Trust-based Communities for Smart Grid Security and Privacy

Seohyun Park, Xiang Li, and Yuhong Liu

Department of Computer Science and Engineering,
Santa Clara University,
Santa Clara, CA, USA
{spark4,xli8,yhliu}@scu.edu

Abstract. In smart grids, two-way communication between end-users and the grid allows frequent data exchange, which on one hand enhances users' experience, while on the other hand increase security and privacy risks. In this paper, we propose an efficient system to address security and privacy problems, in contrast to the data aggregation schemes with high cryptographic overheads. In the proposed system, users are grouped into local communities and trust-based blockchains are formed in each community to manage smart grid transactions, such as reporting aggregated meter reading, in a light-weight fashion. We show that the proposed system can meet the key security objectives with a detailed analysis. Also, experiments demonstrated that the proposed system is efficient and can provide satisfactory user experience, and the trust value design can easily distinguish benign users and bad actors.

Keywords:
Smart Grid, Data Aggregation, Blockchain, Security, Privacy.

1 Introduction

Smart grid enables sharing information about electricity demand and supply in real time, which improves energy efficiency and provides more features to end-users [1]. With Internet of Things (IoT) applications in smart grids, such as smart meters and IoT appliances, two way communications between end-users and the grid allows frequent meter readings, which leads to more accurate prediction and further optimization of supply and demand. Also, certain tasks, like laundry, can be scheduled at off-peak hours to smooth demand fluctuation and reduce the energy bill of end-users [2].

However, two way communication and high frequency meter reading also raise security and privacy concerns. For example, a cyberattack (e.g. man-in-the-middle attack) may gain access to detailed electricity usage data, such as running laundry or watching television during certain hours of a day, which can be leveraged to infer more sensitive information, such as vacancy in the house [3], and may further lead to security risks like break-ins. Data aggregation at local gateways [4] may mitigate some of the privacy issues, but the computational overhead can be high [5].

To efficiently address the security and privacy problems, in this paper, we propose a new design, a community based blockchain solution: the end-users are grouped in communities and each community holds a local blockchain, which aggregates electricity usage data of members in the community and reports back to the service provider. Within a community, a time-based trust model is designed to detect bad actors who are not cooperative or having fraudulent behaviors. Compared to cryptography-heavy schemes, the proposed solution is much more lightweight and efficient.

The contributions of this paper are summarized as follows.

- We proposed a community based blockchain solution to address privacy problems in eletricity usage data reporting in smart grids.
- We proposed a time-based trust model, a lightweight solution to security problems in community operations.
- We did extensive analysis on how the proposed system can achieve the security goals and experimentally verified the performance of the proposed system and the effectiveness of the trust value design.

Organization. The rest of this paper is organized as follows. The related works are discussed in Section II. We describe the proposed system in Section III. In Section IV and Section V, the security and the performance aspects of the proposed system are discussed, respectively. Section VI concludes the paper.

2 Related Works

Various security aspects in smart grids are discussed in [3]. Security in smart grid is crucial as a large amount of data is generated, transmitted and possibly exposed. Various cryptographic countermeasures are developed to fight against security threats [6, 7].

Privacy in smart grids has been studied from multiple aspects. In [8], a privacy preserving scheme using cryptographic methods was proposed for incentive based demand response in smart grids. In [9], the authors proposed autonomous demand side management based on game theoretic energy consumption scheduling while preserving users' privacy. Multiple privacy preserving data aggregation schemes are also proposed [4, 10, 11]. At the core of the works, encryption schemes are designed such that only aggregated data can be learned after decryption.

Traditional cryptography-based methods for smart grid security and privacy may impose a large computational overhead [12] to the light-weight IoT devices in smart grids, also, bad actors may circumvent access control to damage the system. Hence, blockchain techniques are studied in smart grids for privacy and security gains [13, 14]. For general IoT systems, [15] explored the applicability of blockchain techniques, focusing on hardware specifications and standard, while [16] proposed the implementation of local blockchain in IoT without Proof of Work and coins. For smart grids, blockchain-based applications like anonymous authentication with key management [17] and distributed energy trading [18] were proposed. However, existing works do not provide privacy-preserving electricity usage transmission with awareness of bad actors in the system.

Blockchain-based trust management system for bad actor detection is described in [5], in which the trust is between actors (end-users) in the system. In

comparison, we propose to maintain trust at the local community level, which is more suitable for the application we focus on.

