Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams

Nurzhan Zhumabekuly Aitzhan and Davor Svetinovic, *Member, IEEE*

Abstract—Smart grids equipped with bi-directional communication flow are expected to provide more sophisticated consumption monitoring and energy trading. However, the issues related to the security and privacy of consumption and trading data present serious challenges. In this paper we address the problem of providing transaction security in decentralized smart grid energy trading without reliance on trusted third parties. We have implemented a proof-of-concept for decentralized energy trading system using blockchain technology, multi-signatures, and anonymous encrypted messaging streams, enabling peers to anonymously negotiate energy prices and securely perform trading transactions. We conducted case studies to perform security analysis and performance evaluation within the context of the elicited security and privacy requirements.

Index Terms—Security and privacy, decentralized energy trading, blockchain technologies, smart grid systems

1 Introduction

Smart grids (SGs) are expected to provide not only fine-grained consumption monitoring, but also engage increasing number of residential power generation sites into distributed energy trading (e.g., a community microgrid). As such, it is important to equip SGs with a secure energy trading infrastructure capable of executing contracts among trading agents and handling bidding, negotiation and transactions while preserving identity privacy.

A majority of modern financial infrastructures are centralized and implicate involvement of a trusted third party, which handles accounts, processes payments and provides security. The centralized energy trading suffers from scalability and security concerns, e.g.,

1) *Single point of failure:* As a key component of a centralized network, failure of a centralized middleman leads to full disturbance of authentication and payment activities, and obstruct from providing availability and reliability security goals.
2) *Lack of privacy and anonymity:* Following Wood's behavioral modeling of electricity consumption profiling approach [1], a centralized middleman may reveal patterns of an agent's energy generation and predict the agent's daily activities.

These major drawbacks of the centralized infrastructure have motivated us to address the problem of providing identity privacy and transaction security in energy trading using a decentralized approach. The decentralized nature of communication relies upon the cooperation among individual nodes to carry out essential tasks of information propagation. Although public key cryptography could be applied to provide a certain level of security and integrity of information, the most important issue when dealing with public keys is ensuring their authenticity without relying on a trusted third party.

A trustless or semi-trustless decentralized energy trading system could provide transaction security and identity privacy, while relying on cryptographic techniques instead of relying on a trusted third party. To verify our claim we have adapted and implemented a proof-of-concept for decentralized energy trading system where all nodes collectively act as a replacement for a trusted party, and vote on validity of transaction by traversing through history of publicly available distributed chain of transactions. The proposed system, PriWatt[1] is inspired and built upon decentralized digital payment Bitcoin system and decentralized peer-to-peer message authentication and delivery system Bitmessage. Bitcoin system adopts cryptographic proof-of-work along with nested chain of hashed secrets to eliminate need of trusted third party providing security and privacy when an agent trades with complete strangers [2]. Bitmessage provides anonymity in a trustless network through propagating encrypted messages in messaging streams [3]. While sustaining transaction security in a trustless model the PriWatt does not reveal identities of trading parties and keeps their financial profiles private.

Thus, the main contribution of this work is the integration and prototype implementation of blockchain technology, multi-signature approach, and anonymous encrypted message into the PriWatt system, so that transactions within

• *The authors are with the Department of Electrical Engineering and Computer Science, Masdar Institute of Science and Technology, Abu Dhabi 54224, UAE. E-mail: dsvetinovic@masdar.ac.ae.*

Manuscript received 24 Nov. 2015; revised 28 July 2016; accepted 9 Oct. 2016. Date of publication 12 Oct. 2016; date of current version 29 Aug. 2018.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2016.2616861

1. The name is given to the proposed system only to make the presentation in this paper clearer.

1545-5971 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

a decentralized system are enabled with high privacy and security. We have performed performance analysis of the system, and also a qualitative analysis in terms of requirements satisfaction. In addition to helping with the reliability and privacy issues, the proposed system might be of significant help in designing and deploying SGs in challenging environments such as less developed countries, war zones, etc., some of which completely lack not just the power infrastructure, but also financial infrastructure, privacy and security-protecting laws, etc. Decentralized microgrids combined with digital currencies such as Bitcoin can lead to a faster and more robust solution to power problems in such environments and extreme conditions. Our system provides a plausible solution to enable such microgrid designs.

2 Core System Components

PriWatt is a trustless decentralized token-based energy trading system, which provides agents with anonymous communication channel and ability to trade energy ownership in the SG using distributed smart contracts. In this section we discuss the core components of the system.

2.1 Transactions

A transaction is an instance of changing ownership of tokens through digital signing portion of data and broadcasting it to the network. Structure of transaction Tx can be represented by the following equation:

$$Tx = nVersion \| vinNum \| vin \| voutNum \| vout \| nLockTime, \tag{1}$$

where vin and $vout$ are vectors of input and output represented by tuples of elements which are used for transferring ownership of token from previous owner to the current owner, and the current owner to the next owner accordingly. $vinNum$ and $voutNum$ show the number of transaction inputs and outputs. $nLockTime$ is a timeframe past which transactions can be replaced before inclusion in a block. Transactions are linked to each other by including a hash of the previous transactions into a field of the current transaction. Tokens are transferred in either consecutive or non-consecutive order in transactions n, m, k of blocks A, B and C, such that $n \in A$, $m \in B$ and $k \in C$, where $n < m < k$ and $A < B < C$. When a transaction is broadcasted to the network, a sender announces the new owner of the token and every peer through tracing history of the token ownership votes on the validity of the transaction. All existing transactions are placed in publicly available *blocks*, which form a tuple of transactions that are timestamped and chronologically chained to each other, forming a *blockchain*.

2.2 Blockchain

A blockchain is a chronologically ordered chain of blocks protected by solving a proof-of-work. The chaining is done by adding the hash of the previous block to the current block, the hash of the current block to the next block, and so on. Consecutive nested blocks guarantee transactions to come in a chronological order, hence a transaction cannot be changed backdated without changing its block and all the following blocks. While the most of decentralized electronic payment systems suffer from double-spending attack,[2] the blockchain makes it infeasible unless attacker controls over 51 percent of the whole network computational power [4].

The blockchain is controlled through collective voting of anonymous nodes, therefore both honest and adversary nodes participate in generation of blocks. To prevent double-spending attack the system makes generation of consecutive nested blocks (hashes of blocks) computationally difficult. Therefore, the system shall continue to function properly while the collective computational power of the adversary nodes does not dominate the computational power of trustworthy nodes. Since a sub-chain is considered to be a valid only if it is longer than a competitor chain, honest nodes should generate new blocks faster than an adversary tries to catch-up or even outperform a valid blockchain after they backdated altered an old transaction. The system provides this behavior through adopting a proof-of-work system, which is a proof that a node invested necessary resources to do a computationally difficult work.

2.3 Proof-of-Work

The main concept of the proof-of-work is a puzzle which is expensive to solve, but knowing all inputs—cheat to verify. In a nutshell, to generate a block, a node collects pending transactions, hash them into a hash of a merkle tree root, then along with other data iteratively hash this data set until it results in a hash that is less than or equal to a predefined *target value*. Target is a hash value that serves as a threshold, below which a block header must be hashed to generate a block. Target is a 256 bit number with special k numbers of zero significant digits, which constructs PoW difficulty, requires on average 2^k attempts before the puzzle is solved. Finding proof for a given target is a linear function, therefore the lower the target value is, more hashing attempts are required. Periodically, based on computational power involved in the network, the difficulty of solving the puzzle changes. Solving proof-of-work is a probabilistic process because for a hash to change it is required to change inputs to be hashed. To guarantee generation of a new hash for each iteration the system iteratively changes an arbitrary data $nNonce$ and the coinbase field in a coinbase transaction[3] which consequently changes the *hash of merkle root* in a block header. A proof is found by brute-forcing. Proof-of-work is a probabilistic iterative procedure, hence to a certain extent decreases a chance to generate blocks at the same time. Although proof-of-work does not eliminate block birthday collisions its core objective is to prevent the double spending attack. A probability of finding $nNonce$ of proof H for a given target T is

$$P(H \leq T) = T/(2^{256}). \tag{2}$$

Once such a hash is found, a successful node broadcasts the proof along with input transactions and other data of transactions used for finding valid H. Nodes validate the proof by re-computing received tuple and then the block is confirmed as a valid and included into the blockchain.

2. Double spending is an attempt to spend the same token twice.

3. The transaction designated to carry a reward for a node who solved a block.

2.4 Signing Transactions

To validate the authenticity of a transaction, PriWatt (like Bitcoin) uses Elliptic Curve Digital Signature Algorithm (ECDSA) asymmetric cryptography [2]. Systems use OpenSSL tool-kit to generate secp256k1 based Koblitz curve for ECDSA key-pair. Signed transaction allows other peers to verify that the sender is a person he claims to be and possesses tokens he is willing to transfer.

