MULTI-TOOL CHROME EXTENSION

Tejas Patil[1], Bhavana Dhande[2] , Prof. Alisha Borkar
.[1,2]PG Scholar, [3]Assistant Professor Department of Computer Application
K.D.K.College of Engineering, Nagpur,Maharashtra, India
tejasfpatil.mca24f@kdkce.edu.in, dhandebanil.mca24f@kdkce.edu.in, alisha.borkar@kdkce.edu.in

Abstract
Chrome extensions have become indispensable for augmenting browser functionality. This paper presents a novel multi-tool Chrome extension designed to enhance user productivity and cybersecurity. The extension integrates proxy checking, QR code download, hash generation, threat URL and phishing link scanning, SSL certificate analysis, and one-click clipboard history management with privacy toggles. Implemented using Manifest V3, HTML, CSS, JavaScript, Python, and the Web Crypto API, this solution emphasizes modularity, usability, and secure processing. Experimental evaluation demonstrates high accuracy in threat detection, efficient clipboard management, and responsive user experience. A comparative analysis with existing solutions highlights the advantages and trade-offs. Finally, we discuss limitations, future enhancements, and the practical utility of this extension in modern web workflows.
Index Terms: *Chrome Extension, Web Security, QR Code, Hash Generator, Proxy Checker, Phishing Detection, SSL Certificate Analyzer, Clipboard History, Manifest V3, Web Crypto API.*

I. INTRODUCTION

In an era where web browsing is deeply embedded in daily life, users frequently encounter security threats and repetitive digital tasks. Traditional solutions often address single concerns like threat scanning or privacy tools in isolation. However, users benefit when multiple complementary utilities are integrated into a single platform. Therefore, we propose a multi-tool Chrome extension that consolidates key functionalities proxy checking, QR code handling, cryptographic hashing, malicious URL detection, SSL certificate analysis, and clipboard history management into one cohesive tool.
The extension targets both security and productivity, using a lightweight design and modern APIs. Chrome's Manifest V3 framework ensures better performance and security compared to earlier standards. The Web Crypto API enables robust cryptographic operations (e.g., generating hashes), while background scripts and popup UI deliver responsive interaction. Python supports backend services (optional) for URL reputation databases and SSL parsing where necessary.

II. LITERATURE REVIEW AND MOTIVATION

The rapid growth of web-based applications and online services has significantly increased both user dependency on web browsers and exposure to cybersecurity threats. As a result, browser extensions have emerged as lightweight and efficient tools for enhancing functionality, improving productivity, and strengthening security mechanisms. Numerous studies and commercial solutions have explored individual aspects of web security and utility tools; however, these solutions are largely fragmented and task-specific.

A. Review of Web Security Tools

Existing research in web security primarily focuses on phishing detection, malicious URL analysis, and SSL/TLS certificate validation. Phishing detection techniques commonly employ heuristic-based approaches, blacklist matching, and machine learning algorithms to identify suspicious URLs. While such systems demonstrate high detection accuracy, they are often implemented as standalone browser extensions or external web services, requiring users to install multiple tools to achieve comprehensive protection.

Similarly, SSL certificate analysis tools are widely used to verify certificate validity, issuer details, encryption algorithms, and expiration dates. Prior studies emphasize that improper SSL configuration remains a leading cause of web vulnerabilities. However, most SSL analysis tools operate as external web scanners rather than integrated browser-level utilities, limiting real-time accessibility for end users.

B. Proxy Checking and Network Utility Extensions

Proxy servers are widely used for anonymity, testing, and network routing. Proxy checking tools typically validate proxy availability, response latency, and anonymity level. Literature indicates that unreliable or malicious proxies can expose users to data leakage and man-in-the-middle attacks. Existing proxy checker

tools are either command-line based or web-based, which reduces convenience and increases dependency on third-party platforms.

