ABS Smart Contracts on The Ethereum Blockchain

Thore Hildebrandt
hildebrandtthore@gmail.com
https://github.com/ETHorHIL/TruffleSFContract

Abstract. A securitization performed by smart contracts on a blockchain would allow for the issuance of asset backed securities (ABS) bond like tokens which entitle token holders to receive cashflows from the underlying pool of receivables. Securitizations are effectively a set of contractual agreements between transaction parties. Smart contracts allow to engage in coded contracts with untrusted parties and to be certain over their execution. An ABS smart contract framework could reduce costs, risks, paperwork, and friction for all involved parties. The proposed smart contract prototype has been written for the Ethereum blockchain and it replicates the structure of a simplified securitization as a proof of concept.

Table of Contents

0. Introduction

Within the past couple of years, blockchain technology has evolved to allow for more complex use cases than of storing and transferring value. The Ethereum blockchain is an example of a next generation blockchain technology. It allows for smart contracts that establish trust between parties through the execution of codable contract logic. This paper serves as a supporting document to a prototype implementation of a smart contract that replicates a securitization.

1. Securitization

This section aims to give a quick introduction to the complex topic of securitization. There is plenty of material available on the web for further reading. Simply put, securitization is the practice of pooling assets like consumer loans or mortgages and selling the cash flows to investors as securities (bonds). Principal and interest cash flows collected from the underlying debt are redistributed in clever ways to achieve desirable properties for the issued bonds. Various kinds of structures exist, such as mortgage-backed securities (MBS) backed by mortgages, asset-backed securities (ABS) backed by consumer loans, auto loans, and student loans.

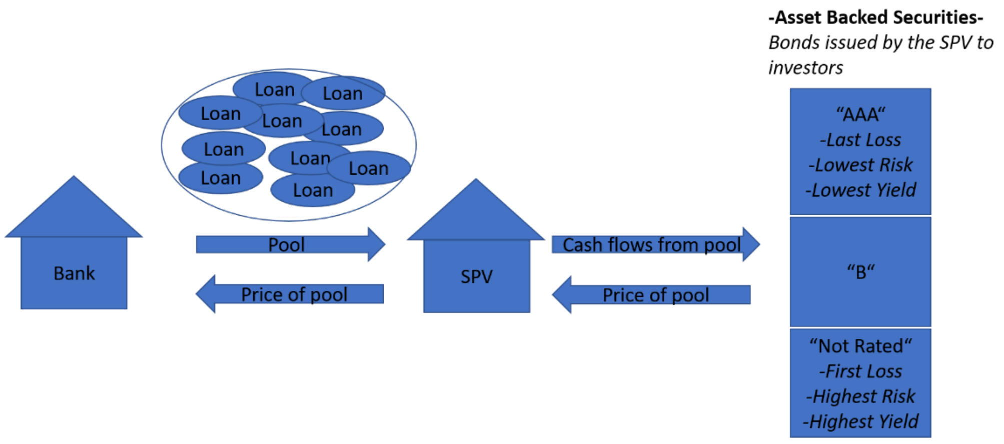
The entity that supplies the underlying assets, usually a bank, is called the originator. The pool of assets is sold to a "special purpose vehicle" (SPV), a company established specifically to fund the assets. The intention is to remove recourse to the originator; in other words, if the originator goes bankrupt, then its creditors don't have a claim on the assets sold to the SPV. This step enables the securities' credit quality (rating) to be de-linked from the originator. The originator typically needs the help of an investment bank to set up the transaction.

To be able to buy the assets from the originator, the issuer SPV issues tradable securities to fund the purchase. The performance of the securities is then directly linked to the performance of the assets. Credit rating agencies rate the securities to provide an external perspective on the liabilities being created and help investors make a more informed decision.

1.1 Credit Enhancement and Tranching

Securities created in a securitization are "credit enhanced," meaning their credit quality has been increased above that of the originator's unsecured debt or underlying asset pool. This increases the likelihood that the investors will receive the cash flows to which they are entitled, and thus enables the securities to have a higher credit rating than the originator.

