Secure Interaction with Piggybacked Key-Exchange

Wolfgang Apolinarski, Marcus Handte, Muhammad Umer Iqbal,
Pedro José Marrón

Networked Embedded Systems
University of Duisburg-Essen
Duisburg, Germany
Email: firstname.lastname@uni-due.de

Abstract

Online collaboration tools are a ubiquitous mediator of many human interactions. In the virtual world, they enable secure interaction by controlling access to shared resources. Yet relying on them to support face-to-face collaboration might be suboptimal as they require Internet connectivity. A more efficient way of co-located resource sharing is the use of local communications. Yet setting up the necessary security mechanisms can be cumbersome. In this article we present PIKE and its variant P2PIKE, key exchange protocols that minimize this configuration effort. Both piggyback the exchange of keys on top of an existing service and exchange keys proactively – before the interaction takes place. In addition to an implementation, we outline two applications and present a thorough evaluation to show the benefits and limitations of our approach.

Keywords: Key-exchange, online services, smart phones

1. Introduction

Online collaboration tools such as Google+, Facebook or Dropbox have become an important and ubiquitous mediator of many human interactions. In the virtual world, they enable secure remote interaction by supporting restricted sharing of resources such as documents, photos or calendars between users. Users are typically identified with a unique identifier and they authenticate themselves by means of passwords or similar mechanisms[1]. The shared

[1]Examples are two-factor authentication or client-specific certificates.

resources can then be tied to different sets of identifiers such as friend lists in Facebook or circles in Google+. To control access to resources, online collaboration tools typically use encrypted communication such as TLS and require authentication upon resource access. Using web-based interfaces, users can access their information from different machines. In addition, many services also provide mobile applications to support access on-the-go and an API for third-party tools to access their resources. Besides providing optimized visualizations, most mobile applications and third-party tools make use of local caching and synchronization to enable disconnected operation.

The success of these collaboration tools indicates that this mediation model can effectively support secure remote interaction. Yet, using them for face-to-face collaboration[2] that take place in the physical world can be suboptimal. The main reason for this is that such collaboration involves multiple partners that interact with each other *at the same time and place.* In such a setting, the various issues arising from a remote connection—such as higher response times or intermittent connectivity—cannot be hidden by caching and synchronization. A much more efficient way to support online collaboration between co-located partners would be to support their online collaboration by means of short-range wireless communication. However, to provide a similar level of security, this would require encryption and the configuration of corresponding authentication mechanisms. Without a mediating online tool, the co-located interaction partners would have to manually exchange authentication or encryption keys which so far has been too cumbersome to be used in practice.

To avoid this problem, we have designed PIKE, a key-exchange protocol that aims at seamlessly extending the support provided by online collaboration tools to enable wireless collaboration among face-to-face collaborators. The basic idea behind PIKE is to piggyback the exchange of keys on top of the existing service infrastructure of an online collaboration tool in a proactive manner. Thereby, we eliminate the need for manual configuration as well as Internet connectivity when the interaction takes place locally.

A prototype application scenario for PIKE is a business meeting for which invitations are shared over a secure connection using Google Calendar. When detecting the invitations, PIKE automatically exchanges keys over the Internet using the Google infrastructure and stores them locally on each partner's

[2]Here, collaboration is not restricted to work context.

device. When the meeting takes place, the keys can be used to establish secure wireless LAN communication among the participants' devices or to authenticate participants without requiring any Internet connectivity. A second scenario envisions two friends that meet spontaneously in the city. Using their smart phones, they want to interact and transfer data (e.g., pictures) securely. A variant of PIKE, P2PIKE enables them to use automatically exchanged peer-to-peer keys to set-up a secure communication.

The contribution of this article is threefold. First, we introduce PIKE as an approach for enabling non-mediated, secure and configuration-free face-to-face collaboration. Second, we describe its implementation as an extensible Android library that integrates with wireless tethering to enable the fully automatic establishment of a secure wireless LAN. Third, we present several applications and an analytical as well as an experimental evaluation indicating that PIKE is broadly applicable and (at least) as secure as the underlying online service.

2. Approach

Our goal with PIKE is to support local collaboration in a configuration-free and secure manner that does not require Internet connectivity during the time the interaction takes place. To achieve this, PIKE exchanges keys piggybacked on an existing service infrastructure before the interaction takes place. This key exchange is typically triggered by a virtual representation of the upcoming interaction such as a meeting entry in a calendar. Additionally, it can be performed with all known users of an online service that use PIKE (e.g., Facebook friends running the PIKE app). The exchanged keys can then be used by applications to secure a wireless LAN communication, e.g., by means of encryption or message authentication.

2.1. System Model and Assumptions

The technical basis for PIKE are mobile *devices*, such as phones, tablets or laptops that share *resources* with each other remotely through a *network*, mediated by a *service*. Regarding these four building blocks we assume:

- *Services:* The service enables secure restricted sharing of resources. This means that the service authenticates its users, models relationships between different users with respect to resource usage and enables the specification and enforcement of access rights. The service performs its access control to resources properly. Meaning that a) it protects the resources from being accessed by illegitimate users and b) it allows access from legitimate users. Yet, beyond proper service operation, we do not assume that the service is necessarily trustworthy. Examples may be Facebook, Google+ or Dropbox.

- *Network:* A network such as the Internet enables devices to access the service regularly. The network may be insecure and, occasionally, it may be unreliable or unavailable, e.g., due to a network outage, an incomplete coverage or unaffordable roaming fees.

- *Devices:* The user's device is able to access the service regularly through the network. For this, the service uses a mobile application that synchronizes the changes to a shared resource or provides an API that can be used for resource synchronization.

- *Resources:* Some of the resources shared between users can be read and edited not only by the creator but also by the interaction partners. For PIKE, we assume that some of the shared resources are used to plan a face-to-face collaboration and thus, provide time information. Examples for such resources are a calendar entry or a message indicating an upcoming meeting. Using P2PIKE, this assumption can be dropped. In Section 2.3.2, we describe the necessary changes to PIKE.

2.2. Design Rationale and Goals

Besides achieving the functional goal of exchanging keys, the desire to maximize PIKE's applicability for various types of face-to-face collaboration defines the following design goals.

- *Full Automation:* To minimize the time required by interaction partners to set up secure communication, PIKE should not require manual configuration. Instead, the key exchange performed by PIKE should be fully automated such that it becomes transparent to them.