3 Time-based Trust Community System Design

In this section, we first discuss the formation and maintenance of local communities of end-users, the organizational structure of the local community based blockchain, and then introduce detailed operations of data aggregation and the time-based trust system.

3.1 Community Formation and Maintenance

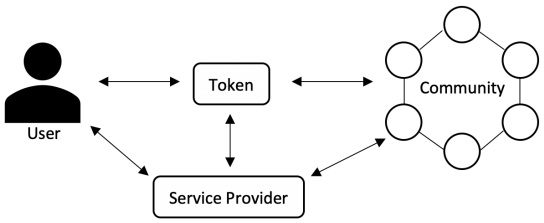

Fig. 1: Registration Process

3.1.1 User Registration Figure 1 shows the registration process when a user signs up with a smart grid service provider. After the user is verified as benign from the service provider, the user generates a pseudo-random number (token) and shares it with the service provider. Once a token is created, user only communicates with the service provider and the community through the token until the account termination is requested. Once the registration is finished, the user is allowed to join a community.

3.1.2 Community Formation There are two main guidelines when forming communities. 1) Each community should have an appropriate size. If the communities are too small, we may not have the security and privacy guarantees from blockchain and data aggregation. If the communities are too large, it may be too costly for the smart meters to maintain a blockchain due to their limited computing power. 2) Members of a community should be in close proximity. On one hand, it reduces the communication cost within community. On the other hand, users with close proximity may share similar electricity usage patterns, generation patterns (for example, households with solar panels) and policies (for example, different regions may have different baseline utility rates). Hence, having location based communities can help the service provider better optimize energy generation.

We see community formation as an adaptive problem [19]. After initial network setup, the service provider stores the community structure and updates the assignment adaptively based on new users' enrollment/termination and the existing structure, without recomputing the community structure for the whole network from scratch.

Algorithm 1 Community Structure Update

Input: G, $\Delta G^{(t)}, t = 0, ..., T$, $\Delta E^{(t)}, t = 1, ..., T$
$C^{(0)}$=Community-Formation(G, $\Delta G^{(0)}$, $\emptyset$)
for t=1 to T **do**
    $C^{(t)}$=Community-Formation(G, $\Delta G^{(t)}$, $C^{(t-1)}$)
    for $v \in \Delta E^{(t)}$ **do**
        $C^{(t)}$ = User-Termination(G, v, $C^{(t)}$)

Algorithm 2 Scan Community

Input: d_{max}, communities C, node v
$SC = \emptyset$
for c in C **do**
    if Distance between c and v is less than d_{max} **then**
        add c to SC
Return SC

Algorithm 3 User-Termination

Input: M_{min}, set of all communities C, users to terminate V, v's community $c_v, \forall v \in V$. $C_{remove} = \emptyset$
for $v \in V$ **do**
    Remove v from c_v.
    if c_v.size $< M_{min}$ and c_v not in C_{remove} **then**
        Add c_v to C_{remove}
$V_{assign} = \cup_{c \in C_{remove}} c$
Community-Formation(G, V_{assign}, C)

Algorithm 4 Community-Formation

Input: d_{max}, M_{max}, set of nodes to assign V, set of communities C.
for node $v \in V$ **do**
    SCs = Scan-Community(C, d_{max})
    if SCs is empty **then**
        Create a new community c and add to C
        Add v to c
    else
        $SC_{avail} = \emptyset$
        for $SC \in SCs$ **do**
            if $SC.size < M_{max}$ **then**
                add SC to SC_{avail}
        if SC_{avail} is not empty **then**
            Find the closest community $c' \in SC_{avail}$ to v
            Add v to c'
        else
            Find the closest community $c^* \in SCs$ to v
            Split c^* to c^1, c^2 based on proximity, $|c^1| \approx |c^2|$.
            add v to the closer of c^1, c^2
Return C

The overall community formation is described in Alg. 1. $C^{(t)}$ is the community structure at time t, $\Delta G^{(t)}$ and $\Delta E^{(T)}$ are the user enrollments/terminations at time t respectively and $G^{(0)}$ is the first network snapshot. The community structure at time t is calculated based on the network structure G, updates at time t and the existing community structure. For the base case $t = 0$, there exists no calculated community structure, so we use empty set as an input to Alg. 4 in this case.

