Dynamic Mobile Anonymity with Mixing

Emin Islam Tatlı, Dirk Stegemann, Stefan Lucks
{tatli,stegemann,lucks}@th.informatik.uni-mannheim.de

Department of Computer Science, University of Mannheim

Abstract. Staying anonymous and not revealing real identity is highly desired in today's mobile business. Especially generic frameworks for different kinds of context-aware mobile business applications should provide communication anonymity of mobile users as a core security feature. For enabling communication anonymity, Mix-net based solutions are widely accepted and used. But directly deploying existing Mix-net clients on mobile devices with limited hardware capacity is not a realistic approach. In addition, different anonymity sensitivities of both applications and users require to enforce anonymity dynamically rather than on a fixed level. In this paper, we present an approach towards a solution that addresses the specific anonymity challenges in mobile business while exploiting the benefits of existing Mix-net frameworks.

1 Motivation

Hiding real identity is inevitable for people who do not like to share their personal information or secrets with others. They do not want others to get to know their meeting schedules with their business partners, which books they buy and read, how much money they have in their bank accounts, which transactions they execute with their credit cards, etc. We can give many more examples. In short, people highly prefer staying anonymous as far as and whenever possible. In the internet age, the number of interactions people have with their environment (i.e. their business partners, companies, public service organizations, etc.) has enormously increased. At the same time, the anonymity requirement remains, although it has become more difficult to meet. Staying anonymous when sending e-mails, visiting web pages, and doing e-commerce is needed but not easy to achieve.

In this paper, we consider the scenario of a generic framework for context-aware mobile business (m-business) applications in which mobile client devices like mobile phones, PDAs, etc., are used for business transactions. Mobile business services are often organized in service-oriented architectures [7] with three main principals. *Service providers* offer paid or free services to their customers. *Mobile users* carry PDAs or other capable mobile devices and are interested in receiving the services offered by the service providers. A *broker* registers the services of service providers and returns the service descriptions to mobile users upon request. The broker plays the role of a trusted third party from the business point of view.

Analyses of the security requirements in mobile business environments indicate that anonymity and protection of the mobile user's personal data is one of the most challenging risks [18, 5]. Although security and anonymity are non-functional properties of a system, its user acceptance directly depends on these features.

We distinguish two types of anonymity – *content anonymity* and *communication anonymity* – which both have to be fulfilled in order to provide complete

anonymity. Pseudonyms can provide content anonymity by keeping the users' real identities secret, but an attacker who is able to sniff incoming and outgoing messages on the network nodes can at least find out which nodes communicate with each other. This type of threat to anonymity can be averted by ensuring unlinkability of user actions [14].

In this paper, we study communication anonymity rather than content anonymity. We will therefore refer to communication anonymity simply by anonymity in the remainder of the paper.

The paper is organized as follows. Section 2 explains the types of existing solutions for communication anonymity. The new anonymity challenges in mobile business frameworks and the suitability of existing solutions in this context are discussed in Sect. 3. Section 4 presents our architecture with a threat model for providing anonymity, and Sect. 5 discusses the pros and cons of our solution approach. Possible future work is explained in Sect. 6, and Sect. 7 concludes the paper.

2 Existing Solutions for Anonymity

In the literature, existing solutions for communication anonymity and unlinkability of user actions are categorized into three groups: *proxies*, *peer-to-peer (P2P) networks* and *Mix-net* [11].

In a proxy-based solution, a trusted proxy (anonymizer) receives user requests, rewrites some parts of the request in order to hide sender-specific data and sends it to the final receiver. Replies from the receiver are in turn forwarded to the real sender. The drawback of this scheme is that users have to trust the proxy and there are no protection mechanisms in the channel between users and proxies. For example, *www.anonymizer.com* is a well-known proxy for anonymous web surfing.

With the increasing popularity of peer-to-peer applications, communication anonymity solutions based on P2P networks have been designed [16, 8]. Unlike anonymizer proxies, there is no need for a trusted party within these systems. Each user shares an encrypted secure channel with other users in the P2P network. Initially, the user chooses a random path (a user group) and sends the message along this path to the final receiver. In mobile Internet communication, mCrowds [1] presents an anonymity solution for P2P networks.

