Sourcing Trust From Peers with Physical Unclonable Functions

Md Sadman Siraj*, Aisha B Rahman*, Cyrus Minwalla†, Eirini Eleni Tsiropoulou* *Senior Member, IEEE*, Jim Plusquellic*, *Senior Member, IEEE*
{mdsadmansiraj96, arahman3, eirini, jplusq}@unm.edu*, cminwalla@bank-banque-canada.ca†
*Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, USA
†Financial Technology Research, Bank of Canada, Ottawa, ON, Canada

***Abstract*—Traditional authentication schemes rely heavily on trusted third parties that act as certificate authorities to source authentication credentials. In this paper, we describe an offline, end-to-end mutual authentication and session key generation protocol, called PUF-based Peer Trust (PPT), where authentication and session key generation between two parties (Alice and Bob) is implemented using physical unclonable functions (PUFs) and authentication information from neighboring peer devices (Charlies). Trust among the parties is established by defining an effort-reward model based on Contract Theory that leverages the trust beliefs provided by a set of peer devices. Alice autonomously selects a set of trusted Teds from the set of untrusted Charlies using a reinforcement learning algorithm based on Stochastic Learning Automata. These Teds participate in a peer trust authentication process designed to help Alice decide if Bob is trustworthy. The Teds are incentivized to provide accurate information (effort) via a reward system based on trust scores. Alice computes a personalized trust belief distribution for all the Charlies and utilizes it to eventually update the trust scores of the selected Teds based on their contribution to Bob's authentication process. An experimental evaluation of the model is carried out using a set of PUF-instantiated FPGAs. Detailed numerical results are presented demonstrating the effectiveness of the proposed scheme and its ability to correctly decide if Bob is trustworthy based on incomplete information provided by peers.**

***Index Terms*—Peer Trust, Contract Theory, Reinforcement Learning, Physical Unclonable Functions, Internet of Things.**

I. Introduction

The convergence of ubiquitous networking, cloud computing, and embedded intelligence has led to the rise of edge computing and the Internet of Things (IoT). Applications for IoT range from home to industrial automation, from local sensor networks to vehicle-to-vehicle (V2V) communications. New challenges emerge as networks evolve from local, constrained environments to large, heterogeneous ecosystems, where the cardinality and capability of individual nodes are always in flux. Maximum utility is derived if the ecosystem can share sensitive information, such as healthcare or payments data.

To achieve this, nodes must use a trust framework to authenticate each other and rely on secure communication channels to transfer information. Centralized approaches, such as standards and certificates, solve the problem but require an established infrastructure to function effectively. Local methods of deriving trust between devices are needed for large, heterogeneous networks where nodes vary greatly in compute power, communications protocols and standards compliance, as certain nodes may not be registered or may be incapable of participating due to sporadic connectivity or insufficient compute power.

Distributed approaches can be effective at establishing trust, the most promising of which is the crowd-sourced trust model, where trust between two IoT devices is derived from transient neighbouring nodes available at that point in time, despite their hardware configuration, compute power and protocol support. Robustness of the method is enhanced if differences in configuration and capability contribute to the derivation of the final trust score, used as a proxy for the level of trust. Additionally, devices should be able to ascertain if the participant is honest or malicious, as an adversary could introduce malicious nodes masquerading as honest ones. Crowd-sourced approaches can be augmented by fusing the protocol with a local, hardware-based root of trust, such as a physical unclonable function (PUF). Equivalent schemes can be constructed from asymmetric encryption algorithms such as Diffie-Hellman and elliptic curve cryptography, but they are not lightweight and necessitate the use of secure hardware to store the key securely at rest. The composition of two distinct controls enables defense in depth, significantly improving the security posture in scenarios where connection to a remote trusted authority may be unavailable.

A. Contributions

Presented in this work is a contract-theoretic mechanism based on incentives for building trust between two devices (termed Alice and Bob) operating in a large, heterogeneous network of IoT nodes. These nodes transact frequently amongst each other using local connectivity, as access to a trust authority is assumed to be sporadic and unpredictable.

The approach is distinct from traditional blockchain solutions in that the final outcome (consensus) is local to Alice and Bob and not a common state shared with all nodes in the system. Furthermore, by relying on simple metrics that all nodes possess, the model establishes independence from the underlying hardware and communications architecture, and is therefore compatible with a wide variety of applications. Crucially, the approach is compatible across network boundaries and can bind cross-network devices in a common trust framework, a property not commonly seen in other approaches. Highlights of the scheme are as follows:

- The paper introduces a peer-to-peer authentication protocol designed to facilitate the secure information transactions between two parties: Alice and Bob. The system relies on a set of local devices, referred to as Charlies, which act as intermediates to authenticate Alice and Bob. The trust between these peers is established through a unique method that uses authentication tokens issued by a trusted authority.
- The proposed protocol introduces a novel offline contract-theoretic (CT) mechanism for selecting the most trusted peers among the set of nodes. The model derives trust in an ad-hoc fashion by crowd-sourcing locally adjacent nodes while staying robust against malicious behavior.
- The CT protocol is fused with an authentication protocol using strong PUFs implementing low-level authentication functions. In particular, the PUF is the hardware authentication source (a local root of trust), while the CT mechanism authenticates based on past behaviour. This two-factor approach significantly improves the resilience of the protocol against adversarial attacks.
- The protocol was implemented and experimentally validated using representative hardware. Results suggest that the protocol is effective at maximizing social welfare while penalizing malicious nodes. While tested with a specific PUF, it is compatible with any strong PUF.

B. Related Work

Distributed and decentralized trust models are an emerging topic of interest in IoT environments, especially in *industrial IoT systems* [1]. A trust-based certificate management framework for industrial IoT networks is introduced in [2] using a clustering architecture, signaling game theory for certificate revocation, and an efficient short-lived certificate verification scheme. A similar dynamic trust management model is proposed in [3] to enhance the security, adaptiveness, and resiliency in managing industrial data. A trust-preserving mechanism for mobile crowd-sensing that uses probabilistic trust assessment is analyzed in [4] to enhance the blockchain transactions by filtering untrusted nodes and ensuring the system's reliability. A mobile edge computing-based trust evaluation scheme is presented in [5] based on a probabilistic graphical model to assess and manage the sensor nodes' trustworthiness and enhance the reliability in smart industrial IoT systems. The authors in [6] follow a game-theoretic approach to incentivize facility nodes and ensure trustworthy service provisioning in dynamic mobile IoT systems via accounting for varying trust scores and service charges. Toward jointly optimizing the resource allocation, network security, and device collaboration in IoT systems, the authors in [7] propose a reputation-based evaluation mechanism combined with the blockchain technology that deals with the information asymmetry among the IoT devices. Toward ensuring trust and fair reward distribution among edge servers, the authors in [8] present a decentralized edge computing platform that integrates blockchain technology and enhances the IoT system performance. A semi-centralized trust management system for IoT data exchange is proposed in [9] by leveraging the blockchain technology and a novel rotation-based consensus protocol to enhance the reliability and address the challenges posed by malicious devices.

Several other fields, such as crowdsourcing [10], internet of vehicles [11], aerial communications among unmanned aerial vehicles [12], sensor networks [13], have established trust models to ensure reliable information exchange among untrusted entities. A mutual authentication protocol for IoT edge systems using Physical Unclonable Functions (PUFs) is designed in [14] to ensure the robust security against various cyber threats while minimizing the computational and hardware overhead. Similarly, a noise-tolerant authentication protocol based on the PUF Phenotype concept is designed in [15] to enhance the security and performance in IoT networks by leveraging machine learning techniques for authentication rather than traditional error correction methods. A hybrid consensus mechanism combining Proof-of-Stake and Practical Byzantine Fault Tolerance is analyzed in [16] to enhance the performance and resilience against dishonest nodes by incorporating trust scores and rewards. A database of 10,000 vulnerable finite state machine designs is populated in [17] to enhance the trust in hardware security by efficiently generating and detecting vulnerabilities through large language models. A decentralized trust management system based on blockchain technology is analyzed in [18] for intelligent transportation systems by using a transparent, consensus-based evaluation model and trusted execution environments.