- *High Security*: In order to truly protect the interactions between the devices, key exchange with PIKE must be secure. Given that the collaboration partners are already using an online collaboration service through remote interaction in order to share resources in a secure manner, the use of PIKE to secure wireless collaboration in face-to-face meetings should result in (at least) the same level of security as offered by the online service.

- *Low Latency:* To avoid situations in which keys are not available for partners, the exchange of keys with PIKE must not introduce a high latency. Specifically, PIKE should be able to provide the exchanged keys quickly after a face-to-face collaboration has been planned, i.e., after a corresponding calendar event has been created or an invitation email has been sent. The key should therefore be exchanged within a few seconds.

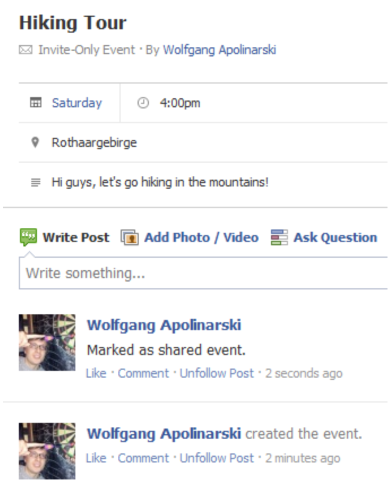
- *High Scalability:* As the face-to-face collaboration may involve groups of different sizes, PIKE should be able to support typical group sizes. This means that it should not only be able to provide keys for office scale meetings, but it should also support larger events such as scientific conferences with a few hundred participants.

2.3. Key Exchange Protocol

As explained in Section 2.1, the mobile application provided by the service contacts the service and synchronizes recent resource changes regularly onto the user's device. If the mobile application is not present, the PIKE or P2PIKE app can use the API provided by the service to do this resource synchronization. We now describe our two approaches for key exchange protocols, PIKE and its variant P2PIKE. Both protocols piggyback on services such as online collaboration platforms. PIKE exchanges keys for a group of users. The key exchange is triggered by a resource (e.g., an event) that is stored at the service. P2PIKE exchanges keys on a peer-to-peer basis with single users of a service. Using our current implementation, both require the users to use a service that is supported by the PIKE application (i.e., Google Calendar or Facebook).

2.3.1. PIKE

PIKE is using an online collaboration platform to support resources like group events. PIKE uses resource changes in online collaboration platforms to trigger key exchanges. Every time a resource changes, PIKE starts to analyze it in order to detect changes in PIKE-enabled resources that trigger a key exchange. An example for such resource changes is the creation of an event on Facebook that has been shared with a set of friends and is *marked as shared event* (c.f., Figure 1(a)).



Hiking Tour

Invite-Only Event · By Wolfgang Apolinarski

Saturday 4:00pm

Rothaargebirge

Hi guys, let's go hiking in the mountains!

Wolfgang Apolinarski

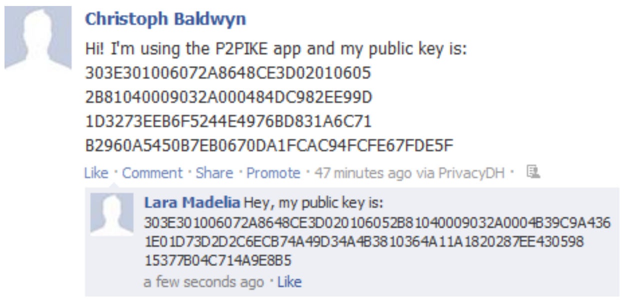
Marked as shared event.

Like · Comment · Unfollow Post · 2 seconds ago

Wolfgang Apolinarski created the event.

Like · Comment · Unfollow Post · 2 minutes ago

(a) PIKE Resource and Trigger



Christoph Baldwyn

Hi! I'm using the P2PIKE app and my public key is:
303E301006072A8648CE3D02010605
2B81040009032A000484DC982EE99D
1D3273EEB6F5244E4976BD831A6C71
B2960A5450B7EB0670DA1FCAC94FCFE67FDE5F

Like · Comment · Share · Promote · 47 minutes ago via PrivacyDH ·

Lara Madelia Hey, my public key is:
303E301006072A8648CE3D020106052B81040009032A0004B39C9A436
1E01D73D2D2C6ECB74A49D34A4B3810364A11A1820287EE430598
15377B04C714A9E8B5

a few seconds ago · Like

(b) P2PIKE Resource

Figure 1: PIKE Resources in Facebook

Once a relevant change has been detected, key exchange takes place. For this, each partner shares a key with the creator of the resource, who in turn shares her or his key with each of the partners. This key exchange is facilitated by sharing capabilities that are already provided by the service. To do this, PIKE performs either a local modification on the triggering resource or, if this is not possible due to a limitation of the mobile application, it uses the API of the service. Once the changes have been made, PIKE waits for the next resource synchronization at which point all partners will receive the key of the resource creator and the resource creator will receive all keys from all other partners. Once this key exchange has taken place, these keys can be used to enable direct secure communication among the devices of the partners.

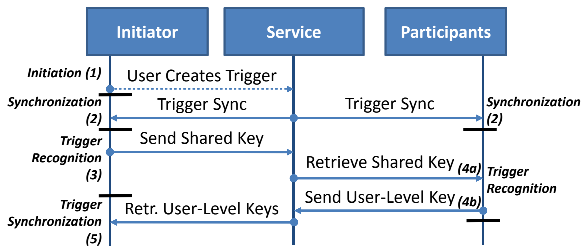
Figure 2(a) depicts the resulting logical protocol flow which is executed remotely by all partners before the face-to-face collaboration takes place. Conceptually, PIKE involves three entities, namely the device of the partner creating the resource (the initiator), the devices of the remaining partners (the participants) and the online collaboration service. To establish keys, these three entities interact with each other using five steps.

1. *Initiation:* The initiator creates the resource that triggers the key exchange (e.g., an event invitation) using the service's UI and marks it

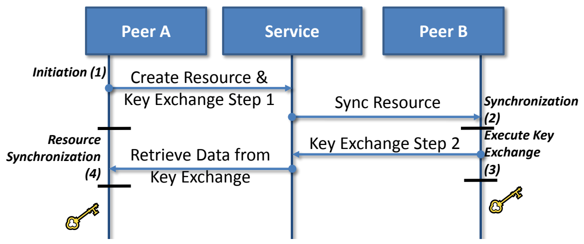
as a PIKE-enabled resource. Thereby, the initiator specifies the set of users (participants) that should be able to access the resource and sets the appropriate access restrictions. The resource is then added to the service which makes it available to the specified users.