In Alg. 4, d_{max} is the maximum allowed distance between a node and the center of a community it can join, where the center of a community is the Jordan center [20] of the subgraph formed by nodes in the community. M_{max} is the maximum number of members a community can hold. For each node v, the algorithm first tries to find communities within close proximity of v using Alg. 2. If no such communities can be found, a new one is created to hold v. Otherwise, the algorithm checks if the communities have open slots for v. If so, v is added to the closest one. If not, the closest community is split to two communities of approximately equal size (depending on whether there's odd or even number of members) and v is added to the closer one of the two new communities. The blockchain of the original community will be forked. A community will not start working until the number of members reaches the minimum threshold M_{min}. Users in such small communities will fall back to traditional reporting methods.

When users V stops service, they will need to be removed from their communities $c_v, \forall v \in V$. As shown in Alg. 3, the size of communities c_v must be checked to make sure they are still large enough. If the size of c_v falls below the minimum threshold M_{min}, it will be removed from the community structure and Alg. 4 will be called for all the remaining members of c_v to find their new communities.

Time Complexity. For Alg. 2, since the max number of communities scanned is n where n denotes the total number of nodes, it's complexity is $O(n)$. Alg. 4 has time complexity $O(n)$ for each new node to be added, as scan community (line 3), check if communities are within size limit (lines 9-11), community split (lines 16-18) can all be done in $O(n)$ time and the rest operations are $O(1)$. Hence, the overall time complexity of Alg. 4 is $O(|V|n)$ where V is the set of nodes that need community assignment. For Alg. 3, the complexity can be $O(n^2)$ in the worst case when community removal is needed for all remaining nodes, but it is a very rare case in practice. In most of the cases, it has $O(|V|)$ complexity. For Alg. 1, at each time step, we will call Alg. 4 at most twice for each node, hence the overall time complexity for Alg. 1 is $O(Tn^2)$ for all T time steps.

3.1.3 Genesis Transaction Once community formation is completed, the service provider generates a local blockchain for each community and the blocks are not shared with other communities. For each community, a genesis block is created by the service provider, so that members of the community can create blocks following the genesis block. Figure 2 shows the structure of local blockchain with the genesis block. The service provider shares a key generated using generalized Diffie-Hellman with a new user to join the community and the policy header gets updated to include a new member in the community.

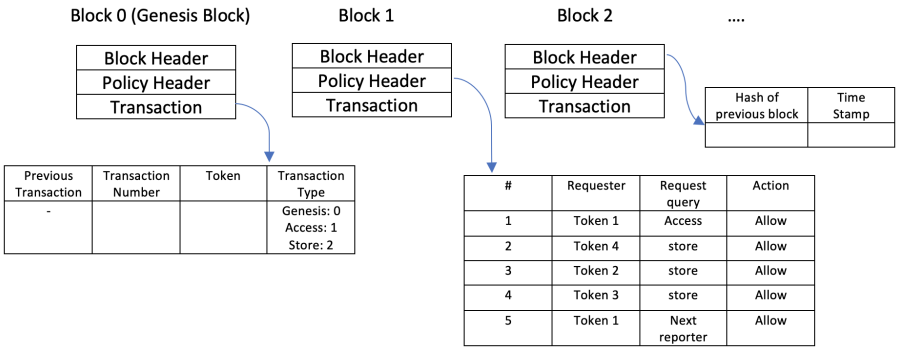

Fig. 2: Local Blockchain structure

3.2 Data Aggregation

For a community c with enough members, each time a meter reading is required, a reporter is selected within the community from the latest block to aggregate the electricity usage data and report to the service provider so all members listed in the policy header can participate in the data aggregation. For each report, every member M_i sends encrypted data D_i from their smart meter to the selected reporter.

$$\sigma_i = x_i H(D_i||M_i||t) \tag{1}$$

where σ_i is a signature signed with private key x_i, H is a cryptographic hash function, and t is the time it is generated. In this process, the records are created in the local blockchain in the form of a transaction. Then, the reporter will aggregate the received data in a privacy preserving way [4].

$$D = \sum_{i=1}^{n} D_i \bmod n^2 \tag{2}$$

After data aggregation, it is signed with the private key of the reporter to create a signature σ_r.

$$\sigma_r = x_r H(D||M_r||t) \tag{3}$$

Then, $D||M_r||\sigma_r$ is reported to the service provider.

Communication Costs. During data aggregation, there will be communication costs within each community to reach consensus. For the communication overhead, each user sends a 2048 bit ciphertext to the reporter, so the reporter receives $2048n$ bits of ciphertext in total, then after the data aggregation, the reporter sends a 2048 bit ciphertext to the service provider. Meanwhile, the consensus can be reached with linear communication cost of upper bound $2loglog log(n)$, and lower bound $loglog log(n)$ given that we have m bits of memory where $m > 3loglog log(n)$ [21]. This result shows that the proposed architecture can be implemented to the hardware that lacks high computing power and memory resources.

