PreAcher: Secure and Practical Password Pre-Authentication by Content Delivery Networks

2025/8/18

CONTENTS

01. Background significance of the topic

02. Research method and process

03. Research results and presentation

04. Summary and future improvements

01.

Background significance of the topic

Background and Motivation

01

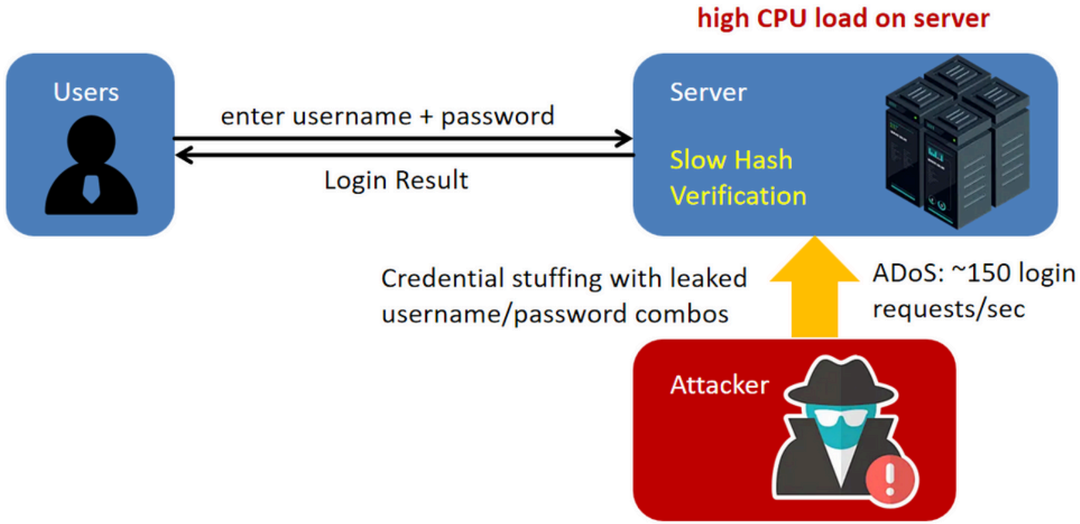
Problem

- Application-layer DoS (ADoS): Attackers can overwhelm servers with only ~150 requests per second, exploiting slow password verification.
- Existing defenses (rate limiting, CAPTCHA, 2FA, CDN bot detection) either harm usability, can be bypassed, or require exposing passwords to CDNs.

02

Current authentication systems

- Password authentication remains the most widely deployed login mechanism.
- Secure sites use slow hashing (e.g., PBKDF2, Argon2) to defend against brute force.
- However, this introduces high computational overhead for each login attempt.



PreAcher : New Research Idea

Core problem : How to let CDNs help filter malicious login attempts without learning user passwords, thereby mitigating ADoS while preserving confidentiality.

Hypothesis

If CDNs could pre-authenticate login attempts using cryptographic transformations of the password, then malicious traffic could be filtered early, preventing server overload, while keeping the password secret.

Conceptual model

- Use an Oblivious Pseudo-Random Function (OPRF) to hide the password input.
- Apply Locality-Sensitive Hashing (LSH) to check password “similarity.”
- CDN forwards only requests that are likely correct to the server for full authentication.

