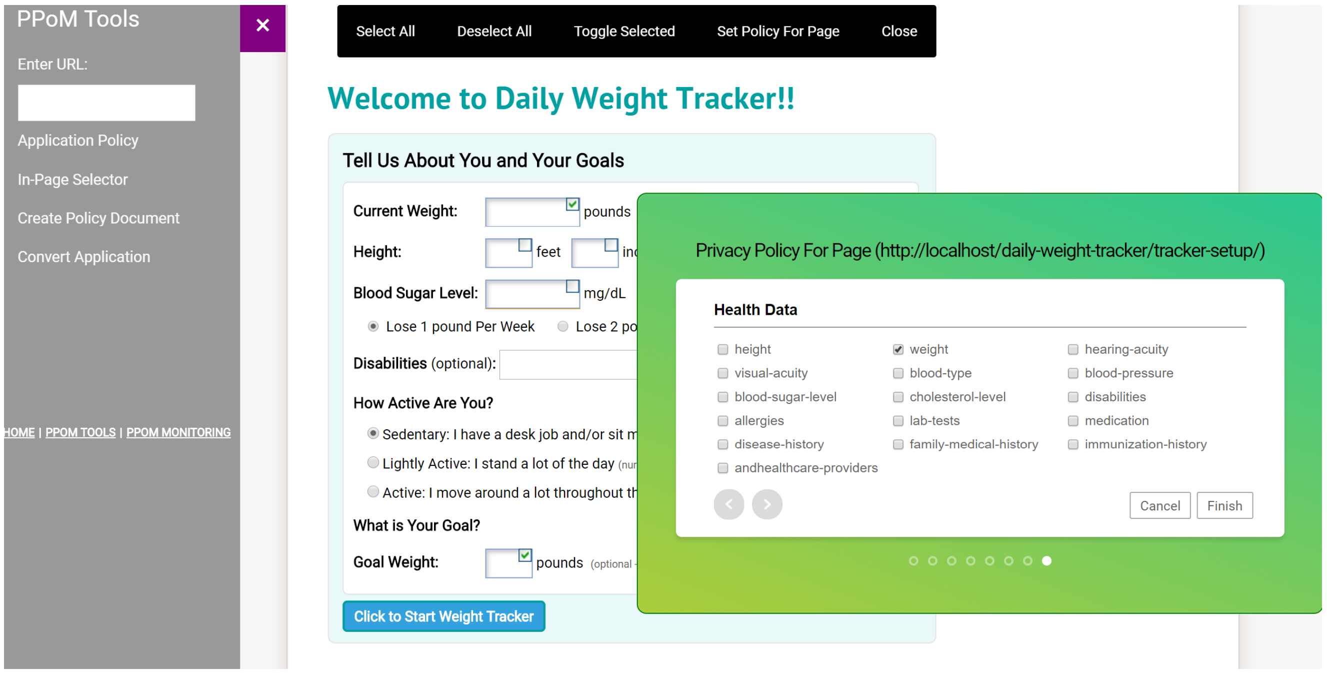
1) In-Page Selector

The *In-Page Selector* aims to show selectable HTML elements in webpages so that an administrator of an e-health application can select usage/user data to be monitored and policies corresponding to each web object, each page, or an entire application. Fig. 7 shows an example execution for the tracker setup page of the *Daily Weight Tracker* application. The selected data and policies are delivered to the *Privacy Policy Generator* and the *Monitoring Code Generator* for further processing.

2) Monitoring Code Generator

When receiving a set of data to be monitored and relevant policies, the *Monitoring Code Generator* produces privacy-aware monitoring codes for an e-health application. To this end, it first checks if each element selected has an ID. If not, it assigns a unique element ID and generates JavaScript code using the assigned ID. Depending on an application type, *static* or *dynamic*, ID generation processes are different.

A *static* application delivers the same HTML code stored in an application's server to all users' browsers, while a *dynamic* application dynamically generates HTML codes with different contents. If an e-health application is a *static* application, the *Monitoring Code Generator* assigns an absolute ID to an element. Let us assume that a webpage has several textboxes and its HTML code is shown in Fig. 8 (a).



Welcome to Daily Weight Tracker!!

