MIDEP: Multiparty Identity Establishment Protocol for Decentralized Collaborative Services

Rasib Khan and Ragib Hasan
SECRETLab, Department of Computer and Information Sciences
University of Alabama at Birmingham, AL, USA
{rasib, ragib}@cis.uab.edu

Abstract—Decentralized collaborative architectures are gaining popularity in all application areas, varying from peer-to-peer communication and content management to cloud and ubiquitous services. However, the public identity of the user is still a major concern, in terms of privacy, traceability, verifiability, masquerading, and other attacks in such environments. We demonstrate two new attacks, identity shadowing and the Man-in-the-Loop (MITL) attacks, which are applicable in particular to multiparty collaborative environments. In this paper, we propose MIDEP, a Multiparty IDentity Establishment Protocol for collaborative environments. The proposed protocol allows a client to establish a secure, multiparty, probabilistic, temporal, verifiable, and non-traceable public identity with the collaborating peers in a decentralized architecture. MIDEP allows a client to avoid identity shadowing and protects the service from the resulting threats as well as from colluded information sharing among the collaborating peers. We illustrate how existing collaborative service frameworks can utilize MIDEP to securely establish the public identity prior to beginning the service session. A prototype implementation is utilized to perform extensive experimental analysis. Our results show that MIDEP is highly suitable in terms of overhead to ensure secure identity establishment for underlying decentralized collaborative services.

Keywords—Collaborative, Decentralized, Multiparty, Identity Establishment, Security, Temporal, Non-Traceable, MIDEP

I. INTRODUCTION

Decentralized collaborative architectures require multiple entities to interact and offer a collective service. A user interacts with multiple entities using the same public identity. In this work, we consider the public identity of the user as a unique representation using a username, signature, token, service identity, or a shared (public) key among the distributed and decentralized entities providing the service. Unfortunately, the use of the same identity across multiple service sites exposes the user to numerous threats from within the service network [1–3]. Decentralized service architectures involving more than two parties expose the user to session misdirection [4–7]. A malicious peer may perform masquerading attacks and session misdirection [4, 7]. An attacker may also track the user and violate the privacy of non-linkability [1, 8]. Collaborative systems do not allow users to validate the number of parties involved in the multiparty protocol, exposing the user to session hijacks and replay attacks [9, 10]. Decentralized services may employ cloaking or anonymous identities using k-tokens [11]. Unfortunately, anonymous identities challenge the provability or auditing of the users with respect to the growing number of tokens over time [12–14].

The user’s service in collaborative systems (Figure 1) suffers from misdirection and colluded information sharing among the peers to exploit the identity of the user. We also introduce the concept of identity shadowing and man-in-the-loop (MITL) attacks, which are applicable to decentralized collaborative architectures involving more than two parties. Let us consider a 3-party protocol among A, B, and C, with each entity interacting with the other two. Each node assumes that its peers are reachable over a direct link or via the third node. In this context, we say that A suffers from identity shadowing for C on the indirect link $A-B-C$ (Figure 2a). The identity shadow for the user can easily allow a malicious party to perform an MITL attack by intercepting the shadowed link (Figure 2b). We also emphasize the requirement of privacy of non-linkability as well as verifiability of the user’s identity for such service architectures. The focus of the problem is therefore not the authentication of anonymous users, but the establishment of a secure interactive public identity of the user at a given decentralized collaborative service site.

In this paper, we introduce MIDEP, a secure Multiparty IDentity Establishment Protocol. MIDEP allows a user to establish a commonly agreed probabilistic public identity from a set of available options. We posit that MIDEP is suitable for decentralized collaborative systems for establishing temporal, non-traceable, and provable/auditable identities for its users. MIDEP can be used as an overlying layer with other collaborative service architectures to ensure protection from linkable identities, identity shadowing, masquerading, and colluded information sharing among the collaborating peers. MIDEP also allows an authorized auditing entity to validate the claim of usage of a particular service by a user at a later time. Our simulated results illustrate the practical feasibility of MIDEP in terms of the computational and temporal overhead for underlying decentralized collaborative services.

Contributions: The contributions in this paper are as follows:

1) We introduce the concepts of identity shadowing and Man-in-the-Loop (MITL) attacks particular to decentralized collaborative services and present a list of desired security properties to be preserved in such environments.

2) We propose the Multiparty IDentity Establishment Protocol (MIDEP) for secure establishment of a user’s public identity for decentralized collaborative services.

3) We illustrate the suitability of MIDEP for collaborative services using a fully-functional prototype and extensive experimental analysis for the proposed protocol.

The rest of the paper is organized as follows. Section II ex-
II. SECURITY CHALLENGES

In this section, we discuss the possible threats and the desired security properties for collaborative services.

