Розгорнуті Нотатки для Презентації
Технічна Специфікація Платіжного Сервісу

Ключові тези та обґрунтування за матеріалами специфікації

1 1. Вибір Бібліотек: Точність понад усе

- **Фундаментальний конфлікт:** Існує компроміс між абсолютною фінансовою точністю (відомою як *Processing Integrity*) та вимогами криптографічного комплаєнсу (стандарти `PCI DSS` / `FIPS`).
- **Архітектурний Пріоритет (№1): Точність**.
  - Використання бібліотек точної арифметики (напр., `Decimal` в C#, `BigDecimal` в Java) є **обов'язковим** для всіх фінансових розрахунків.
  - **Обґрунтування:** Типи з плаваючою комою (`float`, `double`) категорично **заборонені**. Вони вносять помилки округлення (напр., $0.1 + 0.2 \neq 0.3$ у бінарній системі), що призводить до непередбачуваних фінансових втрат, порушення балансів та фундаментальної помилки бізнес-логіки. Це порушує *Processing Integrity*.
- **Наслідок (Продуктивність):** `Decimal` працює значно повільніше, ніж `float` (в 15x-30x разів). Це створює "вузьке місце" в системі.
- **Архітектурне Рішення (Компенсація):**
  - **CQRS** (Command Query Responsibility Segregation / Відокремлення відповідальності команди та запиту).
  - **Як це працює:** Ми розділяємо систему на два потоки:
    - **Команди (Write):** Повільні операції, що змінюють дані (напр., транзакція). Вони використовують `Decimal` і гарантують точність та `ACID`.
    - **Запити (Read):** Швидкі операції читання (напр., показ балансу). Вони можуть читати з оптимізованого кешу або репліки.
  - Це ізолює повільні, але точні операції, не впливаючи на загальну швидкість відгуку системи для користувача.
- **Пріоритет №2 (Мандатний):** Криптографія.
  - Використання лише `FIPS 140-2/3` валідованих криптографічних модулів (напр., `Bouncy Castle FIPS`). FIPS – це стандарт уряду США, що гарантує правильну реалізацію крипто-алгоритмів.
  - Обов'язкове використання `TLS 1.3` з `ECDHE` (Elliptic Curve Diffie-Hellman Ephemeral) для забезпечення *Forward Secrecy* (навіть якщо приватний ключ сервера буде скомпрометовано в майбутньому, минулі сесії неможливо буде розшифрувати).

2 2. Управління Ключами (KMS) та ПСП

- **Життєвий Цикл Ключа:** Весь процес має бути захищеним, автоматизованим та аудійованим:
    1. **Генерація:** Лише всередині захищеного модуля (`HSM`). Ключ ніколи не існує у відкритому вигляді (plain text) поза цим модулем.
    2. **Зберігання:** У захищеному сховищі (`KMS Vault`), зашифрованим іншим ключем (Key-Encryption-Key, KEK).
    3. **Використання:** Доступ до ключа – лише через захищений API (`PKCS#11`), який надсилає дані *до* ключа для шифрування, а не витягує ключ *назовні*.
    4. **Ротація / Архівування:** Автоматизована регулярна зміна ключів.
    5. **Знищення:** Криптографічне стирання (знищення KEK, що робить дані, зашифровані DEK, нечитабельними).
- **HSM (Hardware) vs Cloud KMS (Software/Cloud):**
    - **HSM (Рівень FIPS 3/4), (напр., Thales):** Апаратний пристрій. Найвища фізична безпека. Ключі **ніколи** не покидають пристрій (*non-exportable keys*). Мандатний для захисту PIN-блоків (стандарт `ISO 9564`) та кореневих ключів (KEK).
    - **Cloud KMS (Рівень FIPS 2/3), (напр., AWS KMS, Azure Key Vault):** Хмарний сервіс. Висока масштабованість, легка автоматизація ротації. Ідеально для шифрування даних у стані спокою (*data at rest*).
- **ПСП (CSPRNG):** Генератори псевдовипадкових чисел.
    - Це **не** `Math.random()`. Це *Cryptographically Secure* PRNG.
    - **Вимога:** Стійкість до атак (неможливість передбачити наступне число).
    - **Рекомендація (NIST):** `HMAC_DRBG` (базується на хеш-функції, дуже стійкий) або `CTR_DRBG` (базується на блочному шифрі, швидкий, якщо є апаратне прискорення AES).

