PIKE: Enabling Secure Interaction with PIggybacked Key-Exchange

Wolfgang Apolinarski, Marcus Handte, Muhammad Umer Iqbal, Pedro José Marrón
Networked Embedded Systems
Universitaet Duisburg-Essen
Duisburg, Germany
firstname.lastname@uni-due.de

***Abstract*—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 interaction by controlling access to shared resources. Yet relying on them to support synchronous direct interactions, such as face-to-face meetings, might be suboptimal as they require reliable online connectivity and even then often introduce delays. A much more efficient way of co-located resource sharing is the use of local communications, such as ad-hoc WiFi. Yet setting up the necessary encryption and authentication mechanisms is often cumbersome. In this paper, we present PIKE, a key exchange protocol that minimizes this configuration effort. PIKE piggybacks the exchange of keys on top of an existing service infrastructure. To support encryption or authentication without Internet connection, PIKE relies on triggers for upcoming personal interactions and exchanges keys before they take place. To evaluate PIKE, we present two example applications and we perform an experimental as well as an analytical analysis of its characteristics. The evaluation indicates that PIKE is broadly applicable, scales well enough to support larger events and provides a level of security that is (at least) comparable to the one provided by the underlying service.**

***Keywords*-Key-exchange, online services, smart phones**

I. 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 resources can then be tied to different sets of identifiers such as friend lists in Facebook or circles in Google+. To control the access to resources, the collaboration tools use encrypted communication such as TLS and they 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.

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

The success of these collaboration tools indicates that this mediation model can effectively support secure remote interaction. Yet, using them for personal interactions that take place in the physical world such as collaboration in a face-to-face meeting can be suboptimal. The main reason for this is that such interactions involve multiple partners that interact with each other *at the same time and place*. Thus, the issues arising from a remote connection such as higher response times or intermittent connectivity cannot be hidden by caching and synchronization. To mitigate this, it is possible to support personal interactions by means of local communication. However, to provide a similar level of security, this requires encryption and authentication mechanisms that must be configured. Without the mediating online tool, this requires the interaction partners to manually exchange authentication or encryption keys which does not scale and is 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 non-mediated personal interaction. The basic idea behind PIKE is to piggyback the exchange of keys on top of the existing service infrastructure in a proactive manner. Thereby, we eliminate the need for manual configuration as well as Internet connectivity when the interaction takes place.

A prototype application scenario for PIKE is a business meeting for which invitations are shared securely using Google Calendar. When detecting the invitations, PIKE automatically exchanges keys over the Internet using the Google infrastructure and stores them. When the meeting takes place, the keys can be used to establish secure communication among the participants' devices or to authenticate participants without requiring any Internet connectivity.

The contribution of this paper is threefold. First, we introduce PIKE as an approach for enabling non-mediated, secure and configuration-free personal interactions. 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.

II. 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. The exchanged keys can then be used by applications to secure the local communication, e.g. by means of encryption or message authentication.

A. System Model and Assumptions

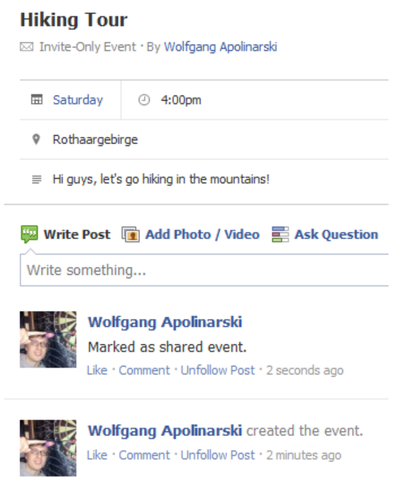
The technical basis for PIKE are mobile *devices*, such as phones, tablets or laptops, used to share *resources* with each other remotely through a *network*, mediated by a *service*. Regarding these four building blocks we assume:

- *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 intolerable international roaming fees.
- *Devices:* The device of the user is able to access the service regularly through the network. For this, the service uses a mobile application that synchronizes the changes to the 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. In addition, we assume that some resources are used to plan an upcoming personal interaction and thus, provide time information. Examples for such resources are a calendar entry or a message indicating an upcoming meeting[2].

