



UNIVERSITÀ DEL PIEMONTE ORIENTALE

Dipartimento di Scienze e Innovazione Tecnologica
**Corso di Laurea Magistrale in Intelligenza Artificiale e
InnovazioneDigitale**

Tesi di Laurea

**Analisi forense dell'applicazione di
messaggistica Briar su dispositivi Android**

Relatori:

Prof. Marco Guazzone

Co-relatori:

Prof. Cosimo Anglano

Prof. Massimo Canonico

Candidato:

Agnese Cabella

A handwritten signature in black ink that reads 'Agnese Cabella'.

Anno Accademico 2023/2024

Abstract

Briar ha recentemente avuto un enorme diffusione che ha portato a un crescente interesse nell'analizzarla.

Proprio da questo grande interesse che inizia il lavoro di questa tesi, il quale si pone come obiettivo quello di comprendere il funzionamento di questa applicazione e gli artefatti generati durante il suo utilizzo.

Si è deciso di analizzare inizialmente il funzionamento dell'applicazione lato utente e l'architettura interna dell'applicazione in modo da presentare i concetti fondamentali e quindi comprendere al meglio i successivi risultati dell'analisi.

Si è proseguito eseguendo gli esperimenti di ogni funzionalità attraverso l'utilizzo del software AnForA. Tali esperimenti hanno prodotto dei dump il cui contenuto è stato puntualmente analizzato per dedurre la correlazione tra i dati presenti nei vari dump e le azioni eseguite dall'utente durante gli esperimenti.

In questo studio non solo ci si è concentrati sulla definizione della correlazione artefatti-azioni dell'utente ma anche in tutte quelle attività accessorie propedeutiche per accedere agli artefatti.

I risultati di questo lavoro hanno lo scopo di aiutare gli analisti forensi nell'analisi degli artefatti generati da Briar e presenti nei dispositivi oggetto d'analisi. L'analista basandosi su questo lavoro può procedere più celermente nella comprensione degli artefatti presenti e nel formulare le proprie ipotesi su quali siano state le azioni eseguite dall'utente, velocizzando in sostanza le proprie attività.

Indice

1 - introduzione	5
2 - Funzionalità di Briar	7
2.1 - Creazione dell'account utente	7
2.2 - Gestione dei contatti.....	8
2.3 - Gestione messaggi in chat private	16
2.4 - Gestione gruppi privati.....	21
2.5 - Gestione del forum	24
2.6 - Gestione del blog	28
2.7 - Gestione del feed RSS	31
2.8 - Gestione delle impostazioni.....	36
3 - Organizzazione dei dati generati da Briar	42
3.1 - Architettura	43
3.2 - File db.key	75
3.3 - File db.mv.db.....	79
3.4 - File per modalità removable drive	91
3.5 - File XML.....	99
4 - Analisi forense di Briar	102
Problematiche riscontrate	103
Gestione dei Messaggi e Metadati nel Sistema Briar.....	106
4.1 - Creazione dell'account utente	107
4.2 - Aggiunta contatti.....	110
4.3 - Introduction	116
4.4 - Modifica dei contatti	126
4.5 - Cancellazione di un contatto.....	128
4.6 - Messaggi privati	130
4.7 - Opzione disappearing messages	141
4.8 - Invio via removable drive.....	143
4.9 - Cancellazione messaggi.....	147
4.10 - Creazione gruppo privato.....	149
4.11 - Invito al gruppo privato.....	152
4.12 - Messaggi in gruppo privato.....	159
4.13 - Abbandono del gruppo privato	162

4.14 - Creazione del forum.....	165
4.15 - Invito al forum.....	167
4.16 - Messaggi nel forum.....	172
4.17 - Abbandono del forum.....	176
4.18 - Import di un feed RSS.....	179
4.19 - Condivisione di un feed RSS.....	181
4.20 - Messaggi nel blog.....	184
4.21 - Cancellazione di un feed RSS.....	188
4.22 - Modifica immagine del profilo.....	191
4.23 - Security.....	194
5 - Conclusioni.....	196
6 - Bibliografia.....	197
7 - Ringraziamenti.....	199

Capitolo 1

Introduzione

Nel mondo interconnesso di oggi, sono molteplici le applicazioni mobile che consentono la comunicazione tra le persone. Tali strumenti, grazie alla loro accessibilità, hanno permesso alla maggior parte della popolazione di tenersi in contatto, eliminando le barriere geografiche e temporali.

Tuttavia, è bene evidenziare che tali tecnologie, benché non abbiano di per sé una natura benevola o malvagia, possono essere utilizzate sia per scopi positivi che negativi. Questo dualismo è particolarmente evidente nelle applicazioni che pongono l'accento sulla sicurezza e sull'anonimato dei propri utilizzatori.

È proprio per questo motivo che un'applicazione mobile come Briar, nata con il nobile scopo di fornire un mezzo sicuro per la comunicazione anonima a individui che vivono in condizioni di sorveglianza o sotto regimi dittatoriali, e progettata per garantire il massimo livello di sicurezza può diventare uno strumento perfetto per fini criminali.

Le sue caratteristiche principali, quali l'assenza di server centrali e la capacità di operare anche senza connessione a Internet (ovvero tramite Bluetooth o Wi-Fi), rendono questa applicazione uno strumento estremamente sicuro per coloro che vogliono evitare la sorveglianza governativa o la censura. Tuttavia, queste stesse proprietà possono essere sfruttate anche per scopi diversi. Criminali e organizzazioni illecite, ad esempio, possono utilizzare Briar per scambiarsi informazioni sensibili, evitando il tracciamento da parte delle forze dell'ordine o degli investigatori proprio grazie alle stesse caratteristiche dell'applicazione che permettono di proteggere i cittadini di governi dittatoriali.

Alla luce di questo doppio utilizzo, è emerso un crescente interesse per l'analisi forense di Briar, soprattutto nel campo della sicurezza informatica e dell'investigazione digitale. L'analisi forense di un'applicazione come Briar risulta fondamentale per comprendere come poter ricostruire le azioni degli utenti, a partire dai dati memorizzati sui dispositivi. Questo tipo di analisi può fornire importanti informazioni per indagini criminali o di sicurezza nazionale, consentendo agli investigatori di accedere a tracce digitali che possono essere cruciali per identificare le attività svolte dall'utente oggetto di indagine attraverso l'applicazione.

