Stablecoins & Lending Protocols

Dan Boneh
Recap: Solidity

Everything is a contract:

- Contracts manage state variables
- Contracts have functions that can be called externally
- Can inherit code from other contracts (contract A is B, C)
- Types of contracts: contract, interface, abstract, library

Global objects: block, msg, tx
An example: ERC20 tokens


- A standard API for fungible tokens. (ERC-721 for non-fungible tokens)

- An ERC20 token is itself a smart contract that maintains all user balances:
  
  ```solidity
  mapping(address => uint256) internal _balances;
  ```

- A standard interface allows other contracts to interact with every ERC20 token. No need for special logic for each token.
ERC20 token interface

function `transfer`(address _to, uint256 _value) external returns (bool);

function `transferFrom`(address _from, address _to, uint256 _value) external returns (bool);

function `approve`(address _spender, uint256 _value) external returns (bool);

function `totalSupply`() external view returns (uint256);

function `balanceOf`(address _owner) external view returns (uint256);

function `allowance`(address _owner, address _spender) external view returns (uint256);
contract ERC20 is IERC20 {
    mapping (address => uint256) internal _balances;

    function transfer(address _to, uint256 _value) external returns (bool) {
        require(_balances[msg.sender] >= _value, "ERC20_INSUFFICIENT_FUNDS");
        _balances[msg.sender] -= _value;
        _balances[_to] += _value;
        emit Transfer(msg.sender, _to, _value);  // write log message
        return true;
    }
}

Tokens can be minted by a function mint(address _to, uint256 _value) onlyOwner;
Anyone can read ERC20 \_balances[]

Transaction Hash: 0x6b85ca95e484d94503d1276456bfc32cc55f6fdb8bb231ff83....

Tells the USDC contract to transfer 10,010.00 USDC from Circle’s account to 0x7656159E42209A95b77aD374d...

<table>
<thead>
<tr>
<th>Storage Address</th>
<th>0x4d3e7741e6c98c0c469419fcf8fa7ec622d7b26345802d22d17415768760f8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before:</td>
<td>Hex → 0x0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>After:</td>
<td>Hex → 0x00000000000000000000000000000000000000000000000000000000000002540be400</td>
</tr>
</tbody>
</table>

recipient’s entry

<table>
<thead>
<tr>
<th>Storage Address</th>
<th>0x57d18af793d7300c4ba46d192ec7aa095070dde6c52c687c6d0d92fd8532b305</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before:</td>
<td>Hex → 0x0000000000000000000000000000000000000000000000000000000000000000266988cda8061</td>
</tr>
<tr>
<td>After:</td>
<td>Hex → 0x0000000000000000000000000000000000000000000000000000000000002669638ce9c61</td>
</tr>
</tbody>
</table>

(Circle’s balance after)
Calling other contracts

Addresses can be cast to contract types.

```solidity
address _token;
ERC20Token tokenContract = ERC20Token(_token);
```

To call the “transfer” function of contract at address _token:

```solidity
tokenContract.transfer(_to, _value);
```
The world of DeFi

borrow 100 B from Compound

transfer 100 from me to Alice

on-chain contracts

Compound (lending)
Uniswap (exchange)
Aave (lending)

ERC20 coin A
ERC20 coin B
ERC20 coin C
ERC721 NFT D
The world of DeFi

ERC20 coin A
ERC20 coin B
ERC20 coin C
ERC721 NFT D

Compound (lending)
Uniswap (exchange)
Aave (lending)

Exchange 10B for 20C

on-chain contracts
DeFi app #1: Stablecoins
Stable Coins

A cryptocurrency designed to trade at a fixed price

• Examples: 1 coin = 1 USD, 1 coin = 1 EUR, 1 coin = 1 USDX

Goals:

• Integrate real-world currencies into on-chain applications

• Enable people without easy access to USD, to hold and trade a USD-equivalent asset
<table>
<thead>
<tr>
<th>Types</th>
<th>Centralized</th>
<th>Algorithmic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collateralized</td>
<td>Custodial stablecoins (USD Coin)</td>
<td>Synthetics (DAI, RAI)</td>
</tr>
<tr>
<td>Undercollateralized</td>
<td>Central bank (digital) currency</td>
<td>Undercollateralized stablecoins</td>
</tr>
</tbody>
</table>
Custodial stablecoins: minting