The last type of solution is Mix-net, which is a more promising approach for the m-business framework compared to proxies and P2P networks. It was first suggested by Chaum for anonymous e-mail communication [4]. A *mix* is a computer which resides between a sender and a receiver. When a mix gets a message, it decrypts it and forwards the remaining part to the next mix or the final receiver. A group of mixes composes a network called Mix-net. Chaum's traditional Mix-net was based on public key operations, but today, Mix-net based solutions relying on symmetric encryption also exist.

Mix-net based solutions are well accepted in academia and have also been designed and deployed for different application scenarios. For example, there are solutions for anonymous communication over ISDN networks (*ISDN-Mixes* [15]) and for anonymous email communication (*smtp-remailers* [13]). Jap [10] and Tor [6] are recent Mix-net implementations focussing on anonymous web surfing. Their anonymity service is based on the SOCKS protocol [12] and can cover any application layer protocol (e.g. ftp, p2p, http, https, etc.). However, Jap does not support

protocols other than HTTP due to some organizational reasons and functions as a HTTP proxy.

Jap has a cascade-style Mix-net, whereas Tor supports free nodes. A cascade of Jap consists of two or three mix nodes in a fixed sequential order. The user can only choose the cascade but no particular mix nodes. Supporting free nodes, Tor allows to choose arbitrary paths through mix nodes.

In Jap, users start the client application and choose a cascade satisfying their anonymity requirement. Each cascade offers different levels of anonymity based on the number of active users on the chosen cascade and traffic parameters. As of September 2005, there were 5 official Jap cascades installed. The default Jap cascade Dresden-Dresden, which is managed by the developers of Jap, has around 1400-1500 simultaneously active users on average [17].

In Tor, since the message route among free mix nodes is chosen randomly, a higher level of anonymity can be achieved compared to the fixed mix order of cascades. On the other hand, dummy messages and time delays for higher anonymity are only supported in Jap, but not in Tor.

3 New Anonymity Challenges

One could integrate Jap or Tor client applications into an m-business framework and enable mobile users to communicate anonymously via Jap or Tor networks. But the m-business scenario yields several specific challenges, with *limited capabilities* of mobile PDAs and the requirement of a *dynamic anonymity* (see Section 3.2) rather than anonymity at a fixed level being the most important ones.

3.1 Limited Hardware Capabilities

In today's Mix-networks, the sender is required to encrypt a message with the symmetric key of each mix in the message route before sending. Therefore, a key handshake process should be executed between the sender and each mix in the route. These encryption and key handshake processes are very heavy and time consuming operations that mobile PDAs cannot put up with. For illustration, Table 1 shows the performance of the required cryptographic operations[1]. The tests were done on both a Zaurus SL-C3000 PDA (416 MHz CPU/64 MB RAM) and an IBM Thinkpad R51 notebook (1.7 GHz CPU/1 GB RAM). For the implementation, we used Bouncy Castle leightweight cryptographic APIs [3]. Note that 100% of CPU power was used during the test computations. Thus, even if we decrease the priority of the cryptographic operations, other applications running on the PDA will hardly be useable while Mix-net clients are executed.

3.2 Dynamic Anonymity

Both Jap and Tor provide a fixed level of anonymity. You start your client application with a particular configuration, and you cannot easily change or manage your anonymity level afterwards. But the m-business framework requires an anonymity feature that enables dynamic updates of anonymity levels for the following reasons:

[1] Not all operations in Table 1 are required for Mix-net clients, but for the sake of completeness, we have also included the execution times for RSA key generation (normally performed offline), decryption and digital signing.

