Capitulo 2 - Billeteras Deterministas por Licencia (BDL)

Índice formal — Marco metodológico y especificación académica (alcance exclusivo: billetera de licencias)

Diseñado por LAEV Blockchain 🦩 Autor: Lerry Alexander Elizondo Villalobos

2. Definiciones formales y glosario técnico

2.1 BDL (Billetera Determinista por Licencia): definición normativa

Una **Billetera Determinista por Licencia (BDL)** es un modelo especializado de billetera criptográfica que incorpora un mecanismo de licenciamiento para controlar y auditar su uso en un conjunto finito de dispositivos autorizados. En términos normativos, una BDL se define como una billetera *no-custodial* de activos digitales (por ejemplo, criptomonedas)

cuyas claves se generan de forma determinista a partir de un seed, pero cuya **operatividad** está regida por una licencia criptográfica. Esto significa que el derecho a usar las claves derivadas (realizar transacciones, firmar operaciones, etc.) está condicionado a poseer una licencia válida y vigente. En una BDL, cada licencia impone límites y políticas técnicos verificables – como el número máximo de dispositivos concurrentes – y todas las acciones relevantes quedan registradas en un *ledger* inmutable de eventos. Esta combinación de **determinismo criptográfico** y **control por licencia** permite mitigar la copiabilidad inherente de las semillas privadas, garantizando que una misma billetera (un mismo conjunto de claves) solo pueda ser utilizada en hasta *N* dispositivos aprobados bajo la licencia correspondiente. En esencia, BDL es una clase formal de billetera que añade una capa de restricción y seguimiento a nivel de arquitectura, sin depender de la mera confianza en el usuario, estableciendo una correspondencia controlada entre una identidad de licencia y sus dispositivos vinculados.

Desde una perspectiva criptográfica y de seguridad, la BDL conserva las propiedades fundamentales de autenticidad y privacidad de una billetera determinista convencional, pero añade nuevas garantías de **no-repudio y trazabilidad** en el uso. Cada operación crítica en una BDL (como autorizar un nuevo dispositivo o ejecutar una transacción) está supeditada a verificaciones adicionales de licencia, y tales eventos quedan **documentados criptográficamente** en el registro de licencias. Esto aporta **auditabilidad** independiente: terceros pueden verificar de forma determinista si se ha respetado la política de N dispositivos y si no ha habido usos fuera de licencia. Arquitectónicamente, una BDL se compone de varios elementos formales (Licencia, DeviceID, ledger de eventos, anclajes externos, etc., definidos más adelante) que interactúan para asegurar que las restricciones de licencia se *enfuerzan* efectivamente (*enforced*) a nivel técnico. La definición normativa de BDL, por tanto, abarca este conjunto de reglas y componentes: es una billetera determinista regida por una licencia digital que impone un cupo de dispositivos y políticas de uso, respaldada por mecanismos criptográficos de verificación y evidencia. En síntesis, una BDL eleva el estándar de control en billeteras criptográficas al introducir un régimen de licenciamiento formal que es verificable públicamente y está intrínsecamente ligado al funcionamiento de la billetera.

2.2 Licencia: identidad criptográfica, cupo operativo, política y régimen de vigencia

En el contexto de BDL, una **Licencia** es la entidad criptográfica central que representa y encapsula los derechos de uso de una billetera determinista. Una licencia actúa como una credencial digital que autoriza a un usuario (o entidad) a operar la billetera bajo ciertas condiciones predefinidas. Técnicamente, la licencia posee varios atributos formales fundamentales que definen su alcance y parámetros:

- **Identidad criptográfica (LicenseID)**: Cada licencia se identifica de manera única mediante un identificador criptográfico derivado de sus datos iniciales. Este *LicenseID* típicamente se construye aplicando una función hash segura sobre una combinación de elementos de la licencia (por ejemplo, la clave pública principal de la licencia, metadatos de emisión y un código de versión), generando un compromiso único. La identidad criptográfica permite **verificar la autenticidad** de la licencia de

forma pública: cualquier evento o transacción asociada a la licencia referenciará este ID, y solo quien posea las claves privadas correspondientes puede legítimamente activar o modificar esa licencia. En términos prácticos, el LicenseID funciona como la "matrícula" o identificador global de la licencia en el sistema, usado para indexar eventos en el ledger y para proveer seudonimato (es decir, no contiene información personal sino un hash verificable).

- **Cupo operativo (límite de dispositivos)**: El cupo operativo es la cuota máxima de dispositivos autorizados simultáneamente bajo esta licencia. Es un parámetro entero (N) definido en el momento de la emisión (evento GENESIS) que indica cuántos **DeviceID activos** puede tener vinculados la licencia a la vez. Por ejemplo, una licencia puede establecer un cupo de 3 dispositivos, significando que solo hasta tres dispositivos distintos podrán estar operativos bajo esa billetera licenciada. Este cupo es **enforceable** a nivel técnico: el sistema de BDL no permite *más* activaciones que las estipuladas, y cualquier intento de exceder el límite N desencadenará mecanismos de alerta o denegación (ver "Enforceability" más abajo). El cupo operativo refleja la política comercial u operativa deseada (p. ej., licencias individuales vs. familiares) y queda registrado de forma inmutable en la licencia (aunque puede modificarse posteriormente mediante eventos de ROTATE si la gobernanza así lo decide, por ejemplo para ampliar el número de dispositivos permitidos).

- **Política**: Bajo este término se agrupan todas las reglas adicionales y configuraciones que rigen el uso de la billetera bajo licencia. La política puede incluir restricciones técnicas como por ejemplo: límites de frecuencia de transacciones por dispositivo, requerimientos de attestation (solo dispositivos que presenten cierta evidencia de hardware pueden activarse), georrestricciones, umbrales de alerta para suspender automáticamente la licencia, etc. Estas políticas se definen en la metadata de la licencia durante su génesis o mediante actualizaciones posteriores, y son **interpretadas y aplicadas** tanto por los clientes (software de la billetera) como por los servicios de verificación. En esencia, la política establece qué está permitido y qué no bajo la licencia, complementando el cupo numérico con condiciones cualitativas. Todas las políticas declaradas forman parte de las reglas de validación determinística: un evento que vulnere una política (por ejemplo, intento de activación en un dispositivo no elegible) será rechazado o generado como ALERT según corresponda. La claridad de la política es crucial para que la *verdad operativa* del sistema sea consistente: las reglas de licencia deben ser formalizadas de tal manera que puedan ser comprobadas automáticamente por cualquier observador independiente.

- **Régimen de vigencia**: Toda licencia tiene asociado un régimen temporal que define su período de validez. Esto puede abarcar licencias *perpetuas* (sin expiración por tiempo, válidas hasta una revocación explícita), licencias *temporales* con fecha de expiración (tras la cual la billetera queda inoperativa a menos que se renueve o extienda mediante un evento autorizado), o incluso licencias condicionadas a eventos (por ejemplo, vigencia mientras se mantenga un contrato de servicio externo). El régimen de vigencia se registra en el evento GENESIS en forma de timestamps de inicio y fin de validez o parámetros de expiración (p. ej., "válido por 1

año desde activación"). Operacionalmente, el software de la billetera debe verificar que la licencia no esté expirada antes de permitir operaciones. Asimismo, la vigencia puede interactuar con el ledger: una expiración podría desencadenar un evento automático de suspensión o requerir un evento de **re-emisión** para continuar operando. En términos de seguridad, un régimen de vigencia claro previene el uso indebido de licencias antiguas o comprometidas a largo plazo, al forzar renovaciones que pueden reevaluar las condiciones de seguridad.