3.3 Time-based Trust System

While having blockchain-based local communities in smart grids brings privacy benefits, maintaining benign communities arises as a new issue. Each member has a power in consensus, so compromised members can affect others. A more severe situation is when a compromised member is selected as a reporter. To reduce the impact from compromised members/bad actors in a blockchain, either proof of work (PoW) [22] or trust [5] may be used. We argue that PoW cannot work well in the case of smart grids, since smart meters have computational resource limits and are unlikely to outperform the bad actors. On the other hand, a carefully designed trust system may be suitable for smart grids.

In the system, trust is used to vote and report. When reporting electricity usage data, the aggregated report is generated with the consensus from community members, and members with higher trust value have higher voting power compared to those with lower trust value. Also, members with low trust values may be reported for suspicious activities. We let trust be a numerical value in range $[0, 1]$. At the time of enrollment, all new members are given a relatively high trust value. Then, two factors may change the trust value: time and interaction. Trust has the time-decaying property because it is more likely to have a firmware update overdue from the service provider and old firmware is more prone to the attacks. To formulate this:

$$T(t) \propto T(0) \times e^{\frac{-t}{\tau}} \tag{4}$$

Where T(t) is the trust value at time t, $T(0)$ is the initial trust value and τ is the time constant. This equation is derived from Newton's law of cooling. Since trust decreases over time, users have to participate in the community's consensus and update the firmware to slow down the trust loss.

We also associate trust with the participation in community activities. We adapt the idea of reputation management in P2P networks [23] and define the value S_{ic} as:

$$S_{ic} = sat(i, c) - \gamma \times unsat(i, c) \tag{5}$$

Where $sat(i, c)$ and $unsat(i, c)$ is the number of satisfactory/unsatisfactory transactions member i had within a community c. Here the unsatisfactory transactions are disagreements to the consensus or simply no participation. $\gamma > 1$ is a tunable parameter to penalize unsatisfactory transactions.

The value is further normalized with a sigmoid function to S'_{ic}, where

$$S'_{ic} = \frac{e^{S_{ic}}}{e^{S_{ic}} + 1} \tag{6}$$

$S'_{ic} \in (0, 1)$ and for newly enrolled users wiht $S_{ic} = 0$, $S'_{ic} = 0.5$. Denote

$$\Delta v = \begin{cases} 0 & \text{latest security update is installed} \\ t - t^r_{v_i+1} & \text{latest security update is not installed} \end{cases}$$

Where t is the current time and $t^r_{v_i+1}$ is the release time of the oldest security update that i have not installed. Also denote f as the security update frequency, the trust value can be represented as

$$T_{ic}(t) = T_{ic}(0) \times e^{\frac{-t}{\tau} \times (1 - \frac{S'_{ic}}{1+\Delta v/f})} \tag{7}$$

Hence, when a member i has a lot of satisfactory transactions ($S'_{ic} \approx 1$) and always keep the security software up-to-date ($\Delta v = 0$), its trust value will be close to 1.

When a community c is split, members of c will carry their trust value in c to their new communities. When a community c is removed, members of c will join new communities as new members. If their original trust value is lower than a threshold, they may need to be re-authenticated before joining.

4 Security Analysis

Confidentiality, Integrity, and Availability (CIA) are the key objectives in the security requirements [24]. In this section, we will first discuss how these three objectives are achieved with the proposed design. Next, we will describe the replay attack scenario and how the proposed design can prevent such attacks. Then, we will run a simulation to demonstrate how the trust system can detect bad actors in the smart grid.

4.1 CIA Analysis

4.1.1 Confidentiality As maintaining the trust value is required for the members, it provides assurance to benign members that they can trust each other for the data transmission. Moreover, the data that they transmit is encrypted and the reporter does not have the private key for other members to decrypt and access the contents. From the outside of the community, an adversary may obtain the data. However, since the data is aggregated, though the adversary may gain the accesses to the data, the data won't be linked to the individual usage.

4.1.2 Integrity Data integrity is a key part in the smart grid both for billing and rewards and for the correct and accurate demand prediction. As each local blockchain generates blocks for each of their reports to the service provider, the integrity can be traced back and validated through the generated blocks. As the blockchain blocks are tamper proof, the data integrity can be maintained.