2. *Synchronization:* After the resource has been added and shared by the service, the devices of all partners will eventually retrieve the change as part of their normal synchronization process. At that point, the device of the initiator as well as the devices of the participants can access the triggering resource.
3. *Trigger Recognition - Initiator:* The initiator's device recognizes the trigger resource retrieved from the service. The local PIKE process creates a key and shares it with all participants simply by attaching it to the resource which will be synchronized with the service again. This key can later be used to protect the communication between the devices of the partners from other entities that are not participating in the interaction.
4. *Trigger Recognition - Participants:* After the initiator has attached its key, it will eventually be propagated to the participants. At this point, their corresponding local PIKE processes retrieves the key and stores it for later use (4a). In order to enable user-level authentication, the PIKE processes on the participants' devices in turn each create a new key and share it with the initiator (4b). Depending on the service, this can be either done by attaching it to the triggering resource or by creating a new resource that is only shared with the initiator.
5. *Trigger Synchronization:* Every time a participant shares a new key (by means of attaching it or by creating a new resource for it), the initiator will eventually receive it on his device. At this point, the PIKE process on the initiator's device learns that this participant wants to join the event, takes the key and stores it for later use.

After the completion of these steps all interaction partners possess the key generated by the initiator and the initiator shares a key with each of the participants. Once the face-to-face collaboration takes place, these keys can be used to enable group communication as well as private communication and user-level authentication between the initiator and the participants. The authentication information can then be used to bootstrap further keys (e.g., using Kerberos), though the applications described in this article do not require this.



(a) PIKE



(b) P2PIKE

Figure 2: Logical Protocol Flow

To speed up the sequential protocol flow described previously, it is possible to execute steps 3 and 4b in parallel. For this, the participants share their keys with the initiator immediately after detecting the triggering resource (without waiting for the initiator to share a key). As part of step 5, they then check for updates on the trigger resource in order to detect the initiator's key as soon as it has been shared. As we describe later on, this simple parallelization can provide a significant speed up.

PIKE distributes a symmetric key for each event resource (i.e., a shared group key) which allows us to support resource constrained devices using symmetric cryptography. Although theoretically possible, in our implementation, we do not support asymmetric key exchanges (like a group key diffie-hellman) here. Asymmetric cryptography could be implemented in PIKE, but would result in a higher processing time. For a similar reason, the individual user-level keys are distributed as symmetric keys (i.e., posted by the participants).

2.3.2. P2PIKE

In general, PIKE uses a triggering resource to support full automation. This automation allows choosing the necessary keys automatically at the time the face-to-face collaboration takes place. In order to drop the assumption of needing a triggering resource, we created P2PIKE, a peer-to-peer version of PIKE which allows exchanging keys with each user individually, i.e., the initiator of an exchange would possess individual keys, one for each user. The exchanged keys can then be used for spontaneous face-to-face collaboration. It depends on the used discovery mechanism (e.g., visual identification of users or the transferring of data between users that includes a user's id) how far automation is supported in P2PIKE.

In contrast to PIKE, P2PIKE is focused on single users (i.e., peers) instead of a group of users. It allows to exchange keys which can then be used for peer-to-peer communication at face-to-face meetings. The usage of the keys is not restricted at all and could be used for the secure exchange of any kind of data. Instead of using a resource to trigger the key exchange, users can either manually trigger the key exchange by choosing from a list of P2PIKE users on the online collaboration platform of their choice. Alternatively, they automatically start to exchange keys with all users of P2PIKE that they are related to (e.g., *friends* in Facebook) when they start to use P2PIKE. If P2PIKE detects a key exchange request from such a friend, it automatically answers the request without user interaction.

As depicted in Figure 2(b), the goal of P2PIKE is the automatic computation of a shared secret key for two peers. Additionally, the service should not be able to passively overhear this key. Therefore, we support asymmetric algorithms like Diffie-Hellman (DH) or RSA which require two messages that can be executed in the following four steps.

First, an asymmetric key pair (e.g., a DH key pair) is created. Then, similar to the procedure performed by PIKE, P2PIKE will use the mobile application or the API of the service. P2PIKE will—depending on the used online service—attach or create a new resource (e.g., Figure 1(b)), adding the public key of the created key pair *(Step 1)*. At the next resource synchronization, the peer will pick up the resource *(Step 2)*, create his own key pair and then attach his public key to it *(Step 3)*. In the next synchronization interval, the device that triggered the key exchange will recognize the attachment from the peer and complete the key exchange *(Step 4)*. At this time, both peers can now compute the shared secret key by extracting the other peer's public key from the resource. The key pairs that were created during the key exchange are not used anymore and deleted after the shared secret key is computed. This key can be used for secure communication similarly to the shared group key that was exchanged using PIKE. In difference to the shared group key, which identifies a communication group, the shared secret key identifies a connection between two peers.

At the time a face-to-face collaboration between the two peers takes place, both can use the exchanged symmetric key to establish a secure communication channel using symmetric encryption which is suitable for resource constrained devices. Also note that—in contrast to PIKE—the key itself is never transmitted to the service, but only the two public keys of the used asymmetric key exchange algorithm.

3. Implementation

To validate the concepts of PIKE and P2PIKE, we have implemented both approaches on top of the Android operating system. The implementation consists of activities (i.e., user interfaces) and services (i.e., background tasks) that not only perform the key exchange but also facilitate secure communication and user-level authentication. As depicted in Figure 3, our implementation of PIKE is modular and can support arbitrary services that exhibit the characteristics described in Section 2.1 by means of a simple plug-in model. To demonstrate this, we have developed two plug-ins for popular services. The first one taps into the Google infrastructure and uses shared calendar entries as trigger. The second plug-in taps into Facebook and uses Facebook events as triggers. Both plug-ins presented here use the API of their respective services. They are independent of any *app* provided by the service provider (like the Facebook app).

