Enhancing Privacy in PUF-Cash through Multiple Trusted Third Parties and Reinforcement Learning

GEORGIOS FRAGKOS*, University of New Mexico, United States
CYRUS MINWALLA*, Bank of Canada, Canada
EIRINI ELENI TSIROPOULOU, University of New Mexico, United States
JIM PLUSQUELLIC*, University of New Mexico, United States

Electronic cash (e-Cash) is a digital alternative to physical currency such as coins and bank notes. Suitably constructed, e-Cash has the ability to offer an anonymous offline experience much akin to cash, and in direct contrast to traditional forms of payments such as credit and debit cards. Implementing security and privacy within e-Cash i.e. preserving user anonymity while preventing counterfeiting, fraud and double spending, is a non-trivial challenge. In this paper, we propose major improvements to an e-Cash protocol, termed PUF-Cash, based on physical unclonable functions (PUFs). PUF-Cash was created as an offline-first, secure e-Cash scheme that preserved user anonymity in payments. In addition, PUF-Cash supports remote payments; an improvement over traditional currency. In this work, a novel multi-trusted-third-party exchange scheme is introduced, which is responsible for 'blinding' Alice's e-Cash tokens; a feature at the heart of preserving her anonymity. The exchange operations are governed by machine learning techniques which are uniquely applied to optimize user privacy, while remaining resistant to identity-revealing attacks by adversaries and trusted authorities. Federation of the single TTP into multiple entities distributes the workload, thereby improving performance and resiliency within the e-Cash system architecture. Experimental results indicate that improvements to PUF-Cash enhance user privacy and scalability.

CCS Concepts: • **Security and privacy → Hardware-based security protocols**.

Additional Key Words and Phrases: internet of things, security, electronic money, digital currency, networks

ACM Reference Format:
Georgios Fragkos, Cyrus Minwalla, Eirini Eleni Tsiropoulou, and Jim Plusquellic. 2020. Enhancing Privacy in PUF-Cash through Multiple Trusted Third Parties and Reinforcement Learning. 1, 1 (December 2020), 25 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Electronic cash, or e-Cash for short, is a digital representation of money, where coins and paper are replaced by tokens that act as a medium of exchange in payments. A suitable electronic money scheme must guarantee security and privacy to bolster confidence in individuals. In addition to typical threats of counterfeiting, theft and risk of loss,

*Authors contributed equally to this research.

Authors' addresses: Georgios Fragkos, gfragkos@unm.edu, University of New Mexico, P.O. Box 1212, Albuquerque, New Mexico, 43017-6221, United States; Cyrus Minwalla, cminwalla@bank-banque-canada.ca, Bank of Canada, 234 Wellington St., Ottawa, K1A0G9, Canada; Eirini Eleni Tsiropoulou, eirini@unm.edu, University of New Mexico, P.O. Box 1212, Albuquerque, New Mexico, 43017-6221, United States; Jim Plusquellic, jplusq@unm.edu, University of New Mexico, P.O. Box 1212, Albuquerque, New Mexico, 43017-6221, United States.

e-Cash must also contend with double-spending risks and leakage of private user information such as spending habits. Although privacy increasingly ranks as the number one concern for users, privacy is also the most difficult to guarantee in an electronic money system. Transactions in traditional financial systems, such as credit and debit transactions, incur hops through multiple intermediaries and settlement may be deferred for a certain number of days. Financial incentives encourage mining and analysis of consumer spending patterns for advertising and marketing domains, while extended exposure of customer data to participating intermediaries runs increased risk of information leakage. A privacy preserving e-Cash protocol can enhance consumer privacy by minimizing the role of intermediaries and obfuscating spending patterns between individuals using the system. PUF-Cash exhibits two attributes that are notably distinct from credit and debit payments: First, the protocol is designed to make the spender anonymous, protecting user privacy. Second, the protocol supports peer-to-peer payments.

In addition to these security and privacy concerns, e-cash systems must also contend with the device-level attacks. The devices used by customers to engage in e-Cash transactions, e.g., typically small dedicated devices or cell phones, are mobile and battery-powered, making them vulnerable to probing, fault injection, side-channel and other types of physical attacks. Physical unclonable functions (PUFs) are hardware primitives that are ideal candidates for integration into e-Cash systems. They are robust sources of entropy and possess enhanced resilience to passive physical attacks. PUFs do not store any secrets (keys and authentication bitstrings) in non-volatile memory. PUFs are tamper evident i.e., their source of entropy can be forever altered when probed, effectively destroying their ability to reproduce encryption keys and authentication bitstrings, and resistant to side-channel attacks. Strong PUFs can produce a virtually unlimited number of secrets for authentication and encryption operations, enabling new capabilities such as frequent re-keying and improved resilience against impersonation and machine learning attacks. By contrast, traditional NVM-based random number generation methods are limited to a finite amount of entropy (related to the size of the NVM) and rely on secure hash and other cryptographic mathematical operations for authentication bitstring and key generation.

This work proposes a major advance to an existing e-Cash system, termed PUF-Cash, which utilizes physical unclonable functions (PUFs) as its foundational building block. Significant changes to the protocol are proposed to dramatically improve privacy and system performance. The protocol and system architecture of PUF-Cash directly address the primary challenges associated with security and preservation of privacy, while supporting both online and offline payments. In this work, PUF-Cash is improved by developing a novel message exchange protocol that utilizes multiple trusted third parties (mTTPs) and reinforcement learning (RE-Learn) algorithms designed to optimize performance and customer privacy. A key component to maintaining privacy in PUF-Cash is the token blinding process, where tokens issued to Alice by the Bank during withdrawal are replaced, or 'blinded', by equivalent but anonymous tokens through a trusted exchange operation conducted by a TTP. Upon spending and redemption, the blinded tokens can be validated but are impossible to trace back to Alice. The updated protocol defines a communication architecture in which multiple TTPs cooperate to obfuscate the relationship between unblinded and blinded e-Cash tokens. The cooperating TTPs run machine learning algorithms to make autonomous decisions related to the distribution of unblinded tokens to other TTPs to optimize transactional throughput and maximize similarity between user behaviour. Having a pool of TTPs where the selection criteria is driven by performance is a more federated and user-centric approach to the typical fixed roles available in existing payment mechanisms. Entities operating TTPs could range from commercial banks, financial technology companies, large technology companies, new startups, among others. The system is permissive by design to accommodate to a wide variety of arrangements. PUF-Cash is extensively vetted using a wide range of parameters on a hardware system prototype, built using cheap, commercial FPGA boards, a wireless network and a compute server, i.e., a test bed that accurately models the components and system architecture of a full

blown commercial implementation of an e-Cash system. A comprehensive evaluation of privacy is conducted using the data collected from the prototype hardware. The specific contributions of this paper over the protocols proposed in [18] and [17] are summarized as follows:

(1) The development of a multiple cooperating TTP exchange architecture and protocol that enhances privacy and performance.
(2) The novel use of reinforcement learning algorithms to select a subset of TTPs to carry out the exchange operations, designed to either add non-determinism and unpredictability for adversaries or to increase the uncertainty in the relationship between withdrawal and spending.
(3) Implementation on a hardware prototype platform which includes nine customers and five multi-threaded implementations of the TTPs wirelessly connected to a multithreaded application running on behalf of the Bank.
(4) A detailed privacy and timing attack analysis using data collected from the hardware prototype.

2 RELATED WORK

Electronic cash and digital currency are synonymous concepts in literature. This is a well-studied topic where the focus is two-fold: preserving the privacy of payments and eliminating double-spending. One of the earliest schemes emerged from Chaum, Fiat, and Naor (CFN) in the 1970s. CFN explored anonymous payments through blind signatures [9] leading to the product termed Digicash [10] which was actually available to the public for a period of time. CFN was intended for online purchases, where the merchant could validate the coin at the time of transaction, eliminating double-spending. In an offline scenario, double-spending was detected, but not eliminated unless the underlying hardware relied on tamper-resistance. Improvements to CFN were introduced through batching techniques [23] to reduce storage and bandwidth requirements, and switching to Sigma protocol to generate a more efficient zero-knowledge proof compared to the cut-and-choose scheme of the original[5]. Divisibility proved problematic in CFN and variant schemes. This problem was solved through the first coin division scheme albeit restricted to powers of two [20], and then further refined by Camenisch, Hohenberger, and Lysanskaya (CHL), who leveraged CFN and Okamato schemes to develop an efficient, arbitrarily divisible solution [7][8]. Existing e-Cash schemes rely on some variant of the factorization or the discrete logarithm problem, both of which are susceptible to Shor's algorithm [24]. Recent attempts towards quantum-hard electronic cash include the use of lattice-based asymmetric key exchanges to implement blind signatures [22], or electronic money built on the super-position of quantum states [2] [3].

The remainder of this paper is organized as follows. Section 3 provides an overview of the proposed PUF-Cash system while Section 4 describes the details of the PUF-Cash message exchange protocol. Section 5 describes the reinforcement learning algorithms for optimizing system throughput and maximizing user privacy. Experiments are presented in Sections 6 and 7 and a privacy analysis is presented in Section 7.4.

3 THE PUF-CASH ELECTRONIC MONEY SYSTEM

PUF-Cash is an anonymous, electronic cash protocol built on physically unclonable functions (PUFs) that is usable in both online and offline contexts [6]. PUF-Cash leverages physical unclonable functions (PUFs) for nearly all security functions, including authentication, cryptographic key and random number generation. The hardware embedded delay (HELP) PUF in particular was used [1] to develop and test the protocol, although any strong PUF with matching primitives can be substituted. Message exchange protocols described in [13] and [21] provide security guarantees within

the protocol. The statistical quality of the bitstrings and keys [11], robustness [12] and resistance to machine learning attacks [14] were also explored.

The heart of PUF-Cash is an immutable indivisible e-Cash Token, represented by a 128-bit random nonce. Each nonce is unique and captures the smallest denomination. For convenience this value is set to one cent but the distinction is arbitrary. Tokens are issued against account balances by the Bank, and are linked directly to customers. A trusted third party (TTP) carries out the token blinding process, which swaps a token issued to Alice by the Bank during the withdrawal operation with an equivalent token that is not linked to Alice. The blinded tokens are identical in value to the issued token and can be validated by the Bank upon deposit, but are unlinkable to a specific withdrawal or user. On deposit, the Bank verifies the token's existence and updates Bob's balance. Note that tokens are ephemeral: They are only issued once and destroyed on redemption. Alice can possess blinded tokens for a long time before spending them, thus, the tokens must be stored in a non-volatile memory (NVM) on her device. NVM storage requirements for a large withdrawal of, e.g., $1,000, is 1.6 megabytes. Much larger amounts can be easily accommodated by low cost commodity NVM modules. Tokens are encrypted at rest, obviating the need for a secure NVM module. Note that Banks and credit card companies are likely to limit withdrawal amounts to several thousand dollars so a small NVM of this size and cost is adequate. PUF-Cash possesses all the security properties of an e-Cash protocol: Double-spending is eliminated in the online scenario and extremely difficult in the offline case. The protocol guards against spoofing, man-in-the-middle, replay and impersonation attacks. Lightweight primitives in the form of an XOR encryption scheme, AES-128 encryption and Diffie-Hellman key exchange enable the protocol to be implemented on low-power and embedded devices.

3.1 Multiple TTPs in PUF-Cash

The introduction of multiple TTPs in PUF-Cash increases operational resilience, system throughput and, through appropriate configuration, user privacy [18]. The new protocol includes provisions for TTP hot-swapping, which addresses the single-point-of-failure problem associated with the original scheme [6] and enables the system to scale according to customer demand. In addition, multiple TTPs significantly decrease the turn-around time for customers engaging in the withdrawal and blinding processes, increasing system throughput.

