International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 14 Issue 6 ǁ June 2026 ǁ PP. 67-79

QRify Secure: An Application-Agnostic Framework for Authenticated QR Code Issuance and Verification

Homam ElTaj[1], Reem Bin Homran[2], Maryam Al-Gafri[3], Dana Al-Sousi[4]
[*1,2,3]*Department of Cybersecurity, Dar Al-Hekma University, Jeddah, Saudi Arabia*
Corresponding Authors: htaj@dah.edu.sa , rwbinhomran@dah.edu.sa

Abstract
Quick Response (QR) codes have become a routine bridge between physical artefacts and digital services, yet the symbology offers no native means of confirming that a scanned code was issued by a trusted party. This authenticity gap is now exploited at growing scale through QR-code phishing and the physical substitution of fraudulent codes. Existing defences are fragmented: payload-encryption tools protect confidentiality without authenticating the issuer; reputation scanners react only to destinations already known to be malicious; dynamic-redirection platforms shift trust onto a redirect server; and academic authenticated-issuance systems demonstrate trusted issuance but remain bound to single application domains. This paper presents QRify Secure, a closed-loop framework that consolidates signed-payload issuance, server-side validation, and explicit user-facing consumption control into a single application-agnostic architecture. The framework exposes the choice between one-time-use and time-limited validity as a first-class issuance decision, and is accompanied by a formal threat model that states precisely which attacks it defends against and which it does not. A working web-based prototype implements the framework, and a security analysis maps each in-scope attack, namely substitution, replay, expired-code reuse, and brute-force guessing, to its corresponding defence. The contribution is the consolidation and cross-domain applicability of established trusted-QR components, not a new cryptographic primitive.
Keywords: *QR code security; authenticated issuance; signed-token verification; threat modelling; one-time-use codes; Quishing; closed-loop verification.*

Date of Submission: 17-06-2026 Date of Acceptance: 30-06-2026

I. INTRODUCTION

Quick Response (QR) codes have become a routine bridge between physical interactions and digital services, encoding the URLs that people scan to pay for parking, board flights, open menus, and authenticate to enterprise resources. The symbology is internationally standardised [1] and has become a pervasive mechanism for encoding machine-readable data in the offline world, adopted at scale across retail, hospitality, healthcare, and public administration [2]. The same property that makes the format operationally attractive, namely the ability to encode arbitrary byte sequences densely and to recover from partial occlusion through Reed-Solomon error correction, also produces a structural opacity that conventional URL inspection cannot resolve: a scanner cannot visually verify a code's payload before decoding it [3].

This opacity is exploited at increasing scale. References to QR-code phishing on high-fidelity reporting sources rose by approximately 248 percent between the third and fourth quarters of 2023, and executives now receive QR-code attacks at roughly forty-two times the rate of other employees, consistent with established whaling tactics that target senior personnel [4]. Recent empirical work on real-world QR-based login deployments confirms that the underlying authenticity gap extends well beyond simple URL redirection, with practical attacks demonstrated against systems already in production [5]. The attack pattern has matured to the point where physical overlay of fraudulent codes onto legitimate posters, menus, and parking signage is a documented commodity tactic [6]. Government and sector cybersecurity bodies have responded with dedicated guidance on QR-code phishing, a clear signal that the threat has moved from a marginal nuisance to a recognised organisational risk [7], [8], [9]

1.1. PROBLEM STATEMENT

Existing mitigations against QR-code abuse are fragmented across commercial and academic approaches that each address a different facet of the problem. Payload-encryption tools protect content confidentiality but provide no means to authenticate that a code was issued by a trusted party [10]. URL-reputation scanners detect known-bad destinations at scan time, yet fail against zero-day domains and provide no integrity guarantee for the underlying code. Dynamic-QR redirection platforms prioritise marketing analytics and rewriteable destinations over cryptographic authenticity, leaving the trust assumption resting entirely on the redirect server. Academic proposals have demonstrated server-validated, consumption-controlled QR codes in domain-specific contexts,

including origin authentication via signed payloads [11], RSA-based attendance with timestamp binding to class schedules [12], dynamic-QR payment with national cryptographic algorithms [13], and signed-token login session handoff with single-use nonces and short-lived expiry [14]. Each, however, is scoped to a specific application domain. What remains absent is a generalised, application-agnostic framework that issues authenticated QR codes with explicit user-facing selection of validity mode, supported by a formal threat model articulated for arbitrary deployment contexts.

1.2. CONTRIBUTIONS OF THIS PAPER

This paper presents QRify Secure, a closed-loop framework for the issuance and verification of authenticated QR codes that generalises across application domains rather than targeting a single use case. The contribution is fourfold. First, it consolidates the established components of trusted-QR design, namely signed payloads, server-side validation, and consumption controls, into a single application-agnostic architecture with explicit user-facing selection between one-time-use and time-limited validity at issuance time. Second, it articulates a formal threat model that distinguishes the attacks the system is designed to defend against, including QR substitution within issuance scope, replay, expired-code reuse, and brute-force token guessing, from those it explicitly does not address, including issuance-server compromise and the scanning of any code not issued by the system. Third, it documents a working web-based prototype that implements the framework. Fourth, it provides a security analysis that maps each in-scope attack to the corresponding defence mechanism and honestly states the limitations against out-of-scope threats. Together, these contributions extend the design work previously deposited as an open-access record [15] with the implementation, evaluation, threat-model, and security-analysis components that the deposit deferred to future work.

1.3. STRUCTURE OF THE PAPER

Section 2 reviews QR-code architecture and the cryptographic considerations that arise from the symbology's lack of inherent encryption. Section 3 situates QRify Secure within prior work, distinguishing encryption-centric, reputation-based, dynamic-redirection, and authenticated-issuance approaches. Section 4 presents the threat model. Section 5 develops the system design, and Section 6 describes the implementation. Section 7 reports the evaluation, Section 8 the security analysis, and Section 9 discusses the limitations of the closed-loop scope. Section 10 concludes with prioritised directions for future work.

II. BACKGROUND

2.1. FROM LINEAR BARCODES TO TWO-DIMENSIONAL SYMBOLOGIES

Barcode technology emerged in the mid-twentieth century to automate item identification in manufacturing and retail. Early systems used linear, one-dimensional symbologies such as UPC, EAN, Code 39, and Code 128, which gained wide adoption for their simplicity and low cost but were constrained by limited data capacity [16]. The need to encode more data in a compact area, with error correction sufficient for variable scanning conditions, drove the development of two-dimensional symbologies including Data Matrix, Aztec Code, PDF417, and QR codes. Among the two-dimensional formats, QR codes gained dominance in consumer-facing mobile applications, supported by international standardisation and the absence of licensing fees, which removed both technical and commercial barriers to adoption [1].