Framework

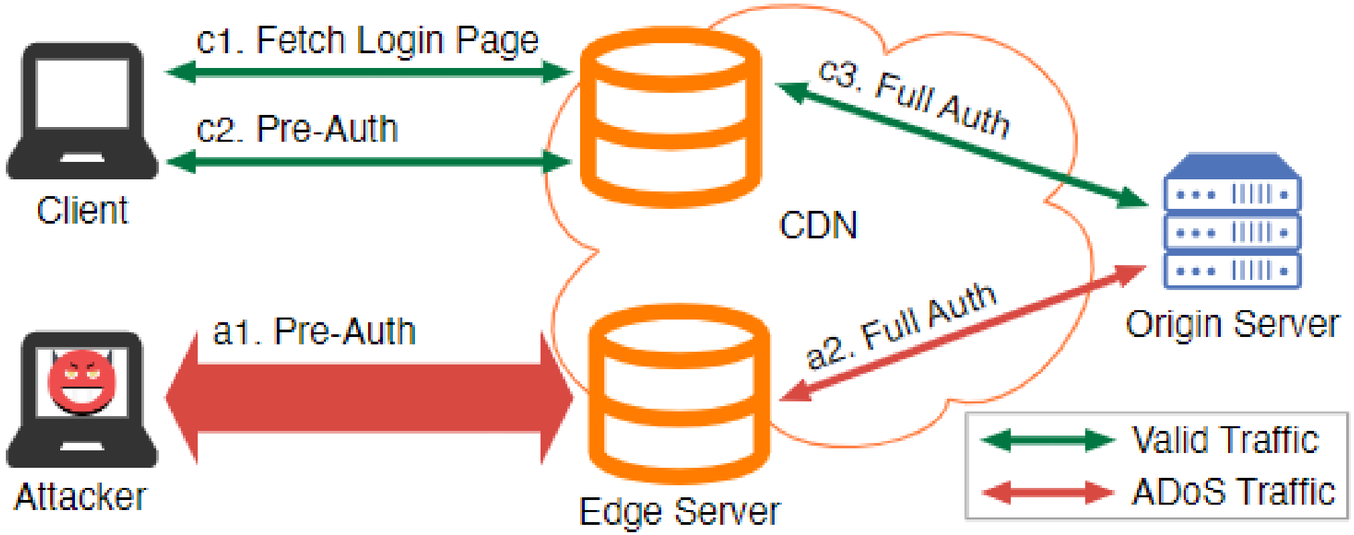
Password → transformed into OPRF output (secure blinding).

CDN checks with LSH → filters out wrong inputs.

Only promising attempts → server for slow hash verification.

Design Goals

Achieving these three goals simultaneously is non-trivial. Author utilize both techniques of **authentication protocols** and the **features of current web development practice** and thus finally design and implement PreAcher shown in this paper.



Security

The system focuses on securing password au-thentication. It should prevent the ADoS attacks exploiting the login interface, while avoiding password exposure to CDNs or other attackers. Besides, the system should retain the CDN's security benefits such as DDoS protection and WAF.

Compatibility

The system should be compatible with the current web ecosystem. Its deployment should not involve multiple stakeholders on the Internet. The system does not require any modification on current browsers, CDN infrastructure, or operating systems. It only requires updating the website's login page and processing.

Efficiency

The system should not introduce much overhead to the login procedure from the perspectives of throughput and latency.

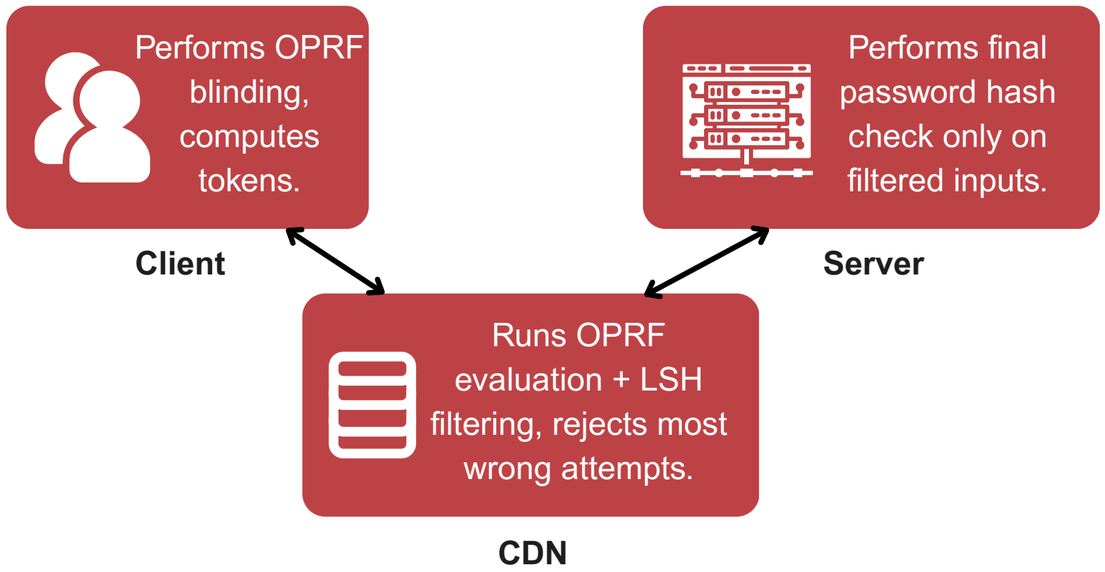
02.