En conjunto, estos cuatro aspectos definen una licencia BDL de manera integral. La **licencia** es así un objeto criptográfico-portable que confiere una identidad *única*, unos derechos limitados en cantidad (dispositivos) y calidad (políticas), y un período de validez determinado. Todas estas propiedades están respaldadas por criptografía (firmas digitales del emisor, hashes en el ID, etc.) y son *vinculantes* dentro del sistema: ni los usuarios ni los operadores pueden alterar los límites o la identidad de una licencia sin generar eventos públicos en el ledger. De este modo, la licencia actúa como el contrato técnico que articula la relación entre el usuario y la billetera determinista, sirviendo a la vez como ancla de confianza (por su identidad verificable) y como mecanismo de control (por sus restricciones de cupo, política y tiempo).

2.3 Dispositivo: identidad criptográfica verificable (DeviceID) y evidencias opcionales de hardware (attestation)

En el modelo BDL, un **Dispositivo** se define como una entidad cliente (por ejemplo, un teléfono móvil, computadora, hardware wallet o entorno de ejecución) que instala y utiliza la billetera bajo una licencia determinada. Cada dispositivo autorizado en el sistema posee una **identidad criptográfica verificable**, denominada *DeviceID*, y puede estar acompañado de evidencias criptográficas de hardware (*attestation*) que refuerzan su singularidad y confianza.

Identidad criptográfica del dispositivo (DeviceID): Para cada dispositivo que desea activarse bajo una licencia, se genera un par de claves criptográficas único (normalmente un par de clave pública/privada de curva elíptica u otro esquema asimétrico seguro). El *DeviceID* es típicamente la huella digital derivada de la clave pública del dispositivo (por ejemplo, un hash de la clave pública concatenado con el identificador de la licencia, para asegurar unicidad global). Esta identidad funciona como el identificador del dispositivo en el registro de licencias y en todos los protocolos de autorización: cuando un dispositivo se activa, su DeviceID queda registrado en un evento ACTIVATE firmado, estableciendo criptográficamente que ese dispositivo posee la correspondiente clave privada. La **verificabilidad** del DeviceID radica en que terceros pueden desafiar al dispositivo a firmar un nonce o mensaje; si el dispositivo puede producir una firma válida con la clave privada asociada a su DeviceID (clave pública), se confirma su identidad. De esta forma, cada dispositivo es una entidad *autónoma y autenticable* dentro del sistema, lo que evita confiar en identificadores meramente declarativos o fáciles de falsificar. Además, el DeviceID está diseñado para ser estable por dispositivo pero no trivialmente clonable: idealmente, las

claves privadas no salen del dispositivo, impidiendo que copiar la seed de la billetera suponga automáticamente duplicar el DeviceID en otro aparato.

Evidencias de hardware (attestation) opcionales: Como capa adicional de seguridad, BDL soporta la presentación de *atestados de hardware* que den fe de la integridad del dispositivo y de la protección de su clave privada. La attestation es un mecanismo criptográfico por el cual el dispositivo provee pruebas firmadas de ciertas propiedades de su entorno. Por ejemplo, dispositivos Android compatibles pueden emplear **Key Attestation** de Google SafetyNet o Play Integrity, donde el sistema operativo genera un certificado firmado por Google indicando que la clave privada del DeviceID se encuentra en un **Keystore respaldado por hardware** (Trusted Execution Environment o Secure Element) y que el dispositivo no está rooteado ni comprometido. De igual manera, un hardware wallet puede presentar un certificado del fabricante asegurando que el dispositivo es genuino y que la clave nunca ha salido del elemento seguro. Estas evidencias, cuando existen, se adjuntan durante el proceso de activación del dispositivo y quedan registradas (o referenciadas) en el evento ACTIVATE correspondiente.

La presencia de attestation permite **clasificar niveles de confianza** para cada DeviceID. Un dispositivo con attestation válida de hardware tendrá un nivel de confianza superior, ya que se tiene alta certeza de que su clave privada es inexportable y está protegida contra manipulaciones (lo que dificulta enormemente su clonación o suplantación). Por el contrario, dispositivos sin attestation (p.ej. una computadora de escritorio sin TPM utilizado, o un teléfono con sistema operativo modificado) serán tratados con mayor precaución: el sistema podría aplicar políticas más estrictas para ellos (como límites de operaciones, mayor monitoreo de alertas, etc.), dado que la ausencia de evidencia hace teóricamente posible que su clave pudiera copiarse a otro equipo. En todos los casos, el DeviceID sigue siendo el factor principal de identidad; la attestation simplemente añade información sobre **las garantías de unicidad** de ese DeviceID respaldadas por hardware.

Desde el punto de vista operativo, registrar un dispositivo implica: (1) generar localmente las claves del DeviceID dentro del dispositivo, idealmente en un módulo seguro; (2) obtener la attestation de hardware si el dispositivo la soporta; (3) enviar la solicitud de activación al sistema de licencias, incluyendo la clave pública y evidencia; y (4) recibir la confirmación mediante un evento ACTIVATE inscrito en el ledger, firmado apropiadamente. Una vez activo, el dispositivo debe conservar su clave privada de forma segura y utilizarla para autenticarse en cada operación. El diseño de DeviceID y attestation en BDL busca asegurar que cada dispositivo sea único, identificable criptográficamente y difícil de falsificar, proporcionando la base para hacer cumplir la restricción de N dispositivos por licencia de forma sólida.

2.4 Registro de licencias: *ledger* basado en *event-sourcing* y verificación determinística

El **Registro de licencias** es la columna vertebral de la arquitectura BDL, actuando como una base de datos distribuida e inmutable de todos los eventos relevantes al ciclo de vida de las licencias y dispositivos. Este registro se implementa siguiendo el patrón de **event-sourcing**, lo que implica que no almacena estados finales directamente, sino que

acumula un historial ordenado de eventos inmutables a partir del cual el estado de cada licencia puede derivarse de manera determinística. En otras palabras, el *ledger* consiste en una secuencia cronológica de eventos (GENESIS, ACTIVATE, DEACTIVATE, etc., definidos en 2.5) y **la verdad del sistema se reconstruye procesando secuencialmente dichos eventos**.