In PUF-Cash, the timing between issuance, blinding, transaction, and deposit/redemption, is highly variable. This property allows the system to operate offline for finite periods of time, and enables the protocol to support a wide variety of payment timelines. The most difficult timeline to deal with while preserving privacy is as follows. Alice does not withdraw issued tokens until the time of purchase. Meanwhile Bob is a fully connected payee, such as a merchant terminal or an online payment processor, that does not store tokens locally. Therefore, when Alice transacts with Bob, Alice will withdraw issued tokens from the Bank, request and withdraw blinded tokens from the TTP, and transfer blinded tokens to Bob for goods and services, all in a very short span of time. This use case is expected to be very common for erstwhile connected users who may feel safest leaving their money in the bank. Under these conditions, Alice may be exposed to a potential timing attack from the Bank, which may be able to correlate the issuance of an unblinded token with the redemption of a blinded token to extract her transaction history. Much of the risk in this timeline stems from the fact that a singular TTP is responsible for the token blinding step, whereas multiple cooperating TTPs can sufficiently mask Alice's behaviour.

3.2 Privacy Guarantees

Privacy issues arise for consumers when other entities are capable of tracking their purchases, as is true for credit cards. The blinding scheme therefore plays a crucial role in disconnecting withdrawal patterns from spending patterns.

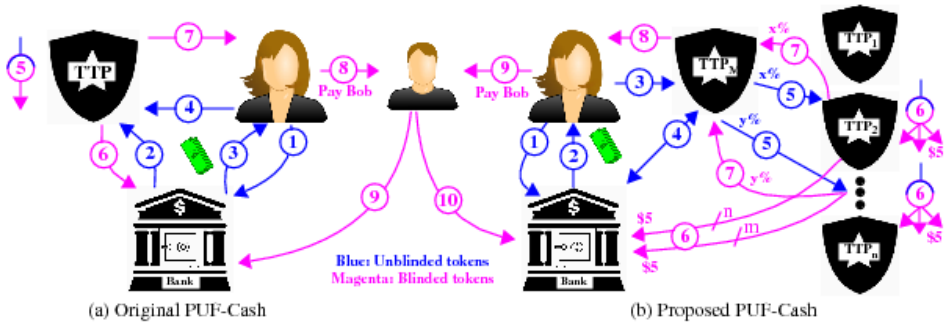

Fig. 1. (a) Withdrawal and exchange under single TTP PUF-Cash (b) Under multi-TTP PUF-Cash. Non-anonymous transactions are colored blue while anonymous transactions are magenta.

If suitably constructed, the blinding scheme will enable users to blend in with other users. In an e-Cash scheme, an attainable ideal is the N-privacy model, where any malicious actor's ability to match a user to a transaction is $1/N$, with N being the total number of users participating in the system.

Under both the single TTP [6] and prior proposed multi-TTP models [18] [17], the blinding process allowed arbitrary amounts and fractions to be blinded. However, arbitrary amounts and fractions are unique and therefore traceable to individual users. Infrequent withdrawal amounts, such as large values, are also identifiable. A malicious Bank, a malicious TTP or another entity with access to historical data could conduct profile building attacks. Coupled with metadata arising from standard communications mediums (such as geolocation of IP addresses), such attacks could succeed even in cases where information is partially available and access is pseudo-anonymous (i.e. users are not linked to individual PUF-Cash device IDs or IPs). Figure 1 illustrates the blinding schemes in both the original protocol relying on a single TTP and the proposed protocol with multiple TTPs. The comparison format highlights improvements in the proposed version. The following steps are performed in the original protocol for withdrawal and blinding:

(1) Alice requests an e-Cash withdrawal. The Bank checks Alice's balance, and if sufficient, issues a set of tokens against her account, decrementing the balance in the process.
(2) The issued tokens are sent first to the TTP.
(3) Once receipt is acknowledged, the tokens are forwarded to Alice. The ordering is crucial to prevent race conditions resulting in double-spend attempts.
(4) Alice contacts the TTP to exchange her issued tokens for blinded tokens.
(5) The TTP carries out the conversion by generating a set of random nonces as replacements for the issued tokens.
(6) The TTP transmits the blinded tokens to the Bank and waits for acknowledgement. The Bank stores the tokens for redemption later during deposits.
(7) Once acknowledged by the Bank, the TTP transmits the blinded tokens to Alice.
(8) & (9) Alice and Bob participate in the remainder of the protocol to transact and then deposit.

The proposed protocol introduces additional privacy-preserving concepts, captured in the following steps:

(1) Alice requests an e-Cash withdrawal in a manner identical to the original protocol. However, unlike the original protocol, withdrawal amounts are limited to fixed values indexed as multiples of the lowest denomination. For the present work, the lowest denomination is limited to 500 tokens, or $5, however, any amount may be used.
(2) The Bank confirms Alice's balance and generates a set of issued tokens. These tokens are transferred to Alice but not the TTP.

(3) Alice selects a Manager TTP, TTP_M, from a subset of TTP, TTP_1 through TTP_n, to conduct the blinding on her behalf. This stage is Layer 1 of the RE-Learn strategies described in Section 5.
(4) The Manager TTP contacts the Bank to validate Alice's tokens. The Bank looks up each issued token in its database. For each confirmed token, the Bank updates its status to received, preventing anyone else from resubmitting the same token. Note that the tokens are bound to Alice on issuance and Alice's PUF is required to decrypt the tokens in the first place. The Bank sends an acknowledgement to TTP_M once all issued tokens are validated.
(5) TTP_M then runs an RE-Learn algorithm to select a set of Submitter TTPs from the set TTP_1 through TTP_n, excluding itself, to perform the exchange operations. Once selected, TTP_M runs a novel privacy-enhancing version of RE-Learn that partitions the blinded tokens into a set of fractions and designates a Submitter TTP to blind each fraction. This stage is Layer 2 of RE-Learn.
(6) The Submitter TTPs further partition the fractions of blinded tokens that they receive into fixed sized $5 chunks (other small denominations are possible), and proceed to generate random numbers that will serve as the blinded tokens for each chunk. The blinded tokens are transmitted to the Bank in fixed sized $5 chunks.
(7) The Submitter TTPs transmit the blinded tokens to the Manager TTP.
(8) Once the Manager TTP has collected all outstanding fractions, it transmits the entire set of blinded tokens to Alice.
(9) & (10) Alice and Bob participate in the remainder of the protocol to transact and then deposit.

This concludes a high-level survey of the protocol. A detailed description is presented subsequently in the following section, covering cryptography constructs and packet structure that lend the protocol its security.

4 PUF-CASH V2.0 TRANSACTION PROTOCOL

The introduction of multiple TTPs necessitated changes to the protocol. Note that the changes proposed in this paper apply solely to the Withdraw and the n_1-to-n_2 blinding steps (referred to as 'Exchange' in the subsequent description). The remaining transactions, including the Customer-to-Customer Payment and Customer Deposit transactions, operate identically to the original protocol and are omitted from the descriptions and evaluations.

The message exchange diagram that characterizes the operations is shown in Figs. 2 and 3. The entities involved are labeled along the top in the figure and include from left-to-right, the customer, Alice, the Bank and a set of TTPs, labeled TTP_1 through TTP_n. The following description numbers the steps in correspondence with those given in the figures as circled numbers. Note that the transaction sequence is shown with respect to the customer Alice. As indicated, the Bank selects a set of TTPs that become members of the PUF-cash network, which can vary from at least 2 (the minimum) to as many as one hundred (our hardware experiments utilize 5 TTPs). As we show, the number of TTPs impacts privacy but only a relatively small number of TTPs are needed to achieve the N-privacy goal discussed earlier. The term n_1 refers to issued tokens and n_2 refers to blinded tokens in the following, to reinforce that the e-Cash tokens are in fact nonces (random numbers).

(0) Each of the TTPs, TTP_1 through TTP_n, contacts the Bank to authenticate and generate a session key, SK_{tx}. The Bank validates that their IP addresses are in its list of trusted TTPs before authenticating. Authentication and session key generation is PUF-based, where the Bank accesses TTP enrollment data stored in a database (not shown). The session keys SK_{t1} through SK_{tn} are then encrypted and distributed to each of the TTPs to enable the

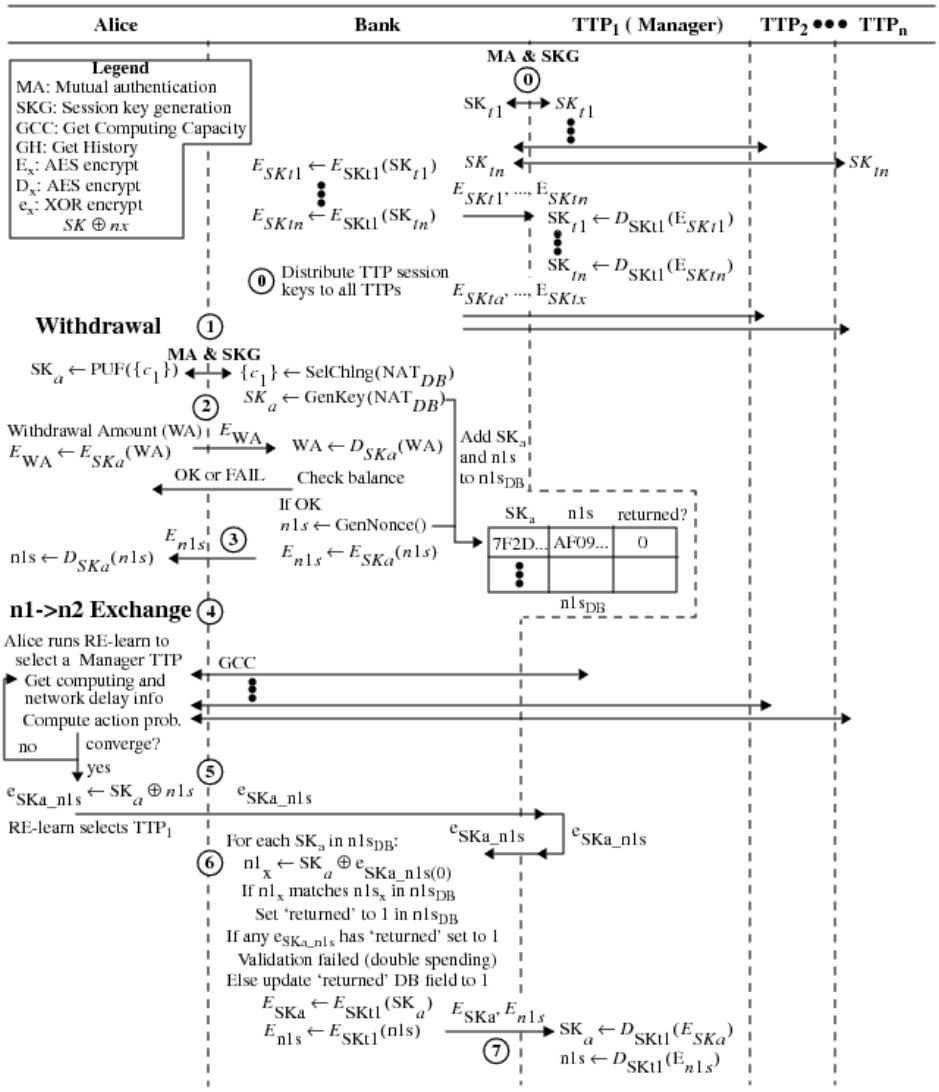

Fig. 2. PUF-Cash Multi-TTP Protocol.

TTPs to authenticate and encrypt communications among themselves. Although not shown, each TTP session key has an associated IP address that is also distributed to the TTPs by the Bank.

(1) Alice requests a withdrawal from her account at the Bank. The Bank first generates random challenges c_x (not shown, see [6]) for carrying out mutual privacy-preserving authentication, and then generates a random challenge c_1 for session key generation. Alice receives c_1 from the Bank and generates a session key SK_a. The Bank applies a software version of the PUF-based session key generation algorithm using enrollment data from the database to generate its copy of SK_a.

