SafeSlinger: Easy-to-Use and Secure Public-Key Exchange

Michael Farb
CyLab / CMU

Yue-Hsun Lin
CyLab / CMU

Tiffany Hyun-Jin Kim
CyLab / CMU

Jonathan McCune
Google Inc.

Adrian Perrig
ETH Zürich, CyLab / CMU

ABSTRACT

Users regularly experience a crisis of confidence on the Internet. Is that email or instant message truly originating from the claimed individual? Such doubts are commonly resolved through a leap of faith, expressing the desperation and helplessness of users.

To establish a secure basis for online communication, we propose SafeSlinger, a system leveraging the proliferation of smartphones to enable people to securely and privately exchange their public keys. Through the exchanged authentic public keys, SafeSlinger establishes a secure channel offering secrecy and authenticity, which we use to support secure messaging and file exchange. SafeSlinger also provides an API for importing applications' public keys into a user's contact information. By slinging entire contact entries to others, we propose secure introductions, as the contact entry includes the SafeSlinger public keys as well as other public keys that were imported. We present the design and implementation of SafeSlinger for Android and iOS, which is available from the respective app stores. An overview video of SafeSlinger is available at: `http://www.youtube.com/watch?v=IFXL8fUqNKY`

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*

Keywords

Trust Establishment; Security & Privacy; Secure Communication.

1. INTRODUCTION

For many current Internet applications, users experience a crisis of confidence. Is the email or message we received from the claimed individual or did an impostor send it? Many useful protocols such as SSL/TLS or PGP have been proposed for entities that already share authentic key material. However, the drawbacks and weaknesses of the global certification process for SSL/TLS [23,28] and the usability challenges of decentralized mechanisms such as PGP [36] are well known. Furthermore, even with these protocols, the root of the problem still remains: how do we obtain the authentic public key from the intended resource or individual?

The human-centric foundation of trust establishment makes this problem universally important; protocols and interfaces need to be designed for a diverse population with varying skills, interests, ages, and cultures. We postulate that usability is one of the major barriers for widespread adoption of cryptography.

The recent proliferation of smartphones offers a promising opportunity to address these challenges, as these devices offer a general computing environment with a powerful processor, high-resolution display, several communication modalities (WiFi, 3G/4G, Bluetooth, NFC), camera, and sensors. Thus, smartphones can enable convenient, spontaneous exchanges of public keys.

We observe that individuals often have physical interactions with resources or other individuals before communicating digitally. Often, people communicate over the Internet or via SMS *after* having met in person. We leverage this *physical* encounter to bootstrap *digital* trust. People who communicate before physically meeting can bootstrap trust through a secure introduction mechanism that is rooted in physical encounters with a common acquaintance.

In this paper, we present SafeSlinger, a system for secure exchange of authentic information between two smartphones and a user interface for secure messaging while preserving secrecy. In essence, SafeSlinger exchanges contact information, containing public keys in addition to standard contact list information such as name, picture, phone numbers, email addresses, etc. Thanks to the association between the individual holding the phone and the exchanged public key, users can later associate digital communication with the previously met individual by verifying a digital signature. To make SafeSlinger usable, the cryptographic aspects are mostly hidden from the user, and we have added several mechanisms to make SafeSlinger tolerant to user error.

We envision SafeSlinger as a general approach to bootstrap secure digital communication. First, we enable groups of up to 10 individuals of physically co-located users to securely bootstrap trust by slinging keys between their devices (a one-time operation).[1] SafeSlinger can also support remote exchanges, as long as users can authenticate the other individuals, e.g., via telephone communication or live video conference. Second, SafeSlinger supports secure phone-to-phone messaging and file transfer, providing both secrecy and authenticity. Once users' devices hold each other's public keys, the SafeSlinger user experience is comparable to that of traditional SMS and MMS messaging today. Third, SafeSlinger enables *secure introductions* without physical meetings by allowing a common acquaintance to facilitate a mutual introduction

[1]For more than 10 users, Ho-Po Key [25] may be used to exchange keys among members in large groups.

MobiCom'13, September 30-October 4, Miami, FL, USA.

using SafeSlinger file transfer. Fourth, we enable other applications to use the SafeSlinger API to add their public key to a contact entry. Now, when a user slings its updated contact list entry to another user, each application's public key is automatically included, and the receiver's corresponding application can extract the public key. This mechanism can enable applications such as secure email, secure SMS, and encrypted file sharing to solve the problem of securely exchanging the public key without requiring a leap of faith.[2]

Contributions. SafeSlinger is the first complete system that provides privacy-preserving and secure group credential exchange without any external trusted parties, restricting the exchanged information to other group members only. SafeSlinger is also the first secure group credential exchange system that can be used remotely over a telephone or video conferencing line. SafeSlinger is designed to be easy to use and defend against all attacks we are aware of. We implement SafeSlinger as an open-source project and make it available for free on Android and iOS app stores.

2. PROBLEM SETTING

In this section, we define the problem we set out to solve, discuss our goals, present assumptions that need to hold, the adversary that we defend against, and present attacks.

2.1 Problem Definition

The basic primitive that we seek to accomplish is to securely exchange information that is associated with the intended individual participating in an exchange. The security properties that we seek are information secrecy (only the intended entities receive the information), and user-verifiable trustworthy association of data to an honest individual (user knows exactly the information that is associated with a specific honest individual).

Note that there are fundamental limits on the ability of an electronic protocol to protect one human against another human with the intent to deceive, hence the adjective "honest." Specifically, when the exchange involves more than two people, the group exchange should produce the exact same security properties that would result from pairwise exchanges between all members of the group. For example, if Fred, George, and Harry perform an exchange, and Harry *behaves adversarially*, we still wish for the information exchanged between Fred and George to satisfy all security properties.

Given such a secure exchange that includes a public key, all subsequent mechanisms for authentic and secret communication can be implemented using well known protocols, relying on the correct binding of the public key to the individual.

2.2 Goals

Our core goal is to enable groups of two or more users to spontaneously exchange their contact list information. To enable such a usable system and maintain high security, we outline four essential properties required in SafeSlinger to exchange contact data:

- **Scalable:** We emphasize groups because we wish to avoid the tedium of a group of users exchanging their information via $N(N-1)/2$ pairwise exchanges.
- **Easy to use:** We also emphasize usability while retaining the security properties described above.
- **Portability:** Another goal is to support heterogeneous platforms, thus enabling interactions among smartphones of several manufacturers and running different operating systems.
- **Authenticity:** Each user should obtain the correct contact information of other users.
- **Secrecy:** Contact information is available only to other group members after a successful completion of the exchange, and remain hidden from external adversaries.

[2]A *leap of faith* is blind trust in an initially received public key, also known as *Trust On First Use* or TOFU).

2.3 Assumptions

Our protocol does need to assume certain user behaviors to execute successfully. First, we assume that users are computer literate and can follow basic instructions. We also assume that users have a natural desire for security and that they do not want to deliberately disclose their private information. We also assume that users can authenticate (in the human sense, e.g., recognizing the other users' appearance, voice, etc.) the individuals that they perform information exchanges with, such that an adversary who impersonates another individual would be detected through personal identification. More specifically with respect to the SafeSlinger protocol, we assume that users perform the following security-relevant operations:

- **Exclusion of unintended participants:** Legitimate users will expel an unwanted bystander who wants to participate in the protocol.
- **Correct member count:** Users need to correctly count the number of group members who participate in an exchange.
- **Identity validation:** Users correctly validate the identity information received from the exchange. They should map the identity information to the people who participate in the exchange, and they should reject information of non-participants.
- **Impersonation detection:** Users verify that no other user has injected information that impersonates them in the current exchange. For example, a malicious user may also inject information about Alice, even though Alice is also participating in the exchange. The risk is that another user may discard the correct information and accept the wrong information.
- **Diligent comparison:** Users perform a comparison of three words with all other participants, even after executing the protocol numerous times without any attack.
- **Diligent error checking and aborting:** Users will abort the protocol and restart the protocol when suspicious or error conditions are encountered.

SafeSlinger is not limited to only smartphone users; the protocol can be implemented on laptop or desktop computers. However, we focus on participants who use their smartphones frequently since smartphones are easy to use, are more available during casual encounters, and also suitable for carrying personal information, such as an electronic business card.