The issued securities are often split into tranches. Each tranche has a different level of credit protection or risk exposure: there is generally a senior ("A") class of securities and one or more junior subordinated ("B," "C," etc.) classes that function as protective layers for the "A" class. The senior classes have first claim on the cash that the SPV receives, and the more junior classes only start receiving repayment after the more senior classes have been repaid. This arrangement is often referred to as a cash flow waterfall. In addition to subordination, credit may be enhanced through other measures like a cash reserve, third party insurance or over-collateralization.



1.2 Advantages and Disadvantages of Securitization

Advantages to originators:

-Reduced funding costs: For example, a company rated "BB" can issue debt at "AAA" rates.
-Reduced asset-liability mismatch: This a concern for banks to create self-funded asset books.
-Lower capital requirements: Accomplished through balance sheet engineering.
-Locked in profits: Future cash flows are being sold for immediate cash.
-Transfer of risks: Credit, liquidity, prepayment, reinvestment and/or asset concentration risk is being sold to parties willing to take the risk .
-Earnings: The sale boosts reported earnings on the balance sheet, which may be desirable for some managers.
-Liquidity: The cash from assets is available for immediate spending or investment.

Disadvantages to Originators:

-Potentially lower portfolio quality: If the "AAA" risks, for example, are being securitized out, this results in the remaining portfolio having materially worse credit quality.
-Costs: Securitizations are expensive due to management and system costs, legal fees, underwriting fees, rating fees and ongoing administration.
-Size limitations: Securitizations often require large-scale structuring, and thus may not be cost efficient for small and medium-sized transactions.
-Limited target-investors: The universe of eligible investors is limited.

Advantages to investors

-Opportunity to potentially earn a higher rate of return: (on a risk-adjusted basis)
-Opportunity to invest in a specific pool of high quality assets: Risk-averse institutional investors, or investors who are required to invest in only highly rated assets, have access to a larger pool of investment options.
-Portfolio diversification: Securitizations may be uncorrelated with other bonds and securities.

Risks for investors:

-Credit default: Defaults on the pool of loans
-Prepayment/reinvestment/early amortization: Amortization risk stems from specific early amortization events or payout events that cause the security to be paid off prematurely.
-Currency interest rate fluctuations: Like all fixed income securities, the prices of ABS move in response to changes in interest rates.
-Servicer risk: The transfer or collection of payments may be delayed or reduced if the servicer becomes insolvent.

2. Blockchain

Just like the previous chapter, this one aims to give a quick introduction to a highly complex topic, namely the blockchain. Plainly put, a blockchain is a database, or rather a distributed database, which functions as a decentralized, digital version of "the truth" that is validated, ever-growing, and cannot be tampered with. The protocols of the blockchain solve the so-called "double-spending problem," i.e. the technology ensures that no conflicting second version of the true history emerges.

The best-known application of blockchain technology is the cryptocurrency payment network Bitcoin, where the blockchain serves as the public ledger for all historically occurred transactions. Knowing the true history of all historically occurred transaction gives certainty about the current Bitcoin balances of each participant. The blockchain solving the double spending problem, in this case, means that one cannot spent a Bitcoin twice and thereby create value out of thin air. Everybody can easily connect to the Bitcoin network, send new transactions, verify transactions, and take part in the competition to create new blocks. The competition creating new blocks is known as mining.

A blockchain implementation consists of two kinds of records: transactions and blocks. *Blocks* hold batches of valid transactions. Each block includes the "hash" (a cryptographic fingerprint) of the prior block in the blockchain, linking the two. The linked blocks form a chain, hence the name Blockchain. The special way new blocks are added to the chain and validated leads to a consensus of the network on the data stored in the blocks. The more blocks are built on any one block, the more unlikely it is that there will ever be a change to its content.