Despite the research efforts identified in the literature, existing trust models in IoT often rely on centralized systems or blockchain and consensus mechanisms. As such, they face challenges related to computational demands, network diversity, and scalability. Our proposed research work addresses these gaps by introducing a contract-theoretic trust model that uses Stochastic Learning Automata and Bayesian methods for decentralized, cross-network trust management. This approach enhances trust-building between IoT devices while minimizes the reliance on centralized infrastructure and accommodates diverse network environments. Specifically, our approach is tailored to scenarios where devices spend a considerable amount of time in local communications with each other, with infrequent connections to a remote service.

The rest of the paper is organized as follows. Sections II-B and II-C introduce the device level authentication model with physical unclonable functions (PUFs) as the root of trust. The contract-theoretic model is outlined in Section III. Section IV describes the experiments and includes a sensitivity analysis, Section V presents a security analysis, followed by the conclusion in Section VI.

II. Peer Trust Protocol

A. High Level Overview

The protocol is divided into an initial provisioning phase executed once and a repeatable in-field authentication phase where two devices (Alice and Bob) authenticate each other by sourcing information from their neighbours. During provisioning, devices are enrolled by the Issuing Authority (IA) in a trusted secure environment. A PeerTrust$_{DB}$ is created per device to bootstrap the initial authentications. This database is updated during successive in-field authentications, which has

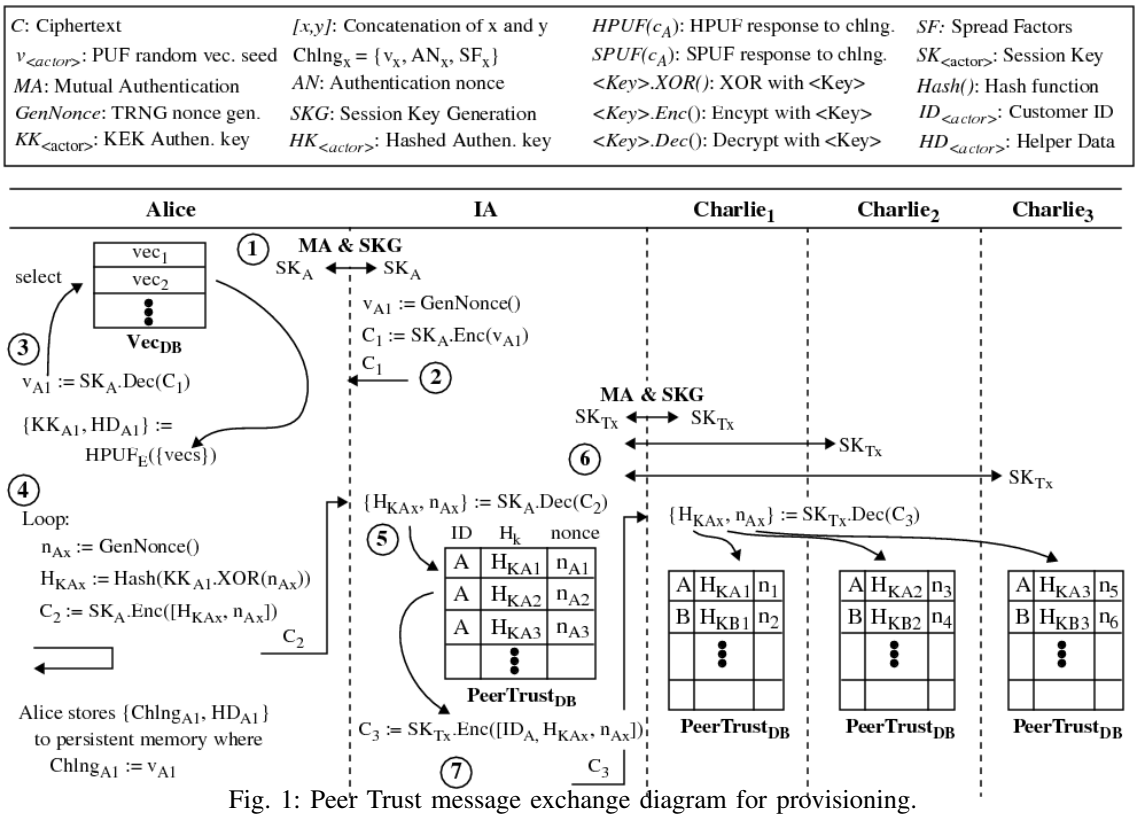

Fig. 1: Peer Trust message exchange diagram for provisioning.

three major stages: Available Charlies are down-selected using a combination of PUF authentication and Contract Theory, culminating in an optimal set of high-trust Teds that possess authentication information on Alice and Bob. This is followed by a mutual PUF-based authentication attested to by the selected Teds. The protocol culminates in Teds generating session key shards for Alice and Bob's shared session key.

B. Provisioning

The Peer Trust enrollment process is illustrated as a message exchange diagram in Fig. 1. Provisioning requires each of the devices to generate authentication tokens (AT) and securely transmit them to the Issuing Authority (IA). During provisioning, the customers, Alice, Bob, etc. obtain a set of challenges from the IA. These challenges are used to generate a master key KK_A, which provides the entropy (randomness), along with a nonce n_x, for the authentication tokens (AT) used in the interactive portion of the Peer Trust protocol. Devices may return for re-provisioning once ATs are exhausted.

The annotations **MA** and **SKG** denote mutual authentication and session key generation respectively. A third *PUF-based security function* (PUF-SF) is used to generated LLK bitstrings, which embed PUF-based security properties into a set of lightweight tokens (ATs) used in the Peer Trust protocol.

The following describes the sequence of steps that customer devices follow to generate the ATs, deliver them to the IA and then have IA distribute unique sets to other customers.

1) A customer, Alice, Bob, Charlie, etc. (labeled Alice in the figure) mutually authenticates and generates a unique shared session key SK_A, with the Issuing Authority (IA) using the PUF-SF functions.
2) IA generates a challenge seed v_{A1}, encrypts it with SK_A and transmits the packet C_1 to Alice.
3) Alice uses the seed v_{A1} to extract a set of vectors $vecs$ from the Vec_{DB}, which stores a shared master set of challenges that she received during provisioning. She applies the vectors to her hardware PUF, $HPUF_E$ in enrollment mode to generate a long-lived key, KK_{A1} and helper data HD_{A1}.
4) Alice generates a sequence of authentication tokens (AT) by calling the TRNG associated with her PUF. Each call generates a nonce n_{Ax} that she XORs with KK_{A1}, and then hashes to produce H_{KAx}, which refers to the keyed-hash output. H_{KAx} along with the nonce n_{Ax} are encrypted with SK_A into packet C_2 then sent to IA.
5) IA decrypts the packet and stores Alice's ID A, H_{KAx} and n_{Ax} in the PeerTrust$_{DB}$. Steps 4 and 5 are repeated n times enabling IA to store a sequence of authentication tokens for Alice.
6) Customer devices, referred to as Charlies, request sets of authentication tokens from IA. They each mutually and privately authenticate and generate shared session keys, SK_{Tx}.
7) IA reads a set of unique ATs from the PeerTrust$_{DB}$, one AT for each of the provisioned devices. It then encrypts the token along with the device ID as C_3 and transmits the packet to Charlie$_x$. Although not shown, the set of ATs are deleted from the IA database to ensure that they are not reused for other devices. Charlie$_x$ decrypts the packet and stores the AT information in its PeerTrust$_{DB}$.