Beside using as a signature the private portion of a public/private ECDSA keypair, it is also used for decrypting the transaction stored in an encrypted *wallet*. The public portion of a keypair is used for generating an address which is a unique strings of 27-34 alphanumeric characters, e.g., 3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy. There are two types of PriWatt addresses, Pay-to-PubkeyHash (P2PKH) and Pay-to-ScriptHash (P2SH), which are results of Base58 encoded concatenation of RIPEMD-160 hashing of the hash of SHA256 hashed ECDSA public key (*pubKey*) or Redeem Script (*rScript*) and thier checksums accordingly. Following equations represent stages of generating an address:

$$hash = RIPEMD160(SHA256(pubKey \oplus rScript)) \quad (3)$$

$$checksum = Truncate(SHA256^2(hash||0x00)) \quad (4)$$

$$address = Base58(hash||0x00||checksum). \quad (5)$$

For P2PKH, once ECDSA generates public key, the system hashes public key first using SHA256 and then RIPEMD160, resulting to a pubkeyHash. Next network id byte is concatenated to the pubkeyHash. To eliminate any typographical errors and confirm that address is valid the system generates a checksum of an address and concatenates it to the digest. It is done by double hashing with SHA256 the result of first concatenation and truncating the result of second concatenation to the first four bytes. The P2SH goes through a similar procedure, but instead of hashing the public key, the redemption script is hashed into a scriptHash. To tackle profile indistinguishability, system forces users to use new address for every new transaction which makes it more difficult to trace the ownership of many addresses by the same user.

2.5 MultiSignatures

PriWatt provides the means to form agreements or contracts without trusting other party via blockchain. Contracts allow parties to perform complex transactions and prevent malicious activities. For example, assume that a customer Alice wants to buy a product from a merchant Bob. They make a contract where Bob will not get paid until Alice gets a product. Similarly, Alice cannot get a product and keep her payment. Moreover Alice and Bob can include Chuck as an arbitrator who can resolve disputes.

In distributed contracts, the system protects against theft by requiring multiple independent parties to sign a transaction before it can be considered as valid. This is achieved through multi-signature transaction where minimum m of n keys must sign a transaction before tokens can be spend. The system uses *multisig pubkey scripts*, where m is the minimum number of signatures which must match a public key; n is the number of public keys being provided.

PriWatt inherits ability of Bitcoin's transactions to specify a script that defines conditions under which a transaction may be redeemed. In Bitcoin, this is performed using scripts that are written in a stack-based, non-Turing complete programming language. The scripting language includes support (OP_CHECKMULTISIG) for multi-signature scripts [5] which require at least t of n specified public keys to provide a signature on the redeeming transaction. By default, multi-signature transactions are currently only relayed with $n < 3$ keys, but may specify up to an absolute limit of $n = 20$.

2.6 Anonymous Messaging Streams

The PriWatt system supports two types of communication: sending a private person to person and message broadcasting. The system protects parties from passive eavesdropping by hiding non-content data, i.e., masking identities of interacting parties through assigning unique strings of 36 alphanumeric characters. Messages are exchanged between peers by forwarding messages on a best effort basis. As a result all active nodes receive all messages and each node attempts to decrypt every message with their private keys. Since everyone receives each encrypted message, the recipient's identity stays anonymous. Eventually, messages make their way to the recipient who decrypts message with their unique private key.

To send a message, the system is required to perform a proof-of-work on a partial hash collision scheme. This technique helps to prevent spamming. The difficulty of proof-of-work should be above certain threshold and proportional to the size of the message. To prevent the network from flooding with re-broadcasted old messages the time of message is included and old messages are not relayed. Also to satisfy scalability with regards to the memory, the system stores all messages only for a short period of time.

Messaging addresses depend on the way they have been generated and are of two types: standard addresses based on salted random number generator and deterministic addresses generated from user defined passphrase. PriWatt forces users to generate new messaging addresses for each new trade negotiation in order to preserve anonymity. Since all users receive all messages the system results to message broadcasting feature. This feature allows anyone with an authenticated identity to anonymously broadcast messages. PriWatt implements its auction offerings based on described message broadcasting feature.

3 Token-Based Decentralized Energy Trading System

PriWatt is a token based energy trading system which allows to trade energy in a peer-to-peer network without a central price signal. In the case studies, we assume that prosumer using photo-voltaic panels harvests solar energy and makes profit by selling the generated energy surplus to the other SG consumers.

In a conventional power grid, energy prices are determined by a central authority. In a SG, agents should be allowed to negotiate energy prices, hence creating a dynamic market of energy trade. Such a market-based energy trade decreases dependency of agents on a central energy provider, as energy supply and demand are matched directly between individual agents, resulting in a more decentralized

and competitive environment. PriWatt satisfies this requirement and enables peers to anonymously negotiate energy price and securely perform trading transactions. We describe the concept through the following two behavioral use cases.

3.1 Case 1: Trading Full Ownership of Stored Energy

Assume that Bob is a prosumer, Alice is a customer and Distribution System Operator (DSO) is a mediator. Prosumer Bob is willing to generate and sell energy. Algorithm 1 gives a detailed step-by-step overview of required procedures and functions for trading energy.

Algorithm 1. Energy Trading Algorithm

procedure ENERGYTRADING
  $txAddr \leftarrow hash(pubKey)$ ▷See equation 5
  $msgAddr \leftarrow hash(pubKey, privKey)$
  $Supplier \leftarrow txAddrB, msgAddrB, E_1$ ▷E_1—generated energy
  $Customer \leftarrow txAddrA, msgAddrA, T$ ▷T—token of energy
  $Mediator \leftarrow txAddrD, msgAddrD$
  procedure INJECTENERGY(E_1)
    $b_\lambda \leftarrow SHA256(txAddrB \| E_1 \| Timestamp)$; ▷$E_1$ to be accessed using $a_{i,j}$ only
    $b_z \leftarrow SHA256(b_\lambda \| RndNum)$; ▷$E_1$ to be locked with b_z
    $msgAddrD$.msg(b_λ, b_z) $\Rightarrow msgAddrB$
    $msgAddrB$.broadcast($E_1, P_1, txAddrB, msgAddrB$)
      ▷Appears as an offering in auction board
  end procedure
  procedure MATCH(E', P') ▷Query to match E', P'—energy and price accordingly
    if $\forall tx \in blockchain$: $\nexists b_\lambda$ **then return** $True$
    else return $False$
    end if
  end procedure
  procedure VALIDATE($txAddr, E'$) ▷Energy amount validation for a given Supplier
    if $\forall txAddr \in Addr, \exists b_\lambda \mid E_{b_\lambda} \geq E'$ **then return** $True$
    else return $False$
    end if
  end procedure
  procedure GENERATETRX
    if $txAddrA$.match.validate($txAddrB, E_1$) **then**
      $msgAddrSh \leftarrow$ genAddr($msgAddrB \| msgAddrA$) ▷Shared msgAddr between Supplier and Customer
      $msgAddrSh$.lock(b_z) $\Rightarrow msgAddrD$
      $txAddrB \Rightarrow multiSig.rScript(OP\_m \| txAddrD \| txAddrB \| txAddrA \| OP\_n \| OP\_CHECKMULTISIG)$
      $msgAddrB$.msg($multiSig.rScript$)$\Rightarrow msgAddrA$
      $txAddrA \Rightarrow rScript \leftarrow (multiSig.rScript)$
      if nodisputes **then**
        $multiSigTx \leftarrow txAddrA$.Sign($rScript$)
        $txAddrA$.broadcast($multiSigTx(T)$) ▷Payment has been completed
      else
        ($msgAddrB \oplus msgAddrA$).msg($multiSig.rScript \oplus rScript$)$\Rightarrow msgAddrD$
        $sigScript \leftarrow txAddrD$.Sign($multiSigTx(T')$)
        $msgAddrD$.msg($sigScript$)$\Rightarrow msgAddrB, msgAddrA$
        ($txAddrA \oplus txAddrB$).Sign($sigScript$).broadcast()
          ▷Payment or Refund has been completed
      end if
    end if
  end procedure
  procedure CHANGESECRETOWNER
    if $multiSig(tx)$.confirmed() **then**
      $msgAddrB$.msg(b_λ)$\Rightarrow msgAddrA$
      $msgAddrA$.msg($b_\lambda, unlock, update$)$\Rightarrow msgAddrD$
      $msgAddrD$.msg(a_λ, a_z)$\Rightarrow msgAddrA$
        ▷See equation 9
    end if
  end procedure
end procedure