The most important cryptocurrencies by market capitalizations as of Jan. 1, 2017:

#	Name	Market Cap
1	Bitcoin	$42,540,577,632
2	Ethereum	$22,701,485,836
3	Ripple	$11,367,600,419
4	NEM	$2,072,151,000

2.1 Decentralization

By storing data across its network, the blockchain eliminates risks that come with data being held centrally. Its network lacks centralized points of vulnerability that computer hackers can exploit. Blockchain security methods use encryption technology. The basis for this are the so-called public and private "keys." A "public key" (a long, randomly generated string of numbers) is the users' address on the blockchain. Bitcoins sent across the network gets recorded as belonging to that address. The "private key" is like a password that gives its owner access to their Bitcoin or other digital assets. Data stored on the blockchain is incorruptible. Every node in a decentralized system has a copy of the entire blockchain. No centralized "official" copy exists and no user is "trusted" more than any other.

2.2 Ethereum: Blockchain 2.0 the World Computer

Ethereum is a public blockchain-based distributed computing platform. In contrast to Bitcoin its purpose isn't merely to store the cryptocurrency, but to allow for smart contract functionality. More specifically, it provides a decentralized virtual machine that can execute code in a peer-to-peer manner. Ethereum has been described by the New York Times as "a single shared computer that is run by the network of users and on which resources are parcelled out and paid for by Ether."

2.3 Smart Contracts

A smart contract is computer code stored on blockchain accounts. The code can be executed by sending a message to it. The smart contract code can represent any logic and is immutable once deployed.

Smart contracts are the next step in the blockchain evolution from simply keeping a record of value entries to automatically executing terms of multiparty agreements. This results in a method by which parties can agree upon terms and have certainty that they will be executed automatically.
Before the blockchain, smart contracts were impossible because parties to an agreement maintained separate databases. With a blockchain, the smart contracts auto-execute, and all parties validate the outcome instantaneously without need for a third-party intermediary.

Some Smart Contract Benefits are: speed and real-time updates, accuracy, security, transparency, low execution risk, few intermediaries, low cost, and ease of implementation.

Use Cases in discussion are amongst others: Trade clearing and settlement, coupon payments, insurance claim processing, microinsurance, internet of things, electronic medical records, population health data access, royalty distribution (media), autonomous electric vehicle charging stations, public-sector record keeping, supply chain and trade finance documentation, product provenance and history, peer-to-peer transactions, voting and prediction markets

2.4 Tokens

The currency that can be transferred on the Ethereum blockchain is not only the native Ethereum currency Ether (ETH). In fact, the ability to mint and issue tokens for all kinds of purposes is an integral strength of the Ethereum blockchain. In this paper, the ABS bonds issued to investors are tokens which entitle the token holder to receive cashflows from the pool.

2.5 Issues

Two big issues for any blockchain base technology are scalability and privacy. One additional issue related to using smart contracts on a public blockchain is that bugs, including security holes, are visible to all but cannot be fixed quickly. One example of this is the June 17, 2016, attack on the Decentralized Autonomous Organisation (DAO), a Venture Capital Fund that consisted entirely of smart contracts and was run on Ethereum. The “buggy” contract was fixed with a controversial system update that restored the hacked transaction on the Ethereum blockchain.

2.6 Practical Example of a Smart Contract

Interacting with the blockchain is easy. The user only needs to install a software (a client) which will connect to the network an download a copy of the blockchain. Once this is done the user can create an account on the blockchain. The account consists of an address that looks like this: “0xde0B295669a9FD93d5F28D9Ec85E40f4cb697Bae” and an arbitrary password which unlocks the account. The next step is to buy some of the cryptocurrency (ETH) on an exchange and send it to the account. ETH serves as the fuel for the smart contracts and small portions are consumed when a smart contract is executed. The software downloaded also allows to send ETH to other accounts, to deploy the code of a smart contract, and to interact with already deployed smart contracts. The example code below can be deployed to its own account on the blockchain, it allows anyone to send ETH to the contract but only allows “Bob” and “Joe” to withdraw funds.

```
contract Example
{
   address Bob;
   address Joe;

   function Withdraw () public
   {
      if (msg.sender ==Bob || msg.sender ==Joe) msg.sender.send(this.balance);
   }

   function () public payable {}

   function Example (address _Bob, address _Joe) {Bob=_Bob; Joe=_Joe;}
}
```

