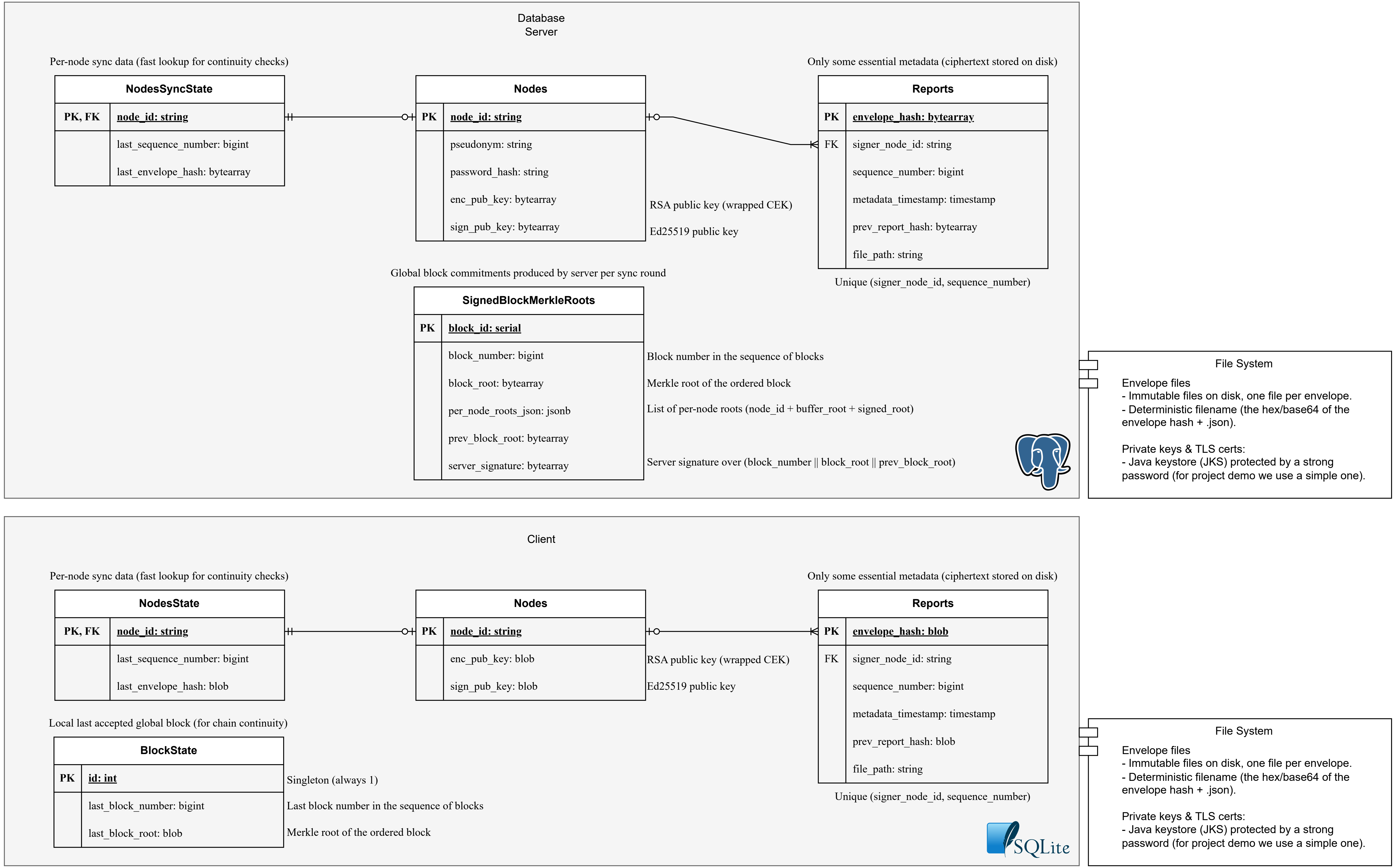

Database
Server

Per-node sync data (fast lookup for continuity checks)