Research method and process

PreAcher Protocol Design

01

Three-party design

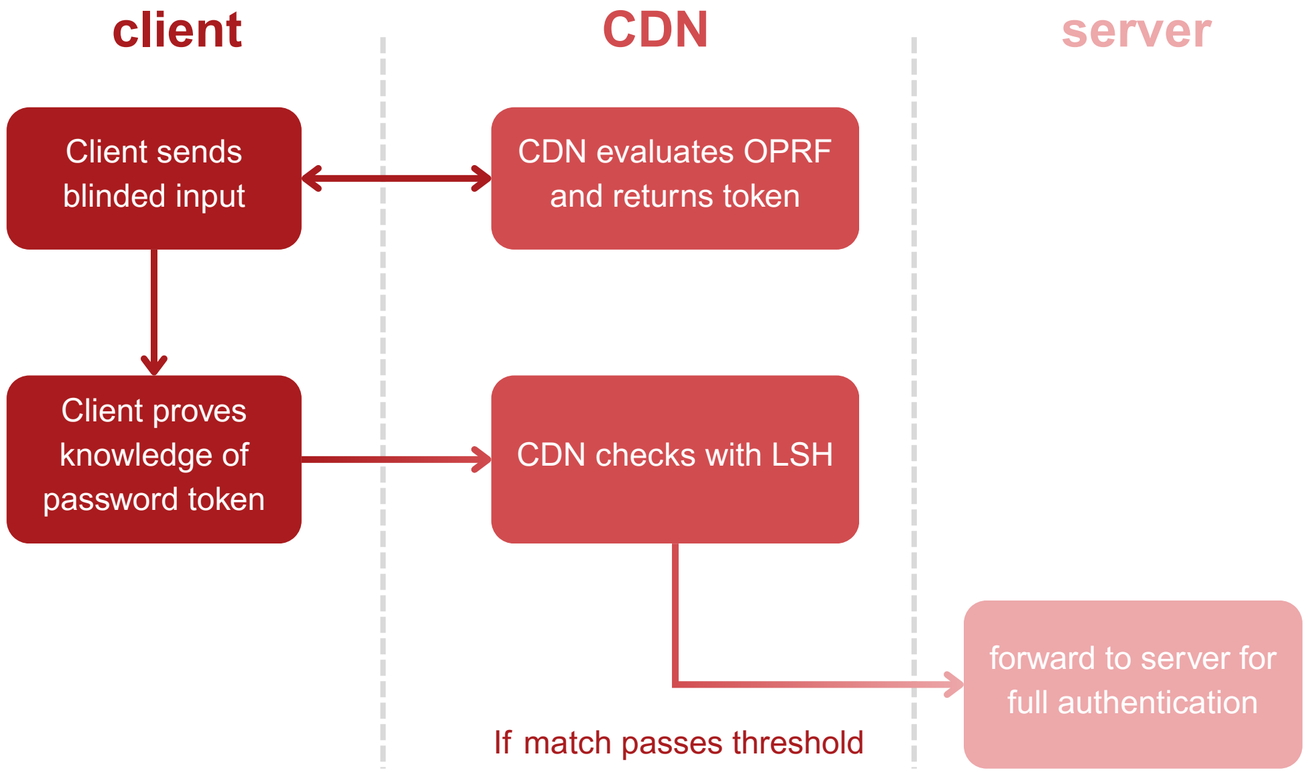


- PreAcher introduces a **three-party architecture** involving the **client**, the **CDN**, and the origin **server**. The client is responsible for **blinding its password** input and later computing the **OPRF output**, so that the CDN never sees the raw password. The CDN plays a middle role: it **evaluates the OPRF** and applies a **Locality-Sensitive Hashing (LSH) filter**, which allows it to **identify and reject most incorrect login attempts** with minimal computation. Finally, the server only performs the full, computationally expensive password **hash verification** if the CDN has already flagged the attempt as highly likely to succeed.
- This division of roles makes the CDN act as a **shield against ADoS traffic**: the heavy cost of slow hashing is paid only when truly necessary. At the same time, **password secrecy is maintained** because the CDN never handles the **plaintext**, and the cryptographic construction ensures that tokens cannot be inverted into the original password.

PreAcher Protocol Design

02

Protocol flow