A. Motivation

Peer-to-peer (P2P) and deterministic distributed systems (e.g., DHT) have been applied to numerous applications [4, 7, 15, 16]. Various services have been ported to P2P to leverage the collaborative nature of the framework [4–7, 17]. Ad-hoc cloud frameworks can also utilize decentralized and collaborative entities within the architecture [18, 19]. Ubiquitous environments are designed around decentralized architectures for providing collaborative services [8, 9]. Service composition using collaborative smart devices is the core concept behind pervasive and localized applications [20, 21]. Location-based applications also require the collaboration of decentralized entities to provide the service [10, 22–24]. Unfortunately, such systems suffer from various shortcomings [25].

An example collaborative environment is illustrated in Figure 1. A user \( U \) requests a decentralized collaborative service from a set of peers \( P \). The user communicates with peer \( P_i \), and one of the peers, \( P_m \), is malicious. The public identity/key of the user may be distributed from a centralized point. This imposes a severe bottleneck in the performance of the system. Moreover, \( U \)’s public identity can be traced to other service sites, violating the privacy of non-linkability. Anonymous authentication techniques allow a probable solution at the cost of limited scalability and storage [13, 14, 26]. The user may consider sending the public key individually to all the cooperating peers. However, collaborative services require the peers to communicate among themselves. This introduces identity shadowing for \( U \) and can be exploited by \( P_m \) to perform forging, masquerading, replay, or MITL attacks. Therefore, we posit that an overlying protocol is required for decentralized service-oriented architectures [8, 19, 27] to establish a secure, commonly agreed, probabilistic, non-linkable, and auditable public identity for the users.

B. Threat Model

In the context of this paper, we refer to the public identity as a unique representation of the user based on a username, signature, token, service identity, or a shared public key among the collaborating peers, \( P \) and one of the peers, \( P_m \), and can be exploited by \( P_m \) to perform forging, masquerading, replay, or MITL attacks.

II. SECURITY CHALLENGES

In this section, we discuss the possible threats and the desired security properties for collaborative services.

A. Motivation

Peer-to-peer (P2P) and deterministic distributed systems (e.g., DHT) have been applied to numerous applications [4, 7, 15, 16]. Various services have been ported to P2P to leverage the collaborative nature of the framework [4–7, 17]. Ad-hoc cloud frameworks can also utilize decentralized and collaborative entities within the architecture [18, 19]. Ubiquitous environments are designed around decentralized architectures for providing collaborative services [8, 9]. Service composition using collaborative smart devices is the core concept behind pervasive and localized applications [20, 21]. Location-based applications also require the collaboration of decentralized entities to provide the service [10, 22–24]. Unfortunately, such systems suffer from various shortcomings [25].

An example collaborative environment is illustrated in Figure 1. A user \( U \) requests a decentralized collaborative service from a set of peers \( P \). The user communicates with peer \( P_i \), and one of the peers, \( P_m \), is malicious. The public identity/key of the user may be distributed from a centralized point. This imposes a severe bottleneck in the performance of the system. Moreover, \( U \)’s public identity can be traced to other service sites, violating the privacy of non-linkability. Anonymous authentication techniques allow a probable solution at the cost of limited scalability and storage [13, 14, 26]. The user may consider sending the public key individually to all the cooperating peers. However, collaborative services require the peers to communicate among themselves. This introduces identity shadowing for \( U \) and can be exploited by \( P_m \) to perform forging, masquerading, replay, or MITL attacks. Therefore, we posit that an overlying protocol is required for decentralized service-oriented architectures [8, 19, 27] to establish a secure, commonly agreed, probabilistic, non-linkable, and auditable public identity for the users.

B. Threat Model

In the context of this paper, we refer to the public identity as a unique representation of the user based on a username, signature, token, service identity, or a shared public key among the collaborating peers, \( P \) and one of the peers, \( P_m \), and can be exploited by \( P_m \) to perform forging, masquerading, replay, or MITL attacks.

II. SECURITY CHALLENGES

In this section, we discuss the possible threats and the desired security properties for collaborative services.

A. Motivation

Peer-to-peer (P2P) and deterministic distributed systems (e.g., DHT) have been applied to numerous applications [4, 7, 15, 16]. Various services have been ported to P2P to leverage the collaborative nature of the framework [4–7, 17]. Ad-hoc cloud frameworks can also utilize decentralized and collaborative entities within the architecture [18, 19]. Ubiquitous environments are designed around decentralized architectures for providing collaborative services [8, 9]. Service composition using collaborative smart devices is the core concept behind pervasive and localized applications [20, 21]. Location-based applications also require the collaboration of decentralized entities to provide the service [10, 22–24]. Unfortunately, such systems suffer from various shortcomings [25].