En términos formales, el registro es un conjunto de *streams* de eventos donde cada stream corresponde típicamente a una licencia (identificada por su LicenseID). Cada nuevo evento que afecta a una licencia se agrega al final de su stream, conservando así un historial completo desde su génesis hasta el presente. Dado que los eventos son inmutables y están firmados, este enfoque garantiza propiedades de **integridad y auditabilidad** muy fuertes: cualquier cambio de estado (por mínimo que sea) queda registrado con evidencia criptográfica de quién lo autorizó y cuándo ocurrió. El estado actual de una licencia (por ejemplo, cuántos dispositivos activos tiene, si está revocada o no, etc.) se obtiene aplicando una función determinista de reducción sobre todos sus eventos en orden. Gracias a esto, es posible realizar **verificación independiente**: cualquier parte puede tomar la lista de eventos de una licencia (obtenida de nodos de la red o de un repositorio público), verificar las firmas y hashes de encadenamiento, y recalcular exactamente el estado final, sabiendo que obtendrá el mismo resultado que el operador original siempre que la secuencia de eventos sea la misma. No existe ambigüedad en el cálculo del estado, lo cual es fundamental para la consistencia global del sistema.

La **verificación determinística** se logra mediante varias medidas. Primero, todos los eventos llevan un identificador único (p.ej., un hash de su contenido) y referencian de manera explícita el LicenseID afectado; además, pueden incluir un número de secuencia o referencia al hash del evento previo de ese LicenseID, creando una cadena criptográfica dentro de cada stream. Esto previene la inserción retroactiva o reordenamiento de eventos: cualquier alteración del historial invalidaría los encadenamientos hash y sería detectada. Segundo, cada evento está **firmado digitalmente** por la entidad autorizada para realizar esa acción (ya sea la clave de gobernanza de la licencia, la clave del dispositivo, o una clave de auditoría según el tipo de evento). Estas firmas garantizan autenticidad y evitan la introducción de eventos foráneos por actores no autorizados. Tercero, las reglas de validación del ledger (ver sección 9.2 del documento) definen claramente qué eventos son válidos dada la secuencia previa – por ejemplo, un evento ACTIVATE solo es aceptado si el número de dispositivos activos antes +1 no excede el cupo, un evento DEACTIVATE solo es válido si el dispositivo a dar de baja estaba activo, etc. Dichas reglas aseguran que **el registro no entre en estados inválidos**; cualquier nodo indexador o verificador al aplicar los eventos uno tras otro comprobará cada regla, rechazando aquellos eventos que violen las políticas o la lógica de la licencia. Como resultado, el ledger actúa como un árbitro objetivo: si dos partes tienen la misma secuencia de eventos válidos, forzosamente concordarán en el estado final, sin necesidad de confianza mutua.

Otra característica importante del registro de licencias es su **naturaleza descentralizada o replicada**. Aunque el diseño BDL no requiere necesariamente de una blockchain pública con consenso por prueba de trabajo (dado que las licencias pueden operar en un contexto permisado), sí favorece la existencia de múltiples réplicas independientes (indexadores) que almacenan y retransmiten los eventos. Esto proporciona redundancia (alta disponibilidad) y evita la dependencia en un único servidor central para la verificación. Cualquier auditor

externo puede correr un nodo que escuche los eventos nuevos, los valide y mantenga una copia local del ledger. Asimismo, mediante mecanismos de **anclaje criptográfico en Bitcoin** (definido en 2.6), se refuerza la inmutabilidad del registro incluso frente a colusión de nodos maliciosos. El resultado es un ledger de eventos que es prácticamente inmutable (apend-only), públicamente verificable y consistente entre todos los participantes.

En resumen, el registro de licencias basado en event-sourcing proporciona una **base de datos histórica completa** y fidedigna para BDL. Cada licencia y sus dispositivos asociados tienen en este ledger su rastro indeleble: creación, activaciones, desactivaciones, alertas, etc. Este enfoque aporta las ventajas de una bitácora financiera tradicional pero con garantías criptográficas modernas: *transparencia auditiva*, *trazabilidad total* y *reproducibilidad del estado*. La confianza en el sistema no depende de secretos ocultos sino de la robustez criptográfica del registro, donde la frase “lo que no está en el ledger no ha ocurrido” sintetiza la filosofía de diseño.

2.5 Tipología de eventos: GENESIS, ACTIVATE, DEACTIVATE, ROTATE, ALERT

El sistema BDL define un conjunto estricto de **tipos de eventos** para describir cualquier cambio de estado o condición relevante en el ciclo de vida de una licencia. Cada evento representa una transición atómica y auditable, con un formato específico y semántica definida. A continuación se detallan los tipos de eventos formales soportados y su significado:

- **Evento GENESIS:** Es el evento fundacional de una licencia determinista. Mediante GENESIS se crea una nueva licencia en el registro, quedando así definido su estado inicial. Este evento incluye los **parámetros normativos iniciales** de la licencia: el *LicenseID* (derivado criptográficamente de los datos del evento), la clave pública principal o llaves de gobernanza asociadas, el cupo operativo autorizado (N dispositivos máximos), la política inicial (reglas de uso, restricciones especiales) y el régimen de vigencia (por ejemplo, “licencia válida hasta el 31/12/2025” o “licencia perpetua”). GENESIS suele estar firmado por la entidad emisora o autoridad de licenciamiento, lo que asegura que la licencia es legítima y aprobada bajo las condiciones acordadas. Arquitectónicamente, este evento marca el comienzo de un nuevo stream en el ledger: a partir de GENESIS, la licencia puede transicionar a otros estados mediante los eventos subsiguientes. En la **interpretación operacional**, GENESIS equivale a emitir una nueva billetera licenciada (similar a emitir un certificado o un contrato de licencia en sistemas tradicionales). Por diseño, solo puede existir un GENESIS por LicenseID, y es inmutable: si se requieren cambios fundamentales, se deberán hacer vía ROTATE o emitir una licencia nueva.

- **Evento ACTIVATE:** Este evento representa la **activación (alta) de un dispositivo** bajo una licencia existente. Ocurre cuando un usuario instala la billetera en un nuevo dispositivo y solicita asociarlo a su licencia. El evento ACTIVATE contendrá el *DeviceID* del dispositivo (clave pública o su hash) y opcionalmente datos de attestation o metadatos del dispositivo (por ejemplo, tipo de plataforma, nivel de confianza). Un ACTIVATE válido aumenta en uno el conteo de dispositivos activos

de la licencia, y por ende el sistema debe verificar que tras este evento no se exceda el cupo máximo permitido – de lo contrario, el evento sería rechazado por las reglas de validación. Normalmente, el evento estará firmado por la clave privada de la **licencia/usuario** que autoriza agregar ese dispositivo **y** también por la clave privada del dispositivo mismo (demostrando posesión). Esto asegura autenticidad doble: solo el titular de la licencia puede permitir nuevas activaciones, y solo un dispositivo que posea la clave privada correspondiente puede darse de alta. Tras un ACTIVATE exitoso, el DeviceID pasa a formar parte del estado activo de la licencia, habilitando a ese dispositivo a operar la billetera (transaccionar, ver saldos, etc.) según la política. En caso de dispositivos duplicados o intentos de reuso de DeviceID, la lógica de eventos lo impediría puesto que un mismo DeviceID no puede activarse dos veces para la misma licencia y, si se intentara con otro LicenseID, los registros de attestation (si existen) o la detección de colisiones lo señalarían. El evento ACTIVATE queda registrado junto con una marca temporal, brindando evidencia pública de cuándo y qué dispositivo fue añadido.

