Lending Protocols

Martinet Lee

/Outline

- Lending Protocol System Architecture
- How to calculate interest?
- Liquidation
- When does a lending protocol fail / lose money?

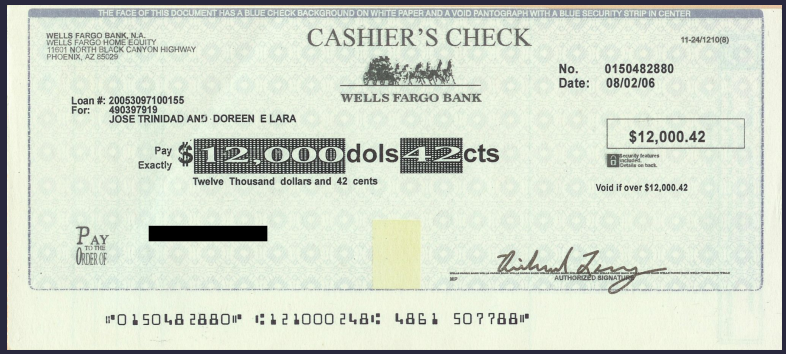
/Traditional Bank: Lending and Borrowing

/Traditional Bank: Risks to consider

- Depositor / Lender
  - What if the depositor put in fake money?
- Borrower
  - What if the borrower does not return money?
- Bank
  - How do the set interests?

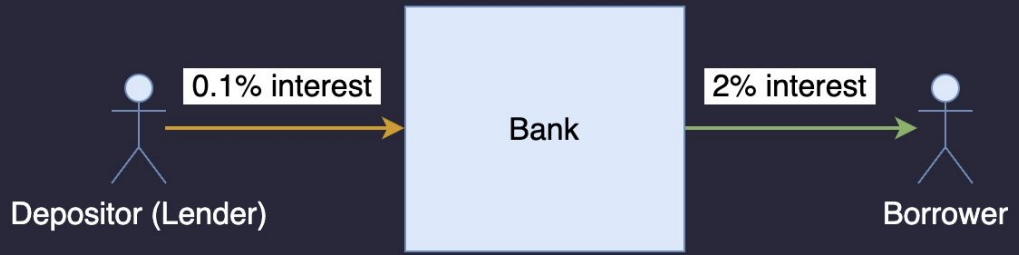
/Traditional Bank: Mitigating the risks

- What if the depositor put in fake money?



/Traditional Bank: Risks to consider

- What if the borrower does not return money?
  - Collateral
  - Reputation
  - Regulation



/Collateral

- The borrower guarantees the payment of loan by providing collateral.
- Who determines the “price” of the collateral?
  - Appraisers hired by banks
- Risks?
- What happens when payment is not there?

/Collateral Risks

- What if the collateral price drops a lot?
- What if price was incorrect..?

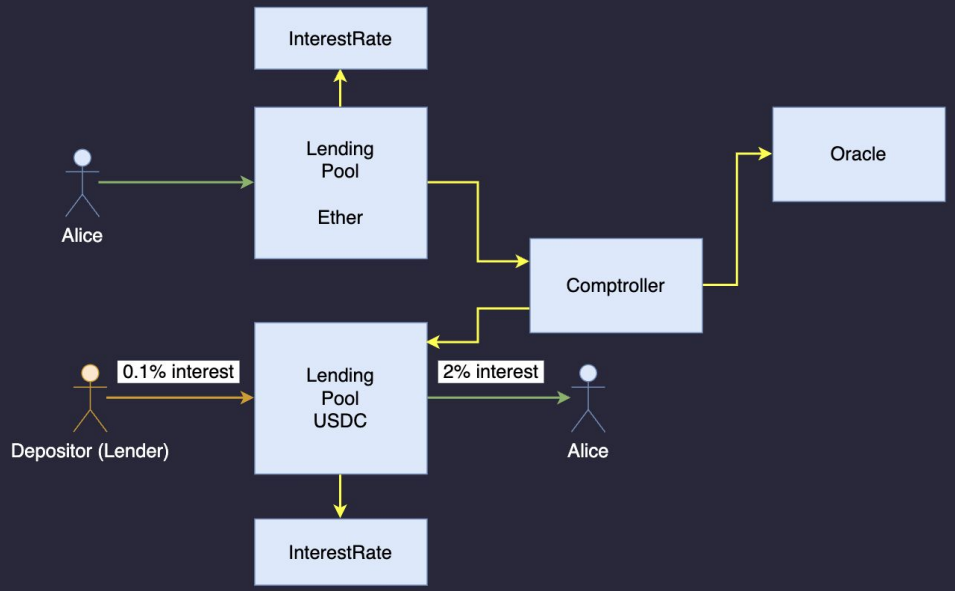
/From Banks to Lending Protocol

- User
  - Lender with ID ⇒ Anonymous
  - Borrower with ID ⇒ Anonymous, no reputation/regulation risk
  - Government ⇒ Liquidator
- Appraiser ⇒ Oracle provider
- Bank ⇒ Admin

/Lending Protocol: Collateral Risk

- What if the collateral price drops a lot?
  - Over Collateralized
  - Collateralization Factor
  - What does the protocol take in as collaterals is important. (Listing policy matters!)

