# Using Smart Contracts and Blockchains to Support Consumer Trust Across Distributed Clouds

**Stephen S Kirkman**[1] **and Richard Newman**[1]
[1]Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 32611, USA
*kirkman@ufl.edu (contact author), nemo@cise.ufl.edu*

**Abstract**— *name of conference: LATE BREAKING PAPER. In this paper, we propose to add blockchains as a mechanism to store cloud attestations. Blockchains are: 1) cryptographically auditable, 2) append only, 3) accessible to all, 4) tamper resistant. Blockchains also require no central trust mechanism (hence, no central point of failure). Smart contracts are a recent advance of blockchain technology that allow for more expressive development and control. Our smart contract gives the consumer the ability to query the blockchain for the location of their data and history of its movement between clouds.*

**Keywords:** Cloud Trust, Cloud Security, Distributed Clouds, Blockchain, Ethereum, Smart Contracts

## 1. Introduction

There is still significant lack of trust in the cloud.[1][2] *Inter-cloud VM migration* significantly impacts consumer trust in the cloud if our data are moved without our consent.[3][4] Due to multiple organizations joining federated clouds, it is hard to know which organization has control of our data.

This paper represents an extension of our research[5] into a policy framework to express the consumer's desires inspired by ORCON. Originator control access control (ORCON) is about data access and control. We used policies inspired by ORCON as a fundamental design philosophy to tackle trust across distributed clouds. We based our framework on four key components:

1) ORCON policy model to *express* our desires
2) A policy *tag* for the data
3) Verification of data movement in the form of *verifying* attestations.
4) Cloud provider to *willingly* commit to agree to comply with consumers policies and participate in the attestations

In this paper, we propose to add blockchains as a mechanism to store cloud attestations. In particular, we propose the use of smart contracts. Smart contracts are a recent advance of blockchain technology that allow for more expressive development and control. The main contribution of our research is to extend our distributed cloud trust policy framework by using smart contracts to store and retrieve attestations for data movement. These are stored on a blockchain.

The rest of this paper is organized as follows. Section II discusses the background of blockchains. Section III covers a next generation blockchain: Ethereum. Section IV proposes our attestation architecture and tests. Section V discusses our results. Section VI is a discussion on transactions and cost. The remaining sections discuss related work, further research, and wrap up with our conclusion.

## 2. Blockchain Background

We propose blockchains for attestation storage. Blockchains are: 1) cryptographically auditable, 2) append only (blocks cannot be removed), 3) accessible to all, and 4) tamper resistant. Blockchains also requires no central trust mechanism, hence, have no central point of failure. In a distributed cloud environment, we want to ensure that if data cross organizational boundaries, then they are tracked and there is no way to manipulate any logs when stored.

Blockchains were made popular by Bitcoin.[6] A blockchain is a public ledger that is computationally infeasible to alter once set.[7] Blockchains are formed through one-way hashes; hashes are computationally irreversible. A blockchain may be used as a chronological store of transactions.

For Bitcoin, an electronic coin is defined as a chain of digital signatures where owners can transfer coins using a digital signature and a hash of the previous transaction.[6] This is illustrated in figure 1 from Nakamoto's original paper. The blockchain is comprised of individual hashed blocks. Each block contains information about transactions, a reference to the preceding block, and an answer to a complex mathematical puzzle. The math puzzle is termed "proof of work." The proof of work is used to validate the data in the block and is the consensus mechanism used to determine the validity of a transaction.[8] The block is added after a majority of computers on the network reach consensus regarding the validity of the transaction. The blockchain grows as new blocks are added to the blockchain. Once added, the block can no longer be deleted. The blockchain is stored on every computer in the participating network and is therefore globally visible.

