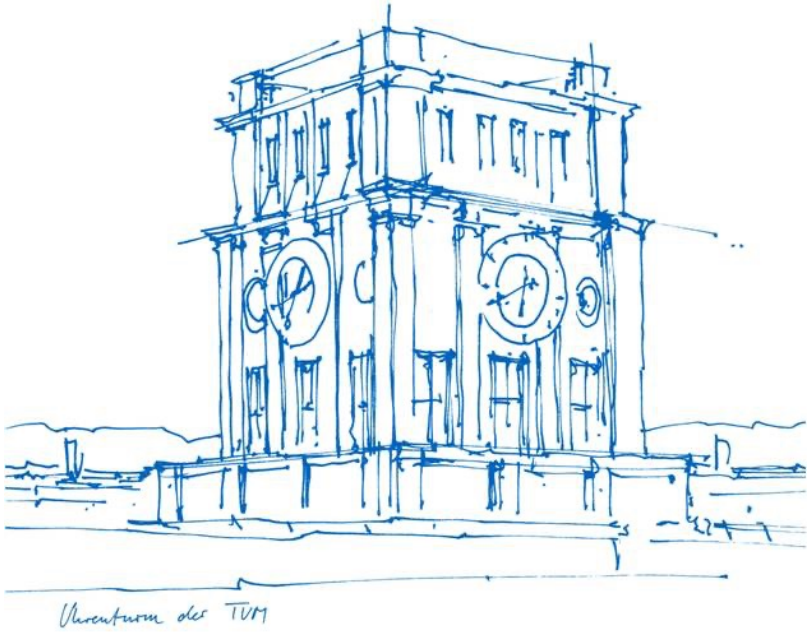
Chair of Connected Mobility
TUM School of Computation, Information and Technology
Technical University of Munich

Synchronized, Refundable and Revocable Offline Blockchain Transactions



Srivatsav Chenna

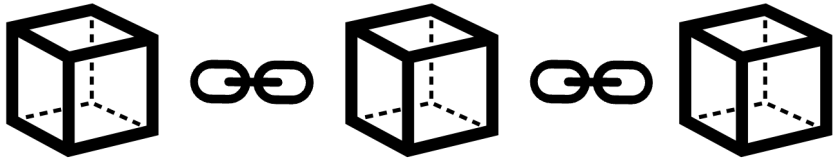
PhD Candidate @ TUM and DENSO

Supervisor: PD Dr. habil. Christian Prehofer

26 September 2025

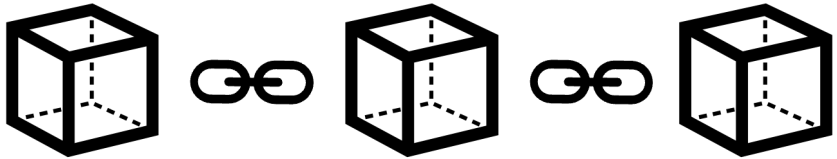
Introduction

- Blockchain is a decentralized, distributed ledger that securely records transactions across many node.
- They provide immutability and transparency of the record
- Applications: Supply Chain, Healthcare, Finance
- Current context: EU is considering deploying digital euro on public blockchains like Ethereum and Solana



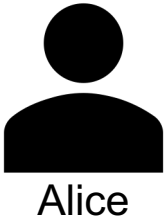
Motivation

- Blockchain systems rely on constant network connectivity for security and immutability of transactions.
- Restricts technology's application in critical scenarios with unreliable or non-existent internet access E.g.
  - Remote areas or during natural disasters
  - Network compromised due to cyber attacks
- Digital currencies/tokens need a reliable medium of exchange (Analogous to cash)

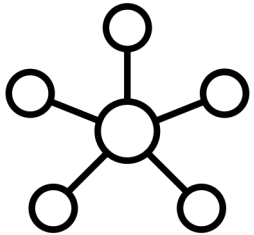


Blockchain Transaction: Alice → Bob

- Create and sign transaction
- Broadcast transaction

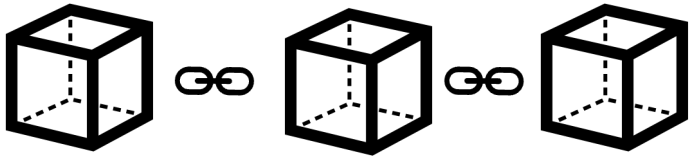


Alice

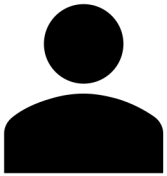


- Verify Alice signature
- Verify Alice has sufficient balance
- For consensus among nodes