- **Evento DEACTIVATE:** Es el evento opuesto al anterior, indicando la **desactivación (baja) de un dispositivo** previamente activo en la licencia. Un DEACTIVATE se emite, por ejemplo, cuando el usuario decide desvincular un dispositivo (p. ej. vende su teléfono o lo envía a soporte técnico), o cuando por motivos de seguridad se desea dar de baja un DeviceID (p. ej. dispositivo perdido o comprometido). El evento especifica cuál DeviceID está siendo removido y suele ir firmado por la parte con autoridad para ello: típicamente el titular de la licencia, aunque podría ser también emitido automáticamente por una política de seguridad (por ejemplo, ante repetidas alertas se podría disparar una desactivación forzosa de ese DeviceID). Al aplicarse un DEACTIVATE en el ledger, el dispositivo correspondiente pasa al estado *inactivo* para esa licencia, liberando así un cupo. A nivel de reglas, se invalida cualquier futuro intento de uso desde ese DeviceID (los clientes deben consultar el estado y abstenerse de operar si su DeviceID fue revocado). Asimismo, si se intentara reactivar el mismo dispositivo luego, debería generar un nuevo DeviceID (por diseño, un DeviceID dado no suele reactivarse; un nuevo par de claves implicaría un nuevo identificador). Desde la perspectiva de seguridad, DEACTIVATE sirve para **limitar daños** en caso de dispositivos extraviados o sospechosos: la licencia puede invalidarlos rápidamente y esa acción queda auditada. Operacionalmente, también permite la rotación natural de dispositivos del usuario (por ejemplo, cuando se cambia de dispositivo principal, se desactiva el antiguo para activar el nuevo, manteniendo el límite N). Este evento, al igual que ACTIVATE, modifica el estado de cuenta de dispositivos de forma determinista: decrementa el contador de activos y elimina la referencia al DeviceID dado.

- **Evento ROTATE:** Este evento refleja cambios o actualizaciones de los **parámetros de la licencia** durante su vigencia, sin necesidad de revocarla completamente. Bajo ROTATE se engloban operaciones como: **rotación de llaves de gobernanza**, actualización de la política de licencia, modificación del cupo de dispositivos, o extensión del período de vigencia. Por ejemplo, si la clave privada principal asociada a la licencia (la del LicenseID o la de firma de eventos) se ve comprometida o simplemente por higiene criptográfica se desea rotar, se emitiría un evento ROTATE proporcionando una nueva clave pública válida en adelante (y firmado por la antigua

clave o por un conjunto de claves de respaldo, según el esquema de gobernanza). De igual modo, si el usuario adquiere una ampliación de licencia para permitir más dispositivos, un ROTATE podría incrementar el cupo operativo de N=3 a N=5, actualizando ese campo en el estado. Cada ROTATE define su *payload* específico (ej. nuevo valor de cierto parámetro) y está firmado por las llaves de gobernanza competentes con un **quórum** suficiente (ver sección 2.8). En la aplicación de eventos, ROTATE puede considerarse un evento que **transiciona el estado** sin afectar dispositivos individuales directamente, pero potencialmente alterando las condiciones bajo las cuales ocurrirán futuras activaciones, alertas, etc. La existencia de ROTATE es vital para dotar a la licencia de flexibilidad y capacidad de respuesta ante contingencias, todo ello sin invalidar la licencia entera. Por supuesto, cada ROTATE queda registrado y anclado, de forma que cualquier auditor puede ver el historial de cambios de políticas o llaves a lo largo del tiempo, manteniendo la transparencia. Las reglas de validación controlan que un ROTATE no viole políticas (por ejemplo, no se podría reducir el cupo por debajo del número de dispositivos actualmente activos sin primero dar de baja algunos, o no se puede rotar claves sin la firma autorizada).

- **Evento ALERT:** Este tipo de evento se genera cuando ocurre una **situación anómala o de interés de seguridad** relacionada con la licencia o sus dispositivos. A diferencia de los otros eventos que son, en general, iniciados por el usuario o la administración de la licencia, un ALERT es típicamente desencadenado automáticamente por el sistema de monitoreo al detectar un intento o condición fuera de la política establecida. Ejemplos de situaciones que producirían un ALERT incluyen: un dispositivo no autorizado que intenta usar la billetera (p. ej. se observó una firma de transacción desde un DeviceID desconocido, indicando posible copia de seed), un exceso de intentos fallidos de autenticación, la presencia de más dispositivos queriendo operar de los permitidos, o cualquier violación de las reglas de uso (como transacciones fuera de horario si la política lo restringe, etc.). El evento ALERT contiene información descriptiva codificada: un **código de razón** que clasifica el tipo de alerta (ver 2.7) y una **medida de severidad** asociada. Puede además incluir metadatos técnicos (por ejemplo, el hash de una transacción rechazada, el DeviceID que causó la alerta, ubicación IP, etc., dependiendo de la naturaleza). Los ALERT no cambian el estado operativo de la licencia por sí mismos (no agregan ni quitan dispositivos), pero sí impactan en su **estado de seguridad**: quedan registrados para crear un historial de incidencias. Múltiples alertas pueden escalar la situación; por ejemplo, acumulación de severidad podría llevar a suspender temporalmente la licencia automáticamente mediante políticas (esto se implementaría quizás como un cambio de estado interno o un evento de suspensión en la lógica de más alto nivel, pero la mera ocurrencia de ALERTs ya sirve como detonante). Un ALERT debe ser **firmado** por la entidad que lo emite: en muchos casos sería un módulo de auditoría o el propio servicio de indexación que detectó la anomalía. Para garantizar confiabilidad, estos módulos de monitorización podrían tener sus propias claves de firma autorizadas bajo la gobernanza del sistema. El registro de un ALERT en el ledger proporciona evidencia irrefutable de que tal incidente ocurrió en un momento dado – lo cual es crucial para la **responsabilidad y el análisis forense**. Además, dado que los ALERT afectan a la puntuación de riesgo de la licencia, su presencia en el ledger permite a cualquier interesado (el usuario, el

proveedor, auditores externos) evaluar la confiabilidad de esa instancia de billetera con solo revisar su historial de alertas público.

En conjunto, esta tipología de eventos cubre todo el espacio de estados y transiciones relevantes para una licencia BDL. Cada evento está cuidadosamente diseñado para ser **ortogonal** a los demás (cada uno cubre un tipo de cambio distinto, evitando ambigüedad) y para llevar la información mínima suficiente que permita su verificación independiente y su anclaje criptográfico. El diseño por eventos garantiza que cualquier modificación en la realidad (un nuevo dispositivo, un dispositivo removido, un cambio de política, un incidente de seguridad) tenga un correlato directo y estructurado en el ledger, sirviendo al principio de *trazabilidad mínima suficiente*. La **secuencia de eventos** completa constituye la historia viva de la licencia, y cada tipo de evento es un capítulo con significado bien definido dentro de esa historia.

2.6 Anclaje en Bitcoin: compromisos criptográficos, Merkle root y pruebas de inclusión temporal

El **anclaje en Bitcoin** es un mecanismo de seguridad adicional que utiliza la cadena de bloques de Bitcoin (u otra cadena pública robusta) como registro externo de los compromisos criptográficos del ledger de licencias. El objetivo principal es aprovechar la inmutabilidad y sello de tiempo distribuido que ofrece Bitcoin para **garantizar la integridad temporal** del registro de eventos de BDL, creando pruebas imposibles (o extraordinariamente costosas) de falsificar acerca de la existencia de determinados eventos para un momento dado.