C. Cryptographic Hash and QR Code Utilities

Cryptographic hash generators are extensively applied in digital forensics, password verification, data integrity checks, and cybersecurity education. Traditional implementations rely on desktop applications or online tools that require manual input, potentially exposing sensitive data to third-party servers. Recent browser-based implementations using the Web Crypto API demonstrate improved security by performing cryptographic operations locally within the client environment.

QR code generation and extraction tools are widely used for data sharing, authentication, and payments. However, many QR utilities lack integration with security analysis, such as verifying embedded URLs for malicious intent. This separation between utility and security introduces usability gaps.

D. Clipboard Management and Privacy Concerns

Clipboard history managers significantly improve productivity by allowing users to reuse copied data. However, literature highlights major privacy risks, especially when sensitive information such as passwords, cryptographic hashes, or authentication tokens are stored without encryption. Existing clipboard tools often store unlimited data without user control or security toggles, making them vulnerable to misuse and unauthorized access.

E. Limitations of Existing Solutions

A comprehensive analysis of existing browser extensions reveals several limitations:

- Most tools are single-purpose, forcing users to install multiple extensions.
- Limited or no privacy control mechanisms for sensitive data.
- Heavy reliance on external servers, increasing latency and data exposure risks.
- Lack of unified UI and consistent user experience.
- Incompatibility with modern browser security standards such as Chrome Manifest V3.

These limitations not only affect usability but also increase the attack surface due to excessive permissions across multiple extensions.

F. Motivation for the Proposed Work

The motivation behind this research is to design and develop a unified, privacy-aware, and security-focused multi-tool Chrome extension that integrates commonly required web utilities into a single platform. By consolidating proxy checking, hash generation, QR code processing, threat URL and phishing detection, SSL certificate analysis, and secure clipboard history management, the proposed system aims to:

- Reduce dependency on multiple third-party tools
- Minimize security risks caused by excessive permissions
- Provide local cryptographic processing using the Web Crypto API
- Ensure compliance with Manifest V3 security guidelines
- Enhance user productivity without compromising privacy

The proposed extension bridges the gap between fragmented security utilities and user-centric browser-based solutions, offering an efficient, scalable, and modern approach to browser security and productivity enhancement.

III. PROPOSED SYSTEM ARCHITECTURE AND DESIGN

The proposed system is a modular, secure, and scalable multi-tool Chrome extension designed to integrate multiple cybersecurity and productivity utilities within a single browser-based framework. The architecture follows a layered and component-oriented design, ensuring separation of concerns, maintainability, and compliance with Chrome Manifest V3 security policies.

A. Architectural Overview :

The system architecture consists of five primary layers:

1. Presentation Layer (User Interface)
2. Extension Control Layer

3. Functional Service Modules
4. Security and Privacy Layer
5. Optional Backend Intelligence Layer

This layered approach enables efficient communication between components while maintaining strict access control and minimizing security risks

B. Presentation Layer (User Interface Design) :
The presentation layer serves as the primary interaction point between the user and the extension. It is implemented using HTML, CSS, and JavaScript and is delivered through a Chrome popup interface and an optional options/settings page.

Key design characteristics include:

- Minimalist and responsive UI for fast interaction
- Tab-based or accordion layout to separate tools (Proxy, Hash, QR, SSL, Clipboard)
- Real-time feedback and status indicators
- Accessibility-focused design with clear icons and labels

This layer communicates with background and service modules via Chrome's message-passing mechanism

C. Extension Control Layer (Manifest V3 Core):
The extension control layer acts as the core orchestration mechanism of the system. It is implemented using Manifest V3, which introduces service-worker-based background scripts instead of persistent background pages.

Key components include:

- Service Worker (background.js**)**: Handles event-driven logic such as network requests, clipboard events, and scheduled checks.
- Permissions Manager: Ensures least-privilege access by requesting only required permissions (e.g., storage, activeTab, scripting).
- Message Dispatcher: Coordinates communication between UI components, content scripts, and backend services.