2.2. QR CODE ARCHITECTURE AND THE ISO/IEC 18004 STANDARD

QR codes were developed in 1994 at Denso Wave to support faster decoding than one-dimensional barcodes in automotive manufacturing [17]. The format is a square matrix of dark and light modules. Three finder patterns positioned at three corners enable scanners to detect and orient the code; alignment patterns inserted for versions two and above compensate for perspective distortion on curved or angled surfaces; and timing patterns of alternating modules support precise grid alignment during decoding [18]. Reserved areas store format information including the error-correction level and masking pattern, and a quiet zone of four modules surrounds the symbol to enable reliable recognition [1].

Data capacity scales with version: version one is a twenty-one by twenty-one module grid, and each succeeding version adds four modules to each dimension, with the maximum standard symbol encoding up to 7,089 numeric or 2,953 byte characters at the lowest error-correction level [1]. Reed-Solomon error correction is configurable at four levels, enabling recovery from partial damage or occlusion. Crucially, the symbology specification defines how data and error-correction codewords are encoded into the matrix but includes no cryptographic primitive. Confidentiality and integrity controls must therefore be applied at the payload level before encoding and removed after decoding [3]. The ability to encode arbitrary byte sequences makes QR codes well suited as a transport for encrypted tokens, signed payloads, or hybrid structures.

2.3. QR CODE VARIANTS AND OPERATIONAL MODELS

The QR code family comprises several variants tailored to different operational constraints. The standard QR Code (Model 2) spans versions one to forty and supports the highest capacity and the full feature set. Micro QR codes reduce symbology overhead for space-limited applications, omitting features such as alignment patterns in exchange for smaller physical size and lower data capacity. The rectangular Micro QR specification extends the family into rectangular form factors suited to long, narrow labels [19]. Beyond physical form, an operational distinction separates static and dynamic codes: static QR codes embed the complete payload directly within the symbol and operate without network dependency, while dynamic QR codes encode a short redirect URL that resolves through a server, enabling content updates after printing but requiring connectivity at scan time. Static codes are susceptible to physical replacement because their payload cannot be authenticated against an issuer at scan time; dynamic codes face the same physical-replacement exposure and shift the trust assumption to the redirect server rather than eliminating it. Table 1 summarises the principal variants.

Table 1. Principal QR code variants and operational forms.

Variant	Symbol form	Capacity (indicative)	Operational note
Standard QR (Model 2)	Square, 21x21 to 177x177 modules	Up to 7,089 numeric / 2,953 byte	General-purpose; full feature set including alignment patterns
Micro QR (M1 to M4)	Square, 11x11 to 17x17 modules	Up to 35 numeric / 15 byte	Space-constrained labels; reduced overhead; omits some features
Rectangular Micro QR (rMQR)	Rectangular, variable	Variable	Long, narrow labels for industrial marking
Static (operational)	Any standard form	Per symbol	Payload fixed at print time; no server dependency; vulnerable to physical replacement
Dynamic (operational)	Any standard form	Per symbol	Short redirect URL; server-resolved; updatable; requires connectivity

2.4. CRYPTOGRAPHIC CONSIDERATIONS FOR QR PAYLOADS

Because the symbology provides neither encryption nor authentication, cryptographic protection for QR-based applications must be applied to the payload before encoding. Three payload-level patterns dominate the deployable literature, alongside research-stage approaches including watermarking, steganography, and blockchain anchoring [20]. Symmetric encryption, for example AES, protects content confidentiality but requires shared-key management between issuer and verifier [10]. Asymmetric signing, for example RSA or ECDSA, establishes origin authenticity and payload integrity through public-key verification, allowing any verifier to confirm that a code was issued by a holder of the corresponding private key [11], [12]. Hybrid schemes combine a signed envelope with an encrypted body, often packaged as a structured token, to provide both confidentiality and authenticity within the same payload. The choice between these patterns shapes what cryptographic properties the encoded payload can carry, and therefore what additional server-side mechanisms are needed to provide replay protection and server-managed validity state. QRify Secure adopts the signed-payload pattern with server-side validity-state management; the design rationale is developed in Section 5.

III. RELATED WORK

Prior work on QR-code security can be grouped into four approaches, organised by where each places the protective control: on the payload through encryption, on the destination through reputation analysis, on the redirection layer through server-managed URLs, or on the issuance and verification flow through cryptographic authentication. This control-placement taxonomy is consistent with the comprehensive QR-applications survey [3] and the systematic review of QR-code security countermeasures [20]. The academic literature on authenticated issuance is where QRify Secure positions itself.

3.1. ENCRYPTION-CENTRIC QR SOLUTIONS

The first category protects QR payload contents through cryptographic encryption applied before the symbol is generated. Commercial tools such as Qrafter Pro support AES-256 encryption of QR contents with password-based decryption at scan time [21], and academic proposals have extended the approach with chromatic multiplexing and wavelength-based decoding for higher-density encrypted payloads [10]. The shared design intuition is that an attacker who substitutes a fraudulent code cannot recover the protected content without the appropriate key. However, encryption-centric solutions do not authenticate the issuer: an attacker who holds a legitimate symmetric key can generate a valid encrypted code, and a verifier presented with a substitute encrypted code will dutifully decrypt and act on it. Encryption alone protects confidentiality but does not establish that a code was issued by a trusted party, nor does it support replay or consumption controls.

3.2. SCANNER-BASED REPUTATION ANALYSIS

A second category shifts the security check from the symbol to the destination. Tools such as the Trend Micro QR Scanner intercept the decoded URL at scan time and check it against threat-intelligence databases of known malicious domains, displaying a warning or blocking access when a match is found [22]. This approach extends established URL-reputation defences to the QR scanning context and protects against destinations already catalogued as malicious. However, reputation-based detection fails against zero-day domains that have not yet entered threat databases, and it offers no integrity guarantee for the underlying code: an attacker can substitute a code pointing to an as-yet-uncatalogued malicious site, and the reputation check will return clean. Empirical work on malicious QR codes underscores the residual exposure, with controlled studies showing that users readily act on fraudulent codes and that conventional cues do little to interrupt the decision to scan [23]. The defence is therefore necessary but not sufficient against adversaries who control fresh infrastructure.

3.3. DYNAMIC REDIRECTION AND MARKETING PLATFORMS

A third category encodes a short URL within the symbol that resolves through a server-managed redirection layer. Platforms such as QR TIGER and Flowcode use this architecture to enable post-printing edits, scan analytics, and rich destination management, prioritising marketing utility over cryptographic authenticity [24], [25]. The trust assumption shifts from the symbol to the redirection server: if the server is compromised or the redirect URL is hijacked, every dependent code is affected. While some platforms offer scan-count limits or scheduled deactivation, the redirect endpoint cannot cryptographically prove that a code is valid at scan time, leaving the trust model dependent on the redirection server's integrity rather than on the code itself.