An example collaborative environment is illustrated in Figure 1. A user \( U \) requests a decentralized collaborative service from a set of peers \( P \). The user communicates with peer \( P_i \), and one of the peers, \( P_m \), is malicious. The public identity/key of the user may be distributed from a centralized point. This imposes a severe bottleneck in the performance of the system. Moreover, \( U \)’s public identity can be traced to other service sites, violating the privacy of non-linkability. Anonymous authentication techniques allow a probable solution at the cost of limited scalability and storage [13, 14, 26]. The user may consider sending the public key individually to all the cooperating peers. However, collaborative services require the peers to communicate among themselves. This introduces identity shadowing for \( U \) and can be exploited by \( P_m \) to perform forging, masquerading, replay, or MITL attacks. Therefore, we posit that an overlying protocol is required for decentralized service-oriented architectures [8, 19, 27] to establish a secure, commonly agreed, probabilistic, non-linkable, and auditable public identity for the users.

B. Threat Model

In the context of this paper, we refer to the public identity as a unique representation of the user based on a username, signature, token, service identity, or a shared public key among the collaborating peers, \( P \) and one of the peers, \( P_m \), and can be exploited by \( P_m \) to perform forging, masquerading, replay, or MITL attacks.

II. SECURITY CHALLENGES

In this section, we discuss the possible threats and the desired security properties for collaborative services.

A. Motivation

Peer-to-peer (P2P) and deterministic distributed systems (e.g., DHT) have been applied to numerous applications [4, 7, 15, 16]. Various services have been ported to P2P to leverage the collaborative nature of the framework [4–7, 17]. Ad-hoc cloud frameworks can also utilize decentralized and collaborative entities within the architecture [18, 19]. Ubiquitous environments are designed around decentralized architectures for providing collaborative services [8, 9]. Service composition using collaborative smart devices is the core concept behind pervasive and localized applications [20, 21]. Location-based applications also require the collaboration of decentralized entities to provide the service [10, 22–24]. Unfortunately, such systems suffer from various shortcomings [25].

An example collaborative environment is illustrated in Figure 1. A user \( U \) requests a decentralized collaborative service from a set of peers \( P \). The user communicates with peer \( P_i \), and one of the peers, \( P_m \), is malicious. The public identity/key of the user may be distributed from a centralized point. This imposes a severe bottleneck in the performance of the system. Moreover, \( U \)’s public identity can be traced to other service sites, violating the privacy of non-linkability. Anonymous authentication techniques allow a probable solution at the cost of limited scalability and storage [13, 14, 26]. The user may consider sending the public key individually to all the cooperating peers. However, collaborative services require the peers to communicate among themselves. This introduces identity shadowing for \( U \) and can be exploited by \( P_m \) to perform forging, masquerading, replay, or MITL attacks. Therefore, we posit that an overlying protocol is required for decentralized service-oriented architectures [8, 19, 27] to establish a secure, commonly agreed, probabilistic, non-linkable, and auditable public identity for the users.

B. Threat Model

In the context of this paper, we refer to the public identity as a unique representation of the user based on a username, signature, token, service identity, or a shared public key among the collaborating peers, \( P \) and one of the peers, \( P_m \), and can be exploited by \( P_m \) to perform forging, masquerading, replay, or MITL attacks.
III. MULTI-PARTY IDENTITY ESTABLISHMENT

In this section, we present the system model and protocol description for MIDEF.

A. System Model

The Multi-party Identity Establishment Protocol (MIDEF) is a supporting protocol for decentralized service-oriented architectures which involve more than two parties collaborating to offer a particular service at a given site at a given time. The operational model assumes that all parties involved in the service are capable of reaching each other over the network. MIDEF focuses on the agreement, establishment, and post-verification of a common temporal identity at the given site of service for a decentralized collaborative system. MIDEF does not include the process of authentication or authorization of the services. The user is assumed to possess a unique identifier (userID), based on which, the subsequent temporal identities (tempID) are generated.

MIDEF operates using the following entities: (a) a client is a user who is interested to establish a temporal, provable, and non-traceable identity among the entities in a decentralized collaborative service, (b) collaborators are the entities which are collectively providing the service, (c) for each of the collaborators, the other collaborators are referred to as peers, and (d) an auditor is an authorized entity who can verify the usage of the servicing identity for the client. A client can use MIDEF and userID to establish a commonly agreed temporal identity among the collaborators. The collaborators also communicate with their peers to ensure resistance to the common attacks on the user’s identity in decentralized collaborative services. MIDEF can work with existing collaborative services to provide secure identity establishment before beginning a session. Service frameworks can request for a temporal identity using MIDEF as an overlay protocol, as shown in Figure 3.