- Record on bockchain ledger



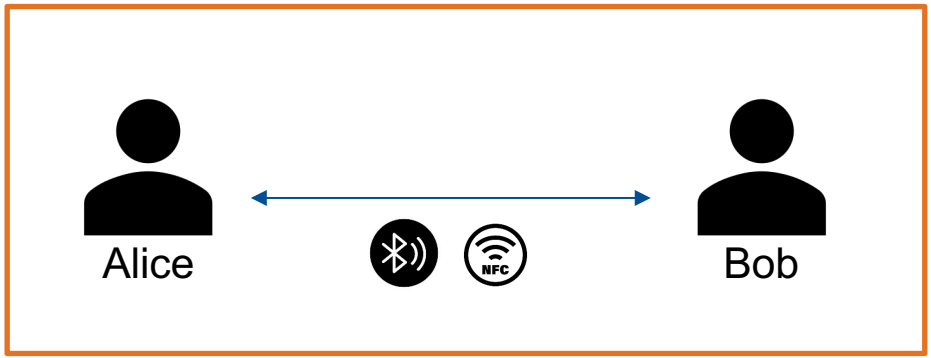
- Verify transaction confirmation



Bob

Offline Blockchain Transaction: Alice → Bob

- Create and sign transaction
- Send transaction to Bob

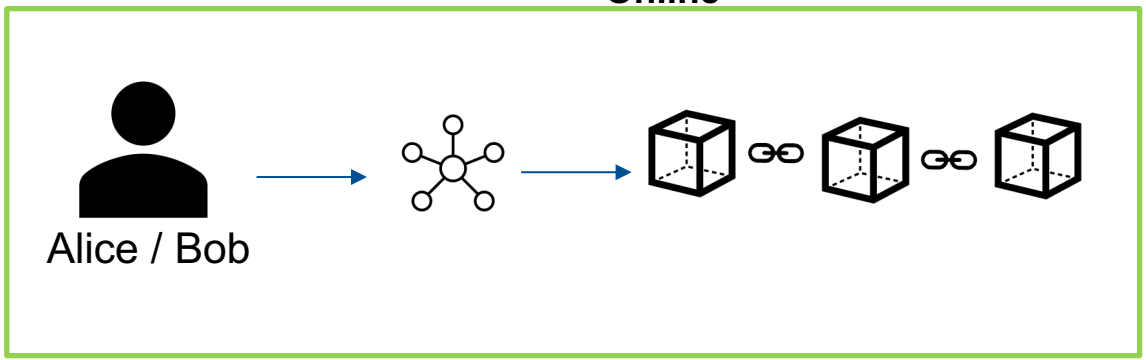


- Verify signature and transaction information
- Record/store transaction locally

Offline

Online

- Alice / Bob can broadcast the transaction to the network



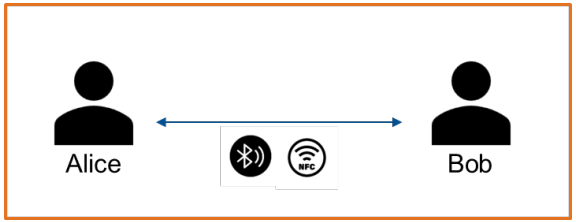
Challenges on Offline Blockchain Systems

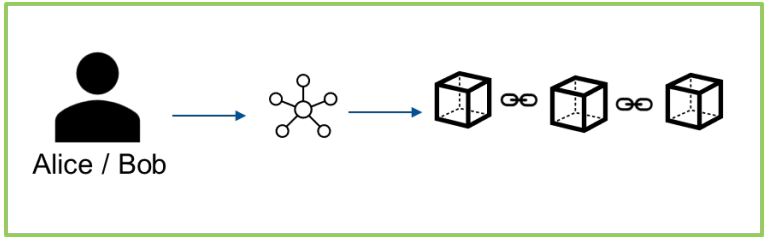
Offline Transaction Exchange

- **Double Spending:** Without online verification, a user could spend the same digital token more than once.
- **Repudiation:** Denying a transaction after it has been completed. Alice or Bob can refuse to record the transaction to the blockchain.
- **Existing solutions secure offline transaction exchange by limiting user control over the digital wallet.**