L'obiettivo principale di questa tesi è quello di eseguire un'analisi forense completa e riproducibile di Briar. In particolare, si mira ad applicare una metodologia forense che permetta di identificare e ricostruire con precisione le azioni compiute dagli utenti, basandosi sui dati memorizzati all'interno del dispositivo.

Per eseguire l'analisi forense di Briar, è stato utilizzato AnForA, un software appositamente progettato per automatizzare e velocizzare le attività di analisi forense su applicazioni mobile. Questo strumento ha giocato un ruolo cruciale, poiché permette di eseguire in modo sistematico e ripetibile una serie di esperimenti su ciascuna funzionalità dell'applicazione. Uno degli obiettivi chiave dell'utilizzo di AnForA è garantire che i risultati ottenuti possano essere facilmente

riprodotti da terze parti, rendendo così l'analisi non solo precisa, ma anche verificabile in un contesto forense.

Un altro aspetto fondamentale dell'analisi è stato lo studio approfondito del codice sorgente di Briar. Questo ha consentito di comprendere i meccanismi interni dell'applicazione, in particolare per quanto riguarda il processo di crittografia e decrittazione dei dati memorizzati all'interno di alcuni file. Attraverso l'analisi del codice, è stato possibile ricostruire i passi necessari per decrittare i dati contenuti in alcuni file della cartella dell'applicazione, file che altrimenti sarebbero rimasti inaccessibili a causa della forte protezione crittografica implementata da Briar.

Il lavoro presentato in questa dissertazione è articolato in quattro capitoli.

Nel secondo capitolo viene fornita una panoramica delle funzionalità di Briar che sono d'interesse dal punto di vista forense. In tale resoconto sono analizzate singolarmente le funzionalità dell'applicazione e come l'utente possa accedere ad esse durante il suo utilizzo.

Il terzo capitolo si focalizza invece sull'architettura interna dell'applicazione, esplorando la struttura del database, la natura e la funzione dei file generati durante l'utilizzo e, soprattutto, il processo di decrittazione dei dati presenti in alcuni di questi file. Viene fornita inoltre una panoramica dei vari algoritmi di cifratura utilizzati dall'applicazione.

Il quarto capitolo è dedicato all'analisi delle singole funzionalità individuate nel primo capitolo. Qui, per ciascuna funzionalità, verranno illustrati in dettaglio gli esperimenti condotti e i risultati ottenuti utilizzando AnForA. Ogni fase dell'analisi è spiegata in modo da fornire un quadro chiaro e completo di come è stato possibile ricostruire le azioni degli utenti, anche in presenza di crittografia o altre forme di protezione dei dati.

Infine, nel capitolo conclusivo, verranno riassunte le difficoltà riscontrate durante l'intero processo di analisi e gli sviluppi futuri di questo studio. In particolare, si discuterà delle funzionalità di Briar che non sono state oggetto d'analisi e delle funzioni future che potrebbero essere aggiunte successivamente all'interno dell'applicazione.

Capitolo 2

Funzionalità di Briar

Briar è una app di messaggistica che mette a disposizione dell'utente diverse funzionalità per comunicare con i propri contatti garantendo la massima sicurezza possibile. Tale protezione è garantita in primis dalla struttura peer-to-peer dell'app che permette la sincronizzazione dei messaggi tra i vari peer senza l'utilizzo di un server centrale e in secondo luogo dall'utilizzo della rete Tor per l'invio dei dati tra i vari peer.

Le funzionalità offerte da Briar alla versione 1.5.11 sono molteplici e permettono attività quali:

- La creazione dell'account utente
- La gestione di contatti
- La gestione di messaggi in chat private
- La gestione di gruppi privati, forum, blog e rss feed
- La gestione delle impostazioni dell'app

2.1 - Creazione dell'account utente

Al primo avvio di Briar, all'utente verrà richiesto di creare un account definendo il nickname e la password. Quest'ultima deve essere di almeno otto caratteri.

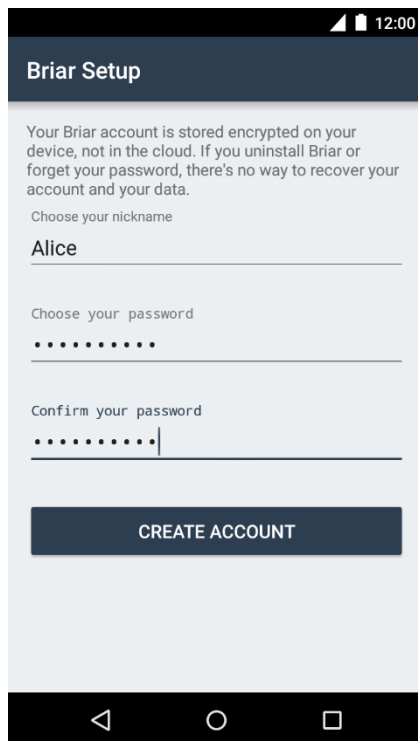


Figura 2.1: Schermata di creazione account

Come rappresentato nella Figura 2.1, una volta definiti nickname e password, toccando il tasto "CREATE ACCOUNT", è creato l'account e l'utente è reindirizzato alla lista dei contatti.

Una volta creato l'account, questo è memorizzato in modo sicuro solo sul dispositivo dell'utente quindi se l'utente si dimentica la password di accesso o l'app Briar è disinstallata, non vi è modo di recuperare l'account.

2.2 - Gestione dei contatti

I contatti di un utente Briar possono essere aggiunti, modificati e cancellati.

2.2.1 - Aggiunta di contatti

Per aggiungere un contatto, l'utente deve toccare l'icona del + in basso a destra nella lista dei contatti e successivamente scegliere tra una delle due opzioni che appaiono (Figura 2.2).