Energy Injecting. To hide his identity Bob creates a new pair of addresses from the corresponding public and private keys for each energy generation session: $txAddrB$ to perform transactions and $msgAddrB$ to anonymously send messages over the messaging stream. Bob opens a transmission channel and feeds energy into the SG. We assume that Bob's smart meter is sealed for tamper resistance. Utility automated reading devices ARM using inverters calculate the amount of energy E_1 transmitted from key pair ($txAddrB$, $msgAddrB$). Utility DSO updates the distributed database and creates a new entry with address $txAddrB$, $msgAddrB$, amount of stored energy E_1, a timestamp $Timestamp$ and two unique hashes:

$$\begin{aligned} b_\lambda &= SHA256(pubKeyB \| E_1 \| Timestamp) \\ b_z &= SHA256(b_\lambda \| RandomNumber). \end{aligned} \tag{6}$$

Utility DSO, using its own messaging address $msg\ AddrD$, sends a private message to Bob's address $msgAddrB$. A message contains two unique secrets keys b_λ and b_z, where the former one is the static key that verifies Bob's ownership over E_1 and the latter one is the temporary key which Bob will later use as a lock to prevent from double-spending E_1. Locking E_1 is required to eliminate supplier's attempt to double-spend ownership of traded energy. If E_1 is not locked while performing transaction with Alice, Bob could sell ownership of the same amount of energy to other buyers. This is especially important when Bob trades partial ownership through a micropayment channel, as discussed in the Case 2. When Bob decides to sell produced energy, he posts the offering at auction board by broadcasting an anonymous message which contains E_1, price P, $txAddrB$ and $msgAddrB$. Bob will b_λ to sell ownership of energy locked by this secret key and b_z to verify his identity when he will request from a shared messaging stream address. The role of DSO is not finished yet; later in the Bob-to-Alice energy trading case, DSO using distributed databases can testify that $pubKey_B$ has amount of E_1 he claims and act as a mediator if a dispute arises.

Matching and Validation. To participate in energy trading and keep her identity private, Alice creates a pair of new addresses $txAddrA$ and $msgAddrA$. Every node, including Alice, receives Bob's broadcasted message which appears in an auction board. Alice triggers a matching procedure, which filters results according to defined price or amount values and the a handler eliminates processed offerings by checking corresponding transactions in the blockchain. Since there can be many suppliers, Alice filters results according to the price and amount. Auction block runs a

matching procedure which is based on checking if the offering already has been processed. To achieve this, auction blockchain handler checks the blockchain for the latest transactions for a given $publicKey$ such that $tx.timestamp < curr.timestamp$. Alice gets a list of active offerings and messaging stream addresses of supplier relevant to her query. Alice uses auction panel to select offers and send private messages to negotiate with potential suppliers. The identity of Alice sustains anonymous behind a pseudonym address $msgAddrA$. If Alice selects Bob as a potential supplier she sends a private message to DSO asking to verify Bob's claim of ownership. DSO checks records in database and replies whether claim is true or false.

Preventing Double-Spending. There are two possible double-spending attacks: double-spending of energy token T from the customer's side and double-spending ownership of an amount of energy b_λ from the supplier's side. To prevent double-spending of E_1 ownership, our system "locks" E_1 from other transactions. The lock request message is sent by Bob to DSO and contains the b_z secret which proves Bob's identity. This unique value also prevents attackers from spoofing Bob's identity and requesting DSO to lock all his ownerships. To send the *lock* request Bob uses a new messaging stream address $msgAddrSh$ shared between him and Alice. The lock request and temporary b_z value is valid until unlock or b_z renew will not be requested.

Multi-Signature Transaction. After E_1 is locked, Bob creates a multi-signature transaction based on $P2SH$ multisig redeem script as following:

$$rScript = OP\_2 \| pubKeyD \| pubKeyB \| pubKeyA \| \\ \| OP\_3 \| OP\_CHECKMULTISIG. \quad (7)$$

This is a case of 2-of-3 multisig pubkey script, where OP_2 and OP_3 *stacks* show that two signatures are required to sign a transaction and three public keys should be provided, accordingly. Bob gives the multisig redeem script to Alice who first ensures presence of her public key $pubKeyA$ and DSO's public key $pubKeyD$, and then hashes the script to generate $P2SH$ redeem script. Then she specifies the input token(s), signs and broadcasts the multisig transaction.

After seeing the payment Bob can send the b_λ to Alice. If there is a dispute between Bob and Alice, one or both of them send evidences and the redeem script to the mediator DSO to resolve the issue. After DSO makes a decision, it attaches unhashed serialized redeem script created by Bob and signs the transaction assigning two new outputs of different shares ($0 < share < 100$ percent) of the token destined to Alice and/or Bob. Then DSO sends a copy of incomplete transaction to Alice and Bob. Either one of them can complete the multi-signature transaction by adding his/her signature to create the following signature script:

$$OP\_0 \| pubKeyD \| [pubKeyB \oplus pubKeyA] \| OP\_3 \| rScript. \quad (8)$$

Once one of them signs a transaction and broadcasts it to the network, nodes need to ensure that the redeem script matches the redeem script hash provided previously. This is performed by matching the signature script with the P2SH digest which Alice used in the first payment attempt. Then nodes evaluate redeem scripts with the two signatures which were used as an input data. If the redeem scripts validated, then both transactions with shares for Alice and Bob will be considered as valid. Nevertheless, if both Bob and Alice do not agree with DSO decision, then none of them will sign the transaction, hence they need to find another mediator. However, if there are no disputes then Alice sends a private message to DSO including b_λ requesting to replace both b_λ and b_z with a_λ and a_z accordingly. The DSO performs this activity using following equation:

$$a_\lambda = SHA256(pubKeyA \| E_1 \| Timestamp) \\ a_z = SHA256(a_\lambda \| RandomNumber). \quad (9)$$

Now Alice can either spend energy E_1 or trade it with someone else using her own unique a_λ and a_z secret keys.

3.2 Case 2: Trading Partial Ownership over a Micro-Payment Channel

Consider a scenario when Alice wants to pay for every single kWh rather than purchasing full ownership of Bob's stored energy. Using BitcoinJ libraries PriWatt establishes a micro-payment channel to pay for each kW at a time. Although Bitcoin allows to perform the small value transactions there are two potential drawbacks. First, there is a fee per every transaction in the Bitcoin, hence for many small transactions collective fee can be overwhelming. Second, Bitcoin has an anti-flood algorithm which down-prioritizes if finds a large number of small consecutive transactions. BitcoinJ libraries for micro-payment channel were implemented as client-server interactions. In the nutshell, the client creates a time locked refund multi-signature transaction, sends it to the server who broadcast it to the network. When a part of server's utilities has been used, client decrements the refund for corresponding value of utilities he used. Then the client updates the transaction and sends it to server. The last two procedures iteratively happen number of times, until either the client requests to close the channel or amount of tokens in the contract ends up. Then server broadcasts the latest version of the transaction with the latest amount to the network.

Time Locked Multi-Signature Refund Transaction. Let's consider that price per kWh is $0.1T$ token. Following the same approach Bob used in the Case 1, consider that Alice creates a multi-signature time locked refund contract $multiSigTx$ which holds a $1T$ token. The entire amount, in this case $1T$ token, is pointed out as a refund back to the client. By default PriWatt defines refund locked time to 24 hours. Alice sends the multi-signature transaction to Bob's $msgAddrB$ using anonymous message stream. Bob receives the multi-signature transaction and broadcasts it to the network.

Splitting the Secret. Bob calculates the maximum number of iterations needed to spend the entire amount. In this case it is 10, hence Alice can purchase maximum 10 kWh. Bob sends a message to DSO requesting to create nine more b_λ unique keys such that every new secret is generated following the equation below:

$$b_{\lambda+1} = SHA256(b_\lambda \| (curr.Timestamp / RandomNumber)). \quad (10)$$

From now on, b_λ transfers ownership only of 0.1 of E_1. In the first iteration for Alice to use 1 kWh of E_1 Bob sends Alice b_λ. Alice sends decrements the refund by $0.1T$ and sends Bob updated version. For the next iterations, Bob sends $b_{\lambda+1}, b_{\lambda+2}$, etc., after receiving updated versions of multi-signature transaction with gradually decremented refund. This happens iteratively until either Alice is done with the contract or original contract runs out of the tokens. Then Alice asks Bob to close the channel and Bob gets the latest updated version of the transaction and broadcasts the contract to the network. If Alice stops communicating to Bob, he still secures his funds because he has the latest updated version of the transaction. Similarly, if Bob stops responding to Alice or for any reasons does not want to close the channel, then once the time locked passes Alice receives remaining part of the token as a refund.