Técnicamente, el anclaje se realiza mediante **compromisos criptográficos agregados**. En intervalos regulares o tras acumular cierto número de eventos nuevos, el sistema BDL toma los hashes de esos eventos (por ejemplo, hashes de cada evento reciente o de la raíz de un árbol que los agrupa) y los combina en una estructura de **árbol de Merkle**. Un árbol de Merkle es una estructura en árbol binario donde cada hoja representa el hash de un dato (en este caso, de un evento o bloque de eventos) y cada nodo interno es el hash de la concatenación de los hashes de sus hijos. El resultado final es un único hash, la **raíz de Merkle**, que actúa como un resumen compacto de todo el conjunto de eventos incluidos. Esta raíz de Merkle es, en sí misma, un compromiso criptográfico: si *cualquier* evento de la colección cambiara, incluso en un solo bit, la raíz resultante sería distinta.

Una vez calculada la raíz de Merkle correspondiente a los eventos (por ejemplo, podría ser un conjunto de todos los eventos de las últimas 24 horas, o de un lote de 100 eventos, dependiendo de la frecuencia óptima configurada), se procede a **anclarla en la blockchain de Bitcoin**. Esto comúnmente se logra a través de una transacción Bitcoin especial que lleva la raíz de Merkle incluida en un campo *op_return* o encubierta en una dirección de salida. Al publicar esta transacción y ser confirmada en la blockchain, la raíz de Merkle queda grabada de forma indeleble en un bloque de Bitcoin, heredando las propiedades de seguridad de la red: descentralización, resistencia a censura y a modificación retrospectiva. Dado que cada bloque de Bitcoin tiene marca temporal (timestamp) aproximada y un orden asegurado por proof-of-work, el momento de confirmación sirve como **prueba de inclusión**

temporal. Es decir, cualquier evento cuya huella esté dentro de esa raíz de Merkle se considera *atestiguado* por la blockchain en o antes de la fecha/hora de ese bloque. Para un auditor externo, proporcionar una prueba de que un cierto evento E fue incluido en el anclaje se reduce a proveer el **Merkle proof** correspondiente: la ruta de hashes hermanos desde el hash de E hasta la raíz anclada, junto con la referencia al bloque de Bitcoin donde figura la raíz. Con esa información, el auditor puede recomputar la raíz localmente y compararla con la registrada en Bitcoin; si coinciden, se confirma que E formaba parte del conjunto anclado. Esto permite verificar, sin ambigüedad, que el evento E existía en el ledger BDL a más tardar en el momento en que se minó ese bloque de Bitcoin (dando una garantía de *timestamp* confiable y ampliamente aceptada).

Las **pruebas de inclusión** así obtenidas fortalecen enormemente la confianza en la integridad del sistema: incluso si un adversario controlara todos los servidores de registro de licencias, no podría eliminar o falsificar eventos pasados sin que el desajuste fuera evidente al confrontar con el anclaje público. Para lograr tal falsificación tendría que reescribir también la historia de Bitcoin (reorganizar bloques) con un costo computacional prohibitivo una vez que suficientes confirmaciones han cimentado el bloque. En la práctica, unas pocas confirmaciones hacen la inserción irreversible para cualquier atacante realista. Además, el uso de Merkle trees hace que el proceso sea eficiente: en lugar de anclar cada evento individualmente –lo que sería inviable por costos y volumen de transacciones– se ancla un único hash que representa potencialmente cientos de eventos, logrando economía de escala.

Desde una perspectiva de diseño, la frecuencia y modalidad de anclaje (cada X eventos o cada X unidades de tiempo) se elige equilibrando **coste vs. garantías de frescura**. Un anclaje más frecuente (por ejemplo cada hora) brinda sellos de tiempo más finos pero cuesta más comisiones en la red Bitcoin; un anclaje más espaciado reduce coste pero deja una ventana temporal mayor en la que eventos no estarían todavía protegidos por la cadena externa. En cualquier caso, una vez anclados, esos eventos quedan protegidos retrospectivamente. Algunos sistemas optan por anclar regularmente (ej. diario) y adicionalmente ofrecer mecanismos on-demand (por ejemplo usando servicios como OpenTimestamps) para sellar ciertos eventos críticos de inmediato. BDL puede adoptar una estrategia similar para eventos de alerta de alta severidad o revocaciones inmediatas, dándoles prioridad en el anclaje.

Es importante notar que el anclaje es unidireccional: la blockchain de Bitcoin nunca "conoce" el contenido de los eventos, solo ve un hash, por lo que **no se filtra información sensible**. Esto cumple con el principio de minimización: se obtiene seguridad adicional sin divulgar datos privados on-chain. Por último, gracias al anclaje, el sistema puede proveer **evidencia externa** de su correcto funcionamiento para terceros independientes, como auditores regulatorios o partes interesadas, sin necesidad de confiar en la palabra del operador de la billetera. Cualquier discrepancia entre el ledger interno y los anclajes públicos sería prueba de manipulación. Así, el anclaje en Bitcoin actúa como la última capa de defensa y confianza, cimentando la **integridad pública verificable** del ecosistema BDL.

2.7 Sistema de alertas: métricas, severidad acumulada y códigos de razón

El **sistema de alertas** de BDL conforma un subsistema de seguridad y monitoreo continuo cuyo fin es detectar usos indebidos o intentos de violación de la política de licencias, cuantificar su gravedad y posibilitar respuestas automáticas o administrativas. Este sistema se apoya en tres conceptos clave: las métricas monitoreadas, la severidad acumulada de las alertas y los códigos de razón que categorizan cada incidencia.

Métricas monitoreadas: Para generar alertas, primero se definen ciertos eventos o condiciones que deben ser vigilados activamente. Estas *métricas* representan comportamientos atípicos o prohibidos según la política de la licencia. Ejemplos de métricas podrían ser: el número de intentos de activación de dispositivos por encima del cupo (p.ej., un cuarto intento de activar cuando el límite es 3), intentos de operar con un DeviceID no registrado, frecuencia excesiva de operaciones desde diferentes IPs o ubicaciones (posible compartición de credenciales), discrepancias horarias significativas en transacciones (posible manipulación del entorno del dispositivo), o detección de clonación de la billetera (dos dispositivos reportando el mismo seed). Estas métricas se instrumentan en el software de la billetera y en los servidores de indexación de eventos: cada vez que ocurre una acción relevante, el sistema comprueba si viola alguna condición. Si lo hace, en lugar de permitir la acción, **se gatilla una alerta**.

