Secure Passwords with Hidden Information and Non-Exposure Shyamsunder Meda, Saikumar Sirikonda, Chandan Teja Vangapati, Dr. T Shyam Prasad

1 IV year Department of Computer Science And Engineering ,2 IV year Department of Computer Science And Engineering ,3 IV year Department of Computer Science And Engineering, 4 Assistant Professor - Department of Computer Science and Engineering

Anurag University, Ghatkesar, Hyderabad, India

1 [20eg105360@anurag.edu.in](mailto:20eg105360@anurag.edu.in), 2[20eg105346@anurag.edu.in](mailto:20eg105346@anurag.edu.in) 3[20eg105353@anurag.edu.in,](mailto:20eg105353@anurag.edu.in,) 4 [shyamprasadcse@anurag.edu.in](mailto:shyamprasadcse@anurag.edu.in)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Abstract** - In the realm of password management, the SPHINX system introduces a groundbreaking approach that ensures security even in the event of a compromise. Leveraging the device-enhanced password authenticated key exchange (DE-PAKE) model, SPHINX separates the stored information from the user’s master password, preventing any leakage even when the device is under full control. This is achieved through the use of an efficient oblivious pseudo-random function (OPRF) scheme, transforming user memorized passwords into high-entropy, mutually independent, and random passwordswithout storing them on the device.

**Index Terms** – SPHINX, Password management, Device-enhanced, Cryptographic proofs, OPRF, High-entropy passwords, Online guessing attacks, Offline dictionary attacks, Server compromise, Master password. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. INTRODUCTION**

**1.1 Overview**

The project described is centered around the concept of Device-Enhanced Password Authenticated Key Exchange (DE-PAKE), which forms the basis for the SPHINX  application. DE-PAKE aims to enhance password security by transforming user memorable passwords into high-entropy random strings using a secondary device,  thereby mitigating vulnerabilities associated with low-entropy passwords. The protocol

involves four parties: the user (U), client (C), server (S), and device (D). The DE-PAKE protocol consists of two phases: initialization and authenticated key  exchange. During initialization, the user selects a password from a dictionary and  communicates with the device and server to establish necessary states for  authentication. In the authenticated key exchange phase, the user interacts with the  device, client, and server to authenticate and establish session keys, protecting  communication with the server.

The SPHINX instantiation of DE-PAKE assumes the availability of a smartphone  during the authentication process, providing improved security properties compared to  regular password-only authentication. However, it acknowledges limitations in  protecting against client compromise, suggesting future integration with two-factor  authentication (TFA) solutions.

The project's contribution over its conference publication includes an extended  analytical comparison of SPHINX with other password managers based on security,  usability, and deployability metrics. Additionally, it presents a comparison of user task  flows in SPHINX, password-only authentication, and other device-based password  managers.

Future work could involve extending SPHINX to support an online instantiation,  providing further security against device compromise and man-in-the-middle attacks.  However, building and testing the full implementation of such an extension are beyond  the current scope.

**1.2 Problem Statement:**

Existing password management systems are susceptible to compromise, leaving  users vulnerable to various cyberattacks. Traditional solutions often generate  low-entropy passwords and rely on plaintext entry of master passwords, making  them targets for online guessing and offline dictionary attacks. Additionally,  compromised password managers can lead to unauthorized access to sensitive  user information, posing significant risks to individuals and organizations alike.  With the ever-increasing sophistication of cyber threats and the widespread  adoption of digital services, the need for robust password management solutions  has become more critical than ever. Addressing these vulnerabilities is essential  to safeguarding user credentials, maintaining privacy, and protecting against  financial loss and reputational damage.

2

**1.3 Problem Illustration:**

|  |
| --- |
| Consider a scenario where a user, Alice, relies on a traditional password manager to store and generate passwords for her various online accounts. Despite her best efforts to create strong passwords, the password manager she uses has inherent vulnerabilities that leave her susceptible to cyberattacks.  One day, Alice receives a phishing email that tricks her into clicking on a malicious link, compromising her password manager's security. Unbeknownst to Alice, the attacker gains access to her master password and all the stored credentials for her online accounts. With full control over Alice's password manager, the attacker can now freely access and exploit her sensitive information.  Furthermore, the passwords generated by Alice's password manager have low entropy, making them vulnerable to brute-force attacks and dictionary-based hacking techniques. Even if Alice had used unique passwords for each account, the compromised password manager puts all her accounts at risk of unauthorized access and potential data breaches.  As a result of the security breach, Alice faces significant consequences. Her personal and financial information could be stolen, leading to identity theft, financial fraud, or unauthorized access to sensitive data. Moreover, Alice's trust in online security is shattered, impacting her confidence in digital services and her overall peace of mind.  This scenario highlights the pressing need for password management solutions that offer robust security measures to protect user credentials, even in the face of compromise. Without such solutions, individuals like Alice remain at risk of falling victim to cyberattacks and the devastating consequences that follow. |