Table 1. Performance of Cryptographic Operations

Operation	**Time Consumption on**	
	Zaurus SL-C3000 (416 MHz)	**IBM Thinkpad R51** (1.7 GHz)
RSA Key Generation *(1024-bit key)*	122 seconds	2.2 seconds
RSA Encryption *(1024-bit key, 64-byte data)*	172 ms	10 ms
RSA Decryption *(1024-bit key,128-byte data)*	856 ms	40 ms
RSA Signing *(1024-bit key,64-bytes data)*	833 ms	55 ms
RSA Verification *(1024-bit key, 128-bytes data)*	169 ms	5 ms
AES Encryption/Decryption *(128-bit key, 2048-byte data)*	583 ms	35 ms
SHA-1 Hash *(2048-byte data)*	111 ms	5 ms

- *Varying sensitivity of applications:* In the m-business framework, the anonymity requirements of applications may totally differ. Consider *finding the nearest restaurant* and *mobile dating* applications. *Finding nearest restaurant* is a typical context-aware application. Holding your PDA, you are interested in getting a list of restaurants which are near to your current location. The second application type, *context-aware mobile dating*, lets users search for suitable chat partners within a particular area. Comparing these two types of applications, the latter requires (at least initially) a very high level of anonymity, while the former may not even need anonymity at all.
- *Varying sensitivity of users:* Users also tend to have different sensitivity for anonymity. Consider a celebrity interested in having a very high level of anonymity. He can even require a high level of anonymity for the *finding the nearest restaurant* application and never wants other people to know the places that he eats at.
- *Enhancing performance:* The previous two requirements point out that enforcing a fixed level of anonymity is a security risk since the anonymity level may be too low. On the other hand, an unnecessary high anonymity level makes applications waste time waiting for cryptographic operations and data transmission delays.

Dynamic Anonymity Parameters We have analyzed different Mix-net architectures and found 7 parameters that affect the anonymity level. These parameters and their effects are as follows:

1. *Path picker:* In the Mix-net, initially a message route should be decided. This parameter specifies the entity that decides this message route. *user* and *firstmix* are possible values of this parameter. *user* implies that the sender itself will choose the message route. In this case, the anonymity level is high, but the fact that the user should encrypt the message for each mix decreases performance. *firstmix* offers better performance, because the user does not decide

the message route and encrypts the message only for the first mix which then chooses the path and encrypts the message accordingly. But at the same time, the anonymity level decreases since the first mix knows both the sender and the receiver.

2. *Mix number:* This parameter specifies the exact number of the mixes that will be used for the transmission of messages from a particular application. If there are more mixes in the message route, traffic analysis becomes more complicated and the anonymity level increases. Setting this parameter value to zero results in a direct communication without any Mix-net nodes in between.
3. *User number:* In a cascade-style Mix-net, the number of active users on a particular cascade affects the anonymity level. This parameter defines a minimum and maximum number of users that should exist on a cascade.
4. *New route*: This parameter specifies a threshold for the number of connections over an established circuit. When this threshold is reached, a new route should be created in order to make traffic analysis more difficult.
5. *Message threshold:* Upon getting a message, the mix can either forward the message to the next mix immediately or keep it in its outgoing pool until a certain condition is fulfilled. This parameter as a condition defines the number of messages that should exist in the outgoing pool before a message from the pool is randomly chosen and forwarded. With this option enabled, traffic analysis becomes more difficult, but also additional delays and/or latency may be created.
6. *Time delay:* This parameter is similar to *message threshold*, but defines the delay threshold for a message to stay in the outgoing pool, rather than the number of messages to collect before a send operation.
7. *Dummy message:* With this parameter enabled, each mix sends extra dummy messages to other mixes when transmitting a regular message in order to complicate traffic analysis.

4 Towards A Solution

In this section, we propose an anonymity solution that considers both hardware limitations and dynamic anonymity requirements. It supports dynamic anonymity at a sufficient degree, provides reasonable performance and exploits the existing Mix-net solutions.

In general, attackers capture the messages over the network, link the collected messages to each other and try to guess who communicates with whom. We will not be able to prevent attackers from intercepting data packets, but we can complicate the analysis of the gathered data by ensuring the unlinkability of exchanged messages. This unlinkability can be achieved to a reasonable degree by the mix-net based anonymity architecture that we describe in the following.