3. An ABS Smart Contract

A working proof of concept of the ABS smart contract discussed below has been written in the Ethereum smart contract programming language (Solidity). The goal of this chapter is to give a high-level overview of this smart contract framework that replicates a basic securitization.

The ABS smart contract is in fact a set of contracts:

-The Smart Loan Contract represents an instance of a single asset to be securitized
-The Securitization Contract provides an interface to participants and schedules processes
-The Escrow Contract makes sure participants can trust each other during the funding period
-The Assetleger Contract manages the pool of assets
-The WaterFall Contract allocates cashflows from the pool to the bonds per bond terms
-The Bond Contract represents the bonds

On a high level, the process looks like this:

1) Bank originates SmartLoans
2) Bank advertises the sale of a pool of SmartLoans to investors
3) Bank deploys the Securitization contract
4) The Securitization contract deploys the Escrow contract and Ledger contract
5) Escrow Contract waits for funds from investors, pool, reserve, etc. to be posted
6) When Escrow contract is satisfied, the steps below are followed, if not abort
7) The Waterfall and Bond contracts are deployed
8) Funds start flowing from the Smartloans to the Ledger, then over the Waterfall to the Bonds
9) The transaction ends when all assets are redeemed, final maturity is reached or the originator calls the transaction
10) Investors can withdraw funds from their Bonds and transfer the Bonds to parties

For simplicity, we assume that the loans to be securitized are smart loans, i.e. smart contract loans that are repaid by the borrower on the blockchain.

3.1 The SmartLoan Contract

The SmartLoan contract represents a simple fixed installment loan which the originator grants to its customers. The originator sends the loan balance to the borrower and deploys the contract code on the blockchain where the borrower is repaying it. The code allows to track the loan, to make payments to the loan and to withdraw funds from it. The loan will notify the originator/servicer when the borrower is delinquent. The SmartLoan contract needs the following parameters to get passed along with deployment:

lenderAddress: This is the address of an account on the blockchain. The account can be controlled by a human or another contract. Whoever controls this account has the right to withdraw paid in funds from the Smartloan contract.
balance: The principal balance of the loan which must be repaid by the borrower.
interestRateBasisPoints: The (fix) interest rate charged on the loan. It is given in basis points, i.e. the input for 5% interest would be 500
termMonths: The term of the loan in moths.

Once the contract is deployed, it exists on the chain and provides the following functions:

Transfer()
This function is essential for transfer of ownership of the loan. The *lenderAddress* owner has the option to grant its rights to withdraw funds from the loan to another party. In our case, this will be another contract on the blockchain to which the originator "sells " the loan.

PayIn()

Allows the borrower to pay an installment to the loan. The installments are of equal size depending on the interest rate, original balance, and loan term.

WithdrawIntPrin ()
Allows the *lenderAddress* owner to withdraw funds from the contract. This function also passes along the status of the loan (see below).