**2. LITERATURE SURVEY**

The landscape of password management has evolved significantly over the years, with  SPHINX emerging as a novel application of Device-Enhanced Password-Authenticated  Key Exchange (DE-PAKE) principles. This survey provides an overview of existing literature in the field, highlighting key research contributions and comparingthemwith the features and advantages offered by SPHINX. 

|  |
| --- |
| **Traditional Password Managers:** |

Traditional password managers have been widely  used, offering basic password storage and generation capabilities. However, they often  rely on low-entropy passwords and plaintext entry of master passwords, leaving users  vulnerable to various cyber threats, including phishing attacks, dictionary attacks, and  compromise of the password manager itself. 

|  |
| --- |
| **DE-PAKE Principles:** |

DE-PAKE, as introduced in [31], provides a foundation for  SPHINX by securely transforming user-memorable passwords into high-entropy  random strings. This cryptographic primitive addresses the limitations of traditional  password management systems by leveraging a secondary device to generate strong,  unique passwords for each service. 

|  |
| --- |
| **Security Features of SPHINX:** |

SPHINX offers several key security guarantees  simultaneously, including resistance to online guessing attacks, offline dictionary  attacks under server and device compromise, phishing attacks, and eavesdropping or  man-in-the-middle attacks on the device-client channel. These features enhance the  overall security posture of password management systems and mitigate common  vulnerabilities associated with traditional approaches. 

|  |
| --- |
| **Usability Advantages of SPHINX:** |

In addition to its robust security features, SPHINX  offers usability advantages such as the use of human-memorable passwords, easy  password updates through device key management, and compatibility with multiple  types of devices, including smartphones, wearables, and online services. These  usability enhancements improve user experience and facilitate seamless integration into  daily routines.

|  |
| --- |
| **Comparative Analysis:** |

Comparative evaluations of SPHINX with traditional  password managers and other password management schemes highlight the unique  advantages offered by SPHINX. While traditional solutions may provide some security  and usability features, SPHINX stands out for its comprehensive security guarantees  and enhanced usability, particularly in scenarios involving compromised servers or  devices. 

|  |
| --- |
| **Future Directions:** |

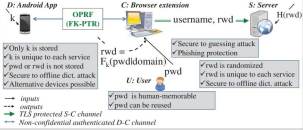
Future research in password management may explore further  

extensions and applications of DE-PAKE principles, as demonstrated by SPHINX.  Additionally, ongoing efforts to improve password security and usability will continue  to shape the landscape of password management solutions, with SPHINX serving as a  compelling example of innovation in the field.

In summary, SPHINX represents a significant advancement in password management,  offering a comprehensive solution that addresses both security and usability challenges  inherent in traditional approaches. Its innovative application of DE-PAKE principles  and robust security features position it as a promising solution for enhancing password  security in an increasingly digital world.

**Existing Method:**

Cracking-resistant password encoding strategies have been proposed in the literature to  render offline dictionary attacks ineffective. They introduce the notion of out putting  decoy passwords to an attacker who compromises the manager and attempts to decrypt  the passwords with a wrong master password. Since the attacker is not aware of the  correct password, any attempt to login with the decoy passwords can be prevented on  the server, and raise an alert.



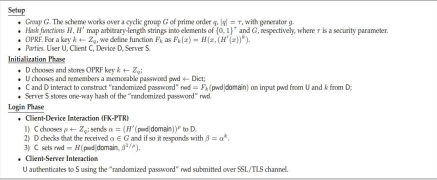
*Figure 2.1. Existing method*

|  |  |  |  |
| --- | --- | --- | --- |
| SI.NO | Strategies | Advantages | Disadvantages |
| 1 | On the security of cracking resistant password vaults. | investigate the construction of  encrypted vaults that resist  such offline cracking attacks  and force attackers instead to  mount online attacks | A password vault can  greatly reduce the  burden on a user of  remembering passwords,  but introduces a single  point of failure |
| 2 | Round-optimal password protected secret sharing and  T-PAKE in the password  only model | the system is secure against  offline password attacks by an  attacker controlling up to t  servers | Time taking and require  sever authentication |
| 3 | Device Enhanced Password  Protocols with Optimal  Online-Offline Protection. | An attacker taking over the  device still requires a full  online attack to impersonate  the user. | Hardware device is  required for using this  method. |
| 4 | A Comparative Usability  Evaluation of Traditional  Password Managers | users were not comfortable  giving control of their  passwords to an online entity  and preferred to manage their  passwords themselves on their  own portable device | Use apps to manage user  passwords which is not  secure |
| 5 | Cracking resistant password  vaults using natural  language encoders. | a new type of secure encoding  scheme that we call a natural  language encoder (NLE). An  NLE permits the construction  of vaults which, when  decrypted with the wrong  master password, produce  plausible looking decoy  passwords. | Use automated NLP  technique which is  defied by system |
| 6 | The emperor’s new  password manager: Security  analysis of web-based  password managers | We identify four key security  concerns for web-based pass word managers and, for each,  identify representative  vulnerabilities through our  case studies | Data analysis on various  attacks is performed |

