SPATE: Small-group PKI-less Authenticated Trust Establishment*

Yue-Hsun Lin† Ahren Studer§ Hsu-Chun Hsiao§ Jonathan M. McCune§ King-Hang Wang†
Maxwell Krohn§ Phen-Lan Lin◦ Adrian Perrig§ Hung-Min Sun† Bo-Yin Yang‡

† National Tsing Hua University § Carnegie Mellon University ‡ Academia Sinica ◦ Providence University

ABSTRACT

Establishing trust between a group of individuals remains a difficult problem. Prior works assume trusted infrastructure, require an individual to trust unknown entities, or provide relatively low probabilistic guarantees of authenticity (95% for realistic settings). This work presents SPATE, a primitive that allows users to establish trust via device mobility and physical interaction. Once the SPATE protocol runs to completion, its participants' mobile devices have authentic data that their applications can use to interact securely (i.e., the probability of a successful attack is 2^{-24}). For this work, we leverage SPATE as part of a larger system to facilitate efficient, secure, and user-friendly collaboration via email and file-sharing services. Our implementation of SPATE on Nokia N70 smartphones allows users to establish trust in small groups of up to eight users in less than one minute. The two example SPATE applications provide increased security with no overhead noticeable to users once keys are established.

General Terms: Design, Human Factors, Security

Categories and Subject Descriptors: C.2.0 [Computer – Communication Networks]: General – *security and protection*; D.2.2 [Software Engineering]: Design Tools and Techniques – *modules and interfaces*; H.1.2 [Models and Principles] User/Machine Systems – *human factors*

1. INTRODUCTION

A decentralized security infrastructure—one that allows informally organized groups of colleagues to communicate securely—remains a great idea in theory. In practice, the idea is not much more accessible now than when it was introduced decades ago, dogged by the usual concerns: how to exchange keys, how to manage keys, how to integrate with existing applications, how to configure security policies, etc.

Consider the example of secure email. One of the most mature systems that provides encrypted, authenticated email exchange is PGP, first introduced in 1991. PGP allows arbitrary pairs of users to exchange email securely, without the need for centralized administrators. However, even as the software becomes streamlined and more popular, non-expert users still have difficulty adopting it, struggling with key management and configuration of security policies [48].

Another important example is file-sharing. Users without centralized infrastructure like NFS or AFS still wish to share files selectively with their friends, while hiding those files from others. Yet recent work shows that popular file-sharing utilities make configuring security policies difficult, and that many users inadvertently expose private files to strangers [17]. Indeed, configuring access-control lists might be too much to ask of casual users.

In light of these security problems, a growing trend that offers promise is *device mobility*—now more prevalent than ever with the proliferation of increasingly sophisticated mobile phones. This paper introduces the SPATE protocol, and the SPATE system built on top of it. The use case for SPATE is a common one: a small ad-hoc group meets in person and wishes to continue collaboration remotely, whether via secure email or secure file-sharing. Using current tools, even this simple scenario is vexing for the average user, requiring baroque, user-visible key exchange protocols and confusing access control decisions. The SPATE system, by contrast, takes advantage of device mobility and face-to-face meetings to simplify future communication considerably.

The foundation of the SPATE system is the SPATE protocol, which runs on mobile phones with the objective of sharing authenticated data among members of a small group. Participants initiate the protocol by invoking an application on their phones and indicating the number of people in the group. The phones then exchange information via Bluetooth. The danger in this scenario is a Man-in-the-Middle (MitM) attack, by which a nearby adversary can inject inauthentic data into the exchange. To prevent such an attack, at the end of the protocol, all mobile devices check that they received the correct number of data items and display a visual hash function [13, 33] computed over the exchanged data. The participants check that all devices agree on the hash. If both checks succeed, then the group participants

*This research was supported in part by grants DAAD19-02-1-0389 and MURI W 911 NF 0710287 from the Army Research Office, and support from the iCAST project, National Science Council, Taiwan under the Grant NSC97-2745-P-001-001. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, iCast, or the U.S. Government or any of its agencies.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MobiSys'09, June 22–25, 2009, Kraków,Poland.
Copyright 2009 ACM 978-1-60558-566-6/09/06 ...$5.00.

have guarantees that: (1) each participant contributed exactly one data element to the collective; (2) that no one outside the group contributed data; and (3) the data distributed is exactly what each individual user's device intended.

The SPATE exchange is agnostic to the type of data exchanged. In addition to flexibility (i.e., users can exchange any data), this can improve security. One obvious use for such a protocol is to exchange public keys, enabling subsequent secure remote collaboration. SPATE is also designed such that long-term secrets need not be stored on mobile devices. This design provides improved security properties when compared to other key exchange protocols (Section 9). Without the corresponding private key on the device, the loss of a mobile device has zero impact on security; the private key remains securely on the user's workstation at home. Of course, without the private key on the phone, there is the drawback that users cannot perform any operations that require the private key, until they can access their workstation.

Device pairing has recently attracted a significant amount of interest from the research community, fueled by the proliferation of wireless mobile devices. Prior work addresses similar key exchanges, but either assumes a public key infrastructure [6,8,23,24,41,42,44], cumbersome key-exchange protocols [5], is vulnerable to malicious bystanders [2], or are restricted to two-party exchanges [3,7,9,27,30,32,38–40,45]. Other works offers mechanisms optimized for large groups of 10 to 30 people [10]. This work focuses on small groups where users can accurately count the group size [25]: eight or fewer. Assuming group size fits a Zipf distribution, the majority of groups will be within the range covered by the SPATE protocol.

This paper presents an implementation of the SPATE protocol as part of a larger SPATE system, filling in the details of how to generate cryptographic keys, how to move keys between one's PC and one's phone, how to exchange keys, and most importantly, how to build real applications that use the exchanged keys. We present two applications: secure email and secure file-sharing. Both exploit device mobility to configure useful secure-by-default policies, without requiring any expert decisions from the users. In the email example, a Thunderbird Mail plug-in enables encrypted and signed email communication by default for email sent among group members after the meeting. The file-sharing application provides a shared folder among all group members, which affords read and write access to group members and denies access to all others. Both applications have the crucial property that security does not require undue inconvenience. We anticipate that our approach will provide a foundation for bootstrapping secure communication for current and future applications.

In summary, this paper offers the following contributions: (1) a description of the SPATE protocol for securely exchanging data among members of a small group; (2) an implementation of the SPATE system on mobile smartphones; and (3) two realistic applications that demonstrate how SPATE enables practical secure-by-default operation.

2. PROBLEM DEFINITION

When meeting face to face, a group can trust that things they see and hear from the other group members have not been modified by a malicious party. Once the group disperses, members would like to continue to have that same level of trust for intra-group communication. Collecting authentic data (i.e., public keys and application-specific data) from other members of the group can facilitate such secure communication. Most security applications are already designed to handle public keys (e.g., X.509 certificates), while other applications can leverage public keys to setup shared keys or passwords within the group. However, for ease of use, some applications may want to share additional information (e.g., email or IP addresses to simplify contacting other members or sharing data within the group). Once groups have a way to exchange authentic data in person, secure collaboration is possible without requiring members to trust a third party.

According to Chen et al. [10], an exchange of authentic information within a group produces a set of data that must fulfill the following three properties:
1. Consistent: Every group member acquires the same set of data.
2. Exclusive: Only group members' data is in the set.
3. Unique: Each member only contributes one data element to the set.

In addition, the exchange protocol should have limited expectations for the users. Humans are impatient and are inaccurate when comparing numbers [45]. To avoid frustrating users, an exchange protocol should run quickly with only a small number of interactions (e.g., taking pictures of or shaking devices) between group members (i.e., for n members a total of $O(n)$ total interactions). To avoid human errors, the exchange should facilitate user-friendly comparisons, rather than requiring several users to compare hexadecimal digits.

2.1 Assumptions

In this work, we make assumptions about the hardware and software available on members' mobile devices, the absence of malicious software (malware), the probability of human error, and the user's diligence when it comes to securing private asymmetric keys.

We assume users' mobile devices are equipped with Bluetooth radios, a color display, a camera, and an installation of our SPATE software. Commodity smartphones can provide all of these hardware requirements.

We assume that group members' mobile devices and workstations (e.g., desktop or laptop) are free of malware. If malware existed on either system, a malicious party could subvert any data distributed or collected during a SPATE exchange. Malware is a serious threat, but is orthogonal to the authentic exchange of data in groups.

We also assume that humans can count and compare images correctly within small groups of 2 to 8 members. Prior studies have shown that users can accurately perform such tasks in small groups [25]. However, in groups of more than 10 members, counting errors become more common.

We assume individuals keep their private keys secret. If a user were to publish their private key or share it with other users, that key no longer provides authentication. For example, if user U shares the private key K_U^{-1} with other individuals, an email signed using key K_U^{-1} provides no guarantee that U was the actual source of the email. The need to keep the private key secret is not unique to systems that use physical interaction to establish trust.

2.2 Trust Model

SPATE's trust model is built upon physical interactions via mobile devices. Having exchanged messages via the de-

vices, user U_a with mobile device M_a trusts a message from M_b if two conditions are satisfied: 1) U_b is physically located in the same place as U_a; 2) the message (or an unforgeable representation of the message) displayed on M_a's screen is identical to the one on M_b. Users can visually verify both conditions. Therefore, users only trust messages they have directly received but not those relayed by someone else.

