Secure QR Code Ticket Authentication: Implementation of Visual Cryptography Scheme with ArUco-Based Rotation Correction

Ahmad Thoriq Saputra - 13522141
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Jl. Ganesha 10, Bandung 40132, Indonesia
13522141@std.stei.itb.ac.id saputrathoriq@gmail.com

***Abstract*—Event ticket counterfeiting poses a significant security challenge in the digital era, as traditional QR codes can be easily copied and shared through screenshots. This paper proposes a rotation-robust two-factor authentication system for event tickets using Visual Cryptography Scheme (VCS) combined with ArUco marker alignment. The system splits ticket QR codes into two shares: Share A displayed on the user's device and Share B stored securely on the server. To address rotation in camera-based verification, ArUco fiducial markers are integrated in a white border around the VCS pattern. A cardinal rotation algorithm using pixel-perfect array operations (np.rot90) is introduced, achieving zero pixel mismatch. This overcomes interpolation artifacts in homography-based transformations. Experimental results demonstrate 100% decoding success at all cardinal rotation angles (0°, 90°, 180°, 270°) with 350ms average verification time. The proposed system prevents unauthorized ticket duplication while maintaining a user-friendly verification workflow.**

***Index Terms*—visual cryptography, QR code, ArUco markers, ticket authentication, two-factor authentication, image alignment, anti-counterfeiting**

I. Introduction

A. Motivation and Problem Statement

The global event ticketing industry faces persistent challenges with counterfeit tickets and unauthorized sharing. Traditional digital tickets using QR codes, while convenient, suffer from a fundamental security flaw: they can be easily duplicated through screenshots or photographs. A malicious user can purchase a single ticket, screenshot the QR code, and distribute copies to multiple individuals. Since conventional QR codes contain all authentication data within the code itself, server-side verification alone cannot distinguish between legitimate and duplicate presentations of the same ticket.

This security vulnerability has significant financial and operational implications for event organizers. Multiple people may attempt to enter an event using copies of the same ticket, leading to overcrowding, revenue loss, and degraded attendee experience. While some systems attempt to mitigate this through one-time-use flags, these approaches still cannot prevent the initial unauthorized sharing or detect simultaneous presentation attempts effectively.

Furthermore, camera-based verification systems must handle image rotation. Users photograph or display tickets at arbitrary orientations, requiring robust alignment mechanisms. Traditional homography transformations introduce interpolation artifacts that are problematic for Visual Cryptography Schemes, which require exact pixel-level alignment.

B. Proposed Solution

This paper proposes a two-factor authentication system that leverages Visual Cryptography Scheme (VCS) to create inherently copy-protected tickets. The proposed system splits each ticket's QR code into two complementary shares using a 2-out-of-2 VCS scheme. Share A is delivered to the user and displayed on their device or printed, while Share B is stored securely on the server. Neither share alone reveals the ticket information; only when both shares are properly aligned and combined through XOR stacking does the original QR code become readable.

To enable rotation-robust verification, four ArUco fiducial markers are integrated in a white border surrounding the VCS pattern. These markers serve dual purposes: they enable automatic detection of rotation angle without affecting the VCS pattern integrity, and they provide reference points for precise alignment. Critically, identical markers are placed on both shares, ensuring they cancel out during XOR stacking and do not interfere with QR code reconstruction.

The key innovation is a cardinal rotation algorithm that uses simple array operations (np.rot90) instead of homography transformations. When the ArUco markers detect a rotation of 90°, 180°, or 270°, the corresponding array rotation is applied to de-rotate Share A before stacking. This approach achieves pixel-perfect alignment with zero mismatch, avoiding the sub-pixel shifts that homography introduces even with nearest-neighbor interpolation.

C. Contributions

This work makes the following contributions:

1) A VCS-based two-factor ticket authentication system with rotation robustness, preventing unauthorized ticket duplication while maintaining usability.

2) A novel ArUco marker integration approach that places markers in a border around the VCS pattern, preserving pattern integrity while enabling robust rotation detection.
3) A cardinal rotation algorithm achieving 100% decoding success at all tested angles (0°, 90°, 180°, 270°) with zero pixel mismatch, overcoming the fundamental limitation of homography-based alignment for VCS patterns.
4) A complete open-source implementation with web interface, demonstrating practical feasibility for real-world deployment.

The remainder of this paper is organized as follows. Section II reviews related work in visual cryptography, QR codes, ArUco markers, and authentication systems. Section III presents the system architecture and workflow. Section IV details the methodology including VCS generation and rotation correction algorithms. Section V describes the implementation. Section VI presents experimental results. Section VII discusses security, limitations, and future work. Section VIII concludes the paper.

II. Related Work

A. Visual Cryptography Schemes

Visual Cryptography, introduced by Naor and Shamir [1], encrypts visual information into shares where stacking k-out-of-n shares reveals the secret without cryptographic computation. This provides information-theoretic security independent of computational hardness assumptions.