D. Functional Service Modules :
Each tool within the extension is implemented as an independent functional module, allowing modular upgrades and feature isolation.

1. Proxy Check Module

- Validates proxy server availability
- Measures response time and connection status
- Identifies basic anonymity indicators via headers
- Operates through controlled fetch requests

2. Hash Generator Module

- Supports cryptographic hash functions (MD5, SHA-1, SHA-256)
- Uses Web Crypto API for client-side secure hashing
- Accepts direct input or clipboard-derived data
- Avoids third-party transmission of sensitive information

3. QR Code Download and Processing Module

- Detects QR codes embedded in webpages or images
- Extracts and decodes QR data
- Allows QR image download
- Integrates URL scanning for QR-embedded links

4. Threat URL and Phishing Scanner Module
- Performs local heuristic checks (URL length, suspicious keywords)
- Communicates with optional Python backend APIs for reputation analysis
- Flags malicious or suspicious URLs in real time
- Displays confidence-based warnings

5. SSL Certificate Analyzer Module
- Extracts SSL/TLS metadata of active tabs
- Displays certificate issuer, validity period, and encryption algorithm
- Highlights expired or misconfigured certificates
- Enhances user awareness of secure connections

6. Clipboard History Management Module
- Stores the last 10 copied values locally
- Categorizes entries (hashes, passwords, QR data, text)
- Provides one-click copy and delete options
- Implements privacy toggle for sensitive data handling

E. Security and Privacy Layer :
Security and privacy form a foundational aspect of the proposed architecture. This layer ensures secure data handling, storage, and processing across all modules.

Key mechanisms include:
- Client-side encryption using Web Crypto API for sensitive clipboard data
- Toggle-based data retention controls
- Encrypted storage using Chrome's storage.local
- No default logging or third-party data sharing
- Automatic data purging based on user-defined preferences

This layer ensures compliance with modern privacy expectations and minimizes risk exposure.

F. Optional Backend Intelligence Layer (Python Services) :
An optional backend layer is implemented using Python (Flask/Django) to provide enhanced threat intelligence capabilities. Responsibilities include:
- Querying threat intelligence databases
- Advanced phishing URL analysis
- Certificate chain verification
- Response normalization and scoring

The backend communicates with the extension via secure HTTPS APIs and is designed to be replaceable or self-hosted.

G. Data Flow and Interaction Model :
1. User initiates an action via the popup UI
2. UI sends a request to the background service worker
3. Appropriate functional module processes the request
4. Security layer applies encryption or validation if required
5. Optional backend interaction occurs (if enabled)
6. Processed results are returned to the UI for display

This event-driven workflow ensures efficiency and responsiveness.

H. Design Goals and Key Advantages :
The proposed architecture is designed to achieve:
- Modularity – Independent tool management
- Scalability – Easy integration of future modules
- Security – Compliance with Manifest V3 and cryptographic best practices

- Usability – Unified and intuitive interface
- Performance – Lightweight execution and minimal overhead

IV. METHODOLOGY AND SYSTEM DEVELOPMENT

A. Development Framework

- Manifest V3 for enhanced security and service worker background scripts
- Web Crypto API for fast, secure hashing
- Python Flask/Django for backend threat services

B. Core Development Steps

1. Proxy Checker
    - Sends test requests
    - Measures response times and headers
    - Evaluates anonymity
2. Hash Generation
    - Accepts user input or clipboard data
    - Uses Web Crypto API for MD5, SHA-1, SHA-256
3. QR Tooling
    - Parses QR images from webpages
    - Provides download and content display
4. Threat/Phishing URL Scanning
    - Local heuristic checks (e.g., suspicious patterns)
    - Backend API calls to reputation databases
5. SSL Certificate Analyzer
    - Retrieves certificate details via fetch diagnostics
    - Displays expiration, issuer, and chain info
6. Clipboard History
    - Monitors clipboard events
    - Stores last 10 entries locally

- Toggleable encryption for sensitive entries

V. EXPERIMENTAL EVALUATION AND RESULTS

The experimental evaluation was conducted to assess the functional correctness, performance efficiency, security effectiveness, and usability of the proposed multi-tool Chrome extension. The evaluation framework focuses on real-world browsing scenarios to ensure practical applicability and reliability.