We assume that the smartphone hardware and software is free of vulnerabilities and malware, as these aspects are outside the scope of this work.

2.4 Adversary Model

We assume that some of the legitimate users that participate in the protocol may be malicious (We call the other users honest). We consider a Dolev-Yao style network adversary that has complete control over all network messages. Furthermore, any Internet server we may use during protocol operations may be malicious. We also assume that other adversaries may be physically present in the space where information exchanges happen. We consider an adversary who wants to break the properties as described in the problem definition, i.e., violate secrecy and authenticity properties of the information exchange and subsequent communication. We consider denial-of-service attacks to be out of scope, as we assume that the users may re-start the protocol as needed.

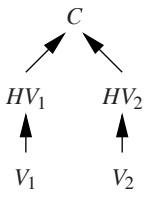


Figure 1: Multi-value commitment structure for authenticating and disclosing either V_1 or V_2.

2.5 Attacks

Secure local exchange of information is a surprisingly intricate and challenging problem. Possible attacks include:

- **Malicious bystander who participates in protocol:** A bystander can overhear conversation and attack the protocol by controlling the local wireless communication, performing a **Man-in-the-Middle (MitM) attack** on all participants.
- **Malicious group member:** A member of the group wants to violate protocol properties, such as mounting an **impersonation attack** by injecting incorrect information for another user, or performing a **Sybil attack** [12] by injecting multiple entries either for fictitious individuals or for individuals who are not present.
- **Group-in-the-Middle (GitM):** In a GitM attack [16], a malicious group member creates separate groups for the victim members and creates a sufficient number of virtual identities such that each group has the correct number of members. Each victim member believes to be part of a group with N members (with the N-1 other members), but instead the victim is in a separate group with the adversary and N-2 fake members. This attack is powerful as it breaks several proposed protocols, as we briefly discuss in Section 9.
- **Malicious server:** For protocols that rely on a back-end server, the server may be controlled by a malicious administrator or become compromised.
- **Information leakage after protocol abort:** An adversary may be able to cause a protocol abort and trigger leakage of private information about a participant.
- **Collision attack on low-entropy hash:** Low-entropy hash values are vulnerable to efficient attacks unless precautions are taken [8, 17, 18, 35, 37].

3. CRYPTOGRAPHIC BACKGROUND

We provide background on the cryptographic mechanisms used in this paper: multi-value commitments and group Diffie-Hellman key agreement.

3.1 Multi-Value Commitments

A cryptographic commitment protocol is used to lock an entity to a value V without disclosing V. Based on the commitment value, the decommitment can be validated and the protocol ensures that the correct value V is disclosed. A commitment protocol for value V can proceed as follows: $C = H(V,R)$, where C is the commitment value, H is a cryptographic hash function that is one-way, collision-free, and has pseudo-random output if R is a random unpredictable nonce. Thanks to the properties of the hash function and the randomness of R, V cannot be inferred from the commitment C. To open the commitment, V and R are disclosed. The collision resistance property of H ensures that it is computationally infeasible to find another V or R that will result in the same commitment C. Note that if V is unpredictable (e.g., a freshly generated ephemeral public key), the additional nonce value R is not needed and we simply have $C = H(V)$.

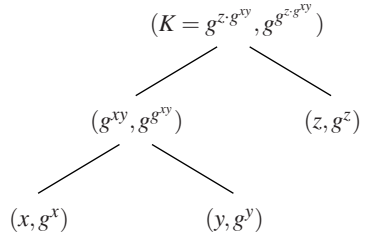


Figure 2: Group DH key agreement tree for 3 members (*mod p* operations omitted to improve readability). The notation is (priv, pub), where the first element in the parenthesis lists the DH private key, and the second lists the DH public key. The group key is the private key K at the root.

In case we want to commit to either value V_1 or V_2 (deciding which to actually release at a future time) with a single commitment, we can employ a tree-like structure (Figure 1). $HV_1 = H(V_1)$, $HV_2 = H(V_2)$, and $C = H(HV_1 \,||\, HV_2)$, where $||$ indicates the concatenation operator. This structure enables decommitment of either V_1 or V_2 without disclosing the other. For example, to decommit V_1, we disclose V_1 and HV_2, and the case for V_2 is analogous. Note that this example is for the case when V_1 and V_2 are unpredictable to the adversary; additional use of nonces is required for well known V_1 or V_2. This type of structure is similar to one-time signatures [24].

In the SafeSlinger protocol, we further make use of hierarchical commitments, where the decommitment value is again a commitment value.

3.2 Group Diffie-Hellman Key Agreement

Group Diffie-Hellman (DH) key agreement is a generalization of the two-party DH key agreement [11], where multiple parties participate to establish a common group key. The Cliques protocol is an example for group DH key establishment [31], and STR [30] and TGDH [15] protocols are tree-based group DH protocols. In the STR protocol, the tree shape is a maximally unbalanced tree (resembling a comb), and the TGDH protocol uses a balanced tree. Research has shown that total protocol latencies are lower for STR in environments where the communication latency dominates over the computation of a modular exponentiation [14], which is the case in mobile environments with smartphones. Hence, we will make use of STR, which we briefly describe in this section.

In group DH protocols, each participant is placed at a leaf node of a binary tree, where each node of the tree has a private and a public key associated with it. The value of a leaf node is the private and public DH keys of the member at that node. The value of the parent node is derived through the DH operation on the values of the two child nodes, for example if the values of the child nodes are x for the private and $g^x \bmod p$ for the public key of the left child, and y for the private and $g^y \bmod p$ for the public key of the right child, the parent value is $g^{xy} \bmod p$ for the private and $g^{g^{xy} \bmod p} \bmod p$ for the public value. Figure 2 illustrates a group DH key agreement with three members, where each node in the tree lists the private and public DH keys. The private key that corresponds to the root node is the shared secret of all group members.

4. SafeSlinger EXCHANGE PROTOCOL

4.1 Overview

The main purpose of SafeSlinger is to enable a set of users to exchange their contact information such that every non-malicious

user receives the correct information about every other non-malicious user. Malicious users may collude and impersonate each other, for example, therefore we cannot provide any guarantees for those parties. Our main goal is to provide security and usability while preventing the attacks described in Section 2.5.

The first hurdle we need to overcome is to enable communication among the users' mobile devices. We currently target Android and iPhone devices, with an effort to make design decisions compatible with future implementations for other platforms (e.g., Windows phones). Unfortunately, current platforms do not offer consistent support for 802.11 ad-hoc mode or seamless creation of a base station to enable other devices to connect to them. Bluetooth communication is also inconvenient because of the slow discovery phase and the inability of iPhones to communicate with non-Apple devices (excepting headsets). NFC is not yet widely deployed, and such communication does not scale beyond pairwise communication. Moreover, none of these local communication mechanisms would enable remote users to exchange their contact information. As a consequence, we use Internet-based communication, where all the mobile devices connect to a cloud server via IP.

When mobile devices initially connect to the cloud server, the server does not know which devices belong to the same group. It is a challenging problem for the server to determine the grouping, especially if many concurrent exchanges are ongoing. We employ the following approach (and discuss alternatives in Section 4.4), which does not leak any sensitive information to the untrusted cloud server. The server assigns a unique ID to each mobile device, which it displays to its user. The devices prompt the users to find and enter the lowest ID. The devices then send that ID back to the server, which can thus perform the grouping. Note that the actual grouping is not security-sensitive, as an intruder can only cause denial of service.

Though the current technical limit for our protocols and implementation is much greater than 10 users, it is unclear that there is much value in scaling even this far due to human limitations. We concentrate our presentation on groups of up to 10 users, leaving it as an open question whether there is a need for protocols that scale further. As prior work shows, users can reliably count the number of participants for groups of up to 8 users, but several people start to make errors for larger groups [10]. As the SafeSlinger protocol fails to complete if only a single person miscounts, we set the threshold at 10 users. Asking users to count the number of participants rules out several attacks, as we discuss in Section 5.