Online Transaction Recording

- **On-chain synchronization of offline transactions:** Transactions are recorded when users come back online, this might cause the network to reject transactions submitted out of order
- **Lost or Compromised Wallets:** Lost or compromised wallet can lead to the permanent loss of all funds and unrecorded transactions.
- **Challenges in transaction recording are not sufficiently addressed in existing solutions**





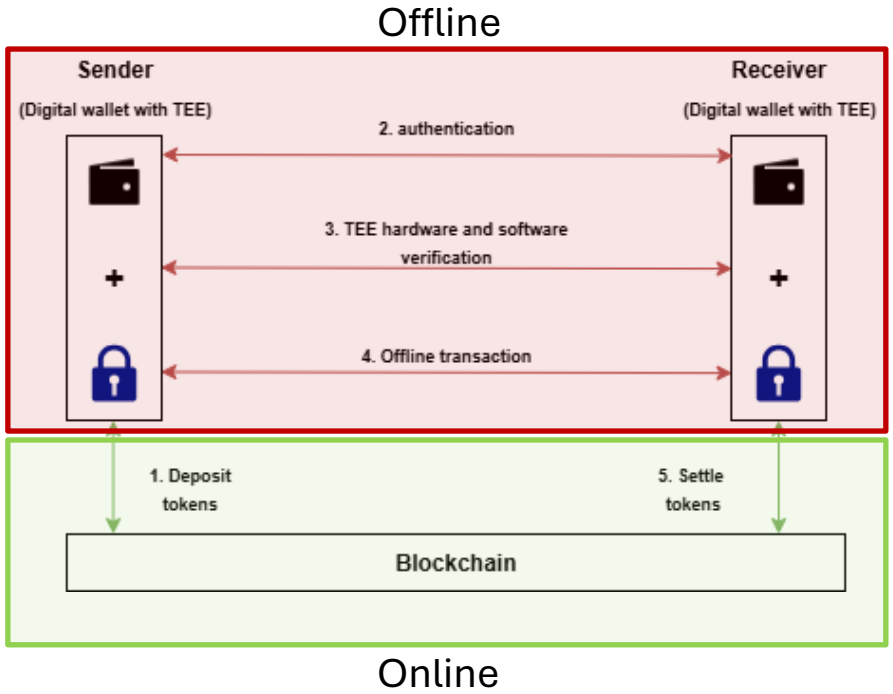
Existing solutions

- Existing solutions utilize a 3 phase framework:
  1. Token deposit phase
  2. Offline transaction phase
  3. Token settlement phase

- The **Trusted Execution Environments** (TEEs) manage the balance – prevents user from double spending