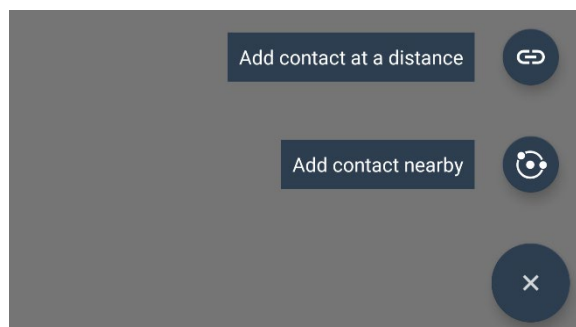


Figura 2.2: Opzioni di aggiunta contatti

L'aggiunta di un contatto può essere effettuata in due modalità principali:

- Attraverso un link (Add contact at a distance/Aggiunta contatto da lontano)
- Attraverso il QR-code (Add contact nearby/Aggiunta contatto da vicino)

2.2.1.1 - Aggiunta contatto da lontano

L'aggiunta di un contatto da lontano consiste nello scambio di un link tra l'utente locale e il contatto che l'utente locale vuole aggiungere alla sua lista dei contatti.

Per effettuare tale scambio, una volta scelta l'opzione di aggiunta contatto da lontano, l'utente locale visualizza la schermata in Figura 2.3 e da tale schermata:

- invia il link `briar://` alla persona che desidera aggiungere (è possibile usare il pulsante "SHARE" per scegliere una app con cui inviare il link).

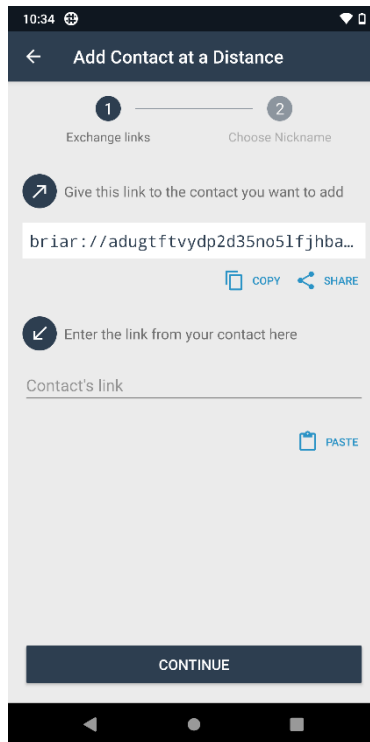


Figura 2.3: Schermata 1 di aggiunta contatto lontano

- incolla il link che riceve dalla persona che desidera aggiungere nel campo di testo sottostante (il link è diverso rispetto a quello inviato dall'utente locale) e clicca su "CONTINUE"

Successivamente visualizza la schermata in Figura 2.4, sceglie un nickname per il nuovo contatto e clicca su "ADD CONTACT".

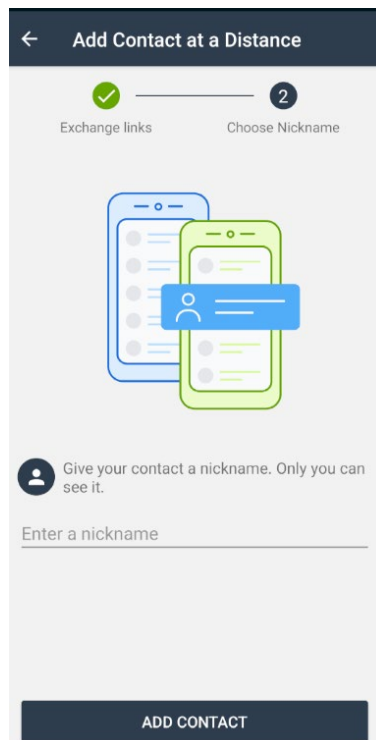


Figura 2.4: Schermata 2 di aggiunta contatto lontano

Il contatto che l'utente locale vuole aggiungere effettua le stesse operazioni dell'utente locale. Dopodiché si visualizza la schermata "Pending Contact Request" (Figura 2.5) che informa l'utente sullo stato di ciascun contatto in sospeso. Briar tenterà di connettersi regolarmente al contatto in sospeso per aggiungerlo. In questa fase il contatto in sospeso è visualizzato solo con il nickname scelto dall'utente locale durante la procedura di aggiunta contatto da lontano.

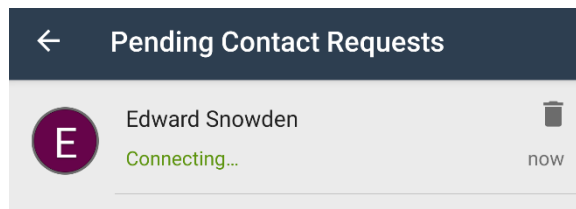


Figura 2.5: Schermata delle richieste di contatto in sospeso

Se Briar non riesce a connettersi al contatto in sospeso dopo 48 ore, la lista dei contatti in sospeso mostrerà il messaggio "Aggiunta del contatto fallita". Entrambi gli utenti che volevano collegarsi devono eliminare il contatto in sospeso dalla lista e aggiungere nuovamente i link l'uno dell'altro.

Una volta che la connessione riesce, l'utente locale e il contatto in sospeso sono aggiunti reciprocamente alle liste dei contatti. Se il nickname inserito dall'utente nella fase di aggiunta contatto è diverso dal nickname scelto dal contatto che si vuole aggiungere in fase di registrazione, il contatto è visualizzato nella lista contatti con il nickname seguito dal nome che il contatto ha scelto quando ha creato il proprio account Briar.

Per esempio se il nickname scelto dall'utente locale è 'Edward Snowden' e il nome scelto dal contatto alla creazione del proprio account è 'ES' allora il nome visualizzato nella lista contatti dell'utente locale sarà 'Edward Snowden (ES)'.

2.2.1.2 - Aggiunta contatto da vicino

L'aggiunta di un contatto da vicino richiede che i due utenti che si vogliono aggiungere l'uno l'altro si incontrino di persona.

Una volta che i due utenti si sono incontrati di persona, per effettuare l'aggiunta dei contatti in questa modalità entrambi devono scannerizzare il codice QR dallo schermo dell'altro utente. In questo modo, ci si assicura di connettersi alla persona giusta, evitando che qualcun altro possa impersonare la persona con cui ci si vuole collegare.

Più nello specifico, entrambi gli utenti che si vogliono collegare tra di loro devono seguire la seguente procedura:

- Selezionare l'opzione 'Add contact nearby'
- Toccare il tasto 'CONTINUE' nella schermata in Figura 2.6



Figura 2.6: Schermata 1 di aggiunta contatto vicino

- Scannerizzare il codice QR del contatto che si vuole aggiungere nella schermata visualizzata (Figura 2.7)
Quando la fotocamera ha scansionato il codice QR, l'utente che sta eseguendo l'aggiunta visualizza il messaggio "Waiting for contact to scan and connect".