B. Design Rationale and Goals

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

[2]It is noteworthy that this assumption can be dropped. However, if the shared resources do not indicate the timing of an upcoming interaction, the key usage cannot be fully automated. Instead, the creator of a resource will have to initiate the usage manually (e.g. by pressing a button) so that the collaborators will be able to benefit from automation.


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

Figure 1. Shared Trigger Resource in Facebook

- *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, the key exchange with PIKE must be secure. Given that the partners are already using the service through remote interaction in order to share resources in a secure manner, the use of PIKE to secure personal interaction should result in (at least) the same level of security as the mediation through the 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 personal interaction has been planned. The key should therefore be exchanged within a few seconds.
- *High Scalability:* As the personal interaction 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.

C. Key Exchange Protocol

As explained in Section II-A, 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 API provided by the service is used for resource synchronization. Every time the resources change, PIKE starts to analyze the resources to detect changes that trigger a key exchange. An example could be the creation of an event on Facebook that has been shared with a set of friends (c.f. Figure 1). Once a change has been found, the key exchange takes place. For this, each partner shares a key with the creator of the resource. The creator of the resource shares his key with each of the partners. For this, all partners use the secure resource 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 the interaction takes place, these keys can be used to enable secure communication among the devices of the partners. To do this, PIKE extracts the keys from the resource and makes them available during the interaction.

Figure 2 depicts the resulting logical protocol flow which is executed remotely by all partners before the personal interaction takes place. Conceptually, PIKE involves three entities, namely the device of the interaction partners creating the resource (the initiator), the devices of the remaining partners (the participants) and the 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. 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. Then, a key is created and shared 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 synchronized with the participants. At this point, each of them retrieves the key and stores it for later use. In order to enable user-level authentication, the participants create a new key on their own and share it with the initiator. 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 initiator takes the key and stores it for later use.

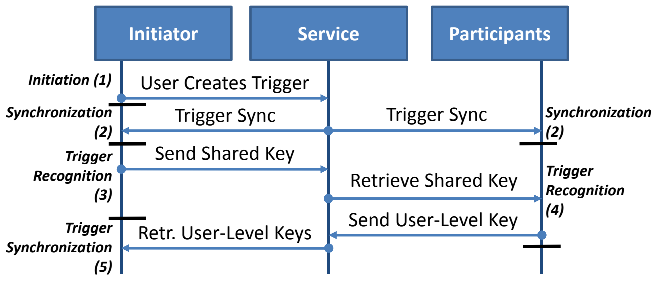


Figure 2. PIKE Logical Protocol Flow

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 personal interaction takes place, these keys (or a derived key) can be used to enable group communication as well as private communication and user-level authentication between the initiator and the participants. The latter can then be used to bootstrap further keys (e.g. using Kerberos), however, the applications described later on do not require this.

To speed up the sequential logical protocol flow described previously, it is possible to execute the steps 3 and 4 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.

III. Implementation

To validate the concepts of PIKE, we have implemented it 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 II-A 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).

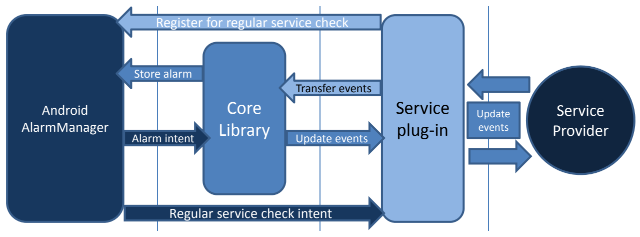
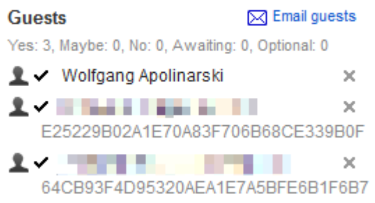


Figure 3. PIKE Architecture for Android



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 the event in Google Calendar

A. *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 an upcoming personal interaction 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 parcelable object which can be passed to the scheduler provided by Android (i.e. the AlarmManager) that 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 the latter, it uses a key derivation function and a challenge-response mechanism using an HMAC (Keyed-Hash Message Authentication Code). For the former, the core library is capable of establishing a WLAN using the tethering capabilities of the Android operating system[3]. If this service is requested by an application, the group key exchanged via PIKE is used to setup a WPA2 protected network. To do this, all devices taking part at the personal interaction generate an SSID as well as a passphrase from the group key. The device of the initiator 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]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.