4.1 The Architecture

Based on our preliminary tests (see Table 1), we conclude that the usability of the m-business framework would be inacceptable if the mobile clients had to deal with all the relevant cryptographic operations. In our architecture as illustrated in Figure 1, mobile clients communicate with service providers through an external Security Provider performing the heavy part of the computations, while mobile

users can still specify the anonymity level for each application individually. This provides the possibility of running applications with very high anonymity. It becomes even possible to eliminate latencies due to anonymity computations only by modifying the relevant policies. The *Security Provider* residing between mobile devices and the Mix-network acts as a facade [9] and runs different types of Mix-net client applications to forward messages over any client, e.g. Jap-client, Tor-client, etc.

Application Manager, *Policy Manager* and *Anonymity Manager* are three main components in the architecture on the mobile client side. The policies created for each individual application specify some parameters affecting the anonymity level. They are stored within the repository of the Policy Manager. When an application wants to send a message, its request is taken by the Application Manager, which forwards this request to the Anonymity Manager. Afterwards, the Anonymity Manager applies to the Policy Manager to retrieve the anonymity policy of this particular application. Based on the parameters in the policy, the Application Manager decides how to proceed with the message transmission.

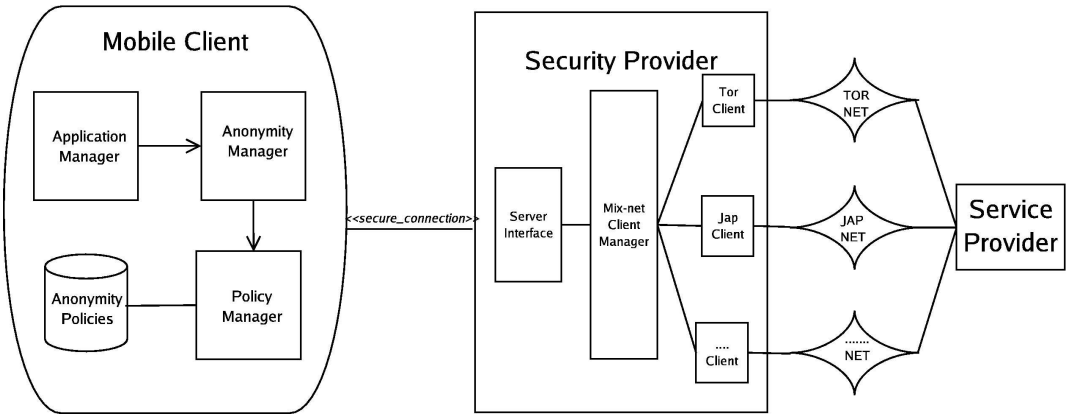

Fig. 1. Anonymity Framework based on Security Provider

Mobile users need to establish only *two* secure channels, i.e. a channel between mobile user and security provider and a channel between mobile user and service provider. The secure channel between mobile user and service providers is established through the secure channel between mobile user and security provider. The required dynamic anonymity based on policies is enforced by running different Mix-net clients managed by the Security Provider.

The Security Provider consists of three main layers. In the first layer, a *server interface* listens for incoming requests, parses the relevant *destination* and *anonymity parameters*, and forwards to the second layer, i.e., to the *Mix-net Client Manager (MCM)*. In the third layer, different Mix-net clients are installed and executed.

MCM is responsible for managing different Mix-net client applications. The idea behind installing and managing more than one Mix-net client application is that dynamic anonymity can be distributed throughout several network types in order to achieve higher anonymity. Besides, MCM can run different instances of the

same Mix-net client with *different* configuration parameters and thereby enhance the dynamic anonymity property. On receiving the payload data, destination and anonymity parameters, MCM chooses a suitable Mix-net client to send the message according to the policy parameters of the mobile user.