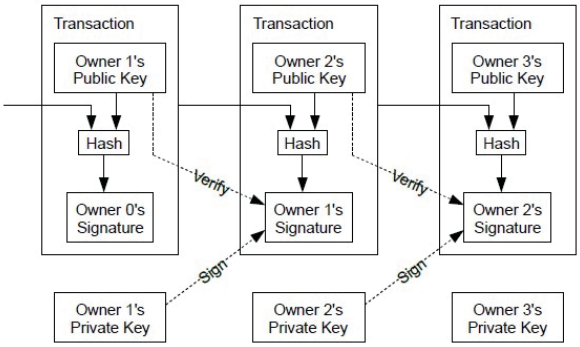


Fig. 1: BitCoin chain of Ownership (From Nakamoto[6])

Proof of work is the driving force behind the creation of the blockchain because it incentivizes people (i.e. miners) to solve complex problems in exchange for digital money. Solving these cryptographic problems serves to verify new blocks to be added to the blockchain. The first miner to solve the problem is rewarded with this currency.[8]

# 3. Ethereum

We have chosen Ethereum[9] as the blockchain in our research. Ethereum is a next generation blockchain technology whose aim is to build a general trustless ledger with the ability to run programs called "smart contracts." It uses a virtual machine on top of the blockchain called the ethereum virtual machine (EVM). This is a key differentiator from Bitcoin. The EVM is used to run the 'smart contracts' which can be considered transactional programs. It also has a different design philosophy than Bitcoin. "Ethereum is 'Turing complete' meaning that developers can create applications that run on the EVM using friendly programming languages modelled on existing languages like JavaScript and Python."[9]

Payment in the form of gas is required for transactions that change the blockchain. These are paid in units of "ether" (more on this later).

## 3.1 Smart Contracts

There are two kinds of accounts in Ethereum: 1) those that represent users, and 2) those that represent contracts. Smart contracts are always running on the blockchain once they are deployed; they are essentially programs. They are waiting idle and activated when they receive a transaction or query. Both contracts and users have their own addresses. Once deployed a contract can accept a transaction and manipulate the state of the blockchain. "All transactions (state transitions) are traceable and can be replayed by any node in the system. In fact they are replayed by any node downloading the blockchain from scratch."[10]

The contract is programmed by the developer, tested, then deployed to the blockchain. Smart contracts are both triggered and deployed by a *transaction*. The contract becomes a permanent part of the blockchain when enough miners have verified the proof of work.

## 3.2 Attestation Smart Contract

We developed a proof of concept for a smart contract to be used in our proposed attestation framework. Our smart contract is designed to store attestations on the blockchain. Once deployed, this smart contract performs two functions (we envision the potential for more later). One function is activated by the cloud who is storing data movement onto the blockchain. Each transaction changes the blockchain so there are transaction fees. The second function is designed to only report what is stored based on a hash of the consumer's public address.

We used the Solidity[9] programming language. Since Solidity is currently in a state of rapid change, we expect further development on our smart contracts. We used a simulated blockchain that runs on the local computer called testrpc[11]. The flow of work for logging a migration and for querying for attestation proceeds as follows

**Cloud:**

$$Intercloud.migrate(consumer, CloudB)$$

- Migrate consumer data: $CloudA \rightarrow CloudB$.
- CloudA will send a transaction (associated with the consumer) to the smart contract on Ethereum network.
- The cloud must participate in the Ethereum network.
- When a new block is created by the Ethereum miners, the new transaction will be logged onto the blockchain.

**Consumer:**

$$Intercloud.getAttest.call(consumer)$$

- When a consumer wishes a data location attestation, they make the request to the contract.
- If the consumer address matches the requested hash, the attestation will be returned, otherwise no data are returned.

There would be no attestation fees to merely read from the Ethereum blockchain. This is termed using a *call*. "When called using *call* the function is executed locally in the EVM and the return value of the function is returned with the function. Calls made in this manner are not recorded on the blockchain and thus, cannot modify the internal state of the contract. This manner of call is referred to as a constant function call. Calls made in this manner do not cost any ether."[12][13]

The code below represents the start of our vision of an InterCloud attestation. We use a dynamic hash table to store consumer data migrations. The consumer blockchain public address is the key for the hash table.

**Algorithm 1** InterCloud Migration Smart Contract

```
1: pragma solidity ^0.4.8;
2: contract InterCloud
3: address[] unauthRequest; // Empty Address for Privacy
4: mapping(address => address[]) public attestations;
5: function MIGRATE(address _consumer, address _receiver)
6:     attestations[_consumer].push(msg.sender);
7:     attestations[_consumer].push(_receiver);
8: function GETATTEST(address _consumer) returns(address[] addresses)
9:     if (msg.sender == _consumer) then
10:        addresses = attestations[_consumer];
11:    else
12:        addresses = unauthRequest;
```

Our smart contract only allows a consumer to request information on their own data. We do not store the details of the data, just the fact that data were migrated. However, it should be remembered that the blockchain itself is public; therefore the data that is stored in the blockchain is available for every miner to verify via the Ethereum protocol. At this juncture, we have not built any additional privacy into the contract.