3.4. ACADEMIC RESEARCH ON AUTHENTICATED QR CODES

The most directly relevant prior work consists of academic proposals that combine cryptographic operations on QR payloads with server-side or application-side validation. The Quick Response Code Secure system establishes originator validity through digital signatures and a client-server cryptographic architecture, verifying every scanned code against a public key and blocking those that fail verification [11]. Subsequent work has explored related designs in specific application domains: an RSA-based attendance system anchors QR-encoded student data to a class timestamp and validates against a backend service [12]; dynamic-QR payment systems have implemented national cryptographic algorithms with comparative analysis against AES and RSA [13]; and a recent web-application proposal encrypts a JSON Web Token containing session, role, and nonce fields within the QR payload, enforces a short-lived validity window, and maintains a server-side replay cache for single-use semantics [14]. Across these systems, the shared architectural pattern is signed or encrypted QR payloads validated against a backend that holds state for the issuance and, in some cases, the consumption of each code.

Each of these systems demonstrates a viable architecture for trusted QR issuance, yet each is also scoped tightly to its application: anti-phishing for general QR codes through origin verification, school-time recording bound to class schedules, financial transactions under regulatory cryptographic standards, and session handoff between mobile authentication and web sessions. None presents a generalised, application-agnostic framework that exposes validity-mode selection as a first-class issuance decision, and none accompanies the architecture with a formal threat model in the sense developed in Section 4. The taxonomies in recent systematic reviews of QR security confirm this scope-bound pattern [3], [20]. Table 2 summarises the differentiation.

Table 2. Comparative positioning of QR-security approaches against QRify Secure.

Approach	Representative systems	Cryptographic property	Consumption control	Scope
Encryption-centric	Qrafter Pro; chromatic-multiplexing schemes	Payload confidentiality only	None native	Content protection
Reputation-based	Trend Micro QR Scanner	None on the code itself	None	Known-threat blocking
Dynamic-redirection	QR TIGER, Flowcode	None on the code itself	Scheduled deactivation; scan-count limits	Marketing and analytics
Authenticated-issuance (academic)	QRCS; attendance, payment and web-login systems	Signed or signed-and-encrypted payloads, server-validated	Bound to application context, not user-selectable at issuance	Application-bound
QRify Secure (this work)	This work	Signed payloads with server-side validation	User-selected one-time-use or time-limited at issuance	Application-agnostic framework

3.5. THE IDENTIFIED GAP

The four categories each contribute valuable defences against distinct facets of QR-related risk, yet none alone meets the requirements of an organisation seeking to issue authenticated QR codes whose validity can be confirmed at scan time and whose consumption can be controlled at the code level. Encryption-centric tools protect confidentiality without authentication; reputation-based scanners react to known threats without integrity

proof; dynamic-redirection platforms manage destinations without cryptographic authenticity; and academic authenticated-issuance proposals demonstrate trusted-issuance architectures, but each is constrained to a specific application domain. The gap that QRify Secure addresses is therefore not the existence of any single feature, but the consolidation of signed-payload issuance, server-side validation, and explicit user-facing consumption control into a single application-agnostic framework, supported by a formal threat model articulated for arbitrary deployment contexts.

IV. THREAT MODEL

This section formalises the threat model that defines what QRify Secure is designed to defend against, and what it is not. The model is the input to the security analysis in Section 8, which maps each in-scope attack to the corresponding defence mechanism.

4.1. System Assets and Trust Assumptions

The threat model identifies four assets protected by QRify Secure: the authenticity of each issued code, meaning its originating from a trusted issuer; the integrity of the encoded payload, meaning its remaining unmodified after issuance; the freshness state of each code, meaning its remaining unconsumed in one-time-use mode and unexpired in time-limited mode; and, derivative of these, the reputation of the issuing organisation and the trust placed by scanning users. The architecture assumes the issuance server is not compromised and is operated by a trusted party, that standard cryptographic primitives hold their guarantees under deployment conditions, that the issuer's signing key remains confidential to the issuance service, and that the scanning user's device is not compromised at the operating-system level.

4.2. Adversary Capabilities

The adversary model assumes an attacker who can reproduce or substitute physical QR codes in any setting where codes are displayed, capture codes in transit between issuer and user, attempt replay of intercepted codes against the verification endpoint, run social-engineering campaigns to induce scanning of substituted codes, and submit guessed token values to the issuance and verification interfaces. The adversary cannot break the cryptographic primitives in use, compromise the issuance server, obtain the issuer's signing key, or modify entries in the issuance database directly. The adversary's reach is therefore at the symbol layer and the verification request, not at the issuance state. These capabilities are consistent with documented QR phishing patterns observed in real-world deployments [4], [6].

4.3. In-Scope Attacks

Four classes of attack are in scope. First, QR substitution within issuance scope: an attacker replaces a legitimate code with a substituted one of the same visual format, intending the system to validate the substitute as if it had been issued [11]. Second, replay of consumed one-time-use codes: an attacker presents a code that has already been validly scanned, either through pre-consumption interception or through post-consumption reuse, a class of attack demonstrated empirically against QR-based login deployments [5]. Third, reuse of expired time-limited codes: an attacker presents a code after its configured validity window has closed. Fourth, brute-force token guessing: an attacker submits guessed token values to the verification endpoint, hoping to enumerate valid codes. Defences against each are presented in Section 8.

4.4. Out-of-Scope Attacks

Several attack classes are deliberately out of scope. Issuance-server compromise: if an attacker gains operational control of the issuance backend, they can issue codes pointing to attacker-controlled destinations, and these will appear authentic to QRify Secure. Supply-chain attacks against the deployed frontend build or its dependencies: a compromised build artefact could substitute payloads at issuance or verification time. Malicious URLs registered by authorised issuers: the system authenticates that a code was issued, not that its URL is benign. Out-of-system QR codes: the verification workflow operates only on codes issued through QRify Secure, and the system makes no claim about codes scanned from other sources, defining the closed-loop boundary discussed in Section 9. Side-channel and cryptanalytic attacks against the underlying primitives: the model assumes standard primitive security and does not attempt to defend against novel cryptanalysis.

V. SYSTEM DESIGN

5.1. Design Rationale and Security Objectives

QRify Secure follows a closed-loop issuance-and-validation model in which QR codes are generated by a trusted backend, registered at issuance time, and verified server-side at scan time. This model was articulated in the prior open-access record [15] and is developed here into a complete design specification. It is shaped by four security objectives. Authenticity and integrity require that a verifier can confirm a scanned code was legitimately

issued and has not been modified since issuance. Controlled use requires that an issuer can constrain when and how often a code may be consumed, supporting both single-use and time-bounded semantics at issuance time. Centralised verification places the validity decision at the backend rather than the scanning client, providing consistent enforcement and auditable outcomes. Operational deployability requires that the architecture remains compatible with standard web infrastructure and does not depend on bespoke hardware. These objectives map directly to the assets and in-scope attacks of Section 4: authenticity and integrity protect against substitution; controlled use addresses replay and expired-code reuse; centralised verification eliminates trust in the scanning client; and operational deployability ensures the system can run on infrastructure the issuer already owns.