(2) Alice encrypts her withdrawal amount WA with SK_a and transmits it to the Bank. The Bank decrypts it and checks that WA does not exceed her account balance. A message is sent back to Alice either confirming or

denying the withdrawal request. If the withdrawal is permitted, the Bank calls *GenNonce()* to generate a set of n_1s corresponding to the withdrawal amount. In the current protocol, one 128-bit nonce represents 1 cent. Note that similar to automatic teller machines (ATMs), Alice must withdraw money in multiples of a minimum amount. We allow the minimum amount to be $5 in PUF-Cash. Therefore, for a withdrawal amount of $5, the Bank generates 500 128-bit n_1 nonces.

The n_1s are added to the $n1s_{DB}$, which has three fields as shown in the center of Fig. 2. The SK_a field stores Alice's session key and the n_1s field stores a copy of the n_1s generated by the Bank. The *returned* field tracks which of Alice's n_1s have been converted into anonymized e-Cash tokens. It is a Boolean array with one Boolean flag corresponding to each of the 128-bit n_1s. The initial values in this array are set to 0 (False).

(3) The Bank encrypts the n_1s with SK_a and transmits them to Alice. Alice decrypts the n_1s and typically stores them locally in an non-volatile-memory (NVM) for future use, or she can immediately begin the e-Cash token anonymization process, identified as the *n1-to-n2 Exchange* operation in the figure. This concludes Alice's withdrawal transaction with the Bank.

(4) The sequence that begins with a circled 4 in Fig. 2 is the beginning of a separate transaction designed to anonymize Alice n_1s to a new set of e-Cash tokens called n_2s. As discussed, once anonymized, Alice can pay Bob with n_2s, and later the Bank can confirm that they are valid but without being able to link them to Alice. The *n1-to-n2 Exchange* operation begins with Alice selecting a Manager TTP. The Manager TTP will act on behalf of Alice during the *n1-to-n2 Exchange*. Engaging a Manager TTP for this operation plays a key role in both maintaining important security properties of the PUF-Cash protocol and in preserving privacy. This goal is addressed using reinforcement learning, labeled RE-Learn in the figure, where Alice uses RE-Learn to select the Manager TTP from a list of available TTPs. We assume Alice has chosen TTP_1 as the Manager TTP in the following.

(5) Once a Manager TTP is selected, Alice XOR-encrypts her n_1s with her session key to generate a set of e_{SKa_n1s} for transmission to the Manager TTP. The Manager TTP simply forwards the e_{SKa_n1s} to the Bank.

(6) The Bank is now charged with validating the e_{SKa_n1s}. To do so, it needs to remove the SK_a component from the e_{SKa_n1s} to recover the n_1s. It carries out an exhaustive search of the $n1s_{DB}$ by retrieving each SK_a, one at a time, from the database, and then recovers the n_1 from the first e_{SKa_n1s} through XOR decryption. The recovered n_1 is compared with the first n_1 stored with this SK_a in the database. If a match occurs, then the remaining e_{SKa_n1s} are processed in a similar fashion using the $n1s$ associated with this database element. For each recovered n_1 that matches an n_1 in the database, the *returned* flag is set to True in the database. If a matched n_1 from the database already has its *returned* field set to 1, then the Bank flags an "Attempt to Double Spend" error on the entire transaction. This prevents Alice from attempting to anonymize her n_1s more than once.

(7) Assuming no "Attempt to Double Spend" errors are generated, the Bank encrypts the matching SK_a and recovered n_1s with the Manager TTP's session key, SK_{t1}, and transmits the encrypted versions to the Manager TTP. The Manager TTP decrypts the SK_a and n_1s. The SK_a will be used later to encrypt the n_1s and newly generated n_2s for transmission to Alice.

(8) Fig. 3 diagrams the remaining transactions associated with the *n1-to-n2 Exchange*. As part of RE-Learn, the Manager TTP selects a set of Submitter TTPs and engages them to generate portions of Alice's anonymized n_2s. The Manager TTP itself does not generate n_2s as this would reduce the uncertainty in n_1-to-n_2 time correlation attacks that may be carried out by the Bank (described in Section 7.5). RE-Learn optimally selects the Submitter TTPs to load balance the n_2 generation tasks to improve system throughput. The selected subset of TTPs,

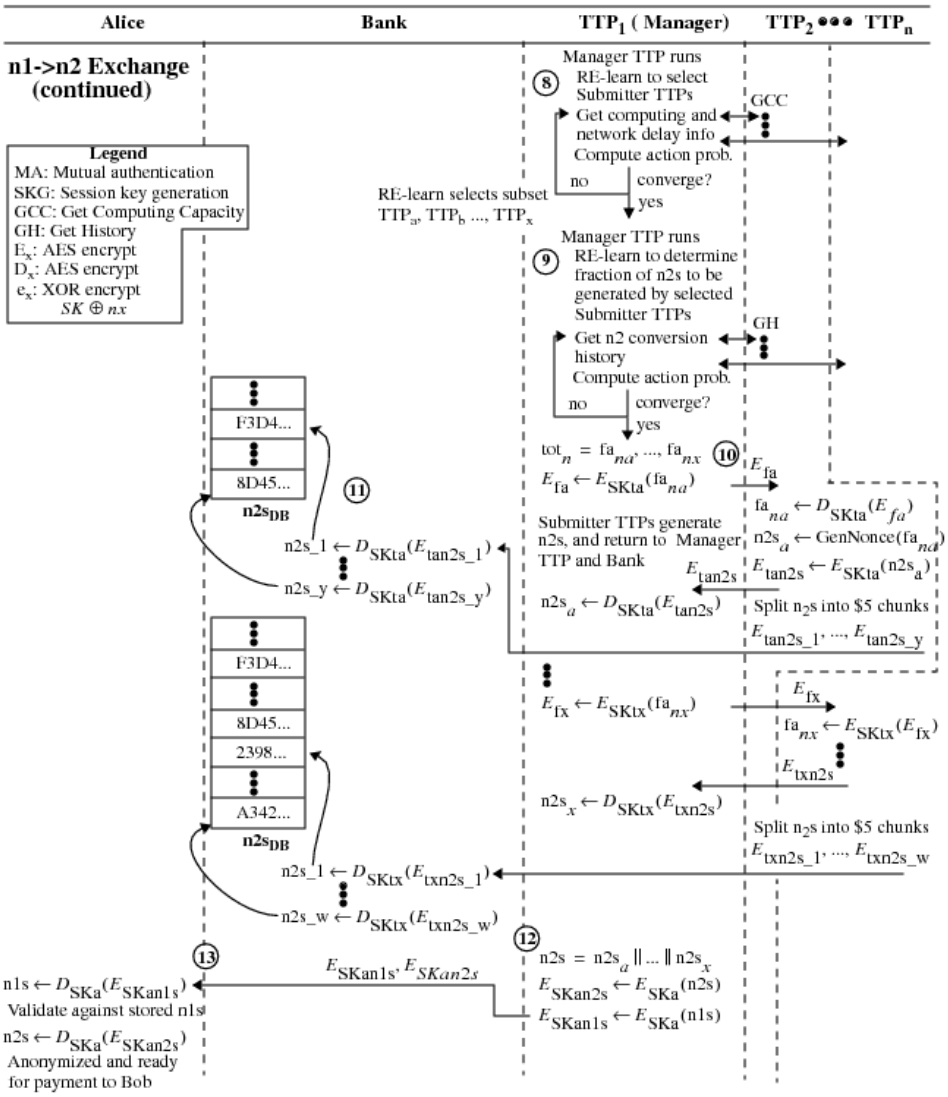

Fig. 3. PUF-Cash Multi-TTP Protocol (continued).

identified as TTP_a through TTP_x, is contained within the larger set of TTPs labeled TTP_2 through TTP_n in Fig. 3. Note however that the Manager does not share the final Submitter TTP selection with the Bank.

(9) The Manager then runs RE-Learn once again to determine the optimal fractional split across TTPs in a manner that maximizes Alice's privacy. The details of RE-Learn are discussed in Section 5.2.

(10) On convergence, a set of fractions, labeled fa_{na} through fa_{nx} in Fig. 3, are produced that represent the n_2 size requests made by the Manager TTP to the Submitter TTPs. The sizes are encrypted and transmitted to each selected Submitter TTP, which then decrypts the requests and calls *GenNonce()* to generate the requisite number of n_2s. Each Submitter TTP partitions the n_2s into $5 sets, then encrypts and transmits them to the Bank. The generated n_2s are also transmitted to the Manager TTP for disbursement to Alice.

(11) The Bank decrypts the n_2 chunks received from the Submitter TTPs and adds them to the $n2s_{DB}$ database.
(12) The Manager TTP also decrypts the n_2 fractions and concatenates them together into a list of n_2s. It then encrypts the n_2s and the n_1s received earlier from the Bank using Alice session key, SK_a, and transmits both the E_{SKan1s} and E_{SKan2s} to Alice.
(13) Alice decrypts E_{SKan1s} and E_{SKan2s} and compares the decrypted n_1s with the n_1s she originally received from the Bank during the withdrawal operation. If they match, then the transaction is validated. She can then use her anonymized n_2s as payment in future transactions with Bob(s).

Note that steps 1 through 7 are not anonymous, i.e., Alice is known to the Bank during these components of the n_1-to-n_2 exchange process. On the other hand, the remaining steps shown in Fig. 3 beginning with step 8 are anonymous. Here, the Manager TTP interacts with the Submitter TTPs independent of the Bank to convert the n_1s to blinded n_2s. A proof regarding the security guarantees of the underlying XOR transport channel is presented in [6].

5 RE-LEARN - STOCHASTIC LEARNING AUTOMATA

From an infrastructure perspective, an electronic cash system is a distributed network on a nation-wide scale. In such a network, the computing needs and resource availability are changing dynamically with system load and network connectivity. Multiple TTPs were introduced to adequately distribute the load based on compute capacity and network conditions, thus enabling PUF-Cash to scale with demand. Reinforcement learning (RL) was introduced in prior work [18] and [17] as an adaptive strategy to optimally select TTPs and roles under dynamically evolving environmental conditions [19]. The described layers optimally select both the subset of available TTPs based on compute capacity, network behaviour, and the fractional division of workload to obfuscate Alice's spending behaviour from the Bank.

5.1 Layer 1

Layer 1 is responsible for optimally selecting TTPs based on three criteria, as per prior work [18]: total compute capacity, current computing load, and network congestion. By optimizing all three criteria within the reward function, the first layer maximizes system throughput. Note that in prior work [18], the Submitter TTPs were chosen and coordinated by Alice. This self-coordination strategy led to a race condition that a malicious Alice could potentially exploit, letting her double-blind her tokens and double-spend her money. In the proposed Layer 1, Alice selects a TTP delegated as the Manager TTP responsible for coordinating the blinding process on behalf of Alice. Since the TTP is a trusted entity, the Bank can trust the TTP to correctly execute the steps of the protocol, preventing double-spending. The Manager TTP in fact never learns Alice's identity, and leverages the Bank to validate Alice as a legitimate customer. Therefore, Layer 1 is executed by both Alice and the Manager TTP separately. First, Alice selects the Manager TTP near-optimally utilizing SLA. Once selected, the Manager TTP runs SLA to near-optimally select the Submitters. The SLA strategy for both of these operations is based on compute capacity, current computing load, and network congestion and is identical to Layer 1 employed in [18] and [17].