As a **brief protocol description**, the mobile devices send a commitment (as described in Section 3.1) to their information to the server, which redistributes it to the other devices (the server is essentially mimicking the communication network and without performing any trustworthy operations). The users then engage in a verification of all exchanged information to ensure that they all possess an identical sets of commitments (to ensure the absence of MitM and GitM attacks) as well as the correct number of commitments based on the user-entered number of group members. For the verification, users perform a comparison of a 3-word phrase that encodes a 24-bit hash value. The words in the 3-word phrase are selected by using PGP Word List (more details are presented in Section 6.3). If the word phrases match, the devices engage in a group-DH protocol (as described in Section 3.2) to derive a group secret key that is finally used to distribute the decryption keys among group members that is used to decrypt contact information.

Two problems must be avoided: (1) users may habituate to click "Match" without performing the word phrase comparison, and (2) an attacker may compute a collision attack on the short 24-bit hash value represented by the 3-word phrase. We address (1) by presenting two decoy word phrases alongside the correct word phrase and asking users to verify which of the three word phrases match a value on other people's devices. The position of the matching phrase is drawn from a different random number generator on each device to prevent an adversary from predicting the location of the matching phrase. This forces the user to perform the comparison, as a random guess by even just a single user will cause the protocol to fail 2/3 of the time.

We address problem (2) by using Short Authentication Strings (SAS) [8,17,18,35,37]. In SAS, all devices first commit to the values that are used in the hash comparison. Once all the commitments are distributed, the devices reveal the decommitments and the short hash comparison can proceed. This approach prevents the collision attack and in Zimmermann's words [37] converts the attack from a "safe attack" into a "daring attack." The collision attack is a safe attack, because the adversary is certain that the attack will succeed as a collision has been found. However, with the commitment, the adversary cannot know ahead of time if the collision indeed will occur, and the attack only succeeds with probability of 2^{-n}, where n denotes the bit length of the authentication string, thus resulting in a "daring attack." In SafeSlinger, $n = 24$.

Another challenge that we address in SafeSlinger is to prevent the server from learning any contact information. We accomplish this by leveraging a group-DH protocol (described in Section 3.2). The group DH protocol establishes a shared secret key among all participants, which is used to encrypt the exchanged information. To prevent MitM attacks, the DH public key is included in the initial commitment and thus validated during the hash comparison. The only information leaked to the server is the IP address used by the devices. In order to provide anonymity, we could leverage other anonymous communication systems in addition to SafeSlinger, such as Tor [33]. The following section describes the intricate details of the SafeSlinger protocol.

4.2 SafeSlinger Exchange Protocol Details

Figure 4 describes the SafeSlinger protocol in detail. For clarity we separate our protocol into its multi-commitment tree setup and two rounds of verification.

Multi-Commitment Generation (S1-S3). In Step 1 (abbreviated S1), the user selects which data to share and enters the total number of protocol participants. In S2, the device computes the values needed for the group DH protocol by randomly generating the ℓ'-bit long DH private key n_i (in the current version, we use $\ell' = 512$). The device also randomly selects nonces to indicate "match" (Nonce match, Nm_i) and "wrong" (Nonce wrong, Nw_i). The device also encrypts the data D_i to share with Nm_i used as a symmetric encryption key (using AES with 256-bit keys): $E_i = \{D_i\}_{Nm_i}$. We use security parameter $\ell = 256$, and we use SHA-3 (256 bits) as hash function H. Figure 3 depicts this multi-value commitment structure for user U_i. Finally, in S3 the device sends the commitment C_i to the server.

Authenticity Verification Round (S4-S15). In the next phase, the server groups the users. First, the server sends a unique ID to the device (S4) which the device displays as it prompts the user to find the lowest ID amongst all devices (S5). In S6, the user enters the lowest ID, which in S7 the device sends to the server.

The server now knows which devices belong to the same group, and distributes ID and commitment pairs (ID_i, C_i) to all group members (S8). Once a device receives all commitments, in S9 it opens up the first level decommitment HN_i, G_i, E_i (Figure 3). If validation of all decommitments is correct (S10), devices compute a hash over all decommitments of C_i, i.e., over the triplets (HN_*, G_*, E_*), sorted by the value of the unique ID_i assigned to each device to ensure that

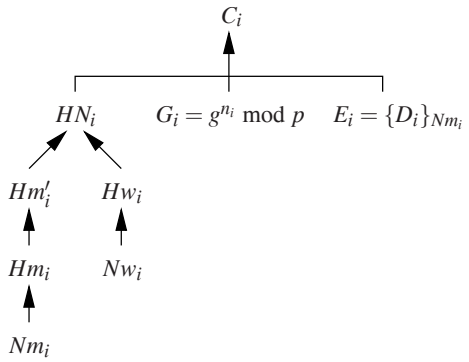


Figure 3: Multi-value commitment structure for user U_i. C_i, HN_i, Hm'_i, Hw_i, Hm_i, Nw_i, Nm_i, are 256-bit values; G_i is 512 bits, and D_i varies in length.

all devices compute the hash over the same triplet ordering. Each device then computes a 3-word phrase that represents 24 bits of the hash (S11), for which on we use the PGP Word List. The device also produces two decoy 3-word phrases, which produces two interesting challenges: (1) if the words in the decoy word phrases match words in the actual word phrase, users may get confused, and (2) if words in different users' decoy word phrases match, users may wrongly select the decoy phrase on their respective devices as a match. To avoid this, we make sure that the decoy phrases do not include any words from the actual word phrase, and we also make sure that all decoy word phrases are mutually exclusive among all devices, which is a challenge to implement. We address this by using the received decommitments as a seed to a PRNG, and having each device draw words from the word phrase for their decoy phrases without replacement. Since the word phrase only contains 512 words in total, this limits the total number of users we can support. More details on the word phrase design are presented in Section 6.4.

S12 and S13 represent the failure and success cases for user verification of 3-word phrases, respectively. If no phrase matches, the user selects "no match", causing their device to send the "wrong" nonce Nw_i (along with Hm'_i to enable verification) to the server (S12). This case is also triggered if the user selects one of the decoy word phrases. This approach cryptographically authenticates the "no match" message based on the commitment C_i and thus prevents injection of an abort message by an adversary. If users correctly selected the matching word phrase (S13), their device reveals the pair of values indicating success (Hm_i, Hw_i), which the server redistributes in S14. Each device can verify that all users selected the correct word phrase (S15). We analyze the success probability of this verification step in Section 5.

Secret Sharing Round (S16-S20). In S16, the devices proceed to construct the group DH tree as described in Section 3.2. The ordering in the tree is determined by the sorted order of the unique IDs ID_*. Since the group DH protocol is intricate, we omit the details for enhanced readability (Section 3.2 preserves more for detail).

Once the secret group key K is established, the devices then proceed to send their final match nonce Nm_i (which also serves as the decryption key for their data D_i) to the group, encrypted under key K (S17). In S18, each device decrypts each received Nm_j from the other devices, verifies its integrity using C_j, and finally uses Nm_j to decrypt E_j to obtain each device's data D_j in S19. In S20, each user can validate the identity information from the decrypted data and save it to his/her address book.