Algorithm 2. Micropayment

procedure MICROPAYMENT
  $multiSigTx \Rightarrow txAddrA$.Sign($rScript, T, lockTime$)
  $msgAddrA$.msg($multiSigTx$)$\Rightarrow msgAddrB$
  $txAddrB$.broadcast($multiSigTx$)
  $i \leftarrow T/P$
  $n \leftarrow 0$
  $msgAddrB$.msg($b_\lambda, i+1$)$\Rightarrow msgAddrD$
  $msgAddrD$.msg($[b_{\lambda+1}, \ldots, b_{\lambda+i-1}]$)$\Rightarrow msgAddrB$
  $msgAddrB$.msg(b_λ)$\Rightarrow msgAddrA$
  for $n \leq i$ **do**
    if $msgAddrA$.msg($multiSigTx$)$\Rightarrow msgAddrB$
        $|\, T' + i * P \leq T$ **then**
      $i \leftarrow n+1$
      $msgAddrB$.msg($b_{\lambda+i}$)$\Rightarrow msgAddrA$
      $T \leftarrow T'$
      $n \leftarrow n+1$
    end if
  end for
end procedure

4 Implementation and Simulation

The main software components of the PriWatt trading infrastructure are:

The Platform Client. The platform client is a smart meter integrated client which manages transaction execution, validation, signing, nodes observation and messages encryption and propagation. The client is written in Python 2.7 and it uses the following libraries and APIs:

1) python-bitcoinlib—Python2/3 library written by Peter Todd used to interact with Bitcoin data structures and system, network and supports interface to Bitcoin's JSON-RPC API. It is forked from Jeff Garzik's bitcoinprc python libraries.
2) libbitcoin toolkit—a set of cross platform C++ libraries for building Bitcoin applications. The toolkit is a successor of speslimo SX, consists of several libraries most of which depend on the foundational libbitcoin library.
3) PyBitmessage API—set of APIs to communicate with a full Bitmessage node using XML-RPC. It also requires OpenSSL.
4) pysolar—a collection of Python libraries for simulating the irradiation of any point on earth by the sun. Performance of photovoltaic energy systems and performance modeling. Includes pandas library for data analysis, numPy package for scientific computing and matplotlib for 2D plotting.

P2P Network and Nodes. To simulate and validate described model we had to mimic smart meter's activities. Different smart meters have been used for SG pilot projects and many researchers discussed their architectural and computational capabilities. We found out that low cost Linux running Raspberry PI devices are widely used to mimic a network of smart meters. Both computational and networking capabilities of Raspberry PI satisfy the requirements for smart meters defined by NIST IR7628 SG Framework [6] and the application layer messages on an IP network framework for an AMI defined by ANSI C12.22/IEEE 1703/MC12.22 [7]. We built a private testnet p2p network of emulated Raspbian machines with 512 Mb RAM each. All nodes in the private network were connected in a p2p network and propagated transactions and messages to each other. One of emulated machines serves as a DNS server and two emulated machines were running an advanced penetration testing distribution—Linux Kali, to mimic attacking nodes.

Blockchain. Size of Bitcoin's distributed blockchain database grows if number of generated blocks and transferred transactions increases. The current Bitcoin blockchain size is around 39.8 Gb. Although Simplified Payment Verification clients could be used to drastically decrease blockchain size, they pose a security threat to system as they rely on third party servers. On a testnet we generated our own private blockchain creating unique merkle hash root and a new genesis block.

Simulation of Solar Energy Generation. To mimic smart meter activities we need to calculate solar energy output of a photovoltaic system. PySolar libraries provide a number of functions to find altitude and azimuth of solar positioning. User defines the following parameters:

1) latitude and longitude
2) simulation start and end time for any specific day
3) total solar panel area
4) solar panel yield
5) performance ratio coefficient for losses of PV system.

Based on the altitude and azimuth, we estimate direct solar irradiation for each second of the simulation period. Finally based on the user defined total solar panel area, solar panel yield, direct solar radiation on tilted panels and performance ratio coefficient for losses of PV system, we calculate PV panel output. We use this output as a generated energy while simulating agent behavior cases, which were discussed earlier.

5 Evaluation

5.1 Systems Performance and Scalability Analysis

PriWatt unstructured P2P network can be described as an undirected connection graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of all nodes and $\mathcal{E}$ the set of connections between the nodes. Node $v_i \in \mathcal{V}$ propagates block $\mathcal{B}_n$, transaction $\mathcal{T}_n$ or message $\mathcal{M}_n$ by announcing inv_i message to the set of its connected

TABLE 1
Performance Comparison

	Centralized	Decentralized
Monitoring rate (s)	2-60	30
Protective relaying (ms)	4	600*
Availability latency (s)	5	10
Operational latency (s)	<1	5.3

nodes $\mathcal{V}_i'$ once the transaction or block has been verified. The subset of nodes $[v_{i+1}..v_{i+k}] \in \mathcal{V}_i'$ who unaware of $\mathcal{B}_n$ or $\mathcal{T}_n$, requests it by sending $getdata$ message, others who already have it simply ignores inv_i. Therefore we consider that every hop is a trip of data in a time interval $\delta = inv_{i+1} - inv_i$, where inv_{i+1} is announcement of v_{i+1} of $\mathcal{B}_n$ or $\mathcal{T}_n$ for further re-broadcast.

With 14 virtual machines running as full mining nodes, we generated 100 blocks of transactions and for each hop of block $\mathcal{B}_n$ we captured inv_i and inv_{i+1} timestamps. The results show that the median for block and transaction propagation is 5.3 sec and the mean is 11.5 sec. Each hop has a propagation delay due to the transmission and verification of block's PoW and each transaction in the block. Therefore, since running time intervals of signature validation and PoW verification are negligible, we presume that transaction verification, which requires history tracing, seizes larger portion of this propagation time delay. Having a correlation between block size and propagation time, we assume that low number of transactions and shorter blockchain made a positive affect on the propagation time. Therefore we performed scalability analysis on larger and more mature Bitcoin's blockchain.

With linear growth of blockchain size, the daily rate of transactions in Bitcoin reached to 241,346, and in the interval of the last two years mean rate grew to 0.88 transaction per second (TPS). Considering current block size limit and block generation rate Bitcoin is able to scale up to seven TPS. However compared to the centralized payment system such as Visa, which scales to 2,000 TPS or approximately 150 million transactions a day, blockchain's bandwidth is too low given current structural limitations.

PriWatt logically splits communication layer into networking and object processing sub-layers to separate time spent for decrypting and networking. In the networking sub-layer every 10 seconds timestamp, stream number and PoW are checked and fed into objectHashHolder thread which pushes input vector to the sendDataThreadMailbox thread to send it to peers. The receiveDataThread places incoming input vectors into a queue for the objectProcessorThread to handle. The receiveDataThread has a 0.6 second delay after receiving an object to mitigate DDoS attacks on the objectProcessor.

5.2 Centralized versus Decentralized

NIST defines logical interfaces that cover communication between control systems of centralized applications such as SCADA master station and customer's smart meter. We identify availability non-functional requirement as a core evaluation metric to measure performance of centralized approach with regards to communication and operational delay. Since no study with regards to communication performance of existing centralized solutions has been conducted, for our performance evaluation comparison we use NIST performance requirements for centralized solutions and compare it with decentralized PriWatt protocol in the Table 1.

The results of the evaluation show that the PriWatt protocol provides five times higher operational latency that is expected by NIST from a centralized solution. This happens because the protocol has to trace history of transaction in the blockchain. Although availability latency of our protocol is twice higher, this can be mitigated if more computational power would be added. The protective relaying of centralized solution constitutes automatic shutdown of the connection if abnormal activity is identified. Centralized solution tackles such situations in 4 ms and PriWatt requires 0.6 sec delay to mitigate DDoS attacks.

5.3 Systems Security Analysis and Evaluation

Double spending is a failure mode of digital payment systems, when it is possible to spend a single digital token twice [4]. We extend this notion to the ability of attacker to sell ownership over the same amount of injected and stored energy twice. Various forms of double spending attack may harm PriWatt system. For example, race attack assumes that the attacker could send two conflicting transactions in rapid succession into the network. Victims who accept a payment immediately on seeing "0/unconfirmed" are exposed to this attack. Finney attack is performed by an attacker by including a non-broadcasted transaction of a token into the attacker's block and solving the block without broadcasting it. The attacker then broadcasts a transaction with the same token to a victim. When the victim accepts the transaction, the attacker broadcasts the previously solved block, which would be confirmed and which invalidates the transaction to the victim. Pre-mined block propagation delay attack aims to overcome the case when victim waits for n confirmations before accepting a transaction executed through pre-mining and deliberately delaying propagation of $n+1$ generated blocks to create a fork and make attacker's blockchain longer. This validates malicious transactions and invalidates the correct transactions. Fifty one percent attack requires attacker to control majority of computational power to alter backdated transactions and generate blocks with malicious transactions to perform double-spending attack.

