Exploring Password Authenticated Key Exchange (PAKE) Protocols: A Usability Evaluation

Thomason Zhao
UW–Madison

Thomas Peng
UW–Madison

ABSTRACT

Password-authenticated key exchange (PAKE) protocols offer a more secure alternative to traditional password-based authentication by establishing a secure session key between a client and server without directly exposing the user's password. This project evaluated the usability of three PAKE protocols - Password Over TLS, Secure Remote Password (SRP), and OPAQUE - by assessing their performance, scalability, and user experience. Through experiments measuring server-side resource utilization and a user study with 136 participants, we found that while PAKE protocols provide stronger security guarantees, they can incur higher computational overhead on the server compared to basic password-over-TLS authentication. However, participants generally felt more secure and preferred the PAKE-based authentication, despite the increased response times. These insights contribute to understanding the practical considerations for deploying PAKE protocols in real-world applications, informing decision-makers on balancing security, efficiency, and user-friendliness.

1 INTRODUCTION

The widespread reliance on password-based authentication has long been a contentious topic in the cybersecurity landscape. Traditional password-based authentication systems are susceptible to various attacks, such as password guessing, dictionary attacks, and credential stuffing, which can compromise user accounts and expose sensitive information. To address these vulnerabilities, researchers have developed more robust authentication methods, with password-authenticated key exchange (PAKE) protocols emerging as a promising solution.

PAKE protocols are designed to establish a secure session key between a client and a server without revealing the user's password, even in the face of an adversary with extensive knowledge of the system. These protocols leverage cryptographic techniques to ensure that after a login attempt, whether valid or invalid, the client and server only learn whether the password matched the expected value, without leaking any additional information. This property is particularly valuable in threat models where adversaries may have access to password leaks, password distributions, and substantial computational power.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The primary goal of this project was to evaluate the usability of different PAKE protocols, assessing their performance, scalability, and user experience. By conducting a comprehensive analysis, we aimed to provide insights into the practical implications of adopting PAKE protocols in real-world authentication systems, addressing the tradeoffs between security, efficiency, and user-friendliness.

To achieve this objective, we reimplemented two PAKE protocols - Secure Remote Password (SRP), and OPAQUE - within a web-based application framework utilizing React.js for the front-end, Node.js for the back-end, and MongoDB for the database. We also implemented a baseline password-over-TLS authentication mechanism as a point of comparison. We then performed a series of experiments to measure the protocols' performance and scalability, focusing on metrics such as CPU usage, memory consumption, and storage requirements. Additionally, we conducted a user experience study to gather feedback on the usability, trustworthiness, and perceived safety of the PAKE protocols from a diverse group of participants.

The findings from this project contribute to the understanding of the practical considerations surrounding the deployment of PAKE protocols in real-world settings. By evaluating the tradeoffs between the security benefits and the system-level impact of these protocols, we hope to inform decision-makers and system designers on the feasibility and trade-offs of incorporating PAKE-based authentication into their applications, ultimately enhancing the overall security posture while maintaining a positive user experience.

2 BACKGROUND AND RELATED WORK

Passwords remain the most common form of user authentication on the internet today. However, traditional password-based authentication systems have long been a point of concern due to their inherent vulnerabilities. Passwords can be easily guessed, stolen, or compromised through a variety of attacks, exposing user accounts and sensitive information to unauthorized access. This has prompted the research and development of more robust authentication methods that can better protect user credentials while maintaining a positive user experience.

2.1 Password-Authenticated Key Exchange (PAKE) Protocols

Password-authenticated key exchange (PAKE) protocols are a class of cryptographic techniques designed to establish a secure session key between a client and a server without explicitly revealing the user's password. These protocols leverage the user's password as the primary authentication factor, while ensuring that the password itself is not exposed during the authentication process. PAKE protocols are particularly useful in scenarios where traditional

password-based authentication mechanisms are vulnerable to offline attacks, such as password guessing, dictionary attacks, and credential stuffing.

The core idea behind PAKE protocols is to allow the client and server to mutually authenticate each other and derive a shared session key, without the server ever learning the client's password. This is achieved through a series of cryptographic operations that ensure the password is only used as a key to unlock the authentication process, rather than being directly transmitted or stored on the server. By preserving the confidentiality of the password, PAKE protocols offer a higher level of security compared to traditional password-based authentication mechanisms.