Tell Us About You and Your Goals

Current Weight: [✓] pounds

Height: feet in

Blood Sugar Level: mg/dL

● Lose 1 pound Per Week ○ Lose 2 po

Disabilities (optional):

How Active Are You?

● Sedentary: I have a desk job and/or sit m

○ Lightly Active: I stand a lot of the day (nu

○ Active: I move around a lot throughout th

What is Your Goal?

Goal Weight: [✓] pounds (optional

Privacy Policy For Page (http://localhost/daily-weight-tracker/tracker-setup/)

Health Data

☐ height	☑ weight	☐ hearing-acuity
☐ visual-acuity	☐ blood-type	☐ blood-pressure
☐ blood-sugar-level	☐ cholesterol-level	☐ disabilities
☐ allergies	☐ lab-tests	☐ medication
☐ disease-history	☐ family-medical-history	☐ immunization-history
☐ andhealthcare-providers		

Figure 7. A screenshot of the enhnaced PPoMT (An administrator of the Daily Weight Tracker application is specifying a page policy for the Current Weight and the Goal Weight objects).

```
<body>
Current Weight:<input id="WEIGHT" type="text">lbs.
Height: <input type="text">feet
        <input type="text">inches
Blood Sugar Level: <input type="text"> mg/dL
......

<input id="BUTTON1" type="submit" value="Click to
Sart Weigt Tracker">
</body>
```

a) A code snippet of the tracker setup page shown in Fig. 6 (a)

```
<body>
Current Weight:
 <input id="WEIGHT" type="text"> lbs.
Height:
 <input id="PPOM-ELEMENT-0001" type="text"> feet
 <input id="PPOM-ELEMENT-0002" type="text">inches
Blood Sugar Level:
 <input id="PPOM-ELEMENT-0003" type="text"> mg/dL
......
<input id="button1" type="submit" value="Click to
Sart Weigt Tracker">
</body>
<script>
   $("#WEIGHT").change(function() {
      monitor($(this), "change");
   });
   $("#PPOM-ELEMENT-0001").change(function() {
      monitor($(this), "change");
   });
   $("#PPOM-ELEMENT-0002").change(function() {
      monitor($(this), "change");
   });
   $("#PPOM-ELEMENT-0003").change(function() {
      monitor($(this), "change");
   });
    ......
   $("#BUTTON1").click(function() {
      monitor($(this), "click");
   });
</script>
```

b) HTML code converted by the PPoMT in case of a static application

```
<script>
$("#WEIGHT").change(function()
   { monitor($(this), "change");});
$("input[type='text']:nth-of-type(2)").change(function()
   { monitor($(this), "change");});
$("input[type='text']:nth-of-type(3)").change(function()
    { monitor($(this), "change");});
$("input[type='text']:nth-of-type(4)").change(function()
   { monitor($(this), "change"); });
......
$("#BUTTON1").click(function()
  { monitor($(this), "click"); });
</script>
```

c) HTML code converted by the PPoMT in case of a dynamic application

Figure 8. An example of application conversion by inserting monitoring code generated by the Privacy Policy Generator into an exisitng application code.

In this example, the *Current Weight* textbox has its ID (`"WEIGHT"`) but other three textboxes (e.g., the *feet*, the *inches*, and the *Blood Sugar Level* textboxes) do not have their IDs. If the webpage is a *static* page, then the *Monitoring Code Generator* automatically creates IDs for three textboxes. For example, `"PPOM-ELEMENT-0001"`, `"PPOM-ELEMENT-0002"`, and `"PPOM-ELEMENT-0003"` for the *feet*, the *inches*, and the *Blood Type* textbox, respectively. The generated monitoring code using the absolute IDs is shown in Fig. 8 (b).

On the other hand, if an application is a *dynamic* application, a path of an element from a root of a Document Object Model (DOM) object is used as a unique ID because an element's path is unique and unchangeable. As you can see in Fig. 8 (c), except the pre-defined ID of the *Current Weight* textbox, for other three textboxes that do not have IDs, their paths are used to identify each textbox. For example, `"input[type=text]:nth-of-type(2)"` and `"input[type=text]:nth-of-type(3)"` for the *feet* and *inches* of the *Height* textbox and `"input[type='text']:nth-of-type(4)"` for the *Blood Type* textbox. Note that the PPoMT uses the PPoM Service so privacy-aware monitoring script code is generated as default, but it is possible to use different monitoring services such as *Google Analytics*.