5.2. System Architecture

QRify Secure is organised into three layers with a clear trust boundary between the client and the backend. The client layer provides the user-facing interfaces for QR issuance and scanning, rendering the issuance form, capturing scanned codes from camera or uploaded image, and presenting verification results. The API layer hosts the issuance, verification, replay-check, and logging endpoints; it is the only layer with authority to mint new codes or to update consumption state, and it constitutes the trusted side of the architecture. All security-critical decisions, including signature verification, consumption-state checks, and validity-window evaluation, occur within the API layer, never at the client. The data layer persists user records, issued-code metadata, scan logs, and token-consumption history within the API trust boundary, and is reachable only through the API endpoints. An optional mobile-wrapper layer integrates the web application into a native shell to provide camera and image-saving functionality, but does not extend the trust boundary. Figure 1 illustrates the architecture and the trust boundary.

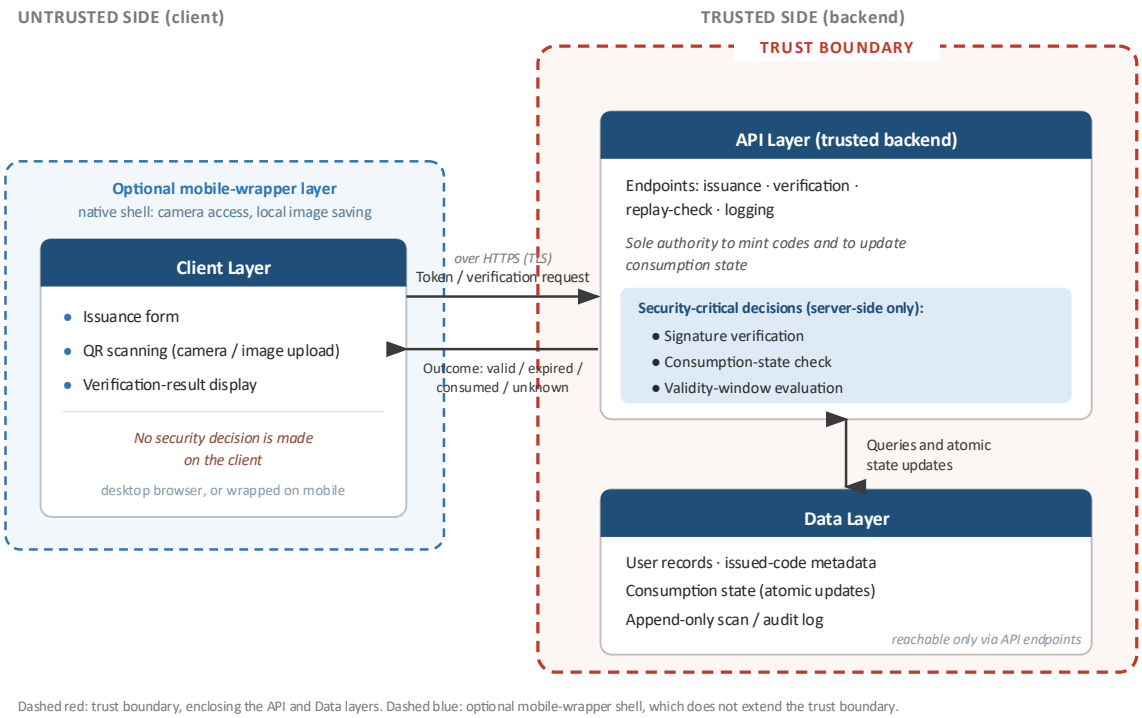

Figure 1: QRify Secure three-layer architecture with the API-client trust boundary.

5.3. Issuance Workflow

The issuance workflow proceeds in six stages. First, the user authenticates to the issuance interface using a username and a password hashed under a current-generation algorithm. Second, the user supplies a destination URL and selects the validity mode, configuring the expiry duration where time-limited. Third, the backend generates a unique token bound to the URL, the validity mode, and the issuance metadata, and signs the token using the issuance service's key. Fourth, the backend persists the issuance record, including the token identifier, the destination URL, the validity mode, the expiry timestamp, and the initial consumption state. Fifth, the backend encodes the signed token into a QR image and returns it to the user. Sixth, the user distributes the QR image through whatever physical or digital channel the application requires. Figure 2 presents the issuance flow.

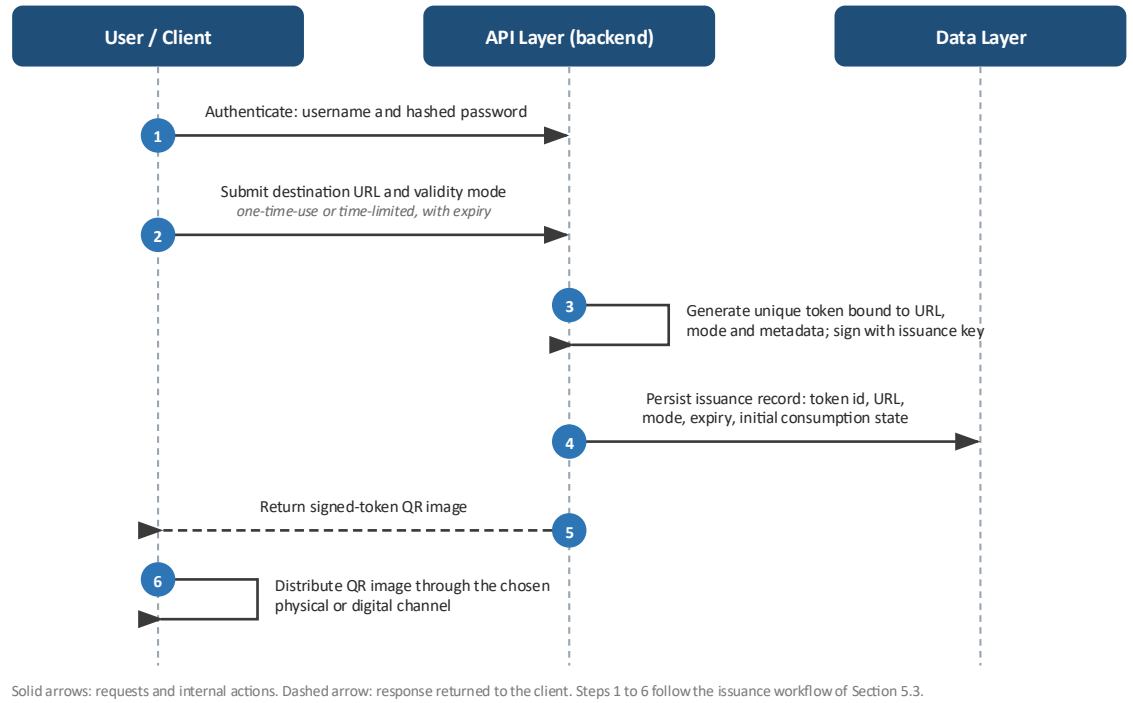

Figure 2: Issuance workflow: from user authentication to signed-token QR delivery.