*Table 2.1. Literature Survey*

**3. PROPOSED METHOD**

**3.1. Illustration**

****

*Figure 3.1.1.* SPHINX

**3.2. Authentication and User Management:**

**3.2.1. User Registration:**

User registration is the process by which individuals sign up for access to the password  management system. During registration, users provide their email address and create  a master password. This master password serves as the primary authentication  credential for accessing the system and managing their passwords. Additionally, to  verify the authenticity of the provided email address and enhance security, an OTP  (one-time password) is sent to the user's email during the registration process. The user  must enter this OTP to confirm their email address and complete the registration process  successfully. This verification step helps ensure that only users with valid email  addresses can register for the system, mitigating the risk of unauthorized access and  account creation by malicious actors.

**3.2.2. User Login:**

User login enables registered users to access their accounts and utilize the  functionalities of the password management system. To log in, users enter their  registered email address and master password into the login interface. This master  password is securely stored and used for authentication purposes. In addition to the  master password, an extra layer of security is implemented through OTP  verification. Upon entering their credentials, a one-time password is generated and  sent to the user's registered email address. The user must then retrieve this OTP  from their email and enter it into the login interface to authenticate their identity  fully. This two-factor authentication mechanism enhances the security of the login  process, reducing the risk of unauthorized access even if the user's password is  compromised.

**3.2.3. Admin Panel:**

The admin panel provides administrators with the necessary tools and  functionalities to manage user accounts and oversee the operation of the password  management system. Within the admin panel, administrators have the capability to  view and manage user accounts, including creating new accounts, modifying  account details, and deactivating or deleting accounts as needed. Additionally,  administrators can reset passwords for user accounts in cases where users forget  their master passwords or require assistance with account access. Furthermore, the  admin panel allows administrators to manage user roles, assigning specific  permissions and privileges to different user accounts based on their roles within the  organization or system. This role-based access control enhances security and  ensures that users have appropriate levels of access to system resources and  functionalities based on their responsibilities and requirements. Overall, the admin  panel serves as a centralized hub for administrators to efficiently manage user  accounts and maintain the security and integrity of the password management  system.

**3.3. Password Management:**

**Password Encryption:**

Password encryption is a crucial aspect of the password management system, ensuring  that user passwords are securely stored and protected from unauthorized access. The  process involves several steps:

**RSA Key Pair Generation:**

Upon user registration, a unique RSA key pair is generated for each user. This  consists of a public key and a private key.

**OPRF Technique for Key Derivation:**

The user's master password is used as input to the Oblivious Pseudo-Random  Function (OPRF) technique.

OPRF derives a cryptographic key from the master password without revealing the  password itself.

**Encryption of Derived Key:**

The derived cryptographic key is encrypted using the user's public RSA key. This ensures that only the corresponding private RSA key, held by the user, can  decrypt the key.

**AES Encryption for Password Storage:**

User passwords are encrypted using Advanced Encryption Standard (AES)  encryption.

The encrypted passwords are stored in the database, ensuring that even if the database  is compromised, passwords remain secure.

**Password Retrieval:**

When a user needs to retrieve a password, the stored encrypted key and password are  decrypted using the following steps:

**Decryption of Stored Key:**

The encrypted key stored in the database is decrypted using the user's private RSA  key. This retrieves the original derived cryptographic key.

**Decryption of Encrypted Password:**

The retrieved cryptographic key is used to decrypt the encrypted password stored in  the database. Th decrypted password is then made available to the user. **Password Generation:**

The password management system also provides functionality for generating strong,  random passwords. Users can customize various parameters for the generated  passwords, including:

**Length:** Users can specify the desired length of the generated password, allowing for  flexibility based on security requirements or platform constraints. **Character Set:** Users can choose the character set from which the password will be  generated, including options such as uppercase letters, lowercase letters, numbers, and  special characters.

By offering these customization options, users can generate passwords tailored to  their specific needs while ensuring they meet the desired level of security.  Additionally, the use of randomness in password generation helps mitigate the risk of  predictability and enhances the overall strength of generated passwords, further  bolstering the security of the password management system.

**3.4. OTP Verification:**

**OTP Generation and Sending:**

OTP (One-Time Password) generation and sending are integral parts of the two-factor  authentication process implemented in the password management system. This  process involves the following steps:

**OTP Generation:**

Upon user login or any action requiring OTP verification, a unique one-time password  is generated by the system. The OTP is typically a randomly generated string of  alphanumeric characters or digits, ensuring unpredictability and security.

**Sending OTP to User's Registered Email:**

Once generated, the OTP is sent to the user's registered email address. The email serves as a secure channel for delivering the OTP to the user, leveraging existing communication infrastructure and ensuring that the OTP remains  confidential.