3) Privacy Policy Generator

The *Privacy Policy Generator* generates an application's policies using the *HIPAA Profile*. To do so, an administrator needs to enter a url of an application and click *Create Policy Document* on the main menu. Alternatively, he/she needs to click *Set Policy For Page* after *In-Page Selector*.

When an administrator selects a web element or a group of elements, a slider is overlapped as shown in Fig. 7 to allow him/her to specify a privacy policy about the selected element(s). Each slide in the slider focuses on one element of a policy (e.g., *Purpose*, *Non-Identifiable*, *Recipient*, *Retention*, *Data Category*, or *Data Type*). This approach significantly reduces task complexity compare to the previous *PPoMT* that displayed all child elements in one page. Such advance in UIs allows more simple and intuitive use of the PPoMT.

4) Application Converter

This component produces a PPoM-enabled application by inserting monitoring codes generated by the *Monitoring Code Generator* into DOM objects for each webpage in an application. A conversion process may be different depending on an application's type. In case of a *static* application, the *Application Converter* can generate the updated HTML code systematically. However, the *PPoMT* provides monitoring script code only if an application is a *dynamic* application. In that case, an administrator must insert the generated monitoring code into the server-side program manually.

D. PPoM Service

The *PPoM Service* provides APIs for privacy-aware monitoring to applications' administrators so that they can embed the APIs in the webpages of their applications. Note that the APIs enable them to specify the type of data to be monitored, including health-related data types defined in the *HIPAA Profile*. Once deployed in an e-health application, the APIs check a patient's user policies, not an application's policies, and collect data that the patient allows to disclose. It does not collect data if a patient prefers not to disclose it, even if a monitoring code is inserted in webpages. By doing so, the *PPoM Service* provides a way to protect a patient's privacy from indiscriminate monitoring.

Monitoring data contain general usage data (e.g., *device category*, *operating systems*, *event*, *time*, etc.) and user data including health data. All data collected are encoded in JavaScript Object Notation (JSON), a lightweight data interchange format, and sent by a patient's web browser to the *PPoM Service* server. The structure of a JSON monitoring data is shown in Fig. 9.

[ELEMENT_ID|ELEMENT_PATH] [EVENT_TYPE] [TIME] [DATA_ TYPE] [DATA] [DEVICE_INFORMATION]

- *ELEMENT ID*: It is a unique ID of a HTML element.
- *ELEMENT_PATH*: In case of dynamic webpages, a path from the root element is used as an ID if an element does not have ID. The path is unique for each element.
- *EVENT_TYPE*: It denotes that a type of an event occurred. The set of event types are as follows: {*entering a page*, *leaving a page*, *clicking an element*, *filling an element*}.
- *TIME*: It denotes the occurring time of an event
- *DATA_TYPE*: It is a type of monitoring data and it must be specified based on the data types in the P3P data schema and the HIPAA Profile.
- *DATA*: It is an actual value of monitoring data.
- *DEVICE_INFORMATION*: It includes a device's *category*, *operating system*, *language*, and *browser information*.

Figure 9. The structure of a JSON object for monitoring data

```
<input id="bloodtype" type="text"
       data-type ="health.bloodtype"/>
```

a) HTML code for the Blood Type textbox

```
{"ELEMENT_ID": "bloodtype",
 "EVENT_TYPE": "TEXTINPUT",
 "TIME": "2016-07-15T12:45:07"
 "DATA_TYPE": "health.bloodtype",
 "DATA": "Type A",
 "DEVEICE_INFORMATION":
  { "DEVICE_CATEGORY":"DESKTOP","OS": "WINDOWS",
    "LANGUAGE": "ENGLISH","BROWSER": "FIREFOX"} }
```

b) JSON Object of the Raw Monitoring Data

Figure 10. Examples of monitoring data.