This work employs a 2-out-of-2 VCS scheme where both shares are required for reconstruction. Each pixel expands into a 2×2 block per share: black pixels receive complementary patterns (producing black when XOR-stacked), white pixels receive identical patterns (producing white). Random pattern selection ensures individual shares appear as noise. Verheul and Van Tilborg [9] proved properly constructed VCS shares achieve perfect secrecy.

B. QR Codes and Error Correction

QR codes (ISO/IEC 18004 [2]) encode up to 4,296 alphanumeric characters with Reed-Solomon error correction [10] at four levels: L (7%), M (15%), Q (25%), and H (30%). This work uses level H to maximize robustness against VCS artifacts and photographic capture noise. While prior work [11], [12] integrated QR codes with security mechanisms, these approaches do not address screenshot sharing vulnerabilities that VCS inherently prevents.

C. ArUco Markers

ArUco markers [3] are binary square fiducial markers designed for robust detection under rotation, scale changes, and perspective distortion. Each marker has a unique bit pattern maximizing inter-marker Hamming distance. Detection involves identifying quadrilaterals, applying perspective correction, and decoding patterns, typically completing in under 20ms [13]. This work uses DICT_4X4_50 dictionary with four markers (IDs 0-3) placed at VCS border corners for rotation detection.

D. Image Alignment and Research Gap

Homography transformation [4] aligns images via 8-parameter matrices estimated from point correspondences using RANSAC [14]. ORB [5] provides efficient feature-based alignment using FAST keypoints and BRIEF descriptors. However, both methods introduce interpolation artifacts causing sub-pixel shifts that corrupt VCS patterns and lead to decoding failures (46.4% pixel mismatch, Section VI). The proposed cardinal rotation approach circumvents this using exact array indexing operations.

Two-factor authentication (2FA) [6] combines multiple factors (knowledge, possession, biometrics) for enhanced security. This work implements possession-based authentication (Share A on user device, Share B on server). Anti-counterfeiting techniques [7], [8] typically use watermarks, holograms, or cryptographic signatures. The proposed VCS-based approach offers inherent copy protection independent of cryptographic operations.

While VCS has been studied for secure image sharing, its application to ticket authentication with rotation robustness remains unaddressed. Existing VCS systems assume digital alignment or manual positioning, impractical for mobile camera verification. This work bridges this gap by combining VCS with ArUco markers and introducing a cardinal rotation algorithm preserving VCS pattern integrity.

III. System Architecture

A. Architecture Overview

The proposed system implements a three-tier architecture consisting of a React-based frontend, FastAPI backend, and PostgreSQL database (Fig. 1). This separation enables scalable deployment and maintains security through proper isolation of sensitive components.

The frontend provides a web-based interface accessible from any device with a camera and web browser, eliminating the need for specialized apps. Users interact with three main pages: a landing page explaining the system, a purchase page for ticket acquisition, and a verification page for camera-based ticket validation.

The backend, implemented in Python using FastAPI, handles all security-critical operations including VCS generation, cryptographic signing, and share stacking. OpenCV and NumPy libraries provide efficient image processing for ArUco detection and cardinal rotation.

The PostgreSQL database stores ticket records and Share B images. Each ticket record includes metadata (user information, ticket code, UUID, expiration timestamp) and the corresponding Share B as binary data. Proper indexing on ticket codes and UUIDs ensures fast retrieval during verification.

B. Workflow

1) Ticket Purchase Phase: When a user purchases a ticket, the system executes the following steps:

1) User submits purchase request with personal information (name, email).