**User Login Attempt:**

When users attempt to log in or perform a sensitive action requiring OTP verification,  they are prompted to enter the OTP received via email.

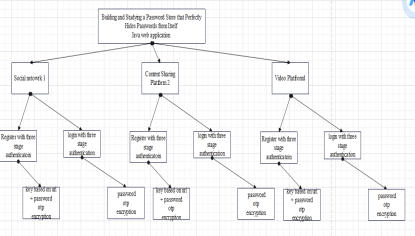
**Entering OTP:**

Users retrieve the OTP from their email inbox and enter it into the designated field in  the login interface. The OTP entered by the user is compared against the OTP  generated and sent by the system.

**Validation and Authentication:**

If the entered OTP matches the one generated and sent by the system and it is within  the valid time period, authentication is successful. Upon successful OTP verification,  users are granted access to their accounts or allowed to perform the requested action.

OTP verification adds an additional layer of security to the authentication process,  requiring users to possess both their master password and access to their registered  email account. This two-factor authentication mechanism enhances the overall  security posture of the password management system, reducing the likelihood of  unauthorized access by malicious actors even in the event of password compromise.



*Fig 3.2.1. Authentication and User Management*

**3.5. USER INTERFACE:**

**Chrome Extension Interface:**

The Chrome extension interface provides users with a seamless and intuitive  experience directly within their web browser. This interface is designed to be user friendly and straightforward, allowing users to easily access and manage their  passwords. Key features of the Chrome extension interface include:

**Simple and Intuitive Design:**

The interface features a clean and minimalist design, ensuring that users can quickly  navigate and understand its functionalities. Intuitive navigation elements, such as  buttons and menus, make it easy for users to perform actions like saving, retrieving,  and generating passwords.

**Password Management Options:**

Users can save passwords for various online accounts directly from the extension  interface. Retrieval of saved passwords is also straightforward, with users able to  quickly access their stored passwords whenever needed. Additionally, the extension  provides functionality for generating strong and random passwords, offering users an  easy way to create secure passwords for new accounts or password updates. **Web Interface:**

The web interface complements the Chrome extension by providing administrators  with access to the admin panel and user account management functionalities. This  web interface is built using Django, offering a robust and feature-rich platform for  managing the password management system. Key aspects of the web interface  include:

**Admin Panel Access:**

Administrators can access the admin panel through the web interface to perform  various administrative tasks, such as managing user accounts, resetting passwords,  and assigning user roles. The admin panel provides administrators with a centralized  dashboard for overseeing system operations and user activities.

**Responsive Design:**

The web interface is designed with responsiveness in mind, ensuring that it adapts  seamlessly to different screen sizes and devices. Whether accessed from a desktop  computer, tablet, or smartphone, the web interface provides a consistent and  optimized user experience.

By combining the Chrome extension interface for users with the web interface for  administrators, the password management system offers a comprehensive solution for  securely managing passwords. Users benefit from a user-friendly interface within  their web browser, while administrators have access to powerful management tools  through the web interface, all built on the Django framework for stability and  scalability.

**3.6. SECURITY MEASURES**

**Data Encryption:**

In our implementation, we prioritize the security of stored passwords by employing  AES encryption. This encryption technique ensures that passwords stored in the  database are protected from unauthorized access, even if the database is compromised.  By encrypting passwords using AES, we add an extra layer of defence, making it  extremely difficult for attackers to decipher sensitive information.

Additionally, we enforce secure communication between the Chrome extension and the  server by using HTTPS protocol. This ensures that data transmitted between the client  and server is encrypted, preventing eavesdropping and man-in-the-middle attacks. By  adhering to HTTPS standards, we maintain the confidentiality and integrity of user data  throughout the communication process.

**Input Validation:**

To mitigate common security vulnerabilities such as cross-site scripting (XSS) and  SQL injection, we implement rigorous input validation mechanisms. Our system  validates user input at various entry points, including registration forms, login  interfaces, and password retrieval requests. By sanitizing and validating user input, we  prevent malicious actors from injecting malicious scripts or SQL queries into our  system. This helps safeguard against potential attacks aimed at exploiting  vulnerabilities in our application, ensuring the integrity and security of user data.

**Rate Limiting and Brute Force Protection:**

We incorporate rate limiting measures to mitigate the risk of brute force attacks  targeting login and OTP verification processes. By limiting the number of login  attempts within a specific timeframe, we prevent malicious actors from repeatedly  guessing user credentials or OTPs. This helps safeguard user accounts from  unauthorized access attempts and strengthens overall system security.

Additionally, we implement brute force protection mechanisms to detect and respond  to suspicious login patterns indicative of brute force attacks. By monitoring login  attempts and identifying anomalous behaviour, our system can automatically block or  throttle access for suspicious IP addresses, mitigating the impact of brute force attacks  on system security. These proactive measures help fortify our system against brute force  attacks and enhance the overall security posture of our password management solution.