Multi-Commitment Generation
Data Selection & Counting
1. $U_i \xrightarrow{UI} M_i$: D_i (the data to be exchanged)
   $U_i \xrightarrow{UI} M_i$: $\tilde{N}_i$ (number of people in the group)

Commitment, Group DH Key Setup
2. M_i : $Nm_i \xleftarrow{R} \{0,1\}^\ell$ ("match" nonce)
   $Hm_i = H(Nm_i)$, $Hm'_i = H(Hm_i)$
   $Nw_i \xleftarrow{R} \{0,1\}^\ell$, $Hw_i = H(Nw_i)$ ("wrong" nonce)
   $HN_i = H(Hm'_i || Hw_i)$ (multi-value commitment)
   $n_i \xleftarrow{R} \{0,1\}^{\ell'}$, $G_i = g^{n_i} \bmod p$ (group DH key)
   $E_i = \{D_i\}_{Nm_i}$ (encryption of data)
   $C_i = H(HN_i \,||\, G_i \,||\, E_i)$ (commitment)
3. $M_i \rightarrow S$: C_i

Authenticity Verification Round
Server Unique ID Assignment, User Grouping
4. $S \rightarrow M_i$: ID_i (unique ID per user)
5. U_i : find lowest unique ID among users $\rightarrow ID_L$
6. $U_i \xrightarrow{UI} M_i$: ID_L (enter lowest ID)
7. $M_i \rightarrow S$: ID_L

Collection and Distribution of Initial Decommitment
8. $S \rightarrow M_i$: ID_j, C_j $(j \neq i)$
   (other users' ID and commitment)
9. $M_i \rightarrow S$: HN_i, G_i, E_i
   $S \rightarrow M_i$: HN_j, G_j, E_j $(j \neq i)$
   (other users' decommitments)
10. M_i : $C_j \stackrel{?}{=} H(HN_j || G_j || E_j)$ $(j \neq i)$ (verify)

Word Phrase Comparison of Integrity of Commitments
11. M_i : WordPhrase($[H(HN_*, G_*, E_*)]_{24}$) (screen)
   $U_i \xrightarrow{UI} M_i$: Select Matching 3-Word Phrase
12. $M_i \rightarrow S$: **if** "no match" or wrong phrase selected:
   Send Hm'_i, Nw_i, Abort protocol.
13. $M_i \rightarrow S$: **else if** "match" & correct phrase selected:
   Send Hm_i, Hw_i
14. $S \rightarrow M_i$: Hm_j, Hw_j $(j \neq i)$
15. M_i : $HN_j \stackrel{?}{=} H(H(Hm_j) || Hw_j)$ $(j \neq i)$ (verify)
   Abort if any verification failed

Secret Sharing Round
Group DH Key Establishment
16. M_i : Computation of group DH tree
   K = Private key of root node (see Section 3.2)

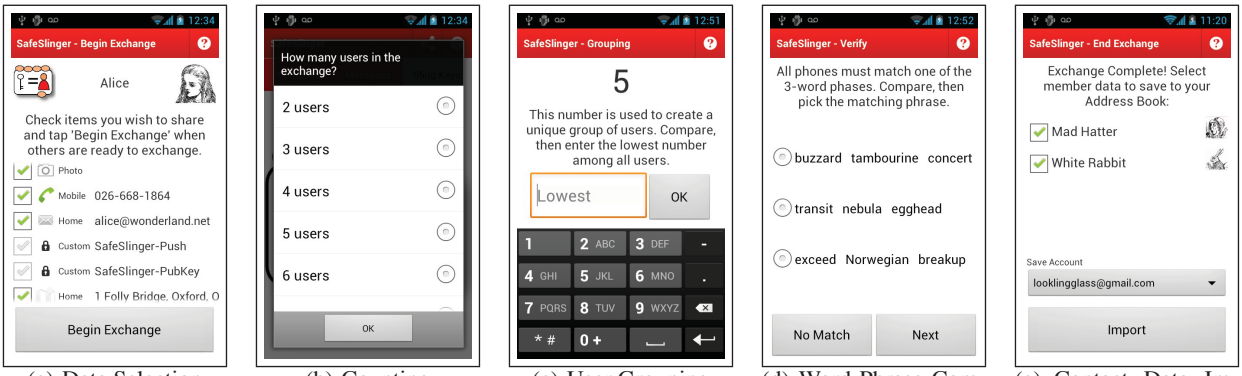
Distribution and Verification of Data Decryption Key
17. $M_i \rightarrow S$: $\{Nm_i\}_K$
   $S \rightarrow M_i$: $\{Nm_j\}_K$ $(j \neq i)$
18. M_i : Decryption of Nm_j $(j \neq i)$
   $Hm_j \stackrel{?}{=} H(Nm_j)$ $(j \neq i)$ (verify)

Decryption of Data and Contact Import
19. M_i : Decryption of E_j with Nm_j $(j \neq i) \rightarrow D_j$
20. $U_i \xrightarrow{UI} M_i$: Save user data D_j $(j \neq i)$

Figure 4: Steps for user U_i $(i \in 1 \ldots N)$ to exchange data D_i with the other $N-1$ users via their mobile devices. $U_i \xrightarrow{UI} M_i$ indicates input by user U_i into mobile device M_i. $M_i \rightarrow S$ represents wireless communication from mobile device M_i to server S. $\{X\}_K$ represents encryption of X with symmetric key K.

4.3 User Experience

Usability aspects drove many of our design considerations to make the experience convenient, efficient, and comfortable to use. SafeSlinger performs the bulk of the protocol operations without user involvement. The following list shows the required user actions in a SafeSlinger exchange, Figure 5 provides the corresponding screenshots.



(a) Data Selection (b) Counting (c) User Grouping (d) Word Phrase Comparison (e) Contact Data Import

Figure 5: Secure contact information exchange sequence.

1. **Data Selection**: The user selects the items of his/her contact information to share, and presses the "Begin Exchange" button to start the protocol (Figure 5(a)).
2. **Counting**: The user counts the number of group members in the exchange, selects the group size on the dialog, and presses "OK" to continue (Figure 5(b)).
3. **User Grouping**: The user sees his/her unique ID (assigned by the server) and is prompted to enter the lowest ID of any user participating in the exchange (Figure 5(c)).
4. **Word Phrase Comparison**: The screen displays three phrases; each containing three words. The user compares the phrases with other members, selects the matching 3-word phrase, and presses "Next" if all members have the same phrase. Otherwise, (s)he selects "No Match" to abort the protocol (Figure 5(d)).
5. **Contact Data Import**: The user is prompted to select which contact entries on the list to import, and into which of his/her contact accounts (Figure 5(e)).

4.4 Discussion

We now discuss the rationale behind the design decisions we made. With respect to the server's grouping approach, we considered numerous alternatives. The Bump [7] application groups its members by having two users "bump" their phones together, and by measuring location, time, and acceleration the server can pair up the two phones. Although this approach is fun for the users, we did not use it for numerous reasons: (1) Bump reveals the user location to the server, which is an invasion of privacy; (2) The approach is not secure as a malicious bystander can simultaneously simulate the bump and often be paired with one of the two devices and steal the user's contact information [32]; (3) Bump does not scale to multiple users; (4) Bump cannot be performed among remote users; (5) Acquiring the device's location can be unreliable and often has a delay of 10 seconds or more; (6) Bump is often unreliable and fails to pair devices.

Another alternative for grouping we considered is to use ambient noise, but this may reveal privacy-sensitive sound to the server, and may also be unreliable in many circumstances. We finally settled on the unique *ID* assignment by the server, and having users find and enter the lowest *ID*. This approach is fast and reliable. Based on the *ID*s, all the server needs is to end up with a connected graph, where each device represents a node and an edge is formed by having one device send the *ID* of another device to the server. We considered the approach of having users simply enter an *ID* of any other intended user, but this may be confusing when multiple users are present, and it can also lead to a non-connected graph when there are more than three users. By having users enter the lowest *ID*, the resulting graph forms a star topology, which always forms a connected component with all participants.

We would like to emphasize that the group of members who perform a SafeSlinger exchange are not required to later communicate within the same group. SafeSlinger is only used for acquiring other users' information in a secure fashion. Since the contact information includes a public key, only that public key is used to subsequently establish secure communication. In particular, all cryptographic values created during the exchange (Figure 4) are erased after the exchange – the sole purpose of their brief existence is to protect a single exchange. Subsequent secure group communication can be easily established by using the exchanged public keys in conjunction with a group DH protocol, regardless of the key exchange session during which the public key was acquired. Thus, there is no mandatory relationship between the group that was used to exchange the information and subsequent membership selection for secure group communication.

5. SECURITY ANALYSIS

We will use the list of attacks and challenges from Section 2.5 to guide this security analysis, along with additional attacks that are specific to the operations of the SafeSlinger protocol.

We first consider attacks by malicious outsiders (who are not legitimate group members). Such adversaries can contact the server, ask for a unique ID, and attempt to join arbitrary groups by sending the server a plausible group ID to join. Since users specify the group size, legitimate clients will detect this attack, as the server will send too many commitments to the participants.