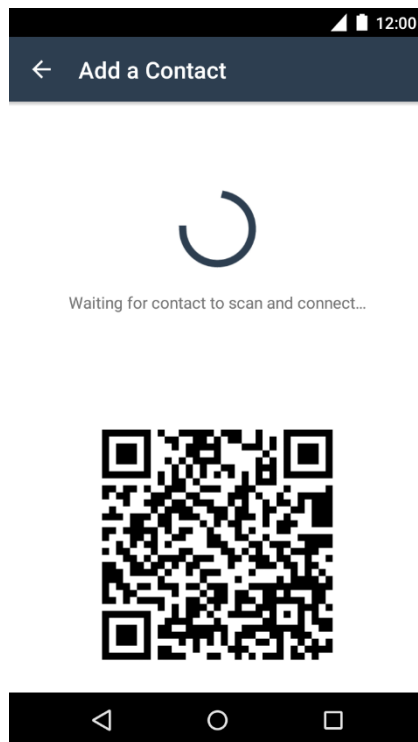


Figura 2.7: Schermata 2 di aggiunta contatto vicino


Una volta che entrambi gli utenti hanno effettuato le azioni precedenti, i dispositivi si scambieranno dati e, dopo pochi secondi, l'utente locale sarà aggiunto alla lista dei contatti dell'altro utente e viceversa.

In questa modalità di aggiunta contatto non è possibile definire un nickname per il contatto inserito quindi il contatto è visualizzato nella lista contatti con il nome scelto dal contatto stesso quando ha creato il proprio account Briar.

2.2.2 - Introduzione di contatti

Quando l'utente locale ha almeno due contatti nella propria lista, può presentare due di essi tra loro. Questa funzionalità permette loro di diventare contatti senza dover effettuare una procedura di aggiunta mediante link o mediante QR code.

Per presentare due contatti tra di loro l'utente locale deve:

- Toccare il nome del primo contatto a cui si vuole inviare una richiesta di presentazione
- Toccare l'icona con i tre puntini  in alto a destra
- Selezionare il tasto 'Make Introduction' dal menù (Figura 2.8)

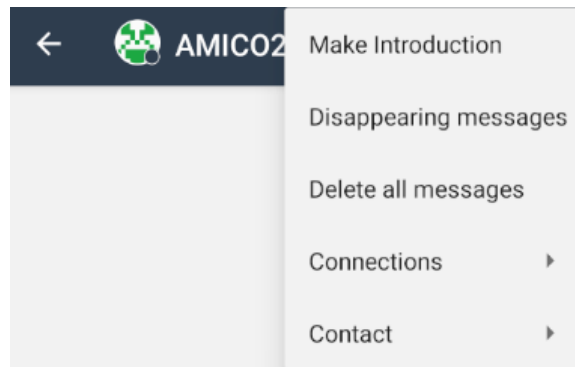


Figura 2.8: Menù del contatto

- Scegliere il secondo contatto che si vuole presentare al primo contatto (Figura 2.9).

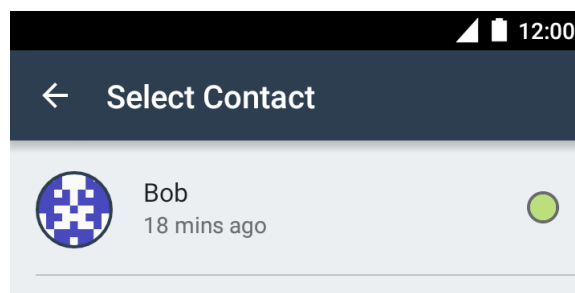


Figura 2.9: Schermata 1 di introduction

- Aggiungere un messaggio opzionale da inviare ai contatti insieme all'invito e toccare il tasto 'MAKE INTRODUCTION' (Figura 2.10)

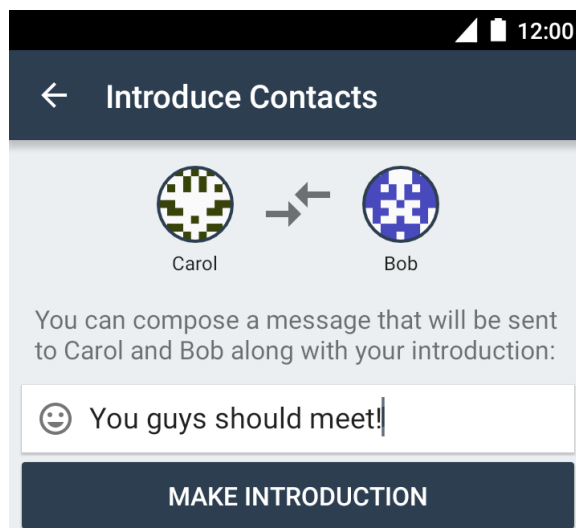


Figura 2.10: Schermata 2 di introduction

A seguito di queste azioni, i contatti a cui si è inviata la richiesta di presentazione vedranno un messaggio (Figura 2.11) che chiede loro se accettano l'introduzione o meno. Solo se entrambi accettano si aggiungeranno l'un l'altro nella propria lista contatti e potranno iniziare a scambiarsi messaggi.

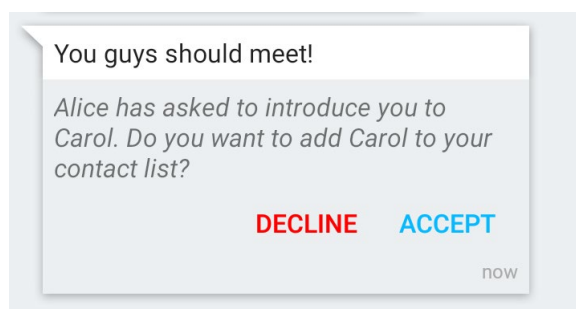



Figura 2.11: Messaggio di richiesta presentazione

2.2.3 - Modifica di contatti

In Briar, l'unica azione di modifica dei contatti già aggiunti è la modifica del nickname associato al contatto.