We now briefly contrast SPATE's trust model with Public Key Infrastructure (PKI) and PGP's web-of-trust.

PKI A PKI certificate authority issues certificates that bind users' digital identities to public keys. The certificates are *unable* to bind a user's *physical identity* to a public key. When exchanging public keys in a PKI, a user needs to present his certificate as proof of the authenticity of an exchanged public key. The security property relies on a shared trusted authority, which may not exist in many settings.

PGP In a PGP key-signing party, user U_a signs a PGP certificate that binds a public key with another user U_b's identity—if U_a believes the identity claimed by U_b. If user U_c trusts U_a, U_c will accept the U_a-signed certificate as a credential for U_b, without interacting with U_b.

SPATE is different in that we do not trust any third party. We assume a stronger trust model where users only trust a public key acquired through direct physical interaction with another user.

2.3 Attacker Model

Attackers can eavesdrop, intercept, and manipulate any message transmitted over the Internet and wireless networks. The attacker can also form a coalition of several group members (insiders), who have control over their own private keys and devices.

The attacker's goal is to manipulate the exchanged data without being detected. Manipulation includes deletion of users' data or modification of existing data. Note that the goal of colluding attackers is to manipulate the data of benign users. Modifying data of other colluding attackers is not considered an attack. An attacker can also contribute bogus data (e.g., another user's public key). However, without the corresponding private key, the impersonator will be unable to perform the operations necessary to assume the victim's identity online (i.e., decrypt or sign data with the appropriate private key).

The attacker can also jam the wireless channel or insert junk data as part of a denial-of-service (DoS) attack. However, we do not consider DoS attacks because they are detectable (users can tell if the protocol aborts or gets stuck abnormally) and cannot alter any data being exchanged.

We consider computationally bounded attackers who cannot break basic cryptographic primitives. Hence, keys cannot be recovered from signatures, and there is a hash function $h()$ that for all intents and purposes behaves as a random oracle. But an attacker can brute-force solutions to "small" problems, such as finding M where $h(M)$ ends with any given 24 bits.

3. BACKGROUND ON HASH COMPARISONS

Protocols that operate with collocated users often require individuals to compare checksums to ensure successful setup or authenticity of exchanged data [26, 28, 29]. For such comparisons, researchers would like a mechanism that is simple for humans, computationally efficient, and has a quantifiable level of security.

Traditionally, such protocols require users to compare a sequence of hexadecimal digits. Hexadecimal digits are computationally efficient to generate and contain a fixed amount of entropy (4 bits per digit). However, humans trying to quickly compare digits often make mistakes (e.g., confuse an 8 for a 0) [45].

Given humans' inability to accurately and quickly compare digits, researchers have proposed using text [15, 18] or visual [13,33] representations of these checksums. The "Loud and Clear" system [18] expresses hashes as syntactically correct sentences, while the UIA system [15] expresses hashes as sequences of dictionary terms (e.g., "meals – abut – yuck"). The entropy of the words is easy to calculate given the size of the dictionary from which the sequence of words is selected. In addition, looking up words in a dictionary is computationally efficient. However, comparison of words may still require significant user effort as a quick glance at the words may not suffice to facilitate an accurate comparison.

Humans are good at quickly detecting differences in images, so visual representations of the checksums present one promising comparison mechanism. "Random Art" [33] and "Flag" [13] express hashes as visual images. Random Art contains an unknown amount of entropy, making security analysis difficult, and is computationally expensive, requiring around ten seconds to generate an image on a mobile device [10]. Flags [13] represent an efficient alternative. However, their images contain limited entropy and lack reference points. Such reference points are important when comparing Flags across mobile devices where screens are often rotated.

3.1 T-Flags for Hash Comparison

For this work, we have developed a new scheme, T-Flags, which contains nearly twice the entropy of the original Flag, includes a visual cue to help users quickly determine the proper orientation during comparison (Section 6 gives examples), and only requires around 60 ms to generate on a mobile phone. In this work, we limited ourselves to 3 bits for 8 colors per rectangle.[1] With 8 rectangles per T-Flag, a T-Flag contains 24 bits of entropy.

To select 8 maximally distinct colors, we need to select colors that appear different independent of display settings (e.g., contrast or brightness) or color blindness. Based on human perception, Glasbey et al. deduce 11 maximally distinct colors [16]. To address color blindness, we eliminated Green. We thus select the following 8 colors: Black, Gray, White, Yellow, Light Pink, Red, Blue, and Brown.

4. SPATE

SPATE is a system that provides a foundation of trust for secure applications. SPATE relies on visual channels and physical interactions rather than pre-existing trusted infrastructure (i.e., PKI) or transitive trust (i.e., PGP) to authenticate data. Our key insight is the use of mobile devices and human interaction to convert physical interaction into digital trust. A group of users who successfully complete the SPATE protocol are guaranteed to have identical and

[1]Ellison and Dohrmann [13] use 6 bits representing 64 colors per rectangle, but with so many colors slight differences in shade may lead to errors during comparison.

authentic copies of data. The data can be anything, e.g., public keys, IP addresses, public-key certificates, or email addresses. The authenticated information can be the basis for a host of different secure applications. For example, to send an encrypted message, the sender needs to know the correct public key and email address of the receiver.

4.1 SPATE Protocol Overview

The SPATE protocol is designed to allow a group of users that meet in person to exchange data which later forms the basis of trust for an application. People often carry their phones or other resource-constrained mobile devices, but may leave their main workstation (i.e., desktop or laptop) elsewhere. As such, we have designed the SPATE exchange to run on mobile devices because they will be present when people physically meet. When security applications and the SPATE exchange are run on different devices, a mechanism is needed to transfer the data between the device and the machine. SPATE thus consists of three steps to allow operation of our secure applications: 1) the one-time creation of application dependent data and imprinting the data on the mobile device, 2) exchange of authenticated data with other users, and 3) retrieval of data from the mobile device. Figure 1 depicts these three steps.[2]

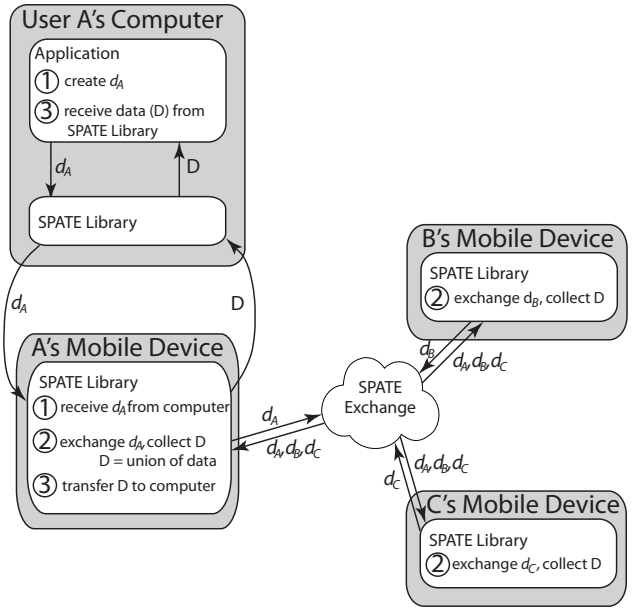

Figure 1: Steps associated with a SPATE exchange between Users A, B, and C. Shown from the perspective of User A.

① **Creation and Imprinting of Data** During the creation and imprinting of data, a workstation (laptop or desktop) is used to create and transfer a user's data. For our prototype, the user's data (d) is a self-signed certificate containing the user's name, email address, public key, and other application-specific data. In general, SPATE users can exchange arbitrary data of their choosing. To securely transfer d from the computer to the mobile device we use standard Bluetooth pairing techniques [28] to setup a secure channel between the two. Initially, pairing requires the user to copy a passkey from the computer to the device. Once the two have been paired, files can be securely transferred between the two. We chose to use Bluetooth simple pairing since users may have already paired their mobile device with their computer (e.g., to exchange calendar or contact information). Given that a secure mechanism for the exchange of data already exists, users may view a SPATE-specific transfer mechanism as unnecessary and cumbersome. If the workstation lacks a Bluetooth adapter, users can utilize a USB cable or any other direct connection to securely transfer data. We avoid using the Internet to imprint data because of networking and security issues. On current networks, mobile devices and most home computers are behind Network Address Translation services which prevent direct connections, stopping either the device or the workstation from acting like a server. Without additional setup, communication on the Internet is vulnerable to Man-in-the-Middle attacks where a third party modifies the data.

② **Exchange of Authenticated Data** Authenticated exchange within a small group in SPATE involves four steps: 1) selection and counting, 2) commitment, 3) distribution, and 4) verification. In the first step, each user selects the data she wants to share with the other group members and indicates to the device the number of physical members in the group (i.e., the group contains N people). Once the device knows the number of members and the data the user wishes to exchange, the device automatically performs the commitment and distribution steps. After the device checks that the received commitments agree with the distributed data, the device computes a T-Flag representation of the received data. To verify that all of the physically-present participants have the same data, users compare the T-Flags displayed by their devices. If everyone has received the same data, the T-Flag on each device should be identical. Section 4.2 contains more details on how a SPATE exchange is performed.