Severidad acumulada: No todas las alertas tienen la misma importancia; por ello, a cada tipo de alerta se le asigna un nivel de **severidad** o puntuación de riesgo. Por ejemplo, un intento aislado de iniciar sesión con un DeviceID inválido podría considerarse de baja severidad (posible error de usuario), mientras que la detección de dos DeviceID distintos usando la misma clave privada (lo que sugiere copia del seed) sería de severidad crítica. La severidad puede representarse numéricamente (p.ej., de 1 a 10) o en categorías (baja, media, alta, crítica), y queda registrada en el evento ALERT. El sistema acumula estas severidades a lo largo del tiempo para cada licencia, manteniendo un **score de riesgo** que es esencialmente la suma ponderada de alertas recientes. Se pueden implementar *ventanas temporales* de acumulación (por ejemplo, considerar las alertas de los últimos 30 días) para que con el tiempo, sin nuevas incidencias, el score vaya reduciéndose (simulando una recuperación de confianza). Esta acumulación de severidad sirve para detectar **patrones persistentes de ataque**: una sola alerta leve no debería provocar pánico, pero muchas alertas leves o una combinación de leves y graves pueden indicar un problema sistémico con la licencia o un intento sostenido de violación. La **severidad acumulada** actúa entonces como un umbral cuantitativo: se pueden fijar niveles en los cuales la licencia entra en estado de sospecha, requeriría revisión manual o desencadenar medidas automáticas (suspensión, refuerzo de autenticación, etc., ver 12.3 del documento).

Códigos de razón (reason codes): Cada alerta generada lleva consigo un código de razón estandarizado que indica la causa o tipo de incidencia detectada. Estos códigos de razón forman un **glosario técnico de incidentes** (listado en el Anexo A del documento) que permite clasificar las alertas de forma uniforme y comprensible tanto para humanos como para sistemas. Por ejemplo, podría existir un código `ALERT01` para “Dispositivo no autorizado intentando transaccionar”, un código `ALERT02` para “Intento de activar dispositivo excediendo cupo”, `ALERT03` para “Posible duplicación de seed detectada”, `ALERT04` para “Múltiples fallas de autenticación consecutivas”, y así sucesivamente. Al emplear códigos en vez de largas descripciones, se facilita que los clientes de billetera muestren mensajes claros al usuario (cada código puede mapearse a una descripción en

varios idiomas), y que sistemas externos de reporte o auditoría automaticen el conteo y análisis de incidentes. Además, los reason codes sirven para integrar BDL con sistemas de *Security Information and Event Management (SIEM)* existentes, ya que estandarizan la notificación de eventos de seguridad.

El **funcionamiento integrado** del sistema de alertas es el siguiente: supongamos que ocurre un evento sospechoso en un dispositivo – por ejemplo, el usuario instala la billetera en un cuarto dispositivo cuando su licencia solo permite tres. Al intentar activarse, el sistema de validación verá que esa activación violaría la política. En lugar de procesarla normalmente, registra un evento ALERT con el código de razón correspondiente ("Cupo excedido"), posiblemente con severidad alta, y rechaza la activación. Este ALERT incrementará la severidad acumulada de la licencia. Si seguidamente el usuario (o un atacante) intenta otras maniobras (quizá intercambiando seeds entre dispositivos), más alertas se irán generando. Llegado cierto punto, si el score acumulado supera un umbral predefinido, la política podría **automáticamente suspender** la licencia (por seguridad, hasta que un administrador la revise) o degradar su funcionalidad (por ejemplo, exigir confirmaciones adicionales para transaccionar). Todo este comportamiento queda regido por configuraciones formales que están definidas dentro de la política de licencia o globalmente en el sistema BDL.

Desde la perspectiva de **seguridad y operatividad**, el sistema de alertas proporciona varias ventajas. Primero, actúa como **detección temprana** de intentos de abuso: incluso si un atacante logra obtener una copia de la seed de la billetera, sus acciones anómalas serán captadas rápidamente (pues tarde o temprano necesitará activar un dispositivo extra, o usar un DeviceID no registrado, etc.). Segundo, crea un registro histórico de incidentes que puede ser analizado para mejorar las defensas: por ejemplo, si ciertas licencias reciben muchas alertas de cierto tipo, podría indicar vectores de ataque comunes o necesidad de educar al usuario. Tercero, las alertas disuaden a los usuarios de violar las condiciones de licencia – sabiendo que cualquier intento quedará registrado y podría invalidar su licencia, se refuerza el cumplimiento voluntario. Finalmente, la presencia de un robusto sistema de alertas permite al ecosistema BDL **mantener la integridad sin imponer custodia estricta**, ya que la vigilancia constante compensa la libertad que el usuario tiene al controlar sus claves privadas. En resumen, métricas bien elegidas, severidades bien calibradas y códigos de razón claros conforman un sistema de alertas capaz de **puntuar el riesgo** de cada licencia de manera objetiva y accionable, elevando significativamente el nivel de seguridad global.

2.8 Gobernanza: llaves, quórum, procesos de revocación y re-emisión

La **gobernanza** dentro de BDL se refiere al conjunto de mecanismos y roles criptográficos encargados de administrar y asegurar el ciclo de vida de las licencias, especialmente en lo relativo a decisiones de alto nivel como revocaciones, modificaciones críticas o re-emisiones de licencias. Dado que BDL busca un equilibrio entre el control por parte de la entidad emisora y la autonomía del usuario final, la gobernanza se implementa mediante esquemas de múltiples claves y quórums de aprobación que distribuyen la confianza y evitan puntos únicos de fallo.

Llaves de gobernanza y quórum: Desde la génesis de cada licencia (evento GENESIS), típicamente se definen una o varias **claves de gobernanza** asociadas. Estas pueden incluir, por ejemplo, la clave privada del emisor o proveedor de la licencia, claves de administradores o contratos inteligentes de control, e incluso la clave del propio usuario titular de la licencia en ciertos casos. A cada clave se le asigna un rol o peso. La gobernanza de la licencia establece un **quórum mínimo** de firmas necesarias para autorizar eventos críticos que afectan a la vida de la licencia, como un ROTATE de claves o una revocación anticipada. Por ejemplo, podría definirse que para revocar definitivamente una licencia se requieren al menos 2 de 3 firmas de un comité (multi-sig 2/3), o que para rotar la clave principal de la licencia se necesita la firma tanto del usuario titular como de la entidad emisora (un esquema de co-firma). Este uso de esquemas *multifirma* (multisig) y de umbrales de firma garantiza que **ninguna entidad individual** pueda tomar acciones unilaterales que comprometan la licencia. A nivel de implementación, un evento sujeto a gobernanza (p. ej., ROTATE que actualiza la política) incluirá en su contenido múltiples campos de firma: los validadores del ledger comprobarán que se hayan adjuntado las firmas requeridas de las claves designadas antes de aceptar el evento. Si falta alguna o no concuerda con el quórum definido, el evento es inválido. Este enfoque de gobierno distribuido añade resiliencia: incluso si una clave de gobernanza es robada o un actor se vuelve malicioso, no podrá por sí solo ni sabotear (revocar) ni alterar (rotar a su conveniencia) la licencia sin consentimiento de los demás guardines. De igual forma, introduce la posibilidad de **roles diferenciados**: por ejemplo, ciertas llaves podrían tener autoridad solo para emitir ALERT manuales o suspender temporalmente una licencia, mientras que otras (o una combinación) se requieren para la revocación permanente. El diseño exacto de llaves y quórum puede adaptarse al caso de uso (más estricto para licencias empresariales, más flexible para usuarios individuales), pero siempre persigue la máxima seguridad con mínima fricción.