	NodesSyncState
PK, FK	node_id: string
	last_sequence_number: bigint
	last_envelope_hash: bytearray

	Nodes
PK	node_id: string
	pseudonym: string
	password_hash: string
	enc_pub_key: bytearray
	sign_pub_key: bytearray

RSA public key (wrapped CEK)

Ed25519 public key

Only some essential metadata (ciphertext stored on disk)

	Reports
PK	envelope_hash: bytearray
FK	signer_node_id: string
	sequence_number: bigint
	metadata_timestamp: timestamp
	prev_report_hash: bytearray
	file_path: string

Unique (signer_node_id, sequence_number)

Global block commitments produced by server per sync round

	SignedBlockMerkleRoots
PK	block_id: serial
	block_number: bigint
	block_root: bytearray
	per_node_roots_json: jsonb
	prev_block_root: bytearray
	server_signature: bytearray

Block number in the sequence of blocks

Merkle root of the ordered block

List of per-node roots (node_id + buffer_root + signed_root)

Server signature over (block_number || block_root || prev_block_root)

File System

Envelope files
- Immutable files on disk, one file per envelope.
- Deterministic filename (the hex/base64 of the envelope hash + .json).

Private keys & TLS certs:
- Java keystore (JKS) protected by a strong password (for project demo we use a simple one).

Client

Per-node sync data (fast lookup for continuity checks)

	NodesState
PK, FK	node_id: string
	last_sequence_number: bigint
	last_envelope_hash: blob

	Nodes
PK	node_id: string
	enc_pub_key: blob
	sign_pub_key: blob

RSA public key (wrapped CEK)

Ed25519 public key

Only some essential metadata (ciphertext stored on disk)

	Reports
PK	envelope_hash: blob
FK	signer_node_id: string
	sequence_number: bigint
	metadata_timestamp: timestamp
	prev_report_hash: blob
	file_path: string

Unique (signer_node_id, sequence_number)

Local last accepted global block (for chain continuity)

	BlockState
PK	id: int
	last_block_number: bigint
	last_block_root: blob

Singleton (always 1)

Last block number in the sequence of blocks

Merkle root of the ordered block

File System

Envelope files
- Immutable files on disk, one file per envelope.
- Deterministic filename (the hex/base64 of the envelope hash + .json).

Private keys & TLS certs:
- Java keystore (JKS) protected by a strong password (for project demo we use a simple one).

Envelope (what nodes store/send)

```
{
 "metadata": {
  "report_id": "abc123",
  "metadata_timestamp": "2025-10-28T12:00:00Z",
  "report_creation_timestamp": "2025-10-28T10:00:00Z",
  "sequence": 42,
  "prev_hash": "b64url(SHA256(previous_envelope))",
  "signer": { "kid": "nodeA", "alg": "Ed25519" }
 },

 "key_enc": {
  "wrap_alg": "RSA-OAEP-SHA256",
  “key_per_node”: [
     {“node”: “self”, "wrapped_cek": "b64url(...)"}, -> because this node still needs to decrypt it
     {“node”: “nodeB”, "wrapped_cek": "b64url(...)"},
     {“node”: “nodeC”, "wrapped_cek": "b64url(...)"},
  ]
 },

 "report_enc": {
  "alg": "AES-256-GCM",
  "nonce": "b64url(encNonce)",
  "ciphertext": "b64url(...)",
  "tag": "b64url(...)"
 }
}
```

Inner plaintext payload (encrypted with CEK)

```
{
 "report": {
  "report_id": "abc123",
  "report_creation_timestamp": "2025-10-28T12:00:00Z",
  "reporter_pseudonym": "shadow_fox",
  "content": {
   "suspect": "john_doe",
   "description": "Alleged involvement in organized crime",
   "location": "Tokyo, Japan"
  },
  "version": 1,
  "status": "pending_validation"
 },
 "signature": "b64url(Ed25519(( report || metadata )))"
}
```