③ **Retrieving Data from the Device** After the user has completed the SPATE exchange, the last step is to upload any collected data from the user's mobile device to the user's workstation. Uploading data from the mobile device to the workstation is similar to the creation and imprinting step: the two devices pair or use a prior association from a previous pairing to establish a secure channel which is used to transfer the collected data.

4.2 SPATE Exchange of Authenticated Data

For applications where a user interacts with other users and requires trust, users need to obtain authentic data from the other users. Each user could pair with every other user to securely exchange data. However, a pair-wise protocol is inefficient in that $O(N^2)$ pairs are needed for a group of N users. The following protocol allows a group of N users numbered $1 \ldots N$ to exchange authentic copies of data $d_1 \ldots d_N$ (where d_i is user i's data) with $O(N)$ interactions. Figure 2 presents an outline of the steps of the exchange protocol.

In a SPATE exchange of authenticated data, the end goal is for each group member to have collected an authentic copy of every other member's data. This exchange consists

[2]If the security application runs on the user's mobile device, the data always resides with the application, removing the need for imprinting or retrieving the data (Steps 1 & 3).

of four major steps: selection and counting, commitment, distribution, and verification. To ensure authenticity, each user only has to count the number of group members present and to perform a final comparison of T-Flags. The mobile devices perform all other steps associated with committing to, broadcasting, and verifying data without requiring any human interaction. It is important to note that the SPATE exchange requires no encryption or signing. As such, unless the user wants to run an application on the device that requires the private key, all of a user's secrets remain on their workstation. With all of the secrets on the workstation, a lost device has zero impact on security. This is more secure and computationally efficient than other protocols (described in Section 9) where the device must perform private key operations.

Selection and Counting (Steps 1–2) The SPATE exchange begins with each user selecting the data the user wishes to share (data d_i for user i) and entering the number of users present in the group (here we represent the user-supplied number as $\tilde{N}$). Both of these items require human intervention. The data to be shared is application-dependent and depends on how the user wants to interact with the other group members. The user must enter the number of physical members in the group. If the device were to simply count the number of messages it receives, a malicious party outside the group could inject wireless messages and infiltrate the group.

Commitment (Steps 3–8) Once the device knows what data the user wants to share and the size of the group, the device generates two commitment [4] values: a protocol commitment and a data commitment. With two separate commitments, SPATE prevents attacks and limits the impact of human errors, unless all group members make a mistake. To generate the protocol commitment, the device generates a random number or nonce (i.e., mobile device i generates n_i) and hashes the nonce ($p_i = h(n_i)$). The device hashes the protocol commitment with this device's data to generate the data commitment (see Step 5). Without the data commitment, an attacker can modify data from some group members without being detected during verification [22]. During such an attack, the malicious party would wait until all but one group member had broadcast their data. The attacker would replace the last user's data (d_N) with a different d'_N such that T-Flag($h(d_1||...||d_N)$ =T-Flag($h(d_1||...||d'_N)$). With knowledge of d_1 to d_{N-1} and only 24 bits of entropy in a T-Flag, an attacker could find such a d'_N in a few seconds. The protocol commitment ensures that if at least one user correctly compares T-Flags within the group, SPATE fulfills the three properties of an authentic group exchange (even if some members are lazy and skip the comparison step). Exact details are explained in more detail in the security analysis in Appendix A. The device records its nonce, protocol commitment, data, and data commitment as the initial members in a set of nonces, protocol commitments, data, and data commitments for this group: sets $\mathbb{N}, \mathbb{P}, \mathbb{D}$, and $\mathbb{C}$, respectively. After generating these values, the device broadcasts the data commitment to the rest of the group (Step 6). At the same time, the device is receiving data commitment broadcasts from the other group members (Step 7), and adding the received commitments to its set of data commitments ($\mathbb{C}$). If the device receives less than $\tilde{N}$ data commitments before a timer threshold, either the user miscounted or a malicious party is preventing a device from contributing its commitment. In such a case, the protocol quits, since at least one of the $\tilde{N}$ devices has failed to contribute a data commitment. If the device receives more than $\tilde{N}$ commitments, either the user miscounted the size of the group, or a malicious party has inserted additional commitments. In such a scenario, the protocol quits and any data is discarded as invalid.

Distribution (Steps 9–10) Once each device has received the correct number of data commitments, devices can begin to exchange data. The device broadcasts its data and the protocol commitment used to generate its data commitment (Step 9). At the same time, the device receives the other devices' data values and protocol commitments and adds those values to the respective sets (Step 10).

Selection & Counting

1. $U_i \xrightarrow{UI} M_i$: d_i (the data to be shared)
2. $U_i \xrightarrow{UI} M_i$: $\tilde{N}$ (number of people in the group)

Commitment

3. M_i : $n_i \xleftarrow{r} \{0,1\}^\ell$, $\mathbb{N} \leftarrow \{n_i\}$
4. $p_i \leftarrow h(n_i)$, $\mathbb{P} \leftarrow \{p_i\}$, $\mathbb{D} \leftarrow \{d_i\}$
5. $c_i \leftarrow h(d_i||p_i)$, $\mathbb{C} \leftarrow \{c_i\}$
6. $M_i \rightarrow *$: c_i
7. $* \rightarrow M_i$: c_j (for $j \neq i$)
   M_i : $\mathbb{C} \leftarrow \mathbb{C} \cup c_j$
8. M_i : if ($|\mathbb{C}| > \tilde{N}$) or `timeout` quit (incorrect number of values)

Distribution

9. $M_i \rightarrow *$: (after receiving $\tilde{N}$ commitments) d_i, p_i
10. $* \rightarrow M_i$: d_j, p_j for ($j \neq i$)
    M_i : $\mathbb{D} \leftarrow \mathbb{D} \cup d_j$, $\mathbb{P} \leftarrow \mathbb{P} \cup p_j$

Verification

11. M_i : for $j \in 1...\tilde{N}$ if $c_j \neq h(d_j||p_j)$ quit (wrong data commitment)
12. M_i : $T - Flag(h(\mathbb{C}||\mathbb{D}||\mathbb{P}))$ (on screen)
13. $U_i \xrightarrow{UI} M_i$: "All N T-Flags Match" or "Some T-Flags Differ"
14. M_i : if "All N T-Flags Match" broadcast n_i else broadcast n'_i ($n'_i \xleftarrow{r} \{0,1\}^\ell$, $n'_i \neq n_i$) quit (discard collected $\mathbb{D}$
15. $* \rightarrow M_i$: n_j (for $j \neq i$)
    M_i : $\mathbb{N} \leftarrow \mathbb{N} \cup n_j$
16. M_i : for $j \in 1...\tilde{N}$ if ($p_j \neq h(n_j)$) or `timeout` quit (wrong protocol commitment)
17. M_i : Save $\mathbb{D}$ if all N values are correct

Figure 2: Steps for user U_i ($i \in 1 \ldots N$) to exchange data d_i with the other $N-1$ users via mobile devices. $U_i \xrightarrow{UI} M_i$ indicates inputs over the user interface from user U_i to their mobile device M_i. Any other transfer of data (e.g., $M_i \rightarrow *$) indicates wireless communication.

Verification (Steps 11–17) Once a device has received the entire set of data, data commitments, and protocol commitments, the verification stage of the protocol begins. The device verifies that the data and protocol commitments match the original data commitments (Step 11) by comparing the

data commitment with the hash of the received nonce and protocol commitment.[3] Provided all of the data commitments are correct, all that remains to ensure authenticity is for the device to verify that the values it received match the values the other devices received and that the other devices received its data. To verify each member's device received the same information, each device displays a T-Flag which represents the hash of the data commitments, data, and protocol commitments exchanged during the protocol (Step 12). At this time, the group members will compare the T-Flags on the devices' screens and indicate to their device if "All N T-Flags Match" or if "Some T-Flags Differ" (Step 13). The use of commitments and a final comparison where users verify **the T-Flags on every device match** ensures with high probability that all of the devices in the group received the same information. With a T-Flag containing 24 bits of entropy, the probability of the same T-Flag on each device with different underlying data is 2^{-24}. (We provide a security analysis in Appendix A.)

Impatient group members may click "All N T-Flags Match" without looking at the T-Flags in the group. In SPATE, the use of protocol commitments and nonces allows the actions of one or more diligent group members to protect such impatient users from saving incorrect data in the case of an attack. After a user indicates the T-Flags agree, the device will reveal its nonce (see Step 14) and expect to receive the correct nonce from the other $N-1$ group members (see Step 15) before the device saves $\mathbb{D}$. An incorrect n is an indicator that a member indicated "Some T-Flags Differ" and dictates that members should discard $\mathbb{D}$ since $\mathbb{D}$ is inconsistent across some of the devices.[4] When all N nonces are correct, every group member agrees that "All N T-Flags Match", and every device will save $\mathbb{D}$. The nonces ensure that any saved data fulfills the three properties needed for authentic information exchange within a group, even if $N-1$ or fewer group members click "All N T-Flags Match" without even looking at their devices.

