SPATE: Small-group PKI-less Authenticated Trust Establishment

Yue-Hsun Lin, Ahren Studer, Yao-Hsin Chen, Hsu-Chun Hsiao, Li-Hsiang Kuo,
Jason Lee, Jonathan M. McCune, King-Hang Wang, Maxwell Krohn,
Adrian Perrig, Bo-Yin Yang, Hung-Min Sun, and Phen-Lan Lin

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 mobile devices 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, file-sharing, and text messaging 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 example SPATE applications provide increased security with little overhead noticeable to users once keys are established.

Index Terms—Authentication; Security; 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 [1]–[3].

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 [4]. 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 are *mobile devices*—now more prevalent than ever with the proliferation of increasingly sophisticated mobile phones. As a personal agent, mobile phones can automate most key management and security configuration. Non-expert users are only required to perform a small number of well-instructed procedures in order to communicate securely with another phone owner. However, prior work fails to efficiently perform authenticated data exchange among a group of participants in a decentralized setting [5]–[9]. An efficient authentic exchange for groups remains a great challenge because wireless communication is unreliable and insecure; without physically interacting with each group member, the members have no guarantee that the group that is physically present is exactly the group in which members are exchanging information wirelessly.

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, file-sharing, or SMS (text-messaging). 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, in contrast, takes advantage of device mobility and face-to-face meetings to establish trust and simplify future communication.

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

- *Y.-H. Lin, Y.-H. Chen, K.-H. Wang, and H.-M. Sun are with the Department of Computer Science, National Tsing Hua University, Taiwan.*
- *A. Studer, H.-C. Hsiao, J. Lee, J.M. McCune, M. Krohn and A. Perrig are with Cylab, Carnegie Mellon University, PA*
- *L.-H. Kuo and B.-Y. Yang are with the Institute of Information Science, and Taiwan Information Security Center, Academia Sinica, Taiwan*
- *P.-L. Lin is with the Department of Computer Science, Providence University, Taiwan*

scenario is a Man-in-the-Middle (MitM) attack, by which a nearby adversary can inject bogus 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 [10], [11] computed over the exchanged data. The participants check that all devices agree on the hash. If both checks succeed, then the group participants 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. In such a scenario, long-term secrets need not be stored on mobile devices. This design provides improved security properties when compared to other key exchange protocols (Section 4). Some applications require their long-term secrets to be stored on the phone (e.g., our secure SMS application). However, if the private key remains on the user's workstation at home, the loss of a mobile device has zero impact on security.

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 [12]–[18], cumbersome key-exchange protocols [19], is vulnerable to malicious bystanders [20], or are restricted to two-party exchanges [5]–[9], [21]–[25]. Other work offers mechanisms optimized for large groups of 10 to 30 people [26]. This work focuses on small groups where users can accurately count the group size [27]: eight or fewer. Assuming group size follows 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 three applications: secure email, secure file-sharing, and secure short message service (SMS/text messaging). These 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. Secure SMS enables users to securely send short messages between mobile devices after the meeting. All 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) three 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 what they see and hear from 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 have already been 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. [26], 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 place limited expectations on the users. Humans are impatient and are inaccurate when comparing numbers [25]. 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 for 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 [27]. 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.

2.2 Trust Model

SPATE's trust model is built upon physical interactions via mobile devices. Having exchanged messages via the devices, 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 [28]–[30]. 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) [25].

Given humans' inability to accurately and quickly compare digits, researchers have proposed using text [31], [32] or visual [10], [11] representations of these checksums. The "Loud and Clear" system [32] expresses hashes as syntactically correct sentences, while the UIA system [31] 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" [11] and "Flag" [10] 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 [26]. Flags [10] 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 7 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 [33]. 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 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], [30]; string comparison that uses the human as a channel to ensure authentic exchange of information [28]–[30], [34]; audio-based comparison where the human user compares the strings via audio representation [32]; visual-based comparison of graphics that encode data [10], [11]; shaking devices to create shared entropy pools [5], [6], [35]; common properties of the wireless channel to establish authentic or secret information [22]; and location-limited channels [21], [23], [24].

Closely related to the SPATE exchange is GAnGS [26]. 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 [19]. 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 [12], [14]–[18]. 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 [20], [36], [37]. 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. 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.

Identity-Based Encryption (IBE) [38], [39] is another method to distribute public keys. In such a system, a subscriber's identity, e.g., email address, is her public key. With IBE, distributing public keys is easier than when using conventional PKI systems; instead of needing certificates, participants only require the correct identity of the other party. However, like traditional PKIs, IBE requires a shared trusted authority.

Finally, there is research using location-limited channels to exchange keys [13], [21], [24]. Talking to Strangers [21] and Capkun's work [13] 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

[1] Ellison and Dohrmann [10] use 6 bits representing 64 colors per rectangle, but with so many colors slight differences in shade may lead to errors during comparison.

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 [24] 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 [40] described a formal language for defining security requirements in CSCW, but ignored how to enforce those requirements. SPATE presents one potential way to enforce them.

5 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 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.