Per poter usufruire di tale funzionalità è necessario:

- Entrare nella chat privata con il contatto di cui si vuole modificare il nickname
- Toccare l'icona con i tre puntini  in alto a destra
- Selezionare la voce 'Contact' dal menù del contatto (Figura 2.12)

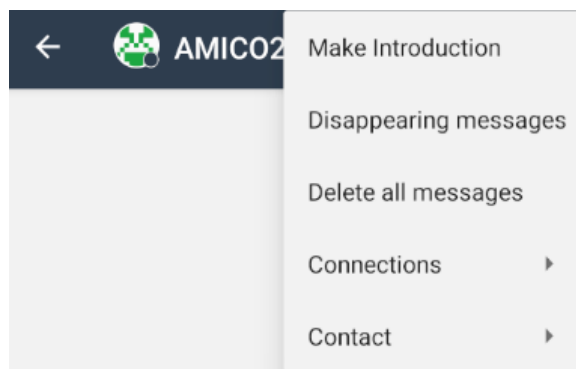


Figura 2.12: Menù chat privata

- Selezionare la voce 'Change contact name' dal menù (Figura 2.13)

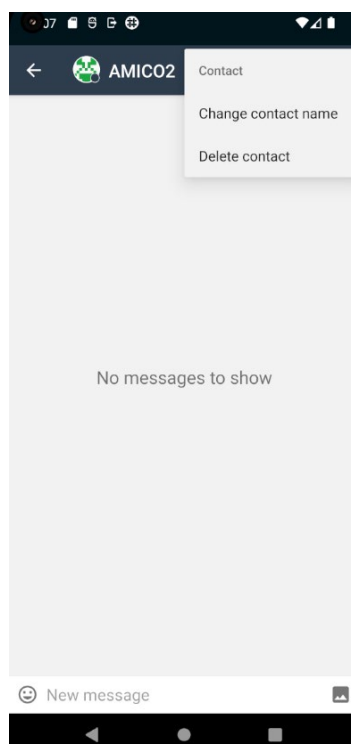


Figura 2.13: Menù 'Contact'

- Inserire il nickname che si vuole associare al contatto nella finestra pop-up che si apre.

A seguito della modifica del nickname il contatto sarà visualizzato nella lista dei contatti in uno dei seguenti modi:


- Con il nickname seguito dal nome scelto dal contatto quando quest'ultimo ha creato il proprio account Briar (se il nickname non è vuoto)
- Solo con il nome scelto dal contatto quando quest'ultimo ha creato il proprio account Briar (se il nickname è vuoto)

2.2.4 - Cancellazione di contatti

La cancellazione dei contatti permette di cancellare un qualsiasi contatto presente nella propria lista dei contatti. Tale funzionalità consente di cancellare un contatto alla volta. Quando un

contatto è eliminato sono eliminati anche tutti i messaggi presenti nella chat privata di quel contatto.

Per effettuare la cancellazione di un contatto è necessario:

- Entrare nella chat privata con il contatto che si vuole eliminare
- Toccare l'icona con i tre puntini  in alto a destra
- Selezionare la voce 'Contact' (Figura 2.14)

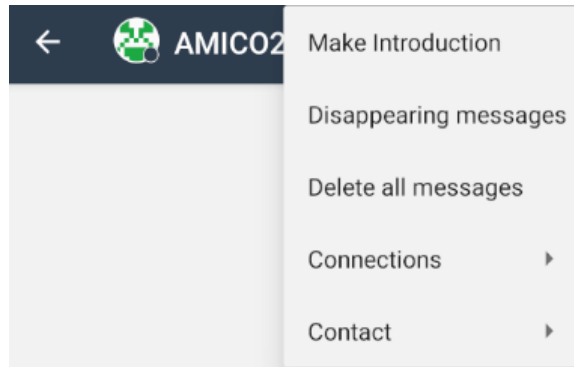


Figura 2.14: Menù chat privata

- Selezionare la voce 'Delete contact' dal menù (Figura 2.15)

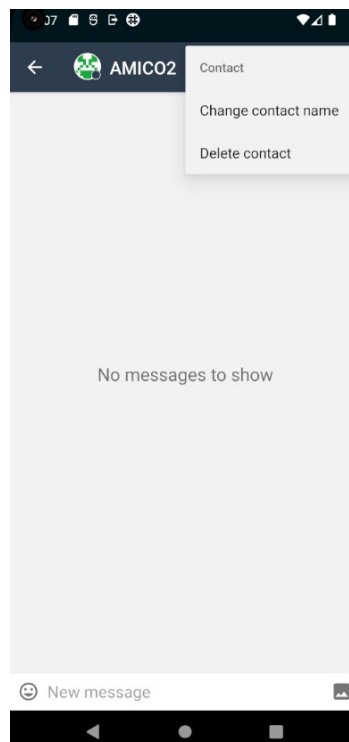


Figura 2.15: Menù 'Contact'

- Dare conferma della cancellazione nella finestra di pop-up che si apre

Quando l'utente locale cancella un proprio contatto, al contatto cancellato non è notificata la sua eliminazione dalla lista dei contatti dell'utente locale. Semplicemente il contatto vedrà l'utente locale perennemente offline.

2.3 - Gestione messaggi in chat private

In Briar un utente ha la possibilità di inviare messaggi privati a ognuno dei contatti della sua lista contatti.

I messaggi privati possono essere messaggi testuali, immagini o inviti a presentazioni o inviti a gruppi privati e forum o richieste di condivisione di un feed RSS.

Per inviare un messaggio di testo privato, nella lista dei contatti l'utente deve toccare il nome del contatto a cui inviare il messaggio, scrivere il testo del messaggio e toccare l'icona ➤ per inviarlo (Figura 2.16).

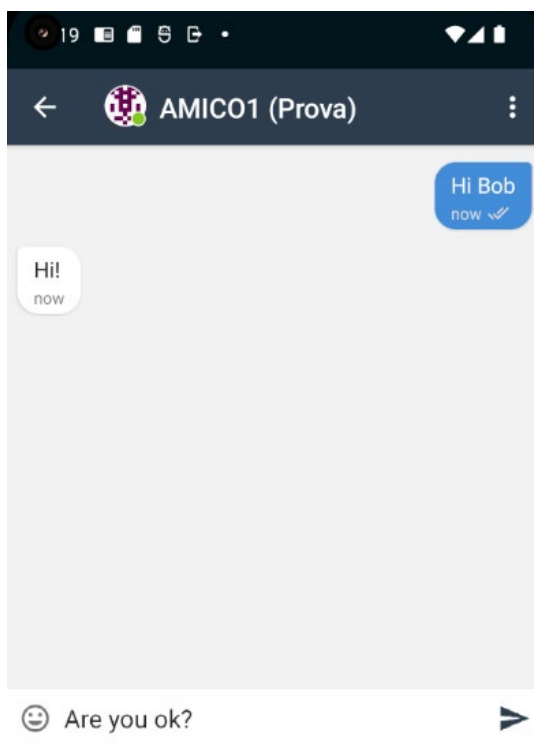


Figura 2.16: Invio messaggio di testo in chat privata

In caso di invio di un'immagine, l'utente deve entrare nella chat privata con il contatto d'interesse (come per un messaggio testuale), successivamente toccare l'icona 📎 (Figura 2.17) in basso a destra e selezionare dal proprio dispositivo l'immagine da inviare.