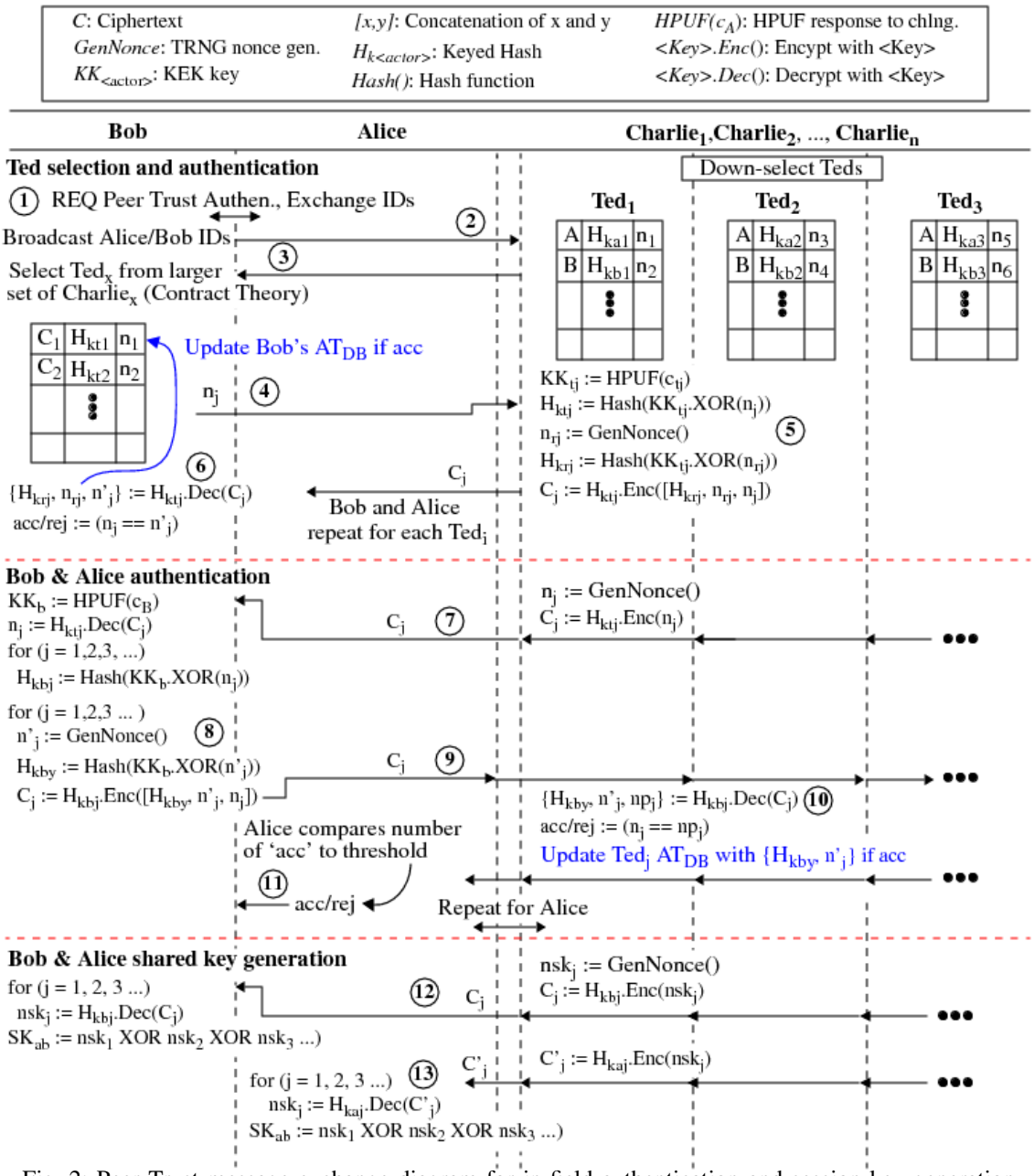

Fig. 2: Peer Trust message exchange diagram for in-field authentication and session key generation.

C. In-Field Authentication and Session Key Generation

The message exchange sequence for the In-Field version of the Peer Trust Protocol is given in Fig. 2. The goal of the protocol is to enable Bob to select a set of trusted peers (Ted_j) who will serve as the authenticators for Alice and Bob, i.e., to provide corroborative evidence to Alice and Bob that they are who they say they are. Alice (the payer) starts the Peer Trust In-Field exchange by contacting Bob to engage in a transaction. For example, Alice may wish to make an eCash payment to Bob for a product or service. Bob, as the recipient of the payment must select a set of trusted Ted_j from surrounding nodes to authenticate Alice. To do so, he broadcasts a message requesting PeerTrust to local neighbours along with his and Alice's ID. A set of locally connected peers, referred to as Charlie_j, respond if they possess ATs for both Alice and Bob. Bob then runs a contract-theoretic algorithm (CT) to down-select to a set of trusted peers, referred to as Ted_j. The following provides details regarding the authentication component of the protocol. A description of the CT algorithm is presented later in Section III. Note that each pairwise communications link (Alice-Bob, Alice-Ted_j, Bob-Ted_j) is a distinct channel and all packet encryption operations use AES-256 in CBC mode.

1) Alice contacts Bob to pay for a product or service. Alice and Bob exchange IDs.
2) Bob broadcasts to the community of locally connected devices, Charlie_j a message containing Alice and Bob's IDs. A set of Charlie_j devices that possess ATs for both Alice and Bob acknowledge that they can assist Bob.
3) Bob consults his PeerTrust_{DB} and selects a subset of Charlie_j for which he has ATs. Bob then runs a contract-theoretic Peer Trust algorithm to down-select further

to a set of the most-trusted Ted$_j$ from the Charlie$_j$ devices (Section III). Bob communicates with Alice to determine if she has AT for each of Bob's selected Ted$_j$. Authentication fails if either Peer Trust fails to find a sufficient number of trustworthy Teds or if Alice does not possess ATs for the selected Teds.

4) Then, Bob extracts the n_j nonces for each selected Ted$_j$ and transmits the matching nonce to each Ted$_j$.
5) Each Ted$_j$ hashes the received n_j with their PUF-generated LLK, KK_{tj} to produce a response H_{ktj}, and then proceeds to generates a new AT for the next authentication operation. Each Ted$_j$ generates a new nonce n_{rj} using a PUF-based TRNG and then hashes it with KK_{tj} to generate H_{krj}. The original nonce and new AT are encrypted with H_{ktj} and transmitted to Bob.
6) Bob receives each of the C_j encrypted packets from the Ted$_j$, decrypts them using his database-stored H_{ktj} and then compares the original n_j with the decrypted version n'_j. For every match, the corresponding Ted$_j$ is authenticated and a secure channel established between Ted$_j$ and Bob. Alice and Bob repeat Steps 4 through 6 for all Teds selected by Bob.
7) **Bob & Alice authentication:** Once all Ted$_j$ authentications succeed, Alice and Bob may proceed to authenticate each other using the ATs supplied by each Ted$_j$. Each Ted$_j$ generates a nonce, n_j, and transmits it to Bob over the secure channel. Bob uses his stored challenges c_B to generate his LLK KK_b, and decrypts the nonces received from the Teds. He constructs a shared session key, H_{kbj} for each Ted$_j$.
8) To refresh ATs for the next authentication, Bob generates a new nonce, n'_j, XORs it with his PUF response and hashes the result to generate a new session key, H_{kby}.
9) Bob then constructs a packet, per Ted$_j$, containing the old nonce, n_j, the new nonce, n'_j, the new session key H_{kby}, encrypts it with H_{kbj} packet and transmits the resulting C_js to each Ted$_j$.
10) Each Ted$_j$ receives the packet, decrypts it using H_{kbj}, compares the decrypted np_j with the previously sent n_j. A successful decryption and comparison denotes a successful authentication, as only Bob's PUF could have produced the unique H_{kbj} in Ted$_j$'s database. Once authentication succeeds, each Ted$_j$ updates Bob's entry in its database with the new AT tuple H_{kby} and n'_j. Each Ted informs Alice of Bob's authentication status.
11) Alice compares the number of successful authentications against a threshold. Bob is authenticated if the number of successful Teds is above the threshold. Steps 7 through 11 are repeated to authenticate Alice. The protocol aborts if either authentication attempt fails.
12) **Shared Key Generation:** Once authentication is successful, each Ted$_j$ generates a new nonce, nsk_j, encrypts them with Alice and Bob's AT-generated session keys, H_{kaj} and H_{kbj}, and transmits the encrypted packets C_j and C'_j to Bob respectively. Bob decrypts the nsk nonces and XORs them together to derive the session key SK_{ab}.
13) Alice does the same to obtain the identical session key SK_{ab}. Alice and Bob are now authenticated and can now communicate over a secure channel.

