Reading sensor data for 4-digit PINs using JavaScript

Author: Maryam Mehrnezhad (m.mehrnezhad@ncl.ac.uk), Apr 2017

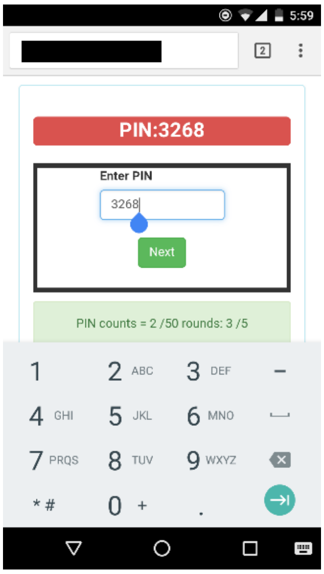
In this help file, we describe the details of our JavaScript code used for reading sensor data (motion and orientation) for 4-digit PINs in a project conducted in Newcastle University, UK. The outcome of this project is published in [1-4]. Our JavaScript code is publicly available on *Github* via this link: https://github.com/maryammjd/Reading-sensor-data-for-fifty-4digit-PINs. This code asks the user to enter fifty 4-digit PINs, each 5 times, and saves the PINs along with their sensor measurements (motion and orientation) in an *m-lab* database. A sample dataset for 10 users is also publicly available via the project's *Github* page. In case of any further questions, please contact the authors.

JavaScript code (client, server, and db)

We setup an account in *mlab.com* and created a deployment (database) named *sensordata*. In this deployment, we created a collection named *sensor*. This collection is in charge of saving json (JavaScript Object Notation) data received by the server as documents. We defined our json structure in our JavaScript code in *Node.js* to include three elements: *type* (status, or sensor type, or time), *data* (value), *ts* (time value). Note that the time in the *type* element is when the data is read on the mobile device, versus the *ts* element is when each record is inserted in the database.

In our JavaScript code (*app.js*), we connect to *mongoDB* and handle the sensor data via the *socket.io* API. All user interactions (beginning PIN entry, entering PINs, and finishing), alongside with the sensor measurements (motion and orientation), are sent to the database by the server. We run the server on a local computer through *node.js* cmd. Once the index page is opened on the phone, the data collection starts.

In our *index.html* file in the client side, we call the *numPad.js* script which presents the users with a GUI where fifty 4-digit PINs are shown (each repeated 5 times). The user needs to enter them in a textbox as shown in Fig. 1. As it can be seen, the number of PINs entered (out of 50) and the number of counts (out of 5) are also shown to users. On each digit entry, our JavaScript code sends a new record (Key Down Key Up) to our database using the *onkeydown* event. Our *numPad.js* file includes two event listeners on the window object which fire on device motion and device orientation DOM events (called *devicemotion* and *deviceorientation*). We have hard-coded the fifty 4-digit PINs in this file. These semi-random PINs are created by using a Matlab code.



[redacted]