5.1 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]

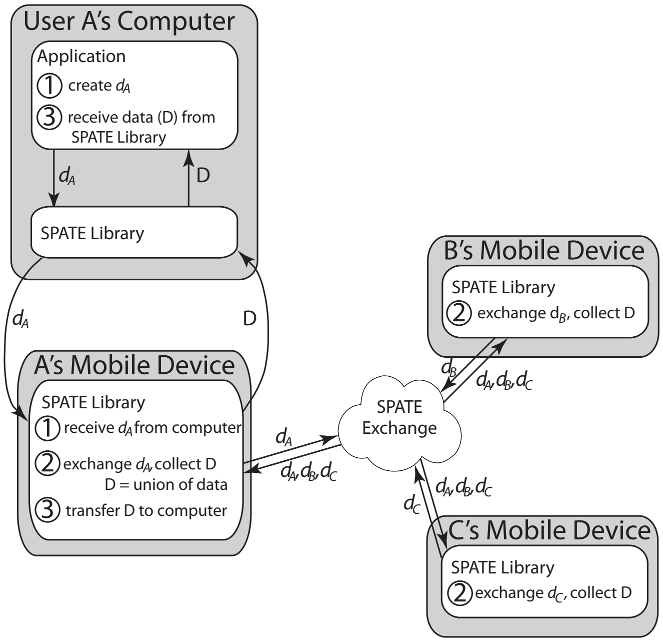


Fig. 1. Steps associated with a SPATE exchange between Users A, B, and C. Shown from the perspective of User A.

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

[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).

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 5.2 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 an association from a previous pairing to establish a secure channel which is used to transfer the collected data.

5.2 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 of four major steps: selection and counting, commitment, distribution, and verification. To ensure authenticity, each user must count only 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 4) where the device must perform private key operations.

```
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
After all phones receive $\tilde{N}$ commitments
9. $M_i \rightarrow *$    : $d_i, p_i$
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 $j \in 1 \ldots \tilde{N}$
                      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 $j \in 1 \ldots \tilde{N}$
                      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
```

Fig. 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.

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 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 [41] 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 [42]. 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). Our prior work contains more details about this use of commitments [43]. 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 fewer 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).

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}. (Our prior work contains a security analysis [43].)

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] An incorrect nonce can be detected by comparing the hash of the nonce to the associated protocol commitment (see Step 16). 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

[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.

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.

SPATE guarantees authentic information exchange if at least one member correctly compares T-Flags within the group. When all members are impatient to check the consistency of T-Flags, an addition to the SPATE verification step (between Steps 11 and 12) can prevent the device from saving bogus data as follows: SPATE periodically instructs the device to display a challenge T-Flag, chosen at random, before displaying the original T-Flag which represents the hash of the exchanged information as shown in Step 12. SPATE ensures that devices with an identical copy of data (i.e., $h(\mathbb{C}||\mathbb{D}||\mathbb{P})$) switch to the challenge phase together. Because the challenge T-Flags are chosen at random, they will be different within the group with high probability ($1-2^{-24(N-1)}$). As a consequence, some users must report "Some T-Flags Differ" in the challenge phase, in which case SPATE proceeds to show the original T-Flags (Step 12). On the other hand, SPATE aborts if all users click "All N T-Flags Match" without even looking at the display. Despite reducing the impact of human errors, this change requires an additional round of T-Flags comparison and broadcast, thus increasing the protocol runtime.

6 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 three applications on top of SPATE. In this section, we present a high-level overview of a secure email application, a secure file sharing application, and a secure short message service (SMS). In the following sections, we present our implementation and evaluation.

6.1 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 [44] 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 [45], 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 received is from the person she 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.

6.2 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 [4]. Furthermore, the file system should maintain accountability information and revocability to help detect and stop misbehaving users. Current solutions (e.g., BitTorrent [46], Dropbox [47], and KaZaA [48]) 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 users upload the configuration file to their respective workstations and

[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.

mount the remote file system. We implemented this application on top of *sshfs* [49], 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* 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.

6.3 Secure SMS

Short messaging service (SMS) is a very popular mobile phone service. In 2008, UK phone subscribers sent more then 1.1 billion text messages every week [50]. As mobile devices become more complex and are used for multiple functions, many attacks becomes possible. For example, criminals may send unsolicited fraudulent messages to subscribers informing them that they have won big prizes. Deceived recipients might send their banking information to the criminals in hopes of claiming such prizes. Several solutions have been proposed to protect users [51]–[54]. However, the security of these solutions depends on the trust establishment and security algorithms applied, with secure key distribution being the main bottleneck.

Our Secure SMS application leverages SPATE to achieve secure key distribution. Unlike the previous applications, Secure SMS uses symmetric cryptography with keys derived from mobile device generated Diffie-Hellman values. The private Diffie-Hellman value is kept secret in phone storage while the public value is distributed during the SPATE exchange. After the exchange, each member uses Diffie-Hellman key agreement to create pairwise keys with every other member. This pairwise symmetric key allows encryption and authentication of SMS messages with limited computation and bandwidth overhead.

Secure SMS can provide the following benefits:

1) *Effective Key Distribution.* Each user's mobile device generates her own Diffie-Hellman values. During a SPATE exchange, members exchange authentic public values which the mobile devices use to agree upon shared keys, without per pair exchanges.
2) *Authentic and confidential SMS.* A successful SPATE exchange ensures the authenticity of the public values and thus the security of the shared key. The shared key is used to authenticate and encrypt SMS messages between the two devices, preventing other devices from impersonating the devices or accessing their communication.
3) *Efficient performance.* Symmetric cryptography requires less computation and bits to provide the same privacy and authentication as asymmetric cryptography. These facts are crucial when using computationally limited mobile devices which pay per message.

