Background

**SELS**
- Increasing use of E-mails List Services for private data exchange.
- Need for Secure ELS which can provide secure exchange of e-mails
- SELS is a protocol providing confidentiality, integrity and authentication functionalities

**Test Bed**
- Proxy Encryption Techniques are used for achieving Deployable Secure Mailing Lists
- Correctness of SELS package is critical as it is a security provider
- Need for an Automated Test Bed and code analysis to ensure error free working

Goals

- Should support Multiple platforms, mail clients and Java versions
- Comprehensive Code analysis for tracing faults and failures
- Design & Development of Automated Test Bed
- Code Validation and Verification
- Developing Integration Test Modules

Challenges

- Design decisions pertaining to available Python Libraries and Tools to develop a robust Test Bed
  - *Pexpect* is not supported on Windows OS
- How to handle the Input Interface? What should be the design of the Automated Input Provider?
  - *NFA Model* : Comparing each input prompt against the entire input prompt space. Enhances maintainability
  - *DFA Model* : Using the control flow to compare input prompt against only the expected input sub-space. Easy integration with the testing modules
- How to ensure maintainability of the test suite for future releases of SELS?
  - Use of *PyUnit*, the Standard Testing Framework and Modular Design
- Is the Test Bed itself fault free?
  - Use of Waterfall Model with Prototyping to minimize errors
- How to improve reliability of the SELS code?
  - Good Exception Handling and Localization of the effect of errors

Research Plan

- Learning SELS, required technologies, COTS security tools and components used
- Installing Apache, Sendmail, Mailman, SELS LS and LM on Linux machine and establishing interfaces between them
- Implementing a virtual shell emulator for Automated Input Provider using only the Standard Python Libraries
- Deploying PyUnit framework for developing Unit Test modules and implementing Automated Integration Tests
- Testing on Linux, MacOs and Windows with five different mail clients and manual code analysis

Design of SELS Automated Test Bed

\[ \text{Design of SELS Automated Test Bed} \]

Research Results

- Successfully developed an Automated Input Provider and Automated Test Bed
- Wrote scripts to Automate buildLScode, createLLkey, createListkey, subscribeUser, subscribeBatchUser
- Performed testing on Linux, MacOs, Windows and three major Email Clients using Automated Test Bed

<table>
<thead>
<tr>
<th></th>
<th>UNIT TESTING</th>
<th>INTEGRATION TESTING</th>
<th>CODE ANALYSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRITICAL</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MAJOR</td>
<td>4</td>
<td>2</td>
<td>(1 TPS)</td>
</tr>
<tr>
<td>MINOR</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

Testing Results

\[ \text{Testing Results} \]

Screen Shot of Automated Test Bed

\[ \text{Screen Shot of Automated Test Bed} \]

Related Work

- Secure Email
  - *PGP*, *S/MIME*
- Commonly used mailing lists
  - Listserv, majordomo, mailman, sympa
- Testing models
  - *Spiral, Butterfly, W*
- Testing tools
  - Proctor, pytest, coverage.py, PyLint