In Section 3.2, the required parameters for dynamic anonymity were presented. Both Jap and Tor support some of these parameters. In a Jap network, one can choose a cascade based on active user numbers (*user number* parameter). In a Tor-network, one can configure minimum and maximum node numbers (*mix number* parameter). Tor also supports creating a new circuit after sending a particular number of messages over an established channel (*new route* parameter). The *dummy message* and *time threshold* parameters are supported by the Jap network but cannot be changed on the client side. Therefore, the Security Provider emulates their effects as follows. It starts dummy Mix-net clients, and if the mobile user sets a value for the dummy message parameter, dummy messages are sent over this dummy Mix-net client. If the time threshold parameter is set, the Security Provider keeps the message for a particular time and then sends it over Mix-net clients. Applying time thresholds, timing attacks against exchanged messages are prevented. Additionally, as an enhancement to dynamic reconfigurability, the mobile user can also specify within his policy which Mix-net client to use.

Figure 2 illustrates templates encoding the configuration parameters of general Mix-net and Mix-net clients. Figure 3 illustrates the policies of applications and their bindings to particular templates.

We have implemented both the gateway and mobile client applications of this architecture and tested the client software on a PDA. The gateway was encoded in J2SE and the mobile client in J2ME with CDC configuration. The Security Provider was installed on the Broker-side. The performance of our implementation is sufficient for not decreasing the usability of the main application while providing a sufficient level of anomymity. We refer the reader to [2] for more details on the implementation such as particular design choices and the communication protocol between the mobile client and the gateway.

4.2 Security Analysis

As presented in the previous subsection, the client connects to the gateway and opens a channel through a mix network to the service provider he wants to communicate with. When the client sends a message to the gateway, the gateway encrypts the message according to the used Mix-net and sends it through the Mix-net to the service provider. We assume the goal of an attacker is to learn who communicates with whom by using traffic analysis. In mix networks, attackers can be categorized as follows.

- *passive or active:* Passive attackers are able to observe the messages sent on the network. Particularly, they can see messages sent from client to gateway, messages between nodes of the mix network, and between the exit nodes of the Mix-net and the service providers. Active attackers can additionally insert, modify, or delete messages in the network.
- *external or internal:* An internal attacker can control one or more mix nodes and/or the gateway. He is able to distinguish dummy messages and actual data messages and can link incoming and outgoing messages of the nodes he controls. An external attacker, on the other hand, cannot gain control over

Fig. 2. Templates for Anonymity Policy

```
<templates>
<template id="template_1">
  <general>
        <parameter name="dummyMessage">5</parameter>
        <parameter name="timeDelay">10</parameter>
        <parameter name="messageThreshold">5</parameter>
        <parameter name="preferredMix">jap</parameter>
  </general>
  <provider name="jap">
        <parameter name="minUser">50</parameter>
        <parameter name="maxUser">250</parameter>
  </provider>
</template>
<template id="template_2">
  <general>
        <parameter name="preferredMix">tor</parameter>
  </general>
  <provider name="tor">
        <parameter name="minNode">5</parameter>
        <parameter name="maxNode">15</parameter>
        <parameter name="createNewRoute">yes</parameter>
  </provider>
</template>
</templates>
```

any node in the architecture. We note that an attacker controlling the gateway would be fatal in our architecture, since the gateway knows the communication partners of the clients.

– *partial or global:* A partial attacker has influence only on a part of the system, while a global attacker can potentially attack the whole system.

An attacker is commonly assumed to be global, passive, and external. However, since the Mix-nets that underly our architecture are not secure against such attackers, our system is not, either. Additionally, a compromised gateway would completely expose the identity of all users running ther traffic through it. However, users are reasonably protected against active, internal and partial attackers who control only small number of mix nodes, not including the gateway.

The intentions of attackers may vary. A service provider may be interested in the identities (i.e., e-mail and postal addresses, etc.) of his competitor's customers in order to alienate them with specific advertisements. For his own customers, he may want to personalize mailings based on data collected by user profiling. Another possible attacker is a black user who tries to find out the providers that his target user is communicating with.

Fig. 3. Application Policy

```
<policies>
        <policy id="p1" belongsto="app_1" anonLevel="template_1" />
        <policy id="p2" belongsto="app_2" anonLevel="template_2" />
</policies>
```

5 Discussion

The presented approach is very realistic and applicable. As extra computations, the mobile devices have to handle only two secure channels. Besides, dynamic anonymity property has been realized at a presumably satisfying degree.