B. *Google Calendar Plug-in*

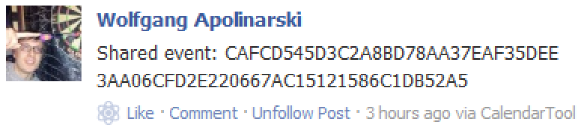
Google Calendar is one of the most used business services for the management of shared calendars. At work, the calendars can be used to be informed about the absence and availability of co-workers. Meetings can then be planned efficiently and co-workers can be invited to join via the service. Additionally,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

C. *Facebook Event Plug-in*

Our second plug-in uses Facebook. This online social network service is used mostly for private interaction. 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

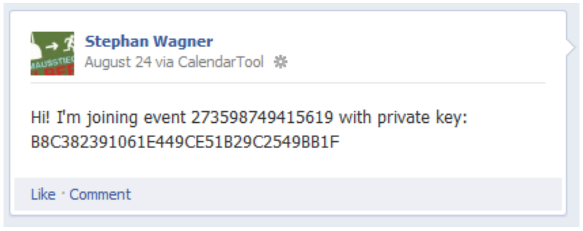


Wolfgang Apolinarski

Shared event: CAFCD545D3C2A8BD78AA37EAF35DEE
3AA06CFD2E220667AC15121586C1DB52A5

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

Figure 5. Shared key, posted at the event's wall



Stephan Wagner
August 24 via CalendarTool

Hi! I'm joining event 273598749415619 with private key:
B8C382391061E449CE51B29C2549BB1F

Figure 6. User-level key, posted at a participant's wall

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 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 constraints the access to the event's initiator (see Figure 6). The initiator can then retrieve the participants' keys by going through their walls.

IV. 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 performs cooperative context recognition using the devices of the participants of a small- to medium-sized meeting. To do this, it requires a low-latency, reliable and private network. The second application provides registration services for large scale events. For this, the application requires secure user-level authentication but it cannot rely on Internet connectivity. Both applications are canonical examples that show the capabilities of PIKE in different conditions.

A. Privacy-preserving Speaker Recognition

As an example for using PIKE during typical personal interactions at work, we developed a cooperative context recognition application using our context recognition system for mobile devices [22]. The goal of this application is to continuously identify the speaker during a meeting for example, to annotate meeting minutes automatically. While this can also be done on a single device by means of voice profiles [23], cooperative recognition is significantly simpler. Under the assumption that all participants of the meeting are carrying a mobile device, we can determine the speaker by analyzing the different audio streams recorded by the participant's devices. For this, a mobile application running on each device continuously captures sound samples and computes the relevant features like the RMS power value. The features are then transmitted to the initiator's device which uses them to determine the (most likely) speaker for the interval. Once the speaker has been determined, the initiator's device broadcasts the result to the participants which can then use them for annotation and live visualization.

Clearly, for an application like the one described above, the samples and results that are exchanged between the participants of the meeting may contain private information, i.e. the recording of the meeting should not be overheard by eavesdroppers. By automatically exchanging keys via a calendar entry and using them to set up a private wireless LAN during the meeting, we can effectively limit the distribution of samples and results to the actual participants. Similarly, by using local communication as opposed to indirect Internet-based communication, we can rely on the fact that the network will be reliable and fast. This can drastically simplify the application since we do not have to consider issues such as time-synchronization between the devices or retransmissions of high-volume data. Instead, we can simply assume that data will be received quickly and data loss will be infrequent and thus, insignificant.

B. 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 user 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 the keys. The keys are then used to provide a user-level authentication at the start of the conference in a fully automated process without the need for an Internet connection.

V. Evaluation

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

A. 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. Thus, PIKE does not add any additional manual step to this process and is therefore fully automatic.