5. APPLICATIONS

The SPATE system allows users to exchange public keys in a secure and convenient way. To demonstrate the usefulness of the SPATE system, we design and implement two applications on top of SPATE. In this section, we present a high-level overview of a secure email application and a secure file sharing application. In the following sections, we present our implementation and evaluation.

5.1 Secure Email

In an ad-hoc group meeting, people may exchange their physical business cards, or simply email addresses, to enable subsequent communication. Each group member needs to distribute her cards to all the other group members, and she will receive a different business card from each of the other group members. Not only does distributing physical cards consume time and resources, but each user then needs to enter the received information into her digital address book later. Distributing vCards [21] using Bluetooth wireless communication may save time by eliminating typing, however, it requires pairwise Bluetooth pairing to provide any authenticity guarantees for the received information. This approach does not scale: even for small groups with 8 users, there are 28 pairs.

Our secure email application provides a convenient mechanism for importing other users' public keys and email addresses. Using the secure email application, a user can imprint a self-generated X.509 public key certificate from their workstation onto their mobile device. During the exchange of authenticated data, she will obtain other users' certificates. When she retrieves the collected certificates, the application will extract the email addresses and names from the certificates and automatically import them into the application's address book. Then, the user can send secret and authentic emails. Our application is built as a plug-in to Thunderbird [31], enabling simple adoption.

We can summarize the features of the secure email application as follows:

1. *Convenient to import contacts.* The user does not have to perform any operation per received certificate. The uploading process is fully batched and automated.

2. *Authenticated and confidential email.* We provide an alternative to PGP- and PKI-based solutions. Thanks to the physical contact between human users, we can assert that the contact information and public key that a user has received is from that person she has met[5].

3. *Compatible with an existing mail client.* Thunderbird is one of the most popular POP/IMAP email clients. Existing Thunderbird users can adopt our application by installing it as a plug-in.

5.2 Secure File Sharing

In many scenarios, people may want to share files after a social gathering. For example, scholars meet at a conference and wish to start up a research project, or students at a party want to share video games and music. In these cases, the participants want to block people outside the group from accessing the files. Also, they would like to share the files with proper access control, but without frustrating management overhead. Good and Krekelberg show that users have trouble correctly setting permissions on files [17]. Furthermore, the file system should maintain accountability information and revocability to help detect and stop misbehaving users. Current solutions (e.g., BitTorrent [11], Dropbox [20], and KaZaA [49]) do not meet these requirements.

We present a secure file sharing application that does satisfy the above requirements. Each user downloads her workstation's configuration file and its public key to her mobile device in advance. During the distribution of certificates with SPATE, a user voluntarily provides her storage space for file sharing. The configuration file of this user is now distributed to other users in the group, and the user collects the other users' public keys. That user uploads other users' public keys to our application, which will automatically create a session for this group of users. They will have a separate directory which only this session's users can access. Other

[3]To ensure the proper protocol commitment, data, and data commitment are compared, all sets are ordered with respect to a unique sender value (e.g., Bluetooth or MAC address), as opposed to the value of the element.

[4]A malicious party can inject an incorrect number to force members to discard data, but this is only a denial of service attack.

[5]Of course, we cannot avoid errors if the person she met gave false information. This problem cannot be solved even if PGP or a PKI is used.

users upload the configuration file to their respective workstations and mount the remote file system. We implemented this application on top of *sshfs* [43], a file system that works over the SSH protocol.

Our application has the following advantages over past solutions:

1. *Secure transport.* SSH tunnels protect file transfers from eavesdropping and tampering.
2. *Convenient access control.* Shares on the server correspond one-to-one with successful SPATE protocol exchanges. They are created automatically, with the policy that only users present at the physical key exchange can access the files in the share.
3. *Accountability and revocability for misbehaving users.* Each user is connected to the remote machine as an individual user. Any misbehavior by the user can be attributed to her user name. For instance, it is suspicious if many sessions simultaneously connect to the server using the same login credentials. The machine owner can then revoke or suspend this user. The file system could also be extended to use the SPATE exchanged public keys to enable non-repudiation for changes made to the shared files via digital signatures.
4. *User-friendliness.* Users do not need to remember hostnames, usernames, or passwords. The host address and usernames are exchanged during the *Distribution of Certificates* phase of SPATE. Since authentication is done using public key authentication in SSH, no passwords are required. Of course, our system does assume that the server machine is globally-routable. Servers behind NAT can work but are more difficult to configure.

6. IMPLEMENTATION

We have fully implemented the SPATE system and two applications on Nokia N70 and E51 smart phones and commodity Dell workstations running Windows XP and Ubuntu Linux. The system contains three parts: 1) the SPATE Mobile Client that supports key exchange for the email and file-sharing applications, 2) a Thunderbird plug-in to enable secure email, and 3) a file sharing application (Figure 3). In the following sections, we describe the implementation details of these three programs.

6.1 SPATE Mobile Client

The SPATE Mobile Client is implemented in C++ for Symbian OS v8.1a (with Nokia Series 60 second generation graphical user interface) running on Nokia N70 smart phones equipped with a digital camera and Bluetooth radio. The size of the Symbian Installation System (SIS) binary for the SPATE Mobile Client is 47 KB, enabling deployment over even bandwidth-limited GPRS networks. We have also ported the SPATE Mobile Client to the newer Nokia E51 with Symbian OS v9.1 (Series 60 third generation); however, we focus on our N70 implementation for comparability with prior work on authenticated exchange [10, 30].

Figure 3 shows the architecture of our SPATE Mobile Client: it includes a library of commonly used functions and email- and file sharing-specific modules. The SPATE library includes communication and visual engines. The communication engine is responsible for data transmission and

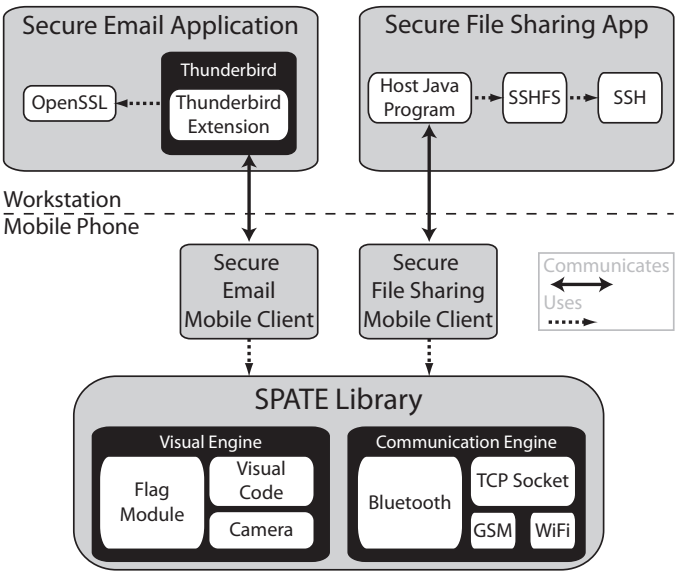

Figure 3: SPATE system overview.

contains the Bluetooth module. The visual engine is used to generate T-Flags. As described in Section 4.2, SPATE requires devices that support message broadcast. Bluetooth does not support broadcast; however, it does support a *piconet* of up to eight devices. We employ Bluetooth piconets to simulate broadcast by forming a star network with a volunteer leader during a SPATE exchange. Our simulated broadcast also has the advantage of isolating different groups in the same physical space, thereby eliminating crosstalk between groups of well-behaved devices (the common case).

Additionally, we desire to circumvent the Bluetooth device and service discovery process, as it can introduce overheads of tens of seconds, as well as user confusion [36]. Thus, we augment our visual engine to also generate, photograph, and decode two-dimensional barcodes (2D barcodes) which we use to circumvent Bluetooth device discovery, as proposed by Scott et al. [36].

The Bluetooth module is used for all data exchange (between mobile devices and between a mobile device and a workstation). Note that this is a design decision we made for our implementation; other communication interfaces (e.g., infra-red, USB, WiFi, or the cellular network) are also viable. Ideally, during the SPATE exchange between multiple Mobile Clients, we would have a broadcast primitive available.

2D Barcodes are generated, photographed, and decoded using the VisualCodes module from Rohs and Gfeller [35], ported to work with newer versions of Symbian OS. The T-Flags module is used at the end of authenticated data exchange; it displays a visual hash on devices' screens (Figure 4).