A more sophisticated version of that attack is where a local adversary jams communication of one of the local devices, and attempt to join the group in place of the jammed user. A similar attack can be performed by a malicious server, which can split the group up into different subsets of users, and fake another set of users (GitM attacks) to ensure that each device encounters the correct number of devices, even though some are virtual identities created by the malicious server. Preventing these attacks requires some amount of user diligence: the users need to ensure that they perform the hash comparison with *all other* users. If one member is singled out by a jammer or malicious server, then that user needs to inform at least one other user to abort the protocol. As long as at least one unsuppressed user is diligent, then these attacks will be prevented. If these attacks are prevented, neither the local jammer nor the malicious server learns any information about the users, as they cannot participate in the group DH protocol to discover the established group key K. The only way to obtain K is to participate

as a user and inject a commitment, which would be detected by the devices, as the number of members is larger than the user entered.

A malicious legitimate participant of the group could launch several attacks. First, attempts to infiltrate additional virtual members into the group, for example through a Sybil attack [12], will fail because the number of virtual members would be larger than the count of physical members which users enter at the beginning of the protocol. A Group-in-the-Middle (GitM) attack as described in Section 2.5 is prevented if users diligently perform the hash comparison step, as members who end up in different groups will have different hashes to compare with high probability. The adversary could also send malicious contact information for himself, e.g., it may attempt to impersonate another person who is intended to be in the group. SafeSlinger defends against this attack by enabling users to verify at the end of the protocol which of their contact entries they actually import into their address book, and if a suspicious entry exists the user can attribute that to the adversary. If the adversary impersonates a user who is present, that user can detect that someone else injected a duplicate entry for herself and inform the others.

The multi-commitment with several stages of decommitment (Figure 3), ensures that no information is revealed unless all members M_i reveal their "match" nonce's hash HNm_i, as otherwise devices will not reveal the decryption key Nm_i. Hence, if any group member detects an anomaly before the hash comparison, all benign devices will abort the protocol. The multi-commitment also prevents the simple collision attack on the low-entropy hash (24 bits) that is used for the comparison, by computing the hash over the ordered triplets (HN_*, G_*, E_*). Since the commitment $C_i = H(HN_i \mid\mid G_i \mid\mid E_i)$ locks in the value of the triplet, an adversary cannot change its triplet or predict any other triplet before the hash is pre-determined through all users' choices.

Due to the peculiarities of how the word lists are selected and to account for potentially complacent user behavior during comparison, we next analyze the success probability of a MitM attack assuming two benign users are engaging in a SafeSlinger exchange. We will consider different user behavior with respect to the 3-word phrase comparison, ranging from diligent to lazy.

If either one of the users diligently performs the word phrase comparison, the success probability is 2^{-24}, offering sufficient deterrence against attacks since a human needs to be involved in each protocol run.

We expect that users will not be completely lazy (i.e., not perform the word phrase comparison and simply select one of the word phrases at random and click "Match") because the protocol would fail with probability 2/3 even in the absence of any adversarial action. We thus expect that some users will be "partially diligent" (or "semi-lazy"), and thus expend minimal effort for comparison to quickly complete the exchange. Assume that two "matching" word phrases $A = \{A_1, A_2, A_3\}$ and $B = \{B_1, B_2, B_3\}$ on the devices of users A and B, respectively. In the absence of a MitM attack $A = B$, otherwise $A \neq B$ with high probability.

We now compute the success probability of a MitM attack if both users are partially diligent, where we assume that the users compare by finding at least one matching word without confirming its position. We do assume that the partially diligent users would abort the exchange by selecting "no match", or select the wrong word phrase in case the match is in a decoy word phrase, which also aborts the exchange. We thus need that $A \cap B \neq \emptyset$. The probability $P[A \cap B \neq \emptyset] = 1 - P[A \cap B = \emptyset]$. All possible combinations of A equals to $\binom{256}{2} \cdot \binom{256}{1}$ because two words are chosen from the "even" list and the other from the "odd" list. The number of combinations that will not result in any matching words is $\binom{254}{2} \cdot \binom{255}{1}$. Thus, $P[A \cap B \neq \emptyset] = 1 - \frac{\binom{254}{2} \cdot \binom{255}{1}}{\binom{256}{2} \cdot \binom{256}{1}} \cong 1.94\%$. This MitM success probability is optimistic, since matching words might also present in decoy word phrases, which users can "accidentally" choose, resulting in protocol failure (the attack would not succeed).

Since some partially diligent users may only compare the first words of the word phrases, we compute the probability that the first word matches, thus, $A_1 = B_1$. We obtain $P[A_1 = B_1] = \frac{1}{256} \cong 0.391\%$.

The success probability for other cases is even lower, thus, we only present these cases here. Even for the pessimistic case where both users are partially diligent, the success probability is upper bounded by 1.94%, which we believe suffices to frustrate adversaries from using this attack in practice.

6. IMPLEMENTATION

The current SafeSlinger implementation comprises three components: (1) a server application running on Google App Engine and written in Python, (2) a client application for Android (v2.2 or higher) written in Java, and (3) a client application for Apple iOS (v5.1.1 or higher) written in Objective-C. SafeSlinger works well over both cellular and Wi-Fi networks, and has been tested on several smartphone devices: Motorola Droid 855, Google Nexus One/S, Samsung Galaxy Nexus, Samsung Galaxy/S2/S3/S4, Apple iPod Touch, and Apple iPhone 3GS/4/4S/5.

6.1 Key Management using Address Book

We rely on the mobile operating systems' address book facility to manage users' contact data and public keys. It is convenient to store users' public key data in a recognizable field in the smartphone's address book, so that any existing synchronization service will seamlessly maintain backups. We have implemented this functionality by adding the name and value (base-64 encoded information) of a new instant messaging (IM) provider to the address book.

At the beginning of an exchange (Figure 5(a)), each user must identify which contact entry they wish to use as their identity, or they may create a new entry instead. During the exchange, a user's contact information is formatted in the vCard 3.0 standard format[3] to be recognizable across multiple platforms. For custom data fields in SafeSlinger (e.g., public keys), we use the IMPP field type (see IETF vCard Extensions for Instant Messaging[4] proposal) with a naming pattern so that fields do not require special handling relative to other contact fields. Our current implementation uses the labels "SafeSlinger-PubKey" for public keys and "SafeSlinger-Push" for push notification tokens.

This design offers flexibility and compatibility for mobile app developers. Other developers can include the exchange of key material (or other data made available by SafeSlinger) without the need to include dedicated support in their applications. Multiple apps can access exchanged public keys and push tokens directly from reading the user's address book without adding additional storage overhead.

6.2 Secrecy During Data Exchange

All exchanged data is encrypted using AES in CBC mode with PKCS#7 padding. The encryption key K_D and initialization vector IV_D are derived from the 256-bit "match" nonce Nm_i: K_D = HMAC-SHA-3$_{Nm_i}(1)$, IV_D = HMAC-SHA-3$_{Nm_i}(2)$. Contact data integrity is achieved through verification of the commitment C_i, hence, no additional Message Authentication Code (MAC) is needed.

[3] http://tools.ietf.org/html/rfc2426
[4] http://tools.ietf.org/html/rfc4770

We use SHA-3 (Keccak) [4] since previous hash algorithms, such as MD5 and SHA-1, have been successfully attacked in either theoretical or practical aspects.

Recall from Section 4 that the "match" nonce Nm_i is distributed in encrypted form using K (the shared secret resulting from the group DH key establishment). We use AES-CBC with PKCS#7 padding, where the AES key is derived from the group DH key K as follows: $K_{CBC} = \text{HMAC-SHA-3}_K(1)$. Since we can validate Nm_i based on the commitment Hm_i, no additional MAC value is needed to ensure integrity.

6.3 Word Phrase Verification

Once all data has been exchanged, it must be verified to ensure the integrity of the exchange. Each device computes a hash of the ordered set of all data exchanged in the protocol. This hash is truncated and represented to the user as a 3-word phrase, with words taken from the PGP Word List. We design this phase to discourage careless comparison.