B. 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 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. Google) itself. Since all exchanged keys are stored in the service provider's database, it would be easy to retrieve them from there. Nevertheless, the service provider itself is not physically present during the personal interaction. As a result, it cannot use the exchanged keys 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 such that it might be possible to be physically present during the interaction, service providers can be combined. The trigger condition 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 could not overhear the personal interaction, 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 at the interaction (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 handled encrypted only, 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 used by the actual implementation.

If one of the participants that is taking part in the personal interaction is malicious, its malicious actions can (possibly) be revealed by the initiator using the user-level keys. The initiator can use them to distinguish the messages from each participant which could result in the attacker's identification. The initiator can then establish a new key by using the user-level keys of all non-malicious interacting partners using an ordinary key distribution protocol. After the re-keying, the group can continue to communicate securely.

Additionally, the user-level keys can be used to establish an encrypted private communication channel between two users in the personal interaction 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, the initiator's device can be used as a trusted third party that verifies the authentication of the other users. It can use the user-level key to certify the identity of the users to each of the participating devices. The exchange of a key over the initiator will then enable the devices to communicate with each other encrypted with this newly derived key.

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

C. Low Latency

Using two Samsung Galaxy Nexus (2x1.2 GHz ARM CPU, 1 GB RAM) mobile phones that were connected to the Internet via WiFi, 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 III. All

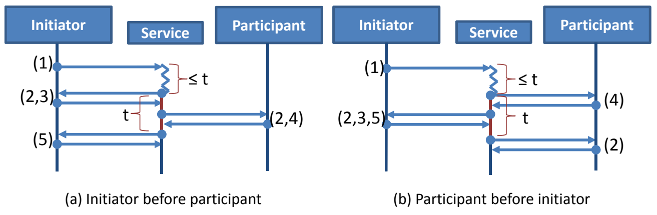


Figure 7. Time needed to perform PIKE, with synchronization interval t

measurements shown in this section 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 could be 60 minutes, i.e. the mobile app checks the service every hour. As depicted in Figure 7, 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 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 $2t$.

Regarding our implementations, the actual message flows are similar to the logical protocol flow presented in Figure 2. 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 each participant, it is always 3. We also measured the necessary time that is needed to perform the PIKE protocol flow. We supposed that the user creates the event in the Google Calendar (either using a mobile phone or the web-interface) manually (i.e. performed step 1). Steps 2 and 3 took 535 ms, since the initiator needs to perform step 5 (255 ms with 2 participants) as well, this gives a total number of 790 ms for the API calls. Each participant needs to execute the steps 2 and 4, this takes 583 ms. As a result, we see that the prototypical implementation of PIKE with Google Calendar exhibits a latency of less than a second. Depending on t, also spontaneous meetings can be supported, nevertheless, a typical value for t like 60 minutes constraints this to meetings known 2 hours in advance.

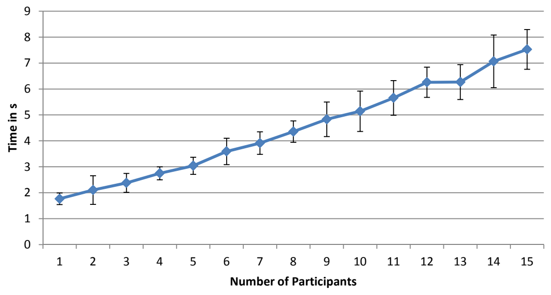


Figure 8. Retrieving user-level keys one-by-one from Facebook

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 suppose that the user creates the event from within Facebook (step 1). The average time necessary to perform the API calls for the steps 2 and 3 is 2193 ms. Step 5 is discussed in the Section *Scalability*. The participants need to perform step 2 and 4, this takes 3067 ms. 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.