6.2 SPATE Exchange Walk-Through

Here we provide a walk-through of a SPATE exchange using our implementation, in accordance with the SPATE protocol from Section 4.2. The only significant departure from the SPATE protocol in Section 4.2 is the additional requirement that the people in the prospective group agree on a

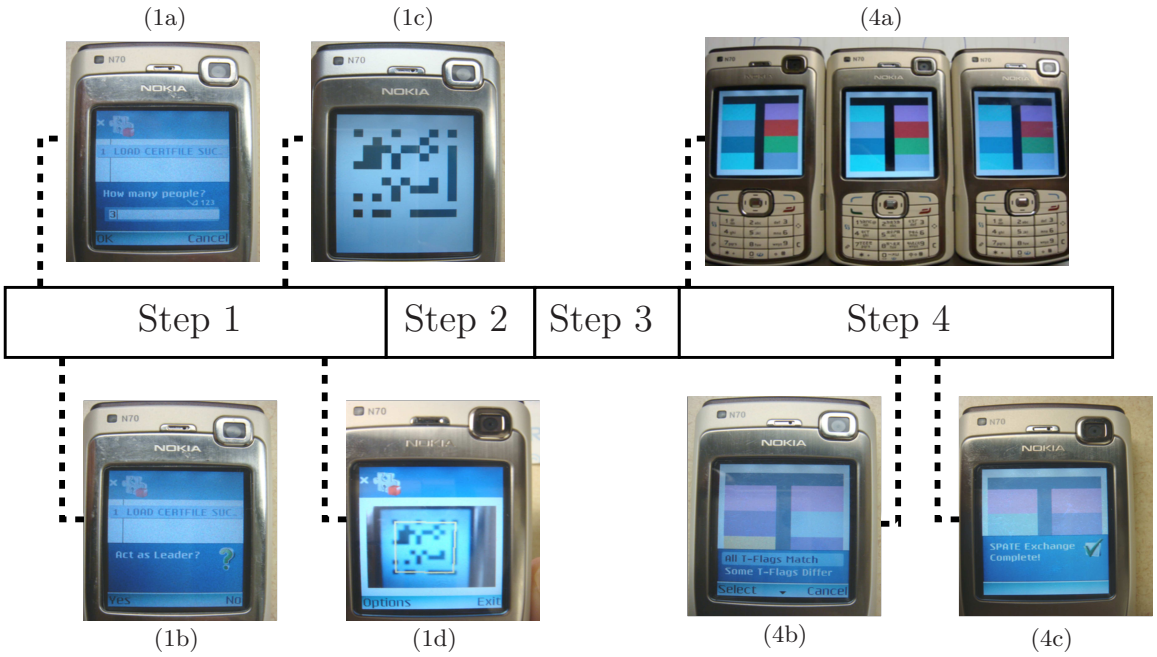

Figure 5: Execution Flow of SPATE Exchange. Step 1: Selection and Counting, Step 2: Commitment, Step 3: Distribution, Step 4: Verification. Steps 1b, 1c and 1d are necessary in our implementation because Bluetooth does not support broadcast.

Figure 4: Our Mobile Client displaying T-Flags on N70 smart phones during a SPATE exchange. The left and center T-Flags are identical, but the right T-Flag is different.

leader to serve as the hub of the star network for simulating broadcast with Bluetooth. Figure 5 provides a chronological breakdown of the individual actions performed by a user during the exchange.

Step 1, *Selection and Counting*, begins automatically when the user starts the email or file-sharing SPATE Mobile Client on her mobile device. Step 1a in Figure 5 shows the Mobile Client prompting the user to count the number of prospective group members: "How many people?" The user may enter a number between 2 and 8 (the maximum number of devices supported by Bluetooth piconets, and the threshold above which humans begin to make counting errors [25]). Once the count has been entered, the Mobile Client prompts the user as to whether she would like to act as the leader for this SPATE exchange: "Act as Leader?" The user may select Yes or No.

The devices must now establish Bluetooth connectivity. We use 2D barcodes to circumvent the Bluetooth discovery process. The leader uses her device's camera to photograph the barcodes on the remaining prospective group members' devices (which encodes each device's Bluetooth address). Step 1c shows a Mobile Client displaying a barcode, and Step 1d shows the leader's Mobile Client successfully decoding the barcode on another device. Once the leader has photographed all members' barcodes (detected automatically since the Mobile Client can compute the expected number of distinct barcodes from the count entered by the user in Step 1a), her device can construct a Bluetooth piconet between all of the devices. The leader's device serves as the master and the remaining devices are slaves. The result is a network with a star topology connecting all prospective group members' devices to the leader's device. The leader's device can then simulate broadcast by unicasting messages to all connected slave devices.

All SPATE protocol operations for Step 2 (*Commitment*) and Step 3 (*Distribution*) are automatically executed by the SPATE Mobile Client. We design our Mobile Client to avoid all non-essential user interactions in an effort to make the exchange as smooth and fast as possible. The final step (*Verification*) again involves the user. If the SPATE protocol successfully verifies all message commitments, then each device will compute the final hash of the prospective group members' public keys and commitments and display it as a T-Flag (Step 4a in Figure 5). The user is prompted to determine whether the T-Flags match. If the protocol fails during the automated message exchange, the user is informed that there has been an error and that she should retry.

It is now the responsibility of the prospective group members to compare the T-Flags displayed by each of their devices. If the users agree that all of the devices are displaying identical T-Flags, they select "All T-Flags Match" (Step 4b). Otherwise, they select "Some T-Flags Differ." If the user indicates that the flags do match, then her device stores the newly received public keys for transmission to her workstation later. It also displays the message, "SPATE Exchange Complete!" (Step 4c).

6.3 Secure Email

We enable secure (with authenticity, integrity, and secrecy if desired) email communication between users without a PKI. We implemented our secure email application as a Thunderbird extension using only 4135 lines of code. The extension uses OpenSSL [12] to generate a public/private signing keypair encapsulated in an X.509 certificate and PKCS12 file for the user. This happens once during initial setup. The certificate includes the user's email address and is imported into Thunderbird as a trusted certification authority (CA). The user's certificate serves as a CA to authenticate future certificates received from other users via the user's Mobile Client. Next, the user can download her certificate from the extension to her mobile phone, thus *imprinting* it with the user's digital identity. She is now ready to participate in SPATE exchanges, as described in the previous section.

After the user has participated in a SPATE exchange, her device will have obtained self-signed public key certificates from other users. She can upload all received certificates from her Mobile Client to the Thunderbird extension. The extension automatically signs[6] the received certificates with the user's private signing key and imports them into Thunderbird's address book. Users can then exchange secure emails through Thunderbird's built-in S/MIME [34] functionality. In accordance with S/MIME, the email content can also be encrypted under the receiver's public key (in addition to being signed by sender's private key).

6.4 File Sharing

Our file sharing program is built for Linux using Java 6 and shell scripts, on top of the *SSH File System (SSHFS)* [43]. SSHFS allows a client to mount a remote file system tunneled through the SSH protocol. When the program is first started, it creates a server configuration file with its host's IP address and the public host key that is used by the host's SSH server. It also generates a public/private signing keypair for the Mobile Client. After key generation, the user imprints her mobile phone (via downloading) with her workstation's configuration file and her public key. Her device is now ready to participate in SPATE exchanges to meet users with whom she would like to share files.

The Mobile Client of the user that volunteers to be the leader of the group during the SPATE exchange will distribute both the file-sharing configuration file and the user's public signing key. Other users only send out their public signing keys. At the end of this phase, every client will have received the leader's server configuration file and the leader will have received all the clients' public key certificates. The leader later uploads the other clients' certificates to her workstation.

[6]Thunderbird does not accept public keys unless they are signed by a trusted CA.

The file-sharing application (acting on behalf of the group leader) briefly requires root privileges to complete the following tasks: 1) generate a group name with the hash of received certificates; 2) create an account for each client, using the filename of her public key as the username; 3) register their public keys as authorized SSH users; and 4) restrict their SSH access to reading and writing files only (e.g., using scponly [37]). Finally, it creates a directory for the group and adds each client into the group. The user may also assign the group a "friendly name" after importing the other users' public keys.

Figure 6: Screen shot from the file-sharing application. The application lists the shared folders on the local host and folders mounted from remote systems.

Each of the non-leader users uploads the received configuration file from their Mobile Client to the file-sharing application running on their workstation. The application will read the server's IP address from the configuration file and append the server's public host key into its list of known hosts (`~/.ssh/known_hosts`). The application mounts the remote file system using the SSHFS engine. Since the server and the client use their exchanged public keys for authentication (i.e., the public key-based authentication method offered by a standard SSH installation), there are no passwords for authentication.

The file-sharing application displays information about currently shared folders, active groups, and active users on the local machine (Figure 6). It also enables the user to mount shared folders on remote machines.

7. EVALUATION

We evaluate the performance of authenticated key exchange using our SPATE Mobile Client implementation. We do not discuss the performance of data exchange between a mobile device and a workstation (i.e., imprinting the device initially and then retrieving newly acquired public keys), since synchronizing data between a mobile device and workstation is a widely available operation.

7.1 Method

We ran SPATE, Seeing-is-Believing [30], and GAnGS [10] on two to eight Nokia N70 smart phones. Each data point represents the average of 10 runs. Time consumed by automated protocol steps (i.e., without involving the human; Step 2 from Figure 1) is recorded in the experiment. We

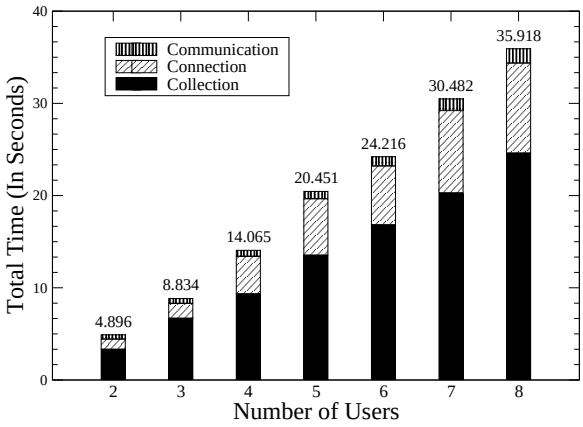

Figure 7: Time consumed during a SPATE exchange.