In SafeSlinger, the word phrase is constructed from the first 24 bits of the 256-bit SHA-3 hash, as indicated in Step 11 of Figure 4. We use the standard PGP approach for converting a 24-bit value into 3 words. PGP uses two word lists – an "even" and "odd" list – with 256 words each. Based on the standard PGP approach, the first 8 bits select a word in the "even" list, the second 8 bits select a word in the "odd" list, and the final 8 bits select another word from the "even" list.

We discourage careless comparison by displaying two unique decoy phrases in addition to the common phrase. In this way, users are forced to compare phrases with at least one other user and actively choose which phrase matches among them. However, to prevent attacks, all users need to validate that they all have the same word phrase. The word phrase provided in this exchange enables out-of-band verification for users in close proximity where they may view each other's screens, or via telephone or teleconference (or any other communication medium where humans can authenticate one another, e.g., by recognizing a familiar voice), where the users can read their word phrases aloud. In some social contexts, it may be considered impolite to look at or take a picture of another user's phone, and thus this protocol also enables screen privacy and comparison of phrases through comfortable conversation.

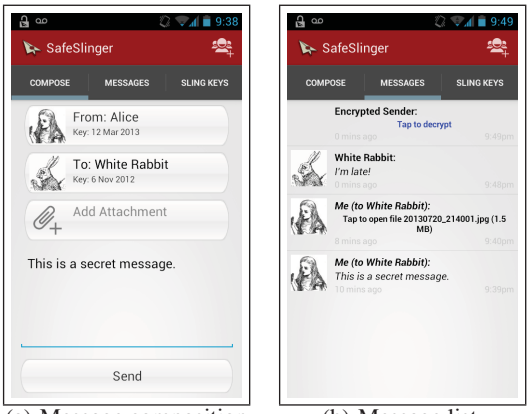
6.4 Word Phrase Collision Avoidance

If a word in our decoy word phrases is the same as in the actual word phrase, users may get confused and select the decoy word phrase as the match. Moreover, the words in a decoy phrase may match the words in a decoy phrase on another device, causing the user to select the decoy phrase which results in an error detected by the local device.

We thus want to select decoy phrases to prevent careless users from choosing the wrong phrase if the actual hash phrase and either of the decoy phrases contain the same word in the same position. Hence, we choose our decoy phrases deterministically such that each decoy word will be unique across all decoy phrases displayed in the group. After computing the actual word phrase, we mark those words as used. Each device will then compute the decoy phrases for all devices, such that no decoy phrase has any matches with any other decoy phrase nor with the actual word phrase.

7. MOBILE APPLICATIONS

We have leveraged our key exchange protocol to create a secure messaging application that supports both text and file exchange across different platforms. We now discuss these applications, and describe an application for providing secure introductions.



(a) Message composition (b) Message list

Figure 6: Secure Messaging Screenshots.

7.1 Secure Rich Messaging Implementation

We have implemented secure rich messaging for Android and iOS (Figure 6). Secure information and shared public keys via SafeSlinger are used to encrypt and authenticate text messages and file data. When SafeSlinger is first installed, it generates 2 RSA 2048-bit key-pairs and requests a push token to serve as that device's identity for the Push Notification service. During a SafeSlinger exchange, the push tokens of all group members are exchanged and imported to the address book alongside the public keys.

Message Secrecy and Authenticity. For data (simple text or files) encryption and authentication, we choose the PKCS#7 encryption format.[5] Our implementation uses the open cryptographic library PolarSSL[6] for both smartphone platforms. One alteration was to use SHA-3 as the hash algorithm for our RSA signature generation instead of SHA-1. Files to be included in a message are formatted as a separate PKCS#7 message, and downloaded independently by the receiving device.

Push Message Notification. To avoid text-messaging charges and conserve battery energy, we make use of smartphone operating systems' support for *push notifications* to deliver message data. Rather than requiring each application developer to implement a background task to keep a separate connection to their server to poll for updates, push notifications are an OS-provided unified update service in which the OS maintains one connection to its own service. Then, each developer can register their application with the OS' service, and the OS only needs to maintain one connection to check for updates across several applications.

We use the native push notification services offered by each smartphone's operating system. Any push notification service may change the push token of the device over time. This presents an issue similar to circumstances where a user may want to generate a new set of key-pairs and wishes to revoke older keys. In either case, we are currently testing an implementation to securely share the updated push tokens and public keys with past exchange participants to improve the user experience.

7.2 Secure Introduction

We leverage the secure rich messaging mechanism to enable *secure introductions*, where a common friend of two users sends those users' respective contact data (including public keys and push tokens) to each other. More concretely, consider Alice with two friends: Bob and Carol. Alice has performed direct SafeSlinger

[5]http://tools.ietf.org/html/rfc5652
[6]https://polarssl.org

exchanges with both Bob and Carol. Alice has thus received authentic SafeSlinger public keys for both Bob and Carol. Likewise, both Bob and Carol have Alice's authentic SafeSlinger public key, but do not otherwise share information.

In a secure introduction, Alice's device first encodes Bob's contact information (which includes Bob's SafeSlinger public key and push token) as a custom vCard and uses the secure message format to provide secrecy and authenticity. Alice's device then encodes Carol's information in the same way. Alice's device then sends (via SafeSlinger) the message containing Bob's information to Carol, and the message containing Carol's information to Bob. Hence, Bob and Carol can each validate that the received information indeed originates from Alice, whom they each trust not to send false information. Now that Bob and Carol have each other's public keys and push tokens, they can use SafeSlinger to securely communicate with each other directly. Bob and Carol have thus successfully built a trust relationship through secure introduction.

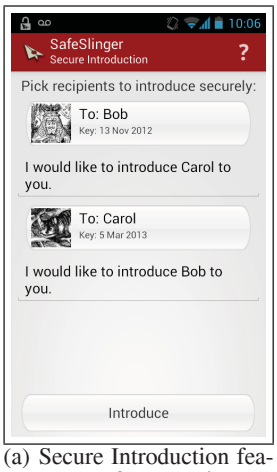
The above example with Alice, Bob, and Carol illustrates one-hop transitive trust: Alice is the "hop" between Bob and Carol. Presently, we only support one-hop secure introductions, and we enforce that both links used in a secure introduction were performed by direct SafeSlinger exchange. Note that these issues represent policy concerns; there is no obstacle to constructing a messaging protocol that supports an arbitrary number of transitive links.

In the SafeSlinger secure introduction implementation, each user picks any two contacts with whom they have performed a direct SafeSlinger exchange before (Figure 7(a)). When an invitee receives an invitation, his/her device verifies the message, i.e., to check that it comes from a user with whom they have performed a SafeSlinger exchange before. Finally, a dialog presents the chain of SafeSlinger exchanges to the introduced user (Figure 7(b)), and prompts for final confirmation before importing the contact.

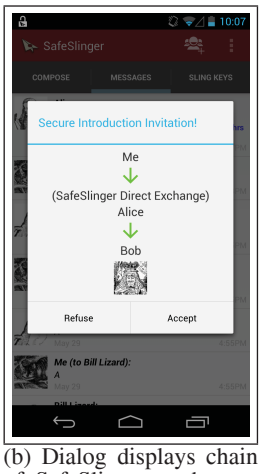
8. EVALUATION

To evaluate the usability of SafeSlinger for daily usage and the willingness to use SafeSlinger for added security and privacy; we conducted a user study with 24 participants. We selected Bump as a baseline application since it is the most popular contact information exchange application available for both Android and iOS.

8.1 Procedure



(a) Secure Introduction feature panel for selecting recipients.



(b) Dialog displays chain of SafeSlinger exchanges from recipient to invitee.

Figure 7: Secure Introduction Screenshots.

We ran three instances of the same study and for each instance, we recruited 8 participants to measure the effectiveness of contact exchanges with varying group sizes. We provided a mobile device to each participant to minimize the risk of exposing the participant's privacy-sensitive data on his/her own phone. For each instance, we used 3 iPhones and 5 iPod Touch devices. We pre-installed Bump and SafeSlinger on all devices and configured them to use wireless access points for communication. For added privacy protection, we assigned a fictitious identity to each participant.

We used a within-subjects design and asked study participants to run Bump and SafeSlinger to exchange contact information. To minimize biases and learning effects, we alternated the order of the applications; i.e., in one instance, a set of 8 participants performed Bump before SafeSlinger, and in the other instance, a different set of participants performed SafeSlinger before Bump.