III. Peer Authentication with Contract Theory

Peer trust plays a critical role in offline ad-hoc networks, where a central authority for customer identification, authentication, and confidential transactions may be absent. Trust is established by crowd-sourcing peer information, supplemented with updated information from their last online interaction with the central authority, i.e. the IA. The proposed model uses an effort-reward-based contract-theoretic model to incentivize peers, denoted as Ted$_j$, to report accurate data. In offline value transfer transactions, such as electronic money transactions, where Alice transfers funds to Bob, Bob relies on peer trust information to verify Alice's trustworthiness. To build this trust model, Bob engages with a set of peers and performs the following actions:

1) Bob broadcasts a request to nearby peers, Charlie$_k$, within the offline local network, asking if they possess authentication tokens (ATs) for Alice and Bob. He collects the network IPs of responders.
2) Bob measures the transmission delay ($distance_k$) and workload ($compute\_congestion_k$) for each Charlie$_k$ using a sequence of trial AES encryptions. He combines this data with their trust levels (TLC) as provided by the central authority during Bob's last online session.
3) Bob selects a subset of trustworthy Charlie$_j$, called Ted$_j$ to assist in the authentication process. The subset size NT depends on the transaction's value, with higher-value transactions requiring more Ted$_j$.
4) Using a stochastic learning automata (SLA) algorithm, Bob evaluates possible subsets of Ted$_j$ based on trust metrics ($\mathbf{PF}$) that integrate current beliefs and performance data, as defined by Eq. 2.
5) Bob applies a Bayesian trust model to refine his trust beliefs about Ted$_j$ based on historical transaction data, providing probabilistic trust estimates in incomplete information scenarios.
6) Bob optimizes a reward contract for Ted$_j$ by pairing their perceived effort with corresponding rewards. The rewards adjust Bob's historical trust records for Ted$_j$ to incentivize their continued participation.
7) If Bob collects enough ATs to meet the authentication threshold, Alice is authenticated, and the session key generation begins. Otherwise, the protocol is aborted.
8) Bob updates transaction records (TR) in the PeerTrust$_{DB}$, capturing trust scores and beliefs for each Ted$_j$. These records are periodically synced with the IA when Bob regains internet access, enabling centralized updates to the PeerTrust$_{DB}$.

The ground truths $belief_0$ for trust levels in the algorithm are derived from Bob's interactions with the IA. Bob provides transaction records reflecting whether the Teds he interacted with delivered valid AT during authentication. The IA aggregates Bob's data with input from other devices, leveraging its global perspective to collect trust data from all customer devices. After computing the updated ground truths based on these interactions, the IA supplies them to Bob, enabling their use in his subsequent offline transactions. In some scenarios,

the IA can also be directly involved with updates to customer ground truths. For example, in an electronic cash scenario, Alice's trust score could be updated positively or negatively based on whether the electronic tokens that Bob receives are authentic or counterfeit.

Initially, all entities in the ad-hoc network possess the same trust belief distribution $belief_0$, which represents the contribution that each entity can provide to assist with the establishment of secure transactions between pairs of entities in the network. Each entity can potentially provide a satisfactory (S_j^t) or unsatisfactory (F_j^t) contribution to the crowdsourcing process with a probability P_s and P_f, respectively. Bob's posterior belief based on the Ted$_j$ provided contribution in the t^{th} transaction is derived from the Bayesian trust belief:

$$belief_{combo[\kappa][j]}^t = \frac{belief_0 P_s^{S_j^t}(1-P_s)^{F_j^t}}{belief_0 P_s^{S_j^t}(1-P_s)^{F_j^t} + (1-belief_0)P_f^{S_j^t}(1-P_f)^{F_j^t}} \quad (1)$$

The SLA algorithm is charged with finding a subset of Charlie$_k$ that achieves a near optimal *personalized feedback* or PF level. To accomplish this, it first constructs an array of all possible combinations of Ted$_j$ subsets of Charlie$_k$, i.e., $NCC = \binom{k}{j} = \frac{k!}{j!(k-j)!}$, and then randomly explores the combinations in this space, computing a PF for each combination using Eq. 2. Here, NT represents the number of Ted$_j$ that Bob wishes to select, NC is the total number of Charlie$_k$ that possess ATs for both Alice and Bob, and $belief_{combo[\kappa][j]}$, $TLC_{combo[\kappa][j]}$, $compute\_congestion_{combo[\kappa][j]}$, $distance_{combo[\kappa][j]}$ are Bob's belief, the Charlie$_k$ trust information as it was received from the central authority the last time that Bob was online, and the current workload on the device k and *norm_distance* for a specific combinational subset κ of Charlie$_k$. The parameter *norm_distance* is defined as the measured $distance_k$ divided by the sum of the $distance_k$ for all Charlie$_k$. The SLA is configured with a learning rate parameter that controls how much of the search space is explored and evaluated before making a selection (details are provided in [19]).

$$\mathbf{PF} = \sum_{j=1}^{NT}\left(\frac{\frac{belief_{combo[\kappa][j]} \times TLC_{combo[\kappa][j]}}{compute\_congestion_{combo[\kappa][j]} \times distance_{combo[\kappa][j]}}}{\sum_{k=0}^{NC} \frac{belief_k \times TLC_k}{compute\_congestion_k \times norm\_distance_k}}\right) \quad (2)$$

After Bob selects the set of Teds who will support Bob's decision in terms of trusting (or not) Alice, Bob considers the $belief_{combo[\kappa][j]}^t$ for the selected Ted and assigns a type $type_j$ to the Teds that supported Bob's crowdsourcing process based on the normalized $belief$. Based on the assigned $type_j$ and the effort provided by each Ted, Bob executes a contract-theoretic model (details are provided in [20]) to determine the optimal $reward_j$ provided to each Ted. Considering the $reward_j$ and the $belief$ that Ted$_j$ had about Bob as received from the central authority the last time Bob was online, Bob updates offline its trust level $TLC_{combo[\kappa][j]}^t$ to Ted$_j$, as $TLC_{combo[\kappa][j]}^t = TLC_{combo[\kappa][j]}^{old} + \alpha[reward_j \cdot belief_{combo[j][\kappa]}]$, where $\alpha \in \mathbb{R}^+$ is a tuning parameter allowing Bob to weigh more (or less) the trust level stemming from the central authority or from the offline transactions with Ted$_j$. Then, Bob utilizes the updated $TLC_{combo[\kappa][j]}$ values to reselect Ted$_j$ among the Charlie$_j$ in a future transaction, while still remaining offline.