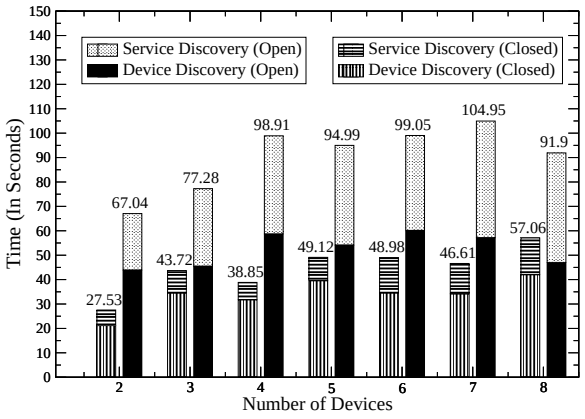

Figure 8: Bluetooth device and service discovery overhead.

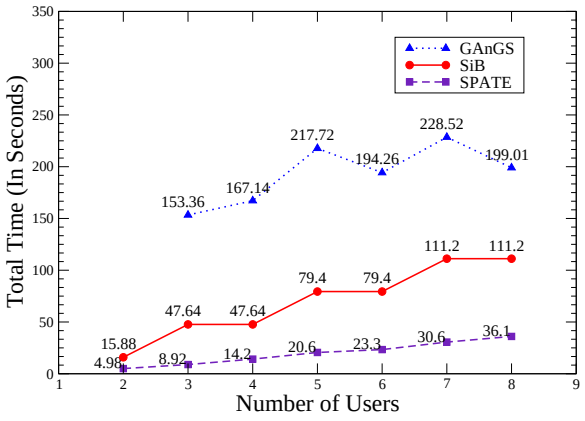

Figure 9: Comparison between SPATE, SiB, and GAnGS.

measured the time consumed by *Collection*, *Connection*, and *Communication*. *Collection* represents the time consumed by the leader while she photographs the 2D barcodes on others' screens. *Connection* represents the time needed to establish a Bluetooth piconet once all barcodes have been photographed. *Communication* includes the time for data transfer during the automated Commitment and Distribution (Steps 2 and 3 from Section 4.2 and Figure 5) steps of the SPATE exchange. For the other systems (SiB and GAnGS), only the total time is measured for comparison. To eliminate human factors in the execution time, these tests are performed by experienced operators of all three systems.

7.2 Results

Figure 7 shows the time consumed by *Collection*, *Connection*, and *Communication*. During *Collection*, the leader needs to photograph $N-1$ 2D barcodes from other users. In our experience, the leader needs 2-3 seconds to successfully photograph one 2D barcode. While this is the leading source of time consumption in our system, 2-3 seconds is considerably less than that which we would have incurred using Bluetooth device and service discovery. To confirm the overhead of Bluetooth discovery (and to validate the results of Scott et al. [36]), we implemented a Symbian C++ program to record the time spent on device and service discovery. We conducted this experiment twice: once in an open cubicle environment with many nearby Bluetooth devices, and once in a closed apartment isolated from other Bluetooth devices. Figure 8 shows our results; each data point is the average of five runs. Even the best-case result requires almost 30 seconds for two devices to discover each other, connect, and query for the desired service.

Once all of the Bluetooth addresses have been collected by the leader during *Collection*, the leader's device establishes a Bluetooth piconet. This results in the *Connection* overhead in Figure 7, which takes roughly one to ten seconds, depending on how many devices are involved.

Once the *Connection* is established, *Communication* consumes less than one second even with a full eight devices. Even with our star network topology, Bluetooth has sufficient bandwidth to rapidly transfer the public keys and commitments, which make up no more than a few kilobytes. Verification of the commitments consumes less than 200 milliseconds. We omit it from the figure since it would not be visible.

The time consumed by the human user to count the number of participants, photograph barcodes, and compare flags also contributes overhead. We find that these operations can be done within 30 seconds by users familiar with SPATE, enabling a complete run of the SPATE system with eight users in approximately one minute.

Comparison with Existing Systems Next, we present a comparison between SPATE and two prior key-exchange systems: Seeing-is-Believing and GAnGS (Figure 9). Note that SiB is designed for key exchange between two parties. Our SiB experiment was performed with two people; we then extrapolated to obtain the expected overhead where each member must pair with every other member for a total of $O(n)$ rounds of pairing. Without the additional time needed to move about the group or keep track of who has paired with whom, our SiB results can be interpreted as best-case.

SPATE, SiB, and GAnGS all use the barcode format proposed by Rohs and Gfeller [35]. In our experience, recognition is efficient and accurate. However, SiB and GAnGS use an extension to support multiple, cycling barcodes where, e.g., four barcodes are cycled every second. We have found that the requirement to successfully recognize all four barcodes significantly degrades usability. With SPATE, the

only information to be conveyed in the barcode is the Bluetooth address and service channel: $48+3=51$ bits. We can therefore employ a single static barcode, greatly improving recognition times. In addition to slower barcode recognition, SiB and GAnGS require bidirectional barcode recognition (i.e., A reads B's barcode, and B reads A's barcode). GAnGS is a multi-round protocol designed for scalability and denial of service resilience. However, for smaller groups multiple rounds introduce overhead and thus slower performance. With a single round, fewer barcodes to recognize, and faster barcode recognition, SPATE outperforms SiB and GAnGS for groups of three to eight users.

8. DISCUSSION

In this section, we discuss a topic not previously addressed.

8.1 Is Counting Necessary?

In SPATE, group members count the number of members present to prevent non-group members from adding their data (see Appendix A for more details). However, if users exchange personally identifiable information (e.g., names and pictures), counting is optional. After running SPATE (without counting), group members can examine the acquired data and verify that they received information from the expected group members and only those people. For example, user A will verify that running SPATE with users B and C yields data with B's name and data with C's name. During this extra verification step, the user can detect any additional data inserted by an outsider, O. If O simply adds itself to the group, A can detect the unexpected data labeled with O's name. If O tries to impersonate a legitimate group member (e.g., O submits a different public key or email, but the same personally identifiable information as C), A will notice the duplicate entries for C. If O tries to delete a group member, the T-Flag comparison will detect the attack. Without counting, SPATE requires the user to press a button to indicate when the commitment phase is complete (i.e., without $\tilde{N}$, the device does not know when it has the proper number of commitments). Therefore, there is a tradeoff to ensure security; users have to count before the commitment phase or carefully examine the data as part of the verification phase.

9. PREVIOUS WORK

This work is preceded by protocols that establish authentic information between two devices, which is often referred to as "pairing". Proposed strategies include: password entry on one or both device(s) [28, 29]; string comparison that uses the human as a channel to ensure authentic exchange of information [26, 28, 29, 47]; audio-based comparison where the human user compares the strings via audio representation [18]; visual-based comparison of graphics that encode data [13, 33]; shaking devices to create shared entropy pools [9, 19, 27]; common properties of the wireless channel to establish authentic or secret information [7]; and location-limited channels [3, 32, 40].

Closely related to the SPATE exchange is GAnGS [10]. Both attempt to distribute authentic information within a group of physically collocated users. However, GAnGS is designed only for the exchange of public keys and requires the installation of the private key on the user's device. In addition, SPATE is more efficient in that users are required to perform fewer total interactions in the absence of infrastructure. Specifically, for N users SPATE requires N interactions while GAnGS requires $3N$.

Within the PGP community, *key signing parties* may be held to authenticate groups of users [5]. The purpose of a key signing party is to extend the web of trust: users gather in a physical location to verify the identity of other attendees (e.g., using a passport or driver's license) and sign the PGP certificates linking attendees' names and public keys. The proposed methods are suitable for forming groups, but cumbersome. Attendees print their names and key fingerprints on slips of paper, to be verified manually by other attendees. Alternatively, a coordinator compiles a list of attendees in advance, and each attendee must be verified at the party. For large groups, comparing each attendee's key fingerprint is awkward and error-prone.

Researchers have also proposed numerous key agreement protocols for groups, which rely on a PKI that issues certificates to each user [6, 23, 24, 41, 42, 44]. These protocols all assume a common trusted certification authority (CA). The CA is needed so that group members can authenticate other members' certificates. Unfortunately, this assumption is invalid in many settings. Different organizations may not have any trusted authorities in common, or group members may lack certificates entirely. The SPATE exchange is complementary to PKI-based schemes, as it can be used to establish the authenticated certificates needed to set up a group key.

Other works have examined key agreement protocols for groups, which rely on string comparison or shared passwords [1, 2, 46]. In contrast to SPATE, all of these schemes aim to establish a shared secret between the group members. After SPATE is used to exchange authentic public keys, it is possible to set up a shared secret within the group using any of the PKI-based schemes since SPATE allows the authentic exchange of public keys. However, a shared secret lacks the properties needed to distribute authentic public keys within a group. Specifically, with only a shared symmetric group key, any member can generate a message authenticator and thus it is impossible to tell which user truly was the source of a message (i.e., member A can claim K^+ is member B's public key and use the shared group key to produce the correct authenticator to support that claim). Also, many prior works do not implement their schemes in a real-world system, which elides numerous practical issues.