3.1. Core Library

The mechanisms of PIKE used commonly across different services are implemented as an Android library (apklib). This core library is responsible for managing the interaction with different plug-ins and it provides functionality to create keys. The latter is used for both the exchanged key for the group (on the initiator's device) as well as the user-level keys. Keys are created using a secure pseudo random number generator that creates 128 bit keys (to support common ciphers such as AES). To further simplify application development, the signaling of a face-to-face collaboration is also implemented as part of the core library. For this, the library manages the meta data such as the list of participants, prospective start and end time, keys, etc. The meta data is represented as a parcelable object which can be passed to the scheduler provided by Android (i.e., the AlarmManager), which fires a notification (a so-called Intent) when the start time has passed - even if the device has been gone to sleep. With this meta data, the Android scheduler can automatically trigger further applications.

In addition, the core library also enables applications to set up secure wireless communication among all devices of the interaction partners and it provides the capability for user-level authentication. For providing user-level authentication, it uses a key derivation function and a challenge-response mechanism using a Keyed-Hash Message Authentication Code (HMAC). For

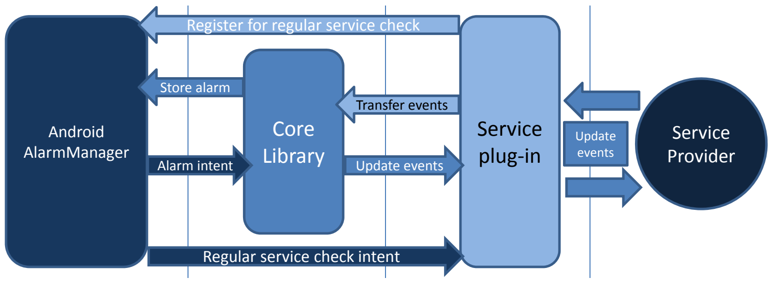


Figure 3: PIKE Architecture for Android

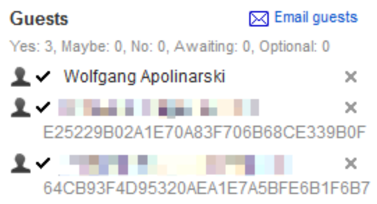
providing secure wireless communication, the core library is capable of establishing a wireless LAN using the tethering capabilities of the Android operating system[3]. If an application requests secure wireless communication, the group key exchanged via PIKE is used to setup a WPA2 protected network. To do this, all devices present at the face-to-face collaboration derive an SSID as well as a passphrase from the group key. The initiator's device then uses this key to automatically configure and start the tethering mode. The devices of the participants automatically add the configuration to the list of preferred networks, such that the device will automatically join if the network is in the vicinity. While the interaction takes place, the devices can use this network to interact securely. After its completion, the devices automatically remove the network configuration and the tethering mode is deactivated.

Using this functionality, it is easy to build further applications. However, in order to be useful, the core must be integrated with an online service. To do this, the core relies on plug-ins which realize the recognition of triggers and the attachment of keys.

3.2. Google Calendar Plug-in

Google Calendar is one of the most used business services for the management of shared calendars. At work the calendars can be used to be informed about the absence and availability of co-workers. Meetings can then be planned efficiently and co-workers can be invited to join via the service.

[3]The tethering capabilities are not part of the official Android API. Yet, the required functions are available on many devices running Gingerbread or higher and they can be invoked via Java reflection.



Guests

Email guests

Yes: 3, Maybe: 0, No: 0, Awaiting: 0, Optional: 0

✓ Wolfgang Apolinarski ×

✓ [redacted] ×
E25229B02A1E70A83F706B68CE339B0F

✓ [redacted] ×
64CB93F4D95320AEA1E7A5BFE6B1F6B7

Figure 4: User-level keys, posted in an event in Google Calendar

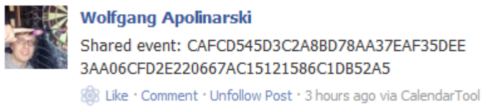
Additionally, Google Calendar is pre-installed on most Android phones making it a perfect candidate for PIKE.

To access Google Calendar, we use its API to regularly synchronize events between devices and the service. Intuitively, we use shared meetings—appointments with multiple guests—as a trigger for the key exchange. In order to distribute the key generated by the creator of the appointment (i.e., our initiator), we use a hidden, non-visible field (shared extended properties). This field is automatically synchronized with all guests (i.e., our participants) and, if the event is set to private, only guests can see these properties. To distribute the user-level keys, we use the comment field of the appointment as shown in Figure 4. Again, if the appointment is private and guests do not have the permission to see the guest list, this field cannot be seen by others except by the creator. So using this approach, the initiator can retrieve all user-level keys and the participants can retrieve the coordinator's key by reading the associated fields of the appointment.

3.3. Facebook Event Plug-in

Our second plug-in uses Facebook. This online social network service is used mostly for private communication. This usually includes the planning of events or trips with multiple persons. Facebook also provides clients for many different mobile operating systems (e.g., iOS, Android, Windows Phone), so users can stay connected as long as they have network access.

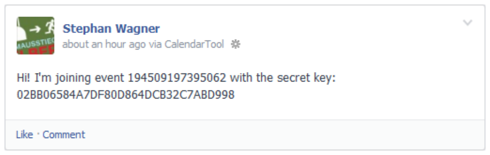
For this implementation, we are using the Facebook Graph API to access and modify data from the social network. As trigger for PIKE we use Facebook events. The participants of events in Facebook can be constrained by the event's initiator. Each event has a private place for discussions only viewable by the event's guests, the so-called *wall*. This wall is used to post the initiators key that is then automatically picked up by the participants' devices. Since participants cannot change the visibility of posts on the event's


Wolfgang Apolinarski

Shared event: CAFCD545D3C2A8BD78AA37EAF35DEE 3AA06CFD2E220667AC15121586C1DB52A5

Like · Comment · Unfollow Post · 3 hours ago via CalendarTool

(a) Shared key, at the event's wall


Stephan Wagner

about an hour ago via CalendarTool

Hi! I'm joining event 194509197395062 with the secret key: 02BB06584A7DF80D864DCB32C7ABD998

(b) User-level key, at a participant's wall

Figure 5: Keys exchanged with PIKE in Facebook

wall, the participant keys are posted to the participants' profiles. Each profile has a place for discussion (also called *wall*). On this wall, posts can be created with a privacy setting that constrains the access to the event's initiator (see Figure 5(b)). The initiator can then retrieve the participants' keys by going through their walls. For similar reasons, P2PIKE uses a post on a profile's wall as resource. The post can only be seen by the two peers and is deleted automatically by P2PIKE, after the successful execution of the key exchange.