Let us define the set of the available TTPs as $T = \{TTP_1, \ldots, TTP_t, \ldots, TTP_{|T|}\}$. Note that the total number of TTP in the entire system can be as large as one hundred but Alice restricts her communication to a predefined subset of TTPs that are in her vicinity. The set of TTPs Alice can utilize are determined and periodically updated by the Bank. Each TTP_t has a computation capability $F_{TTP_t}[\frac{CPU\ cycles}{unit\ operation}]$, derived empirically from the Cora boards [16] used in the experiments, based on the effort required to carry out a small set of dummy plaintext AES encryptions. The latency between Alice and a TTP_t is denoted as $d_{Alice,TTP_t}[s]$, also measured empirically using a network request from Alice

to obtain the computing capacity from the TTPs. Based on the communication delay (denominator of Eq.1) and the computational congestion (numerator of Eq.1), Alice determines a corresponding personalized reward $r^{(ite)}_{c,\mathbf{STR_s}}$ at the ite iteration of the SLA algorithm in the first decision-making layer as follows.

$$r^{(ite)}_{TTP_t} = \frac{\sum\limits_{TTP_t \in \mathbf{T}} \frac{F_{TTP_t}}{|C_{TTP_t}|_{(ite-1)}}}{\frac{\sum\limits_{TTP_t \in \mathbf{T}} d_{TTP_t}}{\sum\limits_{\forall TTP_t \in T} d_{c,TTP_t}}} \tag{1}$$

Given this probability, suitably normalized as $\hat{r}^{(ite)}_{TTP_t}$, Alice determines the action probability vector, where the individual probabilities are subsequently determined based on the SLA update rule (Eq. 2) [26].

$$Pr^{(ite+1)}_{TTP_t,\mathbf{T}} = Pr^{(ite)}_{TTP_t,\mathbf{T}} + b\hat{r}^{(ite)}_{TTP_t,\mathbf{T}}(1 - Pr^{(ite)}_{\mathbf{TTP_t}}),\ ^{(ite+1)}_{TTP_t} = ^{(ite)}_{Manager} \tag{2a}$$

$$Pr^{(ite+1)}_{TTP_t,\mathbf{T}} = Pr^{(ite)}_{TTP_t,\mathbf{T}} - b\hat{r}^{(ite)}_{\mathbf{TTP_t}} Pr^{(ite)}_{TTP_t,\mathbf{T}},\ ^{(ite+1)}_{TTP_t} \neq ^{(ite)}_{Manager} \tag{2b}$$

where $0 < b \leq 1$ is the learning parameter (for smaller values of b the end-user explores more of the available strategies). Eq.2a expresses the probability of selecting the same strategy $\mathbf{STR_s}$ in iteration ite, while Eq.2b represents the probability of selecting a different strategy. At each iteration, Alice may receive an update on the environmental parameters. It is noted that the probabilities are initialized uniformly to prevent bias. Note that TTPs are multi-threaded, therefore a TTP selected to be Manager for Alice could be serving as a Submitter for another customer Charlie.

Once a Manager TTP is selected, the Manager TTP is tasked to conduct a Submitter TTP selection on Alice's behalf. The Manager TTP may select up to Q TTPs, such that $Q \leq |T|$, to distribute her n_1 tokens for conversion. Therefore, the strategy space consists of vectors $\mathbf{STR_s} = [TTP_A, TTP_B, \ldots, TTP_N]$, where $s \in S = \{1, \ldots, s, \ldots, |S|\}$ and $|S|$ is the total number of distinct subsets of the Q TTPs. The reward is designed to select as many Submitter TTPs as possible, up to the maximum given by the list provided by the Bank. This approach enhances privacy although that is not the primary goal of Layer 1. Note that for smaller withdrawals, the number of Submitter TTPs to select is determined by the minimum n_2 exchange amount, which, although configured for 500 tokens ($5) in the experimental implementation, can be set to any value. The Manager TTP does not share her Submitter selection outcomes with the Bank. The reward function ((Eq. 3) is identical to prior work with the user c replaced by the Manager TTP $TTP_{Manager}$, similar to Eq. 1.

$$reward^{(ite)}_{TTP_{Manager},\mathbf{STR_s}} = \frac{\sum\limits_{TTP_t \in \mathbf{STR_s}} \frac{F_{TTP_t}}{|C_{TTP_t}|_{(ite-1)}}}{\frac{\sum\limits_{TTP_t \in \mathbf{STR_s}} d_{,TTP_t}}{\sum\limits_{\forall TTP_t \in T} d_{Manager,TTP_t}}} \tag{3}$$

Similarly, the action probabilities can be defined as per Eq. 4,

$$Pr^{(ite+1)}_{s,\mathbf{STR_s}} = Pr^{(ite)}_{s,\mathbf{STR_s}} + b\hat{r}^{(ite)}_{c,\mathbf{STR_s}}(1 - Pr^{(ite)}_{s,\mathbf{STR_s}}),\ ^{(ite+1)}_{\mathbf{STR_s}} = ^{(ite)}_{\mathbf{STR_s}} \tag{4a}$$

$$Pr^{(ite+1)}_{s,\mathbf{STR_s}} = Pr^{(ite)}_{s,\mathbf{STR_s}} - b\hat{r}^{(ite)}_{c,\mathbf{STR_s}} Pr^{(ite)}_{s,\mathbf{STR_s}},\ ^{(ite+1)}_{\mathbf{STR_s}} \neq ^{(ite)}_{\mathbf{STR_s}} \tag{4b}$$

where, once again, $\hat{r}^{(ite)}_{c,\mathbf{STR_s}}$ is the normalized reward probability and Eq.4a expresses the probability of selecting the same strategy $\mathbf{STR_s}$ in iteration ite, while Eq.4b represents the probability of selecting a different strategy. On convergence, the near-optimal set of TTPs define the selection strategy $\hat{s}$, which serves as the input to Layer 2.

5.2 Layer 2

Unlike Layer 1, the construction of Layer 2 is designed to maximize privacy for the users. An N-privacy model is assumed, where user membership is known but transactional identity is masked. In the proposed protocol, the Manager TTP receives encrypted tokens from Alice and requests validation from the Bank. The Manager TTP then instructs the Submitters to blind specific fractions of Alice's tokens, and returns the blinded tokens to Alice. Layer 2 imposes the following restrictions to safeguard the privacy of users:

(1) All amounts are decomposed into predetermined fixed-size denominations or bins.
(2) The RE-Learn algorithm is employed to choose the fractions in near-optimal manner, where optimality is defined by privacy maximization.
(3) Each Submitter returns blinded tokens to the Bank in the lowest denomination.

The combination of steps thwart timing attacks from the Bank and user profile building attacks by the Manager TTP or other actors with access to historical system data. First, the decomposition of amounts into fixed fractional denominations enforces a degree of user similarity to first order. This behaviour is consistent with modern automatic teller machines (ATMs) where cash withdrawals are fixed to specific denominations. Then, RE-Learn chooses these fractions in an near-optimal way that further strengthens the similarity between users. In the present work, the fixed denominations were chosen as per Table 1. Note that any denominational selection will work as long as larger values are integer multiples of the lowest denomination.

Once the Manager TTP has completed selecting Submitters based on the output of Layer 1, it employs Layer 2 to determine the appropriate fractions per TTP. To accomplish this, the Manager first requests each Submitter TTP for its blinding *history*. The requests for history information are labeled *GH* for 'get history' in Fig. 3. Each Submitter TTP maintains a history data structure as an array of counters or bins, where each bin keeps track of the request count for a specific denomination. The Manager is constrained to make fractions of n_2 requests in units of these fixed denominations. An efficient integer partition technique coupled with a binary search is employed to restrict the learning space to a set of valid solutions that maps to the finite set of denominations. Each solution, denoted as $q = [H_{k,1}, H_{k,2}, \ldots, H_{k,t}]$, denotes the history H for the unique bin k in TTP t from the chosen strategy $\hat{s}$. The set of valid solutions is termed $\mathbf{Q} = [q_1, q_2, \ldots, q_n]$.

RE-Learn executes a *MostFrequentlyUsed* (MFU) strategy in Layer 2 on $\mathbf{Q}$. In the MFU strategy, the reward function, defined in Eq. 5 computes the sum of historical counts for a given valid strategy, $\hat{q}$. The per-strategy reward is normalized over counts of all bins over all TTPs to ensure that the value remains below unity. In effect, the algorithm will evaluate each Submitter TTP's past history and pick bins similar to past blinding requests to reinforce prior bin selections. The action probabilities updates are roughly identical to Eq. 4, where the reward for a particular set of TTPs can be substituted by a reward for a particular set of fractions per TTP. The learning parameter, b, is retained at a value of $b = 0.7$. Since SLA is non-deterministic, convergence is controlled by thresholds on the reward and number of iterations.

$$reward^{(ite')}_{\hat{\mathbf{q}}|\hat{\mathbf{s}}} = \frac{\sum\limits_{\forall TTP^*_t \in \hat{\mathbf{s}}}^{t} \sum\limits_{\forall k \in Q}^{k} H_{k,t}}{\mid \mathbf{STR_s} \mid \cdot \max\left(\sum\limits_{\forall TTP^*_t \in \hat{\mathbf{s}}}^{t} \sum\limits_{k=1}^{N_{bins}} H_{k,t} \right)} \tag{5}$$

A worked example is presented to illustrate the workings of Layer 2. Table 1 captures the denominational breakdown employed in the present work, while Table 2 shows an example history data structure obtained during the hardware experiments described in Section 7. Denominations are restricted to ten bins, with the lowest bin denomination defined

Table 1. Set of Fixed Denominations (Bins)

Denomination (Bin)	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
Value	$5	$10	$15	$20	$25	$30	$35	$40	$45	$50

Table 2. Example history data structures for 5 TTPs with fixed denominations from Table 1

	$5	**$10**	**$15**	**$20**	**$25**	**$30**	**$35**	**$40**	**$45**	**$50**
TTP_1	235	157	123	131	77	72	49	70	29	48
TTP_2	199	138	122	109	94	91	70	76	54	44
TTP_3	195	145	122	109	94	91	70	76	54	44
TTP_4	211	122	132	107	83	87	90	63	63	39
TTP_5	173	135	126	144	98	103	71	48	64	30

at 500 tokens or $5 and the largest bin denomination at 5000 tokens or $50. The example in Table 2 assumes the availability of five TTPs and a $50 dollar withdrawal amount. The history for each of the TTPs are given on the rows while the columns represent the bins labeled with n_2 request sizes from $5 to $50 in increments of $5. TTP_5 was chosen as the Manager TTP, and TTPs 1 through 4 were chosen as Submitter TTPs. Subsequently in Layer 2, valid fraction sets restricted to bin denominations are generated, then the SLA algorithm is executed on the amount of $50. One possible breakdown given the fixed sized bins is $5, $5, $5 and $35 (totaling $50) for TTP_1 through TTP_4, which produces a sum of 235 + 199 + 195 + 90 = 719. RE-learn randomly visits a portion of the other possible breakdowns to find the maximum sum, and returns a combination that is 'near' optimal, where optimal is defined as the largest sum among the valid combinations. Once the Manager TTP makes a selection and transmits the n_2 requests to the Submitters TTPs, each of the Submitters increments the corresponding bin count in their row array of counts. Note that updates to the history are fetched over the duration of the algorithm, so RE-learn adapts to changes in the Submitter TTP history while it runs.

Each Submitter TTP's history is initialized (at time 0) to the same constant value across all denominations. Over time, it can be observed that the Submitters tend to converge to a preferred configuration under the MFU approach. Furthermore, the Submitter TTPs limit the sum of their history table values (as shown across any given row of Table 2), but decrementing all row values by 1 on any update that causes the sum to exceed 1000, in the spirit of a least-recently-used algorithm. This prevents overflow and nudges the Submitters into convergence over time.