These variables represent the status of the loan:

-LenderAddress;
-OriginalBalance;
-CurrentBalance;
-InterestPaidIn;
-PrincipalPaidIn;
-Size of MonthlyInstallment;
-InterestRate in BasisPoints;
-OriginalTermMonths;
-RemainingTermMonths;
-A list of the PaymentDates;
-Next PaymentDate;
-OverdueDays;
-Is the ContractCurrent?;

3.2 The Securitization Contract

The Originator (i.e. the Party that wants to sell a pool of assets to investors) deploys the contract code of the "Securitization Contract" on the blockchain and needs to specify parameters:

Poolinformation:

accountAddresses: the addresses of all SmartLoans in the pool to be sold
OriginalPoolBalance: the aggregate balance of the loans in the pool to be sold
NumberOfLoans: the number of loans in the pool

Sale Price of the Pool (can also be determined in an auction):

ClassAInitialBal: The aggregate balance of class A Bonds to be issued
C*lassAInterestRateBPS*: The interest rate which class A Bondholders receive
ClassBInitialBal: The aggregate balance of class B Bonds to be issued
ClassBInterestRateBPS: The interest rate which class A Bondholders receive

Other Information:

ReserveRequired: The originator can specify a reserve to increase credit enhancement for bonds.
InvestmentPeriodEnd: The duration of the period in which investors can subscribe to bonds. If funding is successful the structure will start, if not investors can reclaim their investment.
TrustedParty: The address of a third party which is accepted to conduct an audit/analysis of the pool and structure.
ExcessFundsReceiver: The address off a party which can withdraw excess funds from the structure.

After deployment of this contract, the smart contract lives on the blockchain and is the central part of the whole structure. It provides functions to deploy the other contracts (except the SmartLoan contract) and all internal processes are triggered from this contract.

The following functions are the most important:

CreateEscrowAndLedger()

Creates two contracts: A ledger, which packages a pool of SmartLoans and allows it to interact with this pool (withdraw funds and read information) and an Escrow contract, which waits for all transaction prerequisites to be satisfied:
-Pool of loans has been transferred into the ledger as specified
-Investor funds are received
-A third party has reviewed the pool and structure and approved claimed properties
-The reserve has been received as specified

CheckEscrow()
checks if the conditions of the above created escrow contract are satisfied

CreateBondsAndWaterFall ()
Once the escrow conditions are satisfied, several other contracts are created:
-Bond contracts for each class of investors that mint Bond tokens and allow the token holder to withdraw funds from the transaction.
-A WaterFall contract, which takes funds and cashflow information from the AssetLedger and calculates how the funds are to be distributed among bondholders.

MoveFundsFromPoolIntoLedger ()
Allows to sweep funds from the individual SmartLoans in the pool into the AssetLedger.

MoveFundsFromLedgerToWaterfall()
Allows to move funds from the Ledger to the WaterFall.

MoveFundsIntoBonds ()
Allows to sweep funds from the WaterFall into the Bond token contract.

3.3 The AssetLedger Contract

The AssetLedger manages and keeps track of all the SmartLoans contracts sold by the originator. At creation of the contract, it is supplied with the following information:

AccountAddresses: A list of all the addresses of SmartLoans sold by the originator
OriginalPoolBalance: The aggregate balance of all SmartLoans at the time of transferral
NumberOfLoans: The aggregate number of SmartLoans transferred
Controller: This is the address of another contract that can give instructions to this contract. This is the Securitization contract mentioned above

Once the contract is created by the Securitization contract, it provides the following functions:

WaterFallset ()
This function tells the AssetLedger contract who is allowed to withdraw funds. This is the WaterFall contract explained below.

PoolTransferred ()
This function is used by the Escrow contract (see below) and verifies that the pool of loans for sale has been transferred as advertised.

WithdrawDueLoans()
This function withdraws funds from SmartLoans that have become due since the last withdrawal. Withdrawals are being made in a scheduled frequency. One important question to be considered and tested is how the limitations of the blockchain (block size and computational cost) affect this function when thousands of loans would be due at once. After withdrawal of funds, this function updates a ledger that keeps track of each individual loan as well as aggregate information of the pool. If this function fails to withdraw funds sufficiently often from a loan,

it decides whether the loan is to be considered defaulted. Funds received from defaulted loans are classified as recoveries.