Procesos de revocación: La **revocación** de una licencia es el proceso mediante el cual se invalida definitivamente su uso antes de su expiración natural, anulando la autorización para todos los dispositivos asociados. En BDL, la revocación se materializa mediante un evento dedicado (en la sección 14.3 se menciona *Evento REVOCATION*) que, una vez registrado, marca la licencia como *revocada*. Los dispositivos activos podrían recibir esta información en forma de alerta o actualización de estado, tras lo cual deberán impedir nuevas operaciones (efectivamente “apagando” la billetera). Para que un evento de revocación sea aceptado en el ledger, debe cumplir con la política de gobernanza: típicamente, conllevará las firmas de quórum de las llaves de gobernanza, certificando que la decisión fue aprobada por la instancia competente. Una vez revocada, una licencia no puede volver al estado activa (es análoga a revocar un certificado digital: se retira la confianza de forma permanente). Las razones para revocar pueden ser varias: detección de compromiso de la clave maestra o del seed, abuso reiterado de la licencia (múltiples violaciones de alerta severas), fin de contrato con el usuario, etc. Desde un punto de vista operativo, la revocación debe ser comunicada de manera confiable a los usuarios afectados; dado que todo es público en el ledger, cualquier cliente BDL al sincronizar verá el evento REVOCATION y podrá notificar al usuario. Para reforzar la eficacia, el sistema podría además utilizar canales fuera de banda (correo, notificaciones push) en caso de revocaciones críticas, pero el ledger ya proporciona la fuente de verdad. Es crucial resaltar que la revocación en BDL no requiere “confiscar” ninguna clave privada (la billetera sigue siendo no custodial), sino que se basa en retirar la licencia asociada a esas claves. El

usuario aún controlaría técnicamente sus fondos (tiene la seed), pero la interfaz licenciada rehusará usarlos; además, cualquier transacción emitida por fuera del sistema ya no estaría cubierta ni reconocida por la licencia, exponiendo al usuario a posibles repercusiones legales o contractuales si hubiese. En definitiva, la revocación es la **última ratio** de control – raramente usada, pero necesaria para mantener la integridad general del ecosistema ante casos de violación grave o compromisos de seguridad.

Procesos de re-emisión: La **re-emisión** se refiere a la creación de una nueva licencia para reemplazar a otra anterior, típicamente cuando la original ha sido revocada o está por expirar, de manera que el usuario pueda continuar operando con mínimas interrupciones bajo un nuevo contrato. Pensemos en un caso de compromiso: si la licencia A se revoca porque su clave fue expuesta, se podría emitir una licencia B nueva para el mismo usuario, transfiriendo posiblemente el saldo de la billetera o simplemente habilitando un nuevo entorno donde reactivar sus dispositivos legítimos. La re-emisión usualmente involucra un **evento GENESIS** nuevo (para la nueva LicenseID), seguido de eventos de **migración controlada**. Esta migración podría implicar, por ejemplo, marcar en la licencia B que ciertos DeviceIDs previamente asociados a A pueden ser activados sin consumir cupo (tras verificación), o proveer una herramienta para transferir fondos de una wallet a otra de forma atómica. Desde la óptica de la gobernanza, la re-emisión puede ser automatizada o manual. Podría haber reglas predefinidas: *"si una licencia es revocada por motivos no atribuibles a mala conducta del usuario (ej: rotura de seguridad), el sistema genera automáticamente otra licencia con iguales parámetros y transfiere el estado legítimo"*. O bien, en entornos regulatorios, podría requerir que un administrador valide la creación de la nueva licencia (firmando el GENESIS de reemplazo). En cualquier caso, se busca que la transición sea **auditada y transparente**: la nueva licencia podría referenciar en su metadata a la anterior (ej: "Re-emitida de LicenseID X por revocación el día Y"), y el ledger reflejaría en ambas el vínculo (la vieja licencia podría tener un evento especial indicando "reemplazada por LicenseID Z"). Esto facilita a auditores seguir el hilo histórico y asegura que no se "pierdan" dispositivos autorizados en el camino.

Es importante destacar que gobernanza también abarca otros aspectos, como la **independencia del auditor** (p. ej., llaves especiales que solo validan evidencias sin poder modificarlas) y los **procesos de actualización de las normas** (por ejemplo, si en el futuro se cambia la política general del sistema BDL, cómo se aprueba e implementa esa migración en todas las licencias existentes). Todas estas actividades están delineadas por protocolos formales y, cuando es posible, respaldados por la misma infraestructura criptográfica (multi-sig, eventos registrados, etc.). La filosofía subyacente es que la administración de un sistema tan sensible como licencias de billeteras debe ser **estructurada y verificable**: ninguna acción de mantenimiento debe ocurrir en la sombra. Cada revocación, reemplazo o cambio de clave debe dejar un rastro, y preferiblemente, requerir consenso de múltiples partes para evitar abusos o errores. De esta manera, la gobernanza en BDL provee un marco de confianza tanto para el proveedor (que sabe que su sistema puede reaccionar a incidentes con control de daños) como para el usuario (que sabe que su licencia no será alterada o retirada sin justificación y sin los debidos procesos). En síntesis, a través de llaves bien gestionadas, quórums bien definidos y procesos de ciclo de vida claros, la gobernanza garantiza la **sostenibilidad y confiabilidad** del ecosistema de licencias a largo plazo.

2.9 *Enforceability*: condiciones técnicas mínimas para garantizar “N dispositivos”

El concepto de **“enforceability”** (garantía de cumplimiento técnico) en BDL se refiere a la capacidad real del sistema para **hacer cumplir** la restricción de que una licencia solo sea utilizada en *N* dispositivos como máximo, respaldando así las afirmaciones de seguridad con hechos técnicos demostrables. En otras palabras, no basta con declarar en la política que “N dispositivos” es el límite; debe haber suficientes medidas técnicas para que, en la práctica, esa limitación no pueda ser superada sin detección o consecuencias. Alcanzar la enforceability perfecta es complejo debido a la naturaleza copiable de las claves privadas, pero BDL establece un conjunto de **condiciones mínimas** que, combinadas, brindan un alto grado de garantía de cumplimiento:

- **Identificador de dispositivo único y difícilmente clonable:** Cada dispositivo autorizado debe poseer un DeviceID verdaderamente único (derivado de una clave única) y vinculado idealmente a atributos no copiables. La generación local de claves privadas en dispositivos (idealmente dentro de enclaves seguros) y la no exportación de dichas claves es fundamental. Si todos los DeviceIDs activos son *criptográficamente distintos* y se aseguran por hardware, resulta impracticable que un atacante clone un DeviceID en otro aparato. Esto no impide que copie la seed entera de la billetera, pero ese segundo aparato, al carecer del DeviceID correcto (y de la attestation correspondiente), sería detectado como intruso. Enforceability requiere pues que la *identidad digital* de los dispositivos no sea trivial de replicar: dispositivos distintos deben producir identificadores distintos, y compartir credenciales entre ellos debe ser tan difícil como sea posible.