Once the Submitter TTPs have generated the n_2 fractions, they transmit them to both the Manager TTP and the Bank. The further partitioning by the Submitter TTPs of the n_2s into $5 chunks is an additional anonymity-enhancing step. Note that although only one conversion request is depicted in 3, the Submitter TTPs are multi-tasking, and therefore, they can be (and likely are in a moderately loaded operational system) simultaneously carrying out n_2 generation tasks for other Manager TTPs. In addition to being oblivious to the Submitters chosen by the Manager TTP, the Bank only ever converts $5 denominations, and is limited to probabilistic guessing which $5 chunks of n_2s belong to Alice. An extensive time correlation analysis is provided in the Section 7.5

6 EXPERIMENT SETUP

The system architecture used in the experimental evaluation of the proposed protocol is shown in Fig. 4. The goal of the experimental evaluation is to measure the network and computing performance of the proposed protocol, and to evaluate the overhead associated with the message exchanges, with a particular focus on the reinforcement learning algorithms employed to make decisions regarding the selection of TTPs and the distribution of e-Cash tokens. The hardware used in the experiment consists of off-the-shelf FPGA boards, in particular, Cora-Z7-07S are used for the TTPs while Zybo-Z7 are used for the customers. The Cora-Z7-07S are equipped with 802.11 wireless network adaptors

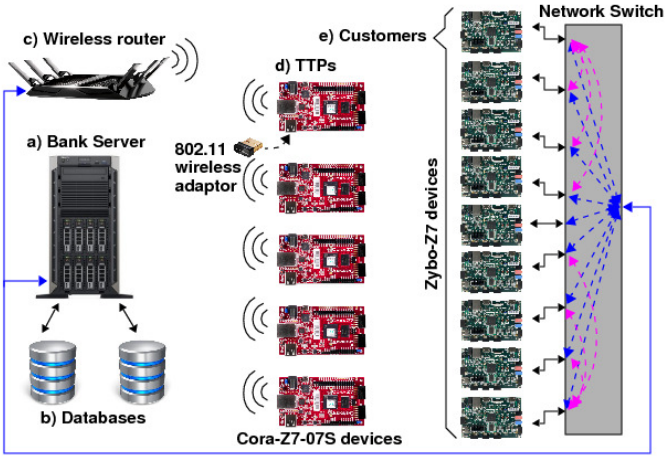

Fig. 4. System architecture for Experiments. a) Multi-threaded Bank application communicates through a wired connection to customers connected to a network switch, b) Bank manages PUF enrollment data and PUF-Cash tokens using a set of databases, c) a wireless router provides network connectivity for the TTPs, d) multi-threaded TTPs run on embedded-system Cora-Z7-07S boards and e) Alice, Bob, etc. (customers) run application on Zybo-Z7 boards. TTPs and customers leverage an FPGA PUF for mutual authentication and session key generation.

and connect to a Netgear NightHawk router. The NightHawk is connected through a 1 GB/sec wired connection to a network switch and the Bank's server, which utilizes a Dell PowerEdge T440 Server with 32 1.8 GHz processors and 128 GB of main memory. The Bank accesses PUF enrollment data for the devices and data related to e-Cash transfers using an sqlite3 relational database system. The enrollment data for 161 devices are stored in the database, which is more than 10x the number of actual devices used in the hardware setup. The database size is 455 MB so each device stores approx. 2.8 MB of enrollment data. The performance impact of using much larger enrollment data sets is presented in [6], and is omitted here.

The nine customers are configured to automatically and continuously make e-Cash Withdraw transactions and then immediately engage in a corresponding Exchange transaction. As mentioned earlier, the timeline for Withdraw and Exchange transactions in PUF-Cash is variable where Exchange transactions can (and often are) carried out when the Customer engages in a Payment transaction, which can occur minutes, hours, days or weeks after the Withdraw transaction. The experimental setup is designed to model a network where Customer transactions are occurring from a much larger number of devices, which is achieved here by having the nine Customer devices continuously repeat Withdraw and Exchange transactions with no delays artificially inserted. This increases the work loads associated with the TTPs and Bank significantly to values more representative of an actual deployment of the PUF-Cash system. Note that each pair of Withdraw/Exchange transactions executes all steps of the protocol diagram shown in Figs. 2 and 3 except step 0.

In order to accommodate the continuous stream of transaction requests generated by the nine customer devices, the protocol components that run on the Bank and TTPs are multi-threaded applications coded in the C programming language using the POSIX Pthread library. A static set of twenty threads are created at startup on each of the TTPs and the Bank to reduce the time overhead associated with creating and destroying threads. Threads are put to sleep immediately upon creation and remain idle until the master socket descriptor detects a network packet from one of the TTP or customer devices. Threads are designed to process one transaction and then voluntarily go back to sleep.

Mutexes are used to protect shared data structures, including the Bank's socket descriptor on the TTPs and the databases accessed by the Bank. A 128-bit OpenSSH version of AES (ECB mode) is used for all encryptions and decryptions.

7 RESULTS AND DISCUSSION

7.1 Summary of PUF Key and Bitstring Statistical Quality Metrics

A detailed description of the statistical quality of the bitstrings and encryption keys is reported on in previous work [15] and [6], which we summarize here. The standard metrics that have emerged for measuring and reporting bitstring quality include Intra-chip hamming distance for reproducibility, Inter-chip hamming distance for uniqueness and the NIST statistical test results and minEntropy for randomness. Our previous evaluation used 500 FPGAs tested at 12 separate temperature-voltage (TV) corners, i.e., all combinations of -40^oC, 0^oC, 25^oC and 85^oC and supply voltages 0.95 V, 1.00 V and 1.05 V. The probability of a failure is approximately 10^-6 for the HELP PUF evaluated using typical parameters within HELP's multi-dimensional challenge-response space. Inter-chip hamming distance is reported as 49.95% (near the ideal at 50%) while minEntropy varied between 91% and 95% (ideal is 100%). The bitstrings and session keys passed all NIST statistical tests.

7.2 Methodology

The starting point for experiments are user withdrawal amounts that will serve as inputs to the blinding stage. For each transaction, a user would sample the prescribed distribution, then conduct the withdrawal and blinding steps of the protocol with minimal delay. The test scenario consists of two distinct user withdrawal patterns, illustrated in Fig. 5. Performance and stability analysis of RE-Learn was conducted with each user sampling from a common distribution, shown in Fig. 5a, with the same mean and standard deviation. For convenience, a Gaussian distribution with a mean value of μ = \$100 and a standard deviation of σ = \$50 was chosen. The minimum and maximum amounts are artificially capped at \$5 and \$200 respectively. Having all users sample from the same distribution was crucial to obtaining consistent metrics on the layers.

User privacy analysis and timing attack analysis was conducted on the second withdrawal pattern, shown in Fig. 5b. This withdrawal pattern consists of each user sampling from a unique Gaussian distribution defined by a distinct mean and standard deviation. The distribution pattern of withdrawal amounts is derived from ATM withdrawal patterns, which states that the majority of Americans (60%) withdraw an average of \$40 per visit to an ATM [25]. Modeling users to be consistent with ATM withdrawal patterns should generate quasi-realistic data and place the analysis on a robust footing. As such, the mean value of each distribution is scattered across the range from \$40 to \$160. A quadratic function is modeled to generate individual mean values per user, with more users clustered at the lower bound of \$40 and increasingly fewer in count until the upper bound of \$160 is reached. The standard deviation is kept common to all users at a value of σ = \$15. Note that even though values are similar at the lower end, each user receives a unique mean value by virtue of the quadratic function, thus ensuring that no two users are identical.

7.3 RE-Learn Performance and Stability

Performance and stability of RE-Learn's Layer 1 is described in prior work [18]. The performance of Layer 2 is assessed in this work, based on the history matrix stored within each TTP. Verifying the correctness of Layer 2 is the first order of business. To that end, the top row of Fig. 6 depicts the final histograms of all TTPs based on the Random approach. In this approach, Layer 2 automatically picks one of the valid fraction splits for all TTPs selected in Layer 1. It can be

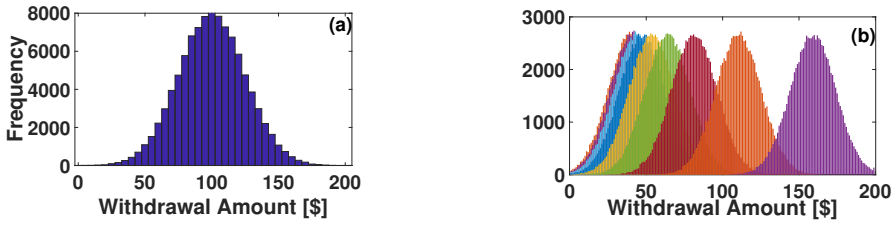

Fig. 5. User withdrawal distributions used in experimental analysis. (Left) Common distribution for all users (Right) Unique, quasi-realistic distribution per user biased towards lower withdrawal amounts.

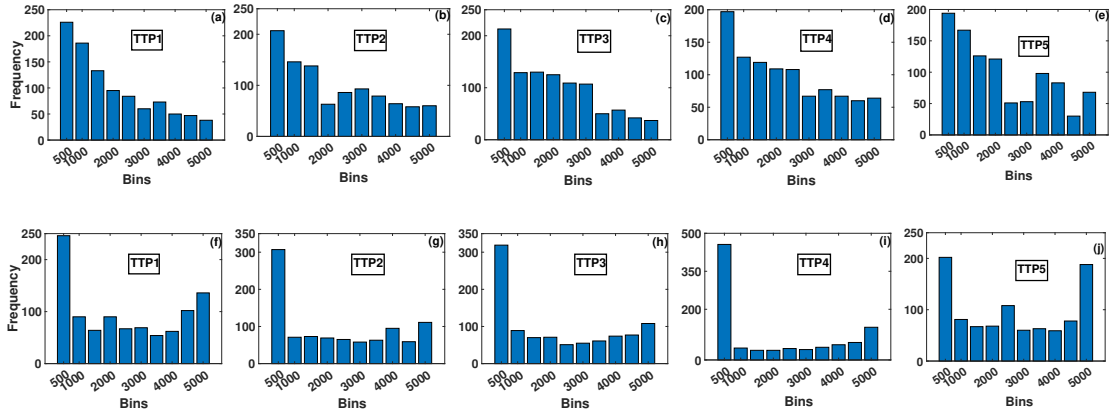

Fig. 6. Denomination histograms for all TTPs based on the Random (baseline) approach in the top row (sub-figures (a) through (e)) and the MFU approach in the bottom row (sub-figures (f) through (j)).

observed that Layer 1 is fairly uniformly distributed. There is a preference for lower amounts, however that can be explained by the distribution of solutions, as lower amounts appear more frequently in the set of valid solutions across the entire search space, with the lowest bin of 500 tokens being the most frequent of all. The difference is evident when switching to the MFU strategy (bottom row of Fig. 6). Here, all TTPs now show a clear preference for the 500 token denomination. This preference is likely tied to input user withdrawals being skewed towards the lower amounts. It is also observed that the preference is not absolute, and TTPs will gravitate towards alternatives when facilitating users with specific withdrawal patterns. This further confirms the assessment that SLA is a stochastic approach that is more like a nudge and less like an absolute decision.