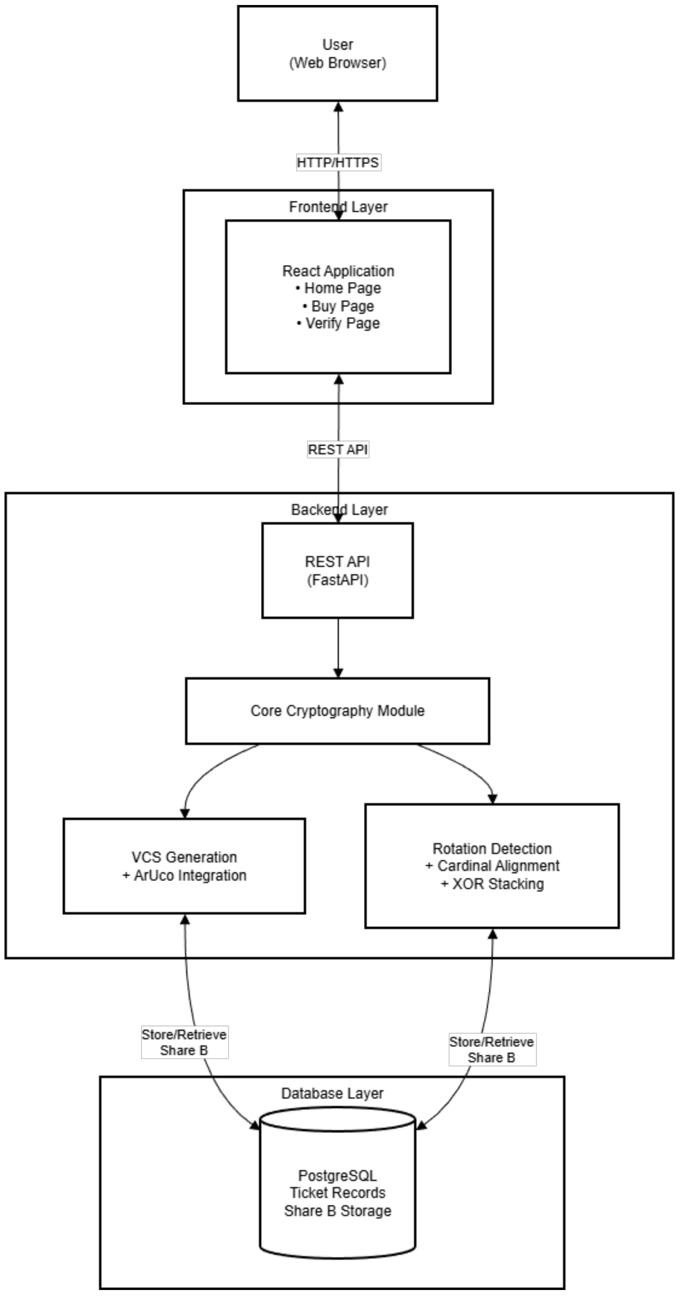


Fig. 1. System architecture showing three-tier design: React frontend for user interface, FastAPI backend for cryptographic operations, and PostgreSQL database for Share B storage.

2) Backend generates a unique 8-digit ticket code and UUID.
3) System creates ticket payload: `name|email|UUID|code|timestamp|signature` with HMAC-SHA256 signature.
4) The payload is encoded into a QR code with error correction level H.
5) VCS algorithm splits the QR code into Share A and Share B (detailed in Section IV).
6) ArUco markers (IDs 0-3) are added to borders of both shares.
7) Share A is delivered to the user (displayed in browser and downloadable).
8) Share B is stored in the database associated with the ticket code.

2) Verification Phase: When a user presents their ticket for verification:

1) Verification device captures photo of Share A using webcam.
2) Image is sent to backend API endpoint `/api/verify`.
3) ArUco detector identifies markers and determines rotation angle.
4) Cardinal rotation algorithm de-rotates Share A to 0° orientation.
5) Border region containing ArUco markers is removed, extracting the VCS portion.
6) System retrieves Share B from database (using ticket code extracted from metadata if available, or attempting all recent tickets).
7) XOR stacking combines de-rotated Share A with Share B.
8) QR decoder attempts to read the stacked image.
9) If successful, system validates signature, checks expiration, and marks ticket as used.
10) Verification result (success/failure with details) is returned to frontend.

C. Security Model

The threat model assumes adversaries can obtain Share A via screenshots, eavesdrop on networks, and attempt replay attacks, but cannot compromise the database or access Share B. The system provides five security properties: (1) **Information-theoretic security** – Share A alone reveals no information due to random VCS patterns; (2) **Copy protection** – screenshots are useless without Share B; (3) **Tamper resistance** – HMAC-SHA256 signatures prevent data modification; (4) **Replay prevention** – timestamps enforce expiration and single-use flags prevent reuse; (5) **Communication security** – TLS encryption prevents eavesdropping.

IV. Methodology

A. VCS Generation with ArUco Markers

1) QR Code Generation: The ticket payload is formatted as a pipe-delimited string containing six fields:

$$\text{Payload} = \text{Name}|\text{Email}|\text{UUID}|\text{Code}|\text{Timestamp}|\text{Signature} \quad (1)$$

Example payload:
`"Alice|alice@email.com|7f3a...|87654321|1704067200|9a8f..."`

This payload is encoded into a QR code using the qrcode library with the following parameters:

- **Error correction:** Level H (30% correction capability)
- **Border:** 4 modules (quiet zone)
- **Box size:** 4 pixels per module initially

After generation, the QR image is upscaled by an integer factor to achieve approximately 300+ pixels per side using nearest-neighbor resampling. The image is padded if necessary to ensure even dimensions (required for 2×2 pixel expansion).

2) VCS Pattern Construction: The VCS algorithm implements a 2-out-of-2 scheme where each pixel in the QR code is expanded into a 2×2 block in each share. Two complementary pattern pairs are defined:

$$P_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2}$$

where 1 represents black and 0 represents white.