# 4. Proposed Attestation Architecture

Figure 2 illustrates our proposed attestation architecture. A peer to peer network (in our case Etherum) forms the glue that enables both clouds and consumers to achieve the mutual goal of trust. In order to store attestation information or retrieve attestation information, both the cloud and the consumer respectively must both be a willing participant in the peer to peer network.

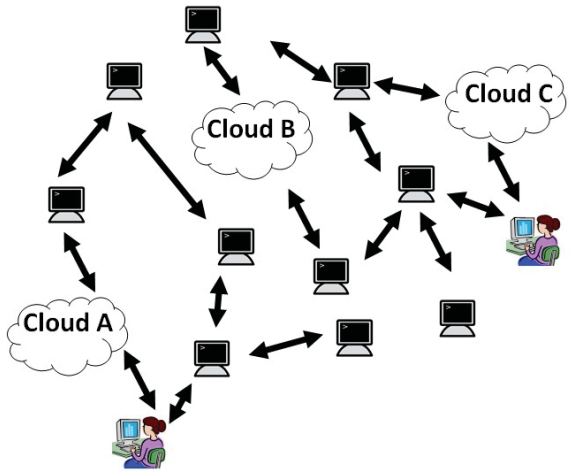


Fig. 2: Attestation Architecture using Ethereum P2P Blockchain

## 4.1 Smart Contract Testing

For our test hardware, we used a laptop with AMD A8-7410 at 2.2 Ghz with 8Gb RAM. We used the Solidity programming language for smart contracts, the RPC test blockchain, and Truffle development tool.

Table 1: Software and Version

| Software | Version | Use |
|---|---|---|
| Solidity[9] | 0.4.8 | Smart Contract Language |
| TestRPC[11] | 3.0.5 | Test Ethereum Blockchain |
| Truffle[14] | 3.2.1 | Testing & Compiling Environment |
| Visual Studio Code[15] | 1.12.1 | Editor |
| Solidity Online Compiler[16] | 0.4.11 | Realtime Compiler plus Gas Estimator |

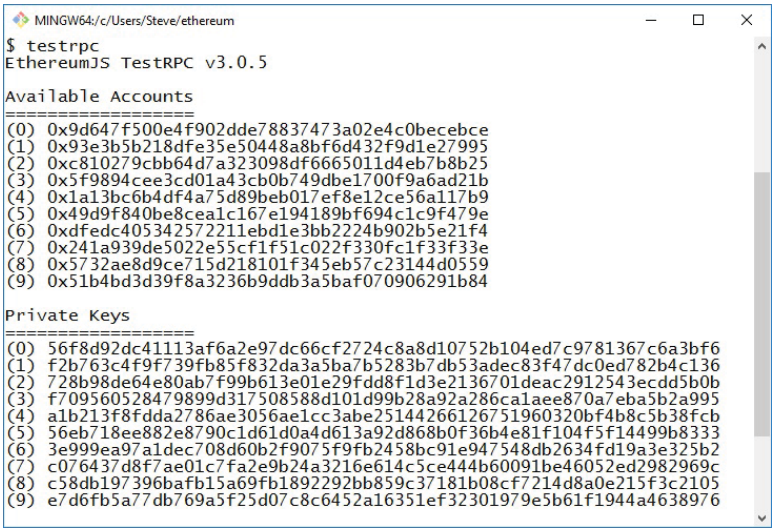


Fig. 3: 10 Public Account Addresses - TestRPC


Fig. 4: Cloud Mnemonics with Truffle

## 4.2 Test Procedures

We used testrpc[11] along with Truffle[14] to test our smart contract. The test blockchain automatically provides 10 accounts. We used these accounts to simulate sending transactions to the blockchain.

We also assigned meaningful cloud names to the addresses for easy reference during our tests. We set the test addresses to cloudA, cloudB, and consumer. Figures 3 and 4 show our mnemonics and default accounts.