PriWatt enforces the rule that only previously unspent transaction outputs can be used as inputs for new transactions. Majority of honest nodes validate this rule while propagating transactions by traversing and tracing throughout their local copy of blockchain. In the absence of a regulating trusted party, PriWatt uses consensus of majority of the nodes to make decision regarding validity of transactions. In general terms, PriWatt sustains transaction security in the presence of $\mathcal{A}$ malicious nodes with ψ computational power, while honest nodes $\mathcal{H}$ possess computational power γ, such that $\gamma \geq 2\psi + 1$.

The integrity of transactions depends on the collision and pre-image resistance of the SHA256 hashing algorithm. The collision attack implies finding two input strings that hash to the same value. If successful, the attacker could be able to create two different public keys with the same address [8], [9]. SHA256, which is based on the Merkle-Damgard arrangement (the hash value of the previous block is used

TABLE 2
Security and Key-Length Comparison of ECC versus RSA/DSA/DH

Security strength	Key size	
	ECC	RSA/DSA/DH
80 bits	160 bits	1,024 bits
112 bits	224 bits	2,048 bits
128 bits	256 bits	3,072 bits
192 bits	384 bits	7,680 bits
256 bits	521 bits	15,360 bits

as a initialization vector to compute hash of the next block), is expected to be collision resistant due to the underlying compression function.

Preimage attack on cryptographic hash functions presumes to find a message that has a specific hash value. In particular, first preimage attack on PriWatt is given only $SHA256(x)$, where x is the victim's original seed. An attacker needs to use a collision search on the elliptic curve $secp256k1$ points to find a seed message x' that hashes to the same private key $SHA256(x') = SHA256(x)$. The second preimage attack is described as given $SHA256(x)$ and x, attacker needs to find a collision on $RIPEMD\text{-}160(SHA256\ (pubkey))$, i.e., to generate a different keypair, which public key hashes to victim's address, i.e., given one-preimage x of hash y where $y = SHA256(x)$, the task is to find another pre-image of hash y: x' so that $y = SHA256(x')$ [10]. For such birthday collisions, which take $\mathcal{O}(2^{\frac{n}{2}})$ time, SHA256 with the output length of 256 bits would take $\mathcal{O}(2^{128})$, which makes it extremely resource consuming attack with low success probability.

The digital signing guarantees the integrity, authentication and non-repudiation of any content. PriWatt digital signatures are implemented using elliptic curve cryptography (ECC). The major advantage of ECC, compared to other types of asymmetric cryptography such as RSA, is that ECC requires shorter key lengths while providing the same security level (see the Table 2). Compared to a 2,048 RSA key, ECC-256 keys are 64,000 times harder to break.

With regards to signature generation and verification performance comparison, we ran 100 rounds of ECC key generation on 4,096 Mb Intel Core i5 CPU M 540 2.53 GHzx4 of 64 bit Linux machine and received following results (see the Table 3). ECC performance evaluation shows that it is more efficient in signature generation compared to RSA, and therefore for smart metering devices with limitations in terms of processing power, memory and communication bandwidth, ECC is more suitable performance wise.

The security of ECDSA is based on the computational intractability of the Elliptic Curve Discrete Log Problem (ECDLP). Security of ECDLP requires ECC curve's quadratic twist to have a large enough prime divisor p. The quadratic twist of $secp256k1$ has a 220-bit prime factor and therefore is considered as twist secured. At the time of this study the most efficient attacks on ECDLP based cryptosystems are Exhaustive Search, Baby-Step, Giant-Step, Pollards ρ, Pollards λ and Pohlig-Hellman. However, the running time analysis shows that even they run in an exponential time with the best sub-exponential running time of $\mathcal{O}(\sqrt{\frac{\pi n}{2}})$ for Pollards ρ and λ methods, are infeasible given today's technology [11].

TABLE 3
Signature Generation and Verification Benchmark Test

	size	sign (s)	verify (s)	sign/s	verify/s
ECC	160	0.0001	0.0003	12,015.9	3,081.8
RSA/DSA	1,024	0.00031	0.000019	3,220.6	53,656.9
ECC	224	0.0001	0.0002	10,658.5	4,770.9
RSA/DSA	2,048	0.00204	0.000066	490.3	15,257.5
ECC	256	0.0002	0.0004	5,729.1	2,297.3
RSA/DSA	4,096	0.017978	0.000268	55.6	3,731.7

A shortcut to exploit signatures could be taken executing duplicated signatures attack on ECDSA signatures. The attack implies finding two identical signatures with the same nonce for different messages. Signature nonce/seed unpredictability and randomization is crucial, as even partially bit-wise equal random seed values suffice for private key restoration. As such in [12], 158 public keys were found to reuse the same nonce in more than one signature.

5.4 System Attacks and Requirements Evaluation

In order to evaluate security requirements of the system, we used the SQUARE method applied as in [13] to elicit and evaluate security and privacy requirements and measurements. We used the SQUARE to elicit, categorize and prioritize security requirements and evaluate our system accordingly. Due to the space limitations, we present only partial results.

According to Schneier attack tree provides a formal, methodical way of describing security of the system [14]. After analysis of vulnerable assets we identified 23 attacks which are illustrated in Fig. 1 and described in a detailed way in the Table 4. The goal as the root node is to disturb the network. We represented attacks as AND and OR nodes according to Schneier. OR nodes are alternatives—there are seven ways to perform double-spending attack. AND nodes represent different steps to achieve the same goal. To perform invalid curve attack, attacker has to perform elliptic curve fault injection AND obtain scalar multiplication of ECDSA key pair. Attacker cannot successfully perform attack unless both sub-goals are satisfied. In the Table 4, we discuss attacks in a detailed way and point out which asset would be targeted during each attack. The attacks are oriented towards both messaging and transaction infrastructures of PriWatt, where messages are encrypted with RSA and hashing is performed with SHA256 and SHA512.

Using scenario-based elicitation technique we simulated and examined different case scenarios. We resulted with the list of security requirements that is aimed to reduce vulnerabilities combat attacks. Table 5 represents elicited security requirements.

The results showed that 69.57 percent of the identified attacks would make a high impact and 26 percent a medium impact on PriWatt. Fifty two percent of attacks have low likelihood level and only 21.7 percent of identified attacks have high level of likelihood.

5.5 System Threats and Limitations

The main goal of this study was a feasibility and qualitative evaluation of PriWatt. However, there are a number of technology-related issues that are visible.

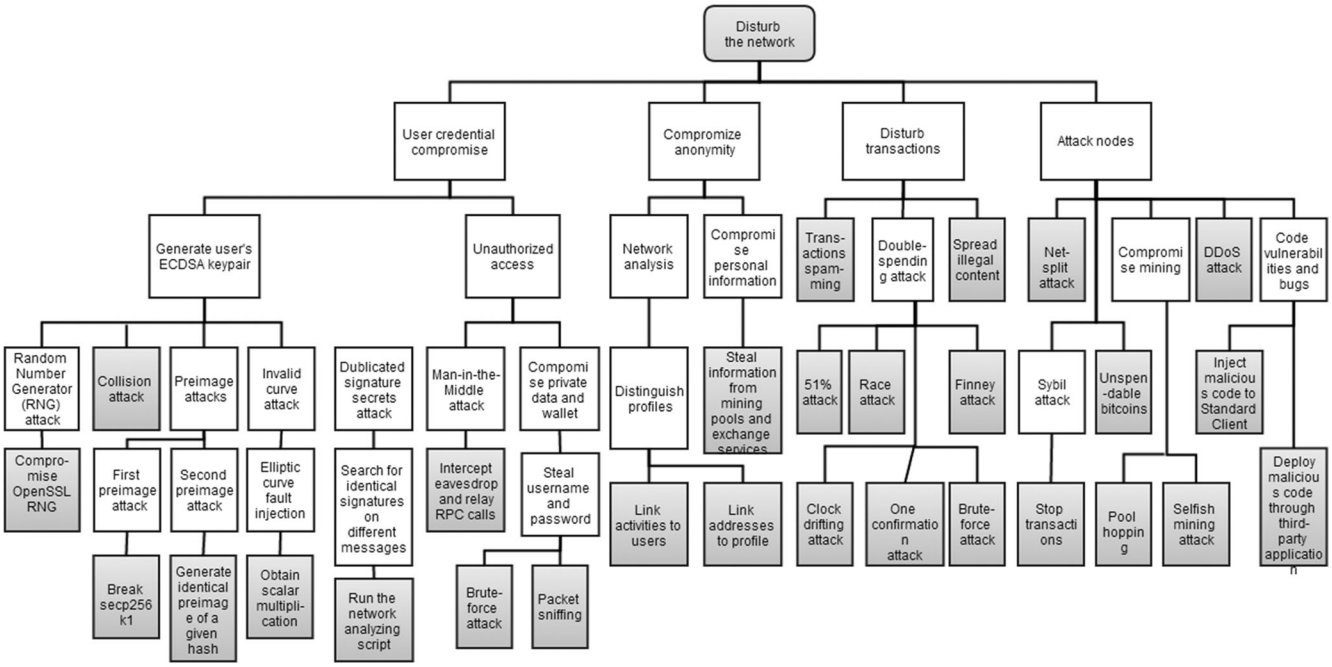

Fig. 1. Attack tree.