1. Deploy ERC-20 contract

2. Custodian

   - mint(Bob, 100)
   - mint(Alice, 35)

3. State
   - Bob: 100
   - Alice: 35

4. _balances
   - 100 USD
   - + 35 USD
Custodial stablecoins: transfers

pay Carol 15$: transfer(Bob → Carol, 15)

(and gas fee)

Transfers are done on-chain (custodian is not involved)

135 USD

Bob: $50
Alice: 35
Carol: 15

_balances
Custodial stablecoins: withdrawal

Withdraw 60 USD

60 USD

BANK

Custodian

135 USD

burn(Bob, 60)

Bob: 85
Alice: 35
Carol: 15

_balances
## Two Examples

<table>
<thead>
<tr>
<th></th>
<th>Coins issued</th>
<th>24h volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>USDC</td>
<td>25.3 B</td>
<td>4.6 B</td>
</tr>
<tr>
<td>USDT</td>
<td>83.7 B</td>
<td>20.8 B</td>
</tr>
</tbody>
</table>
Some issues

Custodian keeps treasury in a traditional bank
- Must be audited to ensure treasury is available
- Earns interest on deposits

Custodian has strong powers:
- Can freeze accounts / refuse withdrawal requests
- Custodian can remove funds from user balances
DeFi app #2: Lending Systems

Goal: explain how decentralized lending works

This is not investment or financial advice
The role of banks in the economy

Banks bring together lenders and borrowers

- deposit assets
- deposit interest
- borrow
- borrow interest

bank spread
(borrow interest – deposit interest)
The role of banks in the economy

Alice will get her deposit back either way

Bank assumes the risk of Bob defaulting

Deposit assets
withdraw deposit

borrow
repay loan

bank spread
(borrow interest – deposit interest)
Same as with a traditional bank:

Alice gives her assets to the CeFi institution to lend out to Bob

- Deposit ETH
- Deposit interest
- Borrow ETH
- Borrow interest

Centralized Finance institution

CeFi

(e.g., Blockfi, Nexo, ...)

Crypto: CeFi lending
The role of collateral

CeFi’s concern: what if Bob defaults on loan?

⇒ CeFi will absorb the loss

Solution: require Bob to lock up collateral

CeFi's concern: what if Bob defaults on loan?

⇒ CeFi will absorb the loss

Solution: require Bob to lock up collateral

CeFi will absorb the loss if Bob defaults on loan.

Solution: require Bob to lock up collateral.

CeFi will absorb the loss if Bob defaults on loan.

Solution: require Bob to lock up collateral.

(1 ETH = 100 UNI)
The role of collateral

Several things can happen next:

(1) Bob repays loan

CeFi

repay 1 ETH

redeem UNI collateral (minus interest)

Bob

debt position:

+ 500 UNI

− 1 ETH

(1 ETH = 100 UNI)
The role of collateral

Several things can happen next:

(1) Bob repays loan
(2) Bob defaults on loan

I can’t repay 1 ETH
redeem remaining UNI collateral
(400 – interest – penalty) UNI

CeFi

Ok, I’ll keep
(100 + penalty) UNI

debt position:
+ 500 UNI
- 1 ETH

(1 ETH = 100 UNI)
The role of collateral

Several things can happen next:

1. Bob repays loan
2. Bob defaults on loan
3. Liquidation: value of loan increases relative to collateral

CeFi

I need to liquidate your collateral (and charge a penalty = 20 UNI)

lender needs to liquidate before value(debt) > value(collateral)

(1 ETH = 400 UNI)

debt position:
+ 80 UNI
− 0 ETH

1 ETH = 400 UNI
Terminology

**Collateral**: assets that serve as a security deposit

**Over-collateralization**: borrower has to provide
\[ \text{value(collarlateral)} > \text{value(loan)} \]

**Under-collateralization**: \[ \text{value(collarlateral)} < \text{value(loan)} \]

**Liquidation**: if \[ \text{value(debt)} > 0.6 \times \text{value(collarlateral)} \]
then collateral is liquidated until inequality flips
(liquidation reduces both sides of the inequality)
Collateral Factor $\in [0,1]$