B. Identity Establishment

The protocol sequence for MIDEF is illustrated in Figure 4 and described as follows.

(Step a) Key-Part Generation: The client possesses a unique userID and uses the Shamir’s Secret Sharing algorithm to divide userID into \( N \)-pieces of IDparts [32]. At least \( K \)-pieces of IDparts are required to reconstruct userID.

(Step b) Send Key-Parts: The client then sends the identity offer message, with \( M \) random pieces of IDparts to each of the collaborators. Here, the value of \( M \) is such that \( (M \times X_C < K) \), where \( X_C \) is the number of collaborators and \( K \) is the minimum number of IDparts required to reconstruct userID [32]. The operation performed by the client is described in Algorithm 1. The client first obtains the list of all collaborators. The IDOffer message is then sent to each of the collaborators and includes \( M \) random IDparts. The message is constructed with the format described in Algorithm 2.

(Step c) Exchange Key-Parts: The \( M \)-pieces received by each collaborator \( X \) may or may not have some common IDparts. Each of the collaborators then exchange \( E \)-pieces of IDparts which they have received from the client, where \( (E \leq M) \). Each of the collaborators \( X \) executes the exchange sequence as described in Algorithm 3. Each collaborator obtains \( E \) random IDparts and sends the IDExch message to the corresponding peers, which is constructed using Algorithm 2.

(Step d) Response Key-Parts: Each collaborator exchanges the \( E \) IDparts with their peers and saves the newly received items to create a set of \( R \) IDparts, where \( R = (M + (E \times (X - 1))) \). The collaborators then respond to the client with the identity offer response message. The sequence of functions for the collaborator is presented in Algorithm 4. Each collaborator dispatches a looped process to receive any exchange parts.
Step f) Temporal Identity Confirmation: The client sends an identity finalized message to all the collaborators to notify the tempID and the duration of time for which the tempID is valid. In case the client wishes to continue the service session at the given site, the client may re-run MIDEP to try again.

Step e) Exchange Verification: The R IDparts received by the client from each of the collaborators are matched with the original N IDparts to verify a valid exchange between each of the collaborators. A successful verification is followed by the construction of the tempID. The tempID is considered to be T pieces of IDparts, where \( 1 \leq T \leq S \), and S is the number of shared common IDparts among all the collaborators. The client executes the sequence of actions described in Algorithm 7. A mismatch implies that there is a malicious collaborator attempting an MITL or peer misdirection attack taking advantage of the identity shadowing. After all of the collaborator responses have been processed, the commonly shared IDparts are obtained from the map. If N is large and M is too small, execution of MIDEP may result in no common IDparts. The client can then restart MIDEP to re-try the identity establishment process.

Algorithm 3  FUNC: BeginExchange(IDOfferMsg)

```
1: Comment: begins identity exchange process
2: Begin:
3: M = IDOfferMsg.Size()  
4: IDOffers[M] = GetIDparts(IDOfferMsg)  
5: for all P in Peers do  
6:   EIDparts[E] = GetRandParts(IDOffers [, E)  
7:   MTtype = “IDExch”  
8:   PartExchMsg = CreateMsg(MType, EIDparts)  
9:   SendMsg(P, PartExchMsg)
10: end for
```

Algorithm 4  FUNC: MIDEPCollaborator()

```
1: Comment: initiates collaborator
2: Begin:
3: Thread: ExchangeReceiverLoop()  
4: IDOfferMsg = ReceiveMessage()  
5: BeginExchange(IDOfferMsg)
6: SendOfferResponse()  
```

Algorithm 5  FUNC: ExchangeReceiverLoop()

```
1: Comment: starts loop to receive IDparts sent from peers
2: Begin:
3: while !AllPeersExchangeCompleted() do
4:   ExchangeReceivedParts = ReceiveMessage()
5:   CurrMap.add(ExchangeReceivedParts)
6:   end while
```

Algorithm 6  FUNC: SendOfferResponse()

```
1: Comment: sends offer response to MIDEPCollaborator
2: Begin:
3: if AllPeersExchangeCompleted() then
4:   MTtype = “IDOfferResp”  
5:   IDOfferRespMsg = CreateMsg(MType, IDpartMap)  
6:   SendMsg(MIDEPClient, IDOfferRespMsg)
7:   end if
```