/Rough Architecture of Compound



https://github.com/compound-finance/compound-protocol

https://docs.compound.finance/

/Liquidation in Compound: CToken

```
/**
 * @notice The liquidator liquidates the borrowers collateral.
 *  The collateral seized is transferred to the liquidator.
 * @param borrower The borrower of this cToken to be liquidated
 * @param liquidator The address repaying the borrow and seizing collateral
 * @param cTokenCollateral The market in which to seize collateral from the borrower
 * @param repayAmount The amount of the underlying borrowed asset to repay
 */
function liquidateBorrowFresh(address liquidator, address borrower, uint repayAmount, CTokenInterface cTokenCollateral)
```

/Liquidation in Compound: CToken

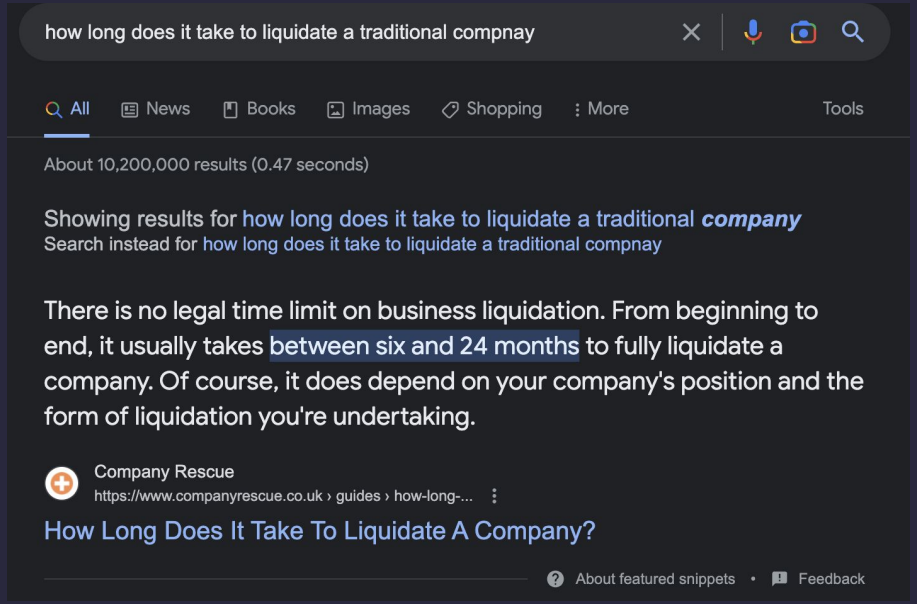
```
/* Fail if repayBorrow fails */
uint actualRepayAmount = repayBorrowFresh(liquidator, borrower, repayAmount);

////////////////////////
// EFFECTS & INTERACTIONS
// (No safe failures beyond this point)

/* We calculate the number of collateral tokens that will be seized */
(uint amountSeizeError, uint seizeTokens) = comptroller.liquidateCalculateSeizeTokens(address(this), address(cTokenCollateral), actualRepayAmount);
require(amountSeizeError == NO_ERROR, "LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED");

/* Revert if borrower collateral token balance < seizeTokens */
require(cTokenCollateral.balanceOf(borrower) >= seizeTokens, "LIQUIDATE_SEIZE_TOO_MUCH");

// If this is also the collateral, run seizeInternal to avoid re-entrancy, otherwise make an external call
if (address(cTokenCollateral) == address(this)) {
    seizeInternal(address(this), liquidator, borrower, seizeTokens);
} else {
    require(cTokenCollateral.seize(liquidator, borrower, seizeTokens) == NO_ERROR, "token seizure failed");
}
```

/TardFi v.s. DeFi: Liquidation

- How does Defi liquidate?
  - Fixed Discount ⇒ 1 tx
  - Auction ⇒ 2tx within hours depending on the code
  - Changing Discount ⇒ 1 tx
- How does TardFi liquidate?

/TardFi v.s. DeFi: Liquidation

- How does TardFi liquidate?
  - Tedious process
  - Long time ..



/When does a Lending Protocol fail?

- When the borrower will never return the borrowed funds.
  - Collateral value < Borrowed value
- How to make it fail?
  - Make collateral value very high temporarily to borrow lots of money
  - List a illiquid collateral value
  - Prevent people from liquidating

/Flashloan

- Since on blockchain, interactions to different protocols can happen in 1 tx - no time passed in between…
  - We do not need to require collateral if the loan is returned immediately.
    - Transfer funds out
    - Call external contract
    - “Require” Tokens being transferred back

/Applications of Flashloan

- Liquidation
- Arbitrage between DEXes in one tx
- Rebalance positions within / across protocols
- “Flashloan” attacks (?)

/THAT'S A WRAP!  (mic drop)