7 Implementation

We have fully implemented the SPATE system and three applications on Nokia N70 and E51 smart phones and commodity Dell workstations running Windows XP and Ubuntu Linux. The system contains four parts: 1) the SPATE Mobile Client that supports key exchange for the email, file-sharing, and SMS applications, 2) a Thunderbird plug-in to enable secure email, 3) a file sharing application, and 4) a Secure SMS application (Figure 3). We have also implemented the SPATE system on Apple iPhones. However, Apple's isolation between applications prevents us from implementing secure SMS on the iPhone. In the following sections, we describe the implementation details of these programs.

7.1 Nokia 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 [8], [26].

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

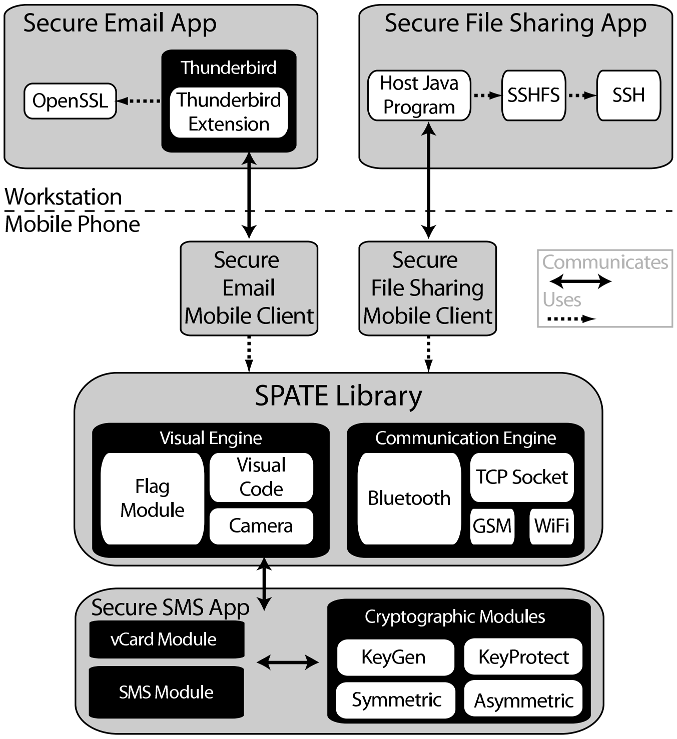


Fig. 3. SPATE system overview.

transmission and contains the Bluetooth module. The visual engine is used to generate T-Flags. As described in Section 5.2, 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 [55]. Thus, we augment our visual engine to generate, photograph, and decode two-dimensional barcodes (2D barcodes) which we use to circumvent Bluetooth device discovery, as proposed by Scott et al. [55].

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, we would have a broadcast primitive available.

2D Barcodes are generated, photographed, and decoded using the VisualCodes module from Rohs and Gfeller [56], ported to work with newer versions of Symbian OS. The T-Flags module is used at the end of an authenticated data exchange; it displays a visual hash on devices' screens (Figure 4).

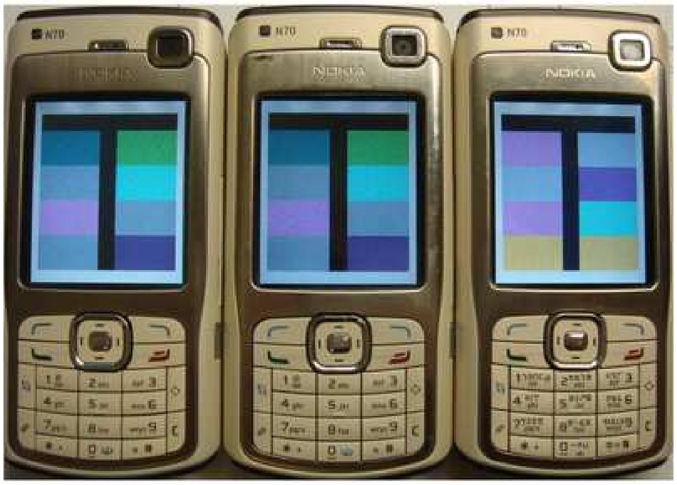


Fig. 4. Our Mobile Client displaying T-Flags on N70 smart phones during a SPATE exchange. The left and center T-Flags are identical, but the right T-Flag is different.

7.2 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 5.2. The only significant departure from the SPATE protocol in Section 5.2 is the additional requirement that the people in the prospective group agree on a *leader* to serve as the hub of the star network to simulate 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 a Mobile Client or initiates an exchange within the Secure SMS App on her mobile device. Step 1a in Figure 5 shows the library 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 [27]). Once the count has been entered, the library 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 SPATE Library displaying a barcode, and Step 1d shows the leader's library successfully decoding the barcode on another device. Once the leader has photographed all members' barcodes (detected automatically since the library 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 remain-

