Technische Verfahrensdokumentation: Client-seitige Ende-zu-Ende-Verschlüsselung personenbezogener Daten

Anwendung: ISQ-Portal (portal.isq.berlin)

Stand: 19.12.2025

Geltungsbereich: Modul zur Erfassung von Schülernamen (Klarnamen)

1. Zusammenfassung (Management Summary)

Das ISQ-Portal implementiert für die Verarbeitung besonders schützenswerter personenbezogener Daten (hier: Klarnamen von Schülern) eine **Zero-Knowledge-Architektur**. Dies stellt sicher, dass Klarnamen ausschließlich auf dem Endgerät des autorisierten Nutzers (Lehrkraft/Schulleitung) im Klartext vorliegen.

Der Serverbetreiber, Datenbankadministratoren sowie potenzielle Angreifer bei einem Server-Compromise haben zu keinem Zeitpunkt Zugriff auf die Klarnamen oder die zur Entschlüsselung notwendigen kryptografischen Schlüssel. Die Sicherheit beruht auf modernen Web-Standards (Web Crypto API) und etablierten Algorithmen (AES-GCM, PBKDF2).

2. Kryptografisches Konzept

Das Sicherheitsmodell basiert auf einer symmetrischen Verschlüsselung mit einem hierarchischen Schlüsselmanagement, das vollständig Client-seitig im Browser ausgeführt wird.

2.1 Verwendete Algorithmen & Parameter

Komponente	Algorithmus	Parameter / Konfiguration	Zweck
Verschlüsselung (Daten)	**AES-GCM** (Advanced Encryption Standard in Galois/Counter Mode)	**Key-Length:** 256 Bit **IV:** 96 Bit (12 Bytes), randomisiert pro Datensatz **Tag-Length:** 128 Bit (Auth Tag)	Authentifizierte Verschlüsselung der Schülernamen und des Master-Keys. Garantiert Vertraulichkeit und Integrität.
Schlüsselableitung	**PBKDF2** (Password-Based Key Derivation Function 2)	**Hash:** SHA-256 **Iterations:** 100.000 (OWASP Empfehlung) **Salt:** 32 Bytes (256 Bit), CSPRNG	Ableitung eines symmetrischen Schlüssels (KEK) aus dem Benutzerpasswort. Schutz gegen Rainbow-Tables und Brute-Force.
Zufallszahlengenerator	**CSPRNG** (Cryptographically Secure Pseudo-Random Number Generator)	Implementierung: window.crypto.getRandomValues()	Erzeugung von Salts, IVs und dem initialen Master-Key.
Duplikatsprüfung	**HMAC-SHA256** (Keyed-Hash Message Authentication Code)	**Key:** Master-Key (Raw Bytes) **Input:** Normalisierter Klarname	Deterministischer "Blind Index" zur Erkennung doppelt angelegter Schüler, ohne die Verschlüsselung zu schwächen.

2.2 Schlüssel-Hierarchie

Das System unterscheidet zwei logische Schlüssel:

1. **Klarnamenzertifikat (Master-Key):**
    - Ein 256-Bit Zufallswert, der einmalig pro Schule/Klasse generiert wird.
    - Mit diesem Schlüssel werden alle Schülerdaten verschlüsselt.
    - Dieser Schlüssel wird **niemals** im Klartext auf dem Server gespeichert. Er liegt dort nur als *Encrypted Blob* (real_name_encrypted_master_key) vor.
2. **Klarnamenpasswort (User Secret):**
    - Ein vom Nutzer gewähltes Geheimnis.
    - Dient ausschließlich dazu, den *Master-Key* zu entschlüsseln (Wrapping).
    - Es verlässt den Client nur als Hash zur Verifikation, niemals im Klartext.

3. Technischer Prozessablauf

3.1 Initialisierung (Schulleitung)

Dieser Prozess findet einmalig statt, um den kryptografischen Kontext ("Tresor") zu erstellen.

1. **Input:** Benutzer vergibt das *Klarnamenpasswort*.
2. **Generierung:**
    - Browser generiert via CSPRNG den **Master-Key** (256 Bit).
    - Browser generiert einen **Salt** (32 Bytes).
    - Browser generiert einen **IV** (12 Bytes).
3. **Ableitung (Key Wrapping):**
    - KEK = PBKDF2(Passwort, Salt, 100k Iterationen)
    - Encrypted_Master_Key = AES-GCM(Master-Key, KEK, IV)
4. **Export (Backup):**
    - Der *Master-Key* wird dem Nutzer als Datei (.txt) oder QR-Code zum Download angeboten ("Klarnamenzertifikat"). Dies dient der Wiederherstellung bei Passwortverlust.
5. **Persistierung:**
    - An den Server gesendet werden: Encrypted_Master_Key, IV, Salt und ein Password_Hash (für Login-Check).
    - Der Server speichert diese Werte in der Tabelle school_project_mappings.

3.2 Zugriff und Entschlüsselung (Lehrkraft)

Dieser Prozess findet bei jedem Zugriff auf die Schülerliste statt.