PIN:3268

Enter PIN

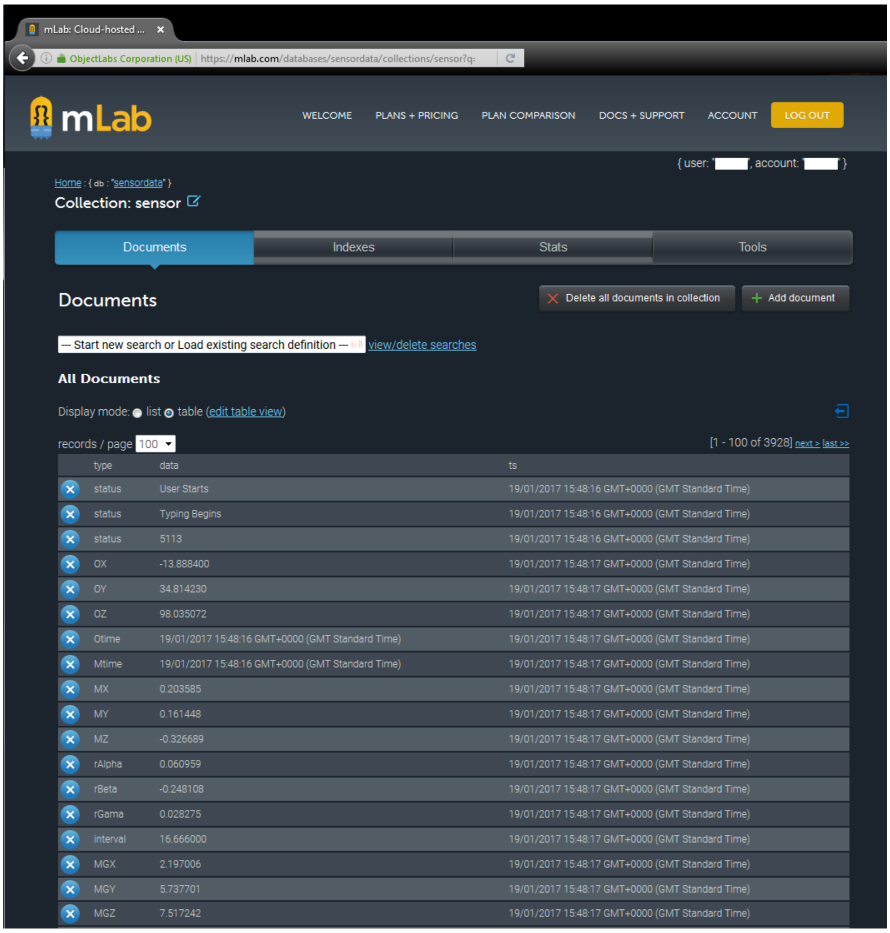
3268

Next

PIN counts = 2 /50 rounds: 3 /5

Figure 1: GUI for PIN entry

This data is arrived and inserted to our *MongoDB* database as shown in Fig 2.



https://mlab.com/databases/sensordata/collections/sensor?q:

{ user: '[redacted]', account: '[redacted]' }

Home : { db : "sensordata" }

Collection: sensor

Documents

— Start new search or Load existing search definition —

All Documents

Display mode: ○ list ● table (edit table view)

records / page 100

[1 - 100 of 3928]

type	data	ts
status	User Starts	19/01/2017 15:48:16 GMT+0000 (GMT Standard Time)
status	Typing Begins	19/01/2017 15:48:16 GMT+0000 (GMT Standard Time)
status	5113	19/01/2017 15:48:16 GMT+0000 (GMT Standard Time)
OX	-13.888400	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
OY	34.814230	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
OZ	98.035072	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
Otime	19/01/2017 15:48:16 GMT+0000 (GMT Standard Time)	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
Mtime	19/01/2017 15:48:16 GMT+0000 (GMT Standard Time)	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
MX	0.203585	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
MY	0.161448	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
MZ	-0.326689	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
rAlpha	0.060959	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
rBeta	-0.248108	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
rGama	0.028275	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
interval	16.666000	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
MGX	2.197006	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
MGY	5.737701	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)
MGZ	7.517242	19/01/2017 15:48:17 GMT+0000 (GMT Standard Time)

Figure 2: M-lab database

As it can be seen, the *type* element can include either the status of the data, the type of the data, or the time that it has been collected from the mobile device. The order of the values for a sample data collection for fifty 4-digit PINs (each PIN 5 times) from a user is saved as presented in the bellow set:

{User Starts,
{{Typing Begins,
5113 (First shown PIN),
a series of Orientation and Motion Data,
Key Down, Key Up (when the first digit is clicked),
a series of Orientation and Motion Data for the first digit,
Key Down, Key Up (when the second digit is clicked),
a series of Orientation and Motion Data for the second digit,
… (the same for the third and fourth digits),
Key Down, Key Up (to show the end of the 4-digit PIN entry),
5113 (First typed PIN which could be different from the shown PIN due to user error),
Typing Ends},
… (the previous process for the first PIN for another 4 times)},
… (the previous process for another 49 PINs),
User Finishes}.