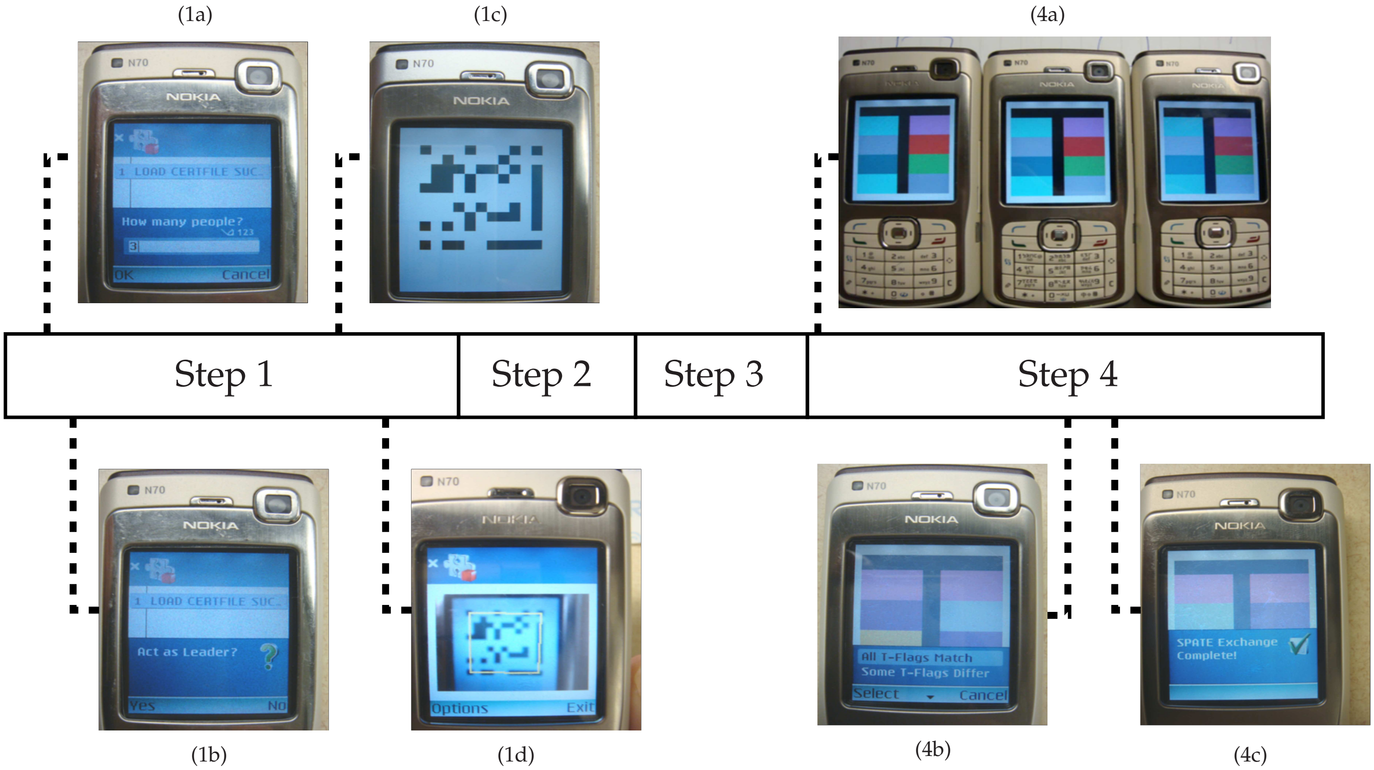


Fig. 5. Execution Flow of a SPATE Exchange. Step 1: Selection and Counting, Step 2: Commitment, Step 3: Distribution, Step 4: Verification. Steps 1b, 1c and 1d are necessary in our implementation because Bluetooth does not support broadcast.

ing 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 Library. We design our library 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 information for use with the associated application. It also displays the message, "SPATE Exchange Complete!" (Step 4c).

7.3 iPhone SPATE Library

SPATE implementation and operation on the iPhone is similar to that on the Nokias, but is hindered by Bluetooth limitations on the iPhone. The library implementation uses Apple's Game Kit Framework to connect collocated devices using Bluetooth. Unfortunately, the kit is structured such that the Bluetooth discovery phase cannot be bypassed. As such, iPhones are forced to use traditional device discovery, rather than capturing barcodes, to connect to other group members. Once the group members have counted and are connected, the remainder of SPATE operation is the same: commitment, distribution, and verification via T-Flag.

7.4 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 [57] 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.

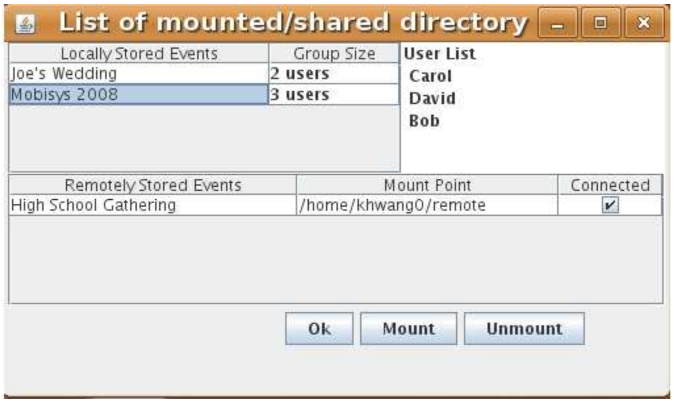
After the user has participated in a SPATE exchange, her device will have obtained self-signed public key certificates from other users. She can upload all received certificates from her Mobile Client to the Thunderbird extension. The extension automatically signs[6] the received certificates with the 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 [58] 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 the sender's private key).

7.5 File Sharing

Our file sharing program is built for Linux using Java 6 and shell scripts, on top of the *SSH File System (SSHFS)* [49]. 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 identify 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.

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 [59]). 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.


List of mounted/shared directory

Locally Stored Events	Group Size
Joe's Wedding	2 users
Mobisys 2008	3 users

User List
Carol
David
Bob

Remotely Stored Events	Mount Point	Connected
High School Gathering	/home/khwang0/remote	☑

Fig. 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.

