Pattern-Based Authentication with AES Encryption: A Visual Password Management Scheme

Nika Pkhaladze, Besarioni Managadze, Giorgi Lasareishvili

July 28, 2025

Abstract

We present a novel password management scheme that combines visual pattern-based authentication with symmetric encryption. Our approach maps user-generated visual patterns to cryptographic keys, which are then used to encrypt randomly generated passwords using AES. This method provides both security through strong encryption and usability through intuitive pattern-based access.

1 Introduction

Traditional password management relies on memorizing complex strings or storing them in plaintext. Our scheme introduces a pattern-to-key mapping function that transforms visual user patterns into cryptographic keys, enabling secure password storage while maintaining user convenience.

2 Cryptographic Framework

2.1 Notation and Definitions

Let $\mathcal{P}$ denote the space of all possible user patterns, and $\{0,1\}^*$ represent the set of all finite binary strings. We define the following character sets:

$$\Sigma_{upper} = \{A, B, C, \ldots, Z\} \tag{1}$$
$$\Sigma_{lower} = \{a, b, c, \ldots, z\} \tag{2}$$
$$\Sigma_{digit} = \{0, 1, 2, \ldots, 9\} \tag{3}$$
$$\Sigma_{special} = \{!, @, \#, \$, \%, \wedge, \&, *, (,), \ldots\} \tag{4}$$

2.2 Core Functions

2.2.1 Pattern Mapping Function

The pattern mapping function $M : \mathcal{P} \to \{0,1\}^{256}$ transforms a user pattern into a 256-bit key:

$$M(pattern) = \text{SHA-256}(\text{serialize}(pattern)) \tag{5}$$

where serialize : $\mathcal{P} \to \{0,1\}^*$ converts the pattern into a canonical string representation.

Algorithm 1 Password Generation Function $G()$

$s_1 \leftarrow \texttt{Random}(\Sigma_{upper}, 2)$
$s_2 \leftarrow \texttt{Random}(\Sigma_{lower}, 2)$
$s_3 \leftarrow \texttt{Random}(\Sigma_{digit}, 2)$
$s_4 \leftarrow \texttt{Random}(\Sigma_{special}, 2)$
$s_5 \leftarrow \texttt{RandomBytes}(32)$
$password \leftarrow s_1 \| s_2 \| s_3 \| s_4 \| \text{hex}(s_5)$
return $password$

2.2.2 Password Generation Function

The password generation function $G : \emptyset \to \{0,1\}^{320}$ produces a random 40-byte password:

where $\texttt{Random}(\Sigma, n)$ selects n random characters from alphabet Σ, and $\|$ denotes string concatenation.

3 Encryption Scheme

3.1 Key Generation and Encryption

For a given user pattern $p \in \mathcal{P}$ and user metadata $(uid, username, url)$, the encryption process is:

$$k = M(p) \tag{6}$$
$$m = G() \tag{7}$$
$$c = \text{AES-GCM}_k(m, iv) \tag{8}$$
$$record = (uid, username, url, c, \text{timestamp}) \tag{9}$$

where iv is a randomly generated 96-bit initialization vector, and the encrypted record is stored in the database.

3.2 Decryption Process

Given a user pattern p' and user metadata $(uid', username', url')$, password retrieval follows:

Algorithm 2 Password Retrieval

$k' \leftarrow M(p')$
$record \leftarrow \texttt{Retrieve}(uid', username', url')$
if $record = \emptyset$ **then**
  return ERROR
end if
$c \leftarrow record.ciphertext$
$m' \leftarrow \text{AES-GCM}_{k'}^{-1}(c)$
return m'

4 Security Analysis

4.1 Security Properties

4.1.1 Password Security

The generated passwords have entropy $H(G)$:

$$H(G) = H(s_1) + H(s_2) + H(s_3) + H(s_4) + H(s_5) \tag{10}$$
$$= 2\log_2(26) + 2\log_2(26) + 2\log_2(10) + 2\log_2(32) + 256 \tag{11}$$
$$\approx 9.41 + 9.41 + 6.64 + 10 + 256 = 291.46 \text{ bits} \tag{12}$$

4.1.2 Storage Security

Assuming AES-GCM provides semantic security, the stored ciphertexts reveal no information about the underlying passwords without knowledge of the correct pattern.

4.2 Security Analysis and Possible Attacks

This section analyzes potential attack vectors against the IntuiPass system and evaluates the security properties of visual password schemes.

4.2.1 Extension-Level Attacks

To compromise a user's visual password within the extension environment, an attacker must satisfy two conditions: (1) gain unauthorized access to the user's Google Account to authenticate with the extension, and (2) either observe the user's visual pattern or systematically enumerate all possible pattern combinations for the specific visual modality (e.g., chess piece arrangements, piano melodies, mathematical formulas).

The attack complexity varies significantly by visual password type:

- **Chess board configurations:** 17^{64} possible arrangements for 16 pieces placed on a 64-cell board.
- **Piano melodies:** Exponential in sequence length and number of available notes.
- **Mathematical expressions:** Dependent on formula complexity and variable ranges.
- **Pixel Art:** 2^{100} number of states for a 10x10 matrix
- **Connect The Dots:** While it's a bit complex to count the exact number of combinations, it's estimated roughly around 2^{72}

4.2.2 Server-Side and Database Attacks

In the event of database compromise, an attacker obtains encrypted visual password representations. However, successful decryption requires brute-forcing the encryption key, which is the 256 bits indistinguishable from random.

5 Implementation Considerations

5.1 Pattern Serialization

Different pattern types require specific serialization methods:

- **Connect-the-dots**: Sequence of numbers of dots
- **Piano sequences**: Ordered list of note identifiers $n_1, \ldots, n_m$
- **Chess patterns**: Sorted set of piece-position pairs $(piece_i, pos_i)$
- **Pixel art**: Binary matrix representation
- **Mathematical formulas**: LaTeX string normalization

5.2 Error Handling

The system implements graceful degradation:

- Invalid patterns result in authentication failure
- Decryption failures run without a notice for the user, it results in authentication failure

6 Conclusion

Our pattern-based encryption scheme provides a balance between security and usability by leveraging visual patterns as cryptographic keys. The combination of strong random password generation and AES encryption ensures security, while pattern-based authentication maintains user convenience.