- TEEs hardware and software is verified to establish trust in the transaction

- The protocols involves in ack exchange to prevent repudiation



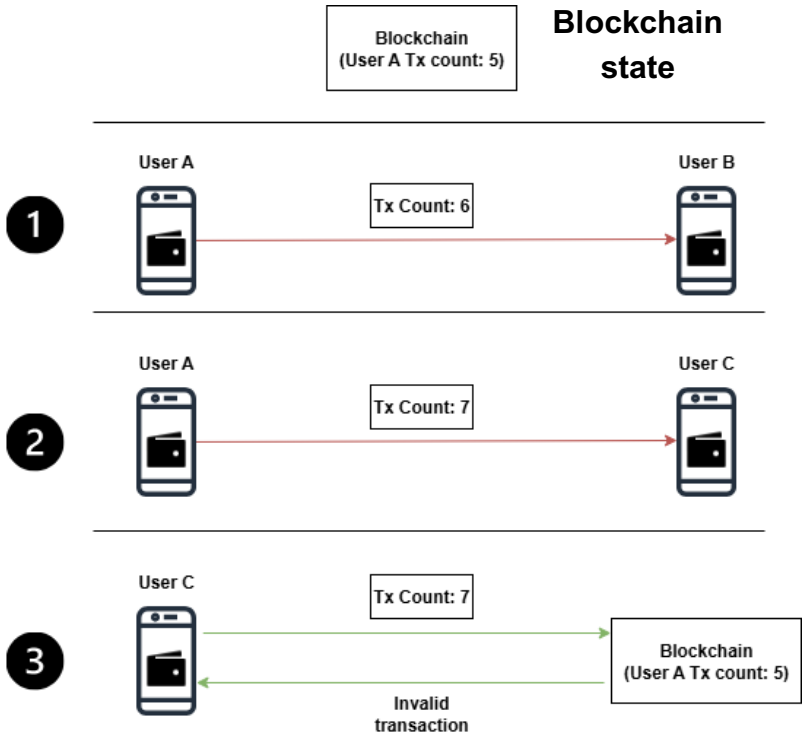
Limitations of Existing Solutions

- While TEEs help prevent double spending , repudiation, and protect offline transaction exchange. The challenges with recording the offline transactions back online is largely unaddressed:
    - **Synchronization Issues:** Offline transactions could get rejected due to outdated nonce or state mismatches when reconnected
    - **No Robust Recovery Mechanism:** Lost or compromised wallets result in irreversible loss of funds and unrecorded transactions
- **There is a need for a Synchronized, Refundable and Revocable offline blockchain system**

Challenges During Online Transaction Recording

- **Synchronization Issues:** Offline transactions risk being rejected if their transaction information does not match the current on-chain state.
  - E.g: Transaction count

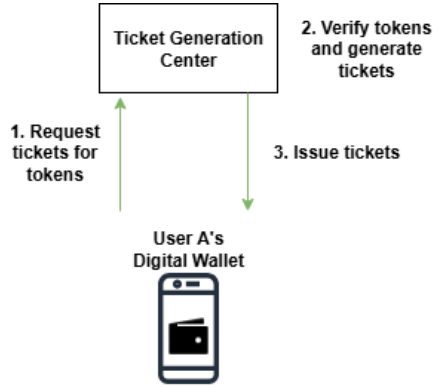
- **Lost or Compromised Wallets:**
  - If a wallet is lost, all unrecorded transactions and deposited funds are lost
  - For compromised wallets, conduct fraudulent offline transactions or prevent legitimate ones from being recorded.



Our solution: Ticket-Based System

- **Idea: Use “tickets” to authorize offline transactions**
- **Online Setup:**
  - Each token/coin deposited into digital wallet is issued a ticket
  - The ticket serves as proof of ownership and authorization for offline transfers
  - Tickets are transferred along with the token/coin during offline transaction.
  - Tickets are generated by a **trusted third party** called **Ticket Generation Center (TGC)**