Given that SLA is a non-deterministic algorithm, convergence is governed by thresholds on the maximum action probability and the iteration count. In the present experiments, a threshold was heuristically established such that the algorithm was deemed convergent when the maximum action probability exceeded 0.95 (or 95%). However, the number of iterations was left unbounded to study the behaviour of the algorithm. Fig. 7 depicts the performance of RE-Learn in Layer 2 at two distinct tiers. The input data for both tiers is sampled from the common distribution (left in Fig. 5). The first row illustrates the action probability curves for candidate solutions where each TTP acted in the role of the Manager for a single transaction. Each curve within a single TTP plot represents a unique candidate solution, where the abscissa is the iteration step and the ordinate is the cumulative action probability for that solution at that step. Transactions were chosen at random, therefore TTP behaviours are uncorrelated to each other. Note that the settling time for most TTPs exceeds 500 iterations before a clear front-runner emerges. TTP 5 is anomalous but difficult to draw

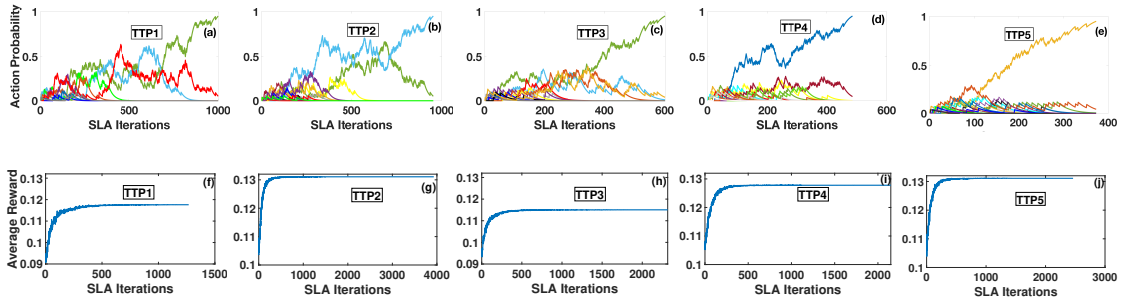

Fig. 7. Action probabilities (top row) for each TTP participating as the Manager in a single transaction. Individual curves denote the probability as a function of iteration count. The average reward curve (bottom row) depicts the average reward per iteration step for all transactions of that TTP.

conclusions from, given that it represents a single transaction. A statistical analysis (not shown) confirms an average iteration of 279.1 steps, an average reward of 0.125 and an average coverage of 80% across the solution space for all TTPs. Both aggregates suggest that the SLA algorithm performed a fairly exhaustive exploration of the search space.

The bottom row illustrates the average reward per TTP across all transactions for that TTP for a given iteration count. Each plot represents a unique TTP Manager, and each point on the curve for a given plot has the abscissa as the iteration step count while the ordinate is the average reward across all transactions for that TTP at that iteration step. As a more holistic estimate of Layer 2's behaviour, the estimate is computed as follows: First the transaction with the maximum iteration count in Layer 2's execution is extracted. Then, each Layer 2 solution space is iterated upon until the iteration count is equal to the maximum iteration count. Note that iterations are conducted on the same solution space as the original Layer 2 execution and the action probabilities can exceed well beyond convergence (95%). This is a necessary step to obtain a unique value for all transactions at a given iteration step. It is noticeable that variations are exceedingly small, although the final reward itself is fairly low, not exceeding 0.15 across all TTPs. This average low reward indicates that the normalization parameter (denominator in Eq. 5) could undergo some refinement, an area of potential further study. However, given that normalization is consistently applied across all rewards, the impact is nullified and not expected to change the near-optimal solution-seeking behaviour of RE-Learn.

Fig. 8 depicts the performance of a single TTP in the role of a Manager across all transactions. It can be observed in Fig. 8a that the reward ranges from 0.05 to 0.25 across all transactions, with the most frequent reward closer to the lower end and a long tail, suggesting a Poisson distribution. Similarly, the iteration count in Fig. 8b ranges anywhere from 0-1200 iterations, a fairly expansive range that is liable to impact system throughput for certain unlucky users. The large variation in iterations can be attributed primarily to the similarity between TTPs. Since all TTPs possessed identical compute capacities, by virtue of being based on the same hardware and software platform, and identical network latencies, by virtue of being co-located on the same network switch, it is reasonable to expect that the SLA algorithm experienced difficulty in converging onto a strategy. The large fluctuations in action probability curves in Fig. 7 lends credence to this theory, where all TTPs experience some early hesitation before finally settling on a strategy. In a production system, TTP compute capacities, run-time load and network latency are expected to be meaningfully different. Future analysis may attempt to replicate the TTPs as virtual machines on cloud platforms to simulate proximal and remote TTPs of differing compute capacities and network latencies.

Alice's selection process for the Manager TTP, accomplished in the first step of Layer 1, can leak information about her withdrawal patterns. The Hidden Markov Model nature of SLA engenders a degree of non-determinism within the system i.e. successive attempts with identical environmental conditions may see Alice select a different TTP as the

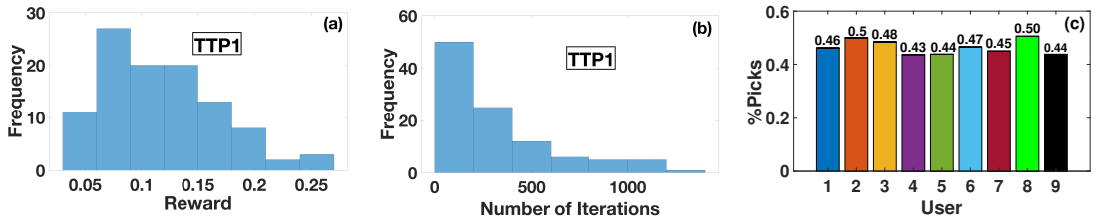

Fig. 8. Analysis of Layer 2 and Layer 0 Behaviour. (a) Layer 2 Reward histogram for TTP1's transaction history. (b) Layer 2 Iteration count histogram for TTP1's transaction history. (c) Non-deterministic behaviour of Layer 0

Manager. This claim was verified by conducting a sensitivity analysis on Layer 0. In the analysis, Alice's propensity to choose the Manager TTP was compared with the TTP receiving the maximum reward. In effect, we wanted to confirm how often SLA convergence at time t corresponded to the maximum reward at time t. Fig. 8c illustrates the results of the analysis. In the plot, the abscissa denotes the user ID and the ordinate denotes the probability that the Manager TTP chosen by SLA corresponds to the TTP with the maximum reward at time t. It can be observed that correlation is less than 51% in all cases, i.e. SLA does not always select the TTP with the maximum reward. Extended analysis indicates that SLA will just as likely select the second-place candidate and on rare occasions, the third-place candidate, with the uncertainty further strengthening Alice's privacy by making it difficult to predict or target any particular TTP as Alice's choice for Manager.

7.4 User Privacy

PUF-Cash is designed to create an N-privacy model, where Alice's spending habits look as similar as possible to Bob, Charlie and other users in the system. Privacy can be assessed by considering an attack scenario. The scenario under consideration is an all-seeing adversary with access to all TTP histories, where the attack is model-building, such that an omniscient adversary, with access to historical data on the fractional splits for every past transaction in the system can discern the withdrawal patterns of individual users even if a particular transaction does not have user or device identity attached to it. Note that none of the entities in the proposed protocol are solely capable of such an attack; however, since any TTP can act as the Manager TTP, each TTP will over time possess partial information on the network's spending activities, a close approximation to an omniscient adversary arises in cases where all TTPs decide to collude and pool their respective histories into a common shared database. Assuming TTPs are operated by commercial entities, such information pooling and exchange is not unheard of in the advertising, marketing and analytics domains.

To assess the likelihood of success of such an attack, we simulate an omniscient adversary. In the present experiment, this is achieved by leaking information from TTPs and tagging transactions with individual user device IDs. Each TTP acting as the Manager catalogs the withdrawal amount, the device ID, and the fractional splits assigned to the submitters for every transaction, noting that this information would not normally be logged in a production environment. Each experiment was run for 100,000 transactions, with each user transacting as quickly as possible to achieve an approximate 12,000 transactions per user per experiment. Individual user histories were extracted by parsing TTP logs and consolidating individual user histories across all TTPs. The privacy-preserving aspects of Layer 2 are statistically compared to the null hypotheses to demonstrate a relative improvement. The baseline to all aspects of Layer 2 is the absence of Layer 2, namely blinding the full withdrawal amount as per the original protocol. Testing the efficacy of the MFU strategy requires a local null hypothesis specific to Layer 2. Here we choose the random strategy. Instead of

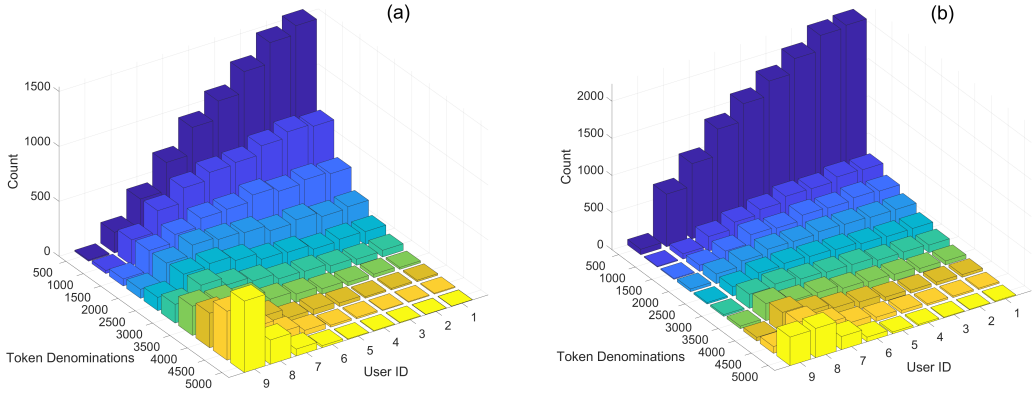

Fig. 9. User bin selection history extracted from experiments. (a) Results from the Random Strategy (b) Results from the MFU Strategy.

running SLA to determine a fractional split, Layer 2 simply selects a valid solution at random. Note that random can prove optimal from a differential privacy perspective, and this aspect will be challenged in the analysis.

Figure 9 illustrates the bin frequency selection per user for the random and preferred (MFU) strategies. Here, the x-axis is the value of the bin (or denomination), the y-axis the user ID and the z-axis the frequency of selection. It can be observed that User 9 with an average withdrawal amount of $160 experiences the greatest frequency with the 5000 token denomination. An overall skewing towards lower bin values can be observed from all users when switching from the Random strategy (Fig. 9a) to the MFU strategy (Fig. 9b). Each user's bin selection frequency represents a unique distribution and this distribution can be contrasted against other users to determine their similarity. Towards that end, a similarity matrix is generated for all users, where a single similarity measure is a pair-wise comparison between any two user histograms. The Bhattacharya similarity coefficient [4] was used as a similarity measure. Table 3 depicts an example similarity matrix obtained by computing the Bhattacharya coefficient against MFU values. Note that values are symmetric about the diagonal, as user similarity is unity in cases of self-comparison.

Once individual similarities are computed, the aggregate similarity per user can be measured by taking the mean of each row. This metric illustrates how similar a user is on average to all other users in the system. Fig. 10 illustrates the aggregate similarity per user, with the analysis repeated for the random and baseline strategy. Here, similarity values from the baseline strategy are computed on the unique withdrawal distributions from Fig. 5. Note that the ideal curve is denoted by a flat line at unity, which would suggest that users look identical from TTP data regardless of their individual withdrawal patterns.

In terms of the overall trend, the similarity scores per user in the baseline curve mirror the input probability density functions, with users at the lower end of the spectrum more similar to each other than users at the higher end. We can observe a dramatic improvement in user similarity when switching from the baseline with full withdrawal amounts to fixed denominations, regardless of strategy (random or MFU). This is a strong case for relying on fixed denomination bins as a privacy-preserving technique, and the analysis confirms that the bins provide the bulk of the privacy here, a full 25% improvement over the original protocol. We can also observe the small but consistent improvement offered by the MFU strategy over the random. Detailed analysis demonstrates an average improvement of 5% with a 7% reduction in the standard deviation, suggesting that users are consistently more similar in the MFU strategy as compared to

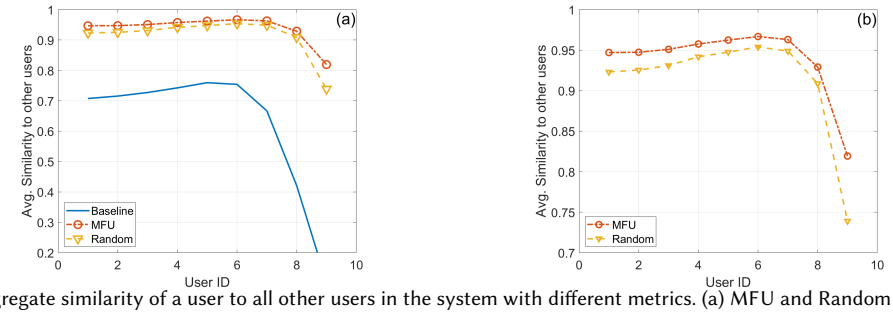

Fig. 10. Aggregate similarity of a user to all other users in the system with different metrics. (a) MFU and Random against Baseline (b) MFU and Random comparison.