Algorithm 7  FUNC: VerifyExchange()

```
1: Comment: verifies exchange and returns TempID[]
2: Return format: Size:Parts[ ]Size(), Parts[part1...partT]
3: Begin:
4: IDpartMap = CreateMap(IDparts[ ])
5: for all X in Collaborator do
6:   IDOfferRespMsg = ReceiveMessage()
7:   R = IDOfferRespMsg.Size()  
8:   RIDparts[R] = GetIDparts(IDOfferRespMsg)
9:   if !check if IDparts belong to user X
10:   isValid = Validate(RIDparts[ ])
11:   if isValid then
12:      for all P in RIDparts[] do
13:         InsertToMap(P, IDpartMap)
14:      end for
15:   else
16:      RestartMIDEP()
17:      end if
18:      end for
19:      if P obtain commonly shared IDparts *)
20:      SharedIDparts[] = CheckMap(IDpartMap)
21:      if SharedIDparts[] Size() == 0 then
22:         RestartMIDEP()
23:      else
24:         TempID[] = GetRandParts(SharedIDparts[ ], T)
25:      end if
26:      return TempID[]
```

Algorithm 8  FUNC: IDFinalizer()

```
1: Comment: begins identity finalize process
2: Begin:
3: TempID[] = VerifyExchange()
4: MTtype = “TempIDFinalized”
5: TempIDMsg = CreateMsg(MType, TempID[])
6: for all X in Collaborators do
7:   SendMsg(X, TempIDMsg)
8: end for
```

is explained in Algorithm 8. The finalizer process on the client is responsible for generating and sending the final tempID. At this point, MIDEP has securely established a common public identity with the particular peers who are expected to collaborate within the decentralized service framework. Any party, without receiving TempIDFinalized, should not be able to interact with the system and process the client’s request for the collaborative service.

Step g) Identity Acknowledgment: After the collaborators receive the identity finalized message, each of them respond with an acknowledgement message to the client. The protocol sequence for MIDEF therefore finishes at this point.

C. Identity Verification

A client may be required to present a proof of usage for a particular service to a certified auditor. The client presents random \((K − 1)\) IDparts, the particular tempID, and the userID to the auditor. The auditor uses the specific tempID and \((K − 1)\) IDparts, and uses Shamir’s Secret Sharing to reconstruct the client’s original userID [32]. A successful match verifies that the client had indeed used the decentralized service along with the collaborators at the given site. This verifiable identity also ensures that the decentralized service architecture is not susceptible to denial of usage (DoU) by the client.

IV. Security Analysis

In this section, we present the design and security analysis for MIDEF in a decentralized collaborative environment.

Key Division Parameters: The client is required to preset the values of \(N\) and \(K\). High values for \(N\) and \(K\), with \(K \approx N\),
will allow the client to have a wider choice for the $M$ random IDparts and will protect the client from colluded information sharing attacks. Property P4 is thus satisfied.

**Offer Size Adjustment:** The value of $K$ and $M$ should be such that $K \gg (M \times X_C)$. Keeping the number of offered pieces low ensures minimum revelation of the IDparts. Properties P3 and P5 are thus satisfied.

**Provable Identity:** The client places $P$ protected pieces of IDparts in a secure storage, where $P \geq (N-K+1)$ and never uses $P$ for any of the phases in MIDEP. The client presents $K$ IDparts to prove the identity in case of DoU. Therefore, $|P| \geq 1$ ensures that there will be at least one piece required for a successful reconstruction of $userID$ and protects the client from all-way colluded information sharing. Properties P4, P5, and P6 are thus satisfied.

**Shared Secret Index Obfuscation:** Reconstruction of $userID$ requires $N$, $K$, and the index of each of the $K$ IDparts. The IDparts, $N$, $K$, and the index of the IDparts are all private in MIDEP. Therefore, an attacker still needs to try $\binom{N}{N-K} \times R^L$ number of combinations to guess the $userID$ with known values for $N$ and $K$. Here, $R$ is the representation format (binary, alpha-numeric), and $L$ is the length of the key. E.g., for $N = K = 15$ and $K \gg (M \times X_C)$, there are more than $(1.30 \times 10^{12})$ possible polynomial combinations. Property P3 and P5 are thus enhanced.

**Avoiding Identity Shadow:** Unlike MIDEP, mutually verified protocols will still suffer from identity shadowing, given that the collaborators do not agree on the identity during the process of identity establishment. Exchange verification from the set of exchanged IDparts implies that each of the collaborators and the client are communicating with the same set of collaborators and peers, and therefore avoids the identity shadow. Properties P1, P2, and P3 are thus satisfied.