4. Applications

To validate the approach, we have developed a number of applications that apply PIKE. In the following, we present two of them. The first one provides registration services for large scale events. For this, the application requires secure user-level authentication but it cannot rely on Internet connectivity. The second one shows how PIKE can be used to establish a private network that can be used for energy-efficient resource sharing. This application requires a low-latency reliable private network in an environment where no public network (like UMTS) is available. Both applications are canonical examples that show PIKE's capabilities in different conditions.

4.1. Configuration-free Conference Registration

As an example for using PIKE in a large-scale event such as a scientific conference, we developed a registration application. At a registration desk for a scientific conference, users register themselves by giving their names, which are then looked up in a list. Using PIKE this time-consuming, error-prone and insecure process can be automated and secured appropriately. To do this, all participants are invited to a Facebook event of the organizers as part of the participant's payment process. When PIKE detects the event, the creator of the event (i.e., the organizer) will provide a key to all participants and the participants will share their user-level keys with the organizer.

At the start of the conference, each participant uses its mobile device to identify itself at the registration desk. To detect the presence of the registration desk, the mobile devices can use the automatic WLAN configuration described previously. On top of that, in order to identify themselves in a secure manner, they send a verifiable message containing their name to the device of the organizer. To create this verifiable message, they compute an HMAC over the message using their individual user-level key which was distributed by PIKE. The organizer can then validate the key and mark the participant as registered.

To perform this process, neither the participants nor the organizers need to perform any manual configuration at the conference site. PIKE detects the shared event days before the start of the conference and exchanges keys. The keys are then used to provide user-level authentication at the conference's start in a fully automated process without needing Internet connection.

4.2. Energy-efficient Resource Sharing

As an example for using P2PIKE, we have developed an energy-efficient resource sharing application that aims at supporting the collaborative collection of GPS traces during a hiking trip. Imagine several friends who want to go hiking in the mountains. During their trip, they will not be able to get a reliable Internet connection. Yet, while they are hiking, they want to be able to display their current location in an offline map to support outdoor navigation. Once they completed their trip, they want to be able to analyze their walking and climbing speeds. To do this, they would like to continuously capture their current GPS coordinates but due their devices' energy constraints each of the friends can only capture a part of the trip.

Our application mitigates this problem by distributing the energy load generated by the GPS receiver among the friends participating in the hiking tour. To do this, the devices of the hiking friends perform GPS data collection in a round-robin fashion and share the results through WLAN with all other devices. To support the traceability of the GPS data, the data must be authentifiable. Using P2PIKE, the friends can spontaneously meet for a hiking tour without the need for a triggering resource since they performed the piggybacked key-exchange in advance and can now use the shared keys to establish secure peer-to-peer communication. To do this, the friends can join an insecure WLAN network (e.g., a WLAN hotspot created by one of them) and use the shared keys to authenticate each other. After successful authentication, one of the friends starts sharing his GPS coordinates. He does that by creating a secure network link to each of the friends (using the keys shared with them), transmitting the GPS coordinates over this link. The whole process does not need an active Internet connection, since P2PIKE performed the key-exchange several days before, without needing a physical meeting of the friends or devices. After joining the network, each friend retrieves the GPS coordinates from the sharer's device and the client application shows the current position (using android's *mock locations* feature). Our measurements[4] show the following: Two devices running only the GPS receiver need each 891 mW (i.e., 1782 mW in total). Using P2PIKE, it is possible to combine a device receiving the GPS coordinates over WLAN (660 mW) and a second device using the GPS receiver and sending the coordinates over WLAN (935 mW) with a lower total consumption of 1595 mW.

5. Evaluation

In the following, we evaluate PIKE and its variant P2PIKE along the four design goals that we introduced in Section 2.2. These are *full automation*, *high security*, *low latency* and *high scalability*. Next, we discuss each of them.

5.1. Full Automation

The key-exchange with PIKE does not require manual interaction. Nevertheless, to establish a key using PIKE, it is necessary for all participants to use a third-party service (e.g., Facebook). Also, the initiator needs to create the shared trigger condition (i.e., event) using this service and adds the guests or group of guests manually using for example the service's web interface. However, managing events and inviting guests are necessary steps that always need to be performed by an event creator, even when not using PIKE.

5.2. High Security

As discussed before, the security of PIKE depends on the security of the service provider. The service provider needs to implement the resource sharing in such a way that resources which are shared with a limited number

[4]Created using a Galaxy Nexus smartphone running Android 4.2.2, 600 aggregated measurements (1 every 100ms) per point.

of users are properly secured. The resources should not be accessible for any other users than the ones that are specified by the event creator. Many service providers (e.g., Facebook and Google Calendar) support this type of resource sharing. In these cases, PIKE does not lower the security provided by the service providers, i.e., the security level is as high as the security level of the used service provider.

Of course, one possible attacker that could tamper the key-exchange is the service provider (e.g., Facebook) itself. When using PIKE, all exchanged keys are stored in the service provider's database, it would be easy to retrieve them from there. When P2PIKE is used, the performed key exchange can only be intercepted by the service provider before the execution of Step 2 (c.f., Figure 2(b)). Here, an active man-in-the-middle attack must be performed by the service provider to be able to decrypt messages during the face-to-face collaboration. Nevertheless, the service provider itself is not physically present during the face-to-face collaboration. As a result, it cannot use the exchanged key(s) of PIKE and P2PIKE since the interaction triggered by the devices is not spread through the Internet or any other publicly accessible network.

To further mitigate this attack, e.g., if a service provider also deploys hotspots or similar devices so that it might be possible to be physically present during the interaction, service providers can be combined. The trigger condition resource is then distributed to several service providers, all of them must be supported by all participants and the initiator. For enhanced security, the exchanged shared key on the service is different, if the service provider differs. The combination (e.g., XOR) of the exchanged shared keys from all the service providers will then be used by the participants. As a result, one service provider cannot overhear the face-to-face collaboration, because it only knows parts of the key. The interaction of all involved service providers is then necessary to retrieve the whole key.