Another advantage of this architecture is that it promotes weak coupling between the mobile devices and particular Mix-net implementations, such that existing and well-established Mix-networks can be integrated in a way that is transparent for mobile devices. Especially when support of an additional Mix-net implementation is to be added to the framework, the client component must only be deployed to the Security Provider but not to mobile devices. From both security and software engineering points of view, this is a very desirable property.

As a drawback, the proposed architecture requires a trusted anonymizer. In case the Security Provider is compromised, anonymity fails. However, we note that the Broker, which already acts as a trusted third party from the business point of view, can be used as a trusted anonymizer. Hence, our solution does not necessarily introduce an additional trusted party.

6 Future Work

Having implemented this architecture, the next step is to study how and at which degree the configuration parameters affect the anonymity level. This is required for the optimal specification of the anonymity templates.

Policies and their configuration parameters provide dynamic anonymity. But it is neither realistic nor practical to expect non-technical mobile users to specify their own policies and parameters for each application. This process should be as easy as possible. For example, users could choose from pre-defined anonymity levels, e.g. ranging from *high* to *low*, for individual applications or the whole m-business client running on the device. The templates for different anonymity levels could be defined based on empirical tests and user interviews.

7 Conclusion

Within a mobile business framework, hiding real identities of mobile users is an important requirement. Dynamic anonymity and the limited capabilities of mobile devices are two new challenges which the existing Mix-net solutions for communication anonymity do not take into consideration.

In this paper, we present an approach towards dynamic mobile anonymity over a Security Provider gateway managing different mix-net clients. Our approach

overcomes the performance problem and satisfies dynamic anonymity at some degree by preserving the benefits of existing Mix-net framework and client applications.

8 Acknowledgment

The work described in this paper has been supported by the "Landesstiftung Baden-Württemberg" and the Ministry of Science, Research and Arts of the state of Baden-Württemberg.

We would like to thank Dominic Schmoigl and Christian Beil for valuable comments and discussions during the architecture design.

References

1. Christer Andersson, Reine Lundin, and Simone Fischer-Hbner. Privacy-enhanced WAP Browsing with mCrowds - Anonymity Properties and Performance Evaluation of the mCrowds System. In *Proceedings of the Fourth annual ISSA 2004 IT Security Conference*, pages 85–90, Johannesburg, July 20044.
2. Christian Beil. Development of a Framework for Dynamic Mobile Anonymity. University of Mannheim, December 2005. Bachelor Thesis.
3. Bouncy Castle Crypto APIs. URL:http://www.bouncycastle.org.
4. David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.
5. N. Diezmann. *Report Mobile Business - Neue Wege zum mobilen Kunden*, chapter Payment - Sicherheit und Zahlung per Handy (in german), pages 155–178. 2001.
6. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
7. Ingrid Duda, Markus Aleksy, and Thomas Butter. Architecture for Mobile Device Integration into Service-Oriented Architectures. In *Proceedings of the 4th International Conference on Mobile Business (ICMB2005)*, Sydney, Australia, July 2005.
8. Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.
9. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
10. Jap: Anonymity and privacy tool for internet. URL:http://anon.inf.tu-dresden.de.
11. Stefan Köpsell, Hannes Federrath, and Marit Hansen. Erfahrungen mit dem Betrieb eines Anonymisierungsdienstes (in german). *Datenschutz und Datensicherheit*, 27(3), 2003.
12. M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS Protocol Version 5. RFC1928, March 1996.
13. Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003.
14. Andreas Pfitzmann et al. Anonymity, Unobservability, and Pseudonymity: A Proposal for Terminology, July 2000.
15. Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.
16. Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.
17. Status of Jap cascades. URL:http://anon.inf.tu-dresden.de/status.php.

18. Emin Islam Tatlı, Dirk Stegemann, and Stefan Lucks. Security challenges of location-aware mobile business. In *Proceedings of the 2nd IEEE International Workshop on Mobile Commerce and Services*, pages 84–93. IEEE Computer Society, 2005.