Traditional power grids use hierarchical centralization where the utility DSO acts as a local parent node and steps down higher voltage from power generators to residential lower voltage distribution. The advantage of this hierarchical centralization is that it provides authentication and system functions as long as DSO is up. The advantage of P2P networks is that they provide increased security and resilience. PriWatt combines hierarchical centralization and P2P network to provide real world model for energy storage authentication while sustaining privacy and anonymity of P2P nodes.

Some of the advantages of PriWatt are that there is no single point of failure and no bottlenecks; privacy and anonymity are better preserved; better aggregation of large computing resources; and authentication. Some of the disadvantages are message redundancy and lack of routing which might lead to scalability issues. For example, perceived redundancy of additional communication within the system that results in extra cost for bandwidth; the optimal routing calculation, etc. These are inherent to decentralized systems, and it is highly unlikely to achieve an advantage over centralized solutions.

Although PriWatt accommodates potential involvement of a mediating party, its purpose is to act as a distributed energy management system. Transaction security and message integrity are provided without reliance on a centralized trusted third party. At the same time, if preferred in a particular deployment, the system could be configured to act as a centralized system. The long term goal should be to provide both options in order to increase resilience towards particular system conditions and extreme cases during the evolution of the system (e.g., majority of nodes being shut down due to some natural catastrophe or similar.)

An even more challenging issue is to solve the problem of the large scale of data replication where each node must be capable of maintaining the complete blockchain ledger. A full solution to this problem is currently being tackled by the whole Bitcoin community, and this problem is of the size and scope beyond the resources allocated to this project. However, the solutions are on the track and the implementers of a derivative of our solution will be able to integrate the optimal enhancements to the transaction ledger issue.

We have not performed the pure performance simulations and comparison between the solutions since they are redundant. The centralized solution is of constant complexity while the decentralized solution is at least linear. As such, the performance of the centralized solution will eventually outperform the decentralized solution. However, this does not negate a need for decentralized solution. While it could be difficult to justify a decentralized SG as a replacement for better performing centralized SG in developed countries with proper energy infrastructure and privacy and security laws; there are extreme environments where a decentralized solution might be preferred (e.g., undeveloped countries, conflict zones, private microgrids, etc.). In addition, even in the developed world, with the introduction of microgrids and SGs, we might see emergence of many more small (private, communal) suppliers and consumers that should be able to integrate with each other and exchange energy and payments. It is not hard to envision, e.g., a solar-powered car that might not just buy but also sell energy to a small microgrid in a foreign country during vacation travels. A decentralized solution such as the one presented in this work, provides a platform for integration, price discovery, bidding, etc., without having to rely on a traditional centralized SG supplier and business model.

Finally, there are also a number of general open issues due to the maturity of the blockchain technology. These issues are either due to the blockchain technology itself, or due to the public and business sectors adoption challenges.

For example, the issue with the mining procedure is that the nodes that participate in the network can contribute their computational power to add blocks into the

TABLE 4
System Security Attacks

No.	System Attack	Description
A1	Random number generator	Due to poor seed randomness and (if) potential backdoor left by NSA, attackers can exploit vulnerabilities of OpenSSL to create public/private key-pair identical to victim's to hijack transaction and messaging addresses and signatures.
A2	Collision attack	Attacker could try to find two input strings that hash to the same value. If succeeded attacker would be able to create two different public keys with the same address [8], [9].
A3	First preimage attack	Given only $SHA256(x)$, where x is victim's original seed, attacker needs to use a collision search on the elliptic curve $secp256k1$ points to find a seed message x' that hashes to the same private key $SHA256(x) = SHA256(x')$.
A4	Second preimage attack	Given $SHA256(x)$ and x, attacker needs to find a collision on $RIPEMD$-160$(SHA256(pubkey))$, i.e., to generate a different keypair, which public key hashes to victim's address, i.e., given one-preimage x of hash y where $y = SHA256(x)$, the task is to find another pre-image of hash y: x' so that $y = SHA256(x')$ [10].
A5	Sybil attack	Attacker injecting huge number of fake puppet nodes needs to build a private sub-network which sieges and isolates victim nodes from the rest network and can perform malicious activities on victim nodes.
A6	Man-in-the-middle attack	Attack is performed through intercepting and relaying transmitted sensitive data. On messaging infrastructure attacker could eavesdrop on network to identify which messages a victim sends before receiving. Although this would identify that victim is a sender of a particular message, because attacker does not know neither who is recipient nor what is written in message the attack does not pose threat on messaging infrastructure. Nonetheless, because all transactions and recipients are publicly available in a block chain the same attack performed on transaction infrastructure would reveal anonymity of trading parties.
A7	Packet sniffing	Similar to the man-in-the-middle, attacker using penetration testing tools such as NMap and Metasploit could monitor network packets and inspect when victim sent packets such as transactions, messages, acknowledgement and version messages. From this knowledge attacker could identify what was sent before receiving.
A8	Network analysis	Attacker using behaviour-based clustering techniques could map addresses to profiles or transactions to addresses by tracing and analyse the network history of publicly announced transactions [15].
A9	Distinguishability of the profiles	Attacker can connect identities to addresses by tracing transfer history of a token in a block chain [16], [17].
A10	DDoS attack	Attacker can flood inbound connection of a node with invalid large-sized data [18].
A11	Clock drifting attack	First, attacker should force a clock drifting forward of one set of nodes and clock drifting backward for another set creating a gap over two hours. If he could do the first operating then attacker can perform double-spending through create an alternative block chain that would be accepted by victim set of nodes. In addition, blocks generated by the minority set would not be accepted, which will lead to instability of the network.
A12	Spreading illegal content	Attacker send illegal content over messages or can insert illegal content into a transaction and broadcast it to the network [19].
A13	Stopping certain transactions and messages	If attacker would occupy outbound connection slots of a certain user he could opt to propagate only certain messages and transactions discriminating against the victim's. Another scenario, if a malicious mining pool would continuously succeed in solving blocks, it could delay certain transaction from being confirmed by not including transactions into solved block [19].
A14	Spamming transactions	Attacker could disturb the network by increase size of transaction through artificially padding valid transactions inside each other in order to broadcast them in a large numbers [19].
	Double-spending attack	Attacker could spend a single digital token or sell ownership over the same amount of injected energy twice (following five variations) [4].
A15	Race attack	Attacker could send two conflicting transactions in rapid succession into the network. Victims who accept a payment immediately on seeing "0/unconfirmed" are exposed to this attack.
A16	Finney attack	Attacker could include a non-broadcasted transaction of a token to himself into a block and solve a block without broadcasting it. Then broadcasts a transaction with the same token to a victim. When victim accepts the transaction attacker would broadcast a block with a transaction to himself, which would be confirmed and invalidate a transaction to a victim.
A17	One confirmation attack	Combination of the race and Finney attacks such that a transaction that even has one confirmation can still be double spent.
A18	Brute-force attack	Attack is similar to Finney attack, but is aimed to overcome case when victim waits for n confirmations before accepting a transaction. Attacker pre-mines n+1 blocks, which creates a fork and makes attacker's block chain longer. This validates malicious transactions and invalidates righteous.
A19	51 percent attack	If attacker would control majority of computational power he could reverse and alter past transaction, and generate blocks with malicious transactions and perform double-spending attack.
A20	Trace users identity and data	If a block generation infrastructure similar to Bitcoin's mining pools would appear in PriWatt protocol, attacker could exploit victim's private information to obtain financial info or mining capabilities.
A21	Attacks on wallet	Attacker would try to obtain private keys from the victim's wallet to get his money.
A22	Invalid curve attack	If attacker could obtain scalar multiplication with secret scalars on any point of *secp*256*k*1's twists using elliptic curve fault injection [20] then he would perform invalid curve attack and controlvictim's public-private key-pair [21].
A23	Duplicated signature secrets	Attacker would recover victim's private key, if he could find two identical signatures (i.e., used the same random nonce value) on different messages [12].