4.1.3 Availability The design with randomized reporters in local communities mitigates the possible unavailability of centralized aggregators like local gateways [4]. In the cases that an attacker tries to steal the aggregated data in the middle of the transmission or to simply block the transmission to achieve denial of service, it will be harder for the attacker to track and steal or make the data unavailable as the reporter changes every time.

If any attempt of an attack is made, instead of the service provider observing the anomaly, it can be enabled with rule based anomaly detection system by

allowing community members to report other participants within the community. We propose five specific rules, summarized in Table. 1, to detect bad behaviors. For example, if a member failed to participate in community activities for certain period of time or denied most of the consensus, its trust value can be lower than a threshold and the other members can report to the service provider to re-authenticate the member. In addition, time-dependent price (TDP) refers to the dynamic pricing the service provider places on the electricity rate to control the demand. Smart appliances are capable of receiving the TDP and run the task when the price is low in the day, which opens up the vulnerability of False Data Injection (FDI) attack. In FDI attacks, malicious users may report data mismatching the TDP. However, those users will have low trust values as they violate the rules, hence the attack can be greatly mitigated.

If there are no threats initiated from attackers, anomaly detection system can be applied to regular operations to remind the members to update their firmware to restore trust, as it naturally decays over time.

Table 1: Behavior Rules for monitoring members

	Rule Descriptions
1	Report a member who does not participate in the consensus.
2	Report a member who falls below the threshold of trust value.
3	Report a reporter who does not agree with the consensus.
4	Share time-dependent price (TDP) with members and report if mismatch occurs.
5	Compare the TDP with previous date and report if mismatch occurs above the threshold.

4.2 Replay Attack Prevention

In this section, we discuss how the proposed system can prevent the replay attack.

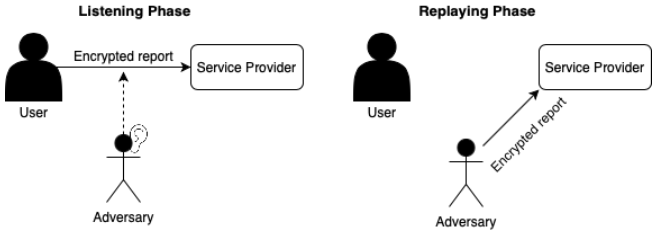

Fig. 3: Replay Attack Procedure

4.2.1 Attack Scenario Replay attack is an eavesdropping attack where an adversary intercepts the conversation between two parties and use the replay of

the message in favor of the adversary. Typical replay attack has two phases shown in Figure 3. First phase is listening to the user's communication and storing the encrypted message with the signature, in our case, the aggregated report. The second phase is replaying or re-sending the exact message without decrypting to the service provider from the adversary. When the encrypted communication occurs with the timestamp included, two parties negotiate the discrepancy interval Δt to accept the valid messages [25] and the replay attack can still occur if the adversary can replay the recorded message within the negotiated discrepancy time period. Suppose Δt is static throughout the communication, the adversary can record the valid message in t_r where t_r is the time spent to record the valid message and replay to the service provider within $\Delta t - t_r$ to be treated as valid messages. A replay attack may prevent a proper connection between two benign parties, resulting in unavailability. It can be significantly harmful to smart grid users as they may lose their service.

4.2.2 Protection Scenario Conventional approach to prevent replay attacks attaches the timestamp with a prefixed discrepancy [8, 26]. However, as discussed above, a replayed message with small delay may still be treated as valid. To be more resistant against replay attack, we show that randomly changing the reporter within the community can be helpful, since it gives an extra verification step for the service provider along with the timestamp. Even if the adversary eavesdrops and replay the report within the negotiated discrepancy, the service provider will expect next report from another user, invalidating the duplicate report and flagging the anomaly.

Randomized reporter also prevents specific nodes in the community from becoming special target to the adversary as well as becoming overloaded both in communication and computational costs. Even if a reporter is attacked, e.g. by a denial-of-service attack, the overall operation can continue with another reporter, and an anomaly report will be sent by other members in the community due to lack of participation.

5 Performance Analysis

5.1 Experiment Setup

Data Set. In this experiment, we use the Pegase 9241 node network from Pandapower [27]. Since we want to simulate a set of smart grid consumers, only the loads and their geographic locations are extracted from the data set. In total, we have 4,461 nodes.