Table 3. User Similarity Values - Preferred strategy, Bhattacharya similarity metric

User ID	1	2	3	4	5	6	7	8	9
1	1	0.99	0.99	0.99	0.99	0.98	0.94	0.86	0.63
2	0.99	1	0.99	0.99	0.99	0.98	0.94	0.86	0.64
3	0.99	0.99	1	0.99	0.99	0.98	0.95	0.87	0.65
4	0.99	0.99	0.99	1	0.99	0.99	0.96	0.90	0.69
5	0.99	0.99	0.99	0.99	1	0.99	0.97	0.91	0.72
6	0.98	0.98	0.98	0.99	0.99	1	0.99	0.94	0.77
7	0.94	0.94	0.948	0.96	0.97	0.99	1	0.98	0.86
8	0.86	0.86	0.87	0.90	0.91	0.94	0.98	1	0.94
9	0.63	0.64	0.65	0.69	0.72	0.77	0.86	0.94	1

random. The small difference between MFU and random strategies speaks to how effective the random strategy is at preserving privacy. Future work will explore other strategies that may be more effective than the MFU approach. Since the upper bound of 100% similarity was not achieved through either technique, there is room to consider more novel approaches in future work.

7.5 Timing Attack Analysis

Another possible attack is the Bank conducting a timing analysis on Alice's withdrawals. Since the Bank validates Alice's issued tokens to the Manager TTP, the Bank now knows exactly when Alice is ready to blind her tokens. Therefore, Alice is subject to a timing attack from the Bank. The Manager-Submitter TTP component added to the protocol is designed to increase the difficulty of the Bank time correlating n_2s to Alice's n_1s. We refer to the metric used by the Bank as its *guess probability*. The evaluation is carried out using data collected from several experimental configurations of the hardware test bed described in the Section 6. We ran a set of one hour experiments and several specialized five hour experiments to enable specific features of the protocol to be validated.

To determine the privacy improvements that the proposed protocol provides over the original, we configure our test bed to emulate the original by tasking only one TTP with carrying out the Exchange operation. In this scenario, the Bank delivers n_1 tokens to the TTP in step 2 and then the TTP returns blinded tokens in step 6 as shown in Fig. 1a. Privacy is provided by the random ordering of customer exchanges within the multi-threaded application that is running on the TTP. The guess probabilities computed using this configuration are compared with those computed using all five TTPs in the proposed version of the protocol. The former configuration is called **OneTTP** while the latter is called **FiveTTP**.

The guess probabilities are computed for both versions of the protocol by analyzing Bank activity. The Bank application is also multi-threaded, with a separate thread dedicated to processing each transaction. For example, the withdrawal by Alice, the n_1 validation by the Manager TTP and the n_2 transmissions by the Submitter TTPs to the Bank are handled by separate threads. Message exchanges between the TTP and Bank are augmented with additional information (beyond what would normally be available to the Bank) to make it possible to determine how often the Bank's guesses are correct. In particular, we return the customer IP address with each n_2 chunk transmitted by the Submitter TTPs to the Bank (step 6 in Fig. 1b) to enable the n_2 chunks to be unambiguously tied to the customer. We use the term *actual probability* in reference to this exact calculation.

The Bank's timing attack correlates the Manager TTP's n_1 validation operation with the Bank to the Bank's receipt of the n_2 \$5 chunks from the Submitter TTPs. First, the n_1 and n_2 transactions are partitioned into time slots. Each time slot is defined to be 1 second in duration for convenience, although microsecond resolution is used in computing the results. The time instant at which an exchange operation begins (step 3 in Fig. 1b) is recorded. The initial exchange operation is referred to as the *Manager-Exchange*, which occurs when Alice selects a Manager TTP, transmits her n_1s to the Manager TTP, and then the Manager TTP contacts the Bank to validate her n_1s. Her identity as well as the amount that she is exchanging are revealed to the Bank during this transaction, so the Bank knows that once it validates Alice's n_1s, the Manager TTP will task a set of Submitter TTPs to generate and return sets of n_2 chunks to the Bank that represent Alice's blinded tokens.

The bar graph in Fig. 11 gives an example of the data processed in our analysis. The bar graphs portray activity at the Bank in a series of 1 second snap shots, with each snap shot giving the number of n_2 chunks received by the Bank during that time interval. The left-most stacked bar labeled (1) gives the number of n_2 chunks received in that snap shot from each of the five TTPs. This information is available to the Bank to mount the time correlation attack. The remaining bars labeled (2) through (10) partition the total number of n_2 chunks given by the left bar across the nine customers. This information is leaked in our experiments during the Submitter TTP transfers, but would be absent in a production system.

A guess probability can be computed by the Bank for each n_2 chunk, where the probability represents the certainty the Bank has that the n_2 chuck actually belongs to Alice. The guess probability depends on the amount that Alice exchanges, the number of n_2 chunks it receives from all Submitter TTPs after the Manager-Exchange operation begins and the speed at which the Submitter TTPs respond to the Manager TTP requests to generate and transmit n_2 chunks to the Bank. An upper bound on the latter defines the time interval the Bank can use as a limit in its evaluation of guess probability. The worst case scenario is assumed, that is the case which maximizes the Bank's ability to correctly guess Alice's n_2s. For example, if Alice exchanges \$25, which produces 5 n_2 chunks, the Bank's guess probability using data from the top snap shot in Fig. 11 would be computed as 5/18 = 27.8% (the height of the left-most bar is 18). Therefore, any one of the 18 n_2 chunks has a 27.8% chance of belonging to Alice. The Bank cannot leverage the individual TTP values to improve this guess because it is unaware of which Submitter TTPs were selected by the Manager TTP and how the Manager chose to distribute the 5 n_2 chunks to the Submitters. It can however improve its guess probability by noting that the Manager TTP does not return any of Alice's n_2s according to the protocol. Therefore, the n_2 chunks from one of the TTPs can be eliminated, e.g., if TTP_2 is the Manager, than the denominator becomes 15 and the guess probability increases to 33%. Note that the protocol can be easily modified to reduce the effectiveness of the Bank's guesses. One possible approach is to have the Manager TTP randomly insert small delays before issuing requests to the Submitter TTPs to generate n_2 chunks.

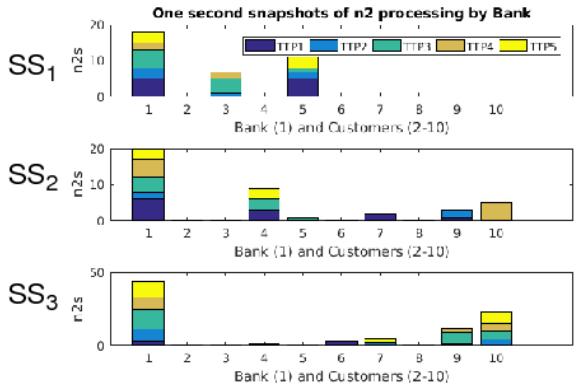

Fig. 11. Time snapshots (SS_1 - SS_3) of the Bank's n_2 processing behavior.

Unfortunately for the Bank, the guess probabilities are actually worse because the Submitter TTPs cannot immediately return the n_2 chunks because of processing and network congestion. Therefore the Bank must extend its analysis over multiple time snap shots. The Bank can upper bound the number of one second time intervals that it needs to consider if it has knowledge of the worst case *Manager-Exchange completion time*, i.e., the time interval beginning with a Manager-Exchange operation and ending at the instant the last of the Submitter TTP n_2 transfers takes place. For example, if Alice's Manager-Exchange begins within the first snap shot of Fig. 11 and the last of the 5 n_2 chunks arrive at the Bank in the last time snapshot, then the denominator increases to 69 (assuming again TTP_2 is the Manager) and the Bank's guess probability drops to 5/69 = 7.2%. The Bank can improve this somewhat by using exact times, as opposed to rounding off to 1 second time intervals (we report our results assuming the Bank uses exact times).

The Manager-Exchange completion time has a large impact on the accuracy of the guess probabilities, and varies widely during the protocol execution. The completion times, in seconds, are plotted in Fig. 12a using data from experiments under the OneTTP and FiveTTP test-bed configurations. To determine how the number of customers impacts the results, we add one customer after every 30 minutes of run time (the dotted vertical lines delineate the regions with different numbers of customers). For example, the left-most region labeled 'customer1' plots transaction times with only 1 customer application running while the region labeled '+ customer2' shows the transaction times when two customers applications are running, and so on. The total number of completion times plotted along the x-axis are 22,377 and 19,402 under the OneTTP and FiveTTP configurations, respectively.

The average transaction time and variability in the curves associated with the OneTTP configuration increase noticeably as customers are added, while changes occur only gradually under the FiveTTP configuration. The right-most values associated with the average curves are approximately 9 seconds and 3.7 seconds, respectively. Also, the worst case transaction time for the OneTTP experiment is 38 seconds, which reduces to less than 12 seconds for the FiveTTP experiment. This clearly reflects the performance benefit of using multiple TTPs. The transaction rate at which Submitter TTPs transmit n_2 chunks to the Bank also has a large impact on the guess probabilities, with larger rates adversely impacting the guess probabilities. Table 4 gives the rates per TTP associated with the two configurations, measured in the region with all nine customers engaged. The rates are lower for the FiveTTP configuration because the customers are not able to generate withdrawal and exchange requests fast enough to keep the TTPs fully engaged. The ratio of 4.06/9 = 0.45 is used later to normalize the guess probabilities to enable an apples-to-apples comparison.

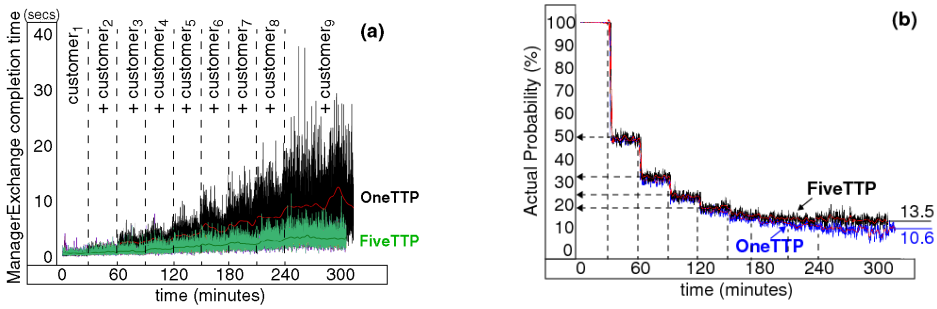

Fig. 12. Comparison of exchange completion times and probabilities between configurations. (a) OneTTP and FiveTTP experiment ManagerExchange completion times, and (b) actual probabilities computed using an 8 time slot window.

Table 4. Transaction rates per second per TTP for OneTTP and FiveTTP experiments.