The nodes periodically reconnect to the IA to submit histories, refresh authentication tokens, reward honest Teds assisting Alice and Bob, penalize malicious ones, and aggregate beliefs across nodes and transactions to form a global system trust view. In a peer-based model, any given node can alternate roles across transactions: as Alice when initiating, Bob when responding, or Ted when assisting. The IA aggregates a belief for each peer j using the local beliefs reported by the other nodes based on transactions where j acted as a helper. Eq. 3 defines the incremental belief update for peer j at time t and includes the success or failure of new transactions reported by a reconnecting peer, Bob. The belief at $t-1$ represents the aggregation up to that point, while t reflects updates based on the current transaction outcome.

$$belief_j^t = \frac{belief_j^{t-1} \cdot P_s^{S_j^t}(1-P_s)^{F_j^t}}{belief_j^{t-1} P_s^{S_j^t}(1-P_s)^{F_j^t} + (1-belief_j^{t-1})P_f^{S_j^t}(1-P_f)^{F_j^t}} \quad (3)$$

The trustworthiness is updated incrementally, as per Eq. 4, where r_j^t is the reward and $\alpha \in \mathbb{R}^+$ is a weight factor.

$$T_j^t = T_j^{t-1} + \alpha(r_j^t \cdot belief_j^t) \quad (4)$$

The trust scores for each peer are derived by normalizing their trustworthiness values across all peers using Eq. 4, where N denotes the total number of nodes in the system.

$$p_j^t = \frac{T_j^t}{\sum_{j=0}^{N} T_j^t} \quad (5)$$

Multiple transactions across nodes can be batched within a single update interval. While global ordering is not required, transactions from a single node must be reported in execution order. Once new transactions are integrated, each node j's latest $belief_j^t$ becomes its new $belief_0$. As nodes connect and report asynchronously, their ground-truth $belief_0$ updates, propagating the global belief across the network. However, the aggregated belief lags behind the ground truth, with the delay depending on reconnection frequency and transaction volume. Thus, the IA's belief is a delayed approximation of the system's true state.

IV. Experimental Results

A. Experiment Setup

In this section, we report on characteristics of a hardware implementation of the Peer Trust protocol. The experimental test bed in shown in Fig. 3 and consists of a set of internet-connected ZYBO-Z7-10 SoC FPGAs and a Dell PowerEdge T440 server. The customer devices are instantiated with a strong PUF called the Shift-Register Reconvergent-Fanout (SiRF) PUF [21]. The Issuing Authority (IA) represents the central authority and maintains the non-anonymous timing (NATm) and anonymous timing (ATm) sqlite3 databases (DB). These databases are used for PUF-based mutual authentication

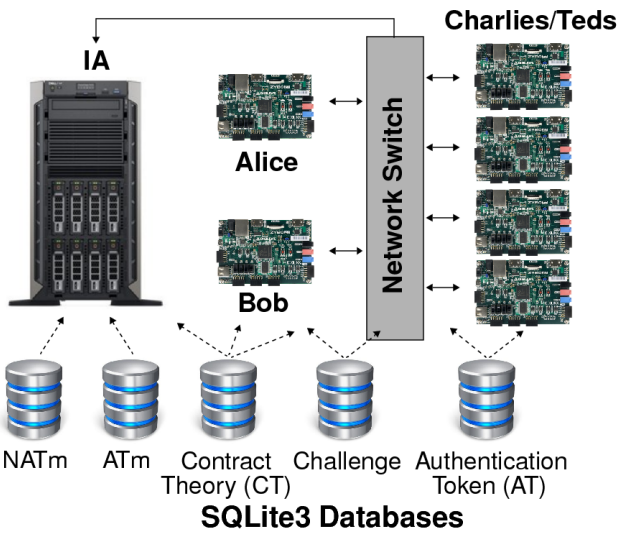

Fig. 3: Experiment setup with IA, Alice, Bob and a set of Charlies/Teds.

and encryption operations and are pre-loaded with provisioning data from 16 Zybo-Z7 devices. The IA runs a multi-threaded application to provide AT and CT-related services to the customer devices, Alice, Bob and the Charlies. The customer FPGA devices run a single-threaded Peer Trust application. The IA and customer devices are connected through a wired network and a 100 Mb network switch. The following SLA and CT parameters are used in the hardware evaluation of the Peer Trust protocol, $\mu_0 = 0.2$, $a_h = 0.51$, $a_l = 0.49$, $S_x = 0$, $F_x = 0$, $\lambda = 0.7$, $\alpha = 0.01$, and $LR = 0.05$, unless explicitly stated otherwise.

The experiment setup configures Alice and Bob to down-select two Ted devices from a set of four Charlie devices using SLA, and once selected, the two Teds participate in the Peer Trust offline authentication protocol between Alice and Bob. Note that the model allows the number of required Teds to be specified by Bob, and the number of valid Charlies can vary from this lower bound upward to any number. The experimental results presented in this paper utilize the configuration shown in Fig. 3 but the results for larger numbers of devices are very similar to the results presented here. The experiments carried out evaluate several features of the Peer Trust protocol, focusing on the behavior of the CT beliefs, trustworthiness and trust scores recorded by Bob and those associated with the periodic updates to the IA's ground truths.

Different test scenarios are shown in Fig. 4. All test scenarios are run for 100 iterations, where one iteration is defined as a complete execution of the Peer Trust (PT) authentication protocol as described in Fig. 2. Also, in all test scenarios, Alice begins the PT authentication operation by contacting Bob, which is followed by Alice and Bob exchanging IDs. Bob then attempts to find a set of eligible Charlies by broadcasting a request for assistance to all devices on the local area network. Charlies respond to the request only if they possess ATs for both Alice and Bob. The authentication token (AT) enrollment process carried out at the beginning of each run guarantees that the four Charlies have ATs for both Alice and Bob, enabling all four to participate in all of the authentication transactions.

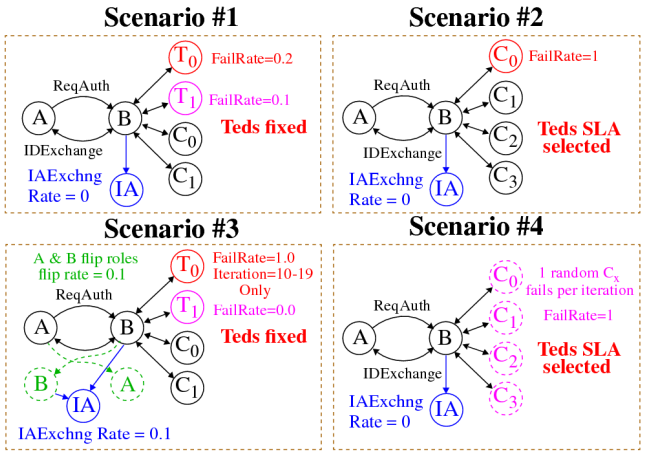

Fig. 4: Contract theory testing scenarios.

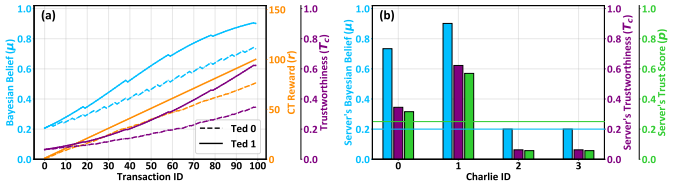

Fig. 5: Test Scenario 1.