5.4. VERIFICATION WORKFLOW

The verification workflow proceeds in five stages. First, the scanning client decodes the QR image using a standard QR decoding library, recovering the embedded token. Second, the client transmits the token to the verification endpoint over TLS. Third, the backend verifies the token's signature against the issuance service's verifying key, rejecting tokens whose signatures do not validate. Fourth, the backend retrieves the corresponding issuance record and checks the current consumption state and expiry; codes consumed in one-time-use mode or expired in time-limited mode are rejected. Fifth, the backend updates the consumption state, records the scan event in the audit log, and returns the outcome to the client. The result distinguishes among valid, expired, consumed, and unknown states. The state check at the fourth stage is the load-bearing security property: it is what prevents replay of consumed codes and reuse of expired codes, beyond what signature verification alone provides. Figure 3 presents the verification flow.

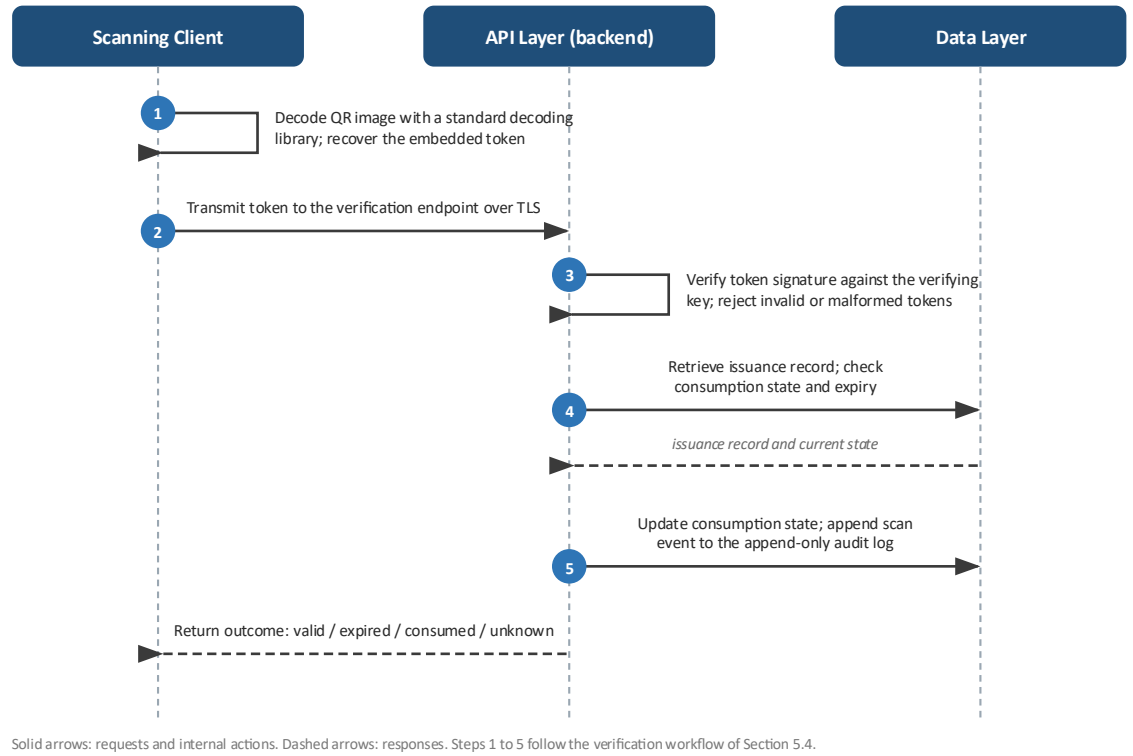

Figure 3: Verification workflow: from QR scan to validity decision.

5.5. Validity Control: One-Time-Use and Time-Limited Models

QRify Secure exposes two consumption models that the issuer selects at issuance time. The one-time-use model invalidates a code after its first successful verification, providing strong replay defence for single-action scenarios such as one-time authentication, single-use ticket scanning, or document-handoff confirmations. The time-limited model accepts repeated verifications within a configured validity window and rejects scans after the window closes, supporting scenarios such as event entry valid for the event's duration, meeting-share links valid for the meeting interval, or transaction approval valid for a short authorisation window. Both models are enforced server-side by checking and updating the persistent consumption state at each verification request, and the choice between them is a first-class issuance decision rather than a deployment-time configuration. Exposing the selection at issuance allows a single deployment to serve heterogeneous use cases without redeployment. The verification endpoint also handles two terminal states regardless of model: a token that has expired or been consumed is rejected with the appropriate state response, and a token that matches no issuance record is rejected as unknown. The system therefore distinguishes among four verification outcomes in user-facing communication: valid, expired, consumed, and unknown.

5.6. Data Layer and Persistence

The data layer holds the state required for verification to function and for issuance to be auditable. Its functional requirements are durable storage of every issued code's metadata; persistent consumption state that can be queried and updated atomically at verification time; an append-only scan log that records each verification event for audit, capturing the token identifier, the timestamp, the outcome, and where available the scanning context; and concurrency-safe transactional semantics so that consumption decisions cannot race when the same code is presented simultaneously through different clients. The audit log supports both operational monitoring and incident response. The design is technology-agnostic at this layer: any relational or document store that provides ACID transactions and durable persistence can satisfy the requirements; the specific technology used in the prototype is presented in Section 6.

VI. IMPLEMENTATION

6.1. Technology Stack

The prototype implements the architecture of Section 5 on a modern web-application stack. The frontend uses React with Vite and TypeScript for the issuance and scanning interfaces, with client-side routing and a utility-first CSS framework for the visual layer. The API layer runs on a lightweight serverless framework executed

within a managed serverless platform, providing a deployment surface suited to small-to-medium issuance volumes without bespoke server provisioning. The data layer is hosted on a managed PostgreSQL service with HTTP-mediated access. Cryptographic operations use the JOSE library for JSON Web Token handling and a bcrypt implementation for password hashing. QR generation uses a standard QR-encoding library on the issuance side, and QR decoding uses a standard decoding library on the scanning client. A native WebView wrapper integrates the web application into a mobile shell for camera-based scanning and local image saving. This stack differs from the Python and Flask stack proposed in the prior open-access record [15]; the substitution was made during implementation to leverage managed serverless infrastructure for deployment simplicity.

6.2. CRYPTOGRAPHIC COMPONENT

Cryptographic operations in the prototype follow the JOSE specification for JSON Web Token handling. The issuance service constructs a token bound to the QR identifier, the destination URL, the validity mode, and the issuance metadata, signs the token, and persists the issuance record. The verification service retrieves the token, verifies its signature, and queries the data layer for the consumption-state check described in Section 5.4. The architecture in Section 5 is primitive-agnostic and accommodates any JOSE-compliant signing algorithm; the deployed prototype's specific algorithm is a deployment-time configuration. The asymmetric RSA cryptography of the original design proposal is not realised in the current build, and asymmetric signing with an explicit key-management policy is identified as the lead near-term direction in Section 10. Password storage uses a bcrypt implementation at a standard work factor.