**5. IMPLEMENTATION**

**5.1. MODULES:**

**5.1.1. Overall Implementation:**

**Frontend Module:**

In the frontend module of our project, we focus on developing the user interface  components for the Chrome extension. This involves leveraging web technologies such  as HTML, CSS, and JavaScript to create an intuitive and user-friendly interface. We  design and implement various functionalities essential for password management  directly within the extension interface. This includes features like user registration,  login, password management, OTP verification, and password generation. Through  carefully crafted UI elements and interactive elements, we aim to provide users with a  seamless experience while interacting with the password management system.  Additionally, we ensure that the frontend components are responsive and compatible  across different devices and screen sizes, optimizing accessibility and usability for all  users.

**Backend Module:**

In the backend module of our project, we establish the foundation for handling the core  functionalities and business logic of the password management system. We set up a  Django project, a powerful web framework for Python, to facilitate backend  development. Within this Django project, we implement RESTful APIs using Django  REST Framework, enabling seamless communication between the frontend and  backend components of the system. These APIs serve as the bridge for transmitting data  and requests between the client-side extension and the server-side application.  Additionally, we focus on implementing robust business logic, data validation  mechanisms, and database interactions within the backend. This includes tasks such as  user authentication, password encryption, OTP verification, and user management. By  meticulously designing and implementing the backend functionalities, we ensure the  security, reliability, and scalability of our password management system, laying a solid  foundation for its overall operation.

**5.1.2. User Module:**

**User Authentication:**

Within our system, user authentication serves as the first line of defense against  unauthorized access. We implement a robust authentication mechanism utilizing both  the master password and OTP verification, ensuring a multi-layered approach to user  verification. This involves developing endpoints dedicated to user registration, login,  and logout functionalities, enabling users to securely access their accounts. By  incorporating OTP verification alongside the master password, we enhance the security  posture of our system, mitigating the risk of unauthorized access. Through meticulous  development and testing, we strive to deliver a seamless authentication experience,  instilling confidence in users regarding the protection of their accounts and sensitive  information.

**Password Management:**

Efficient and secure password management lies at the heart of our system's  functionality. We implement a comprehensive suite of functionalities geared towards  facilitating the seamless management of passwords for individual users. This includes  features for saving, retrieving, and generating passwords, empowering users to  effortlessly manage their credentials. To ensure the security of stored passwords, we  implement robust encryption techniques and adhere to best practices for secure storage  in the database. By prioritizing security without compromising usability, we aim to  provide users with a reliable and intuitive platform for managing their passwords  effectively.

**User Interface Enhancements:**

User interface enhancements play a pivotal role in shaping the overall user experience  of our password management system. We prioritize the design of user-friendly  interfaces that facilitate seamless interaction with password management features.  Through thoughtful interface design, we aim to streamline user workflows and enhance  usability. Additionally, we incorporate features such as password strength indicators  and customization options for password generation, empowering users with tools to  create and manage strong, unique passwords effortlessly. By continuously refining and  optimizing the user interface, we strive to deliver a cohesive and intuitive user  experience that fosters user engagement and satisfaction.

**5.1.3. Admin Module:**

**Admin Module:**

Within our system, the admin module is dedicated to facilitating administrative tasks  and ensuring the smooth operation of the password management system. At its core,  the admin module focuses on implementing role-based access control (RBAC) to  distinguish between regular users and administrators, granting specific privileges to the  latter. This includes defining admin privileges for managing user accounts, resetting  passwords, and performing other administrative tasks essential for system management  and security.

**Admin Privileges:**

One of the primary objectives of the admin module is to enforce role-based access  control, delineating distinct roles and privileges for administrators. Through RBAC, we  establish granular control over administrative actions, allowing administrators to  perform tasks like managing user accounts, resetting passwords, and overseeing system  settings. By defining specific admin privileges, we ensure that administrative actions  are performed by authorized personnel, mitigating the risk of unauthorized access or  misuse of administrative functionalities.

**Admin Panel:**

To facilitate administrative tasks, we develop a dedicated admin panel interface  accessible only to authorized administrators. This admin panel serves as a centralized  hub for administrators to perform various administrative tasks efficiently. We  meticulously design interfaces within the admin panel for managing user accounts,  viewing activity logs, and accessing system settings. By providing administrators with  intuitive and user-friendly interfaces, we empower them to navigate and utilize  administrative functionalities seamlessly, enhancing productivity and efficiency in  managing the password management system.

**Security Enhancements:**

In addition to facilitating administrative tasks, the admin module incorporates robust  security enhancements to safeguard sensitive administrative functionalities. This  includes implementing additional security measures such as multi-factor authentication  (MFA) and access control lists (ACLs) for admin functionalities. MFA adds an extra  layer of security by requiring administrators to authenticate their identity through  multiple factors, reducing the risk of unauthorized access. Furthermore, ACLs enable  fine-grained control over access to administrative functionalities, ensuring that only  authorized personnel can perform sensitive actions. By prioritizing security in admin  functionalities, we reinforce the overall security posture of the password management  system, safeguarding against potential security threats and unauthorized access.