Simulation Method. To simulate a real smart grid with user enrollment and termination, we start the smart grid with 3,000 random users and simulate 100 time steps. At each step, we randomly terminate 1-2 smart grid users and randomly enroll 1-2 non-smart grid users. In each experiment below, we ran the simulation 1,000 times and report the average numbers.

Metrics Considered. We consider both the efficiency of community formation and user experience, with the following metrics:

– **Number of community splits.** Throughout the simulation, how many communities are split. Community split may increase operational burden

and may be bad for users in them. As splitting a community requires communicating with every member in the community, forking the blockchain, and assigning members to new communities.

- **Number of community removals.** Throughout the simulation, how many communities are removed due to size smaller than minimum after user termination. Community removal causes disturbance to members of the removed communities.
- **Number of non-community users.** The average number of users who can not join a community with adequate size in each time step. Those users may be distant from the existing communities and may have to report to the service provider on their own, resulting in a higher security risk.
- **Running time.** Total time taken for community formation simulation.

5.2 Simulation Results

We observe the change in the above four metrics varying the maximum community size and the maximum allowed distance between a member and the community center. The max community size varies between $[100, 300]$ and max allowed distance varies between $D/5$ to D, where D is the maximum distance between any two nodes. Min community size is set to be $1/4$ of max community size.

Since there are two variables, we use heat maps to illustrate the experiment results in Figure 4 to Figure 7.

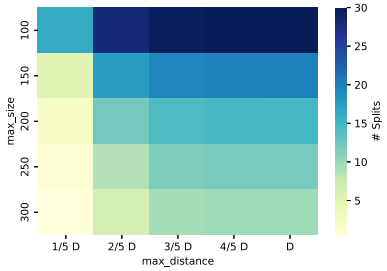

Fig. 4: Number of Community Splits

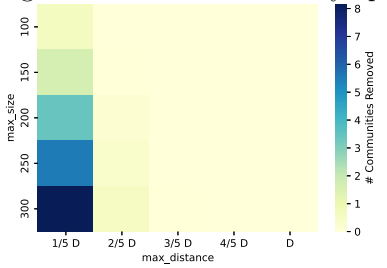

Fig. 5: Number of Community Removals

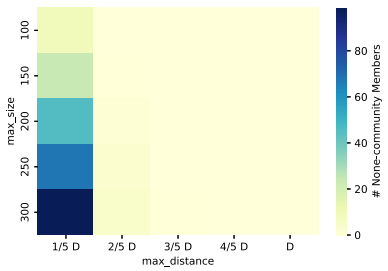

Fig. 6: Number of Non-community Users

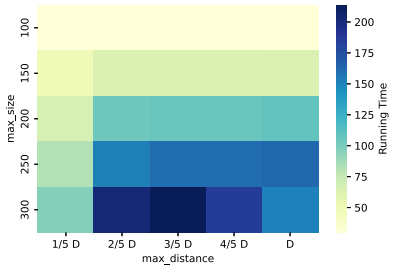

Fig. 7: Running Time (seconds)

From Figure 4, we can see that more splits may happen when the maximum community size is smaller and the max allowed distance is larger. Figure 5 and Figure 6 shows similar trends for the number of non-community users and the number of community removals. Both values are small when max distance is at least $2/5D$. For non-community users, it is expected, since a large enough max distance grants the users to find suitable communities to join. For number of community removals, when max distance is small, the members in a community must be closer to each other, so when some of the members terminated service, it is harder to find new members to join. As for running time, we can see in Figure 7 that the time taken for community formation is longer when max community size is larger, since more updates are needed within a community when a new member joins. In general, the algorithms can finish initial community setup and all updates efficiently, as the total running time never exceeds 200 seconds in all cases. It is notable that the running time does not always grow monotonically with max distance, as in the case when max community size is 300. It is possibly due to large time consumption from rare community removal cases for max distances $2/5D$ and $3/5D$, since there tend to be more communities to choose from when enrolling members from a removed community. While for larger max distance, community removal will not happen at all.

5.3 Trust Updates

In this section, we track trust value updates for different types of users to show that the proposed trust system can easily differentiate the anomaly users.

We construct the users by varying two parameters: probability to participate in community activities (P_a) and probability to install a security update (P_s). At each time step, a user will participate in data aggregation and reporting with probability P_a. If the latest security update is not installed, the user will install it with probability P_s.

Throughout this experiment, we set the trust parameters as follows: time constant $\tau = 2$, unsatisfactory transaction penalty constant $\gamma = 3$. Also, we assume a report is sent for every time step and a security update happens every 24 time steps ($f = 24$).