To evaluate the usability for group key exchanges, we asked participants to form small groups of 2 members and run the application to collect the other group member's contact information. We asked participants to repeat with a different partner once more. Then, we asked participants to form groups of 4 members, and run the application to collect all other group members' contact information. We asked them to repeat with different group members once more. Finally, we asked participants to form a large group of 8 members and run the protocol twice.

When participants finished running both Bump and SafeSlinger, we asked them to complete a brief survey. After the study, we ran a raffle to select two participants and gave iPod Nanos as prizes.

Demographics. We recruited participants from a university mailing list, and we selected 24 participants who have been using smartphones for at least 2 years. The participants' age range was 21–39 (μ = 26.9, σ = 5.3), and 17 were male and 7 were female. Among 24 participants, 18 were students and 6 were professionals. In terms of ethnicity, 16 were Asian, 6 were Caucasian, and 2 were Hispanic. For their current phone model, 11 participants were using the iPhone and 13 were using an Android-based smartphone.

8.2 Results and Observations

We measured the total amount of time and bandwidth that participants consumed to complete the contact information exchanges for varying group sizes.

Figure 8 shows the timing analysis. According to this figure, participants took less time (about 50%) to exchange contact information using Bump than using SafeSlinger for the small group size (of 2). This indicates that participants found Bump easier and faster to operate for the small group size. However, as the group size increased, the execution time increased quadratically for Bump whereas SafeSlinger had linear increase. For example, four users spent more than 1 minute and eight users spent almost 4 minutes to exchange information using Bump since only two people can bump at once. Also, numerous Bump exchanges kept on failing. For SafeSlinger, on the other hand, four users spent only 27 seconds to exchange their contact information, and eight users spent only 39 seconds. These results demonstrate the efficiency and usability of SafeSlinger for exchanging contact information compared to Bump.

Figure 8 also presents the standard deviation of the execution time in Bump and SafeSlinger, and SafeSlinger resulted in a small standard deviation since participants performed SafeSlinger with very few errors; only 1 error occurred for the instance with a large group size during the entire study. For Bump, however, participants needed to repeat running the protocol due to various types of errors, such as connection failure or fetching wrong contacts. In contrast

Table 1: Means and paired-samples T-test results from participants' feedback. The higher mean that is statistically significant from the other is highlighted in bold. For all results, $N = 24$.

	Easy to use min:1 max:5	Annoyance min:1 max:5	Security of app. min:1 max:5	Likely to use min:1 max:5
Bump	3.3 ± 1.4	$\mathbf{3.8 \pm 1.1}$	2.3 ± 1.1	2.4 ± 1.1
SafeSlinger	$\mathbf{4.2 \pm 1.1}$	2.1 ± 1.0	$\mathbf{4.3 \pm .7}$	$\mathbf{3.6 \pm 1.1}$
T-test	$t(23) = 2.6,\ p = .015$	$t(23) = 6.1,\ p < .0001$	$t(23) = -7.6,\ p < 0.0001$	$t(23) = 5.1,\ p = .139$

with Bump, SafeSlinger led to minimal deviation even though it required more user interactions to complete the protocol.

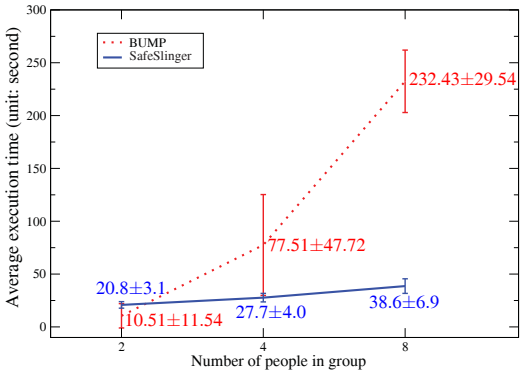


Figure 8: Average execution time for contact exchange in various group sizes using Bump and SafeSlinger.

Participants experienced security and privacy vulnerabilities of Bump during our study; since we conducted each instance of the study in a single room, participants experienced receiving wrong contact(s)' information if they happened to bump around the same time. Consequently, participants became noticeably frustrated during the Bump experiments. On the other hand, participants did not experience any security or privacy vulnerabilities using SafeSlinger.

Bandwidth is a limited resource given that mobile users have limited data plans. Hence, we measured the amount of data usage on each mobile device during the user study, and the results show that each device uses at most 11 KBytes for an exchange between 2 people and 65 KBytes for an exchange among 8 people. For comparison purposes, we also measured data usage for different apps on Android and iOS devices, and we found that: (1) playing a YouTube video requires over 10 MBytes per minute, (2) a voice-only Skype call requires around 0.9 MBytes per minute, and (3) reading an article (including text and images) on the Yahoo news website consumes about 350 KBytes per minute on average. Thus, SafeSlinger has modest bandwidth requirements for an exchange.

8.3 Feedback from Participants

At the end of the study, we asked 5-point Likert scale questions to get further feedback. For all the statistical analyses in this section, we used paired-samples T-tests and Table 1 summarizes the means and the T-test results.

We asked how easy participants found each application to use (1: very challenging to use, 5: very easy), and participants mentioned that SafeSlinger was *easier* to use than Bump with statistical significance. We also asked how annoying participants found each application to use (1: very pleasant, 5: very annoying), and participants responded that Bump was *more disturbing* to use than SafeSlinger with statistical significance. For the security measurement, we asked how secure participants felt that only the intended recipients received their contact information using Bump and SafeSlinger, and participants answered that they felt *more secure* while using SafeSlinger than while using Bump with statistical significance. In terms of likeliness to use in near future (1: very unlikely to use, 5: very likely), participants provided the answers that they are *more likely* to use SafeSlinger than Bump with statistical significance. Based on the participants' feedback, we can conclude that SafeSlinger is usable for daily use with enhanced security and privacy as compared to Bump.

9. RELATED WORK

Closely related research by Asokan-Ginzboorg [2], Abdalla et al. [1], Valkonen et al. [34], and Laur and Pasini [19] provides techniques for secure local establishment of a shared secret key without relying on a PKI or any prior trusted information. Unfortunately, a secret shared among nodes cannot be used to provide integrity and authenticity for exchanged messages, because any malicious group member with the key could have created that message. Therefore, a shared secret is insufficient for secure exchange of authentic information. In particular, GitM attacks are possible, exploiting the absence of message authenticity.

PGP key-signing parties [6] enable users to obtain each other's public key, but they are cumbersome for participants. SafeSlinger can be viewed as a more modern and usable approach using smartphones to securely exchange public keys.

Silent Circle [27] is a recent end-to-end secure communication system for both iOS and Android platforms, providing data encryption for different services, e.g., email, VoIP, and text messaging. Silent Circle enables key exchange between users based on DH, and adopts Short Authentication Strings (SAS) to defend against MitM attacks. Since their implementation is proprietary, one cannot verify the security of their system.

The OTR protocol [5] is a cryptographic protocol to secure communication for instant messaging services. Similar to SafeSlinger, OTR provides authentication and secrecy for messages, but in addition also provides perfect forward secrecy and repudiability. OTR requires that communicating parties initially know each other's public keys, which could be provided by SafeSlinger. Similarly, SafeSlinger could use OTR's approach to provide perfect forward secrecy and repudiability.

The most closely related works provide secure group-based exchange of contact information: GAnGS [10], SPATE [21], and Nithyanand et al. [26].

Nithyanand et al. recently studied the usability of secure group association protocols [26]. Their results with user studies conclude that the ideal group credential exchange protocol does not use a leader (i.e., is peer-based), requires users to count and input the number of participants, and requires users to verify Short Authentication Strings (SAS). Hence, their study confirms the approaches we have selected for SafeSlinger.

The GAnGS protocol [10] was designed to scale trust establishment to large groups of users. Unfortunately, GAnGS requires users to perform many operations and is thus cumbersome to use.