Let us assume that a patient enters "*A*" in the *Blood Type* textbox, which its HTML code is shown in Fig. 10 (a). Then, the monitoring data captured on that textbox is encoded a JSON object as shown in Fig. 10 (b). At this time, the values of *EVENT*, *TIME*, and *DEVICE_INFORMATION* are automatically collected by JavaScript's Built-in functions. Note that *blood type* is one of the health data type defined in the *Health* data schema.

To obtain monitoring data, the `monitor` JavaScript function presented in Fig. 11 (a) must be embedded in webpages of an e-health application prior to monitoring. When a target event is occurred, the `monitor` function captures monitoring data and creates a JSON object. To do so, the function gathers necessary information by using JavaScript built-in functions and properties. Then, it invokes the `jQuery.ajax` function to communicate with the server-side scripts (the `receiveData` function shown in Fig. 11 (b)). The `jQuery.ajax` function converts a JSON object into a string and sends it to the *PPoM Service* server through the HTTP POST method. When receiving a JSON string, the `receiveData` PHP module in the *PPoM Service* server converts the string into a JSON object and stores monitoring data in its database as shown in Fig. 12.

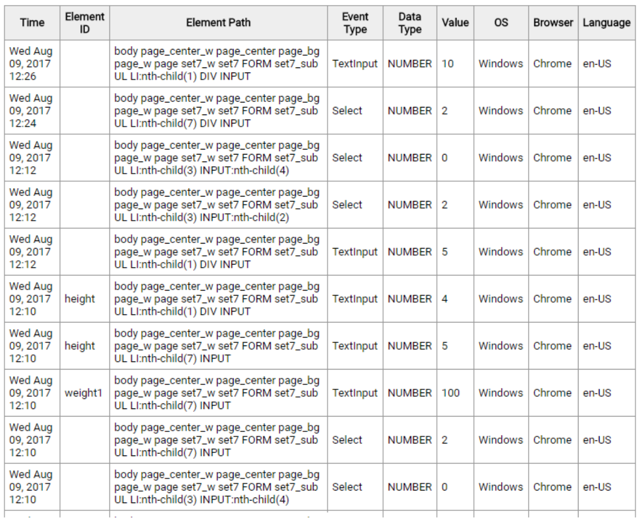
```
function monitor(object, event) {
   var monitoredData <= ID, EVENT_TYPE, TIME,
   DATA_TYPE, DATA, DEVICE_INFORMATION from the
   parameters object and event;
   jQuery.ajax({
      type: "post",
      url: "/PPoM/monitoring.php",
      data: JSON.stringify(monitoredData),
      contentType:"application/json; ",
      dataType: "json"                  });       }
```

a) JavaScript Function on the application side

```
function receiveData($monitoredData) {
   $object = json_decode($monitoredData);
   $link <= connection to database;
   $sql <= create INSERT query to store the
        monitored information ($object)
   mysqli_query($link, $sql);          }
```

b) PHP function in the PPoM service

Figure 11. Pseudo code of the PPoM Service that are used on both sides, the application side and the server side.