In a typical PAKE protocol, the client provides their username and password to the server, and the server responds with a challenge or some form of cryptographic information. The client and server then engage in a series of message exchanges, performing various computations and verifications to establish the shared session key. At the end of the protocol, both the client and server are confident that the other party has successfully authenticated, without revealing the password to either party.

2.2 Threat Model and Research Goal

The primary threat model considered in this project assumes that all adversaries have access to any exposed information, such as password leaks or password distributions, as well as some computational power, such as the ability to precompute password hashes. This threat model reflects the reality of modern cybersecurity challenges, where attackers often possess significant resources and knowledge about the target system.

Within this threat model, we identified two main adversarial roles:

- **Adversarial Client:** A malicious client that attempts to gain unauthorized access by exploiting any vulnerabilities in the authentication process.
- **Adversarial Server:** A malicious server that tries to learn a user's identity and credentials by manipulating the authentication process or storing sensitive information.

The research goal of this project was to evaluate the usability of different PAKE protocols, assessing their performance, scalability, and user experience. By conducting a comprehensive analysis, we aimed to provide insights into the practical implications of adopting PAKE protocols in real-world authentication systems, addressing the tradeoffs between security, efficiency, and user-friendliness.

2.3 Existing PAKE Protocols

To achieve our research goal, we selected two PAKE protocols to evaluate - Secure Remote Password (SRP) and OPAQUE. These protocols represent different approaches to password-authenticated key exchange and offer varying levels of security and usability. We also implemented a baseline - Password Over TLS - authentication mechanism as a point of comparison to assess the performance and security benefits of PAKE protocols.

- Password Over TLS (baseline): This protocol leverages the Transport Layer Security (TLS) protocol to encrypt the communication between the client and server, ensuring that the password is transmitted securely. The server then checks the user's password by comparing it to the stored value in the database.
- Secure Remote Password (SRP): SRP is a PAKE protocol that allows the client and server to authenticate each other and establish a shared session key without explicitly revealing the password. The protocol involves a series of computations and message exchanges to verify the password without transmitting it in plain text.
- OPAQUE: OPAQUE is a more recent PAKE protocol that utilizes an oblivious pseudorandom function (OPRF) to perform the password-based key exchange. This protocol aims to provide stronger security guarantees and protect against a wider range of attacks, such as offline dictionary attacks and server compromise.

By evaluating these different PAKE protocols, we aimed to gain a comprehensive understanding of the tradeoffs between their usability, performance, and security characteristics, ultimately providing insights to guide the adoption of PAKE-based authentication in real-world applications.

3 EVALUATION PLAN

4 EXPERIMENTAL MEASUREMENTS AND RESULTS

To evaluate the performance and scalability of the PAKE protocols, we conducted a series of experiments on the server-side implementation. The goal was to measure the impact of the different PAKE protocols on the system's resource utilization, including CPU usage and storage requirements, to understand the practical implications of adopting these protocols in real-world applications.

Our experiments simulated a high-pressure user registration and login scenario, with 10 concurrent user threads performing 1,000 rounds of registration and login. The results showed a significant difference in the CPU usage patterns between the PAKE protocols and the basic password-over-TLS authentication.

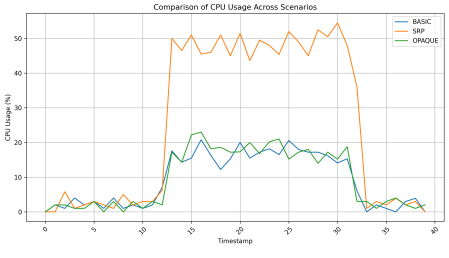

Figure 1: Comparison of server-side CPU usage

4.1 Server-side CPU Usage

One of the key performance metrics we examined was the server-side CPU utilization over time. The PAKE protocols, by nature, involve more complex cryptographic operations compared to a basic password-over-TLS authentication, which could potentially lead to higher CPU consumption.

The Password Over TLS protocol exhibited a relatively stable and low CPU usage throughout the experiment, as the server-side

logic primarily involved hashing the password and comparing it to the stored value in the database.

In contrast, the SRP protocol demonstrated a much higher and more variable CPU usage over time. The server-side computations required for the SRP protocol, such as generating ephemeral keys and deriving the session proof, placed a significantly higher demand on the CPU resources. This could be a potential bottleneck for servers handling a large number of concurrent authentication requests.

The OPAQUE protocol also showed elevated CPU usage compared to Password Over TLS, but it was generally lower and more stable than the SRP protocol. The server-side operations in OPAQUE, such as the oblivious pseudorandom function (OPRF) and the encryption/decryption of the client's private key, appeared to be more efficiently implemented, leading to a more manageable CPU utilization.