TABLE 5
PriWatt versus Security and Privacy Requirements

No.	Security and privacy requirements	Solution
R1	The system should provide transaction security while dealing with contracts; prevent mediator to exploit and alter the transaction in his own interest.	PriWatt system maintains security of transactions in distributed contracts through multi-signatures technology where minimum m of n keys must sign a transaction before tokens can be spend. Having more signatures k than which victim exposes to mediator $m-1$, victim may use k to spend tokens before the mediator.
R2	The system should prevent attacker to identify identity of message and transaction sender and receiver.	This requirement is satisfied through messages propagation system of PriWatt; all messages are encrypted with public key of the receiver and sent to everyone. Anonymity of the receiver is preserved because no one knows to whom message is sent.
R3	The system should prevent attacker to intercept and alter transferred message or transaction.	Using Linux Kali we intercepted packets and performed attacks #6 and #7. Although attacks were successful we proved that attacker would not be able to reveal content of packets as PriWatt encrypts messages with RSA and both messages and transactions are signed with private key, hence only a receiver would be able to access content of a message. Even if message was interrupted and altered a minor change of content would break signatures and integrity of the message.
R4	The system should prevent attacker to generate identical to victim's public-private key-pair.	Although hashcash SHA256 and SHA512 relies on the hash partial preimage resistance property, its design is similar to broken SHA1. However, PriWatt uses two hash iterations, which increases immunity for collision and preimage attacks.
R5	The system should prevent key-pair seed's pseudo-randomization.	Every ECDSA signature includes a random nonce value which must never be repeated. PriWatt mixes user input with pseudo random nonce generation increase key-pair seed randomization and obtains non-biased pseudo random number generation.
R6	The system should prevent attacker to violate confidentiality of trading parties or overtake transaction if mediator involvement would be required.	PriWatt system sustains confidentiality of agents in presence of mediator using pseudonyms and multi-signature transactions. Mediator has no knowledge who are trading parties and cannot overtake a transaction becase multi-signature address requires minimum 2 out of 3 signatures to spend a token.
R7	The system should prevent attacker from hijacking transactions.	PriWatt uses asymmetric cryptography to prevent transaction hijacking by signing transactions with private key. Attacker is unable to add changes to transaction without breaking signatures.
R8	The system should prevent attacker to spend digital token or ownership over generated energy more than one time.	There are two types of double-spending exist in PriWatt system: double-spending of token and double-spending of ownership of stored energy. Key components of token double-spending are proof-of-work and nested chain of blocks linked to each other. Double-spending of energy ownership is tackled through locking that amount using b_z secret key and shared messaging address.
R9	The system should prevent attacker from stopping propagation of transactions and messages or shutting down the network.	PriWatt outbounds connection to only one IP address per a range of each network class and uses hard coded DNS servers fall-back nodes to acquire trustworthy nodes.
R10	The system should prevent attacker to masquerade ones identity or lie that it possesses sufficient amount of energy for trading.	PriWatt system uses hybrid topology of p2p network and hierarchical centralization. Hierarchical centralization provides authorization therefore a parent DSO node would validate identity of stored energy owner.
R11	The system should prevent unauthorized access to wallets.	PriWatt wallet is a simple .dat file that could be easily encrypted using various encryption storage tools such as TrueCrypt, Tomb and Ecryptfs-utils.
R12	The system should provide IP anonymization using Tor services.	PriWatt requires headless bitcoind and bitmessagemain running in daemon mode. Configurations could be set to redirect traffic to Proxy IP 127.0.0.1 on 9,050 sockPort and rout traffic through Tor.
R13	The system should be able to easily verify transactions and combat spamming and double-spending attacks.	PriWatt adopts re-usable hard to find, but easy to verify proof-of-work algorithm in order to sustain the system and prevent the double-spending attack.
R14	The system should provide trustworthy connection for users with reliable nodes to connect.	Similar to Bitcoin, PriWatt has two ways to observe nodes: DNS servers and permanent IP addresses of reliable peers. DSO as mediators could be used to serve as DNS servers. When pair of secret keys b_λ and b_z generated, PriWatt uses timestamp of parent DSO nodes rather than timestamp of child nodes. This also tackles clock drifting attack.
R15	The system should prevent attacker to perform denial of service and over-flooding.	PriWatt inherits number of Bitcoin's solutions to tackle DoS and over-flooding such as: restricts the block and script size, maximum number of signature checks for transaction input, value pushed while evaluating a script and etc.
R16	The system should use trustworthy nodes to relay current timestamp.	PriWatt uses hardcoded DNS servers to relay current timestamp.
R17, R18	The system should prevent attacker to link addresses or transactions to a particular user. The system should obligate agents to use new key-pair for every new transaction.	PriWatt client forces agents to generate and use a new key-pair of $txAddr$ and $msgAddr$ addresses for every energy injection and transaction. This increases address unlinkability and untraceability of transactions.
R19	The system should prevent block birthday collision to avoid disputes between sub-blockchains.	Given set of input transactions many nodes blocks could generate the same blocks simultaneously leading to multiple parallel sub-chains of blocks. The system combats forks through *proof-of-work* technique which decreasing probability of blocks birthday collisions.
R20	In presence of malicious node with large computational power, the system should be fault tolerant to computational power race attack.	PriWatt fluctuate the hardness of the proof-of-work according to the number of agents and computational power that is involved in solving blocks.

blockchain, and this process consumes a large amount of energy itself. The cost for electricity might be even higher than the rewards miners will get. The other issue with mining procedure is the 51 percent attack. Due to the low profit of mining procedure, miners can collaborate with each other by joining mining pools to share the rewards. If the computational power of a single mining pool exceeds 50 percent of the total computational power in the network, this entity will have the ability to fabricate transactions and add them to the blockchain [22].

Even with all the potential benefits that blockchain technology can offer, the public and business sectors still have many challenges to overcome even in areas other than energy. The number of transactions blockchain technology can process per second needs to be able to sustain the huge volume of real world transactions. Currently, Bitcoin blockchain can process seven transactions per second, while payment systems tend to process 2 to 56 thousand transactions per second, depending on the time of the year. Due to the fact that assets are recorded in physical form, it will take years to fully digitize everything. Before this goal can be reached, some blockchain applications will need to be able to synchronize the digital ledger with previous records. Moreover, the mechanism of blockchain technology does not allow senders to transact the assets they do not own, which limits the current market convention (i.e., no support for energy credits). There are also many business challenges for corporations to move to blockchain technology.

6 Related Work

In recent years there were conducted several studies on tackling privacy issues of households that can be violated from energy consumption profiles. To ensure privacy of customer's data both trusted based model, that constitutes involvement of trusted third party energy institution, and non-trusted models were proposed.

Mihaylov et al. [23] discuss a mechanism for trading renewable energy in SGs. The authors argue that unlike proposed techniques that rely on predictions, prosumers are billed by distributed system operator according to their actual consumption and rewarded based on actual production. To support payment process Bitcoin based NRGcoin conceptual model was proposed.

Acs and Casteluccia [24] propose a novel trustless scheme for light-weight smart meter which aggregates reading data of consuming devices and adds noise. To avoid the trusted party, the authors propose a novel Laplacian Perturbation Algorithim. Remarkably, the algorithm allows supplier to compute noised and encrypted aggregation of reading data from a set of peers without getting access to the readings of a particular node. Also, the authors point out that the system is robust from adversary nodes and smart meter failures. This is a promising solution for centralized systems.

Dimitriou and Karame [4] discuss the ways to enhance the privacy of users in the SG throughout the reporting and billing phases. The proposed novel anonymous and trustless taxonomy model preserves data privacy during smart meter energy consumption data aggregation and trading. The authors compare centralized versus decentralized solutions, i.e., Central Bank and Bitcoin respectively, for rewarding households and conclude that latter provides necessary functionality to the system.