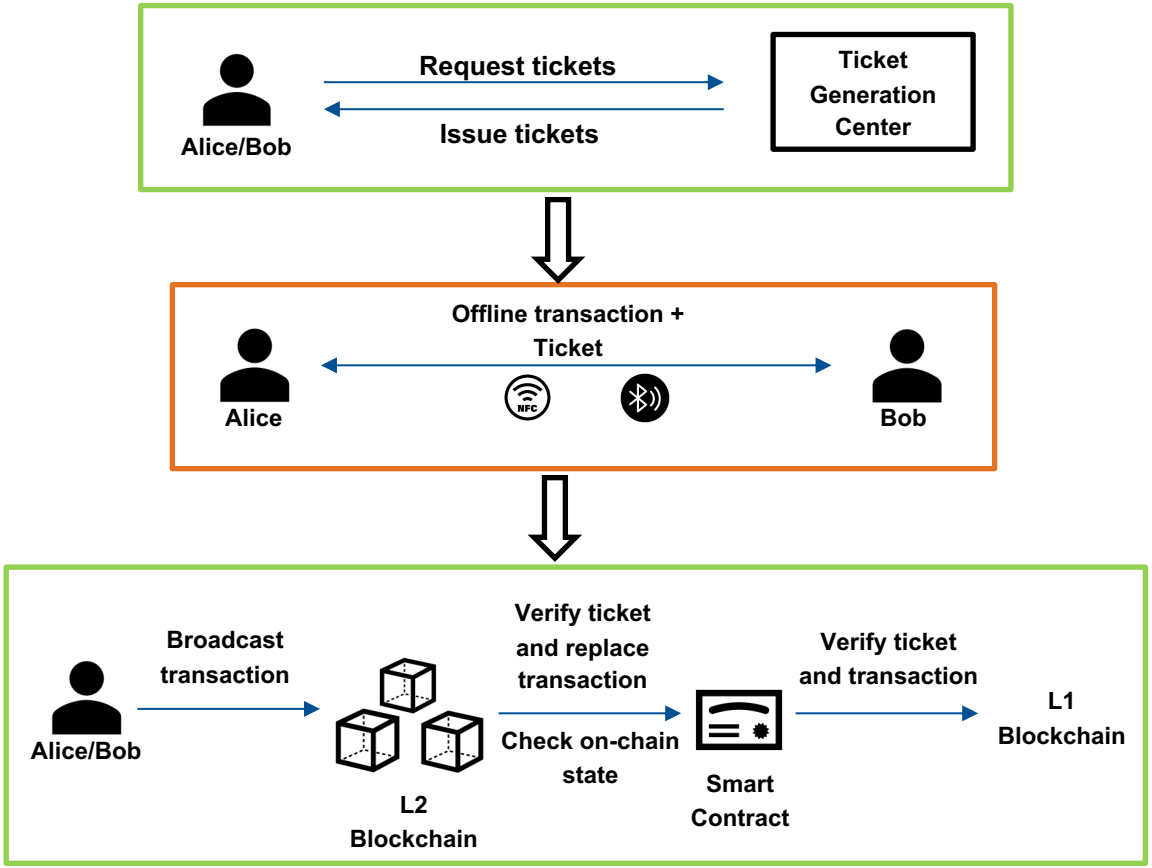
- **Online Transaction Recording:**
  - Intermediary phase – offline transactions are updated according to on-chain state on an **L2 network**
  - Final verifications are made by a **smart contract**



Ticket

```
{
"sender": "A",
"tokenId": 1,
"nonce": 1,
"expiration-timestamp": 1716403200
}
```

Simple Overview of the Ticket-based system



Layer 2 Blockchain

- **L2 blockchains:** secondary networks built on top of a main blockchain to handle transactions off-chain. E.g. Bundle multiple transactions together

  - The L2 system verifies the ticket and checks if the transaction information is synchronized with on-chain state
  - If outdated, replacement transaction is created to reflect the blockchain's current state.
  - Replacement transaction, ticket and the original transaction are forwarded to the smart contract.
  - L2 functions as a trusted relayer.

```
{
"from": "P",
"nonce": 1, //Outdated nonce value
"gas_price": ...,
"gas_limit": ...,
"recipient": "S",
"value": 0,
"data": "safeTransferWithTicket(A, B, C, tokenId, Ticket)",
"v,r,s": "Transaction signature"
}
```

Original transaction

```
{
"from": "P",
"nonce": 3, //Updated nonce value
"gas_price": ...,
"gas_limit": ...,
"recipient": "S",
"value": 0,
"data": "safeTransferWithTicket(A, B, C, tokenId, Ticket)",
"v,r,s": "Transaction signature"
}
```

Replacement transaction

Smart Contract

- **Smart contracts:** self-executing code that runs on a blockchain. They automatically execute the actions when pre-defined conditions are met
  - Smart contract verifies ticket, replacement transaction and the outdated transaction signature
  - If all checks pass, smart contract authorizes the transaction
  - Smart contract is also responsible for the refund and revocation process

```
{
"from": "P",
"nonce": 1, //Outdated nonce value
"gas_price": ...,
"gas_limit": ...,
"recipient": "S",
"value": 0,
"data": "safeTransferWithTicket(A, B, C, tokenId, Ticket)",
"v,r,s": "Transaction signature"
}
```