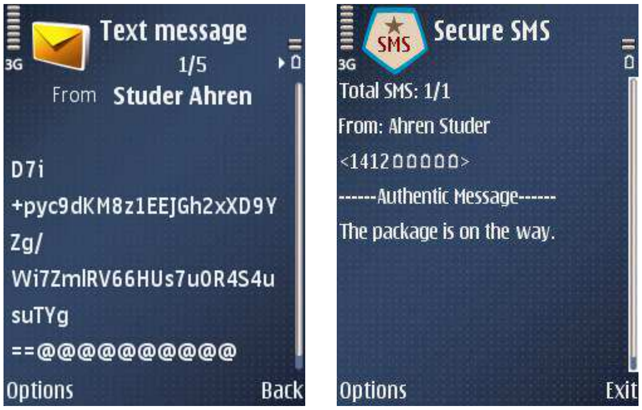
The file-sharing application displays information about currently shared folders, active groups, and active users on the local machine (Figure 6). It also enables the user to mount shared folders from remote machines.

7.6 Secure SMS

Secure SMS leverages a SPATE exchange and an application on the mobile phone to provide secret and authentic communication between group members without a PKI. The secure SMS application consists of three modules: a vCard module, a SMS module, and a cryptographic module. The vCard module creates a vCard from the device's information and extracts information from other devices vCards after a successful SPATE exchange. The SMS module handles the sending or receiving of SMS messages. The cryptographic module contains primitives to perform symmetric and asymmetric cryptographic operations and KeyGen and KeyProtect modules to generate and protect secrets, respectively.

To use secure SMS, the application creates the user's vCard, distributes the vCard and receives others' vCards during a SPATE exchange, uses the received vCards to established shared keys, and uses a shared key to securely send or receive messages. vCard generation includes the generation of Diffie-Hellman values using the KeyGen module and packing the public value with the user's name, phone number, and any other data the user wants to share (e.g., a personal photo). After a SPATE exchange, the vCard module extracts the public Diffie-Hellman value and contact information from the received vCards. Rather than performing key agreement for every message, the cryptographic module performs

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



Text message

1/5

From Studer Ahren

D7i
+pyc9dKM8z1EEJGh2xXD9Y
Zg/
Wi7ZmlRV66HUs7u0R4S4u
suTYg
==@@@@@@@@@@

(a) Transferred Data

Secure SMS

Total SMS: 1/1

From: Ahren Studer

<1412□□□□□□>

------Authentic Message------

The package is on the way.

(b) Decrypted & Verified Message

Fig. 7. The data transferred as part of secure SMS and the decrypted message at the receiver.

Diffie-Hellman key agreement after each successful exchange to generate shared secrets. When not in use, the KeyProtect sub-module uses a password to encrypt the user's private Diffie-Hellman value and any shared secrets stored on the phone.

When a user wants to send a secure SMS, the shared key is retrieved and session encryption and authentication keys are generated and used to secure the message before it is sent. After defining the recipient and entering the desired message, the user enters her password to access any secrets stored on the phone. With access to the shared keys, the cryptographic module generates an encryption key for use with AES-CBC and an authentication key for use with HMAC-SHA1. These symmetric operations provide space and computational efficiency compared to using asymmetric encryption or authentication. The cryptographic module encrypts the message, and generates the corresponding MAC. HMAC-SHA1 takes input ciphertext and IV (initialization vector) to output the MAC [60], [61]. Finally, the SMS Module sends the data over the cellular network (see Figure 7 (a) for an example ciphertext). After receiving the encrypted message, the receiver enters her password to access the corresponding shared key. The cryptographic module can then decrypt and verify the authenticity of the message (see Figure 7 (b) for the decrypted and verified message).

8 Evaluation

We evaluate the performance of authenticated 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 data), since synchronizing data between a mobile device and workstation is a widely available operation.

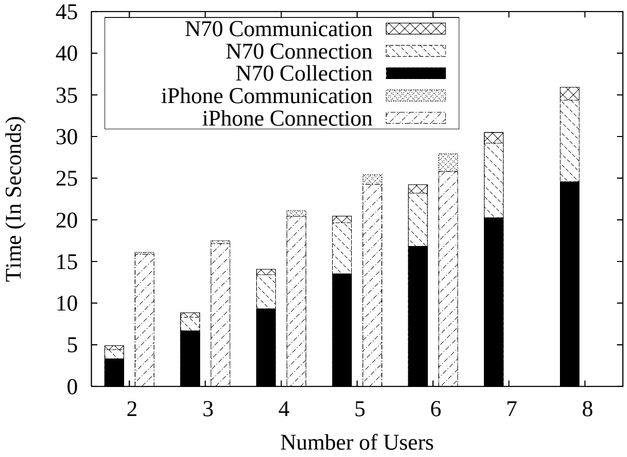


Fig. 8. Time consumed during a SPATE exchange.

8.1 Method

We ran SPATE on two to eight Nokia N70 smart phones and two to eight Apple iPhones. 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 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. *Communication* includes the time for data transfer during the automated Commitment and Distribution (Steps 2 and 3 from Section 5.2 and Figure 5) steps of the SPATE exchange. The iPhone forces applications to use traditional Bluetooth discovery and thus lacks any collection (see Section 7.3). We also ran Seeing-is-Believing [8] and GAnGS [26] on the Nokia N70s to compare the total runtime for each protocol. To eliminate human factors in the execution time, these tests were performed by experienced operators of all three systems.

