# Distributed Ledger

Smart Contract Security

Hyperledger Fabric Extension

Cem Basoglu

# Smart Contracts

- Dezentrale Ausführung von Business-Constraints / -Logik
- Sicherstellung der Integrität ohne zentrale Autorität
- Transparente Transaktionen
- Entwicklung in Solidity
- Kompiliert zu Bytecode für Ethereum Virtual Machine (EVM)
- Unveränderlichkeit von Transaktionen

⇒ Unveränderlichkeit erfordert präventive Maßnahmen um Schwachstellen zu vermeiden

# Typische Schwachstellen

# Beispiel #1

```
contract auction {
  mapping (address => uint) private userBalances;
  function transfer(address to, uint amount) {
      if (userBalances[msg.sender] >= amount) {
         userBalances[to] += amount;
         userBalances[msg.sender] -= amount;
      }
  }
  function withdrawBalance() public {
      uint amountToWithdraw = userBalances[msg.sender];
      msg.sender.call.value(amountToWithdraw)();
      userBalances[msg.sender] = 0;
  }
}
```

⇒ Re-Entrency bzw. Cross-function Race Condition

# Beispiel #2

```
contract auction {
  address currentLeader;
  uint highestBid;

  function bid() {
      if (msg.value <= highestBid) { throw; }
      if (!currentLeader.send(highestBid)) { throw; }

      currentLeader = msg.sender;
      highestBid = msg.value;
  }
}
```

⇒ Denial of Service

# Sicherheitsmaßnahmen und Best-Practices

# Checks-Effects-Interactions-Pattern

1. Eingabe Parameter und State prüfen
    - Funktionsparameter validieren (Null-Check, ungültige Werte, ...)
    - Berechtigungen prüfen (Besitzer, Teilnehmer, ...)
    - Statevariablen abgleichen (Guthaben, ...)
2. State Änderungen vornehmen
    - Guthaben aktualisieren
    - Assets übertragen
3. Smart Contract Interaktionen durchführen
    - Guthaben senden
    - externe Funktionen aufrufen

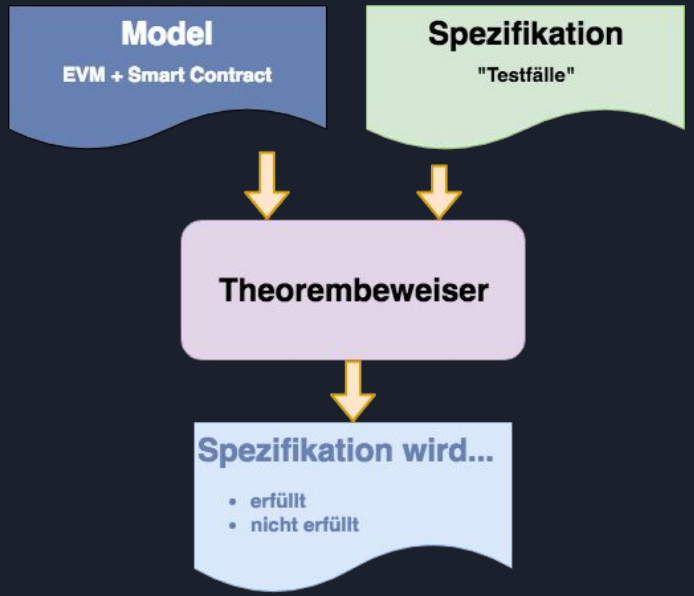
# Update- und andere Schutzmechanismen

- Ausführung pausieren
  - State Variable zum pausieren die von relevante Funktionen geprüft wird
  - Funktion zum ändern der State Variable
- Auszahlungen drosseln
  - zukünftige Auszahlung wird vorgemerkt
  - Auszahlung wenn Schwellwert überschritten
- Externe Integritätsprüfung
  - Externer Dienst pollt Smart Contract
  - Prüfung der Integrität ( z.B. Contract Guthaben == Summe State-Guthaben)
- Updates
  - Interface mit Update-Funktion und State-Variablen als Parameter
  - Smart-Contracts implementieren das Interface
  - Funktion mit Adresse als Parameter zur neuen Version

# Formale Verifikation und Smart Contracts

# Formale Verifikation

- Model
  - Abbildung des Systems in einem Modell
  - Syntax je nach Theorembeweiser
- Spezifikation
  - Aussage die es zu widerlegen/beweisen gilt
  - i.d.R. als logischer Ausdruck (Prädikatenlogik)
- Theorembeweiser
  - HOL Light
  - Isabelle/HOL
  - Z3
- Ergebnis
  - Ob ein Zustand im System erzeugt werden kann, in der die Spezifikation nicht erfüllt wird.