Kawasmi et al. [25] focus on privacy and security goals by specifying a complete system-of-system model for an anonymous carbon emission trading infrastructure. The authors propose a decentralized system which incorporates Bitcoin peer-to-peer digital currency with carbon emission trading. The system allows trading agents to register, trade and keep track of earned carbon emission reduction credits through a distributed chain of exchange records (blockchain) and digital pseudonymous contracts. It is worth noting that the system carries out all functionalities without revealing identities of agents or requiring central authority to manage transactions.

Efthymiou and Kalogridis [26] discuss a secure mechanism which allows to anonymize readings sent by a smart meter. The presented mechanism includes trusted escrow services to aggregate the reading data for anonymization, but this could be vulnerable for forensic analysis.

Molina-Markham et al. [27] propose a system that allows a smart meter to report its billing whiteout reporting its usage. In the proposed architecture, the smart meter provides aggregated information, including neighboring consumption information to the energy supplier, which could be used to predict the future energy demand. Smart energy system requirements and scheme to protect private data leakage via smart metering billing were discussed by Jawurek et al. in [28]. The authors introduce additional component integrated into smart meter, which transmits only billing data that is signed by the smart meter and verified by the energy supplier.

Sundramoorthy et al. [29] discuss data aggregation and usage for designing domestic energy-monitoring system. The authors have conducted the analysis on the ways to collect and store vulnerable data, and identified data processing and data controlling stakeholders.

7 Conclusion

In this paper we addressed the problem of providing transaction security in decentralized SG energy trading without reliance on trusted third party. We implemented a token-based private decentralized energy trading system that enables peers to anonymously negotiate energy prices and securely perform trading transactions. We used blockchain technology, multi-signatures and anonymous encrypted message propagation streams to provide certain levels of privacy and security. Our system uses peer-to-peer community-based data replication method where transactions are protected from failure since they are replicated among all active nodes. In addition, the proof-of-work, as in Bitcoin, allows the system to overcome Byzantine failures and to combat double-spending attacks which are critical in any electronic payment system.

We modeled and simulated energy trading case scenarios among peers in a SG. We performed security and performance analysis and evaluation. We simulated network related attacks and demonstrated our claim that the system is resistant to significant known attacks; it does not reveal identities of trading parties; and it keeps financial profiles secure and private. We identified and discussed potential attacks and elicited security and privacy requirements.

Overall, we found that the appropriate combination of blockchain technology, multi-signatures and anonymous encrypted message propagation streams presents a feasible and reliable direction towards decentralized SG energy trading with higher privacy and security compared to the traditional centralized trading solutions.

Acknowledgments

Davor Svetinovic is the corresponding author.

References

[1] G. Wood and M. Newborough, "Dynamic energy-consumption indicators for domestic appliances: Environment, behaviour and design," *Energy Buildings*, vol. 35, no. 8, pp. 821–841, 2003.
[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://goo.gl/MqyCW7, Accessed on: Oct. 10, 2016.
[3] J. Warren, "Bitmessage: A peer-to-peer message authentication and delivery system," 2012. [Online]. Available: https://goo.gl/1RwR5, Accessed on: Oct. 10, 2016.
[4] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 906–917.
[5] K. Okupski, "Bitcoin developer reference," 2014. [Online]. Available: https://goo.gl/3PBc9Q, Accessed on: Oct. 10, 2016.
[6] NIST, "Introduction to NISTIR 7628 guidelines for smart grid cyber security," 2010. [Online]. Available: https://goo.gl/KNCmyN, Accessed on: Oct. 10, 2016.
[7] A. Moise and J. Brodkin, "ANSI C12.22, IEEE 1703, and MC12.22 transport over IP," 2011. [Online]. Available: https://goo.gl/HqHwlQ, Accessed on: Oct. 10, 2016.
[8] F. Mendel, T. Peyrin, M. Schläffer, L. Wang, and S. Wu, "Improved cryptanalysis of reduced RIPEMD-160," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2013, pp. 484–503.
[9] Y. Sasaki, L. Wang, and K. Aoki, "Preimage attacks on 41-step SHA-256 and 46-step SHA-512," *IACR Cryptology ePrint Archive*, vol. 2009, 2009, Art. no. 479.
[10] B. Preneel, A. Bosselaers, and H. Dobbertin, "The cryptographic hash function RIPEMD-160," *CryptoBytes*, vol. 3, no. 2, pp. 9–14, 1997.
[11] M. Musson, "Attacking the elliptic curve discrete logarithm problem," MSc Thesis, Acadia Univ., Wolfville, NS, Canada, 2006.
[12] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow, "Elliptic curve cryptography in practice," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2014, pp. 157–175.
[13] H. Suleiman and D. Svetinovic, "Evaluating the effectiveness of the security quality requirements engineering (square) method: A case study using smart grid advanced metering infrastructure," *Requirements Eng.*, vol. 18, no. 3, pp. 251–279, 2013.
[14] B. Schneier, "Attack trees," *Dr. Dobb's J.*, vol. 24, no. 12, pp. 21–29, 1999.
[15] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using P2P network traffic," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2014, pp. 469–485.
[16] M. Fleder, M. S. Kester, and S. Pillai, "Bitcoin transaction graph analysis," arXiv:1502.01657, 2015, https://arxiv.org/abs/1502.01657
[17] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2013, pp. 34–51.
[18] D. Schwartz, "What protection does bitcoin have against denial of service (DoS) attacks?" 2011. [Online]. Available: https://goo.gl/cZo7zt, Accessed on: Oct. 10, 2016.
[19] P. Piasecki, "Design and security analysis of bitcoin infrastructure using application deployed on Google apps engine," M.S. thesis, Politechnika Lodzka, Łódź, Poland, 2012.
[20] I. Biehl, B. Meyer, and V. Müller, "Differential fault attacks on elliptic curve cryptosystems," in *Proc. Annu. Int. Cryptology Conf.*, 2000, pp. 131–146.
[21] A. Antipa, D. Brown, A. Menezes, R. Struik, and S. Vanstone, "Validation of elliptic curve public keys," in *Proc. Int. Workshop Public Key Cryptography*, 2003, pp. 211–223.
[22] A. Extance, "The future of cryptocurrencies: Bitcoin and beyond," *Nature*, vol. 526, pp. 21–23, 2015.
[23] M. Mihaylov, S. Jurado, K. Van Moffaert, N. Avellana, and A. Nowé, "NRG-X-change—A novel mechanism for trading of renewable energy in smart grids," in *Proc. SMARTGREENS*, 2014, pp. 101–106.
[24] G. Ács and C. Castelluccia, "I have a DREAM! (DiffeRentially privatE smArt Metering)," in *Information Hiding*. Berlin, Germany: Springer, 2011, pp. 118–132.
[25] E. Al Kawasmi, E. Arnautovic, and D. Svetinovic, "Bitcoin-based decentralized carbon emissions trading infrastructure model," *Syst. Eng.*, vol. 18, no. 2, pp. 115–130, 2015.
[26] C. Efthymiou and G. Kalogridis, "Smart grid privacy via anonymization of smart metering data," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, 2010, pp. 238–243.
[27] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin, "Private memoirs of a smart meter," in *Proc. 2nd ACM Workshop Embedded Sens. Syst. Energy-Efficiency Building*, 2010, pp. 61–66.
[28] M. Jawurek, M. Johns, and F. Kerschbaum, "Plug-in privacy for smart metering billing," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2011, pp. 192–210.
[29] V. Sundramoorthy, G. Cooper, N. Linge, and Q. Liu, "Domesticating energy-monitoring systems: Challenges and design concerns," *IEEE Pervasive Comput.*, vol. 10, no. 1, pp. 20–27, Jan.–Mar. 2010.

Nurzhan Zhumabekuly Aitzhan received his MSc degree in Computing and Information Science from Masdar Institute of Science and Technology, UAE. He is currently with Accenture, technology consulting, specializing in Business Intelligence and Data Warehousing. His research interests include Big Data analytics, data mining and pattern recognition, and information privacy and security.

Davor Svetinovic is an Associate Professor at Masdar Institute of Science and Technology, UAE. His primary areas of expertise are software systems engineering and systems security. He received his PhD (2006) and MMath (2002) degrees in Computer Science from University of Waterloo, Canada. He received his double BSc (2000) degree with Honours in Mathematics and Computer Science from Bishop's University, Canada. Previously he worked as a visiting professor and research affiliate at the Massachusetts Institute of Technology (MIT); and as a postdoctoral researcher at Lero – the Irish Software Engineering Center, Ireland, and Vienna University of Technology, Austria. He leads the Strategic Requirements and Systems Security Group (SRSSG), and he has extensive experience working on complex multi-disciplinary research projects. He has published over 50 papers in leading journals and conferences. His current research interests include software engineering, systems security and privacy, smart grids energy trading, and digital currencies.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.