H-RAM and E-Voting

. . .

Report

. . .

Course Project: CS241
Software Engineering

Group Leader: Prateek Mohanty [1301CS33]
Don K. Dennis [1301CS17]
Ritik Prasad Mathur [1301CS38]
Sheik Sameeruddin [1301CS41]
Ritobroto Maitra [1301CS50]

April 16th, 2015
Spring Semester, 2015
IIT Patna

1 H-RAM

1.1 Motivation

The memories of hostel life form an integral part of any student's college life. Bonds are forged not only through fun and activities, but through projects undertaken and executed together. The concept of "wingmates" speaks of it all. However, the current room allocation system at IIT Patna does not take into consideration these issues. A group of students may prefer to stay together; or stay further from a particular student; or even prefer a room on a particular side of the hostel. Currently, he/she does not have the freedom to choose that. The Hostel Room Allocation and Management Service aims to bridge that gap, where a student can choose a room for himself, as well as his preferred neighbours. Moreover, it is also necessary to implement a strong concurrency control mechanism in case multiple students prefer to select a particular room.

1.2 Software Requirement Specifications

The FUNCTIONAL requirements of the service are as follows:

- The user should be able to fill a preference for rooms by selecting them from a floor plan.
- The user should be able to fill a list of preferences for neighbours.
- In case more than two users select the same room, the system should resolve it fairly.

The NON-FUNCTIONAL Requirements are:

- The room selection module should accurately represent the floor plan.
- The floor plan should not have to be entered manually completely for each separate design from scratch.
- The preference for neighbours should remain private.

1.3 Algorithm

1.3.1 Room Map Generator

Before using the actual algorithm, we need to generate a room map from the floor plan (usually available as a PDF or a JPEG file). If the administrator needs to map every area of the floor plan manually to a specific record in the database, it becomes extremely cumbersome and prone to errors. Instead, we have developed a suite of tools to help the administrator in this work.

- The administrator has to make an image map of his static image. This can be done easily by any of the several tools available on the market:
  - Linux : GIMP (FreeWare)
  - Windows : GIMP (FreeWare), Adobe PhotoShop (ShareWare), XMap (FreeWare)

  - OS X : GIMP (FreeWare), Adobe PhotoShop (ShareWare), MapSpinner (FreeWare), PaintX (Bundled)

- We have created a tool called Room Number Creator. This C-based tool gives as output a text file containing the format for room number generation.
- We have created a GUI-based tool called Bob The Builder based on Qt/C++. This takes as input parameters the output of the Room Number Generator and the image map; and creates a HTML code file.
- This finally completes the process of generating the room map, which can be used in the final user interface for room selection.

1.3.2 Room Selection

The room selection portal will be kept open for a short, specified period of time, and the users will have to register before the portal opens. Once the portal opens, the following steps take place:

- The user can select 5 friends in an order of preference. This selection is the key to allocation algorithm described in the next subsection.
- Once the user has selected the friends according to his order of preference, he will be redirected to the room selection page.
- In this step, the user can choose as many rooms as he wants according to his order of preference.
- The user can iterate over this stage as many times as he wants before the final allocation stage. Before the timer is out, he has to save his final preference. Unless his preference is saved, it will not be considered.

1.3.3 Room Allocation

Once the room selection stage is over, the server will allocate the rooms in the following manner:

- First, it discards all unsaved choices. These will be allocated rooms randomly at the end.
- For the saved choices, it iterates over all the choices, and allocates the rooms where there is no conflict. After this step, a number of rooms are expected to be filled.
- For allocating the rest of the rooms, we define the following parameters:
  - All rooms, $R_i, \forall i = 1(1)n$, a set of coordinates (x_i, y_i) are calculated initially.
  - For each friend with preference number $F_i, \forall i = 1(1)5$, a weight, w_i is calculated according to the formula $w_i = 6 - F_i, \forall i = 1(1)5$.
  - For a particular room R_i, we consider the user X. Let the number of friends of X who have already been allocated a room be n, where $0 \leq n \leq 5$.