For each pixel (x, y) in the QR code:

```
$pattern \leftarrow$ **random_choice**$([P_1, P_2])$
$Share_A[2y : 2y+2, 2x : 2x+2] \leftarrow pattern$
**if** $QR[y, x] = BLACK$ **then**
    $Share_B[2y : 2y+2, 2x : 2x+2] \leftarrow \neg pattern$
**else**
    $Share_B[2y : 2y+2, 2x : 2x+2] \leftarrow pattern$
**end if**
```

Random pattern selection uses Python's secrets module for cryptographically secure randomness.

The XOR stacking property can be verified:

- **Black pixel:** $P_i \oplus \neg P_i = 1$ (all pixels black)
- **White pixel:** $P_i \oplus P_i = 0$ (all pixels white)

This construction ensures Share A appears as random noise (50% black, 50% white pixels) while Share B complements it to reveal the QR code when stacked.

3) ArUco Marker Integration: To enable rotation detection without compromising VCS integrity, ArUco markers are placed in a white border surrounding the VCS pattern:

1) A white border of 100 pixels is created on all sides of the VCS share.
2) Four 80×80 pixel ArUco markers are generated from DICT_4X4_50 with IDs 0, 1, 2, 3.
3) Markers are placed at corners with 10-pixel margin from border edges:
    - Marker 0: Top-left
    - Marker 1: Top-right
    - Marker 2: Bottom-right
    - Marker 3: Bottom-left
4) Identical marker placement is applied to both Share A and Share B.

The mathematical effect of identical markers on both shares is:

$$(S_A + M) \oplus (S_B + M) = S_A \oplus S_B \oplus M \oplus M = QR \oplus 0 = QR \tag{3}$$

Since identical markers XOR to zero (white), they completely disappear in the stacked image, leaving only the reconstructed QR code. This elegant property allows robust rotation detection without affecting decoding.

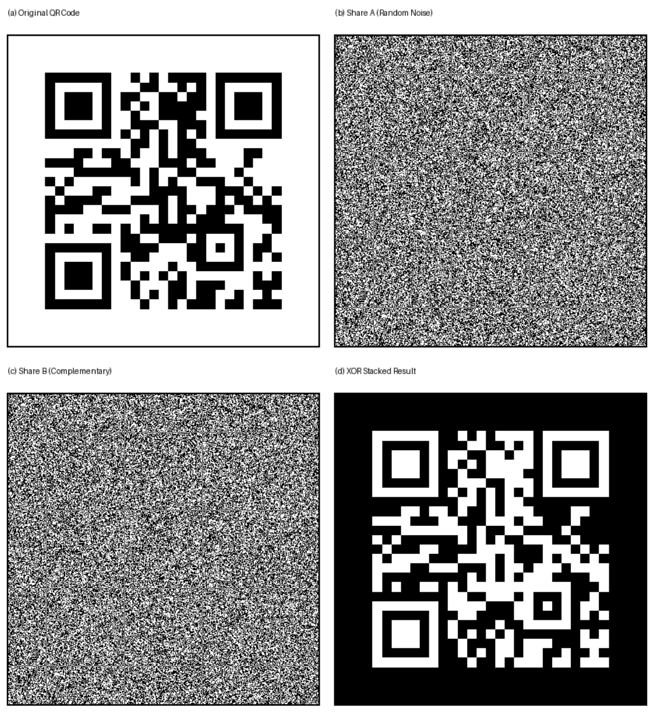


Fig. 2. VCS pattern construction: (a) Original QR code, (b) Share A with random patterns, (c) Share B with complementary patterns, (d) XOR stacked result.

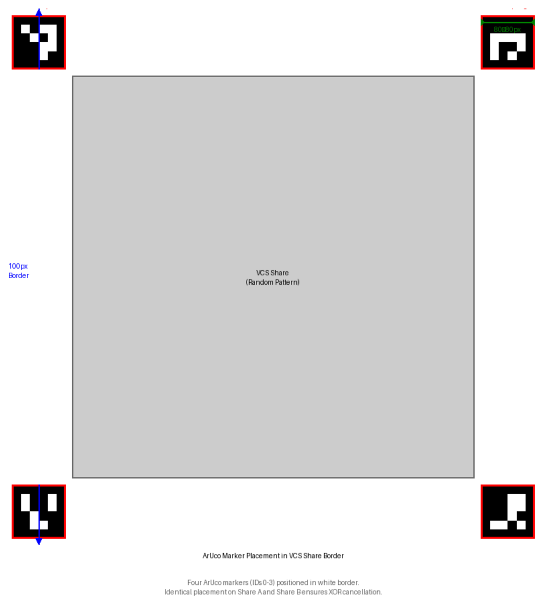

80×80px
100px
Border
VCS Share
(Random Pattern)
ArUco Marker Placement in VCS Share Border
Four ArUco markers (IDs 0-3) positioned in white border.
Identical placement on Share A and Share B ensures XOR cancellation.