Bob runs the CT algorithm to select a set of Teds. In order to select the Teds using CT, Bob needs to obtain two distinct sets of information from the Charlies. First, he needs an estimate of compute congestion and network delay. For compute congestion, Bob requests that each Charlie measures and transfers the time taken to carry out a set of trial encryptions using `gettimeofday()` system call. Bob obtains an estimate of network delay by timing the round trip delay of a dummy message transferred between him and each Charlie.

Bob reads the next set of information from his PeerTrust$_{DB}$ database, where he extracts the current state of the following CT parameters: belief, trustworthiness and trust scores, for each of the Charlies. He also extracts the total number of AT authentication successes and failures, where an AT authentication failure is defined as a failed attempt of Bob to authenticate a given Charlie using the PT in-field authentication process described earlier. An AT authentication can fail, for example, if one of the Teds fails to provide a valid AT during the pairwise authentication that occurs between Bob and each of the Teds after the CT process completes. The state of the success or failure field in the CT database is incremented for each Charlie depending on the outcome of the pairwise authentications. An authentication failure is emulated in our experiments in a controlled (forced) fashion to enable the behavior of the CT algorithm to be evaluated under different fail case scenarios, corresponding, in part, to the test scenarios described below.

The parameter *IAExchange Rate* (Fig. 4) refers to an exchange of information between Bob and the IA, which occurs in actual deployments when Bob is able to reconnect to the internet. An *IAExchange Rate* of 0 indicates that Bob exchanges CT record updates with IA only after all 100 PT iterations in the experiment, otherwise, it indicates periodic exchanges occur at a rate equal to the value of the parameter times the total number of transactions, e.g., 0.1 indicates an exchange every 10 transactions in our experiments.

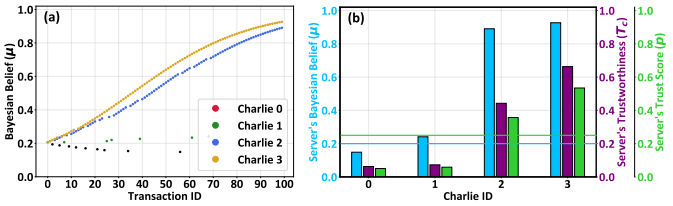

Fig. 6: Test Scenario 2.

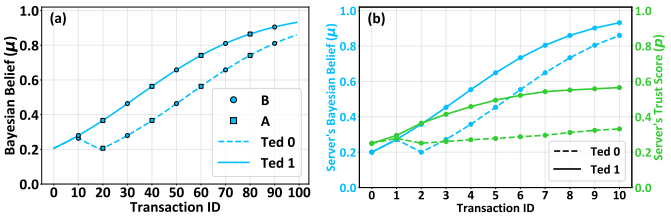

Fig. 7: Test Scenario 3.

B. Test Scenario Results

In test scenario #1, the goal is to evaluate the behavior of the CT beliefs and trust scores on Bob's device, in test cases where specific Teds experience different AT failure rates. The output of the SLA selection function is ignored and the two Teds are fixed to specific Charlies, as shown by the legend in Fig. 5a. This is equivalent to an offline scenario in which only two Charlies have ATs for Alice and Bob. The failure rates of Ted_0 and Ted_1 are set to 0.2 (one fail in every five transactions) and to 0.1 (one fail in every ten), respectively. Given that Ted_1 experiences a lower fail rate, Bob's belief should increase faster for Ted_1 than his belief for Ted_0. The beliefs computed by Bob for the two Teds reflect the expected behavior (Fig. 5a). Specifically, the belief of Bob for Ted_1 increases faster compared to Ted_0, given the lower fail rate, resulting in a higher experienced cumulative reward and trustworthiness. The IA's belief, trustworthiness, and trust scores also reflect lower values for Ted_0 compared to Ted_1 in the final computed ground truths, with the initial values annotated with horizontal lines in Fig. 5b. Also, it can be observed that the trustworthiness and corresponding trust scores of the unselected Charlies decreased due to their lack of involvement in the authentication operations.

Test scenario #2 models the behavior of a malicious Charlie, C_0, which experiences constant failures when authenticating with Bob. This test case determines whether the proposed approach is effective in reducing the probability of Bob selecting C_0 as a Ted. The SLA selection process is enabled to choose the most trusted Charlies in order for the latter ones to act as Teds. The results reveal that Bob's belief regarding the malicious Charlie C_0 decreases over time given its failure to provide appropriate services to Bob (Fig. 6a). On the other hand, Bob's belief in C_2 and C_3 increases, as they were selected more frequently due to their consistent success in meeting service expectations.

Interestingly, Bob's belief about C_1 remains low despite the fact that this device succeeds with authentication every time it was selected. The small belief occurs initially because the process implemented for measuring C_1's network and compute congestion is subject to variation caused by interrupt handling within the Linux OS running on the devices. The larger values for network delay and compute congestion increases the value of the denominator of Eq. 2, reducing the likelihood of selection in the SLA algorithm. The CT algorithm design leads to this behavior as once the cumulative beliefs of other devices increases beyond a certain threshold, it is difficult for devices penalized by early unlucky events to recover to a point where they become eligible for selection once again. One approach to avoid this anomalous behavior is to implement performance measurements as slowly changing, running averages, which would desensitize them to stalls caused by interrupt service handling. Although not shown here, we confirmed this approach to be effective for eliminating this type of anomaly.

Focusing on the IA bar graph of Fig. 6b, the belief in C_2 and C_3 increases at the expense of the C_0 and C_1, whose belief values decrease relative to their initial values (represented by the horizontal lines). This shift stems from the failure of C_0 to adequately serve Bob and the inability of C_1 to recover from the low performance measurement. Similarly, the IA's trustworthiness and corresponding trust score for C_2 and C_3 both increase as a result of their successful selection and service to Bob. However, C_3 exhibits higher trustworthiness values compared to C_2, primarily because it was selected more frequently. This increased selection leads to a greater cumulative reward for C_3, thus, enhancing its trustworthiness and, correspondingly, its trust score.

Test scenario #3 models an offline scenario where T_0 is the authentic device when selected by Bob, but is malicious when Alice and Bob's roles are reversed, i.e., when Alice evaluates the trustworthiness of Bob. We force T_0 to fail for a sub-set of transaction iterations, specifically iterations 10 through 19, while assigning success to the remaining authentications. The expectation is that the belief for T_0 will decrease rapidly during the ten failed transactions and then recover over the remaining transactions. This test scenario also sets the IAExchng Rate to 0.1, which indicates Alice or Bob reconnects to the IA after every 10 iterations. The periodic reconnections to IA enables the beliefs, trustworthiness and trust scores of Alice and Bob to be evaluated both locally and globally by the IA.

The CT Beliefs are plotted in Fig. 7a which illustrates that Ted_0's belief drops between iteration 10 and 20, but then start recovering over the remaining transactions to nearly the maximum value of 1.0. The impact of the periodic IA updates are also noticeable after every 10 transactions. Fig. 7b shows the Beliefs and Trust Scores computed by the IA with the points representing the updates after each of the 10 transactions. Although the IA's Beliefs track the device local Beliefs, the Trust Score for T_0 does not recover, which indicates that Trust Score maintains a longer memory of the failed authentications.

Test scenario #4 is constructed to determine whether the CT fairly selects among the Charlies when the fail rates are equal in the environment where AT failures are equally likely over time for all devices. Here, we allow SLA to select two random Teds to authenticate with Bob, but also force two random Charlies to fail on each transaction. Therefore, the two Teds selected have a 50% chance of failing in each transaction. A

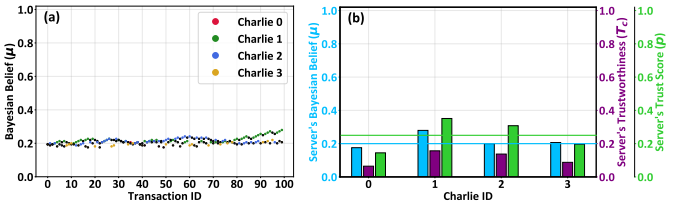

Fig. 8: Test Scenario 4.