8.2 Results

Figure 8 shows the time consumed by *Collection*, *Connection*, and *Communication* for the N70 and iPhone based implementations. Unfortunately, in our experience, the, iPhone's underlying Bluetooth library becomes unstable when more than 6 devices attempt to form a piconet, so we only present results for 2 to 6 devices.

During *Collection*, the leader photographs $N - 1$ 2D barcodes from the 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 the Nokia implementation, 2-3 seconds is considerably less than the delay 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. [55]), we implemented a Symbian C++ program to record the time spent on device and

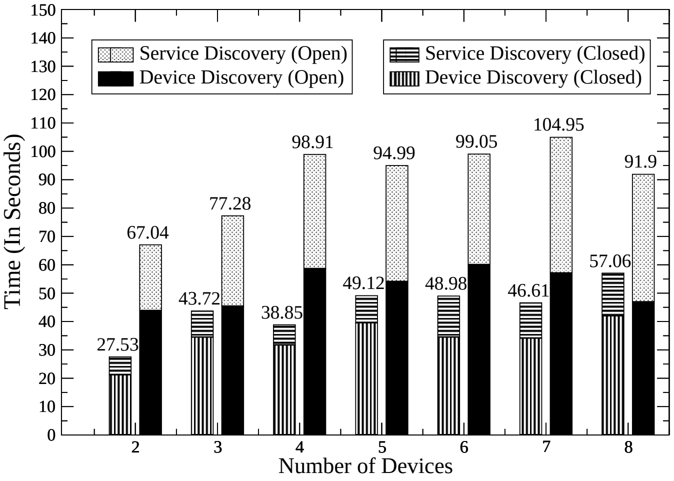


Fig. 9. N70 Bluetooth discovery overhead.

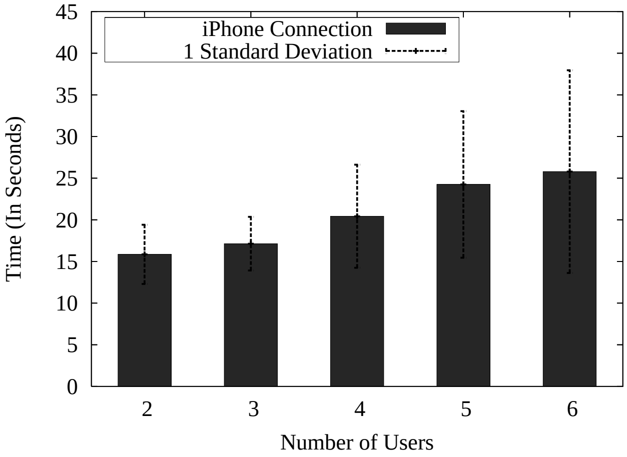


Fig. 10. iPhone Bluetooth discovery overhead.

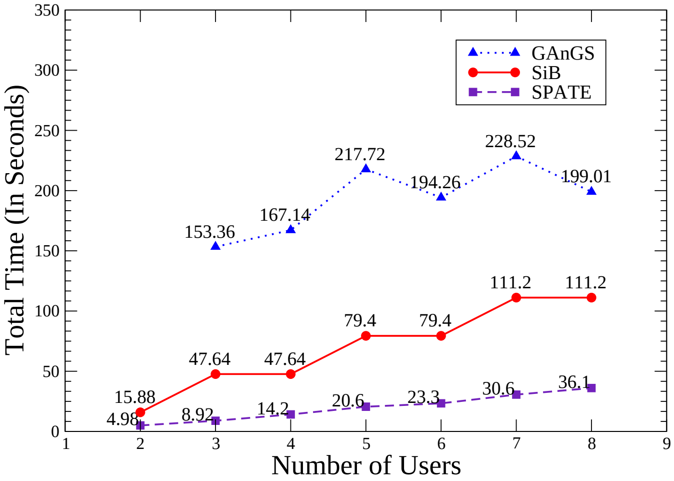


Fig. 11. Comparison between SPATE, SiB, and GAnGS.

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 9 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 *N70 Collection*, the leader's device establishes a Bluetooth piconet. This results in the *N70 Connection* overhead in Figure 8, which takes roughly one to ten seconds, depending on how many devices are involved.

The iPhone implementation forces devices to perform Bluetooth discovery. This causes highly variable *iPhone Connection* times as seen in Figures 8 and 10. Figure 10 represents the average time needed for the iPhone implementation to form a piconet with 2 to 6 devices while in a closed environment. Compared to the N70, the iPhone provides faster average Bluetooth connections (Figure 9 versus Figure 10). However, the current iPhone Bluetooth library fails to provide the same scalability (i.e., piconets of 7 or 8 devices cannot be reliably established in a reasonable amount of time) and suffers from high variability even with zero interfering devices.

Once the connections are established, *Communication* consumes less than five seconds even with a full eight devices. Even with our star network topology, Bluetooth has sufficient bandwidth to rapidly transfer the data, commitments, and nonces, 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 exchange with eight users in less than one minute.

Comparison with Existing Systems Next, we present a comparison between SPATE and two prior key-exchange systems using the N70 smart phones: Seeing-is-Believing and GAnGS (Figure 11). 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 [56]. 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.

9 Discussion

In this section, we discuss whether counting is necessary or not.

In SPATE, group members count the number of members present to prevent non-group members from adding their data. 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.

10 Conclusion

We have presented SPATE, a system for authentic exchange of information in groups of up 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.