**5.1.4. Deployment:**

As part of the deployment phase, our implementation plan entails deploying the Chrome  extension to the Chrome Web Store for public distribution, ensuring widespread  accessibility for users. Simultaneously, we deploy the backend server to a reliable  hosting platform, prioritizing scalability and availability to accommodate varying user  loads and maintain uninterrupted service. This involves configuring the server  environment, database, and network settings meticulously for production deployment,  guaranteeing optimal performance and security. By organizing the implementation plan  into modules, we streamline development efforts, enabling focused attention on specific  functionalities and components of the password management system. This structured  approach fosters collaboration among team members, ensuring comprehensive  coverage of all project aspects and ultimately delivering a robust and user-friendly  password management solution to our users.

|  |
| --- |
| **AES Encryption for Password Storage:** |

• After hashing the password into an elliptic curve using the browser extension's hashing mechanism, the resultin password (P) is encrypted using AES encryption:

• *Pencrypted*=*AESencrypt*(*P*,*keyAES*)

• Where:

|  |
| --- |
| *Pencrypted* |

is the encrypted password. 

• is the hashed password.

|  |
| --- |
| *P* |

• is the AES encryption key.

|  |
| --- |
| *keyAES* |

|  |
| --- |
|  |

37

2.

|  |
| --- |
| **RSA Authentication for Secure Communication:** |

|  |
| --- |
| • To authenticate users and ensure secure communication, RSA encryption is used for encrypting user credentials (username and password) before transmission:  • *C*=*RSAencrypt*(*Credentials*,*public*\_*keyRSA*)  • Where: |

• is the encrypted credentials.

|  |
| --- |
| *C* |

• is the concatenated user credentials.

|  |
| --- |
| *Credentials* |

• is the public key of the system.

|  |
| --- |
| *public*\_*keyRSA* |

3.

|  |
| --- |
| **Browser Extension for Hashing Password:** |

|  |
| --- |
| • The browser extension hashes the input password (P) into an elliptic curve using a "Hash-into-Elliptic-Curve" function: *H*(*P*)=*HEC*(*P*) • Where: |

• is the hashed password.

|  |
| --- |
| *H*(*P*) |

• is the hash function for mapping the password onto an

|  |
| --- |
| *HEC* |

elliptic curve.

4.

|  |
| --- |
| **Integration of Technologies:** |

|  |
| --- |
| • The integration of AES encryption, RSA authentication, and the browser extension's hashing mechanism ensures robust security measures for password management: |

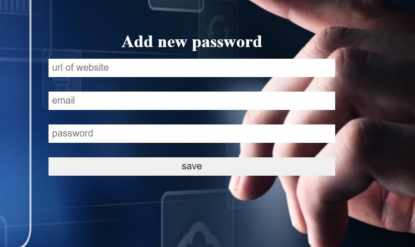
|  |
| --- |
| • AES encryption protects stored passwords from unauthorized access:  • *Pencrypted*  • RSA encryption ensures secure communication by encrypting user credentials: *C*  • The browser extension enhances password security during input by hashing passwords onto an elliptic curve: *H*(*P*) |

**6. EXPERIMENT SCREENSHOTS**

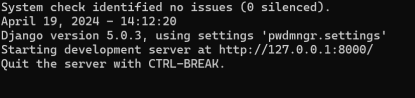
**6.1 Home Page**

****

**6.2 Adding Passwords:**

****49

**6.3 Server:**

****

50

**7. OBSERVATIONS**

• **Experimental Setup**

**Software Requirements:**

• Operating system : Windows XP/7/10.

• Coding Language : Python, Django

• Tool : Visual Studio

• Database : PostgreSQL

**Hardware Requirements:**

• System : Pentium IV 2.4 GHz.

• Floppy Drive : 1.44 Mb.

• Ram : 1 GB

**Parameters with formulas:**

1. **Device-enhanced Password Authenticated Key Exchange (DE-PAKE)**

|  |
| --- |
| **Model Parameters:** |

• : User's master password.

|  |
| --- |
| *P* |

• : Stored information.

|  |
| --- |
| *S* |

• : Secret key derived from the user's master password and the stored

|  |
| --- |
| *K* |

information.