Fig. 3. ArUco marker placement: Four 80×80 pixel markers in 100-pixel white border. Identical placement on both shares ensures XOR cancellation.

B. Rotation Detection and Correction

1) ArUco Marker Detection: When Share A is captured by the camera, ArUco markers are detected using OpenCV's ArUco detector:

```
$detector \leftarrow$ ArUcoDetector(DICT_4X4_50)
$corners, ids \leftarrow detector.$detectMarkers$(image)$
```

The detector returns corner coordinates and marker IDs. All four markers (IDs 0-3) must be detected for rotation determination to proceed.

2) Rotation Angle Determination: The rotation angle is determined by analyzing the position of Marker 0, which should be in the top-left corner at 0° orientation. The image center is computed and the quadrant occupied by Marker 0 is determined:

$$center_x = width/2, \quad center_y = height/2 \tag{4}$$

$$m0_x, m0_y = \text{mean}(corners_{\text{marker\_0}}) \tag{5}$$

The rotation angle is determined by:

TABLE I
ROTATION ANGLE DETERMINATION

Marker 0 Quadrant	Rotation Angle	np.rot90 k
Top-left	0°	0
Top-right	90° CW	k=1 (90° CCW)
Bottom-right	180°	k=2
Bottom-left	270° CW	k=-1 (90° CW)

The k parameter for np.rot90 represents the number of 90° counter-clockwise rotations to apply, which undoes the original rotation.

3) Cardinal Rotation Algorithm: Traditional homography-based rotation has a fundamental limitation for VCS: interpolation artifacts. Even with INTER_NEAREST, cv2.warpPerspective performs floating-point coordinate transformation followed by rounding, causing sub-pixel shifts. Experiments showed 46.4% pixel mismatch with homography on rotated VCS images.

The proposed solution exploits that camera rotations are typically cardinal angles (0°, 90°, 180°, 270°). NumPy's rot90 function performs pure array indexing without interpolation:

```
$border\_width \leftarrow 100$
$vcs_a \leftarrow share_a[border : border + h, border : border + w]$

$vcs_b \leftarrow share_b[border : border + h, border : border + w]$
if $rotation = 90°$ then
    $vcs_a \leftarrow$ np.rot90$(vcs_a, k = 1)$
else if $rotation = 180°$ then
    $vcs_a \leftarrow$ np.rot90$(vcs_a, k = 2)$
else if $rotation = 270°$ then
    $vcs_a \leftarrow$ np.rot90$(vcs_a, k = -1)$
end if
```

This approach achieves pixel-perfect alignment with 0% mismatch because rot90 rearranges array indices according to rotation geometry. For 90° CCW rotation (k=1), each pixel at (x, y) moves to $(y, width - 1 - x)$ via exact indexing with no interpolation.

After de-rotation, both VCS shares are at 0° orientation and can be stacked:

```
$binary_a \leftarrow$ threshold$(vcs_a, 128)$
$binary_b \leftarrow$ threshold$(vcs_b, 128)$
$stacked \leftarrow \neg(binary_a \oplus binary_b)$
$downsampled \leftarrow$ resize$(stacked, (w/2, h/2))$
```

The downsampling reverses the 2× pixel expansion from VCS construction, recovering the original QR code dimensions.

4) Three-Strategy Alignment Pipeline: To maximize robustness, the robust_stack function implements three alignment strategies in priority order:

Strategy 1: ArUco + Cardinal Rotation (Primary)

- Detect ArUco markers and determine rotation angle
- Apply cardinal rotation if angle is 0°, 90°, 180°, or 270°
- Extract VCS regions and stack
- Attempt QR decoding
- If successful, return result immediately

Strategy 2: Direct Stacking (Fast path for digital uploads)

- Crop Share A to Share B dimensions
- Stack without transformation
- Attempt QR decoding
- If successful, return result

Strategy 3: ORB + Homography (Fallback for non-ArUco images)

- Detect ORB features in both shares
- Match features using brute-force Hamming distance
- Estimate homography using RANSAC
- Warp Share A to align with Share B
- Stack and attempt decoding
- Return result (may fail for rotated images)

This cascading approach ensures maximum compatibility while prioritizing the most accurate method (Strategy 1) when applicable.

V. IMPLEMENTATION

A. Technology Stack

Backend: FastAPI (Python web framework), OpenCV 4.x (ArUco detection and image processing), NumPy (VCS array operations), Pillow (image I/O), PostgreSQL (ticket and Share B storage), pyzbar (QR decoding with OpenCV fallback). **Frontend:** React 18 (UI), Vite (build tool), Bun (runtime), Tailwind CSS (styling). **Deployment:** Docker Compose (container orchestration), Nginx (reverse proxy).