Finally, there is research using location-limited channels to exchange keys [3, 8, 40]. Talking to Strangers [3] and Capkun's work [8] use demonstrative identification over a location-limited channel (e.g., infrared) to exchange authenticated public keys. Talking to Strangers may be used for groups, but it lacks a step for member verification. Thus, the scheme is vulnerable to malicious members who mount Sybil attacks; the multiple identities of one member would go undetected. Capkun's work only discusses how to establish a security association between two devices which physically interact or share a trusted "friend" (much like PGP's web-of-trust). The Resurrecting Duckling protocol [40] leverages a direct physical connection between devices for key setup. In the protocol, a mother duck (i.e., the group leader) defines and distributes a key to the ducklings (i.e., the other members of the group). During setup, a policy is uploaded. The policy specifies what actions a duckling will take. Thus, the mother duck's policy can direct the ducklings to support group communication. Unfortunately, this requires that the

mother duck is completely trusted. In addition, there are several practical issues with using Resurrecting Duckling for groups. First, imprinting ducklings is a sequential operation. Every duckling needs to touch the mother duck, and she becomes a choke point in the group formation process. Second, the scheme requires a special interface that supports physical contact. Finally, like most other group schemes, Resurrecting Duckling has not been implemented in a real-world system to the best of our knowledge.

The field of Computer Supported Collaborative Working (CSCW) is closely related to many of the applications that would use SPATE. After a group meets and performs a SPATE exchange, the next logical step is to use CSCW while the group is physically separated. Within the CSCW field, little has been done about how to secure applications. Foley and Jacob [14] described a formal language for defining security requirements in CSCW, but ignored how to enforce those requirements. SPATE presents one potential way to enforce those requirements.

10. CONCLUSION

We have presented SPATE, a system for authentic exchange of public key information in groups of two to eight people. SPATE represents a unique point in the design space for ad hoc group key establishment. We trade off scalability and denial-of-service resilience for speed and ease of use. Indeed, only symmetric cryptographic primitives are employed on the mobile device. An additional benefit is that SPATE has no need to store secrets (e.g., a private key) on the mobile device, thereby excluding an entire class of security vulnerabilities stemming from lost or stolen devices.

We rely on the user to accurately compare images across other users' devices and count the number of prospective group members, but we limit the maximum group size to eight people. In our experience, the resulting system is easy and fun to use, finally providing the opportunity to achieve easy-to-use secure email and secure file sharing.

ACKNOWLEDGEMENTS

We would like to thank Ghita Mezzour, our shepherd Kevin Fu, Ben Ransford, and the anonymous reviewers, for their insightful comments and feedback that greatly improved the quality of this work.

11. REFERENCES

[1] Abdalla, M., Bresson, E., Chevassut, O., and Pointcheval, D. Password-based group key exchange in a constant number of rounds. In *Public Key Cryptography (PKC)* (2006), pp. 427–442.

[2] Asokan, N., and Ginzboorg, P. Key-agreement in ad-hoc networks. *Computer Communications 23*, 17 (Nov. 2000), 1627–1637.

[3] Balfanz, D., Smetters, D., Stewart, P., and Wong, H. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS)* (2002).

[4] Blum, M. Coin flipping by telephone. In *Advances in Cryptography* (August 1982), pp. 11–15.

[5] Brennen, V. A. The keysigning party howto. `http://cryptnet.net/fdp/crypto/keysigning_party/en/keysigning_party.html`, Jan. 2008.

[6] Burmester, M., and Desmedt, Y. Efficient and secure conference key distribution. In *Security Protocols—International Workshop* (Apr. 1997), vol. 1189 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 119–129.

[7] Cagalj, M., Capkun, S., and Hubaux, J.-P. Key agreement in peer-to-peer wireless networks. *IEEE (Special Issue on Cryptography) 94* (2006), 467–478.

[8] Capkun, S., Hubaux, J.-P., and Buttyan, L. Mobility helps security in ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (MobiHoc)* (2003).

[9] Castelluccia, C., and Mutaf, P. Shake them up! a movement-based pairing protocol for cpu-constrained devices. In *Proceedings of ACM/Usenix MobiSys* (2005).

[10] Chen, C.-H. O., Chen, C.-W., Kuo, C., Lai, Y.-H., McCune, J. M., Studer, A., Perrig, A., Yang, B.-Y., and Wu, T.-C. GAnGS: Gather Authenticate 'n Group Securely. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)* (Sept. 2008).

[11] Cohen, B. Bittorrent. `http://www.bittorrent.com`, Apr. 2001.

[12] Cox, M. J., and Engelschall, R. S. Openssl: Open source toolkit implementing for ssl/tls. `http://www.openssl.org/`, May 1999.

[13] Ellison, C., and Dohrmann, S. Public-key support for group collaboration. *ACM Trans. Inf. Syst. Secur. 6*, 4 (2003), 547–565.

[14] Foley, S. N., and Jacob, J. Specifying security for CSCW systems. In *8th IEEE workshop on Computer Security Foundations* (1995).

[15] Ford, B., Strauss, J., Lesniewski-Laas, C., Rhea, S., Kaashoek, F., and Morris, R. Persistent personal names for globally connected mobile devices. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (Nov. 2006).

[16] Glasbey, C., van der Heijden, G., Toh, V. F. K., and Gray, A. Colour displays for categorical images. *Color Research and Application 32*, 4 (June 2007), 304–309.

[17] Good, N. S., and Krekelberg, A. Usability and privacy: a study of kazaa p2p file-sharing. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI 03)* (2003).

[18] Goodrich, M. T., Sirivianos, M., Solis, J., Tsudik, G., and Uzun, E. Loud and clear: Human-verifiable authentication based on audio. In *International Conference on Distributed Computing (ICDCS)* (2006), p. 10.

[19] Holmquist, L. E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., and Gellersen, H.-W. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Proceedings of Ubicomp* (2001).

[20] Houston, D., and Ferdowsi, A. Dropbox. `https://www.getdropbox.com/`, Sept. 2008.

[21] Howes, T., and Smith, M. RFC 2425: A MIME content-type for directory information., Sept. 1998.

[22] Jakobsson, M. Issues in security and privacy (lecture slides). `http://www.informatics.indiana.edu/markus/i400/`, 2006.

[23] Just, M., and Vaudenay, S. Authenticated multi-party key agreement. In *Advances in Cryptology – (ASIACRYPT)* (1996), vol. 1163 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 36–49.

[24] Kim, Y., Perrig, A., and Tsudik, G. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)* (Nov. 2000), pp. 235–244.

[25] Kuo, C. *Reduction of End User Errors in the Design of Scalable, Secure Communication*. PhD thesis, Carnegie Mellon University, 2008.

[26] Laur, S., and Nyberg, K. Efficient mutual data authentication using manually authenticated strings. In *Cryptology and Network Security (CANS)* (2006), pp. 90–107.

[27] Lester, J., Hannaford, B., and Gaetano, B. Are you with me? - using accelerometers to determine if two devices are carried by the same person. In *Proceedings of Pervasive* (2004).

[28] Linksky, J. et al. Simple Pairing Whitepaper, revision v10r00. `http://www.bluetooth.com/NR/rdonlyres/0A0B3F36-D15F-4470-85A6-F2CCFA26F70F/0/SimplePairing_WP_V10r00.pdf`, August 2006.

[29] Lortz, V., Roberts, D., Erdmann, B., Dawidowsky, F., Hayes, K., Yee, J. C., and Ishidoshiro, T. Wi-Fi Simple Config Specification, version 1.0a. Now known as Wi-Fi Protected Setup, February 2006.

[30] McCune, J. M., Perrig, A., and Reiter, M. K. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proceedings of the IEEE Symposium on Security and Privacy* (May 2005).

[31] Mozilla. Thunderbird 2. `http://www.mozilla.com/en-US/thunderbird/`, Dec. 2008.

[32] NFC Forum. NFC Forum: Specifications. `http://www.nfc-forum.org/specs/`.

[33] Perrig, A., and Song, D. Hash visualization: A new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)* (July 1999), M. Blum and C. H. Lee, Eds., pp. 131–138.

[34] Ramsdell, B. RFC 3851: Secure/multipurpose internet mail extensions (S/MIME) version 3.1 message specification, July 2004.

[35] Rohs, M., and Gfeller, B. Using camera-equipped mobile phones for interacting with real-world objects. *Proceedings of Advances in Pervasive Computing* (Apr. 2004), 265–271.

[36] Scott, D., Sharp, R., Madhavapeddy, A., and Upton, E. Using visual tags to bypass bluetooth device discovery. *ACM Mobile Computer Communications Review 9*, 1 (January 2005), 41–53.

[37] scponly. `http://sublimation.org/scponly/`, 2009.

[38] Soriente, C., Tsudik, G., and Uzun, E. BEDA: Button-enabled device association. In *International Workshop on Security for Spontaneous Interaction (IWSSI)* (2007).

[39] Soriente, C., Tsudik, G., and Uzun, E. HAPADEP: Human asisted pure audio device pairing. In *Information Security Conference (ISC)* (Sept. 2007).

[40] Stajano, F., and Anderson, R. J. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols Workshop* (1999), pp. 172–194.