1. **Authentifizierung:** Benutzer gibt *Klarnamenpasswort* ein.
2. **Laden:** Client lädt den verschlüsselten Context (Encrypted_Master_Key, Salt, IV) vom Server.
3. **Unwrapping:**
    - Client berechnet erneut KEK = PBKDF2(Passwort, Salt).
    - Client entschlüsselt: Master-Key = AES-GCM-Decrypt(Encrypted_Master_Key, KEK, IV).
    - *Sicherheitsmerkmal:* Ist das Passwort falsch, schlägt die Integritätsprüfung des GCM-Modus fehl (Exception).
4. **Session:** Der *Master-Key* wird temporär im Arbeitsspeicher (RAM) der JavaScript-Runtime gehalten. Er wird nicht im localStorage oder in Cookies gespeichert.

3.3 Datenverarbeitung (Schüler anlegen/lesen)

Schreiben (Verschlüsseln):

1. Benutzer gibt Name "Max Mustermann" ein.
2. Client generiert einen **neuen, einzigartigen IV** (12 Bytes) für diesen Datensatz.
3. Verschlüsselung: Ciphertext = AES-GCM("Max Mustermann", Master-Key, Pupil_IV).
4. Blind Index: Hash = HMAC-SHA256("max mustermann", Master-Key).
5. Server speichert: Ciphertext, Pupil_IV und Hash in Tabelle pupils.

Lesen (Entschlüsseln):

1. Client lädt Liste der Schüler (Chiffretexte + IVs).
2. Client iteriert über die Liste und führt für jeden Eintrag aus:
    - Klarname = AES-GCM-Decrypt(Ciphertext, Master-Key, Pupil_IV).
3. Die Klarnamen werden ins DOM (HTML-Tabelle) injiziert.

4. Datenschema & Speicherformat

Die Speicherung in der MariaDB-Datenbank erfolgt in binär optimierten Feldern (VARBINARY/BINARY), um Encoding-Probleme zu vermeiden. Für den Transport (JSON API) werden binäre Daten Base64 kodiert.

Tabelle: school_project_mappings (Schlüsselverwaltung)

Feld	Typ	Inhalt	Sicherheitsrelevanz
real_name_encrypted_master_key	VARBINARY(255)	Der mit dem Passwort verschlüsselte Master-Key.	Ohne Passwort mathematisch nicht entschlüsselbar.
real_name_master_key_iv	BINARY(12)	Initialisierungsvektor für den Master-Key.	Notwendig für AES, öffentlich bekannt (nicht geheim).
real_name_kdf_salt	VARBINARY(64)	Zufallswert für die PBKDF2-Ableitung.	Verhindert Rainbow-Table-Angriffe.

Tabelle: pupils (Personenbezogene Daten)

Feld	Typ	Inhalt	Sicherheitsrelevanz
real_name_encrypted_fullname	VARBINARY(255)	AES-256 verschlüsselter Name (inkl. Auth Tag).	Enthält die personenbezogenen Daten. Für den Server unlesbar.
real_name_encryption_iv	BINARY(12)	**Einzigartiger** IV pro Schüler.	Verhindert Mustererkennung (gleicher Name = gleicher Ciphertext).
real_name_name_hash_index	VARCHAR(64)	HMAC-SHA256 Hash des Namens.	Erlaubt Duplikatsprüfung. Da es ein Einweg-Hash ist (One-Way), ist keine Rückrechnung auf den Namen möglich (Pre-Image Resistance).

5. Bedrohungsmodell & Risikobetrachtung

Szenario A: Datenbank-Leak (SQL Dump wird gestohlen)

- **Folge:** Angreifer besitzt nur binären "Datenmüll" und IVs.
- **Schutz:** Ohne das *Klarnamenpasswort* der Lehrkräfte ist eine Entschlüsselung (Brute-Force) aufgrund von AES-256 faktisch unmöglich.

Szenario B: Admin-Zugriff (Insider Threat)

- **Folge:** Ein Datenbank-Admin des ISQ-Portals greift direkt auf die Tabellen zu.
- **Schutz:** Identisch zu Szenario A. Zero-Knowledge garantiert, dass auch privilegierte Administratoren keine Einsicht haben.

Szenario C: Passwort-Verlust

- **Folge:** Lehrkraft vergisst Passwort.
- **Lösung:** Wiederherstellung nur über das extern gesicherte **Klarnamenzertifikat** (physischer Ausdruck/Datei im Safe der Schulleitung) möglich. Ohne dieses Backup sind die Daten verloren (Design for Security).

Szenario D: XSS (Cross-Site Scripting)

- **Risiko:** Ein Angreifer injiziert Schadcode in den Browser der Lehrkraft, während diese eingeloggt ist.
- **Schutz:** Das ISQ-Portal setzt strikte **Content Security Policies (CSP)** ein und verwendet HttpOnly / Secure Flags für Session-Cookies, um das Risiko von XSS zu minimieren. Der Master-Key liegt nur im flüchtigen RAM und ist schwieriger zu exfiltrieren als persistente Daten.

6. Fazit

Die Implementierung im ISQ-Portal erfüllt höchste Anforderungen an den Datenschutz ("Privacy by Design"). Durch die konsequente Ende-zu-Ende-Verschlüsselung verbleibt die Datenhoheit vollständig bei den Schulen. Eine Kenntnisnahme der Klarnamen durch das ISQ oder Dritte ist technisch ausgeschlossen.