	OneTTP	FiveTTPs					
TTP #	TTP_1	TTP_1	TTP_2	TTP_3	TTP_4	TTP_5	Avg.
Trans. rate	8.84	4.43	4.01	4.14	3.97	3.77	4.06

Similarly, the actual probabilities represent the best the Bank can achieve with guessing. They are computed as discussed by counting the number of n_2 chunks returned by the Submitter TTPs divided by the total number of n_2 chunks returned within the time window for each Manager-Exchange operation. In this case, we do not eliminate the n_2 chunks returned by the Manager TTP. The actual probabilities plotted in Fig. 12b use a window of 4 seconds and 9 seconds for the OneTTP and FiveTTP configurations respectively, which corresponds to the average completion time from Fig. 12a. The expected probabilities are given by $1/N$, where N is the number of end-users, which is reflected fairly well in the curves of Fig. 12b. The probabilities after all nine customer applications are running are given on the right side in the figure as 13.5% and 10.6% respectively, well within the statistical noise limits of the expected value of 1/9 = 11.1%. Thus, the proposed protocol does deliver $1/N$ privacy.

Our goal is to show the level of improvement in privacy that is possible using the FiveTTP configuration over the base-case OneTTP configuration. The additional uncertainty created for the Bank in the FiveTTP configuration derives from two sources: (a) the Bank does not know which Submitter TTPs were selected by the Manager TTP, and (b) multiple TTPs return a larger volume of n_2 tokens per time slot than a single TTP, which increases the number of customers that the Bank must differentiate between. We use data from the 1 hour experiments in this analysis. Similar to the 5 hour experiments, start times are staggered, but by only 30 seconds in these experiments. Therefore, all 9 customers are running after 240 seconds have elapsed. Data is depicted for only the first 300 seconds of the 1800 seconds in Figs. 13a and 13b because the curves remain invariant after this time interval. The curves in Fig. 13a show the Bank's guess and actual probabilities for the OneTTP configuration for several different time window sizes, namely 3, 9 and 25. The Bank's guess probabilities are large at approximately 50% when a window size of 3 is used, which is good for the Bank. However, the certainty the Bank has on associating n_2 chunks to Alice's n_1s is false because the actual probabilities are near zero for this window size. The near zero actual probabilities are explained by the Manager-Exchange completion times on the right side of Fig. 12a, where it takes approximately 9 seconds on average for the TTP to transmit Alice's n_2 chunks after the Bank receives the exchange request. Thus, most of the n_2 chunks are simply not present in the set the Bank is guessing on.

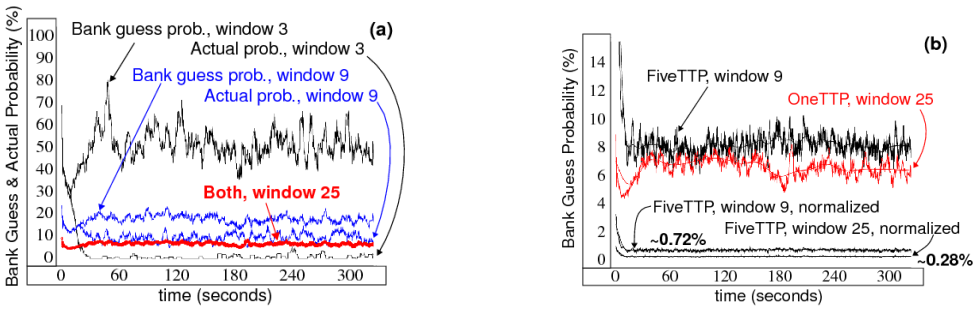

Fig. 13. Sensitivity analysis and comparison of guess probabilities. (a) Sensitivity analysis of OneTTP guess probabilities against window size. (b) Comparison of guess probabilities for OneTTP and FiveTTP configurations.

Increasing the window size to 9 reduces the Bank's guess probability significantly, from approximately 50% to 20%. Although the actual probabilities are non-zero now, the Bank's certainty is still false in approximately 50% of the cases. Increasing the window size to 25 makes the Bank's guess probabilities nearly equal to the actual probabilities. This is true because the larger window size now includes more than 99% of the completion times, as shown in Fig. 12a. The Bank's average guess probability is reduced again to approximately 6.5% with the window size set to 25, but the guesses now correspond to possible n_2 candidates for Alice's n_1s.

A similar approach to deciding the proper window size can be applied to data collected from the FiveTTP configuration. From Fig. 12a, a window size of 9 includes more than 99% of the completion times. The curve labeled "FiveTTP, window 9" in Fig. 13b plots the Bank's guess probabilities using this window size. The average value is approximately 8% (when evaluated at 300 seconds), and is in fact worse than the OneTTP value of approximately 6.5%. We mentioned earlier that the TTPs transmit at a slower rate under the FiveTTP configuration, and indicated that the rates need to be equal for an apples-to-apples comparison. The rate ratio relating FiveTTP to OneTTP is 0.45, which indicates that each of the 5 TTPs in FiveTTP is processing at a rate of less than one half the rate of the single TTP in OneTTP. The curve labeled "FiveTTP, window 9, normalized" represents the original curve multiplied by 0.09, which effectively increases the volume of n_2 tokens by a factor of 11. The Bank's guess probability is now reduced to approximately 0.72%, which when compared with the 6.5% for the OneTTP configuration reflects the improvement in privacy. Also note that the FiveTTP configuration reduces the turn-around time for Alice by nearly 3X, from 25 seconds to 9 seconds. In cases where turn-around time is not critical, a countermeasure that randomly delays the Manager TTP's requests to the Submitter TTPs can be inserted. For example, if the inserted randomized delays are set such that the Bank's time window increases to the value used in the OneTTP configuration at 25 seconds, the Bank's guess probabilities decrease again to approximately 0.28%, as shown by bottom curve in Fig. 13b, which enhances privacy even further.

8 CONCLUSIONS AND FUTURE WORK

This paper proposes an electronic cash protocol termed PUF-Cash that leverages physical unclonable functions, multiple trusted third party entities and a reinforcement learning algorithm to create a novel e-Cash token blinding scheme that adds significant levels of uncertainty for adversaries and system authorities attempting to track customer spending behaviors. In addition to protecting customer privacy, the blinding strategy is secure against threats from adversaries, including theft, impersonation and fraud. In addition to preserving privacy, the introduction of multiple TTPs increases system throughput and resiliency. The proposed reinforcement learning algorithms utilize past processing histories of the TTPs as a mechanism to balance workloads, and in a unique twist, preserve user privacy. A privacy analysis

using data collected from an experimental prototype shows that the combination of multiple TTPs and reinforcement learning improves customer anonymity under several attack scenarios. In one scenario, multiple TTPs collude to learn customer spending behavior, while in another scenario, the Bank attempts to correlate blinded e-Cash tokens with specific customers. The results demonstrate that the proposed method provides N-privacy, which is the maximum level of privacy achievable in a system with N customers.

ACKNOWLEDGMENTS

The research of Mr. Fragkos and Dr. Tsiropoulou was conducted as part of the NSF CRII-1849739.

REFERENCES

[1] J. Aarestad, P. Ortiz, D. Acharyya, and J. Plusquellic. 2013. HELP: A Hardware-Embedded Delay PUF. *IEEE Design and Test* 30, 2 (April 2013), 17–25.
[2] Scott Aaronson. 2009. Quantum Copy-Protection and Quantum Money. In *Proc. IEEE Conf. on Comput. Complexity*. https://doi.org/10.1109/ccc.2009.42
[3] Scott Aaronson and Paul Christiano. 2012. Quantum Money from Hidden Subspaces. In *Proc. ACM Symposium on Theory of Computing* (New York, New York, USA). 41–60. https://doi.org/10.1145/2213977.2213983
[4] A. Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. 35 (1943), 99–109.
[5] Stefan Brands. 1994. Untraceable Off-line Cash in Wallet with Observers. In *Advances in Cryptology - CRYPTO' 93*. Springer, 302–318.
[6] Jeff Calhoun, Cyrus Minwalla, Charles Helmich, Fareena Saqib, Wenjie Che, and Jim Plusquellic. 2019. Physical Unclonable Function (PUF)-Based e-Cash Transaction Protocol (PUF-Cash). *Cryptography* 3, Article 3 (2019), 21 pages.
[7] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. 2005. Compact E-Cash. In *Advances in Cryptology - EUROCRYPT 2005*. Springer, 302–321.
[8] Jan Camenisch, Anna Lysyanskaya, and Mira Belenkiy. 2007. Endorsed E-Cash. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society.
[9] David Chaum. 1983. Blind Signatures for Untraceable Payments. In *Advances in Cryptology*. Springer US, Boston, MA, 199–203.
[10] David Chaum, Amos Fiat, and Moni Naor. 1990. Untraceable Electronic Cash. In *Advances in Cryptology*, Shafi Goldwasser (Ed.). Springer, 319–327.
[11] Wenjie Che, Venkata K. Kajuluri, Mitchell Martin, Fareena Saqib, and Jim Plusquellic. 2017. Analysis of Entropy in a Hardware-Embedded Delay PUF. *Cryptography* 1, 1 (2017). https://doi.org/10.3390/cryptography1010008
[12] Wenjie Che, Venkata K. Kajuluri, Fareena Saqib, and Jim Plusquellic. 2017. Leveraging Distributions in Physical Unclonable Functions. *Cryptography* 1, 3 (2017).
[13] Wenjie Che, Mitchell Martin, Goutham Pocklassery, Venkata K. Kajuluri, Fareena Saqib, and Jim Plusquellic. 2016. A Privacy-Preserving, Mutual PUF-Based Authentication Protocol. *Cryptography* 1, 1, Article 3 (2016), 17 pages. https://doi.org/10.3390/cryptography1010003
[14] W. Che, M. Martinez-Ramon, F. Saqib, and J. Plusquellic. 2018. Delay model and machine learning exploration of a hardware-embedded delay PUF. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 153–158.
[15] Wenjie Che, Fareena Saqib, and Jim Plusquellic. 2017. Novel Offset Techniques for Improving Bitstring Quality of a Hardware-Embedded Delay PUF. *Transactions on VLSI* 26, Article 4 (April 2017), 733-743 pages.
[16] Digilent Corporation 2018. *Cora Z7 Reference Manual*. Digilent Corporation, Pullman, WA. https://media.digikey.com/pdf/Data%20Sheets/Digilent%20PDFs/Cora_Z7_RM_Web.pdf
[17] G. Fragkos, C. Minwalla, J. Plusquellic, and E. E. Tsiropoulou. 2020. Artificially Intelligent Electronic Money. *IEEE Consumer Electronics Magazine* (2020), 1–1. https://doi.org/10.1109/MCE.2020.3024512
[18] G. Fragkos, C. Minwalla, J. Plusquellic, and E. E. Tsiropoulou. 2020. Reinforcement Learning Toward Decision-Making for Multiple Trusted-Third-Parties in PUF-Cash. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. 1–6. https://doi.org/10.1109/WF-IoT48130.2020.9221344
[19] Phil Mars. 2018. *Learning algorithms: theory and applications in signal processing, control and communications*. CRC Press, 530 Walnut St suite 850, Philadelphia, PA 19106, United States.
[20] Tatsuaki Okamoto. 1995. An Efficient Divisible Electronic Cash Scheme. In *Advances in Cryptology - CRYPT0' 95*. Springer, 438–451.
[21] Jim Plusquellic and Matt Areno. 2018. Correlation-Based Robust Authentication (Cobra) Using Helper Data Only. *Cryptography* 2, 3 (2018). https://doi.org/10.3390/cryptography2030021
[22] Markus Rückert. 2010. Lattice-Based Blind Signatures. In *Advances in Cryptology - ASIACRYPT 2010*, Masayuki Abe (Ed.). Springer, 413–430.
[23] Berry Schoenmakers. 1998. *Security Aspects of the Ecash™ Payment System*. Springer, 338–352.
[24] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (Jan 1999), 303–332.
[25] National Cash Systems. 2019. *ATM Statistics*. Retrieved June 30, 2020 from http://www.nationalcash.com/statistics/
[26] Yuhua Xu, Jinlong Wang, and Qihui Wu. 2016. Distributed learning of equilibria with incomplete, dynamic, and uncertain information in wireless communication networks. In *Game Theory Framework Applied to Wireless Com. Net*. IGI Global, 63–86.