To use a smart contract there are two main steps: 1) develop and debug, 2) migrate (deploy) to blockchain. As shown in the following figure, after developing and troubleshooting smart contract code, it is necessary to compile

it and deploy it to the blockchain. In our case we deploy it to our test blockchain.

# 5. Results

In the test run shown in Figure 5 at the end, we deployed the smart contract, ran simulated migrations, and finally executed an attestation request. On an operational blockchain, it would take time for the miners to validate new transactions before attestation data would become available on the blockchain.

Cloud A migrates consumer data to cloud B (assuming the consumer's policy allows it). Cloud A must send a transaction to the blockchain attesting to this. It costs gas for every transaction in a real blockchain. The output from the first command is the receipt from the transaction in the truffle test environment. Note the gas used in the receipt is the estimated cost of gas for the sender. In the second command, another migration is accomplished, this time from cloud B to cloud C. In order to send from different accounts, the truffle environment allows for specifying the sender of the transaction.

In the final commands, a consumer wishes to confirm where their data have been migrated. A consumer issues the getAttest function to retrieve the results, but the request is not coming from the consumer whose attestation is theirs, so this request responds with no data. The second request is coming from the sender whose public address matches the hash into the stored results. The responses from the smart contract represent the raw addresses and do not capture our testing mnemonics. This solution is a first step in a distributed attestation framework supporting cloud trust.

# 6. Transactions and Costs

Homestead is the first production release of Ethereum. The Homestead[13] documentation explains what a transaction includes: "1) Recipient of the message, 2) A signature identifying the sender, 3) VALUE field - The amount of wei to transfer from the sender to the recipient, 4) An optional data field, which can contain the message sent to a contract, 5) STARTGAS value, the maximum number of computational steps the transaction execution is allowed to take, and 6) GASPRICE value, the fee the sender is willing to pay for gas."[13]

Please note that the wei is the smallest denomination of ether; it is like a micro-penny to a dollar, but smaller. If you use the Ethereum public chain, the network charges a fee for the transfer of ether and any computation steps executed in a contract. By charging for computation, Ethereum discourages attacks and abuse, while subsidizing the overall processing capability and consensus mechanisms of the blockchain (via miners).

Gas and ether are related, but also separate. Gas is strictly a function of computation. Ether is more ambiguous and is a reflection of the free market. "Miners have the choice of including the transaction and collecting the fee or not. If the total amount of gas used by the computational steps spawned by the transaction, including the original message and any sub-messages that may be triggered, is less than or equal to the gas limit, then the transaction is processed. If the total gas exceeds the gas limit, then all changes are reverted, except that the transaction is still valid and the fee can still be collected by the miner."[13] This applies to the Ethereum public chain and is not considered in our tests at this time.

At the time of this research, the Homestead documentation says that using smart contracts for just queries are free; we are guarded about this. This represents the best case in that consumers would not have to pay gas for requesting their attestations. In the worst case, our proposed consumer requested attestations would cost some gas since the primary purpose of gas is for both running contract code *and* making a change to the blockchain. At present, the query is free and demonstrated in our tests.

## 6.1 An Example

The transaction cost is comprised of two factors:

- $gasUsed$: (fixed) based on computations per contract
- $gasPrice$: (variable) changes with market price of a unit of gas

$$totalCost := gasUsed * gasPrice$$

If we know that the transaction will consumes 3 gas. The approximate cost is calculated using the default gas price (.02e12 wei as of May 2017)[17] would be:

$$3 * 0.02e12wei = 0.6e11wei \text{ (60 billion wei)}$$

$$1ether = 1e18wei$$

Therefore the total cost would be: 0.00000006 ETH $\approx$ 0 BTC $\approx$ \$0.0 (i.e. so small it does not compute).[13][18]

## 6.2 Our Transaction Costs

These are estimates only. Our test environment provides a simulated $gasUsed$ for each transaction and is based on the computations performed within the smart contract. Our transaction cost $\approx$75,577 gas.