# Smart Contract Verification Tools

MAIAN

- Symbolische Modell Verifikation
- Spezialisiert auf
  - Prodigal
  - Suicidal
  - Greedy
- Automatischer Ausschluss von False-Positives
- Eigene Spezifikationen möglich aber undokumentiert

Mythril

- Symbolische Modell Verifikation
- Erkennt 19 Schwachstellen-Typen u.a.
  - Re-Entrency
  - Ungesicherte kritische Funktionen
  - Overflow/Underflow
- Kontrollflussgraph zur Visualisierung
- API für eigene Spezifikation
- Baut auf Z3 Theoremproover auf

# Hyperledger Fabric Extension

Secure Certificate Storage

# Hyperledger Fabric Client SDK

- Interaktion mit den Fabric-Netzwerk (Peers, Ordering-Service)
- Autorisierung der Nutzer durch Zertifikate
- Signierung der Anfragen mit dem privaten Schlüssel
- Nur unter Node.js lauffähig

## ⇒ Folgen für Webapplikationen

1. Interaktion mit dem Distributed Ledger nur im Backend möglich
2. Unnötige weitere Schicht da Business Logik im Chaincode sichergestellt wird
3. Erhöhung der potenziellen Angriffsfläche durch...
    a. ... vorhalten der Zertifikate + Keys im Backend (Zusätzliche Authentifizierung im Backend nötig)
    b. ... übertragen der Zertifikate + Keys mit jeder Anfrage

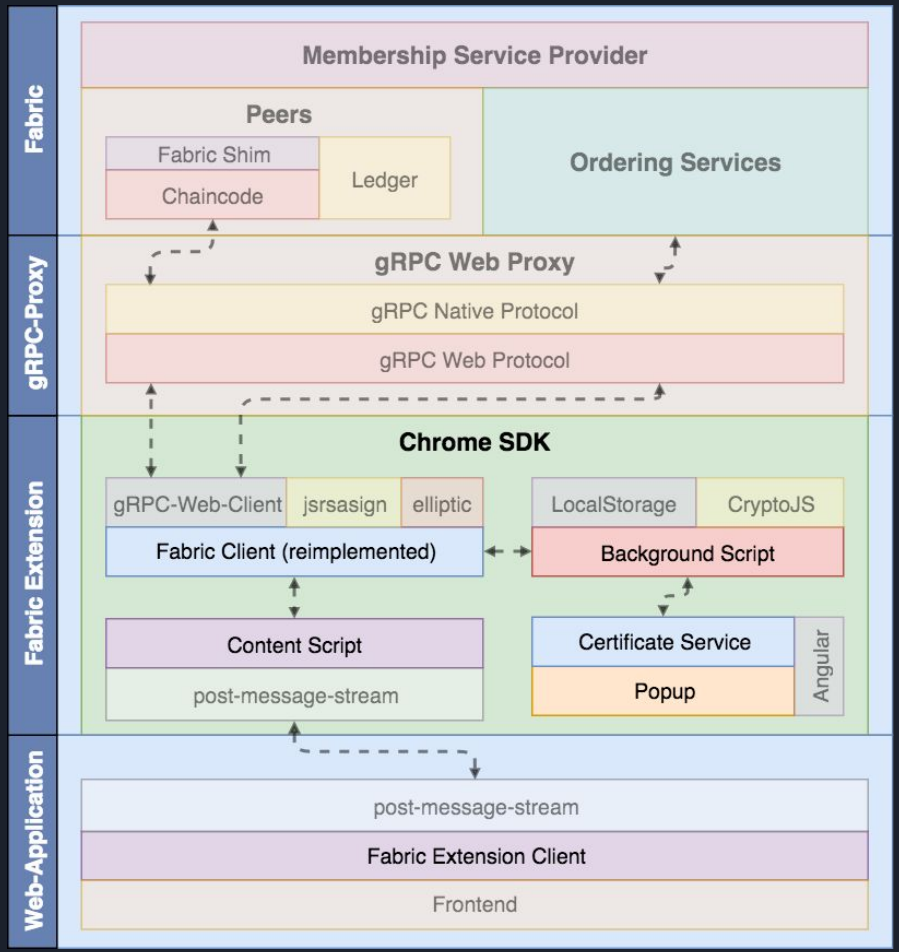
# Fabric Browser Extension

- Sichere Verwahrung der Zertifikate und privaten Schlüssel
- Passwortgeschützte Anmeldung und Verschlüsselung
- Interaktion mit dem Hyperledger Fabric Netzwerk
- Nur die mit dem privaten Schlüssel signierten Anfragen werden übertragen

⇒ Zertifikate und Schlüssel verbleiben beim Client

# Architektur

- Fabric Client
  - Interaktion mit dem Fabric Netzwerk
  - Signierung der Anfragen
  - Implementierung des gRPC-Web-Protokolls
- Fabric Extension Client
  - Schnittstelle für den Zugriff aus der Webapplikation
  - Indirekte Interaktion mit dem Fabric Netzwerk
- Popup / Certificate Service
  - Angular Frontend für die Erweiterung
  - Verwaltung der Zertifikate



# Demo

# Vielen Dank!
# Fragen?

Nein! Da gibt es kein Portal zu!