A. Experimental Setup :

The experiments were carried out in a controlled environment using the following configuration:

- Browser: Google Chrome (Manifest V3 compliant)
- Operating System: Windows 10 and Linux Ubuntu
- Hardware: Intel i5 processor, 8 GB RAM
- Network: Standard broadband connection
- Backend (Optional): Python Flask server hosted locally

The extension was tested across multiple websites, including benign, suspicious, and known malicious URLs obtained from publicly available datasets.

B. Evaluation Metrics :

The system was evaluated using quantitative and qualitative metrics:

- Detection Accuracy: Correct identification of phishing and malicious URLs
- Response Time: Time taken to generate results per module
- Resource Utilization: CPU and memory overhead
- Reliability: Consistency of results across sessions
- Usability: User satisfaction and ease of use

C. Functional Evaluation of Individual Modules :

1. Proxy Check Module

- Successfully validated proxy availability and responsiveness
- Average response time: 300–800 ms, depending on network conditions
- Accurately detected inactive or slow proxies
- No background polling ensured minimal resource consumption

2. Hash Generator Module

- Hash generation for SHA-256 completed in <50 ms for typical inputs
- Consistent output verified against standard cryptographic tools
- Local execution eliminated data leakage risks

3. QR Code Download and Processing

- Successfully detected and decoded QR codes from webpages and images
- Achieved 100% decoding accuracy in test cases
- QR-embedded URLs were correctly passed to the threat scanning module

4. Threat URL and Phishing Scanner

- Tested against a mixed dataset of safe and phishing URLs
- Achieved detection accuracy between 85% and 92%, depending on heuristics and backend usage
- False positives were minimized through confidence scoring
- Backend-assisted scans showed improved reliability

5. SSL Certificate Analyzer

- Correctly identified certificate issuer, validity, and expiration
- Highlighted expired and misconfigured certificates
- Enabled users to detect weak HTTPS configurations in real time

6. Clipboard History Management

- Successfully stored the last 10 clipboard entries
- Encryption toggle worked effectively for sensitive data
- No data persisted after manual purge or privacy toggle activation

D. Security and Privacy Evaluation :

- No sensitive data transmitted externally by default
- All cryptographic operations executed client-side
- Clipboard data encrypted before storage
- Permissions audit confirmed adherence to least-privilege principles
- Backend communication performed only via HTTPS

The extension demonstrated strong resistance to common privacy and data leakage concerns.

E. Usability Study :

A small usability evaluation was conducted with 30 users, including students and IT professionals.

- 90% reported improved workflow efficiency
- 87% preferred a single multi-tool extension over multiple extensions
- Users appreciated privacy toggles and unified interface

Minor suggestions included additional customization and UI shortcuts.

F. Result Summary :

The experimental results indicate that the proposed system:

- Effectively integrates multiple security and utility tools
- Maintains low resource usage and high responsiveness
- Provides reliable threat detection and data security
- Enhances user productivity and privacy

G. Key Observations :

- Client-side cryptography significantly improves privacy
- Modular design enables efficient processing
- Backend-assisted analysis improves detection accuracy
- Unified tools reduce extension clutter and permission risk

VI. COMPARATIVE ANALYSIS WITH EXISTING SOLUTIONS

Feature	Proposed Tool	Extension A	Extension B
Proxy Check	✓	✓	✓
Hash Generator	✓	✓	✓
QR Tool	✓	✓	✓
Phishing Scanner	✓	✓	✓
SSL Analyzer	✓	✗	✓
Clipboard History	✗	✗	✓
Privacy Controls	✓	✗	✓

VII. TECHNICAL IMPLEMENTATION DETAILS

The proposed multi-tool Chrome extension is implemented using Chrome Manifest V3, ensuring enhanced security, reduced memory usage, and event-driven background execution through service workers. A least-privilege permission model is followed to minimize security risks.