- For each of those n friends, we calculate the Euclidean distance of the room allocated to the friend from the room R_i in question. We call this distance d_i.
- To account for the preference, we find the weighted Euclidean distance $e_i = \frac{d_i}{w_i}$, where d_i is the distance for the friend corresponding to w_i.
- We calculate a value, called the V-value, as follows : $V = \frac{n}{\frac{\sum_{i=1}^{n} e_i}{n}}$, ie, $V = \frac{n^2}{\sum_{i=1}^{n} e_i}$.
- After calculating this V-value for every user with interest in the room R_i, we allocate the room to the one having the highest V-value.
- If there exists a tie even after this, we allocate the room randomly among the contenders.

- We iterate over the entire process with one less room and one less user until all users saved their choices have been allocated rooms.
- Finally, the users did not save their choices are allocated rooms randomly from the ones remaining, completing the process.

1.4 Framework

To complete the H-RAM service, we use the Django (Python) framework using SQLite 3, client-end JavaScript to enable the room selection, C for building the room number creator and Qt/C++ for Bob the Builder.

1.5 Further Work

We may enhance the system further by including inventory management and a complaint ticketing system for issues related to the hostel. We may also allow the user to vacate a room at the end of his program or before, thus freeing the room for further reallocation.

2 E-Voting

2.1 Motivation

The E-Voting scheme is a solution to several problems faced by traditional voting schemes based on paper. The most obvious of these issues is the cumbersome process involving manual counting, and the unavoidable human errors that come with it. Moreover, paradigms worldwide are moving towards online solutions because they are more reliable, cost-effective and above all, convenient.

A simple analysis of e-voting versus paper-voting throws up the following observations:

- E-voting is certainly more convenient.

- E-voting removes the need for manual intervention, and thereby removes aspersions of errors or intentional faults in the counting system or the procedure.
- E-voting brings about *provably* secure aspects to it, if implemented smartly. item E-voting allows a much greater level of transparency about the entire procedure, without compromising the privacy of a ballot.

Our aim in implementing an e-voting scheme was to provide a viable solution addressing the above; all the while maintaining the traditional requirements of any voting scheme, which are explained in the next section.

2.2 Software Requirement Specifications

The E-voting scheme demands the following FUNCTIONAL requirements:

- The user should be able to register as a valid voter. All corresponding checks should be in place.
- The user should be able to select the candidates he wants to vote for among the ones he is eligible to vote for.
- The user should be able to verify that his vote was successfully counted.
- An aspiring candidate should be able to nominate himself with the verifiable support of two more users.
- One user should be able to cast only one vote.

In addition, the following NON-FUNCTIONAL requirements are also there:

- The system should not compromise the privacy of a vote.
- The system should not allow one user to fraud and duplicate someone else's vote.
- In case a member is pressurized to vote for a particular candidate, there must be a mechanism to prevent this.

2.3 Protocol

We propose the following protocol to address the requirements.

2.3.1 Parameters

Before we delve into the actual protocol, let us define certain parameters which will be useful to build the system.

- To address the question of security, we implement an encryption scheme, henceforth referred to in the following manner:
  - $E_k(M)$ refers to encrypting a message M using a key k. Sometimes, for convenience, we may simply write $E(M)$.

- $D_k(C)$ refers to decrypting a ciphertext C using a key k. Sometimes, for convenience, we may simple write $D(C)$. Essentially, for a symmetric key encryption scheme, $D_k(E_k(M) = M$ and for an asymmetric key scheme which uses keys k_1 and k_2, $D_{k_2}(E_{k_1}(M) = M$

- To address the question of authenticity, we use hash-function based authentication. In the current implementation, we use digital signatures based on asymmetric key cryptosystems. However, in revisions before deployment, we aim to remove the need of a key for authentication, and use Merkle-tree based authentication schemes which use derivates of the Lamport Signature Scheme.

  - $H(M) = h$ refers to a hash function H which takes as input an arbitrarily long message M and generates a fixed-length digest or a fingerprint, h.

2.3.2 Nomination Protocol

The nomination procedure works prior to the voting period for a fixed time.

- The user can nominate himself for a particular post for which he is eligible (eligibility criteria includes the current batch and hostel at this point).
- He also chooses two supporters who will support his nomination.
- Each of these supporters receives a mail containing a link where he can choose to verify his support for the nomination.
- At the end of the nomination period, the administrator checks the following:
  - If the supporters have verified from their link
  - If the candidate has no existing disciplinary action against him
  - If the candidate has a CPI > 6.0 at the time of nomination
  - If the candidate has no course backlogs.
- If the candidate is approved by all the above points, the administrator publishes his name in the final list of nominees using a module that allows him to dynamically build this list.