- The PreAcher login sequence is compact, requiring only **two additional round trips** beyond the traditional TLS session. In the first phase, the client blinds its password-derived input and sends it to the CDN. The CDN evaluates the OPRF and returns the evaluated token to the client. In the second phase, the client unblinds and proves possession of the correct password token. The **CDN then checks** this token against its **stored LSH table**, which captures the distribution of likely password encodings.
- If the **match passes the similarity threshold**, the CDN forwards the request to the server for the final slow hash verification. If the token is unlikely to correspond to the legitimate password, the CDN immediately rejects the request, sparing the server from **unnecessary computation**. This structure makes PreAcher highly efficient: the expensive operation is deferred to the server only for promising attempts, while the CDN performs lightweight checks at scale.

PreAcher Protocol Design

03

Security guarantee

CDN cannot recover plaintext password.

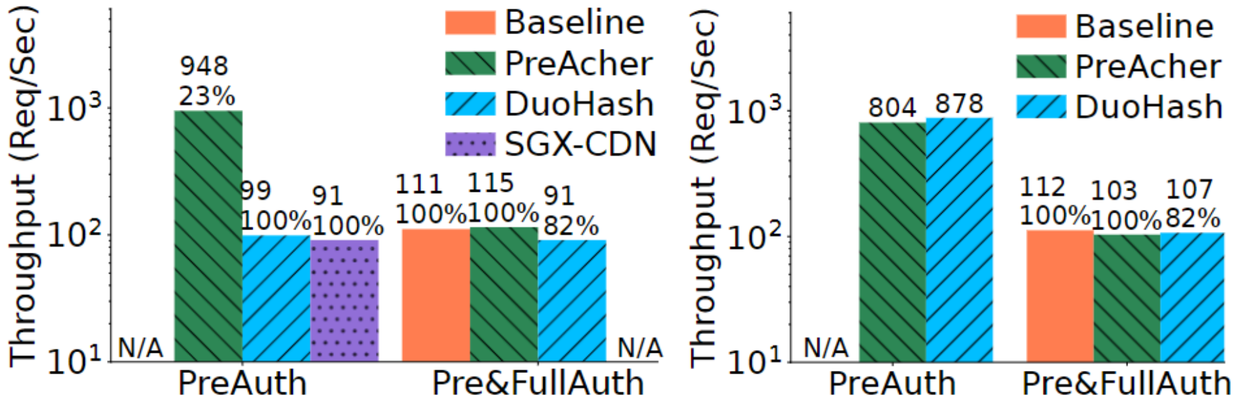
Offline dictionary attacks are statistically bounded.