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, secure file sharing, and secure SMS.

References

[1] A. Whitten and J. Tygar, "Why Johnny can't encrypt," in *USENIX Security*, Aug. 1999.

[2] S. Sheng, L. Broderick, J. Hyland, and C. Koranda, "Why johnny still cant encrypt: Evaluating the usability of email encryption software," in *Symposium On Usable Privacy and Security*, 2006.

[3] S. Gaw, E. W. Felten, and P. Fernandez-Kelly, "Secrecy, flagging, and paranoia: adoption criteria in encrypted email," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2006, p. 600.

[4] N. S. Good and A. Krekelberg, "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.

[5] C. Castelluccia and P. Mutaf, "Shake Them Up! A movement-based pairing protocol for CPU-constrained devices," in *Proceedings of ACM/Usenix MobiSys*, 2005.

[6] J. Lester, B. Hannaford, and B. Gaetano, "Are you with me? - Using accelerometers to determine if two devices are carried by the same person," in *Proceedings of Pervasive*, 2004.

[7] C. Soriente, G. Tsudik, and E. Uzun, "BEDA: Button-Enabled Device Association," in *International Workshop on Security for Spontaneous Interaction (IWSSI)*, 2007.

[8] J. M. McCune, A. Perrig, and M. K. Reiter, "Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 2005.

[9] C. Soriente, G. Tsudik, and E. Uzun, "HAPADEP: Human Asisted Pure Audio Device Pairing," in *Information Security Conference (ISC)*, Sep. 2007.

[10] C. Ellison and S. Dohrmann, "Public-key support for group collaboration," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 4, pp. 547–565, 2003.

[11] A. Perrig and D. Song, "Hash Visualization: A New Technique to improve Real-World Security," in *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)*, M. Blum and C. H. Lee, Eds., Jul. 1999, pp. 131–138.

[12] M. Burmester and Y. Desmedt, "Efficient and Secure Conference Key Distribution," in *Security Protocols—International Workshop*, ser. Lecture Notes in Computer Science, vol. 1189. Springer-Verlag, Apr. 1997, pp. 119–129.

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

[14] M. Just and S. Vaudenay, "Authenticated Multi-Party Key Agreement," in *Advances in Cryptology – (ASIACRYPT)*, ser. Lecture Notes in Computer Science, vol. 1163. Springer-Verlag, 1996, pp. 36–49.

[15] Y. Kim, A. Perrig, and G. Tsudik, "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.

[16] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A Secure Audio Teleconference System," in *Advances in Cryptology (Crypto)*, ser. Lecture Notes in Computer Science, vol. 403, International Association for Cryptologic Research. Springer-Verlag, 1990, pp. 520–528.

[17] M. Steiner, G. Tsudik, and M. Waidner, "Key Agreement in Dynamic Peer Groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 8, pp. 769–780, Aug. 2000.

[18] W.-G. Tzeng and Z. Tzeng, "Round-Efficient Conference-Key Agreement Protocols with Provable Security," in *Advances in Cryptology – (ASIACRYPT)*, ser. Lecture Notes in Computer Science, vol. 1976, International Association for Cryptologic Research. Springer-Verlag, 2000, pp. 614–628.

[19] V. A. Brennen, "The Keysigning Party HOWTO," http://cryptnet.net/fdp/crypto/keysigning_party/en/keysigning_party.html, Jan. 2008.

[20] N. Asokan and P. Ginzboorg, "Key-agreement in ad-hoc networks," *Computer Communications*, vol. 23, no. 17, pp. 1627–1637, Nov. 2000.

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

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

[23] NFC Forum, "NFC Forum: Specifications," http://www.nfc-forum.org/specs/.

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

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

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

[27] C. Kuo, "Reduction of End User Errors in the Design of Scalable, Secure Communication," Ph.D. dissertation, Carnegie Mellon University, 2008.

[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, Aug. 2006.

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

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

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

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

[33] C. Glasbey, G. van der Heijden, V. F. K. Toh, and A. Gray, "Colour displays for categorical images," *Color Research and Application*, vol. 32, no. 4, pp. 304–309, Jun. 2007.

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

[35] L. E. 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 *Proceedings of Ubicomp*, 2001.

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

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

[38] S. A., "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1984, pp. 47–53.

[39] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in CryptologyCRYPTO 2001*. Springer, 2001, pp. 213–229.

[40] S. N. Foley and J. Jacob, "Specifying security for CSCW systems," in *8th IEEE workshop on Computer Security Foundations*, 1995.

[41] M. Blum, "Coin flipping by telephone," in *Advances in Cryptography*, Aug. 1982, pp. 11–15.

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

[43] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun, and B.-Y. Yang, "SPATE: Small-group PKI-less authenticated trust establishment," in *Proceedings of the 7th Annual International Conference on Mobile Systems, Applications and Services (MobiSys 2009)*, Jun. 2009.

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

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

[46] B. Cohen, "Bittorrent," http://www.bittorrent.com, Apr. 2001.

[47] D. Houston and A. Ferdowsi, "Dropbox," https://www.getdropbox.com/, Sep. 2008.

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

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

[50] MDA: Mobile Data Association, "The Q1 2008 UK Mobile Trends Report," http://www.swiftcrm.net/MDA_Q1_2008_UK_mobile_report.pdf, 2009.

[51] L. Barbi, "Spidersms-sending and reception of encrypted sms," 2008.