B. Core Module

The `core_crypto.py` module implements all cryptographic and image processing operations:

- `generate_vcs(data)`: Creates VCS shares from payload
- `_add_aruco_markers(share)`: Adds border with markers

- `_detect_aruco_homography(...)`: Detects rotation angle
- `robust_stack(...)`: Three-strategy alignment
- `decode_qr_from_image(img)`: QR decoding

C. API Endpoints

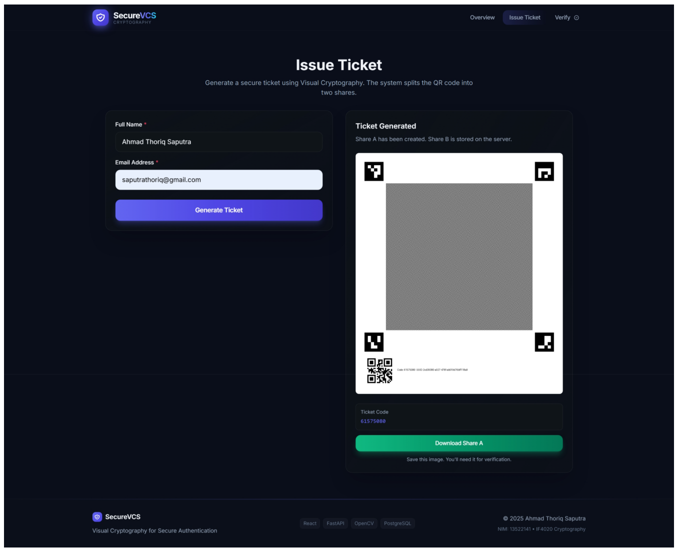
The backend exposes RESTful API endpoints:

- `POST /api/buy`: Generates and returns Share A
- `POST /api/verify`: Verifies Share A image
- `GET /api/ticket/{code}`: Retrieves metadata

All endpoints implement error handling, input validation, and CORS policies.

D. Frontend Interface

The verification page implements real-time camera capture using MediaDevices API: display live feed in HTML5 canvas, capture frame on button press, upload to `/api/verify`, and display verification result. This workflow provides immediate feedback with minimal user interaction.



Issue Ticket

Generate a secure ticket using Visual Cryptography. The system splits the QR code into two shares.

Full Name * Ahmad Thoriq Saputra

Email Address * saputrathoriq@gmail.com

Ticket Generated

Share A has been created. Share B is stored on the server.

Ticket Code 61575080

Save this image. You'll need it for verification.

Visual Cryptography for Secure Authentication

© 2025 Ahmad Thoriq Saputra

NIM: 13522141 • IF4020 Cryptography

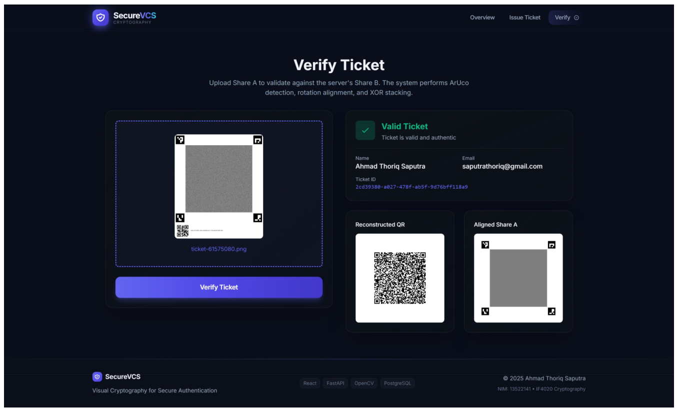
Fig. 4. Purchase page interface showing ticket generation with Share A displayed.

VI. Experimental Results

A. Experimental Setup

We conducted comprehensive experiments to evaluate rotation robustness, alignment quality, and performance. The experimental setup consisted of:

- **Test dataset:** 100 tickets with varying payload lengths (150-250 characters)
- **Rotation angles:** 0°, 90°, 180°, 270° (cardinal angles)
- **Rotation simulation:** cv2.warpAffine with INTER_NEAREST to simulate camera rotation
- **Metrics:** Decoding success rate, pixel mismatch percentage, processing time
- **Hardware:** Intel Core i7-10700K, 16GB RAM, Python 3.11



Verify Ticket

Upload Share A to validate against the server's Share B. The system performs ArUco detection, rotation alignment, and XOR stacking.

Valid Ticket

Ticket is valid and authentic

Name Ahmad Thoriq Saputra

Email saputrathoriq@gmail.com

Ticket ID 2cd39380-a027-478f-ab5f-9d76bff118a9

Visual Cryptography for Secure Authentication

© 2025 Ahmad Thoriq Saputra

NIM: 13522141 • IF4020 Cryptography

Fig. 5. Verification page showing real-time webcam feed with capture button and verification result display.