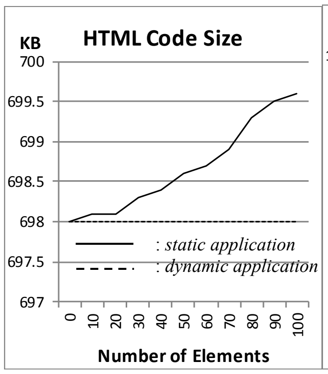


Time	Element ID	Element Path	Event Type	Data Type	Value	OS	Browser	Language
Wed Aug 09, 2017 12:26		body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(1) DIV INPUT	TextInput	NUMBER	10	Windows	Chrome	en-US
Wed Aug 09, 2017 12:24		body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(7) DIV INPUT	Select	NUMBER	2	Windows	Chrome	en-US
Wed Aug 09, 2017 12:12		body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(3) INPUT:nth-child(4)	Select	NUMBER	0	Windows	Chrome	en-US
Wed Aug 09, 2017 12:12		body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(3) INPUT:nth-child(2)	Select	NUMBER	2	Windows	Chrome	en-US
Wed Aug 09, 2017 12:12		body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(1) DIV INPUT	TextInput	NUMBER	5	Windows	Chrome	en-US
Wed Aug 09, 2017 12:10	height	body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(1) DIV INPUT	TextInput	NUMBER	4	Windows	Chrome	en-US
Wed Aug 09, 2017 12:10	height	body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(7) INPUT	TextInput	NUMBER	5	Windows	Chrome	en-US
Wed Aug 09, 2017 12:10	weight1	body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(7) INPUT	TextInput	NUMBER	100	Windows	Chrome	en-US
Wed Aug 09, 2017 12:10		body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(7) INPUT	Select	NUMBER	2	Windows	Chrome	en-US
Wed Aug 09, 2017 12:10		body page_center_w page_center page_bg page_w page set7_w set7 FORM set7_sub UL LI:nth-child(3) INPUT:nth-child(4)	Select	NUMBER	0	Windows	Chrome	en-US

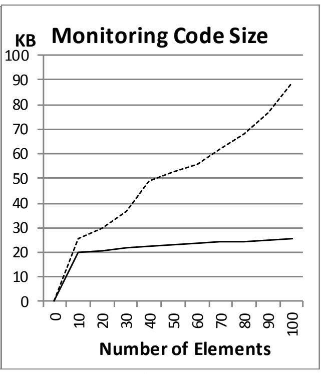
Figure 12. Example monitoring data stored in the database of the PPoM Service

IV. EVALUATION RESULTS

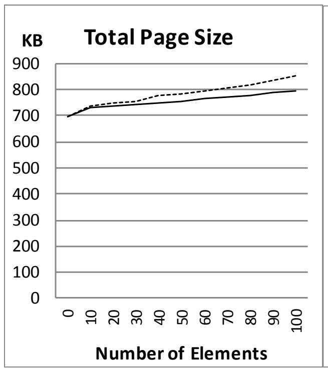
To test the performance of the prototype, we first develop two types of sample e-health applications that each has ten webpages containing different numbers of monitoring elements without IDs. As the first step, we evaluate the performance of the *PPoMT,* the *PPoM Browser*, and the *PPoM Service*, according to the evaluation plans described in [5].



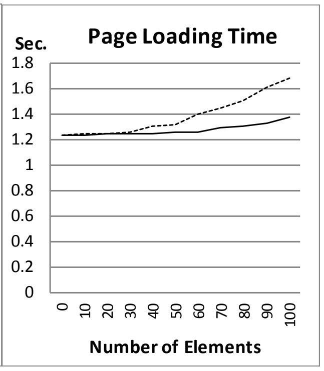
(a)



(b)



(c)



(d)

Figure 13. Evaluation Results.

As shown in Fig. 13, the size of HTML code that generated by the *Application Converter* for *dynamic* webpages is zero because the *PPoMT* will not generate HTML code for a *dynamic* application, while the size of code for *static* webpages increases linearly as the number of the monitoring elements increases (see Fig. 13 (a)).

In case of *static* webpages, the size of the generated monitoring code remains steady once it reaches a certain size, even though the number of monitoring elements increases linearly. However, the size of monitoring code for *dynamic* webpages linearly increases according to increase of the number of monitoring elements (see Fig. 13 (b)). This is because specifying paths from DOM root is costly, especially for complex web pages.

As you can see in Fig. 13 (c) and (d), in *static* webpages, the number of monitoring elements does not affect the size of converted webpages and the page loading time. However, in *dynamic* webpages, the increase in the number of monitoring elements affects the page loading time. If a *dynamic* page is complex, the loading delay becomes a big obstacle. It is one of our challenges to find out a way to minimize the loading delay caused on dynamic webpages.

To evaluate the privacy protection of the *PPoM Browser* and the *PPoM Service*, we generated five hundreds of different sets of patients' privacy preferences (i.e., allow or disallow monitoring on particular web elements) and user activities on a sample *static* application (i.e., navigating webpages, clicking buttons, or entering data in input elements). Using the sets of synthetic user policies and activities, we tested the privacy protection on the prototype. Towards this, we measured two factors: the *failure ratio* (*c/(a+c)* in Table I) and the *successful blockade ratio* (*h/(g+h)* in Table II).