SendFunds()
This function lets the controller address (Securtization contract) send the funds stored in this contract to the WaterFall contract. Along with the funds, status information is being passed:

-PrincipalThisPeriod
-InterestThisPeriod
-RecoveriesThisPeriod
-DefaultThisPeriod

PoolSnapshot()
Provides a snapshot of the aggregate pool as well as the status of individual loans. This function can be used for real time reporting of the pool and allows investors or third parties to automate their processes.

3.4 The Escrow Contract

The Escrow contract is responsible for establishing trust between investors and the originator, i.e. the seller of the pool of loans. It makes sure that the pool for sale is being transferred to the AssetLedger contract (see below) as advertised. It also ensures that the reserve was sent as advertised. Within the investment period, it waits for investments from investors and it allows a third party to confirm the advertised quality of the pool on behalf of the investors. If sufficient funds have been received, then the transaction kicks off. If funding fails, every party can claim back its investment. In future versions, it would be possible to implement an auction for investors to determine the capital structure and pricing of the bonds.

Upon deployment on the blockchain, the Escrow contract requires the following information, which is provided by the previously deployed Securitization contract:

poolAccounts: This is the list of addresses of SmartLoans in the pool for sale
requiredFundsA: The notional amount of class A bonds required to sell the pool
requiredFundsB: The notional amount of class B bonds required to sell the pool
reserveRequired: The reserve to be posted by the seller of the pool to improve credit quality
investmentPeriodEnd: The date at which the Escrow contract decides whether the transaction failed or will kick off
trustedParty: This is the address of the third party which analyses the pool on behalf of the investors
owner: The address of the controlling contract. This is the Securitization contract which deployed this contract and controls its behaviour
originator: The address of the originator (seller of the pool)
waterFall: The address of the Waterfall contract which is responsible for distribution of the funds.

The contract provides the following functions once deployed:

PoolTransfer ()
This function checks with the AssetLedger contract if the advertised pool has been received.

InvestorPayInA() / InvestorPayInB():
These functions are available to investors. They allow for investments being made up to the investment limit for each class. The investments are being stored in the contract and mapped to the accounts from which they have been received.

PayReserve()
Lets the originator pay the reserve as advertised.

GetInvestorsBalanceA() GetInvestorsBalanceB()
These functions are used later when Bond tokens are minted. They tell the bond token contract the amount of funds received from each investor.

PoolValid()
Allows the third party, which performed an analysis of the pool, send a validation.

CheckState()
This function checks and returns the status of the escrow contract.

RevertInvestment()
If the transaction is deemed to have failed after the investment period, this function reverts the investments of investors and sends back the reserve and pool to the investor.

OriginatorWithdrawFunds()
This function allows the originator to withdraw the funds invested by investors after all requirements for the transaction to kick off have been fulfilled.

3.5 WaterFall Contract

The WaterFall contract is responsible for distribution of the funds collected by the AssetLedger to the Bonds. A very simplified allocation has been used here for illustration purposes. Third parties, e.g. for asset servicing or additional credit enhancement, can be paid from this contract by adding their fee into the waterfall logic.

Upon creation of this contract, it needs the following information which its creator (the Securitization contract) provides:

ReserveAmount: The reserve fund is provided by the originator and improves credit quality
classAInitialBal: The aggregate balance of class A bonds
classAInterestRateBPS: The interest rate paid to class A bondholders
classBInitialBal: The aggregate balance of class B bonds
classBInterestRateBPS: The interest rate paid to class B bondholders
assetsAddress: The address of the AssetLedger contract where funds come from
liabilitiesAddressA: Address of the contract responsible for class A bonds
liabilitiesAddressB: Address of the contract responsible for class B bonds
ownerAddress: Address of the "Securitization" contract, which is the controlling contract
excessFundsAddress: The address where excess funds go to. Excess funds are interest funds received from the asset ledger after all deductions for Bond interest and top up of the reserve.

The following functions are available after the contract is deployed.