[41] Steer, D., Strawczynski, L., Diffie, W., and Wiener, M. A secure audio teleconference system. In *Advances in Cryptology (Crypto)* (1990), vol. 403 of *Lecture Notes in Computer Science*, International Association for Cryptologic Research, Springer-Verlag, pp. 520–528.

[42] Steiner, M., Tsudik, G., and Waidner, M. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems 11*, 8 (Aug. 2000), 769–780.

[43] Szeredi, M. SSH filesystem. `http://fuse.sourceforge.net/sshfs.html`, Jan. 2005.

[44] Tzeng, W.-G., and Tzeng, Z. Round-efficient conference-key agreement protocols with provable security. In *Advances in Cryptology – (ASIACRYPT)* (2000), vol. 1976 of *Lecture Notes in Computer Science*, International Association for Cryptologic Research, Springer-Verlag, pp. 614–628.

[45] Uzun, E., Karvonen, K., and Asokan, N. Usability analysis of secure pairing methods. In *Usable Security (USEC)* (Feb. 2007).

[46] Valkonen, J., Asokan, N., and Nyberg, K. Ad hoc security associations for groups. In *Security and Privacy in Ad-Hoc and Sensor Networks (ESAS)* (2006), pp. 150–164.

[47] Vaudenay, S. Secure communications over insecure channels based on short authenticated strings. In *Advances in Cryptology (Crypto)* (2005), pp. 309–326.

[48] Whitten, A., and Tygar, J. Why Johnny can't encrypt. In *USENIX Security* (Aug. 1999).

[49] Zennström, N., Friis, J., and Kasesalu, P. KaZaA media desktop. `http://www.kazaa.com`, Mar. 2001.

APPENDIX

A. SECURITY ANALYSIS OF SPATE

We now analyze the security of SPATE by considering the various stages of the protocol and the various attacks that an adversary may attempt to exploit. As we discuss in our problem definition (Section 2), several vulnerabilities are outside the scope of this paper, most notably vulnerabilities of cryptographic primitives, denial-of-service attacks, and malware on systems that tamper with the execution. The security properties we need to demonstrate for the SPATE system are: consistency, exclusivity, and uniqueness for the exchanged data. Intuitively, these properties ensure that each group member's data is correctly sent to all other members, and that no data of an outsider is present.

More formally, each group user u_i of group G has data d_i, and after the SPATE protocol run, each user obtains exactly the set $\mathbb{D} = \bigcup_{i \in G} d_i$. Consistency requires that each user obtains exactly the same set, exclusivity requires that $u_j \notin G \Rightarrow d_j \notin \mathbb{D}$, and uniqueness requires that $u_i \in G \Rightarrow d_i \in \mathbb{D}$ and $|G| = |\mathbb{D}|$.

The SPATE protocol has three main phases: creation and imprinting of data, authenticated exchange of data, and retrieving data from the device. We now argue why each of these phases preserves the three security properties.

During the first phase (creation and imprinting of data) and third phase (data retrieval), data d_i is sent to the mobile device and the set $\mathbb{D}$ is retrieved from it. The exchanging devices belong to the same party, so we may assume the existence of an authenticated data transfer channel, perhaps via a secure wireless device pairing protocol [26, 28, 47]. As long as an adversary cannot alter the data exchanged, these two phases are safe, since the data is meant to be publicized.

In this section, we argue that the authenticated data exchange phase is robust against attacks. We first discuss how consistency is achieved, then uniqueness and exclusivity.

Consistency At the conclusion of a SPATE exchange, the data $\mathbb{D}$ held by the group should be consistent (i.e., every member has the same set of data $\mathbb{D}$). To violate this consistency, one of two things needs to happen: Either (i) all users believe "All N T-Flags Match" despite at least two distinct T-Flags appearing; or (ii) all devices show identical T-Flags despite an inconsistent set $\mathbb{D}$. The protocol and data commitments guarantee consistency by preventing these events. The use of a protocol commitment in SPATE ensures that when one group member detects different T-Flags, every member of the group is alerted of the difference. The exchange of data commitments before revealing data and protocol commitments ensures that an attacker cannot give different group members different values which produce the same T-Flag.

We assume that at least one member of the group dili-

gently compares the T-Flags on the devices. As such, when at least two distinct T-Flags are displayed, the careful user will notice the discrepancy. This careful user clicks "Some T-Flags Differ" on the device M_i, which will cause it to release n_i' (the wrong nonce) and exit (Step 14 of Figure 2). Without the correct nonces from all devices, the other group members' devices will infer that some member of the group detected two different T-Flags. With this sure indication that something is awry, every device in the group will discard the collected set $\mathbb{D}$. To trick the other group members into saving inconsistent data, an attacker could try to impersonate the diligent user's device M_i by broadcasting the correct nonce, n_i, and blocking the incorrect nonce. However, the attacker only knows p_i (the diligent user's disclosed protocol commitment). To learn a value n such that $h(n) = p_i$, the attacker must defeat the pre-image resistance property of the hash function. Given a cryptographically secure hash function, finding such a value is computationally infeasible. Therefore, when at least one user compares the T-Flags, the diligent user will release an incorrect nonce and alert the whole group that the T-Flags are inconsistent. Once group members receive the wrong nonce (or timeout if an attacker blocks the diligent user's device's transmission containing the incorrect nonce), the group will know about the inconsistent T-Flags and discard the inconsistent data.

Without a way to conceal different T-Flags, an attacker trying to distribute inconsistent data in a SPATE exchange needs different inputs to the T-Flags algorithm which produce the same output. Without data commitments, an attacker could find such inputs. With data commitments, the probability of identical T-Flags with inconsistent data is negligible (i.e., P(equal T-Flags | inconsistent $\mathbb{D}$) $= 2^{-24}$). Consider the protocol without the data commitments, i.e., suppose instead of broadcasting c_i and later (d_i, p_i), everyone simply broadcasts d_i. A malicious party could wait until all but one member of the group have broadcast their data, intercept the last member's data (d_L), calculate a new value d_L', and broadcast that value to the group. Here, the attacker's goal is to create equal T-Flags, derived from unequal inputs (d_L versus d_L'). Such an attack exploits T-Flags's weak second pre-image resistance property, and requires 2^{23} hash computations on average (since T-Flags's output space size is only 2^{24}). A modern computer can perform a hash operation in about 1 μs, so under ten seconds of processing on a single computer would yield a break of the protocol.

Such an attack is feasible because the malicious party knows all of the group members' data before it has to generate the value d_L'. To prevent such an attack, the system must prevent a member from crafting their inputs to the T-Flag algorithm after learning the other group members' inputs. The use of data commitments does just that by binding group members to their inputs to T-Flags before other group members reveal their inputs.

It is important to note that the group members will wait until they hear all N data commitments to reveal their protocol commitment and data. Assuming the hash function is pre-image resistant, the data commitment leaks no information about the two inputs (i.e., the data and the protocol commitment). Even if a recipient knows d_i and c_i (where $c_i = h(d_i||p_i)$), the only way to discover p_i is by guessing. This means that waiting until all other group members have revealed their data commitments provides no advantage to an attacker. The attacker also gains no advantage by waiting to be the last to reveal her data and protocol commitment. The collision-resistant property of the hash function prevents the attacker from finding other data and protocol commitment pairs (d, p) which yield the same c. Without knowledge of group members' protocol commitments, a malicious party cannot precompute different inputs to the T-Flag algorithm that produce the same output. The best approach for a malicious party to produce identical T-Flags with different underlying values is simply to guess randomly. In such a case, the probability of two T-Flags matching when the inputs are different is the probability of the lower 24 bits of the underlying hash's outputs matching (i.e., $1/2^{24}$).

Exclusivity & Uniqueness At the end of the SPATE exchange the set $\mathbb{D}$ does contains only group members' data and only contains one element from each member. Once we know that the set of data is consistent and that each member has counted accurately, the pigeonhole principle ensures exclusivity and uniqueness are achieved. An exclusive set of data for a group of N members contains data from only those N members. A unique set of data for a group of N members contains one piece of data from each of the N members. During the exchange each device ensures that its set contains its own data (see Step 3 in Figure 2). At the same time, when users count correctly ($\tilde{N} = N$), the device only completes the exchange when N elements are in the set (Step 8). When there are more than N elements, the protocol quits and discards the data. If the protocol times out before N elements are collected, the protocol also quits. We also know that the sets are consistent for all N group members. If the set lacks exclusivity, some entity besides the N group members has added some element to the set. If the set lacks uniqueness, one group member has added two or more elements to the set. However, if either of these events occur, the group members' sets will be inconsistent or the set will be too large and the protocol will quit and report an error. Each member's set is consistent so the check that the set contains N elements ensures exclusivity and uniqueness. When an outsider adds data or a group member contributes multiple elements the cardinality of a consistent set must be larger than N. All N members have the same set and each member verifies their data is in the set so N is the smallest cardinality of the set. However, the checks in SPATE ensure that only N elements exist in the set. Based on the pigeonhole principle it is impossible for outsider- or additional insider-data to exist in the collected set. Given that users count correctly and devices only accept a set of the corresponding size, a consistent set must only contain the N elements from the N group members and thus exclusivity and uniqueness are ensured.