The approximate cost, using the previous gas price would be:

$$75,557 * 0.02e12wei = 1.5e15wei \text{ (1.5 quadrillion wei)}$$

$$1ether = 1e18wei$$

Therefore the total cost would be: 0.0015 ETH $\approx$ .00012 BTC $\approx$ \$.33[13][18]

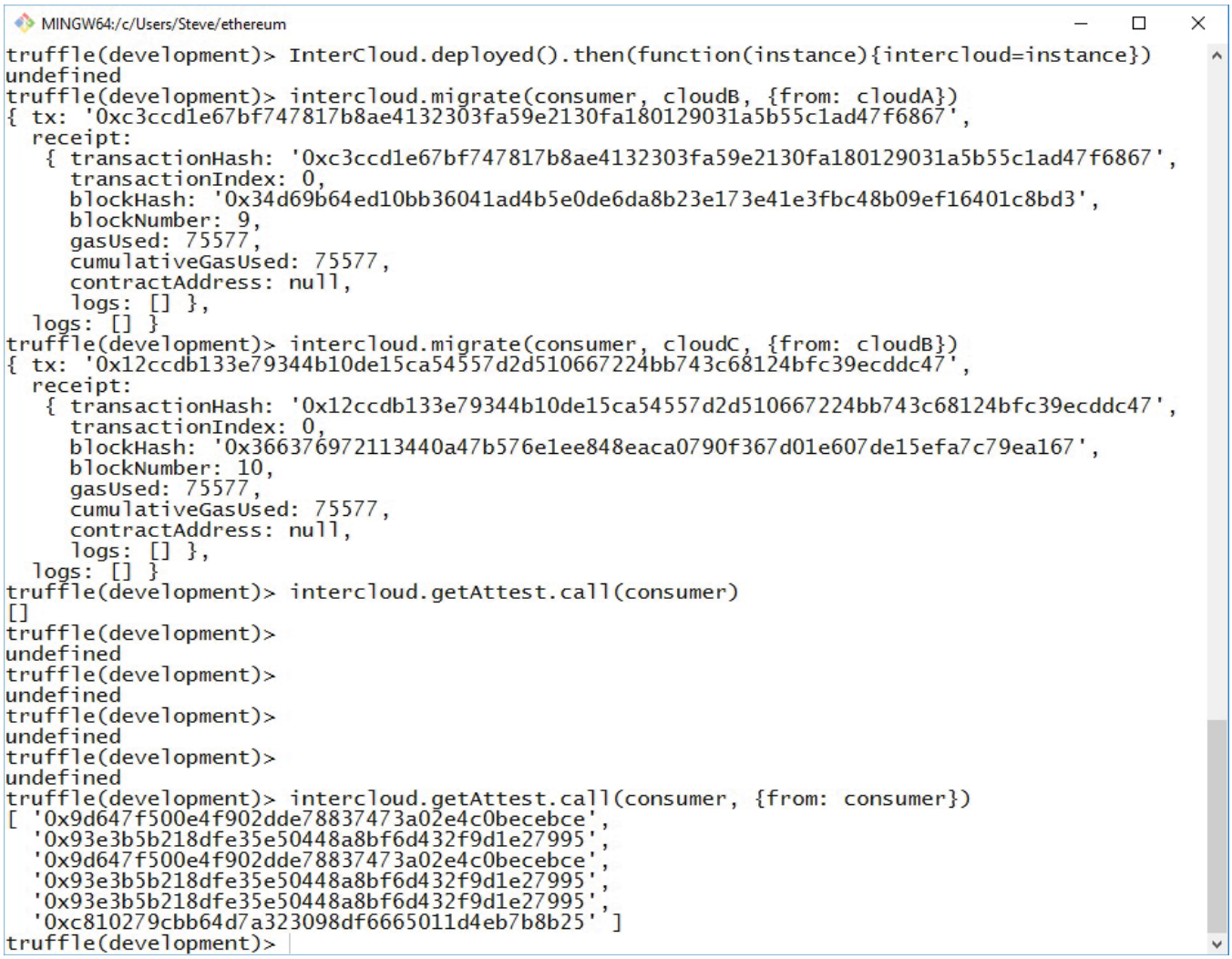


Fig. 5: Testing Migration and Attestation Functions

### 6.3 Advantages & Disadvantages

There are both advantages and disadvantages to using blockchains and Ethereum. For the advantages, we noted that blockchains are append only, public, and tamper resistant. Ethereum's blocks, in particular, are produced much faster than Bitcoin's because they use a different protocol, called Ghost[19], which allows for stale blocks and faster overall block processing. There is debate over the merits of this method which is outside the scope of this research. With regards to security, since the Ethereum Virtual Machine is built on top of the blockchain and it is Turing complete, additional security, privacy functionality might be added. A malicious user might send inaccurate data to the contract (to the extent the contract allows it). Like a cloud, they would still have to pay the transaction fee in order to store bad data on the blockchain.