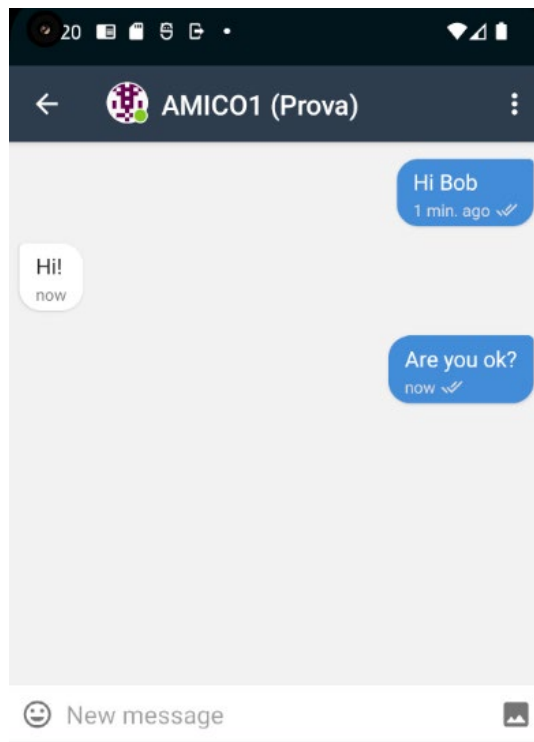


Figura 2.17: Invio immagine in chat privata

A ogni immagine è possibile associare un testo opzionale che l'utente può scrivere una volta selezionata l'immagine da inviare.


Se il destinatario del messaggio privato è offline, il messaggio privato, sia esso testuale che immagine, è recapitato appena il destinatario e il mittente sono entrambi online.

2.3.1 - Invio via removable drive

Tutti i messaggi presenti nella chat privata con un contatto (messaggi testuali, immagini o inviti a gruppi privati e forum) possono essere inviati attraverso l'utilizzo di un disco removibile.

Tale funzionalità è messa a disposizione da Briar per permettere la comunicazione anche nelle situazioni in cui non vi è connessione alla rete. Questa funzione invia i messaggi non ancora inviati a un determinato contatto. Se vi sono messaggi diversi scritti in chat private diverse deve essere eseguito un invio via removable drive per ogni chat privata.

Per utilizzare tale funzionalità, l'utente locale deve:

- Entrare nella chat privata con il contatto a cui vuole inviare i messaggi.
Nella chat privata possono essere già presenti messaggi da inviare o meno. Nel secondo caso l'utente locale scrivere il/i messaggi che vuole inviare.
- Toccare l'icona  in alto a destra
- Selezionare la voce 'Connections' dal menù (Figura 2.18)

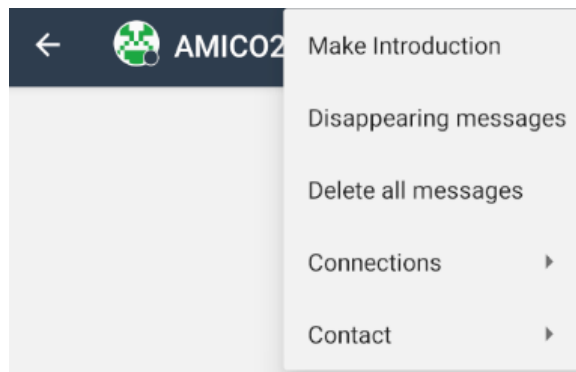


Figura 2.18: Menù chat privata

- Selezionare la voce 'Connect via Removable Drive' dal sottomenù di 'Connections' (Figura 2.19)

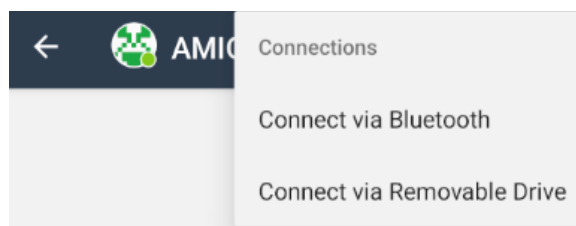


Figura 2.19: Sottomenù 'Connections'

- Toccare il tasto 'SEND DATA' (Figura 2.20)

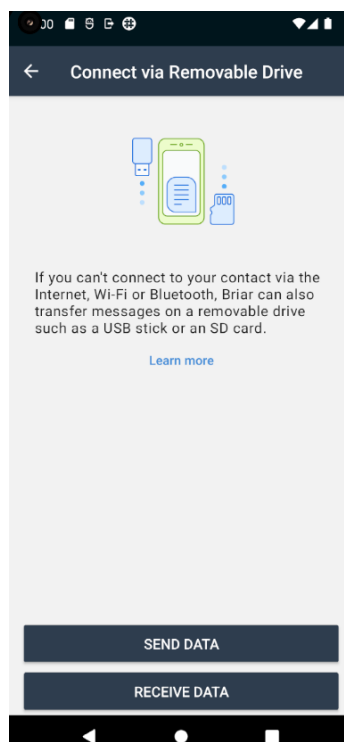


Figura 2.20: Schermata removable drive

- Selezionare il file per l'export dei messaggi.
In questa fase è creato il file contenente i messaggi da inviare e tale file è salvato su un disco removibile collegato al dispositivo dell'utente locale.

Tale file è criptato per garantire che nessun altro oltre al mittente e al destinatario possa leggere il contenuto dei messaggi.

Se non sono presenti messaggi da inviare l'app notifica ciò e non permette di proseguire all'export dei messaggi.

Una volta conclusa questa procedura, l'utente locale si occuperà di far recapitare fisicamente al destinatario dei messaggi il disco removibile in cui è salvato il file.