Possible attacks that do not involve the service provider as an attacker are attacks from devices that take part in the collaboration (i.e., have received the key for secure communication) or attacks from devices that do not possess the exchanged key, but are just in the vicinity of the interacting partners. For the latter devices, it is not possible to attack the interaction as long as the communication is always encrypted, using a secure encryption protocol (e.g., AES). Any kind of attack is then not based on the security of PIKE, but on the security of the mechanism that is used for encryption.

Additionally, the user-level keys can be used to establish an encrypted private communication channel between two users in the collaboration group (e.g., to remove threats from malicious users or devices). If one of the users that wants to open a private channel to another user is the initiator, the user-level key can be used directly. If two participants want to open a private channel, either the initiator's device can be used as a trusted third party that verifies the authentication of the other users or a key that was exchanged by P2PIKE can be used. The exchange of a shared peer-to-peer key will then enable the devices to communicate with each other using encryption.

If the initiator of a face-to-face collaboration group is malicious, the whole communication is compromised. From the participant's view, this can only be avoided by not taking part at the face-to-face collaboration at all. In the end, this results in ignoring trigger resources coming from malicious devices or users, i.e., not starting a face-to-face collaboration with malicious initiators. PIKE will not protect the participants' communication, if the initiator of the key-exchange is malicious.

5.3. Low Latency

Using two Samsung Galaxy Nexus (2x1.2 GHz ARM CPU, 1 GB RAM) mobile phones that were connected to the Internet via WLAN, we measured the latency of our approach. The mobile phones were running Android OS version 4.1.1. To perform the measurements, we were using the prototypical implementations presented in Section 3. All values represent the average of 100 measurements performed by one smart phone.

The minimum latency of the overall approach depends on the synchronization interval t. A typical value for t is 60 minutes, i.e., the service is checked every hour. We now describe the performance of PIKE and P2PIKE.

5.3.1. PIKE

As depicted in Figure 6, there are two possible scenarios: After the creation of the shared trigger, it is either picked up by the initiator's device (a) or by a participant's device (b) after a maximum time of t. For (a), the initiator picks up the shared trigger and adds the shared key to the trigger. It cannot retrieve the user-level keys, since they are not yet available. It then waits for the next synchronization interval. In the meantime, the participants synchronize the trigger resource, retrieve the shared key and attach their individual user-level keys to the trigger. Fulfilling this combined step, they have finished the PIKE message exchange and do not need to synchronize

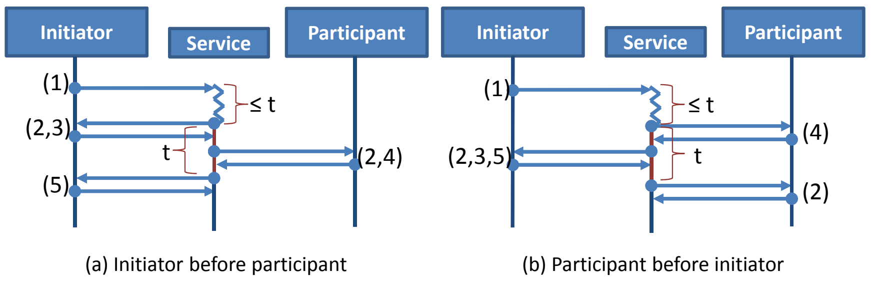


Figure 6: Time needed to perform PIKE, with synchronization interval t

this trigger, again. After the synchronization interval, the initiator synchronizes the trigger resource, now retrieving all user-level keys. Since the first pick up time is not higher than t and the second wait time is exactly t, the PIKE message exchange is finished after a maximum time of $2t$. Similar to that, in (b), the total time to execute PIKE also results in $2t$.

Depending on the used service, this can be further improved by retrieving the *created_time* (if existing) of the shared trigger. The initiator can then retrieve the *created_time* and reschedule the synchronization interval such that the next PIKE synchronization takes place at the time $created\_time + t$. Since between the *created_time* and $created\_time + t$ all participants will have submitted their user-level keys, the initiator is able to retrieve all user-level keys (and finish the PIKE message flow) after a time of only t.

Regarding our implementations, the actual message flows are similar to the logical protocol flow presented in Figure 2(a). The numbers of messages per step in the implementations differ from that, because one logical step might include several API calls. Using our implementation based on Google Calendar, we find that for the initiator, the minimum number of messages (and Google API calls) is 3 (leaving out step 1); for a participant it is always 3. We also measured the average time that is needed to perform the PIKE protocol flow (σ depicts the standard deviation). We assume that the user creates the event using Google Calendar (either using a mobile phone or the web-interface) manually (i.e., performed step 1). Steps 2 and 3 took 535ms (σ = 43ms), since the initiator needs to perform step 5 (255ms (σ = 59ms) with 2 participants) as well, this gives a total number of 790ms (σ = 73ms) for the API calls. Each participant needs to execute steps 2 and 4 which takes 583ms (σ = 170ms). As a result, we see that the prototypical implementation of PIKE with Google Calendar exhibits a latency of less than a second.

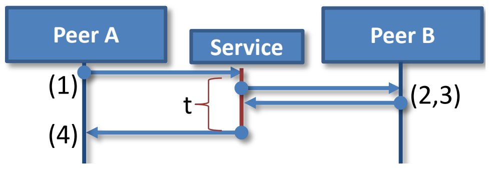


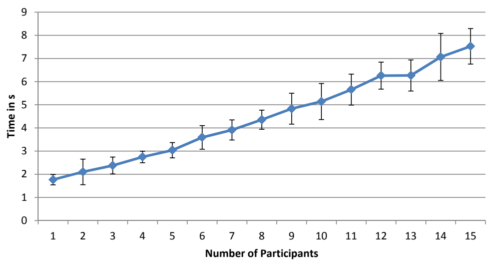
Figure 7: Time needed to perform P2PIKE

Depending on t, also spontaneous meetings can be supported. However, a typical value for t like 60 minutes constrains this to meetings known two hours in advance.