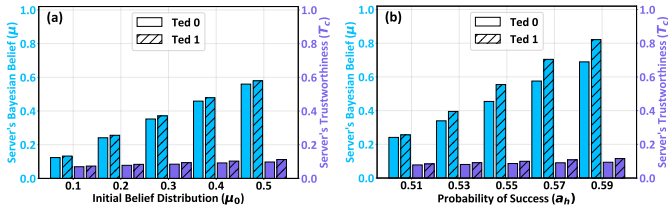

Fig. 9: Sensitivity Analysis.

50% fail rate is extremely high, but is used here to represent the behavior that would normally occur over a much longer time period. Given the probabilities of succeeding and failing are equal, the CT beliefs should remain close to the initial value of 0.2, and therefore, the probability of choosing any particular Ted should be equal. The belief plot shown in Fig. 8a confirms this expectation, with some small excursions occurring at the end of the run. The belief and trust score values computed by the IA (Fig. 8b) reflect the devices' final values, where variations in the heights are due to these small excursions.

C. Resource Utilization and Performance Metrics

The SiRF PUF implementation consumes 5842 LUTs and 4377 FFs, while KK key generation is upper bounded by 1.5 second (additional details can be found in [21]). The execution of the CT algorithm represents the second dominant component of the total runtime. The CT runtime depends on the number of iterations carried out by the SLA component to reach convergence, and the non-linear optimization component of contract theory. The parameters given earlier provide a good trade-off between accuracy and runtime, with the average number of iterations of 400 and an average runtime of 250 milliseconds, which varies over a range from 211 to 542 milliseconds. Note that the runtime includes fetching network-based capability information from the Charlies.

The size of each AT in the PeerTrust$_{DB}$ is 68 bytes, making a database which stores ATs for 1 million customers approximately 68 MB in size, which can be easily accommodated on typical flash memory cards. Communication overhead is summarized as follows in reference to the numbers in Fig. 2. The exchange in steps 2 and 3 are 4-byte IDs, while in step 4, Bob sends a 32-byte n_js to each of the Teds, and the Teds respond with 96-byte encrypted packets. Step 7 involves the Teds sending a 32-byte encrypted nonce to Bob while step 9 has Bob send 96-byte packets to the Teds. Finally, the Teds send 32-byte packets to Alice and Bob in steps 12 and 13. Steps 4 through 9 are repeated by Alice, making the total number of bytes exchanged per Ted approximately 328 bytes.

D. Sensitivity Analysis

In this section, we conduct a sensitivity analysis of the proposed framework based on Test Scenario #1 where Ted$_0$ and Ted$_1$ are selected, representing high and low failure rates, respectively, over 10 transactions between Bob and the selected Teds. Fig. 9a – 9b illustrate the impact on IA's belief and trustworthiness values for the two selected Teds under varying initial belief μ_0 distributions and probabilities of success a_h for their service to Bob.

The results reveal that as Bob's initial belief μ_0 in the selected Teds increases, the final belief and trustworthiness derived by the IA also increase. A steeper increase is observed for Ted$_1$, which has a lower failure rate in terms of successfully serving Bob. This confirms the proper functioning of the proposed framework, as Ted$_1$'s favorable performance leads to a higher belief and trustworthiness developed by the IA. The initial belief serves as an offset, influencing both the final belief and trustworthiness.

Similarly, as Bob's initial probability of success a_h regarding the services provided by the selected Teds increases, the belief and trustworthiness generated by the IA also increase, with a steeper trend for Ted$_1$ which reflects its positive behavior. Additionally, the analysis shows that the probability of success plays a more dominant role in shaping the IA's belief and trustworthiness compared to the initial belief, however, with a smaller relative increment for trustworthiness. The initial belief functions more as an offset, while the probability of success significantly affects the belief update process and ultimately acts as a primary factor in the final evaluation of the belief and trustworthiness.

V. Security Analysis

The protocol as described previously derives trust through the foundational building blocks of multi-party communication between secure hardware devices provisioned at a trusted Issuing Authority (IA). The physical PUF device and the low-level authentication protocols used during enrollment were studied exhaustively in [21] and [22]. In this work, we focus on the security aspects of the in-field authentication protocol. In all cases, the goal of the attack would be for an adversary to have an honest Bob authenticate a dishonest Alice, where Alice is either acting alone, in concert with one or more dishonest Charlies, or a Mallory. In this analysis, we consider both passive and active attacks.

A. PUF primitive

The protocol is instantiated using low-level primitives [22] sourced from a strong PUF with high information complexity [22] and TRNG capabilities. The prior work demonstrates machine-learning resistance of the PUF. The PUF primitive is critical to the low-level authentication and encryption functions and therefore acts as a local root of trust for distributed interactions. Additionally, except for the first two packets, all packets in the protocol are encrypted using ephemeral, one-time-use keys, ensuring confidentiality and integrity. The first packet contains REQ and ID information, while the second is a plaintext nonce, n_j, sourced from Bob's AT database and unique to each Ted. This nonce is pairwise-bounded between Bob and Ted$_j$, i.e. upon receiving this nonce, Ted$_j$ must use its PUF to regenerate its LLK to provide the matching

H_{ktj} expected by Bob. If H_{ktj} is not what Bob expects, the decryption will fail. Likewise if n_j does not match, Ted$_j$ will fail authentication.

B. Eavesdropping

As noted earlier, each pairwise communications link (Alice-Bob, Alice-Ted$_j$, Bob-Ted$_j$) is a distinct channel and all packets are encrypted using AES-256 encryption in CBC mode. While an eavesdropping observer can capture Alice and Bob's ID from the initial REQ message or the initial nonces, the information does not compromise the protocol in the later steps.

C. Replay and Fraud

As noted earlier, initial values for all devices are provisioned in a trusted environment in a secure facility. Crucially, all ATs (Alice, Bob and participating Teds) are refreshed during an authentication operation, ergo any single AT is used only once and provisioned tokens are replaced with new versions over time. It can be observed from the exchange that PUF authentication layer is interleaved with the contract theoretic layer. Both layers must verify for authentication to succeed. For PUF authentication, each Charlie must produce a valid authentication token on Alice that Bob can reference in his local database. These tokens, while initially provisioned by the IA, are refreshed in-field by the device itself, attesting to the distributed nature of authentication. It is impossible for a Charlie to produce a token on demand that was not previously generated by Bob using his LLK. Any attempts to impersonate or replay a token would result in authentication failure, causing the protocol to abort.

D. Collusion

The Peer Trust model incentivies honest behaviour. The trust scores record behaviour over time, with untrusted Charlies being increasingly rejected from participation in future transactions. However, two or more high trust Teds could also try to collude with each other to extract the shared session key or authenticate a fraudulent Alice, where fraudulent implies that Alice is not provisioned by the IA and may not be a PUF at all. Starting with session key extraction, since the key shards are ephemeral and combined through an XOR operation, they constitute the equivalent of a one-time pad and successful exfiltration would require all Teds to collude with each other. While this is possible, it becomes quickly infeasible as the number of Teds grows large. The precise determination of a suitable threshold of Teds will be studied in future work. The latter scenario is more realistic: A fraudulent Alice impersonating a PUF device could be constructed if an attacker is able to exfiltrate the AT database from a legitimate device using side-channel or NAND mirroring attacks. As countermeasures, the database is encrypted at rest using a PUF-based key-encryption-key (KEK). In KEK mode, the secret key is derived from the response to a challenge stored in non-volatile memory, making it impossible to recover the key without powering on the device and applying the challenge to the PUF. During processing, a trusted execution environment (TEE) can protect individual records against side-channel attacks. A TEE was not used here but is trivial to add to protect software-implemented steps.