5.3.1 Trend of Trust Value. To have a clear picture of how trust value changes overtime, we simulate reporting and security updates for 100 time steps for the following types of users:

- **Ideal User:** $P_a = P_s = 1$, the user participates in all activities and always install updates as soon as possible.
- **Normal Benign User:** $P_a = P_s = 0.9$, the user is generally cooperative but may forgot to participate or update time-to-time.
- **Benign User Slow Update:** $P_a = 0.9, P_s = 0.3$, the user is generally cooperative but is slow on keeping software up-to-date.
- **Updated but not Too Active:** $P_a = 0.6, P_s = 0.9$, the user is generally up-to-date for software but often ignores community activities.
- **Inactive:** $P_a = 0.2, P_s = 0.2$. The user is mostly inactive.

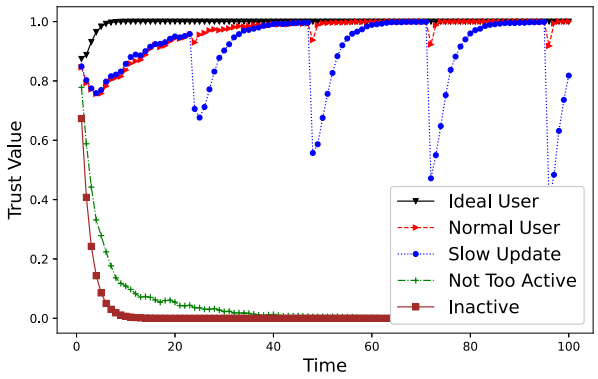

Fig. 8: Trust changes for different types of users

From Figure 8, we can see that users with high P_a value generally have high trust. If they do not always install security update on time, their trust value may have certain dips, but the trust value can recover. However, for users with low P_a, their trust value will soon become very low, the lower the P_a value, the faster the drop. Even high P_s won't help.

5.3.2 Time to Become Untrusted. To better understand the impact of P_a and P_s to how soon the user becomes "untrusted" (trust value below a threshold, we vary P_a, P_s and log the first time step that the average trust value drops below 0.1. For each pair of P_a, P_s, we run simulation for $1,000$ time steps. The result is illustrated in Figure 9. It is clear that users will never become untrusted as long as they are relatively active ($P_a >= 0.8$) and will eventually install security updates ($P_s >= 0.2$). Otherwise, the user will soon become untrusted. It is expected since we do want the system to penalize unsatisfactory transactions more. For security updates, it is fine to install them with a delay.

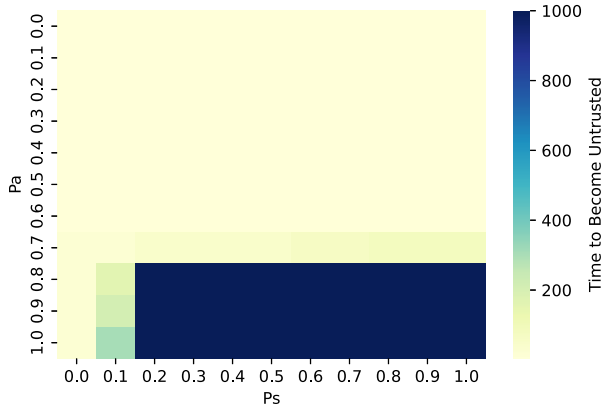

Fig. 9: Time for a User to Become Untrusted

6 Conclusion

In this paper, we designed a trust-based community system with local blockchains to address security and privacy issues in smart grid, with a focus on problems in frequent meter readings. The proposed trust manage system helped with efficient data aggregation and can also be used to detect bad actors. We conducted detailed security analysis to show that the proposed system meets the key security objectives. Also, we demonstrated the performance of the proposed system through experiments.

Acknowledgement

This work was supported by NSF CNS-1948550.