**Outsider Collaborator Protection:** The client offers the IDparts to each of the collaborators with probability: $P(offer) = \frac{(N-N)}{N!}$. The (trusted) collaborators exchange the IDparts with probability $P(exchange) = \frac{(N-E)}{M!}$. E.g., for $N = 15$, $M = 3$, and $E = 2$, $P(offer)$ and $P(exchange)$ are $(3.66 \times 10^{-4} \times R^{-L})$ and $(1.67 \times 10^{-1} \times R^{-L})$ respectively. The combined probability $(P(offer) \cap P(exchange))$ is $(6.11 \times 10^{-5} \times R^{-2L})$. An attacker needs to overcome these odds to guess the IDparts corresponding to a valid collaborator. Properties P1, P2, and P3 are thus enhanced.

**Privacy of Non-Linkability:** The tempID allows an entity to interact with a system without revealing the $userID$ identity. Additionally, the tempID may be re-set at any time the client desires. These features allow protection against replay attacks and linkable identities. Property P4 is thus satisfied.

**Enhancement of Non-Linkability:** The client can use progressive/degressive IDpart generation using different values of $N$ and $K$. This ensures a greater probabilistic protection of non-linkable privacy. The client can use $(N = K)$ and $(|P| \geq 1)$ for $Min \leq N \leq Max$ where $Min$ depends on $(1 + X_C)$ and $Max$ depends on the computational feasibility of the auditor. Given $P_n = (P(offer) \cap P(exchange))$ for each $(N = K)$, the probability of guessing a traceable identity is $(P(offer) \cap P(exchange))$.

### V. Prototype Implementation

In this section, we present the architecture and implementation details for the MIDEP prototype.

**MIDEP Client:** The client had been implemented both as a PC (Java) and an Android application. The system architecture for the implementation is shown in Figure 5a. The offer handler creates the IDparts using an open source Shamir’s Secret Share library [33]. The key store is a placeholder for all the IDparts. The communication handler sends/receives messages to/from the collaborators. The collaborator observer keeps track of the collaborators and their status. The verification engine is responsible for asserting the exchange process and invokes the identity finalizer to confirm the tempID. The service interface is the client’s internal communication channel for an underlying collaborative service framework which would use the tempID as the identity of the client.

**MIDEP Collaborator:** The collaborator had been implemented as a Java application. The implementation architecture is shown in Figure 5b. The communication handler receives/sends messages from/to the client and the peers. The exchange handler executes the exchange process with the peers. The peer observer and key store is used to keep track of the incoming/outgoing IDparts. The verification engine confirms the tempID, sends back the acknowledgement, and stores the tempID on the client ID store. The service interface is the collaborator’s internal hook to communicate with the underlying service framework to support the establishment of a temporal identity for a client.

### VI. Experiments and Results

In this section, we present the experimental evaluation for the MIDEP prototype implementation.

**A. Experimental Setup**

We utilized the prototype implementation to perform experimental measurements and performance analysis. The PC client was running on a dual core Intel Q9550 2.83 GHz machine with 4 GB RAM and Ubuntu operating system. The Android client was running on an ASUS Nexus 7 tablet with 1 GB RAM and Quad-core 1.2 GHz Cortex-A9 processor. The
collaborators were running on Model-B Raspberry Pi-s with 512 MB of RAM, 700 MHz Low Power ARM1176JZ-F processor, and 100 MB Ethernet port. The Raspberry Pi MIDEP collaborators were connected over Ethernet to a Buffalo WZR-1750DHP Air Station wireless router. The PC and Android MIDEP clients were connected to the network over WiFi. We also implemented a PC and an Android identity verification tool to be used by an auditor for user ID verification.

B. Key Generation

We measured the time taken for IDpart generation for both the PC and Android MIDEP clients. The times were recorded for 100 iterations for each value of \( N \) and \( K \), where \( N = K \in [5 \ldots 15] \), and are illustrated in Figure 6a and 6b. The average times for each of the cases are summarized in Table I. The PC was more powerful than the Nexus 7 tablet, and maintained a linear trend with increasing values of \( N \) and \( K \). The PC client, for most cases, took less than 5 ms for generating the IDparts. The range of average times lied between 1.473 ms and 0.904 ms. The Android client was slower in computation and took longer time with higher degrees of the polynomial. The computation was still around 40 ms for most cases, with the average range between 32.731 ms and 5.640 ms. The results for both cases show that the IDpart generation introduces a minimal overhead.