6.3. QR GENERATION AND DECODING

QR generation in the issuance flow encodes the signed token into a Standard QR Code (Model 2) symbol at the version and error-correction level required by the token's byte length. The symbol is returned as an image embedded in the issuance interface and is available for download. QR decoding at scan time processes camera frames or uploaded images and recovers the embedded payload. The decoded payload is the signed token; the client transmits the token to the verification endpoint without performing local validity decisions.

6.4. VERIFICATION SERVICE

The verification service is an API-layer endpoint that performs four operations in sequence on each received token: it verifies the signature, rejecting tokens with invalid signatures or malformed structure; it retrieves the corresponding issuance record from the data layer; it evaluates the consumption-state and expiry fields against the current request, rejecting tokens consumed in one-time-use mode or expired in time-limited mode; and it updates the consumption state for one-time-use codes, marking them consumed before returning the outcome. The state update and the result return are transactional with respect to the data layer, preventing race conditions when the same token is scanned concurrently through different clients. The outcome distinguishes among valid, expired, consumed, and unknown states.

6.5. LOGGING AND AUDIT TRAIL

The data layer maintains an append-only scan log alongside the issuance records. Each verification request, regardless of outcome, generates a log entry capturing the token identifier, the timestamp, the verification outcome, and where available the requesting context. The log supports operational monitoring, by surfacing unusual scan patterns such as elevated unknown-token rates, and incident response, by enabling reconstruction of the verification history around a contested event. Log entries are not user-editable; the schema enforces append-only semantics at the database level. The QR-history feature exposed to issuance users is derived from the issuance records, not from the audit log, preserving the separation between operational and audit data.

VII.EVALUATION

7.1. EVALUATION METHODOLOGY

The evaluation methodology for QRify Secure is functional verification against the threat model of Section 4 and the design objectives of Section 5. Three deliberate scope decisions shape the methodology. First, the evaluation is qualitative rather than quantitative: scan-to-result latency, throughput under load, and storage overhead are not measured, because the prototype's purpose is to demonstrate architectural correctness and workflow coherence rather than to establish performance benchmarks. Second, the evaluation does not include a formal user study; user-facing behaviour is assessed against the capstone test protocol and the design's trust-communication objective, not against measured human-subjects performance. Third, the evaluation does not include adversarial penetration testing; the implementation's defences are described and reasoned about in Section 8, but the testing protocol exercises functional flows rather than simulated attacks. The evaluation reports functional-verification outcomes, comparative positioning against the surveyed alternatives, and operational observations.

7.2. FUNCTIONAL VERIFICATION

The testing protocol exercised six test cases covering issuance, verification, validity-mode handling, and result communication. Each test case was executed against the deployed prototype and produced outcomes that matched the expected behaviour. Table 3 summarises the test cases and maps each to the threat-model elements of Section 4. Of the four in-scope attacks, the protocol directly exercises the defence against expired-code reuse, via TC-05. The defences against the remaining three are present in the implementation logic, namely signature verification at every verification request for substitution, the consumption-state check for replay of consumed codes, and unknown-token rejection together with token-space size for brute-force guessing, but were not directly exercised by the present protocol. This is a methodological limitation of the evaluation; an empirical adversarial-testing protocol that exercises each defence directly is identified as a near-term direction in Section 10.

Table 3. Functional test cases mapped to security properties and in-scope attacks.

ID	Test description	Security property exercised	In-scope attack defended
TC-01	Generate a QR code in one-time-use mode for a supplied URL	Issuance integrity (signed-payload generation)	Substitution
TC-02	Confirm the interface clearly signals one-time-use semantics	User-facing trust signal	UX support; not a direct defence
TC-03	Generate a QR code in time-limited mode with a configured expiry	Issuance integrity in time-limited mode	Substitution
TC-04	Verify a code before its expiry or consumption; expect valid	State-check positive path	Baseline behaviour; not a defence
TC-05	Verify a code after its allowed lifetime; expect expired	State-check expiry detection	Expired-code reuse
TC-06	Confirm verification results are clearly communicated	User-facing trust signal	UX support; not a direct defence

7.3. COMPARATIVE POSITIONING

To position QRify Secure against the surveyed alternatives of Section 3, the four in-scope attacks from the threat model are used as comparison dimensions. Each approach is rated on whether it defends against each attack: a rating of Yes indicates a cryptographic or architectural defence; Partial indicates a defence that addresses some but not all instances; No indicates no native defence; and System-dependent indicates that some systems in the category defend while others do not. Table 4 summarises the comparison. It shows that QRify Secure provides explicit defences against each of the four in-scope attacks, distinguishing it from the surveyed alternatives in two ways. First, the commercial categories lack the integrity-and-freshness combination at the code level: encryption protects content but not origin, reputation checks address known threats but not code legitimacy, and dynamic-redirection shifts trust to a redirect server. Second, the academic authenticated-issuance category provides the necessary cryptographic primitives but ties consumption controls to a specific application context, so defences against replay and expired reuse are present in some systems and absent in others. QRify Secure exposes the consumption-mode selection as a first-class issuance decision applicable across deployment contexts, which is what enables consistent defence against all four in-scope attacks regardless of application domain.

Table 4. Comparative threat-defence positioning across QR-security approaches.

Approach	Substitution	Replay of consumed	Expired-code reuse	Brute-force guessing
Encryption-centric	Partial (confidentiality, not authenticity)	No	No	No
Reputation-based	Partial (URL check; zero-day blind)	No	No	No
Dynamic-redirection	No (no code-level authenticity)	No	Partial (scheduled deactivation only)	No
Authenticated-issuance (academic)	Yes (signature)	System-dependent	System-dependent	System-dependent
QRify Secure (this work)	Yes (signed payload)	Yes (server-side consumption check)	Yes (server-side expiry check)	Yes (token-space size and unknown-token rejection)

7.4. OPERATIONAL OBSERVATIONS

Beyond the formal test cases, operational use of the prototype across the capstone development period yielded several qualitative observations. The scan-to-result latency was near-instantaneous in normal use, with no perceptible delay reported in the capstone's own qualitative assessment. The result presentations for valid, expired, and consumed states distinguished cleanly in the user interface, supporting the trust-signal objective of Section 5. The mobile-wrapper layer operated correctly on standard Android and iOS browsers for both camera-based scanning and image upload, with the verification result returned to the wrapped interface as in the desktop browser. No formal user study was conducted to measure trust-signal effectiveness or task completion under adversarial conditions; this is identified in Section 10 as a near-term direction.

VIII. SECURITY ANALYSIS

The threat-model elements of Section 4 are now mapped to the implementation mechanisms of Section 6. Each in-scope attack is paired with the architectural and implementation feature that defends against it. The mapping makes the manuscript's defensive claims explicit and falsifiable: a reviewer who disagrees with a claim can identify exactly which mechanism is asserted and challenge it. Table 5 summarises the mapping.

Table 5. Threat-to-defence mapping for QRify Secure.