- The PreAcher protocol is carefully constructed to **provide provable security properties**. First, the CDN is cryptographically prevented from recovering the plaintext password: the OPRF ensures that any password input is blinded, and the CDN only sees **randomized tokens**. Second, the **combination of OPRF and LSH** significantly reduces the feasibility of offline dictionary attacks. **Without LSH**, attackers observing CDN traffic could test many candidate passwords offline, but with LSH constraints and bounded query parameters (K,QK, QK,Q), the probability of a successful attack drops below 1% under recommended settings .
- Moreover, **indistinguishable failure messages** make it unclear to attackers whether rejection occurred at the CDN stage or the server stage, further complicating adversarial probing. Together, these properties give PreAcher strong resilience: it shields servers from ADoS load, maintains password confidentiality, and provides formal security bounds against brute-force attempts.

03.

Research results and presentation

Evaluation: Throughput & CPU Efficiency



PreAcher achieves **an order-of-magnitude higher throughput** compared to DuoHash and SGX-CDN, while consuming far less CPU. On Cloudflare's infrastructure, PreAcher's per-request CPU time is **almost 3× lower than DuoHash**, and SGX-CDN is impractical due to massive overhead.

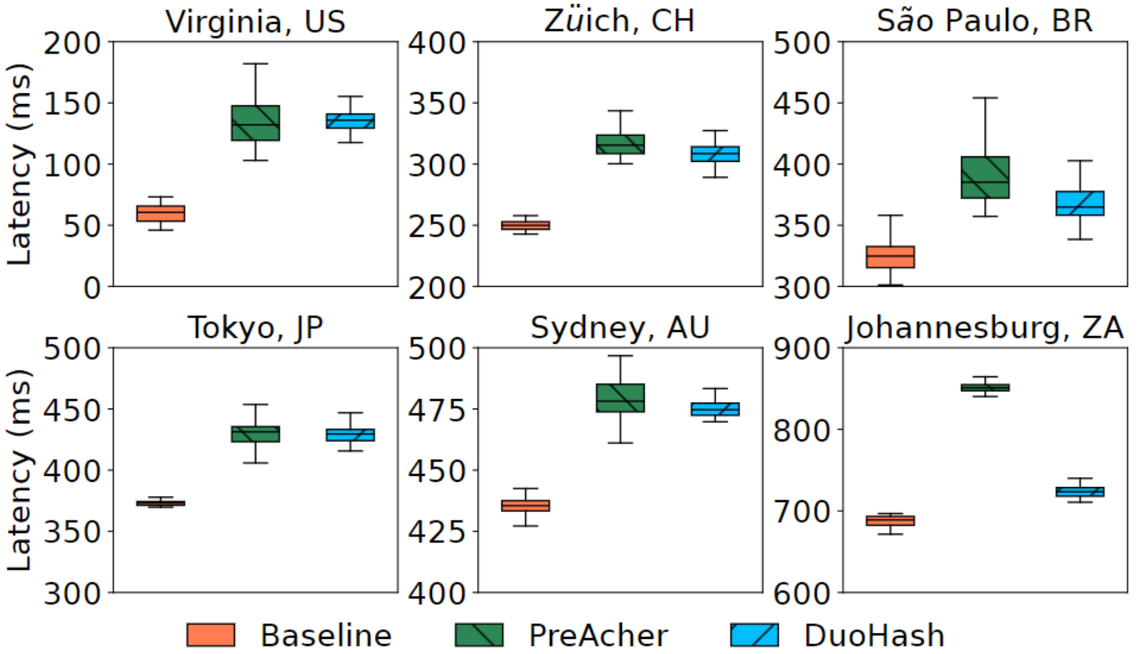
Reason

This efficiency comes from **shifting most computations to lightweight OPRF evaluations and LSH filtering** at the CDN. Unlike DuoHash or SGX-CDN, which require costly cryptographic or enclave operations per request, PreAcher minimizes expensive computations and ensures that the **origin server only processes likely-correct requests**.