We summarize the advantages:

- Append only (blocks cannot be removed)
- Accessible to all. Public P2P network.
- Tamper resistant. Block cannot be overwritten just appended
- No central trust mechanism required (hence, no central point of failure)
- Ethereum: Faster creation of new blocks versus Bitcoin, see Ghost protocol[19].
- Ethereum: Added security in the attestation process is possible.

For disadvantages we know that blockchains are made possible by peer to peer networks. However, peer to peer networks are voluntary. Furthermore, the cloud must pay ether for each migration log. This is the price for decentralized trust. There are models that might be used to bring down the cost (e.g. consumer shares load). The cloud could provide misleading data to the blockchain, but it would not benefit them. As a matter of fact, it would cost them.

Therefore, the cloud is incentivized to provide accurate data. It is in their best interest not to provide false information. Furthermore, the account submitting the data are logged into our blockchain storage during a transaction.

These are the disadvantages:

- Participation in the blockchain is not compulsory.
- To send a transaction, it costs the cloud ether.
- Potential for inaccurate data in blockchain.
- The blockchain is public.
- Nothing to impede malicious users from inserting fake data.
- Privacy concerns.

From the advantages listed, particularly the flexibility of running event-based programs on Ethereum compared to other blockchain platforms, we believe Ethereum is the best choice to implement our inter-cloud data migration attestation framework.

# 7. Related Work

## 7.1 Blockchains

Other systems have made use of blockchains: Namecoin(DNS)[20], Onename(PKI)[21], and Blockstack[22]. Blockstack tackles some of the shortcomings of previous naming systems like Namecoin. Its main contribution is the separation of the control plane from the data plane and the introduction of a virtualchain. The virtualchain provides the control logic and acts as a gatekeeper to control what gets inserted into the blockchain. This makes it similar to the smart contract. The smart contract, however, takes the abstraction one step higher as it allows for a wider application base.

## 7.2 Cloud Trust

CloudTrustProtocol[23] allows customers to query the cloud via an API for information related to various service attributes. Rather than query the cloud, our framework provides the consumer the option to state which clouds they trust and which they do not. Cloud Access Security Brokers (CASB) play the role of security middleman between the provider and consumer and consolidating security policy, sign-on, and more. This is a centralized model and the brokers would need to be vetted via a trusted third party.[24] Our model requires no trusted third party and managing the attestations is decentralized. Santos[25] implemented a 'policy-sealed' data trust system for a single cloud domain. It seals (encrypts) and unseals (decrypts) customer data based on the trustworthiness of a cloud node based on the integrity guarantees provided by TPMs. TPMs are trusted platform modules on motherboards designed to provide integrity guarantees at boot time.[26] They do not address trusting the cloud as a whole.

# 8. Further Research

We plan to research ways to include privacy of the attestations, find more uses for smart contracts in the cloud environment to support trusted computing, and develop more realistic tests. According to Ethereum's creator, Vitalik Buterin, "Because the Ethereum protocol is 'turing complete', it is possible to implement advanced cryptography on top of it."[27] Ethereum is evolving rapidly and we will be able to update our results and smart contracts as this takes shape.

*A note on public versus private blockchain:*
We will continue to use either a local test environment or a private blockchain. However, our research ultimately supports using the public blockchain because of the inherently distributed trust that public decentralization provides despite the computation and transaction costs. It is possible to use a private Ethereum blockchain in which case Ether is not required.[27] But there would need to be some token value system defined for the private Ethereum blockchain. For further information on public versus private blockchains, see [28],[29].

# 9. Conclusion

The cloud has an incentive to use blockchain technology for attestation storage because it is inherently trustworthy. Miners might or might not be users of the cloud to enable a synergistic relationship. The accounts held by smart contracts will only fire after receving a transaction or call. In our case, this would be a cloud provider.[30]

We believe the incentives of using a public ledger will encourage trust in both directions. We experimented with a blockchain to provide for storage of attestations. The nature of attestations demands storing them in a verifiable distributed database that does not reside in the confines of the cloud providing the proof. The blockchain is appropriate for this use.