E. MITM

In the MITM scenario, an adversary, Mallory, spies on all communications channels, intercepting and manipulating packets as needed. Given that an MITM attack is a superset of eavesdropping, and since MITM can act as the ultimate colluder, the primary attack on the protocol stems from an MITM adversary. In the worst-case scenario, Mallory can be assumed to be working with a fraudulent Alice. If Alice is compromised for some reason, then Mallory, in partnership with Alice, could access a valid database provisioned by the IA. This would allow Mallory to (a) supply ATs authenticating Alice to Bob, (b) use the right session key (HK) to encrypt communications, authenticating itself as a Ted. Mallory can effectively pretend to be any number of Teds (or Charlies) to Bob using this approach. Additionally, if Mallory pretends to be all of the Teds participating in the exchange, then they can learn (or create) all of the shards created in the session key generation stage and eavesdrop on Alice and Bob's transaction (confidentiality leakage).

The Peer Trust layer incentivizes Charlies to be honest. Mallory's behaviour would penalize innocent Charlies who would then fail to be selected in subsequent transactions with Bob. While this might also pose a barrier to Mallory for future transactions, they could simply cycle the Charlies being impersonated such that Bob always sees a fresh set of Charlies. Therefore it is crucial that the database stored on local devices is protected as mentioned earlier. The PUF's hardness provides the main defense against such attacks. The likelihood of Mallory brute-force guessing an ephemeral key in transit is 2^{-128}, therefore considered highly unlikely. Additionally, Bob expects Mallory to send packets encrypted by a session key that is only possible to generate by either successfully model-building the PUF to simulate Teds at the time of transaction or physically attacking in-field devices to exfiltrate the AT database. Thus we show that all attacks revert to circumventing the hardness of modeling the PUF primitive itself.

VI. Conclusion

In this paper, a dual layer trust model is presented such that Alice and Bob can authenticate each other while off-line but in the presence of other devices and construct a secure channel for communications. A dual layer trust model combines PUF-based authentication with an offline contract-theoretic crowdsourcing framework that incentivizes honest behaviour. A PUF-based provisioning and in-field protocol is fused with a Peer Trust model that exploits the concept of Bayesian trust belief and optimal selection of devices via a stochastic learning automata algorithm fusing the two layers with compute/network characteristics. The algorithms were implemented and tested on representative hardware to replicate real-world behaviour. Four distinct scenarios were tested to demonstrate the efficacy and sensitivity of the algorithm under different conditions. Results indicate that the contract-theoretic framework efficiently maximizes the level of crowd-sourced trust, despite distinct patterns of malicious behaviour or altered environmental conditions. Directions for future work may include experiments encapsulating additional scenarios, a detailed sensitivity analysis and a formal security model.

REFERENCES

[1] P. Zeng, A. Liu, C. Zhu, T. Wang, and S. Zhang, "Trust-based multi-agent imitation learning for green edge computing in smart cities," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1635–1648, 2022.

[2] C. Boudagdigue, A. Benslimane, A. Kobbane, and J. Liu, "Trust-based certificate management for industrial iot networks," *IEEE Internet of Things Journal*, vol. 10, no. 14, pp. 12 867–12 885, 2023.

[3] ——, "Trust management in industrial internet of things," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3667–3682, 2020.

[4] L. Zhang, G. Feng, S. Qin, X. Li, Y. Sun, and B. Cao, "Trust-preserving mechanism for blockchain assisted mobile crowdsensing," *IEEE Transactions on Computers*, vol. 72, no. 11, pp. 3113–3126, 2023.

[5] T. Wang, H. Luo, W. Jia, A. Liu, and M. Xie, "Mtes: An intelligent trust evaluation scheme in sensor-cloud-enabled industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2054–2062, 2020.

[6] P. Dass and S. Misra, "Fact: Facilitating trustworthy services in mobile iot systems," *IEEE Systems Journal*, vol. 17, no. 4, pp. 5511–5518, 2023.

[7] Z. Zhou, F. Ye, J. Gao, S. Zhang, and X. Geng, "Ensuring long-term trustworthy collaboration in iot networks using contract theory and reputation mechanism on blockchain," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2420–2437, 2024.

[8] L. Cui, S. Yang, Z. Chen, Y. Pan, Z. Ming, and M. Xu, "A decentralized and trusted edge computing platform for internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3910–3922, 2020.

[9] Y. Liu, C. Zhang, Y. Yan, X. Zhou, Z. Tian, and J. Zhang, "A semi-centralized trust management model based on blockchain for data exchange in iot system," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 858–871, 2023.

[10] S. Fu, X. Huang, L. Liu, and Y. Luo, "Bfcri: A blockchain-based framework for crowdsourcing with reputation and incentive," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 2158–2174, 2023.

[11] P. K. Singh, R. Singh, S. K. Nandi, K. Z. Ghafoor, D. B. Rawat, and S. Nandi, "Blockchain-based adaptive trust management in internet of vehicles using smart contract," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3616–3630, 2021.

[12] M. Huang, A. Liu, N. N. Xiong, and J. Wu, "A uav-assisted ubiquitous trust communication system in 5g and beyond networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3444–3458, 2021.

[13] S. Hriez, S. Almajali, H. Elgala, M. Ayyash, and H. B. Salameh, "A novel trust-aware and energy-aware clustering method that uses stochastic fractal search in iot-enabled wireless sensor networks," *IEEE Systems Journal*, vol. 16, no. 2, pp. 2693–2704, 2022.

[14] S. Abdolinezhad and A. Sikora, "A lightweight mutual authentication protocol based on physical unclonable functions," in *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2022, pp. 161–164.

[15] H. Fei, O. Millwood, P. Gope, J. Miskelly, and B. Sikdar, "Phenoauth: A novel puf-phenotype-based authentication protocol for iot devices," in *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2024, pp. 309–319.

[16] H. Qushtom, J. Mišić, V. B. Mišić, and X. Chang, "A two-stage pbft architecture with trust and reward incentive mechanism," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11 440–11 452, 2023.

[17] D. Saha, K. Yahyaei, S. Kumar Saha, M. Tehranipoor, and F. Farahmandi, "Empowering hardware security with llm: The development of a vulnerable hardware database," in *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2024, pp. 233–243.

[18] X. Chen, J. Ding, and Z. Lu, "A decentralized trust management system for intelligent transportation environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 558–571, 2022.

[19] G. Fragkos, C. Minwalla, E. E. Tsiropoulou, and J. Plusquellic, "Enhancing privacy in puf-cash through multiple trusted third parties and reinforcement learning," *J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 1, sep 2021. [Online]. Available: https://doi.org/10.1145/3441139

[20] G. Fragkos, C. Minwalla, J. Plusquellic, and E. E. Tsiropoulou, "Local trust in internet of things based on contract theory," *Sensors*, vol. 22, no. 6, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/6/2393

[21] J. Plusquellic, "Shift register, reconvergent-fanout (sirf) puf implementation on an fpga," *Cryptography*, vol. 6, no. 4, 2022. [Online]. Available: https://www.mdpi.com/2410-387X/6/4/59

[22] J. Plusquellic, E. E. Tsiropoulou, and C. Minwalla, "Privacy-preserving authentication protocols for iot devices using the sirf puf," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–16, 2023.