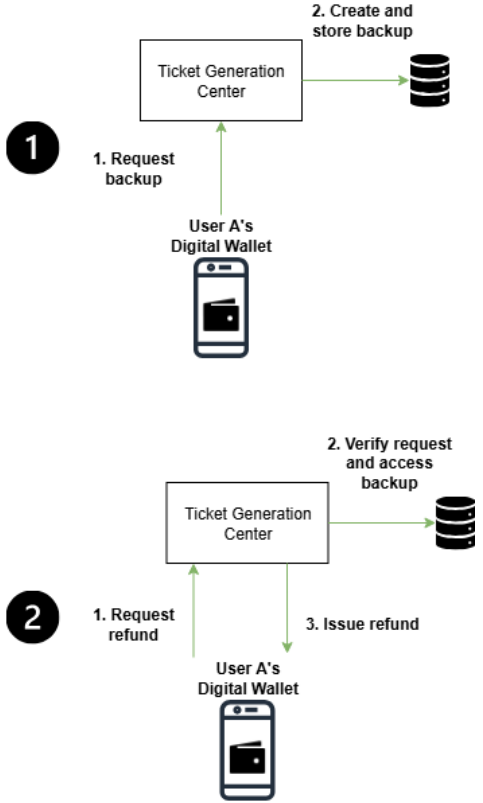
Original transaction

```
{
"from": "P",
"nonce": 3, //Updated nonce value
"gas_price": ...,
"gas_limit": ...,
"recipient": "S",
"value": 0,
"data": "safeTransferWithTicket(A, B, C, tokenId, Ticket)",
"v,r,s": "Transaction signature"
}
```

Replacement transaction

Refund and Revocation

- **Generate Backup:** The TGC securely backs up all issued tickets for the user.
- **Process Refund:** If a wallet is lost, the user can request a refund.
  - Verifying the user's formal complaint or wait for tickets to expire
  - The secure back-up is accessed to process the refund
- **Revoke Transactions:** If a wallet is compromised, wallet address can be revoked by the TGC.
  - The TGC then updates a revocation list within the smart contract
- Both the revocation and refund requests are sent to the smart contract by the TGC
- The smart contract verifies the requests to process the refund or add address to the revocation list



Implementation

- **We implemented a system designed to enable secure token transactions without a constant internet connection.**

- **Layer 2 Network**
  - Polygon Layer 2 test network

- **Layer 1 Network**
  - Ethereum-based Hardhat L1 blockchain network

Both L1 and L2 networks were deployed locally. This allowed us to simulate the full transaction flow, from offline execution on Layer 2 to final settlement on Layer 1.

- **Smart Contract**
  - A custom smart contract was designed based on the ERC-20 token standard

Evaluation

- **We analyzed our system based on gas consumption and transaction latency.**

- **Gas Consumption:**
  - "gas" is a unit that measures the computational effort needed for a transaction.
  - We measure the gas used by our custom smart contract to execute a token transfer.

- **Transaction Latency:**
  - We measure the time from when a smart contract call is submitted to the Polygon network until it is confirmed.

- We compared the performance metrics against a standard token smart contract.

- **This helps us assess the impact of our modifications on the blockchains's performance.**

Results

- **Gas Consumption**
  - Our custom smart contract consumes nearly double the of a standard ERC-20 contract.
  - This means the user needs to pay more in transaction fee for the additional processing
  - Gas efficiency can be significantly improved through code optimization.

- **Transaction Latency**
  - Our custom contract had a latency only slightly higher than the standard contract
  - The performance overhead of our custom implementation is minimal and does not cause significant transaction delays.

	Transaction Latency (s)	Gas Consumption (gas units)
Stanadard Token Smart Contract	3.83	31106
Custom Smart Contract	3.88	65763

Acknowledgements and References

[1] S. Chenna and C. Prehofer, "Combining Verifiable Credentials and Blockchain Tokens for Traceable and Offline Token Operations," *2023 IEEE 9th World Forum on Internet of Things (WF-IoT)*, Aveiro, Portugal, 2023, pp. 1-6, doi: 10.1109/WF-IoT58464.2023.10539518.

[2] A, Buran, S. Chenna and C. Prehofer, A Ticket-Based Solution for Synchronized, Refundable and Revocable Offline Blockchain Transactions, ICBTA 2025 (To be Published, Dec)

Research contributors:

M. Sc. Emin Adem Buran, TUM
M. Sc. Mohamed Rahmouni

Thank you!

BACKUP

Our solution:

- **Ticket Generation Center (TGC):** Issues unique, cryptographically signed "tickets" for each NFT a user wants to transact offline.
- **Layer 2 (L2) Blockchain:** The primary network for recording offline transactions. It can automatically handle synchronization issues and generate new transactions if needed.
- **Custom Smart Contract:** Verifies tickets and transactions. It also manages refund and revocation lists to protect users.