CalcWaterFall()
This function calculates how the funds are being distributed among the bonds. The waterfall structure used here is simplified for demonstration purposes. Therefore, first interest due is calculated, thereafter the available cash is applied in the following sequence:

1) Interest to A Bonds
2) Interest to B Bonds
3) Top up of reserve (the reserve covers interest shortfalls during the life of the transaction and is available for credit enhancement at the end of its life)
4) Principal to A Bonds
5) Principal to B Bonds

SendFundsA()/ SendFundsB()

Sends funds due to the A (B) bonds as calculated by the waterfall. This function is executable from the Securitization contract in a scheduled frequency. Along with the funds, the information on split of interest and principal is being passed.

Withdraw()
This function draws the funds from the asset ledger and is executable by the Securitization contract in a scheduled frequency.

3.6 The Bond Contract

The Bond contract is a bond token contract. It allows its creator to mint an arbitrary fixed supply of tokens and distribute the tokens to address owners on the blockchain. The token holders have the right to withdraw funds directly from the Bonds contract depending on the fraction of overall supply the token holder possesses. The Bond contract keeps track of inflows and outflows of funds. The A Bonds and B Bonds are being derived from this more general Bonds contract.

The Bond contract requires the following inputs to be deployed:

initialSupply: This is the initial number of tokens to be minted. The number of tokens will equal the notional amount of the respective class, i.e. if Class A is $1 million in total, then 1 million tokens will be minted.
waterfallAddress: The address of the WaterFall contract from which the funds can be drawn to the bonds.

Once deployed per the parameters above, the contract provides the following functions:

Transfer()
This function lets any token holder transfer all or part of his tokens to some other address on the blockchain if he chooses to do so. In other words, he or she can sell the bonds to any other person with this functions.

Withdraw()
Allows token holders to withdraw their portion of interest and principal funds from the Bond contract.

PayIn()
This function allows the Securitization contract to sweep funds from the Waterfall contract into the Bond contracts. This is being done on a predefined frequency.

CheckStatus()
This function allows to check the status of the overall Bond contract as well as the status of individual token holders.

4. Summary

A transaction that uses a smart contract as described above would have many benefits. The effort to setup a standardized transaction could be reduced in terms of time and cost needed for legal and investment banking services. Third parties for asset reporting, calculation and trustee services are not needed, execution is certain, and the execution cost are negligible. The period of cash being trapped in the structure is reduced to a few minutes. In traditional transactions, it can take several weeks for an installment received from a borrower to be allocated to investors. Cash does not have to sit in bank accounts and is thereby not exposed to the risk that the bank is going into bankruptcy. Pool data can be queried directly from assets at any time; therefore, data quality is maximally reliable. Real time updates of high-quality asset performance data enable automated credit quality updates. The investor universe could be broadened, since micro investments are possible and reduced cost can enable smaller originators to securitize assets. The concept could be applied to peer-to-peer lending and help solve the supply problem since individual lenders would not have to rely on their own estimation of credit risk.

A big challenge is adoption and a lack of experience, several steps need to be taken before investors would be willing and able to participate. Legal questions about the treatment of smart contracts are unaddressed. What happens if one party tries to reverse a transaction executed by a smart contract through a “real world” court? Legislation must reach global consensus to be effective. The most likely short-term solution is supporting documentation that handles edge cases and connects the smart contracts to current legislation. To be a feasible option for institutional investors and originators, tokens and other blockchain assets need to comply with many regulatory frameworks. Important motivations for securitizations are eligibility as repo collateral and balance sheet engineering. Several other services and institutions need to be in well-established existence before this setup is viable. The technology is still young. It may be faced with unprecedented attack vectors and we may see advances in technology that render the existing structures obsolete.

5. Conclusion

It is possible and relatively easy to implement an ABS smart contract prototype that provides basic functionalities. The next step would be to perform thorough security testing and add advanced functionalities. A fully functional and secure ABS smart contract could provide many benefits for investors and originators. However, such a smart contract would have to build on many, not yet existing, blockchain services and “real world” legal frameworks.