References

1. Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. Smart grid—the new and improved power grid: A survey. *IEEE communications surveys & tutorials*, 14(4):944–980, 2011.
2. A Rezaee Jordehi. Optimisation of demand response in electric power systems, a review. *Renewable and sustainable energy reviews*, 103:308–319, 2019.
3. Pardeep Kumar, Yun Lin, Guangdong Bai, Andrew Paverd, Jin Song Dong, and Andrew Martin. Smart grid metering networks: A survey on security, privacy and open research issues. *IEEE Communications Surveys & Tutorials*, 21(3):2886–2927, 2019.
4. Rongxing Lu, Xiaohui Liang, Xu Li, Xiaodong Lin, and Xuemin Shen. Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. *IEEE Transactions on Parallel and Distributed Systems*, 23(9):1621–1631, 2012.
5. Omaji Samuel, Nadeem Javaid, Adia Khalid, Muhammad Imrarn, and Nidal Nasser. A trust management system for multi-agent system in smart grids using blockchain technology. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.
6. Wenye Wang and Zhuo Lu. Cyber security in the smart grid: Survey and challenges. *Computer networks*, 57(5):1344–1371, 2013.
7. Zakaria El Mrabet, Naima Kaabouch, Hassan El Ghazi, and Hamid El Ghazi. Cyber-security in smart grid: Survey and challenges. *Computers & Electrical Engineering*, 67:469–482, 2018.
8. Yanmin Gong, Ying Cai, Yuanxiong Guo, and Yuguang Fang. A privacy-preserving scheme for incentive-based demand response in the smart grid. *IEEE Transactions on Smart Grid*, 7(3):1304–1313, 2015.
9. Amir-Hamed Mohsenian-Rad, Vincent WS Wong, Juri Jatskevich, Robert Schober, and Alberto Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *IEEE transactions on Smart Grid*, 1(3):320–331, 2010.
10. Weiwei Jia, Haojin Zhu, Zhenfu Cao, Xiaolei Dong, and Chengxin Xiao. Human-factor-aware privacy-preserving aggregation in smart grid. *IEEE Systems Journal*, 8(2):598–607, 2013.
11. Hawzhin Mohammed, Samet Tonyali, Khaled Rabieh, Mohamed Mahmoud, and Kemal Akkaya. Efficient privacy-preserving data collection scheme for smart grid ami networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

12. Himanshu Khurana, Mark Hadley, Ning Lu, and Deborah A Frincke. Smart-grid security issues. *IEEE Security & Privacy*, 8(1):81–85, 2010.
13. Andrija Goranović, Marcus Meisel, Lampros Fotiadis, Stefan Wilker, Albert Treytl, and Thilo Sauter. Blockchain applications in microgrids an overview of current projects and concepts. In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 6153–6158. IEEE, 2017.
14. Muhammad Baqer Mollah, Jun Zhao, Dusit Niyato, Kwok-Yan Lam, Xin Zhang, Amer MYM Ghias, Leong Hai Koh, and Lei Yang. Blockchain for future smart grid: A comprehensive survey. *IEEE Internet of Things Journal*, 8(1):18–43, 2020.
15. Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.
16. Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE, 2017.
17. Jing Wang, Libing Wu, Kim-Kwang Raymond Choo, and Debiao He. Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure. *IEEE Transactions on Industrial Informatics*, 16(3):1984–1992, 2019.
18. Naiyu Wang, Xiao Zhou, Xin Lu, Zhitao Guan, Longfei Wu, Xiaojiang Du, and Mohsen Guizani. When energy trading meets blockchain in electrical power system: The state of the art. *Applied Sciences*, 9(8):1561, 2019.
19. Thang N Dinh, Nam P Nguyen, Md Abdul Alim, and My T Thai. A near-optimal adaptive algorithm for maximizing modularity in dynamic scale-free networks. *Journal of Combinatorial Optimization*, 30(3):747–767, 2015.
20. S Mitchell Hedetniemi, EJ Cockayne, and ST Hedetniemi. Linear algorithms for finding the jordan center and path center of a tree. *Transportation Science*, 15(2):98–114, 1981.
21. Giulia Fanti, Nina Holden, Yuval Peres, and Gireeja Ranade. Communication cost of consensus for nodes with limited memory. *Proceedings of the National Academy of Sciences*, 117(11):5624–5630, 2020.
22. Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols (extended abstract). secure information networks (s. 258-272), 1999.
23. Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003.
24. Nikos Komninos, Eleni Philippou, and Andreas Pitsillides. Survey in smart grid and smart home security: Issues, challenges and countermeasures. *IEEE Communications Surveys & Tutorials*, 16(4):1933–1954, 2014.
25. Dorothy E Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, 1981.
26. Zhitao Guan, Yue Zhang, Liehuang Zhu, Longfei Wu, and Shui Yu. Effect: An efficient flexible privacy-preserving data aggregation scheme with authentication in smart grid. *Science China Information Sciences*, 62(3):1–14, 2019.
27. Leon Thurner, Alexander Scheidler, Florian Schäfer, Jan-Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, 2018.