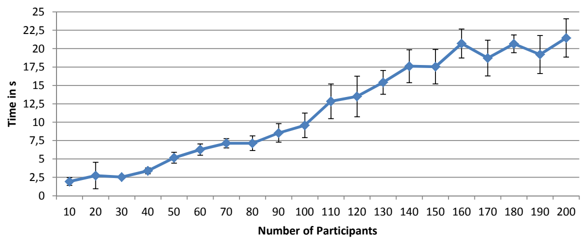
In Facebook the number of messages changes, because the initiator has to retrieve the user-level keys from each participant's wall. Also, the Facebook Graph API does not give as many detailed information on the events as the Google Calendar API with only one API call. The minimum number of messages (and API calls) for the event initiator is therefore $4 + (n - 1)$ (also performing the steps 2, 3 and 5, with n being the number of participants), while the number for each participant is still 4 (performing step 2 and 4). Using the Facebook batch API, n increases only every 50 participants. As discussed before, we assume that the user creates the event from within Facebook (step 1). The average time necessary to perform the API calls for steps 2 and 3 is 2193ms (σ = 394ms). Step 5 is discussed in the Section *Scalability*. The participants need to perform step 2 and 4, this takes 3067ms (σ = 354ms). We conclude that the Facebook API has a higher latency, but is still within reasonable limits, with a latency of less than 30 seconds for 200 participants.

5.3.2. P2PIKE

In contrast to PIKE, P2PIKE executes all steps automatically, including the initial first step. As can be seen in Figure 7, P2PIKE needs a time of t to perform the key exchange with another peer (assuming that their synchronization interval is not completely time-synchronized). At first, Peer A creates a resource and executes the first step of the key exchange and waits then the time of t before synchronizing the resource, again. During that time, Peer B retrieves the resource and executes Step 2 of the key exchange. Peer B now already has computed the shared secret key. Then Peer A accesses the resource, receives the public key attached by Peer B during the second step of the key exchange and computes the shared secret key. The total time



(a) One-by-one retrieval



(b) Batch retrieval

Figure 8: Retrieving user-level keys from Facebook

for the execution of P2PIKE is therefore t.

The measurements of our prototypical implementation are based on 100 Facebook test users and show that Peer A (the initiating peer) needs 3823ms for the execution of step 1 and 4 (1378ms (σ = 531ms)+2445ms (σ = 430ms)), i.e., the two API calls. Similarly, Peer B only needs 2336ms (σ = 185ms) for the execution of Step 2 and 3. All measurements include the execution time for an ECDH key exchange (160 bit) to obtain a common key. We conclude that the prototypical implementation of P2PIKE exhibits a low latency.

5.4. Scalability

The execution time of P2PIKE fully depends on the latency of the protocol execution. Since each execution of the P2PIKE protocol flow will create one shared key between two peers, the costs (in time and resources) increase quadratically with the group size. As shown in Section 5.3, this would result in a latency of less than 4 seconds for each peer. For PIKE, scalability depends on the used service. If PIKE is used with Google Calendar, an event contains all information, including user-level keys (comments). A single API call returns the necessary information. The message's length increases slightly with each participant since a comment is added. Nevertheless, this results in high scalability.

Using Facebook, the number of requests from the event initiator to the service depends heavily on the number of participants. Therefore, we measured PIKE with an increasing number of participants, using single requests (Figure 8(a), showing a clear linear increase, 100 measurements per point, error bars show the standard deviation ($\pm\sigma$)) or batch requests (Figure 8(b), 10 measurements per point, error bars show $\pm\sigma$). Batch requests bundle

50 requests (i.e., one for each participant) in one. Both measurements indicate a linear increase from 1765ms (1 participant) to 7526ms (15 p.) for the single requests and from 1935ms (10 p.) to 21453ms (200 p.) using batch requests. Although the implementation based on Facebook is slower, it still achieves a high scalability using batch requests. In summary, PIKE reaches the scalability goal of supporting a few hundred participants.

6. Related Work

When compared with other approaches, PIKE exhibits three main differences. First, it can easily scale to hundreds of collaboration partners. Second, it provides user-level as opposed to device-level authentication. Third, it does not require Internet connectivity during the face-to-face collaboration.

To provide user-level authentication, some approaches use server-based context-detection as the mechanism to allow or deny access to resources [1, 2]. PIKE is not based on a central server, but uses third-party infrastructure (i.e., online services) to establish shared and user-level keys. In [3] and [4] the authors describe how context, such as microphone recordings, can be used directly to create a shared key between devices. According to them, this works well inside buildings, but ceases to work properly if used outside. In contrast to that PIKE can be used both indoors and outdoors. Using context data from the smartphone's sensors also exhibits scalability issues. Imagine sensor data from an accelerometer which needs devices to be shaken together (up to a duration of 20s) such that the accelerometer detects (almost) identical data. Smart-Its friends [5] and similar approaches [6, 7] generate a key based on this data. However, this approach is only feasible for a small amount of devices. The combination of different context features for key-exchanges is presented by PINtext [8] or Mayrhofer et al. [9]. While enhancing security by combining context features, this combines their disadvantages, e.g., using the accelerometer and the microphone at the same time requires shaking the devices and constrains the key-exchange to indoor locations. Although these approaches do not need an Internet connection, context dependent key-exchanges need precise time synchronization between devices. For PIKE, a time-shift of several minutes is still acceptable.

Other mechanisms to perform a key-exchange are based on near field communication (NFC) technology. In the Bluetooth specification [10], NFC is mentioned as a possible mechanism to pair devices. Suomalainen et al. [11] show that NFC can be used for the negotiation of keys for other network types, mostly because direct proximity (usually 1-10 cm) is necessary for successful communication in NFC. Some authors describe the inherent security of NFC for man-in-the-middle attacks [12], but show that eavesdropping can be done easily. Others point out possible security and privacy breaches, for example by using a unique id [13]. Since NFC is usually used for short-range communication, Francis et al. [14] describe a relay attack that can be used to circumvent this limitation. PIKE does not establish security on a device-to-device basis, but takes a user-centric approach. Since the keys are already exchanged when the interaction takes place, it is immune to attacks that rely on a specific communication technology. As a result, the shared exchanged key created by PIKE can be used for any communication technology, including NFC. Nevertheless, establishing group keys using NFC still needs physical interaction between devices which, as mentioned before, does not scale. ProxiMate [15] is another key exchange protocol that retrieves a shared key from RF signals. To obtain a key, it is necessary to put the pairing devices into physical proximity, similar to NFC. According to the authors, the protocol is resistant to attackers that are more than 6.2 cm away (at 2.4 GHz). The protocol could be used to exchange a shared key between devices, but it might be impossible to put all users' devices in the necessary proximity, if the number of devices is too large. PIKE does not have constraints on the number of devices, also supports higher numbers, and is able to create group and user-level keys in a completely automated fashion. In SPATE [16], a small group of users is able to exchange a key by comparing hash codes (the so-called T-Flags). This enables SPATE to establish a key for secure interaction at additional costs, i.e., all users have to recognize and compare the T-Flag images. To start the key exchange process, SPATE requires the initiating device to scan a bar-code from all other devices' displays (retrieving the devices' network addresses). Since SPATE also needs the number of group members typed in manually, SPATE and related approaches like Seeing-is-believing [17] and GAnGS [18] need manual configuration, while PIKE enables an automatic key exchange, relying on the friend relationship of online collaboration services.