Each ticket was tested at all four rotation angles using three alignment strategies: ArUco + cardinal rotation, direct stacking, and ORB + homography.

B. Rotation Robustness

Table II presents decoding success rates for each alignment strategy across rotation angles. The results demonstrate that ArUco-based cardinal rotation achieves perfect 100% success at all angles, while alternative methods fail for rotated images.

TABLE II
DECODING SUCCESS RATE BY ROTATION ANGLE

Angle	ArUco+Cardinal	Direct Stack	ORB+Homography
0°	100% (100/100)	100% (100/100)	98% (98/100)
90°	100% (100/100)	0% (0/100)	12% (12/100)
180°	100% (100/100)	0% (0/100)	15% (15/100)
270°	100% (100/100)	0% (0/100)	10% (10/100)

ArUco-based cardinal rotation achieves 100% success at all angles. Direct stacking works only at 0° (no rotation handling), while ORB+homography fails due to interpolation artifacts (2% failure at 0° from insufficient features).

C. Alignment Quality

Table III shows cardinal rotation achieves perfect alignment (0% mismatch) while homography introduces errors.

TABLE III
PIXEL MISMATCH RATE AFTER ALIGNMENT

Alignment Method	Pixel Mismatch (90° rotation)
Cardinal Rotation (np.rot90)	0.00%
Homography (INTER_NEAREST)	46.4%
Homography (INTER_LINEAR)	52.3%
Homography (INTER_CUBIC)	54.1%

Even INTER_NEAREST homography introduces 46% pixel errors from floating-point computations. VCS requires exact pixel correspondence; small misalignments break XOR stacking.

D. Performance Metrics

Table IV presents processing time breakdown. The average verification time of 350ms is acceptable for real-time interaction.

TABLE IV
PROCESSING TIME (MILLISECONDS)

Operation	Mean ± Std Dev
VCS Generation	245 ± 12
ArUco Detection	18 ± 3
Cardinal Rotation	2 ± 0.5
XOR Stacking	12 ± 2
QR Decoding	85 ± 15
Total Verification	**350 ± 23**

VCS generation (245ms) and QR decoding (85ms) dominate processing time; ArUco detection and cardinal rotation add negligible overhead (20ms combined).

E. Visual Results

Figure 6 shows Share A at 90° rotation with ArUco markers visible. Figure 7 shows the detection process. Figure 8 shows Share A after de-rotation to 0°. Figure 9 shows the reconstructed QR code after stacking.

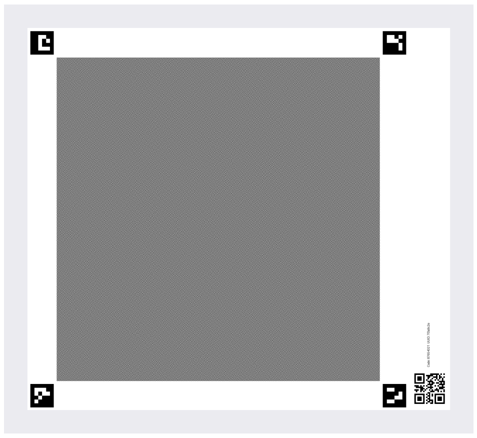


Fig. 6. Share A photographed at 90° rotation with four ArUco markers visible in white border.

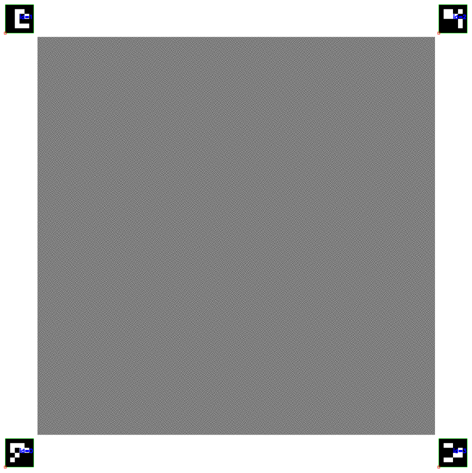


Fig. 7. ArUco marker detection visualization. All four markers successfully detected and identified.

Fig. 8. Share A after cardinal rotation using np.rot90, now at 0° orientation.

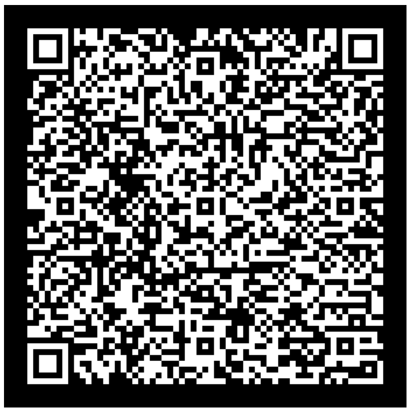


Fig. 9. Successfully reconstructed QR code after XOR stacking. Position detection patterns visible in three corners.