2.3.3 Voting Protocol

In the very beginning, we introduce of concept of Dual Passwords. We believe this is important, as it addresses an issue that is more social rather technical.

The framework is described as follows:

- Good Password The user will use this password when he is voting out of free will, and his vote will be counted.
- Evil Password The user will use this password when under duress, and although the user interface will not react to recognize this change, his vote will not be counted.

Using these tools, let us now describe the voting protocol in detail. All communications back and forth between the client and the server are over a secure communications channel, and is not mentioned in every step separately.

- The user chooses his preferred list of candidates, B (ballot) and authenticates himself using the password P.
- On submission of his vote, a number R is generated as the hash of his ID, ballot and a random salt, ie $R = H(ID|B|S)$. The salt is generated using a client side JavaScript based Mersenne Twist Generator. He sends to the server $(ID|B|R|P)$. [The symbol | denotes concatenation.]
- On the server side, if P is checked to be the good password, a *flag* against the user is set to 1 (default 0) - to indicate the user has participated in the voting process. Then, the number R is stored separately, and the ballot B is signed and counted after verification from the user end. If the bad password is used or if the flag is already set to 1, nothing happens.
- Once the voting period is over, the user can query the system with R. If the flag is set to 1 and R is with the server, then the user can verify that his vote was counted.

The implications of this protocol are as follows:

- It provides authentication.
- It provides fraud detection - by checking the password associated with an ID, we can be assured that only the designated person is voting.
- Once the flag has been set to 1, the user cannot vote again. This provides uniqueness.
- The system can know which persons have voted from the flag, but since the ballot B itself is not stored, it can not correlate a vote with a voter. Moreover, even from the number R, it is computationally infeasible to find out the details.
- The voter can verify his vote.
- The dual-password scheme provides an effective anti-rigging mechanism where a voter under pressure can just vote using his bad password.

2.3.4 Framework

Our work is based on the Django (Python) framework using SQLite3, using SJCL and PyCrypto Libraries and independent implementations of ECC, Rijndael-128 and SHA256.

2.4 Further Work

While the system is self-sufficient, further work may be focused in the following areas:

- We can introduce the concept of blinding during the voting stage to provide full anonymity even from a sufficiently powerful adversary.

- We can introduce Merkle-tree based Keyless Signature Infrastructure to do away with keys, as key-management and storage is a weak link for an attacker to focus on.
- Currently, the administrator has to check the course backlogs and CPI manually. We can integrate the system to communicate with the academic records to do this automatically.

3 Summary

H-RAM and E-Voting uses:

- Django as the main framework
- SQLite 3 for database
- Python-based implementations of algorithms and schemes
- C-based implementation for Room Number Generator
- Qt/C++ - based implementation for Bob the Builder
- JavaScript based implementation for various client-end services
- JQuery library for Mapster
- Twitter-BootStrap on HTML5 for front-end design
- Git for version control

Total Lines of Code: 26,789.

Moreover, working in a team - starting from ideas to implementation and finally testing and debugging - allowed us to learn and develop our skills in a more complete way than it would have been possible otherwise.

4 Acknowledgements

We acknowledge the support, criticism and encouragement of Dr. Samrat Mondal, Mr. Nilotpal Chakraborty and Mr. Sumit Mishra during the development of the software. Our ideas on e-voting are based mostly on the following public-domain articles, papers and technical reports, in addition to various IRC and Git brainstorms :

- *Accuracy, Integrity, and Security in Computerized Vote-Tallying*; Saltman, Roy G.; National Bureau of Standards, New York, NY 10019-6908; NBS Special Publication; August 1998
- *A Practical Secret Voting Scheme for Large Scale Elections*; Fujioka, A.; Tatsuaki, O.; Kazuo, O; NTT Laboratories, Tokyo; Journal of Cryptology, 2004
- *CryptoBytes*; Rivest, Ronald L.; Volume 7, No. 2, Fall 2004; RSA Laboratories.