3 3. Архітектурні Варіанти: Мікросервіси

- **Цільові Метрики:** >20,000 TPS (Транзакцій за секунду) та доступність 99.99% (”Four Nines” – 52 хвилини простою на **рік**).
- **Рішення: Мікросервісна Архітектура**.
    - **Обґрунтування:** Монолітна архітектура не витримає такого навантаження. Мікросервіси дозволяють **незалежне масштабування** (можна додати 100 серверів для ”Ledger”, але лише 5 для ”Авторизації”) та **ізоляцію збоїв** (збій у сервісі звітів не зупинить транзакції).
- **Ключові Ізольовані Сервіси:**
    - **Ledger Service (Бухгалтерія):** Критично. ”Священна” частина системи. Використовує `Decimal` та `ACID`-сумісні бази даних.
    - **Crypto Service (Криптографія):** Критично. **Єдиний** сервіс в усій системі, який має право ”говорити” з `HSM`. Ізоляція цього сервісу радикально звужує ”attack surface” та спрощує аудит PCI DSS.
    - **Сервіс Авторизації:** Високодоступний, використовує швидкий кеш.

- **Комунікація між Сервісами:**
    - **mTLS (Mutual TLS):** Для внутрішньої комунікації (нульова довіра / Zero Trust). Не просто клієнт перевіряє сервер (як у браузері), а **обидва** сервіси перевіряють сертифікати один одного.
    - **Message Broker (напр., Kafka):** Для асинхронної комунікації. Забезпечує стійкість до пікових навантажень. Якщо "Ledger" не встигає, транзакції шикуються в чергу, а не втрачаються.

4 4. Реалізації: API, PKCS та Case Studies

- **Золотий Стандарт API: PKCS#11 (Cryptoki)**.
    - Це "lingua franca" – стандартний "драйвер" (API) для взаємодії програмного забезпечення з **будь-яким** `HSM` (Thales, nCipher тощо).
    - Це дозволяє системі бути "агностичною" до постачальника `HSM`.
- **Case Study 1: STRIPE (Токенізація)**
    - **Технологія: Токенізація** (`Stripe Elements`).
    - **Суть:** PAN (номер картки) шифрується на стороні клієнта (в браузері) і **ніколи не потрапляє** на сервери мерчанта. Мерчант оперує лише "токеном" (безпечним замінником, напр., `tok_123abc`).
    - **Цінність:** Це **радикально** знижує зону дії (scope) та відповідальність мерчанта за `PCI DSS`.
- **Case Study 2: SWEDBANK (KMS)**
    - **Технологія:** Централізований **KMS** (CKMS/TKMS – Terminal KMS).
    - **Суть:** Банк має тисячі POS-терміналів. `TKMS` – це система, що автоматизує весь життєвий цикл ключів (генерація, ротація, безпечне розповсюдження) на кожен окремий термінал у "полі".
- **Case Study 3: VERIFONE (P2PE)**
    - **Технологія: P2PE** (Point-to-Point Encryption).
    - **Суть:** Дані картки шифруються **всередині** захищеного модуля платіжного терміналу (перша "точка") в момент зчитування. Дані залишаються зашифрованими аж до шлюзу процесера (друга "точка").
    - **Цінність:** Мережа мерчанта (напр., Wi-Fi у кав'ярні) бачить лише зашифровані дані, що знову ж таки знижує їхні ризики та `PCI-Scope`.