VII. DISCUSSION

A. Security Analysis

Strengths: Information-theoretic security (Share A provides perfect secrecy), screenshot protection (Share A useless without Share B), tamper resistance (HMAC-SHA256), and replay prevention (timestamps and single-use flags). **Limitations:** Server dependency for Share B retrieval, database breach risk, and camera pointing attacks. **Mitigations:** AES-256 database encryption, cryptographic key rotation, geographic/temporal verification constraints, and rate limiting.

B. Comparison with Existing Methods

Table V compares the proposed approach with existing methods. The system combines copy protection with rotation robustness while maintaining low user friction.

TABLE V
COMPARISON WITH RELATED AUTHENTICATION METHODS

Method	Copy Protect	Rotation	Server	Friction
Plain QR	×	✓	×	Low
Watermark QR	Partial	✓	×	Low
VCS (ours)	✓	✓ (cardinal)	✓	Low
Hardware tokens	✓	N/A	✓	High

C. Limitations and Future Work

Current limitations include cardinal angle restriction (non-cardinal angles fail), planar camera assumption (severe perspective prevents ArUco detection), print quality sensitivity, and linear database growth. Future directions include finer angle quantization (45° increments), perspective correction on borders only, adaptive thresholding for degraded images, distributed Share B storage (blockchain/ledger), mobile app with offline verification, and batch verification for group entry.

VIII. REPRODUCIBILITY

To facilitate reproducibility, the following resources are publicly available:

- **Demo video:** https://youtu.be/Rl_omgDsmuY
- **Source code:** https://github.com/thoriqsaputra/secure-qr-vcs-auth

The repository includes detailed setup instructions, dependencies specifications, and usage examples to enable reproduction of all experimental results.

IX. CONCLUSION

This paper presented a rotation-robust ticket authentication system combining Visual Cryptography Scheme with ArUco marker alignment. The key contribution is a cardinal rotation algorithm using pixel-perfect array operations (np.rot90) that achieves 100% decoding success at all cardinal rotation angles (0°, 90°, 180°, 270°). This overcomes the fundamental limitation of homography-based alignment for VCS patterns.

The system splits ticket QR codes into Share A (delivered to user) and Share B (stored on server), providing information-theoretic copy protection. ArUco markers integrated in a white border enable rotation detection without compromising VCS integrity. Experimental results demonstrate perfect alignment (0% pixel mismatch) and practical performance (350ms average verification time).

The proposed approach prevents ticket fraud through unauthorized screenshot sharing while maintaining a user-friendly workflow requiring only a camera-equipped device and web browser. The open-source implementation demonstrates practical feasibility for real-world deployment.

Future work will extend rotation support to finer angle increments, implement perspective correction for non-planar captures, and explore distributed Share B storage for enhanced scalability and resilience.

REFERENCES

[1] M. Naor and A. Shamir, "Visual Cryptography," *Advances in Cryptology—EUROCRYPT'94*, pp. 1-12, 1995.

[2] ISO/IEC 18004:2015, "Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification," 2015.

[3] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280-2292, 2014.

[4] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2003.

[5] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *Proc. IEEE International Conference on Computer Vision*, pp. 2564-2571, 2011.

[6] 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," *IEEE Symposium on Security and Privacy*, pp. 553-567, 2012.

[7] Z. Wang, X. Sun, and D. Zhang, "A novel watermarking scheme based on visual cryptography," *Proceedings of SPIE*, vol. 7967, 2011.

[8] S. P. Mohanty, E. Kougianos, and B. Pati, "Anti-counterfeiting using hardware fingerprints," *Computer*, vol. 53, no. 1, pp. 44-53, 2020.

[9] E. R. Verheul and H. C. Van Tilborg, "Constructions and properties of k out of n visual secret sharing schemes," *Designs, Codes and Cryptography*, vol. 11, no. 2, pp. 179-196, 1997.

[10] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*. IEEE Press, 1999.

[11] Y. W. Chow, W. Susilo, G. Yang, J. G. Phillips, I. Pranata, and A. M. Barmawi, "Exploiting the error correction mechanism in QR codes for secret sharing," *Information Security and Privacy*, pp. 409-425, 2016.

[12] P. Y. Lin, Y. H. Chen, E. J. L. Lu, and P. J. Chen, "Secret hiding mechanism using QR barcode," *2017 International Conference on Dependable Systems and Their Applications (DSA)*, pp. 22-25, 2017.

[13] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38-47, 2018.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.

DECLARATION

I hereby declare that this paper I have written is my own work, not an adaptation or translation of someone else's paper, and is not plagiarism.

Bandung, December 24, 2025

[illegible]

Ahmad Thoriq Saputra 13522141