In-scope attack	Defence mechanism	Sections
Substitution within issuance scope	Signature verification at every verification request; forged tokens fail the signature check before state lookup	5.4, 6.2, 6.4
Replay of consumed one-time-use codes	Server-side consumption-state check at verification; transactional state update prevents race conditions	5.4, 5.5, 6.4
Reuse of expired time-limited codes	Server-side expiry-state check against the issuance record at verification	5.4, 5.5, 6.4
Brute-force token guessing	Token-space size; unknown-token rejection at verification; rate limiting as a deployment-time concern	5.4, 6.4

8.1. DEFENCES AGAINST IN-SCOPE ATTACKS

Against QR substitution within issuance scope, the verification service requires every received token to carry a signature produced by the issuance service's signing key. A substituted token, unless forged by an attacker who has obtained the signing key, will fail signature verification at the first step of the verification pipeline and never reach the data-layer state check. The substitution defence therefore rests on the integrity of the signing key, held confidential by the issuance service per the trust assumptions of Section 4.1.

Against replay of consumed one-time-use codes, the consumption-state check at verification time rejects any token whose issuance record records prior consumption. Because the state update at the time of valid consumption is transactional with respect to the data layer, two concurrent verification attempts of the same one-time-use token cannot both succeed: at most one wins the transaction and consumes the token, and the other receives a consumed-state response. Against expired-code reuse, the expiry-timestamp comparison against the current verification time rejects time-limited tokens presented after their validity window; the implementation logic is symmetrical with the consumption check.

Against brute-force token guessing, the verification endpoint returns an unknown-token response for any token whose identifier does not match a current issuance record. The defence relies on the token-space size: at the token length used, the probability of guessing a valid token is negligible within practical attack-volume thresholds. Rate limiting at the API layer further constrains attack throughput, although this is an operational deployment concern rather than an architectural property of the system.

8.2. ACKNOWLEDGED LIMITATIONS AGAINST OUT-OF-SCOPE ATTACKS

Several attack classes from Section 4.4 are honestly acknowledged as out of scope. Issuance-server compromise gives an attacker the authority to issue codes pointing to attacker-controlled destinations; QRify Secure does not defend against this because the trust model places the issuance server inside the trust boundary. Supply-chain attacks against the deployed frontend build can introduce code-substitution behaviours at the client layer that bypass the trusted backend's decisions; defending against this requires reproducible builds and integrity-checked distribution, a deployment-process concern rather than an architectural one. Malicious URLs registered by authorised issuers are an insider-threat class that QRify Secure does not defend against the system authenticates that a URL was issued by an authorised party, not that the URL is benign, and operators must combine the framework with content-policy controls if they require both. Out-of-system QR codes are excluded by design, defining the closed-loop boundary discussed in Section 9. Side-channel and cryptanalytic attacks against the

underlying primitives are also out of scope: the model assumes that standard cryptographic primitives hold their published guarantees under deployment conditions.

IX. DISCUSSION AND LIMITATIONS

9.1. THE CLOSED-LOOP SCOPE AS A DESIGN CHOICE

QRify Secure does not attempt to defend against Quishing in the wild on arbitrary out-of-system QR codes. The verification workflow operates only on codes issued through the system itself, and the user-facing interface communicates this constraint by rejecting any token that matches no current issuance record. This is a deliberate scope decision rather than a limitation. The architecture is designed for organisational use cases where the issuer controls the issuance channel, such as ticketing, event entry, document verification, secure-link distribution, and internal authentication. Within these use cases, the closed-loop boundary is what enables the strong cryptographic and freshness guarantees that the framework provides. Extending the system to defend against arbitrary out-of-system codes would require a fundamentally different architecture, plausibly involving crowd-sourced threat intelligence or browser-level interception, named as a future direction in Section 10.

9.2. DEPLOYMENT CONSIDERATIONS

Three deployment considerations shape QRify Secure's operational profile. First, the framework requires connectivity at scan time: every verification involves a request to the issuance service, and offline verification is not supported by the closed-loop architecture, so deployment in network-constrained environments would require fallback mechanisms the present design does not provide. Second, the issuance service must be operated by a trusted party with appropriate key management; an organisation deploying QRify Secure inherits responsibility for the confidentiality of the signing key and the integrity of the issuance database. Third, the user-facing trust signal depends on the scanning interface running QRify Secure's verification path: codes scanned by a generic QR scanner outside the system will simply resolve to the encoded URL without the framework's freshness or substitution checks.

9.3. LIMITATIONS

The principal limitations of the present work are honestly stated. The deployed prototype does not realise the asymmetric RSA signing of the original design proposal; the JOSE-based signing in the current build provides integrity sufficient for the closed-loop threat model but does not extend to use cases requiring non-repudiation or decentralised verification. The functional evaluation of Section 7 exercises a subset of the in-scope attack defences and does not include adversarial penetration testing or a formal user study, and the operational observations are qualitative rather than measured. The audit log is append-only at the database level but the implementation does not include tamper-evident chaining of log entries, which would be required for high-assurance forensic deployments. Each of these limitations corresponds to a named future-work direction in Section 10.

X. CONCLUSION AND FUTURE WORK

10.1. CONCLUSION

QRify Secure addresses a gap that the existing QR-security literature has filled for specific applications but has not consolidated for general use. Encryption-centric tools protect QR contents without authenticating issuers, reputation-based scanners react to known threats without integrity proof, dynamic-redirection platforms manage destinations without cryptographic authenticity, and academic authenticated-issuance systems demonstrate trusted-issuance architectures but remain scoped to their respective application domains. This paper has presented QRify Secure as an application-agnostic framework that consolidates signed-payload issuance, server-side validation, and explicit user-facing consumption-mode selection into a single architecture, supported by a formal threat model articulated across deployment contexts. The deployed prototype operates within the closed-loop scope and provides explicit defences against the four in-scope attacks identified in Section 4. The framework's value is the consolidation and the cross-domain applicability, not a novel cryptographic primitive.

10.2. FUTURE WORK

Three near-term directions are prioritised. First, asymmetric signing with an explicit key-management policy: implementing RS256 or ECDSA signing in the existing JOSE configuration, with a dedicated keypair and a rotation policy, would close the gap between the design proposal and the implementation, and would extend the framework to use cases requiring non-repudiation, such as legal-grade document authentication or cross-organisational trusted issuance. Second, empirical adversarial testing: an explicit penetration-testing protocol that exercises each in-scope attack defence in turn would convert the methodological evaluation of Section 7 into a security-validated evaluation. Third, an organisational pilot deployment in a real issuer context, such as event ticketing or secure-document handoff, accompanied by a small user study to measure trust-signal effectiveness and task completion. Each direction is independent and can be pursued in parallel.