D. Scalability

For PIKE, scalability depends on the used service. Using Google Calendar, an event contains all information, including user-level keys (comments). A single API call returns the necessary information. The length of the returned message increases with each participant slightly, 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

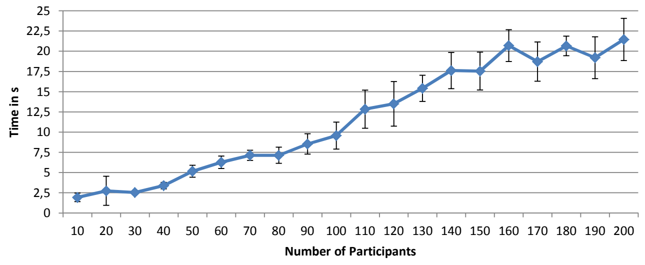


Figure 9. Retrieving user-level keys using Facebook's batch API

8, showing a clear linear increase, 100 measurements per point, error bars show the standard deviation ($\pm\sigma$)) or batch requests (Figure 9, 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 1765 ms (1 participant) to 7526 ms (15 participants) for the single requests and from 1935 ms (10 participants) to 21453 ms (200 participants) 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.

VI. Related Work

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

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 foreign 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 in- and outdoors. Using context data coming 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.

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.

VII. 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 personal interaction. 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 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 core security mechanism into the GAMBAS middleware. There, PIKE is used to limit the access to context information that is gathered by mobile devices. Thereafter, we plan to investigate ways dealing with malicious initiators. A simple way is to share additional keys between participants, however, without further concepts, this approach does not scale well.

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, and M. Mickunas, "Context and location-aware encryption for pervasive computing environments," in *Perv. Comp. and Comm. PerCom Workshops 2006. 4th Annual IEEE Int. Conf. on*, march 2006.

[2] A. Tripathi, T. Ahmed, D. Kulkarni, R. Kumar, and K. Kashiramka, "Context-based secure resource access in pervasive computing environments," in *Perv. Comp. and Comm. Workshops. Proc. of the 2nd IEEE Ann. Conf. on*, march 2004.

[3] D. Schürmann and S. Sigg, "Secure communication based on ambient audio," *Mobile Computing, IEEE Trans. on*, 2011.

[4] P. Robinson and 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, and 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, and M. Löw, "Key generation based on acceleration data of shaking processes," in *Proc. of the 9th int. conf. on Ubiquitous computing*, ser. UbiComp '07, 2007.

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

[8] S. Sigg, D. Schürmann, and Y. Ji, "Pintext: A framework for secure communication based on context," in *Mobile and Ubiquitous Systems: Comp., 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 Ad-hoc and Sensor Networks*, 2007.

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

[11] J. Suomalainen, J. Valkonen, and N. Asokan, "Security associations in personal networks: a comparative analysis," in *Proc. of the 4th European conf. on Security and privacy in ad-hoc and sensor networks*, ser. ESAS'07, 2007.

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

[13] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger, "Nfc devices: Security and privacy," in *Availability, Reliability and Security. ARES 08. Third Int. Conf. on*, march 2008.

[14] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, "Practical nfc peer-to-peer relay attack using mobile phones," in *Proc. of the 6th int. conf. on Radio freq. identification: security and privacy issues*, ser. RFIDSec'10, 2010.

[15] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Mandayam, "Proximate: proximity-based secure pairing using ambient wireless signals," in *Proc. of the 9th int. conf. on Mobile sys., app., and services*, ser. 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 *Proc. of the 7th int. conf. on Mobile sys., app., and services*, ser. MobiSys '09, 2009.

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

[18] C. hsin Owen Chen, C. wei Chen, C. Kuo, Y. hao Lai, J. M. Mccune, and A. Studer, "Gangs: Gather authenticate n group securely," in *Proc. of the ACM Annual Int. Conf. on Mobile Comp. and Networking (MobiCom)*, 2008.

[19] O. Foundation, "Openid authentication 2.0 - final," December 2007. [Online]. Available: 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.

[22] M. Iqbal, M. Handte, S. Wagner, W. Apolinarski, and P. Marron, "Enabling energy-efficient context recognition with configuration folding," in *Perv. Comp. and Comm. (PerCom), IEEE Int. Conf. on*, march 2012.

[23] H. Lu and Brush, "Speakersense: energy efficient unobtrusive speaker identification on mobile phones," in *Proc. of the 9th intl. conf. on Pervasive computing*, 2011.