- **Validación estricta de la licencia en cada operación crítica:** Para garantizar N dispositivos, el sistema debe implementar verificaciones de licencia en los momentos clave del uso de la billetera. Por ejemplo, al abrir la aplicación, al intentar firmar una transacción o revelar una clave pública nueva, el cliente de billetera debería verificar que: (a) la licencia asociada sigue activa y no revocada/expirada, y (b) que el DeviceID actual figura como activo bajo esa licencia. Esta verificación puede realizarse consultando el ledger localmente (si el cliente mantiene un caché de eventos) o preguntando a un servicio de indexación distribuido. Lo importante es que **ninguna operación importante proceda sin confirmación de que el dispositivo está autorizado**. Así, aunque un usuario malintencionado tenga la seed en un dispositivo clandestino, si intenta usarla, sus acciones (como transmitir una transacción firmada) deberían ser rechazadas por falta de autorización. En un escenario ideal, incluso la generación de nuevas direcciones estaría supeditada a esta validación – aunque dado que las direcciones en una HD wallet se derivan localmente, quizás la enforceability se centre más en las salidas de fondos y activaciones de dispositivos. En síntesis, cada dispositivo actúa bajo una “sesión” de licencia que debe revalidarse periódicamente (p.ej., con cada contacto con la red) para asegurar que sigue teniendo permiso.

- **Protección del material criptográfico y del entorno de ejecución:** Para evitar que un usuario eluda el control de N dispositivos simplemente exportando su seed o

claves a múltiples entornos, es crucial dificultar esa exportación. Las implementaciones de BDL deben encarar esto con una combinación de diseño de software y uso de hardware seguro. Por ejemplo, la billetera podría **no mostrar la frase mnemónica (seed)** al usuario de forma convencional; en su lugar, podría generar la seed y mantenerla cifrada en el dispositivo, ofreciendo backups cifrados que solo puedan restaurarse vía el proceso oficial de activación (que registraría un nuevo DeviceID). Adicionalmente, apoyarse en enclaves (TEE) o módulos de seguridad (TPM, Secure Element) asegura que la clave privada no pueda ser leída fácilmente del dispositivo original. Si bien usuarios avanzados y determinados podrían todavía encontrar maneras (nada es infalible si tienen control total del dispositivo), la meta es elevar la barrera lo suficiente para que no valga la pena o para que el intento deje rastro. Complementariamente, la aplicación debe blindar sus procesos contra manipulaciones: por ejemplo, empleando verificaciones de integridad (firmas de código, attestation del sistema operativo) para que no se pueda parchear la app quitando las comprobaciones de licencia. Estas medidas juntas constriñen el espacio de maniobra de un atacante *interno* (el propio usuario malicioso), haciendo que cualquier duplicación de la billetera requiera esfuerzos sofisticados y arriesgados.

- **Detección y respuesta a violaciones de cupo:** Aun con medidas preventivas, debe contemplarse la posibilidad de que ocurran violaciones (o intentos) y garantizar que se **detecten de inmediato**. Esto lo proporciona principalmente el sistema de alertas descrito (2.7). Si, por ejemplo, un usuario logra cargar la misma seed en un segundo dispositivo no autorizado (quizá extrayendo la clave en un entorno no oficial), en cuanto ese dispositivo trate de interactuar con la red (por ejemplo, enviar una transacción o sincronizar con el servidor de licencias), su ausencia de DeviceID válido generará un ALERT. De modo similar, si un usuario intenta activar oficialmente un cuarto dispositivo sin haber dado de baja otro, el intento será denegado y registrado. La enforceability exige que *ningún uso por encima de N pase desapercibido*. Cada intento fuera de política queda plasmado en el ledger (vía ALERT) y por tanto es conocido. Además, la respuesta a dichas detecciones debe ser contundente: acumulación de alertas podría, según la política, **suspender automáticamente** la licencia para evitar daños mayores, protegiendo así los fondos y la integridad del sistema hasta aclarar la situación. En casos menos severos, podría simplemente notificar al usuario y al proveedor. Lo importante es que haya **consecuencias técnicas**: por ejemplo, tras un ALERT crítico, el sistema podría requerir re-autenticación completa de todos los dispositivos o invalidar sesiones, de modo que un dispositivo clonado no pueda transaccionar aunque tuviera momentáneamente la clave.

- **Garantía mediante anclaje y auditoría de N dispositivos:** Las medidas anteriores operan principalmente en tiempo real, pero la enforceability también se extiende a la posibilidad de **auditar ex-post** que en ningún momento se sobrepasó el límite N sin registro. Gracias al anclaje criptográfico en Bitcoin (2.6), cualquier auditor puede verificar los eventos de una licencia y constatar, por ejemplo, que nunca hubo más de N activaciones simultáneas sin correspondientes desactivaciones. Si existiera una discrepancia (por ejemplo, 4 DeviceIDs activados y solo cupo para 3, sin alerta ni revocación asociada), eso supondría una violación de las garantías del sistema,

detectable públicamente. La arquitectura BDL busca impedir que tal situación ocurra a nivel de lógica, pero el anclaje provee una **prueba independiente** de que no ocurrió. En un lenguaje más formal, se podría decir: dado el ledger anclado, está garantizado que para cada licencia en cada punto temporal, el número de dispositivos activos <= N; de lo contrario, habría evidencia de inconsistencia anclada (lo cual no sucede asumiendo hash SHA-256 indestructible y PoW de Bitcoin seguro). Así, la propiedad de enforceability se convierte también en un **invariante verificable** del sistema.

- **Escalabilidad a "N dispositivos" múltiples:** Por último, una condición complementaria es que el sistema BDL debe ser capaz de administrar de forma eficiente múltiples dispositivos sin relajar estas garantías. Es decir, soportar N dispositivos no debe implicar, por ejemplo, permitir claves compartidas entre dispositivos (eso rompería la unicidad). Cada dispositivo adicional añade eventos y verificaciones, pero la arquitectura está pensada para escalar linealmente con ellos sin comprometer la seguridad. Enforceability implica que incluso para licencias con N más alto (p.ej., una licencia corporativa que permita 10 dispositivos), todas las condiciones anteriores se mantienen efectivas. La política podría recomendar ciertos topes (quizá por encima de cierto N, el riesgo aumenta, pero aun así se puede controlar). En definitiva, garantizar "N dispositivos" significa garantizarlo para cualquier N configurado, preservando la robustez del modelo.

En resumen, la *enforceability* en BDL descansa en varios pilares: **identificación fuerte de dispositivos**, **control de activación centralizado**, **protección contra extracción de claves**, **monitoreo activo con alertas** y **auditoría inmutable**. Solo al converger todas estas condiciones podemos afirmar con rigurosidad que "una licencia, N dispositivos" no es una simple directriz de uso sino una restricción técnicamente aplicada. Este enfoque responde directamente al principio de "no-simulacro técnico": no proclamar una seguridad que no pueda sustentarse. Al cumplir con estas condiciones mínimas, BDL ofrece una garantía excepcionalmente alta de que un usuario malicioso no podrá explotar la naturaleza copiable de las wallets deterministas sin ser descubierto y contrarrestado, preservando así la premisa fundamental del licenciamiento criptográfico.