- Max value that can be borrowed using this collateral
- High volatility asset $\implies$ low collateral factor
- Relatively stable asset $\implies$ higher collateral factor

Examples: (on Compound)
- ETH, DAI: 83%,
- UNI: 75%,
- MKR: 73%
Health of a debt position

BorrowCapacity = $\sum_i \text{value}(\text{collateral}_i) \times \text{CollateralFactor}_i$

(in ETH)

\[
\text{health} = \frac{\text{BorrowCapacity}}{\text{value}(\text{TotalDebt})}
\]

health < 1 $\implies$ triggers liquidation until (health $\geq$ 1)
Example: Aave dashboard (a DeFi lending Dapp)

- DAI is deposited as collateral
- UNI is borrowed
- The borrowing interests the borrower needs to pay
- In Aave, the collateral is also lent out. Hence the borrower can also earn interests.

Credit: Arthur Gervais
Why borrow ETH?

If Bob has collateral, why can’t he just buy ETH?

• Bob may need ETH (e.g., to buy in-game assets), but he might not want to sell his collateral (e.g., an NFT)

• As an investment strategy: using UNI to borrow ETH gives Bob exposure to both
The problem with CeFi lending

Users must trust the CeFi institution:

• Not to get hacked, steal assets, or miscalculate
• This is why traditional finance is regulated

• Interest payments go to the exchange, not liquidity provider Alice

• CeFi fully controls spread  (borrow interest – deposit interest)
DeFi Lending

Can we build an on-chain lending Dapp?

⇒ no central trusted parties

⇒ code available on Ethereum for inspection
A first idea: an order book Dapp

Order Book Protocol

LENDERS

- Supply Assets
- Receive Interest

BORROWERS

- Supply Collateral
- Borrow Assets
- Pay Interest

(large institutions, banks)

Credit: Eddy Lazzarin
Challenges

• **Computationally expensive**: matching borrowers to lenders requires many transactions per person (post a bid, retract if the market changes, repeat)

• **Concentrated risk**: lenders are exposed to their direct counterparty defaulting

• **Complex withdrawal**: a lender must wait for their counter-parties to repay their debts
A better approach: liquidity pools

Over-collateralized lending: Compound and Aave

Liquidity Providers (earn interest) → supply assets

Compound/Aave Dapps gather liquidity

Compounded/ Aave borrow assets
Example: Compound cTokens

Liquidity Provider

supply assets

10 ETH, 1000 DAI, 500 UNI

mints cTokens for Alice (ERC-20)

X cETH, Y cDAI, Z cUNI

Value of X, Y, Z is determined by the current exchange rate:
Token to cToken exchange rate is calculated every block

4 markets

Compound

DAI
ETH
UNI
AXS
Borrowers

4 markets

DAI

ETH

UNI

AXS

Bob’s accrued interest increases ETH/cETH exchange rate

⟹ benefit cETH token holders (ETH liquidity providers)
Consider the ETH market:

- **Supplying ETH**: adds to UnderlyingBalance\textsubscript{ETH}
- **Borrowing ETH**: adds to totalBorrowBalance\textsubscript{ETH}
- **Interest**: added repeatedly to totalBorrowBalance\textsubscript{ETH}

\[
\text{ExchangeRate}_{\text{ETH}/\text{cETH}} = \frac{\text{UnderlyingBalance}_{\text{ETH}} + \text{totalBorrowBalance}_{\text{ETH}} - \text{reserve}_{\text{ETH}}}{\text{cTokenSupply}_{\text{ETH}}}
\]

\[\implies\text{As totalBorrowBalance increases so does ExchangeRate}\]
The interest rate: constantly updates

**Key idea:** determined by demand for asset vs. asset market size

**Utilization ratio:**  
\[ U_{\text{ETH}} = \frac{\text{totalBorrowBalance}_{\text{ETH}}}{\text{availableBalance}_{\text{ETH}} + \text{totalBorrowBalance}_{\text{ETH}}} \]

higher totalBorrowBalance, or lower availableBalance in contract  

\[ \text{interestRate}_{\text{ETH}} = \text{BaseRate}_{\text{ETH}} + U_{\text{ETH}} \times \text{slope}_{\text{ETH}} \]
Example: Compound DAI market