Evaluation: Latency

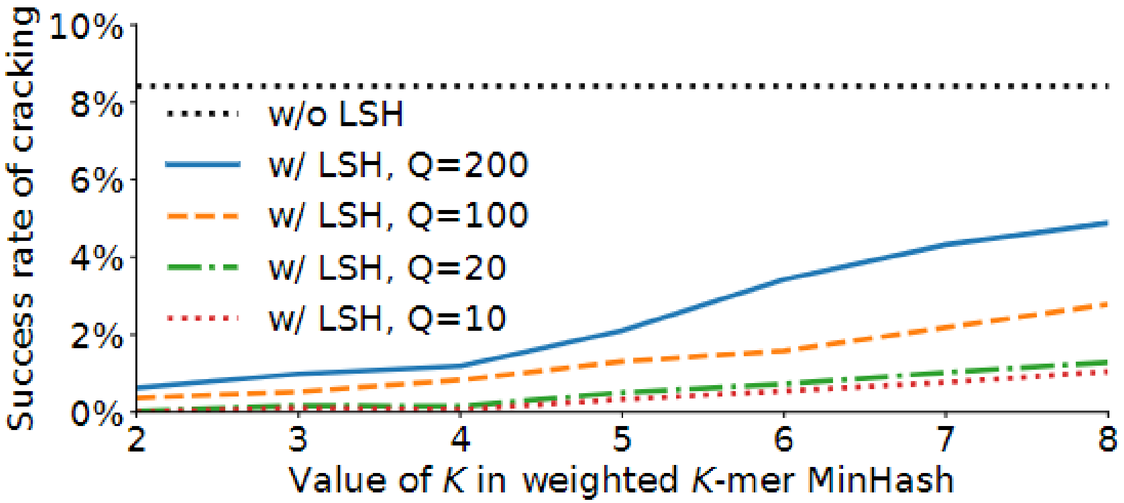
Reason

The two additional RTTs required for pre-authentication are offset by **lightweight operations** at the CDN, meaning that even though communication increases slightly, computation is much faster. This design ensures **real-world deployability**, since most applications can tolerate tens of milliseconds of delay.



PreAcher introduces **only modest latency overhead** compared to the baseline: typically 40–75 ms, with one outlier (Johannesburg) caused by routing issues. The overhead is **comparable to DuoHash**, while maintaining far better throughput and scalability.

Evaluation: Security Against Offline Attacks



PreAcher reduces the **offline dictionary attack success rate** from ~8.42% (OPRF-only) to ~0.20% (with OPRF + LSH, K=4, Q=20). Even with higher query counts, the success probability remains well below 1%. Furthermore, the **collision probability is negligible** (Pcol<10−7P_{col} < 10^{-7}Pcol<10−7), ensuring robustness against guessing.

Reason

This improvement comes from the **combination of OPRF and LSH constraints**, which limits the number of effective offline guesses an attacker can make. By requiring similarity matches rather than exact matches, PreAcher **reduces the information leakage** attackers could exploit, while still allowing the CDN to filter incorrect inputs effectively.

Comparison

Baseline	DuoHash	SGX-CDN	PreAcher
Forwards all requests	Double hashing at CDN	Relies on Intel SGX	Pre-authentication on CDN
exposes passwords, no ADoS defense	secure but very inefficient (~99 req/s throughput)	secure but hardware-dependent (~91 req/s throughput)	
			High throughput (~948 req/s PreAuth, 97 req/s under attack), low CPU use, deployable

04.

Summary and future improvements

Summary

This paper proposes a novel security mechanism called “**PreAcher**”, specifically designed to **counteract ADoS/DDoS attacks** that exploit password-based authentication while **preserving the confidentiality of user passwords**. Compared to previous research, PreAcher’s unique **three-party authentication mechanism**, which allows CDNs to **perform efficient pre-authentication without accessing plaintext passwords**, effectively addresses Application-layer Denial-of-Service (ADoS) attacks that arise from the **high computational cost of password authentication**.