C. Key Combination

The key combination for IDparts is required for identity verification by an authorized MIDEP auditor. We implemented an auditing tool for identity verification for both PC and Android. The times were recorded for 100 iterations for each value of \( K \), where \( K \in [5 \ldots 15] \), given that the IDparts were generated with \( N = K \). The measured times are illustrated in Figure 7a and Figure 7b. The average times for each of the cases are summarized in Table I. The key combination operation requires the Lagrange basis polynomial computation for the given degree of \( K \) and leads to a high utilization of system memory. The general trend for the PC application was linear till the value of \( K = 13 \), being within 15 ms, with an average of 5.569 ms, after which the required time increased to 29.461 ms. The Android verification tool showed a much more evident exponential pattern. The average time required till \( K = 10 \) was less than 127.703 ms, which grew rapidly to an average of 689.413 ms for \( K = 15 \). However, the verification tool will most likely be running on a PC with an authorized auditor, in which case, the performance will barely effect the usability of the system.

D. Protocol Execution

We created a total of 20 protocol simulation scenarios with varying number of Raspberry Pi collaborators (\( 2 \leq X_C \leq 5 \)), offer sizes (\( 2 \leq M \leq 5 \)), and exchange sizes (\( 1 \leq E \leq 5 \)). Each of the 20 scenarios were executed for 100 iterations, for both the PC and Android MIDEP clients. We measured the time taken for the clients to establish a tempID using MIDEP, that is, from generating the IDparts till the receipt of the acknowledgment from the collaborators. The simulation summary is presented in Table II. The measured times for the clients are illustrated in Figure 8 and Figure 9.

For both the clients, we observed an increase in time with increasing number of collaborators. The offer size (\( M \)) and exchange size (\( E \)) did not have much effect on the required
The average times for the PC client varied between 61.79 ms and 81.01 ms for 2 collaborators (cases 1 to 4). With 5 collaborators (cases 15 to 20), the average times varied between 108.21 ms and 155.38 ms. Given the limited resources on the tablet compared to the PC, the Android client had a higher required time. The average times for the Android client was between 135.73 ms and 185.11 ms for 2 collaborators (cases 1 to 4), and between 207.61 ms and 282.25 ms for 5 collaborators (cases 15 to 20). The wide variety of simulation scenarios show that MIDEP is suitable for collaborative service frameworks for establishing the temporal identity for both desktop and mobile applications.

### E. Successful Identity Establishment

We recorded the number of common IDparts (SharedIDparts) obtained from the collaborating exchange. Having no common IDparts implies that it was a failed attempt to establish a MIDEP identity. Table II summarizes the percentage of successful MIDEP identity establishments. Figure 10a and Figure 10b illustrate the average and range of shared IDparts for each of the simulation scenarios.

We observed that M and E had varying effects with the number of collaborators ($X_C$) on the success rate. The success rate for both the PC and Android MIDEP had similar results. For cases 1 to 4 ($X_C = 2$), the success rate was at 100%.

### TABLE II: Simulated MIDEP Execution Results

<table>
<thead>
<tr>
<th>Case</th>
<th>Size</th>
<th>Collab (X_C)</th>
<th>PC Client</th>
<th>Android Client</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(M)</td>
<td>(E)</td>
<td>Avg. Time (ms)</td>
<td>Success (%)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1, 1</td>
<td>69.49</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1, 1</td>
<td>63.11</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1, 1</td>
<td>61.79</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3, 3</td>
<td>81.01</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1, 1, 1</td>
<td>81.05</td>
<td>96</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1, 1, 1</td>
<td>90.67</td>
<td>82</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>1, 1, 1</td>
<td>86.57</td>
<td>78</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>3, 3, 3</td>
<td>102.73</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>1, 1, 1</td>
<td>108.08</td>
<td>83</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>1, 1, 1</td>
<td>93.87</td>
<td>78</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>2, 2, 2</td>
<td>115.40</td>
<td>100</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>2, 2, 2</td>
<td>98.51</td>
<td>100</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>4, 4, 4</td>
<td>135.35</td>
<td>100</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>1, 2, 3, 4</td>
<td>123.71</td>
<td>100</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>1, 1, 1, 1</td>
<td>125.39</td>
<td>82</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>1, 1, 1, 1</td>
<td>108.21</td>
<td>70</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>2, 2, 2</td>
<td>136.29</td>
<td>100</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>2, 2, 2, 2</td>
<td>118.81</td>
<td>100</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>4, 4, 4, 4</td>
<td>155.38</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>1, 2, 3, 4, 5</td>
<td>148.68</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 10: Case-wise Comparison of Shared IDparts

(a) PC MIDEP Client
(b) Android MIDEP Client

Figure 11: ROC Curves for Collaborator Size ($X_C$) Prediction

Case[5 . . . 20], the success rate was lower with smaller values of $M$ and $E$. Increasing the value of $M$ with a fixed value for $E$ reduced the success rate. An increase in the value of $E \geq 2$ drastically improved the success rate for all cases.