[52] A. Grillo, A. Lentini, G. Me, and G. F. Italiano, "Transaction oriented text messaging with Trusted-SMS," in *Proceedings of the 2008 Annual Computer Security Applications Conference*. IEEE Computer Society, 2008, pp. 485–494.

[53] Kryptext, "Kryptext-Offers software to encrypt SMS text messages from mobile to PC," http://www.kryptext.co.uk/.

[54] CryptoSMS, "CryptoSMS-protecting your confidential sms messages," http://www.cryptosms.com/, 2008.

[55] D. Scott, R. Sharp, A. Madhavapeddy, and E. Upton, "Using Visual Tags to Bypass Bluetooth Device Discovery," *ACM Mobile Computer Communications Review*, vol. 9, no. 1, pp. 41–53, Jan. 2005.

[56] M. Rohs and B. Gfeller, "Using Camera-Equipped Mobile Phones for Interacting with Real-World Objects," *Proceedings of Advances in Pervasive Computing*, pp. 265–271, Apr. 2004.

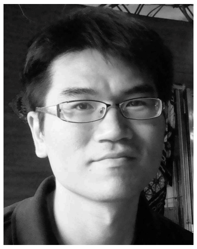
[57] M. J. Cox and R. S. Engelschall, "OpenSSL: Open Source toolkit implementing for SSL/TLS," http://www.openssl.org/, May 1999.

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

[59] "scponly," http://sublimation.org/scponly/, 2009.

[60] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," *Advances in Cryptology – ASIACRYPT 2000*, pp. 531–545, 2000.

[61] H. Krawczyk, "The order of encryption and authentication for protecting communications (or: How secure is SSL?)," in *Advances in Cryptology – CRYPTO 2001*. Springer, 2001, pp. 310–331.

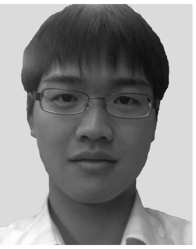


Yue-Hsun Lin is a Ph.D. candidate in Computer Science at National Tsing Hua University. His research interests include wireless sensor nework, network security, and applied cryptography.

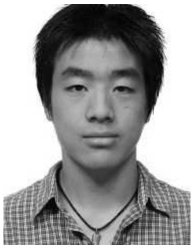
Ahren Studer is a Ph.D. candidate in Electrical and Computer Engineering at Carnegie Mellon University working with Adrian Perrig. His research interests revolve around authentication and trust establishment in ad hoc networks.

Yao-Hsin Chen is a Ph.D. candidate in Computer Science at National Tsing Hua University. His current research interests include provable security, applied cyptography, and network security.

Hsu-Chun Hsiao is a Ph.D. candidate in Electrical and Computer Engineering at Carnegie Mellon University working with Adrian Perrig. Her main research interest is in network security and privacy.



Li-Hsiang Kuo received his B.E. degree in Computer Science and Information Engineering from National Taiwan University. He was a research assistant for Dr. Bo-Yin Yang in the Institute of Information Science at the Academia Sinica since 2008. His research interests include cryptography and algorithms.



Jason Lee Jason Lee is a graduate student at Carnegie Mellon University. He earned his Bachelors Degree in Computer Science at Carnegie Mellon and is currently pursuing a Masters Degree in the same field under professor Adrian Perrig. His main area of research is secure communication between mobile devices.

Jonathan M. McCune is a Research Systems Scientist for CyLab at Carnegie Mellon University. He earned his Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University, and received the A.G. Jordan thesis award. Jonathan's research interests include secure systems, trusted computing, virtualization, and spontaneous interaction between mobile devices.

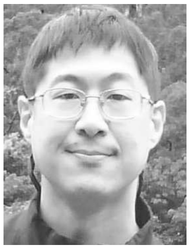
King-Hang Wang received his Ph.D degree in Computer Science from the National Tsing Hua University. Currently he serves at the Center of Cyber Logistic at the Chinese University of Hong Kong. His research interests include provable security, digital rights management, steganography.

Maxwell Krohn received his PhD degree in Computer Science at MIT. He co-founded and co-built several community Web sites, some vintage (TheSpark.com), others live and kicking (SparkNotes.com and OkCupid.com). His research interests are in operating systems, distributed systems and security.

Phen-Lan Lin received her Ph.D.degrees in Electrical Engineering from Southern Methodist University. She is a professor in the Department of Computer Science and Information Engineering at Providence University. Her current research interests are in the fields of multimedia security, network security, medical imaging, and visual inspection.

Adrian Perrig earned his Ph.D. degree in Computer Science from Carnegie Mellon University. Currently he is a Professor in Electrical and Computer Engineering, Engineering and Public Policy, and Computer Science at Carnegie Mellon University. Adrian serves as the technical director for Carnegie Mellon's Cybersecurity Laboratory (CyLab). He is a recipient of the NSF CAREER award in 2004, IBM faculty fellowships in 2004 and 2005, and the Sloan research fellowship in 2006.

Hung-Min Sun received his Ph.D. degree in computer science and information engineering from National Chiao-Tung University. Currently, he is a professor with the Department of Computer Science, National Tsing Hua University. His research interests include information security, wireless network security, cryptography and multimedia security.



Bo-Yin Yang received his Ph.D. degree in mathematics from MIT and taught math in Taiwan. Currently he works in the Institute of Information Science at the Academia Sinica, where he leads the most active crypto group in Taiwan, specializing in efficient implementations and cryptanalysis.