• (): One-way function used in the DE-PAKE protocol.

|  |
| --- |
| *f* |

2.

|  |
| --- |
| **Oblivious Pseudo-Random Function (OPRF) Scheme Parameters:** |

• : User's master password.

|  |
| --- |
| *x* |

• : Randomized password generated by the OPRF scheme.

|  |
| --- |
| *y* |

• (): OPRF function used for password transformation.

|  |
| --- |
| *g* |

51

3.

|  |
| --- |
| **AES Encryption Parameters:** |

• ′′: Encrypted password.

|  |
| --- |
| *P* |

• : Initialization Vector.

|  |
| --- |
| *IV* |

• (): AES encryption function.

|  |
| --- |
| *E* |

4.

|  |
| --- |
| **RSA Encryption Parameters:** |

• : Public key for encryption.

|  |
| --- |
| *PKpub* |

• : Private key for decryption.

|  |
| --- |
| *PKpriv* |

• : Cipher text.

|  |
| --- |
| *C* |

• : Message to be encrypted.

|  |
| --- |
| *M* |

• (): RSA encryption and decryption function.

|  |
| --- |
| *RSA* |

5.

|  |
| --- |
| **Salted One-way Hash Parameters:** |

• (): Hash function.

|  |
| --- |
| *H* |

• : Random value added to the password before hashing.

|  |
| --- |
| *salt* |

• : Resulting hashed password.

|  |
| --- |
| *hash* |

|  |
| --- |
|  |

6.

|  |
| --- |
| **Domain-specific Parameters:** |

• : Website domain.

|  |
| --- |
| *D* |

• Function incorporating website domain into password

|  |
| --- |
| *fD*(): |

computations.

|  |
| --- |
| **Formulas:** |

1. DE-PAKE:

|  |
| --- |
| *K*=*f*(*P*,*S*) |

2. OPRF:

|  |
| --- |
| *y*=*g*(*x*) |

3. AES Encryption: ′

|  |
| --- |
| *P*′=*E*(*P*,*IV*) |

4. RSA Encryption:

|  |
| --- |
| *C*=*RSA*(*PKpub*,*M*) |

5. Salted One-way Hash:

|  |
| --- |
| *hash*=*H*(*P*∣∣*salt*) |

6. Domain-specific Computation:

|  |
| --- |
| *fD*(*x*) |

52 

**8. SPHNIX vs Other Password Managers**

**TABLE 8.1 :** SPHINX vs. Other Password Managers. “ODA-Resistance” denotes  resistance to offline dictionary attacks.

**Notes:** ± Enhanced by the use of two uniqueness parameters, domain name and OPRF  key. They provide higher resistance compared to password-only, but still an ODA is  possible after the phishing attack. Unless the user chooses a randomized master  password. ∗ Unless the user chooses a randomized password for each account. †  Establishment of confidential channel necessary. ‡ Establishment of confidential and  authenticated channel necessary. • Establishment of confidential and authenticated  channel necessary.

53

**9. CONCLUSION:**

Passwords are a “necessary evil”. In this paper, we attempted to respond to the growing  security and usability problems with passwords by proposing SPHINX, a cryptographic  password manager that can address most security and usability problems with  passwords from the client/user side alone (i.e., transparent to most existing web  authentication services). SPHINX is a password management approach, built atop an  existing oblivious PRF (OPRF) scheme, that transforms a human-memorable password  into a random password with the aid of a device without the need to store the passwords  on the device. SPHINX offers 12 several key security guarantees, namely, resistance  to: (1) online guessing attacks, (2) offline dictionary attacks under server compromise,  (3) offline dictionary attacks under device compromise, (4) phishing attacks, and (5)  eavesdropping and man-in-the-middle attacks on the device-client channel. SPHINX  also boasts to provide almost the same level of user experience as that of authentication  using an easy to memorize password. Unlike other password managers, SPHINX  perfectly hides passwords and the master password from itself, and thus remains secure  under the realistic threat of the compromise of password managers. Also, unlike other  password managers, SPHINX does not require a confidential device-client channel. At  the same time and like many other password managers, SPHINX can resist online  guessing, offline dictionary under web service compromise and phishing attacks. We  designed and implemented a smartphone-based instantiation of SPHINX. Our  performance and analytical evaluation of this instantiation shows that it is efficient,  highly secure, likely simple to use, and easy to deploy in practice.

54

**10. REFERENCES / BIBILOGRAPHY**

[1] 1Password: Simple, Convenient Security. https://1password. com/.  [2] Anonymous hackers claim to leak 28,000 PayPal passwords on global protest day.  http://goo.gl/oPv2h.

[3] Blizzard servers hacked; emails, hashed passwords stolen. http: //goo.gl/OTNWJC.  [4] Current Android Malware . http://goo.gl/0sWbXz.

[5] Dashlane Password Manager. https://www.dashlane.com/.

[6] Fine the source of your leaks. https://www.leakedsource.com/. [7] Hackers compromised nearly 5M Gmail passwords. http://goo. gl/IRu07u.  [8] LinkedIn Confirms Account Passwords Hacked. http://goo.gl/ AWB5KC.  [9] Nanohttpd java app. https://nanohttpd.com.

[10] One Year Of Android Malware. http://goo.gl/2UkUJS.

[11] Password Manager - Remember, delete, change and import saved passwords in  Firefox. https://goo.gl/Qve4l7.

[12] Password Manager, Auto Form Filler, Random Password Generator & Secure  Digital Wallet App. https://lastpass.com/.