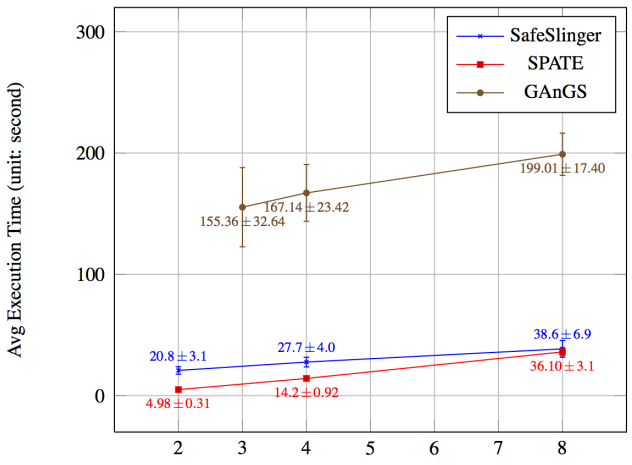


Figure 9: Average execution time for contact exchange in various group sizes using GAnGS, SPATE, and SafeSlinger.

SPATE [21] was designed for contact exchange in smaller groups (8 or fewer people). SafeSlinger offers numerous improvements over these prior systems: (1) GAnGS and SPATE require the acquisition of a 2D barcode displayed by another phone, which can require a significant amount of time, especially in some lighting conditions such as bright sunshine; (2) GAnGS and SPATE use a visual hash function for users to perform comparison of the hash values – such a visual hash cannot support remote execution and enables users to simply click "match" without actually performing the comparison; (3) GAnGS and SPATE require leader selection among a group of participants, which may be burdensome for people during the exchange; (4) GAnGS and SPATE enable bystanders to learn everyone's contact information, disclosing potentially sensitive private information. SafeSlinger offers significant improvements by addressing all these issues. In particular, the privacy protection achieved with the group DH protocol is critical for people who want to protect their privacy, even from a potentially malicious server.

In terms of efficiency of contact exchange (i.e., the average execution time to perform an exchange among varying group sizes), GAnGS requires significantly more time for a group exchange than SPATE or SafeSlinger: GAnGS takes over two minutes even for just 3 users as shown in Figure 9. (Note that the minimum group size supported in GAnGS is 3 instead of 2.) SPATE is markedly close to SafeSlinger's speed. For a smaller group of 2 or 4 users, SPATE is the most efficient protocol among all existing systems. SafeSlinger requires more time for the exchange because SafeSlinger's design involves more participant effort to eliminate SPATE's disadvantages as mentioned above. Figure 9 also shows the standard deviation of the execution time in all schemes. Compared to GAnGS, SafeSlinger and SPATE resulted in a small standard deviation since the SPATE and SafeSlinger protocols are more efficient and require fewer user interactions to complete.

Many researchers have studied device pairing or key setup between two devices [3, 7, 9, 13, 20, 22, 29]. These systems, however, do not generalize to more than two parties, as they would encounter the issues we describe in Section 2.5.

10. CONCLUSION

To realize the vision of secure online communication, we need to overcome several human challenges: some users are ambivalent about security or privacy, most users lack security expertise, and many users prefer convenience over security and may not want to expend much effort for security.

To overcome these challenges, we designed SafeSlinger as an easy-to-use application that offers many benefits to drive usage. Per Metcalfe's law, the utility of a system grows with the square of the number of users. Our goal is thus to provide immediate utility to enable epidemic growth.

We achieve immediate utility through the robust exchange of contact list information between different smartphone platforms, which does not require any location information or leakage of private information outside the participating phones. SafeSlinger also provides secure messaging and file transfer that is immediately usable. SafeSlinger supports secure local and remote exchange of contact list information, setting up a secure channel between users.

We have released the SafeSlinger application both for Android and iOS devices with the intention to provide a free and easy-to-use system that enables secure communication. We publish the full documentation and source-code to enable independent verification of all security operations. With these steps, we anticipate that SafeSlinger will help to bring secure communication to the masses, so that people can enjoy secret communication and validate the origin of messages with confidence.

11. ACKNOWLEDGMENTS

Numerous people helped with this project. We especially would like to thank Jason Lee, Manish Burman, and Gurtej Singh Chandok for their help with early versions of the application. We would also like to thank Virgil Gligor for his helpful suggestions. We thank Sascha Fahl, Henning Perl, and Matthew Smith for their help with analyzing the SSL certificate validation code. We also acknowledge the following students for their effort on analyzing the security of the SafeSlinger code in their class project: Ryan Caney, Yuan Tian, Sagar Medikeri, Merly Knox, Wenyao Li, Zhuangyou Xu, Shuyang Wang, Kaili Li, Pranjal Jumde, Bargav Ravikumar, Nagarathnam Muthusamy, and Sujay Jain.

We thank Moxie Marlinspike for interesting discussions and for suggesting the "sling" metaphor. We also thank Morley Mao and the anonymous reviewers for their comments and suggestions on how to improve the paper.

This research was supported by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389, W911NF-09-1-0273, and MURI W 911 NF 0710287 from the Army Research Office; and by support from NSF under awards CNS-1050224, CCF-0424422, and CNS-1040801. 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 expressed or implied, of ARO, CMU, NSF or the U.S. Government or any of its agencies.

12. REFERENCES

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

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

[3] 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.

[4] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. The keccak sponge function family. http://keccak.noekeon.org.

[5] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *Proceedings of ACM Workshop on Privacy in the Electronic Society*, pages 77–84. ACM, 2004.

[6] V. A. Brennen. The Keysigning Party HOWTO. http://rhonda.deb.at/projects/gpg-party/gpg-party.en.htmll, Jan. 2008.

[7] Bump technologies. http://bu.mp/.

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

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

[10] 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 ACM MobiCom*, Sept. 2008.

[11] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, Nov. 1976.

[12] J. R. Douceur. The Sybil attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS)*, Mar. 2002.

[13] 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.

[14] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, July 2004.

[15] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information Systems Security*, 7(1):60–96, May 2004.

[16] C. Kuo, A. Studer, and A. Perrig. Mind your manners: Socially appropriate wireless key establishment for groups. *Proceedings of First ACM WiSec*, Mar. 2008.

[17] S. Laur, N. Asokan, and K. Nyberg. Efficient mutual data authentication using manually authenticated strings. Report 2005/424, Cryptology ePrint Archive, Nov. 2005.

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

[19] S. Laur and S. Pasini. Sas-based group authentication and key agreement protocols. In *Proceedings of Public Key Cryptography*, 2008.

[20] 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.

[21] 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 ACM/Usenix MobiSys*, June 2009.

[22] 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.

[23] M. Marlinspike. Breaking SSL with null characters. In *Presented at Black Hat*, 2009.

[24] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology (Crypto)*, pages 369–378, 1987.

[25] G. Mezzour, A. Studer, M. Farb, J. Lee, J. McCune, H.-C. Hsiao, and A. Perrig. Ho-Po Key: Leveraging physical constraints on human motion to authentically exchange information in a group. Technical Report CMU-CyLab-11-004, CyLab, Carnegie Mellon University, Dec. 2010.

[26] R. Nithyanand, N. Saxena, G. Tsudik, and E. Uzun. Groupthink: Usability of secure group association for wireless devices. In *Proceedings of Ubicomp*, Sept. 2010.

[27] Silent Circle Co. Silent circle – global encrypted communications service. https://silentcircle.com, Sep. 2008.

[28] C. Soghoian and S. Stamm. Certified lies: Detecting and defeating government interception attacks against SSL. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, July 2010.

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

[30] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A Secure Audio Teleconference System. In *Advances in Cryptology (Crypto)*, 1988.

[31] M. Steiner, G. Tsudik, and M. Waidner. Key Agreement in Dynamic Peer Groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, Aug. 2000.

[32] A. Studer, T. Passaro, and L. Bauer. Don't bump, shake on it: The exploitation of a popular accelerometer-based smart phone exchange and its secure replacement. In *Proceedings of ACSAC*, 2011.

[33] Tor Project Team. Tor: Anonymity Online. https://www.torproject.org, Dec. 2007.

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

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

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

[37] P. R. Zimmermann. Pgpfone, pretty good privacy phone, owner's manual, version 1.0 beta 7. ftp://ftp.pgpi.org/pub/pgp/pgpfone/manual/pgpfone10b7.pdf, July 1996.