TABLE I. FAILURE RATIO IN THE PPOM SERVICE

	Monitored	Not Monitored
Allowed	(a) 4,897	(b) 345
Not Allowed	(c) 0	(d) 3,477

TABLE II. SUCCESSFUL BLOCKADE RATIO IN THE PPOM BROWSER

	Sent	Blocked
Allowed	(e) 5,242	(f) 0
Not Allowed	(g) 0	(h) 3,477

The *failure ratio* evaluates privacy protection on the server side (the PPoM Service server) and the *successful blockade ratio* evaluates protection on the client side (the *PPoM Browser*). As shown in Table I and II, none of user/usage data that patients do not allow to be monitored was captured by the *PPoM Service* and none of the unauthorized data was sent from the *PPoM Browser*. However, as shown in Table II, we found some data loss in the *PPoM Service* server. The *PPoM Browser* sent 5,242 data (*a*+*b*), but the *PPoM Server* received only 4,897 data (*a*). It may be caused by heavy load of transactions or overheads for checking application policies and user policies. To figure out the source of the data loss, we will investigate the prototype's monitoring service in the future.

V. CONCLUSION

There is an urgent need for privacy protection on e-health applications. Although e-health applications can help people access healthcare service in an easy and convenient way at the reduced cost, many people hesitate to use e-health applications due to privacy concerns. To address the privacy issue, we have proposed the Privacy-Preserving online Monitoring framework, in short PPoM, and developed a prototype. In this paper, we address the usability issues on the previous prototype and describe improvements on our development with detailed examples. To achieve our ultimate goal, however, the following tasks must be completed in the future:

- A way to reduce data loss ratio on the *PPoM Service*.
- A method to reduce the size of the monitoring codes generated by the *PPoMT*, especially for dynamic applications.
- Usability test for the prototype
- Field test with actual patients and e-health applications.

REFERENCES

[1] M. Kim and Y. Jung, "A Development of Privacy-Preserving Monitoring System for e-Health Applications," Proc. the 5th international conference on global health challenges (Global Health 2016), IARIA, October, pp. 64-70, 2016.

[2] J. R. Mayer and J. C. Mitchell, "Third-Party Web Tracking: Policy and Technology," Proc. IEEE Symp. on Security and Privacy (SP '12), IEEE Press, pp. 413-427, 2012.

[3] M. Bilenko and M. Richardson, "Predictive client-side profiles for personalized advertising," Proc. ACM SIGKDD conf. on Knowledge discovery and data mining (KDD '11), ACM New York, pp. 413-421, 2011.

[4] A. McDonald and L. F. Cranor, "Beliefs and Behaviors: Internet Users' Understanding of Behavioral Advertising," TPRC 2010 Social Science Research Network (SSRN), August 16, pp. 1-31, 2010, Available from http://ssrn.com/abstract=1989092 [retrieved: 1 December, 2017].

[5] Y. Jung, "Toward Usable and Trustworthy Online Monitoring on e-health Applications," International Journal On Advances in Life Sciences, vol. 8, numbers 1 and 2, pp. 122-132, June, 2016

[6] Y. Jung and M. Kim, "HIPAA-Compliant Privacy Policy Language for e-health Applications," Proc. the 6th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare, Procedia Computer Science, Vol. 98, Elsevier, September 19-22, London, United Kingdom, pp. 283-289, 2016.

[7] P3P 1.1. http://www.w3.org/TR/P3P11/ [retrieved: 1 December, 2017].

[8] A P3P Preference Exchange Language (APPEL) version 1.0. https://www.w3.org/TR/P3P-preferences/ [retrieved: 1 December, 2017].

[9] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "An XPath-based preference language for P3P," Proc. the 12th international conference on World Wide Web (WWW '03), ACM, New York, pp. 629-639, 2003.

[10] S. Leuthold, P. Schmutz, J. A. Bargas-Avila, A. N. Tuch, and K. Opwis, "Vertical versus dynamic menus on the world wide web: Eye tracking study measuring the influence of menu design and task complexity on user performance and subjective preference," Comput. Hum. Behav. 27, 1, January, pp. 459-472, 2011.

[11] B. Shneiderman, "Designing the user interface: strategies for effective human-computer interaction," Pearson Education, India, 2010.