We have conducted initial tests using an Ethereum contract to track data. The tests are promising, however further tests and improvements are needed as well as research exploring how the market rate of transactions will impact our design. We believe blockchains are a key to decentralized trust in the cloud.

# Acknowledgments

This research was conducted with Government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.

# References

[1] "Business trust in data security in the cloud at an all-time low," *http://www.globalservices.bt.com/uk/en/news/business_trust_in_data_security_in_cloud_at_all_time_low*, 2014.

[2] R. Dolan, "Security remains a major obstacle to cloud adoption, study finds," 2015, https://www.datapipe.com/blog/2015/03/25/security-remains-a-major-obstacle-to-cloud-adoption-study-finds/.
[3] Z. ur Rehman, O. K. Hussain, E. Chang, and T. Dillon, "Decision-making framework for user-based inter-cloud service migration," *Electronic Commerce Research and Applications*, vol. 14, no. 6, pp. 523–531, 2015.
[4] C.-H. Suen, M. Kirchberg, and B. S. Lee, "Efficient migration of virtual machines between public and private cloud," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 549–553.
[5] S. Kirkman and R. Newman, "Bridging the cloud trust gap: Using orcon policy to manage consumer trust between different clouds," *accepted for publication, IEEE Edge*, 2017.
[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[7] M. Pilkington, "Blockchain technology: principles and applications," *Research Handbook on Digital Transformations, edited by F. Xavier Olleros and Majlinda Zhegu. Edward Elgar*, 2016.
[8] A. Wright and P. De Filippi, "Decentralized blockchain technology and the rise of lex cryptographia," *Available at SSRN 2580664*, 2015.
[9] "Ethereum," *https://www.ethereum.org/*, 2017.
[10] A. VanAmmers, *Edgar Guide https://forum.ethereum.org/discussion/2116/in-what-ways-can-storage-history-be-accessed*, 2017.
[11] "Testrpc," *https://github.com/ethereumjs/testrpc*, 2017.
[12] *Ethereum Homestead Documentation http : / / ethdocs . org / en / latest / contracts-and-transactions / contracts . html # interacting-with-a-contract/*, 2016.
[13] *Ethereum Homestead Documentation Release 0.1 https : / / media . readthedocs . org / pdf / ethereum-homestead / latest / ethereum-homestead.pdf*, 2016.
[14] "Truffle," *https://github.com/trufflesuite/truffle*, 2017.
[15] "Visual studio code," *https://code.visualstudio.com/*, 2017.
[16] "Solidity online compiler," *https : / / ethereum . github . io / browser-solidity*, 2017.
[17] "Ether stats," *https://ethstats.net/*, 2017.
[18] "Ether exchange," *http://ether.price.exchange/*, 2017.
[19] "What is the ghost protocol for ethereum, urlhttps://www.cryptocompare.com/coins/guides/what-is-the-ghost-protocol-for-ethereum/ ," 2017.
[20] "Namecoin," 2016. [Online]. Available: www.namecoin.info
[21] "Onename." [Online]. Available: www.onename.com
[22] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, 2016.
[23] C. S. Alliance, "Ctp data model and api, rev. 2.13," *https : // downloads . cloudsecurityalliance . org / assets / research / cloudtrust-protocol/CTP-Data-Model-And-API.pdf*, 2015.
[24] CASB, "Cloud access security brokers," *https : / / totalproductmarketing . com / the-growing-importance-of-cloud-access-security-brokers/*, 2016.
[25] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu, "Policy-sealed data: A new abstraction for building trusted cloud services," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 175–188.
[26] TCG, *Trusted Platform Module http://www.trustedcomputinggroup.org/work-groups/trusted-platform-module/*, 2016.
[27] V. Buterin, *Ethereum in 25 Minutes, Devcon 2 https://www.youtube.com/watch?v=66SaEDzlmP4*, 2016.
[28] *Public vs Private Chain https : / / blog . slock . it / public-vs-private-chain-7b7ca45044f/*, 2016.
[29] *Ethereum Blog hhttps : / / blog . ethereum . org / 2015 / 08 / 07 / on-public-and-private-blockchains/*, 2015.
[30] "What is ethereum, urlhttp://ethdocs.org/en/latest/introduction/what-is-ethereum.html?" 2016.