Cases 1 to 3 ($M \in [2, 3, 5], E = 1, X_C = 2$) had two SharedIDparts on average. Case 4 ($M = 5, E = 3, X_C = 2$) resulted in a greater number of SharedIDparts with an average between 5 and 6. Cases 5 to 7 ($M \in [2, 3, 4], E = 1$, $X_C = 3$) had decreasing success rate. Case 8 ($M = 4, E = 3, X_C = 3$) had 100% success rate, with an average of 6 SharedIDparts. Case 14 ($M = 5, E \in [1 . . . 4]$, $X_C = 4$) and case 20 ($M = 5, E \in [1 . . . 5]$, $X_C = 5$) had different values of $E$ for the each collaborator, and still had a success rate of 100%, with 5 and 8 SharedIDparts on average. The results show that the success rate is directly proportional to $M, E$, and $X_C$.

### F. Collaborator Prediction

Given that $M$ and $S$ are known to the client, we analyzed the suitability of MIDEP to identify unwanted collaborators using our simulation results. We created a nominal logistic model to predict the number of collaborators ($X_C$) using $M, S,$ and $M*S$ as input variables. Figure 11 illustrates the receiver operation characteristic (ROC) curve for the sensitivity (true positive, TP) versus specificity (false positive, FP) of the model. The summary of the model is presented in Table III. The model performed similar for both clients ($p < 0.0001$) with $M$ and $M*S$ being primary indicators ($p < 0.0001$). The TP rate decreased with increasing $X_C$ for the same number of cases. $X_C = 2$ and $X_C = 3$ had the highest enclosed ROC area and the best prediction results. The ROC area and TP rate reduced for $X_C = 4$ and $X_C = 5$. However, the area is still well above the marginal break-point. The results for $X_C = 2$ and $X_C = 3$ compared to $X_C = 4$ and $X_C = 5$ depicts the necessity for a larger number of cases with increasing collaborators.

We believe that MIDEP can be applied for further machine
TABLE III: ROC and TP for Collaborator Size Prediction

<table>
<thead>
<tr>
<th>Midep Client</th>
<th>Collaborator Size (ROC Area, TP %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$X_C=2$</td>
</tr>
<tr>
<td></td>
<td>$X_C=3$</td>
</tr>
<tr>
<td></td>
<td>$X_C=4$</td>
</tr>
<tr>
<td></td>
<td>$X_C=5$</td>
</tr>
<tr>
<td>PC  ($p &lt; 0.0001$)</td>
<td>0.9002, 0.9253, 0.7149, 0.7775, 63.48</td>
</tr>
<tr>
<td>Android ($p &lt; 0.0001$)</td>
<td>0.9029, 0.9178, 0.7222, 0.7892, 58.03</td>
</tr>
</tbody>
</table>

learning techniques for adaptive security by predicting the number of peers within a collaborative system.

VII. RELATED WORK

Traditional key agreement protocols, unlike MIDEP, are not suitable for decentralized collaborative environments [34]. Token-based approaches for anonymity may provide a suitable solution for collaborative frameworks [12, 14]. Unfortunately, such techniques are based on a centralized servers and have high costs for storage and scalability [11, 13, 26]. Access control in collaborative systems is also a crucial security component [35]. Most solutions depend on centralized and federated systems for key-agreement and access control [36, 37]. Conversely, MIDEH does not require centralized services for establishment and verification of the public identity. Additionally, MIDEH allows flexible configurations as well as post-verification by authorized auditors. Biometrics, such as keystroke dynamics, have also been proposed for collaborative systems [38]. However, such systems are only suitable for human-interactive systems, suffer from varying precision, and require previous data to perform the verification. MIDEH focuses on decentralized temporal identities, does not require prior interactions, and ensures protection against the threats emerging from identity shadowing in collaborative systems.

VIII. CONCLUSION

Decentralized collaborative environments have evolved greatly and been applied to various architectures and domains. In this paper, we proposed a Multiparty IDentity Establishment Protocol (MIDEH) for allowing a user to securely establish a public identity among multiple collaborating entities in a decentralized framework. The proposed protocol allows a temporal, non-linkable, verifiable, and probabilistic identity for a servicing user, and protects an underlying decentralized service framework from identity exploitation attacks by malicious collaborating peers. Our simulations show that MIDEH can be easily integrated with minimal overhead as an overlay protocol with decentralized frameworks to establish a public identity for collaborative services. Our future work includes adaptation of MIDEH on top of existing decentralized collaborative frameworks for ubiquitous and localized services.

ACKNOWLEDGMENT

This research was supported by a Google Faculty Research Award, the Department of Homeland Security Grant FA8750-12-2-0254, and by the National Science Foundation CAREER Award CNS-1351038.

REFERENCES