REFERENCES

[1] ISO/IEC, *Information technology - Automatic identification and data capture techniques - QR code bar code symbology specification.*, International Standard (ISO/IEC jointly published) ISO/IEC 18004:2024, Geneva, Switzerland., 2024. Accessed: Dec. 14, 2025. [Online]. Available: https://www.iso.org/standard/83389.html

[2] R. Siddiqi, S. Singh, L. Zuck, and C. Kanich, "Tracking, But Make It Offline: The Privacy Implications of Scanning QR Codes Found in the World," in *Proceedings of the 2023 Workshop on Technology and Consumer Protection*, National Science Foundation, 2023. [Online]. Available: https://conpro23.ieee-security.org/papers/siddiqi-conpro23.pdf

[3] H. A. M. Wahsheh and F. L. Luccio, "Security and Privacy of QR Code Applications: A Comprehensive Study, General Guidelines and Solutions," *Information*, vol. 11, no. 4, p. Art. 217, Apr. 2020, doi: 10.3390/info11040217.

[4] Insikt Group®, "Security Challenges Rise as QR Code and AI-Generated Phishing Proliferate," Recorded Future, CTA-2024-0718, Jul. 2024. [Online]. Available: https://assets.recordedfuture.com/insikt-report-pdfs/2024/cta-2024-0718.pdf

[5] X. Zhang *et al.*, "Demystifying the (In)Security of QR Code-based Login in Real-world Deployments," in *Proceedings of the 34th USENIX Security Symposium (USENIX Security '25)*, Seattle, WA, USA: USENIX Association, Aug. 2025, pp. 3161–3180. [Online]. Available: https://www.usenix.org/conference/usenixsecurity25/presentation/zhang-xin

[6] J. Nadeau, "Quishing: A growing threat hiding in plain sight | IBM," IBM Think – Security. Accessed: Dec. 21, 2025. [Online]. Available: https://www.ibm.com/think/insights/quishing-growing-threat-hiding-plain-sight

[7] U. S. Department of Health and Human Services (HHS) and Health Sector Cybersecurity Coordination Center (HC3), "QR Code-Based Phishing (Quishing) as a Threat to the Health Sector," HC3: White Paper 202310231700, Oct. 2023. [Online]. Available: https://www.hhs.gov/sites/default/files/qr-codes-and-phishing-as-a-threat-to-the-hph-white-paper.pdf

[8] National Cyber Security Centre (NCSC) Ireland, "Quick Guide: QR Code Phishing & Scams," National Cyber Security Centre (NCSC) Ireland, Ireland, Jan. 2025. [Online]. Available: https://www.ncsc.gov.ie/pdfs/Quick_Guide_QR_Code_Phishing_Scams.pdf

[9] Canadian Centre for Cyber Security, "Security Considerations for QR Codes," Canadian Centre for Cyber Security, Awareness Series ITSAP.00.141, Jan. 2024.

[10] P. N. S. Agustin-Crescencio, L. Hernandez-Gonzalez, P. Guevara-Lopez, O. U. Juarez-Sandoval, J. Ramirez-Hernandez, and E. S. Estevez-Encarnacion, "A Comprehensive QR Code Protection and Recovery System Using Secure Encryption, Chromatic Multiplexing, and Wavelength-Based Decoding," *Applied Sciences*, vol. 15, no. 17, Sep. 2025, doi: 10.3390/app15179708.

[11] V. Mavroeidis and M. Nicho, "Quick Response Code Secure: A Cryptographically Secure Anti-Phishing Tool for Qr Code Attacks," in *Computer Network Security: 7th International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2017, Warsaw, Poland, August 28-30, 2017, Proceedings*, vol. 10446, Warsaw, Poland: Springer, 2017. doi: 10.1007/978-3-319-65127-9_25.

[12] A. I. Irawan, I. H. Santoso, Istikmal, and M. Rahayu, "Implementation of QR Code Attendance Security System Using RSA and Hash Algorithms," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 13, no. 1, pp. 53–59, Jan. 2024, doi: 10.22146/jnteti.v13i1.4395.

[13] Y. Zhou, B. Hu, Y. Zhang, and W. Cai, "Implementation of Cryptographic Algorithm in Dynamic QR Code Payment System and Its Performance," *IEEE Access*, vol. 9, pp. 122362–122372, 2021, doi: 10.1109/ACCESS.2021.3108189.

[14] W. Zita, "Secure QR Code Login System with Role-Based Access for Web Applications," Aug. 2025, *TechRxiv*. doi: 10.36227/techrxiv.175616741.16408346.

[15] H. El-Taj, M. Al-Gafri, D. Al-Sousi, and R. Bin Homran, "QRify Secure," *Journal of International Research for Engineering & Management (JOIREM)*, vol. 4, no. 1, pp. 1–7, Jan. 2026, doi: 10.5281/zenodo.18151237.

[16] R. Wudhikarn, P. Charoenkwan, and K. Malang, "Deep Learning in Barcode Recognition: A Systematic Literature Review," *IEEE Access*, vol. 10, pp. 8049–8072, 2022, doi: 10.1109/ACCESS.2022.3143033.

[17] DENSO WAVE, "QR Code Standardization," QRcode.com. Accessed: Dec. 14, 2025. [Online]. Available: https://www.qrcode.com/en/about/standards.html

[18] L. Karrach, E. Pivarčiová, and P. Božek, "Identification of QR Code Perspective Distortion Based on Edge Directions and Edge Projections Analysis," *J. Imaging*, vol. 6, no. 7, p. 67, Jul. 2020, doi: 10.3390/jimaging6070067.

[19] ISO/IEC, *ISO/IEC 23941:2022 - Information technology - Automatic identification and data capture techniques - Rectangular Micro QR Code (rMQR) bar code symbology specification*, International Standard (ISO/IEC jointly published) ISO/IEC 23941:2022, 2022. Accessed: Dec. 21, 2025. [Online]. Available: https://www.iso.org/standard/77404.html

[20] D. Njuguna and J. Ndia, "Quick Response Code Security Attacks and Countermeasures: A Systematic Literature Review," *JCS*, vol. 7, no. 1, pp. 1–20, Feb. 2025, doi: 10.32604/jcs.2025.059398.

[21] K. Erkan, "Encrypting QR Codes with Qrafter." Accessed: May 23, 2026. [Online]. Available: https://qrafter.app/blog/posts/encrypting-qr-codes

[22] Trend Micro, "Scan QR Codes Safely with the Trend Micro QR Scanner," TrendLife Blog. Accessed: May 23, 2026. [Online]. Available: https://www.trendlife.com/en-us/blog/2018/06/18/scan-qr-codes-safely-with-the-trend-micro-qr-scanner/

[23] F. Sharevski, A. Devine, E. Pieroni, and P. Jachim, "Phishing with Malicious QR Codes," in *Proceedings of the 2022 European Symposium on Usable Security*, in EuroUSEC '22. New York, NY, USA: Association for Computing Machinery, Sep. 2022, pp. 160–171. doi: 10.1145/3549015.3554172.

[24] V. V., "What is a Dynamic QR Code, and How Can You Create and Use It | QR Tiger." Accessed: May 23, 2026. [Online]. Available: https://www.qrcode-tiger.com/how-do-dynamic-qr-codes-work

[25] Alex, "How to edit where a Flowcode scans to." Accessed: May 23, 2026. [Online]. Available: https://help.flowcode.com/en/articles/6927428-how-to-edit-where-a-flowcode-scans-to