Il destinatario, ricevuto il disco removibile, esegue la stessa procedura effettuata per l'invio dall'utente locale con l'unica differenza che nella schermata in Figura 2.20 toccherà il tasto 'RECEIVE DATA'.

2.3.2 - Opzione disappearing messages

Per i messaggi presenti nelle chat private è possibile attivare l'opzione di disappearing messages.


Tale opzione setta un timer di sette giorni per ogni messaggio inviato nella chat privata in cui si è attivata l'impostazione.

Scaduto il timer tutti i messaggi all'interno della conversazione privata (messaggi di testo, inviti a gruppi privati e forum) sono cancellati come segue:

- Per il mittente i messaggi sono cancellati sette giorni dopo che sono stati inviati
- Per il destinatario i messaggi sono cancellati sette giorni dopo che sono stati letti

Se l'opzione è attivata da uno solo dei due utenti, al primo messaggio inviato dall'utente che ha attivato l'opzione, questa verrà attivata anche per il destinatario.

Per attivare o disattivare questa impostazione l'utente locale deve:

- Entrare nella chat privata con il contatto su cui vuole attivare o disattivare tale impostazione
Nella chat privata possono essere già presenti messaggi da inviare o meno. Nel secondo caso l'utente locale scrivere il/i messaggi che vuole inviare.
- Toccare l'icona  in alto a destra
- Selezionare la voce 'Disappearing messages' dal menù (Figura 2.21)

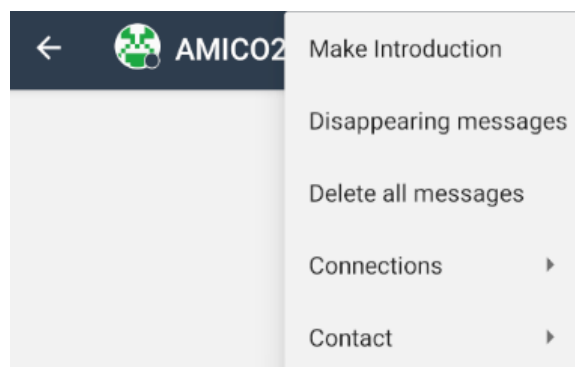


Figura 2.21: Menù chat privata

- Attivare o disattivare l'opzione (Figura 2.22)

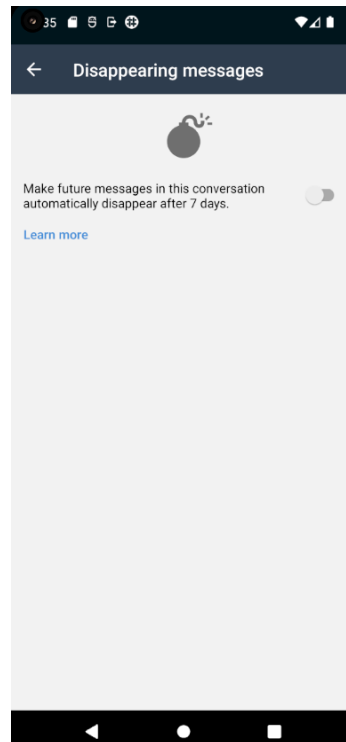



Figura 2.22: Schermata gestione opzione Disappearing messages

2.3.3 - Cancellazione di messaggi

Nelle chat private è possibile cancellare tutti i messaggi scambiati con un determinato contatto.

Per eliminare tutti i messaggi l'utente locale deve:

- Entrare nella chat privata con il contatto che vuole eliminare.
- Toccare l'icona  in alto a destra
- Selezionare la voce 'Delete all messages' dal menù (Figura 2.23)

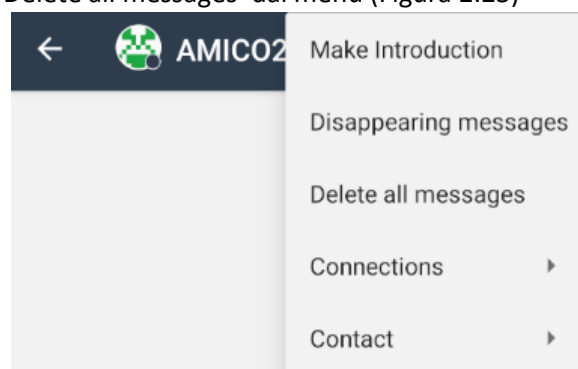


Figura 2.23: Menù chat privata

- Dare conferma della cancellazione del contatto nella finestra di pop-up che si apre.


2.4 - Gestione gruppi privati

Un gruppo privato è un gruppo in cui solo il creatore può invitare altri utenti (suoi contatti) a partecipare.

Un utente può prendere parte a un gruppo privato o come creatore (nel caso abbia creato lui stesso il gruppo privato) o come invitato (nel caso in cui sia stato invitato dal creatore a partecipare al gruppo).

2.4.1 - Creazione gruppo privato

Per creare un gruppo privato, l'utente deve:

- Aprire il menu principale di Briar toccando l'icona 
- Selezionare la voce 'Private Groups' (Figura 2.24)

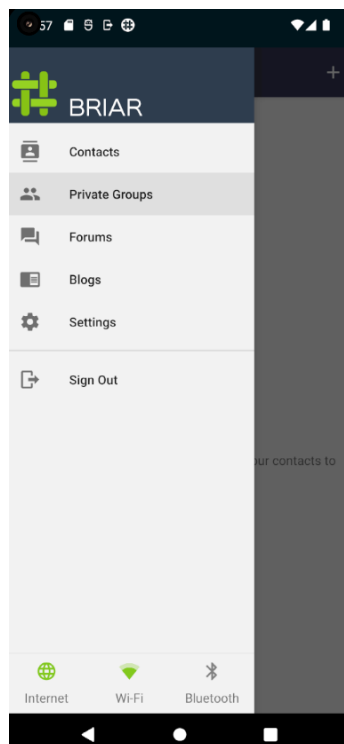



Figura 2.24: Menù principale

- Toccare l'icona in alto a destra  per creare un nuovo gruppo
- Scegliere il nome del gruppo
- Toccare il tasto 'CREATE GROUP' (Figura 2.25)