Many existing key-exchange mechanisms need an Internet connection during the key-exchange phase and they use a different service provider for authentication purposes. Examples are OpenID [19] and OAuth [20, 21]. While OpenID allows the user to use one account for several different services and service providers, OAuth creates an access token, mostly used by applications that want to access a service on behalf of the user. A service that uses

OpenID needs to trust the service that provides the account creation and authorization. The latter is therefore a trusted third-party that is actively involved in the authorization process. In contrast, using the OAuth process, the user actively grants rights from a service to another one. Usually, the granted rights are displayed to the user and must be confirmed manually. PIKE does not need a third party for account verification during the interaction, but uses a third-party service directly to exchange a shared key beforehand. Thereby, it retrieves and stores the exchanged key and thus, PIKE does not require an Internet connection later on.

7. Conclusion

Online services have become an important and ubiquitous mediator of many human interactions. In the virtual world, they enable secure interactions by mediating the access to shared resources. PIKE extends the support provided by online services to enable non-mediated secure face-to-face collaboration. PIKE is configuration-free and broadly applicable to different scenarios ranging from typical small-scale meetings up to large scale conferences and events. As indicated by the results of our evaluation, PIKE and its variant P2PIKE is (at least) as secure as the service used to enable resource sharing and exhibits an acceptable latency of at most 2 synchronization cycles. Finally, depending on the resource sharing API provided by the underlying service, PIKE can scale well. Currently, we are integrating PIKE as a security mechanism into the GAMBAS middleware. There, PIKE is used to limit the access to context information that is gathered by mobile devices.

Acknowledgments

This work is supported by UBICITEC e.V. (European Center for Ubiquitous Technologies and Smart Cities) and GAMBAS (Generic Adaptive Middleware for Behavior-driven Autonomous Services) funded by the European Commission under FP7 with contract FP7-2011-7-287661.

References

[1] J. Al-Muhtadi, R. Hill, R. Campbell, M. Mickunas, Context and location-aware encryption for pervasive computing environments, in:

Pervasive Computing and Communications Workshops 2006 (PerCom'06), 2006.

[2] A. Tripathi, T. Ahmed, D. Kulkarni, R. Kumar, K. Kashiramka, Context-based secure resource access in pervasive computing environments, in: Pervasive Computing and Communications Workshops 2004 (Percom'04), 2004.

[3] D. Schürmann, S. Sigg, Secure communication based on ambient audio, Mobile Computing, IEEE Trans. on.

[4] P. Robinson, M. Beigl, Trust context spaces: An infrastructure for pervasive security in context-aware environments, in: Security in Pervasive Computing, 2004.

[5] L. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, H.-W. Gellersen, Smart-its friends: A technique for users to easily establish connections between smart artefacts, in: Ubicomp 2001: Ubiquitous Computing, 2001.

[6] D. Bichler, G. Stromberg, M. Huemer, M. Löw, Key generation based on acceleration data of shaking processes, in: Conference on Ubiquitous computing (UbiComp), UbiComp '07, 2007.

[7] R. Mayrhofer, H. Gellersen, Shake well before use: Authentication based on accelerometer data, in: Pervasive Computing, 2007.

[8] S. Sigg, D. Schürmann, Y. Ji, Pintext: A framework for secure communication based on context, in: Mobile and Ubiquitous Systems: Computing, Networking, and Services, 2012.

[9] R. Mayrhofer, The candidate key protocol for generating secret shared keys from similar sensor data streams, in: Security and Privacy in Adhoc and Sensor Networks, 2007.

[10] Specification of the bluetooth system, core version 2.1 + edr (July 2007). `http://www.bluetooth.org/Technical/Specifications/adopted.htm`

[11] J. Suomalainen, J. Valkonen, N. Asokan, Security associations in personal networks: a comparative analysis, in: Security and privacy in ad-hoc and sensor networks, ESAS'07, 2007.

[12] E. Haselsteiner, K. Breitfu, Security in near field communication (nfc), in: Workshop on RFID Security, 2006.

[13] G. Madlmayr, J. Langer, C. Kantner, J. Scharinger, Nfc devices: Security and privacy, in: Availability, Reliability and Security. ARES 08, 2008.

[14] L. Francis, G. Hancke, K. Mayes, K. Markantonakis, Practical nfc peer-to-peer relay attack using mobile phones, in: Radio Frequency Identification: Security and Privacy Issues, RFIDSec'10, 2010.

[15] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, N. Mandayam, Proximate: proximity-based secure pairing using ambient wireless signals, in: Mobile systems, applications, and services, MobiSys '11, 2011.

[16] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. M. McCune, et al., Spate: small-group pki-less authenticated trust establishment, in: Mobile systems, applications, and services, MobiSys '09, 2009.

[17] J. M. Mccune, A. Perrig, M. K. Reiter, Seeing-is-believing: Using camera phones for human-verifiable authentication, in: IEEE Symposium on Security and Privacy, 2005.

[18] C. hsin Owen Chen, C. wei Chen, C. Kuo, Y. hao Lai, J. M. Mccune, A. Studer, Gangs: Gather authenticate n group securely, in: Mobile Computing and Networking (MobiCom'08), 2008.

[19] O. Foundation, Openid authentication 2.0 - final (December 2007). URL `http://specs.openid.net/auth/2.0`

[20] E. Hammer-Lahav, The oauth 1.0 protocol (April 2010).

[21] D. Hardt, The oauth 2.0 authorization framework, draft-ietf-oauth-v2-31 (July 2012).