Market Liquidity: 377,443,771 DAI

- # of Suppliers: 18,468
- # of Borrowers: 2,750
- Collateral Factor: 83%
- cDAI Minted: 26,810,077,978
- Exchange Rate: 1 DAI = 45.26986803778856 cDAI

Utilization:
- Current (40%)
- 60%

APYs at 60% utilization:
- Borrow APY: 3.82%
- Deposit APY: 1.93%

(Oct. 2022)
Liquidation: debt > BorrowCapacity

If user’s health < 1 then anyone can call:

liquidate(borrower, CollateralAsset, BorrowAsset, uint amount)

- address of borrower being liquidated
- Liquidator wants cTokens in this asset (e.g., cDAI)
- Liquidator is providing this asset (e.g., ETH)

This function transfers liquidator’s ETH into ETH market, and gives the liquidator cDAI from user’s collateral
Liquidation: debt > BorrowCapacity

If user’s health < 1 the anyone can call:

liquidate(borrower, CollateralAsset, BorrowAsset, uint amount)

This function transfers liquidator’s ETH into ETH market, and gives the liquidator cDAI from user’s collateral (at a discounted exchange rate -- penalty for user)

Liquidator is repaying the user’s ETH debt and getting the user’s cDAI (e.g., cDAI) (e.g., ETH)
What is liquidation risk?

Historical DAI interest rate on Compound (APY):

Demand for DAI spikes

\[\Rightarrow\] price of DAI spikes

\[\Rightarrow\] user’s debt shoots up

\[\Rightarrow\] user’s health drops

\[\Rightarrow\] liquidation ...

To use Compound, borrower must constantly monitor APY and quickly repay loans if APY goes too high (can be automated)
• Liquidity providers can earn interest on their assets
• DeFi lending usage:

**Compund outstanding debt**

(Oct. 2023) $815M
Compound liquidation statistics:

Caused by collateral price drops or debt APY spikes
What is a flash loan?

A flash loan is taken and repaid in a single transaction

⇒ zero risk for lender  ⇒ borrower needs no collateral

“Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit”
Use cases

• Risk free arbitrage

• Collateral swap

• DeFi attacks: price oracle manipulation
Risk free arbitrage

Alice finds a USDC/DAI price difference in two pools

Uniswap
USDC → DAI

1 USDC = 1.002 DAI

Aave (flash loan provider)

Flash loan 1M USDC

1M USDC

1.002M DAI

1.002M DAI

1.001M USDC

1 USDC = 1.001 DAI

Curve
DAI → USDC

Repay 1M USDC loan

keep 0.001M USDC

All in a single transaction
Collateral swap

start:
Alice @Compound

-1000 DAI
+1 cETH

borrowed DAI using ETH as collateral

Take 1000 DAI flash loan
Repay 1000 DAI debt
Redeem 1 cETH
Swap 1 cETH for 3000 cUSDC
Deposit 3000 cUSDC as collateral
Borrow 1000 DAI
Repay 1000 DAI flash loan

(a single Ethereum transaction)

end goal:
Alice @Compound

-1000 DAI
+3000 cUSDC

borrowed DAI using USDC as collateral
function flashLoan(address _receiver, uint256 _amount) {
    ...
    // transfer funds to the receiver
    core.transferToUser(_reserve, userPayable, _amount);

    // execute action of the receiver
    receiver.executeOperation(_reserve, _amount, amountFee, _params);

    ...
    // abort if loan is not repaid
    require(availableLiquidityAfter == availableLiquidityBefore.add(amountFee), "balance inconsistent");
}
## Flash loans amounts on Aave (in 2021)

<table>
<thead>
<tr>
<th>Date</th>
<th>FALSHLOAN_USD</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 22</td>
<td>624.5M</td>
</tr>
<tr>
<td>May 5</td>
<td>520.9M</td>
</tr>
<tr>
<td>May 21</td>
<td>515.0M</td>
</tr>
<tr>
<td>May 19</td>
<td>265.7M</td>
</tr>
<tr>
<td>Aug 3</td>
<td>163.7M</td>
</tr>
</tbody>
</table>
Next lecture: Decentralized Exchanges (DeX)