

QR Code Validation Update

Here is the **split version of your sequence diagram** presented in **three logical parts**, ideal for step-by-step presentation:

✓ Diagram 1: QR Code Generation

```
mermaid

sequenceDiagram
    participant QG as QR Generator System
    participant VD as Vendor DB
    participant CSG as Cryptographic Signature Generator

    %% QR Code Generation Process
    rect rgb(232,245,233)
    QG->>VD: Store QR Metadata (Payload + Timestamp)
    VD-->>QG: Return Vendor Validation URL
    QG->>QG: Embed Validation URL into Payload
    QG->>CSG: Request Signature (Payload + URL + Timestamp)
    CSG->>CSG: Compute Hash and Sign using Private Key
    CSG-->>QG: Return Digital Signature
    QG->>QG: Embed Signature into QR Code
    end
```

✓ Diagram 2: Signature Validation on Scanning

```
mermaid

sequenceDiagram
    participant QS as QR Scanner/App
    participant CSV as Cryptographic Signature Validator

    %% QR Scan and Signature Validation
    rect rgb(227,242,253)
    QS->>CSV: Validate Signature (Payload + URL + Timestamp + Signature)
    CSV->>CSV: Compute & Verify Signature using Public Key
    alt Signature Valid
        CSV-->>QS: Signature Verified ✓
    else Invalid Signature
        CSV-->>QS: Signature Invalid ✗
    end
    end
```

✓ Diagram 3: Payload Comparison & Threat Intelligence Check

```
mermaid

sequenceDiagram
    participant QS as QR Scanner/App
    participant VE as Vendor Endpoint
    participant TDB as Threat Intelligence DB

    %% Payload Integrity & Threat Validation
    rect rgb(255,249,196)
    QS->>VE: Fetch Original Payload using Vendor Validation URL
    VE-->>QS: Return Original Payload
    QS->>QS: Compare Scanned Payload with Vendor Payload
    end
```

```

alt Payloads Match
  QS->>TDB: Check URL in Payload Against Threat Intelligence DB
  TDB-->>QS: Return Threat Analysis Result
  alt URL Safe
    QS-->>QS: Display Trust Indicator ✓
  else URL Malicious or Suspicious
    QS-->>QS: Display Warning ✗
  end
else Payload Mismatch
  QS-->>QS: Display Tampering Alert ✗
end
end

```

Let me know if you'd like these diagrams as exportable SVGs or bundled into a presentation-friendly format like **PDF slides** or **Markdown with titles**.



please create a slide explanation for each sequence diagram

Here's a **slide-wise explanation** of each of the three sequence diagrams, structured for presentation clarity. You can directly use this for a deck (e.g., PowerPoint, Google Slides, or PDF).

🎯 Slide 1: QR Code Generation Process

Title: Secure QR Code Generation with Embedded Signature and Vendor URL

Key Steps:

1. **QR Generator System** prepares the payload (e.g., content, timestamp).
2. It stores this metadata in a **Vendor Database** to create a verifiable reference.
3. The **Vendor Endpoint URL** (used for validation) is embedded into the payload.
4. The payload (including the vendor URL and timestamp) is sent to the **Cryptographic Signature Generator**.
5. The generator computes a **SHA-256 hash** of the payload and signs it using a **private RSA/ECC key**.
6. The **signed payload** is embedded back into the final QR code.

Outcome:



QR code now contains secure payload + embedded validation URL + digital signature.

🎯 Slide 2: Signature Validation at Scan Time

Title: Digital Signature Validation During Scanning

Key Steps:

1. The **QR Scanner** extracts:
 - Payload
 - Timestamp
 - Vendor URL

- Embedded digital signature
- 2. It sends these to the **Cryptographic Signature Validator**.
- 3. The validator:
 - Recomputes the hash from the payload
 - Verifies the signature using the **public key**
- 4. Based on verification:
 - If valid → Signature Verified 
 - If invalid → Signature Invalid 




Outcome:

Authenticates that the payload was **untampered** since it was signed.

Slide 3: Payload Integrity & Threat Validation

Title: Payload Integrity Comparison & Threat Intelligence Check

Key Steps:

1. The scanner sends the **Vendor Validation URL** to the **Vendor Endpoint**.
2. The vendor returns the **original payload and metadata**.
3. The scanner compares:
 - **Scanned payload** vs **Vendor payload**
 - If mismatch → Tampering Alert 
4. If matched:
 - The scanner sends the embedded URL to the **Threat Intelligence DB**
 - Based on the result:
 - If Safe → Trust Indicator  displayed
 - If Unsafe → Warning  shown

Outcome:

Ensures both **payload authenticity** and **URL safety** with layered defense.

Would you like this formatted into a Google Slides-compatible `.pptx` or `.pdf` deck? I can generate that for download.