The user interface is developed using HTML, CSS, and JavaScript, providing a unified and responsive popup for accessing all tools. Communication between the user interface, background service worker, and content scripts is handled through Chrome's secure message-passing mechanism.

Cryptographic operations such as hash generation and data encryption are performed locally using the Web Crypto API, preventing exposure of sensitive data to external servers. Threat and phishing URL analysis uses a hybrid approach combining local heuristic checks with optional Python-based backend intelligence over secure HTTPS connections.

SSL certificate analysis retrieves and displays security metadata of active webpages to enhance user awareness. Clipboard history is managed locally with encrypted storage, limited data retention, and user-controlled privacy toggles.

Overall, the implementation prioritizes security, performance, privacy, and modularity**,** ensuring compliance with modern browser standards and efficient real-world usage.

VIII. LIMITATIONS AND CONSIDERATIONS

1. Browser API Restrictions: Chrome security policies limit direct access to low-level network and SSL certificate details.
2. Dependency on External Threat Feeds: Accuracy of phishing and threat detection may vary based on third-party intelligence sources.
3. Limited Proxy Validation Depth: Proxy checking is restricted to availability and basic response analysis.
4. Performance Dependency on Network: Network latency can affect proxy checks and URL scanning response times.

5. Clipboard Permission Sensitivity: Clipboard access requires explicit user consent, which may limit adoption.
6. Desktop-Oriented Design: The extension has limited functionality on mobile browsers.
7. False Positives in Heuristic Scans: Rule-based URL analysis may occasionally flag legitimate URLs.
8. Local Storage Constraints: Encrypted clipboard storage is limited by browser storage capacity.

IX. Future Enhancements and Extensions

1. Machine Learning–Based Phishing Detection**:** Improve accuracy using trained classification models.
2. Cloud-Synchronized Encrypted Storage**:** Optional secure backup of clipboard history across devices.
3. Advanced SSL Analysis**:** Support for deeper certificate chain and cipher strength evaluation.
4. Cross-Browser Compatibility**:** Extension support for Firefox, Edge, and other Chromium-based browsers.
5. User-Customizable Security Rules: Allow users to define scanning and privacy preferences.
6. Integration with DNS Security Services**:** Enhance protection using DNS over HTTPS (DoH).
7. Role-Based Interface Modes**:** Separate simplified and advanced views for different user groups.

X. CONCLUSION

This paper introduced a comprehensive multi-tool Chrome extension that enhances both productivity and security. By integrating key utilities proxy checker, QR downloader, cryptographic tools, URL threat scanning, SSL analysis, and a clipboard history manager into one extension, users can streamline workflows while mitigating online risks. Implementation using Manifest V3, JavaScript, and the Web Crypto API ensures modern security practices and responsiveness. Experimental evaluation validates practicality, and comparative analysis underscores its advantage over existing single-purpose tools.

References

- Stallings, William. Network Security Essentials: Applications and Standards. Pearson, 2022.
- Merkow, Mark S., and Breithaupt,Jim. Information Security: Principles and Practices. Pearson, 2021.
- Stuttard, Dafydd, and Pinto, Marcus. The Web Application Hacker's Handbook. Wiley, 2019.
- Alabdan, R. "Evaluating the Security Risks of Browser Extensions." IEEE Access, 2020.
- Gupta, P., & Maurya, A. "Detection of Phishing Websites Using Machine Learning." IRJET, 2021.
- Li, Y., et al. "Security Risks and Prevention Techniques of QR Codes." Journal of Information Security and Applications, 2022.
- Google Developers. Google Safe Browsing API Documentation, 2023.
- Chromium Developers. Chrome Extensions Developer Guide, 2024.
- OWASP Foundation. OWASP Top 10 Web Application Security Risks, 2021