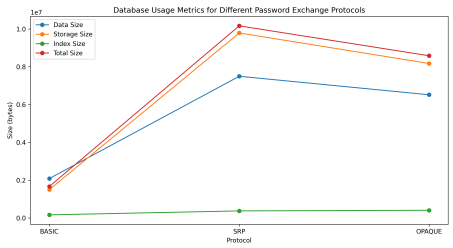

Figure 2: Comparison of server-side storage usage

4.2 Server-side Storage Usage

Another aspect we investigated was the impact of the PAKE protocols on the server-side storage requirements. The storage usage is an important consideration, as PAKE protocols often require storing additional data, such as salts, session information, or encrypted client keys, to facilitate the authentication process.

Our experiments revealed that the PAKE protocols, in general, required more storage space compared to the basic password-over-TLS authentication. The SRP protocol and the OPAQUE protocol had roughly the same storage usage, with the SRP protocol being slightly higher.

For the SRP protocol, the server needed to store the user's username, the generated secret value (V), and the salt (s) for each registered user. This additional data overhead amounted to an increase in storage requirements compared to the baseline password-over-TLS protocol.

The OPAQUE protocol, on the other hand, required storing a more complex data structure, including the encrypted envelope (E) containing the client's private key, the client's public key (CKp), and the salt (s) specific to each user's username. The storage usage for the OPAQUE protocol was slightly lower than the SRP protocol, but still significantly higher than the baseline.

The Password Over TLS protocol, being the simplest in terms of server-side implementation, had the smallest storage footprint, as it only required storing the user's hashed password along with the salt. The storage usage for the Password Over TLS protocol was roughly half of the SRP and OPAQUE protocols.

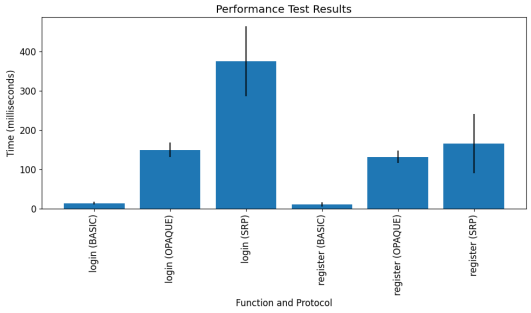

Figure 3: Comparison of client-side per-request response times

4.3 Client-side experience

In addition to the server-side performance metrics, we also examined the client-side experience during the authentication process. This was an important aspect to consider, as the user-perceived responsiveness and efficiency of the authentication flow can significantly impact the overall user experience.

The Password Over TLS (baseline) exhibited the fastest and most consistent response times throughout the simulation. The client-side request processing was relatively straightforward, involving the transmission of the username and password, followed by the server's verification and response. This simplicity translated to a smooth and responsive user experience, with minimal latency.

In contrast, the SRP protocol demonstrated significantly longer and more variable response times compared to Password Over TLS. The client-side implementation of SRP required a more complex series of message exchanges, including the generation of ephemeral keys, the computation of the session proof, and the verification of the server's response. This additional computational overhead and the back-and-forth communication between the client and server resulted in a less responsive user experience, with some requests taking considerably longer to complete.

The OPAQUE protocol also exhibited longer response times compared to Password Over TLS, but the overall performance was better than SRP. The client-side operations in OPAQUE, such as the oblivious pseudorandom function (OPRF) and the encryption/decryption of the client's private key, were more optimized than the SRP protocol, leading to a more consistent and responsive user experience. However, the response times for OPAQUE were still noticeably longer than the baseline Password Over TLS protocol.

5 MILESTONE

The project will follow a series of milestones to ensure structured progress towards the system's development and evaluation.

5.1 Milestone 1: System Design and Initial Development

- Deadline: 2024-03-04
- Goals:
  - Complete a detailed literature review.
  - Finalize the system architecture.
  - Begin development of the authentication prototype.

5.2 Milestone 2: Prototype Testing and Iteration

- Deadline: 2024-04-05
- Goals:
  - Complete the prototype development.
  - Conduct initial internal testing and debugging.
  - Gather early feedback and iterate on the prototype.

5.3 Milestone 3: System Evaluation and Finalization

- Deadline: 2024-04-24
- Goals:
  - Perform comprehensive system testing, including security and performance evaluations.
  - Finalize the user interface based on usability testing feedback.
  - Prepare the project report and documentation.

REFERENCES