Traditional protection methods like rate limiting, CAPTCHA, and two-factor authentication often **require storing plaintext password** on CDNs or servers, making them **vulnerable to man-in-the-middle attacks or data leaks**. PreAcher integrates CDNs into the authentication process through a three-party protocol involving the client, CDN, and origin server. It uses **Oblivious Pseudorandom Functions** (OPRF) to conceal passwords from the CDN and Locality-Sensitive Hashing (LSH) to enable the CDN to **detect likely incorrect passwords without full knowledge of the correct one**. The CDN conducts “pre-authentication,” filtering most invalid logins before they reach the server, thus mitigating ADoS while preserving password secrecy.

The design is **compatible with current infrastructure**, requiring no client/browser or CDN infrastructure changes—only website-side deployment. Evaluations on testbeds and the Internet (Cloudflare) show that PreAcher maintains **high throughput under attack**, significantly **reduces CPU usage** compared to alternatives (e.g., DuoHash, SGX-CDN), and **adds acceptable latency** (42–72 ms in most regions). It also reduces the risk of offline dictionary attacks from a passive CDN attacker, lowering cracking success rates from 8.42% to 0.20%.

Overall, PreAcher offers a deployable, efficient, and secure solution to enhance password-based authentication resilience against both ADoS attacks and password exposure risks.

Summary and Remaining Limitations

Key Contributions and Findings

- ADoS Mitigation: PreAcher effectively mitigates Application-layer Denial-of-Service attacks at the login interface by introducing CDN-based pre-authentication, which filters out the vast majority of incorrect password attempts before they reach the server.
- Balanced Trade-off: Compared to DuoHash and SGX-CDN, PreAcher achieves a better balance of security, performance, and deployability. It scales to nearly 10× higher throughput while consuming much less CPU, with only modest latency overhead.
- Cryptographic Soundness: By using Oblivious Pseudorandom Functions (OPRF) and Locality-Sensitive Hashing (LSH), PreAcher ensures that the CDN cannot access plaintext passwords, nor can it reliably attempt large-scale offline guessing.

Remaining Limitations

- Scope Restriction: PreAcher is designed only for password-based login endpoints. Other interfaces (e.g., session resumption, password reset, API endpoints) remain vulnerable to ADoS.
- Attacker Model Assumption: Security analysis assumes that the CDN is only a passive adversary (i.e., it can observe traffic but does not actively tamper with messages). In reality, CDNs could be compromised or behave maliciously.
- Password Weakness: PreAcher does not address inherent weaknesses of passwords themselves, such as password reuse across sites or weak, easily guessable choices.

Future Improvements and Research Directions

Toward Broader Authentication Security

- Integration with Passwordless Authentication: PreAcher's pre-authentication concept could be extended to WebAuthn / FIDO2. Here, the CDN could filter malicious traffic before invoking cryptographic hardware verification, further reducing exposure to ADoS.
- Expansion to APIs: The mechanism could be adapted for API key or token verification in API gateways. By filtering invalid tokens early, similar protection could be extended to REST/GraphQL APIs and microservices, which are increasingly targeted by DoS attackers.

Enhancing Security Models

- Use of Trusted Execution Environments (TEEs): Integrating PreAcher with Intel SGX or ARM TrustZone could protect against active CDN compromise, ensuring that even a malicious CDN cannot manipulate or exfiltrate password data.
- Multi-Party Computation (MPC): An alternative is combining PreAcher with MPC protocols to distribute trust, so no single CDN or server fully controls authentication data.

Adaptive Defense Strategies

- Dynamic Security Policies: Adjusting LSH parameters (K, Q) or introducing rate-dependent response delays based on real-time traffic conditions could strengthen resilience under adaptive attacks.
- Machine-Learning-Aided Filtering: CDN-level filters could integrate anomaly detection models, complementing the LSH-based mechanism.