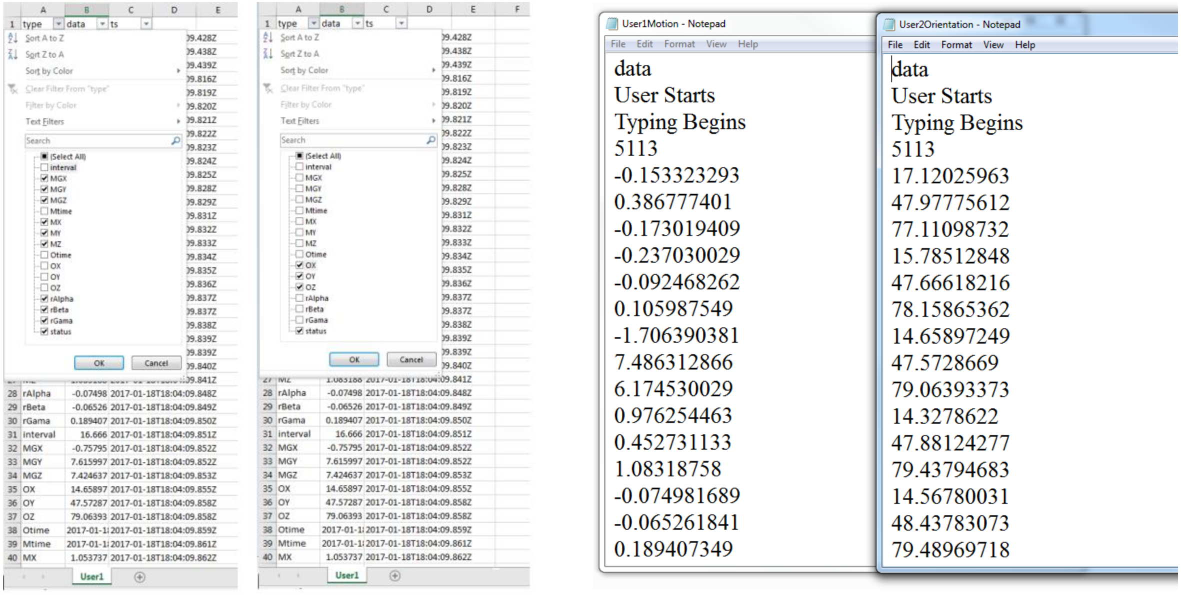
Data Exportation

After we collected data for each user, we exported the data to an *Excel* file on a local computer for further processing in *Matlab*. Next, we deleted all the documents in the related collection in *mlab.com* for the next user data collection. We used the following command through *MongoDb* cmd for exportation (the username and password are set on the time of creating the *sensordata* development):

```
mongoexport -h ds033818.mlab.com:33818 -d sensordata -c sensor -u <username> -p <password> -o sensor.csv --csv -f "type","data","ts"
```

Since the browser leverage a wrapper API to provide the motion and orientation sensor readings through JavaScript, similar to the Android sensor manager API, the same reading to native apps is provided here (except for the sampling rate). This means that these sensor readings are provided *onSensorChanged* event (with lower frequency). While analysing our measurements, we noticed that the resolutions of the orientation data and the motion data are different. Due to this technical issue and for simplicity while working with this data in *Matlab*, when converting data from Excel files to text files, we created two different text files (User<no.>Motion and User<no.>Orientation) for motion and orientation separately. We repeated the same process for each user using the *Sort & Filter* feature in *Excel* as shown in the Fig. 3. As can be seen, we only include the records that we need and we exclude the unnecessary ones (e.g. interval and times).

When the text files were created for all users, we imported them to *Matlab* and performed further analysis on them as explained in our papers [1-4].



■ (Select All)
☐ interval
☑ MGX
☑ MGY
☑ MGZ
☐ Mtime
☑ MX
☑ MY
☑ MZ
☐ Otime
☐ OX
☐ OY
☐ OZ
☑ rAlpha
☑ rBeta
☑ rGama
☑ status

■ (Select All)
☐ interval
☐ MGX
☐ MGY
☐ MGZ
☐ Mtime
☐ MX
☐ MY
☐ MZ
☐ Otime
☑ OX
☑ OY
☑ OZ
☐ rAlpha
☐ rBeta
☐ rGama
☑ status

	A	B	C
1	type	data	ts
28	rAlpha	-0.07498	2017-01-18T18:04:09.848Z
29	rBeta	-0.06526	2017-01-18T18:04:09.849Z
30	rGama	0.189407	2017-01-18T18:04:09.850Z
31	interval	16.666	2017-01-18T18:04:09.851Z
32	MGX	-0.75795	2017-01-18T18:04:09.852Z
33	MGY	7.615997	2017-01-18T18:04:09.852Z
34	MGZ	7.424637	2017-01-18T18:04:09.853Z
35	OX	14.65897	2017-01-18T18:04:09.855Z
36	OY	47.57287	2017-01-18T18:04:09.858Z
37	OZ	79.06393	2017-01-18T18:04:09.858Z
38	Otime	2017-01-1	2017-01-18T18:04:09.859Z
39	Mtime	2017-01-1	2017-01-18T18:04:09.861Z
40	MX	1.053737	2017-01-18T18:04:09.862Z

User1Motion - Notepad

```
data
User Starts
Typing Begins
5113
-0.153323293
0.386777401
-0.173019409
-0.237030029
-0.092468262
0.105987549
-1.706390381
7.486312866
6.174530029
0.976254463
0.452731133
1.08318758
-0.074981689
-0.065261841
0.189407349
```

User2Orientation - Notepad

```
data
User Starts
Typing Begins
5113
17.12025963
47.97775612
77.11098732
15.78512848
47.66618216
78.15865362
14.65897249
47.5728669
79.06393373
14.3278622
47.88124277
79.43794683
14.56780031
48.43783073
79.48969718
```

Figure 3: Converting excel files to text files

References

[1] M. Mehrnezhad, E. Toreini, S. Shahandashti, and F. Hao, “TouchSignatures: Identification of User Touch Actions based on Mobile Sensors via JavaScript”, In the Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS 2015, Singapore, Apr 14-17, 2015, ACM, P 673-673.

[2] M. Mehrnezhad, E. Toreini, S. Shahandashti, and F. Hao, “TouchSignatures: Identification of User Touch Actions and PINs based on Mobile Sensors via JavaScript”, Journal of Information Security and Applications, Elsevier, V 26, Feb 2016, P 23-38.

[3] M. Mehrnezhad, E. Toreini, S. Shahandashti, F Hao, “Stealing PINs via Mobile Sensors: Actual Risk versus User Perception”, The 1st European Workshop on Usable Security, EuroUSEC 2016, Workshop at the Privacy Enhancing Technologies Symposium (PETS 2016), Jul 18, 2016, Germany.

[4] M. Mehrnezhad, E. Toreini, S. Shahandashti, F Hao, “Stealing PINs via Mobile Sensors: Actual Risk versus User Perception”, International Journal of Information Security, Springer, April 2017, Pages 1-23.