[13] RoboForm: World’s Best Password Manager. https://www. roboform.com/. [14] RSA breach leaks data for hacking securid tokens. http://goo.gl/ tcEoS.  [15] RSA SecurID software token cloning: a new how-to. http://goo. gl/qkSFY.  [16] Russian Hackers Amass Over a Billion Internet Passwords. http://goo.gl/aXzqj8.  [17] A. Adams and M. A. Sasse. Users are not the enemy. Commun. ACM, 42(12),  1999.

[18] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure  against dictionary attacks. In Advances in Cryptology – Eurocrypt, 2000. [19] D. J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange. Elligator: elliptic-curve  points indistinguishable from uniform random strings. 2013.

55

[20] J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70  million passwords. In Security and Privacy (SP), 2012 IEEE Symposium on. IEEE,  2012.

[21] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano. The quest to replace  passwords: A framework for comparative evaluation of web authentication schemes. In  Security and Privacy, 2012.

[22] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart. Crackingresistant password  vaults using natural language encoders. In 2015 IEEE Symposium on Security and  Privacy, pages 481–498. IEEE, 2015.

[23] S. Chiasson, P. C. van Oorschot, and R. Biddle. A usability study and critique of  two password managers. In Usenix Security, 2006.

[24] I. Dacosta, M. Ahamad, and P. Traynor. Trust no one else: Detecting MITM  attacks against SSL/TLS without third-parties. In European Symposium on Research in  Computer Security, 2012.

[25] W. Ford and B. S. Kaliski Jr. Server-assisted generation of a strong secret from a  password. In Enabling Technologies: Infrastructure for Collaborative Enterprises,  2000.

[26] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and  oblivious pseudorandom functions. In Theory of Cryptography. 2005.  [27] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The  most dangerous code in the world: Validating ssl certificates in non-browser software.  In Proceedings of the ACM Conference on Computer and Communications Security,  2012.

[28] M. Gollam, B. Beuscher, and M. Durmuth. On the security of cracking-resistant  password vaults. In ACM Conference on Computer and Communications Security,  2016.

56

[29] J. A. Halderman, B. Waters, and E. W. Felten. A convenient method for securely  managing passwords. In Proceedings of the 14th international conference on World  Wide Web. ACM, 2005.

[30] S. Jarecki, A. Kiayias, and H. Krawczyk. Round-optimal password-protected  secret sharing and T-PAKE in the passwordonly model. In Advances in Cryptology– ASIACRYPT. 2014.

[31] S. Jarecki, H. Krawczyk, M. Shirvanian, and N. Saxena. DeviceEnhanced  Password Protocols with Optimal Online-Offline Protection.  http://eprint.iacr.org/2015/1099. ASIACCS 2016.

[32] A. Karole, N. Saxena, and N. Christin. A Comparative Usability Evaluation of  Traditional Password Managers. In International Conference on Information Security  and Cryptology, 2010.

[33] A. Karole, N. Saxena, and N. Christin. A comparative usability evaluation of  traditional password managers. In Information Security and Cryptology-ICISC. 2011.  [34] M. R. Karthiga and M. K. Aravindhan. Enhancing performance of user  authentication protocol with resist to password reuse attacks. International Journal Of  Computational Engineering Research, 2(8), 2012. [35] Z. Li, W. He, D. Akhawe, and  D. Song. The emperor’s new password manager: Security analysis of web-based  password managers. In USENIX Security Symposium, 2014.

[36] M. Mannan and P. C. van Oorschot. Using a personal device to strengthen  password authentication from an untrusted computer. In Financial Cryptography and  Data Security. Springer, 2007.

[37] D. McCarney, D. Barrera, J. Clark, S. Chiasson, and P. C. van Oorschot. Tapas:  design, implementation, and usability evaluation of a password manager. In  Proceedings of the 28th Annual Computer Security Applications Conference. ACM,  2012.

[38] R. Morris and K. Thompson. Password security: a case history. Commun. ACM,  22(11), 1979.

[39] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password  authentication using browser extensions. In USENIX Security Symposium, 2005. [40] M. Shirvanian, S. Jarecki, H. Krawczyk, and N. Saxena. Sphinx: A password store  that perfectly hides passwords from itself. In IEEE International Conference on  Distributed Computing Systems, 2017.

[41] F. Stajano. Pico: No more passwords! In Security Protocols XIX, pages 49–81.  Springer, 2011.

[42] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying wolf:  An empirical study of ssl warning effectiveness. In USENIX Security Symposium,  2009.

[43] L. Whitney. LastPass CEO reveals details on security breach.  https://goo.gl/WyNjjb, 2011.

[44] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and  security: Empirical results. IEEE Security and Privacy, 2(5), 2004. [45] K.-P. Yee and K. Sitaker. Passpet: convenient password management and  phishing protection. In Symposium on Usable privacy and security, 2006.  [46][Characterization and evolution. In Security and Privacy (SP), 2012.