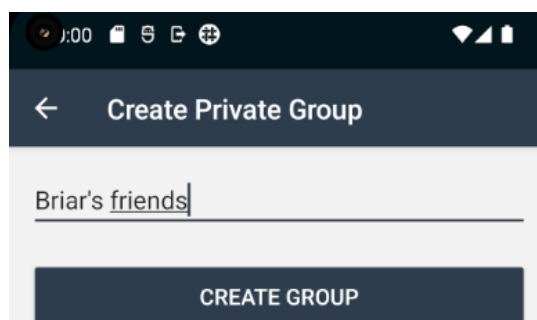


Figura 2.25: Schermata creazione gruppo privato

Come mostrato in Figura 2.26, il nuovo gruppo è aggiunto alla lista dei gruppi privati dell'utente

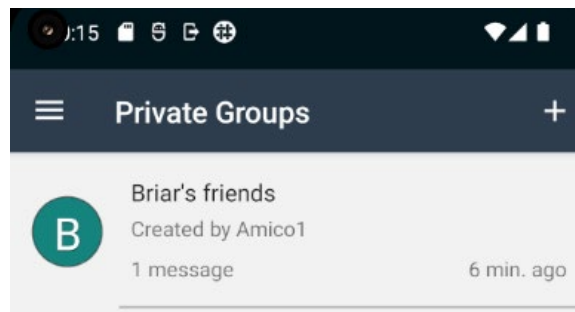



Figura 2.26: Lista dei gruppi privati

Quando un gruppo privato è stato appena creato l'unico membro del gruppo è il creatore.

2.4.2 - Invito al gruppo privato

Per aggiungere membri a un gruppo privato è necessario che l'utente che ha creato il gruppo inviti i propri contatti a partecipare ad esso.

Se il creatore vuole invitare i suoi contatti a partecipare al gruppo privato deve:

- Entrare nel gruppo privato
- Toccare l'icona  in alto a destra
- Selezionare la voce 'Invite Members' del menù (Figura 2.27)

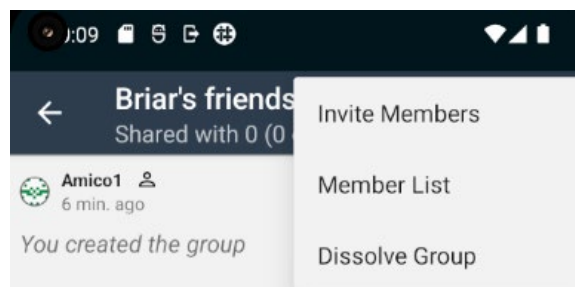


Figura 2.27: Menù del gruppo privato

- Selezionare il/i contatti a cui vuole inviare un invito a partecipare al gruppo e premere l'icona di spunta in alto a destra (Figura 2.28)

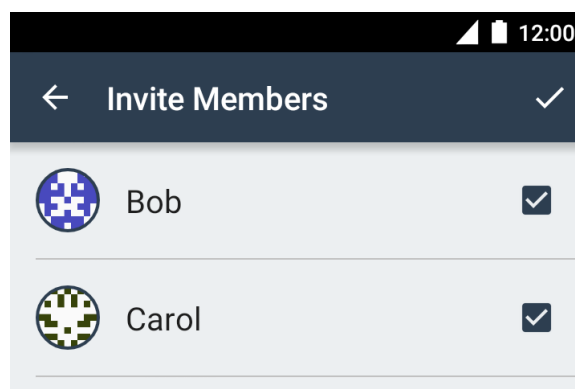


Figura 2.28: Lista dei possibili contatti da invitare

- Aggiungere un messaggio opzionale che sarà inviato ai contatti selezionati insieme all'invito e toccare il tasto 'SEND INVITATION' (Figura 2.29)

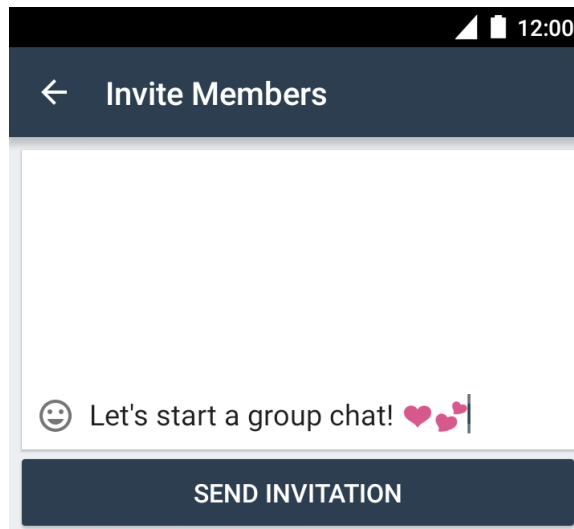


Figura 2.29: Schermata di inserimento messaggio opzionale

Una volta ricevuto l'invito, l'invitato diventa membro del gruppo privato solo ed esclusivamente se accetta l'invito ricevuto.

2.4.3 - Messaggi in gruppo privato

I messaggi nei gruppi privati sono organizzati in threads e ogni membro può rispondere a un messaggio o iniziare un nuovo thread.

La Figura 2.30 mostra un esempio di gruppo privato in cui sono stati inviati molteplici messaggi.

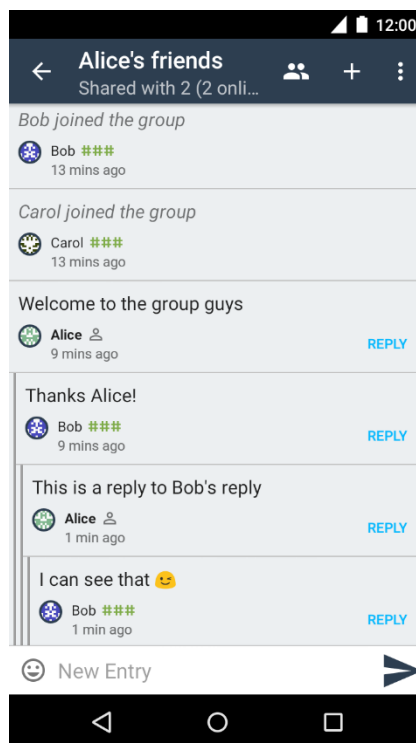


Figura 2.30: Schermata messaggi gruppo privato

Per iniziare un nuovo thread, l'utente deve scrivere nella textbox 'New Entry' (Figura 2.30) e toccare l'icona ➤. Invece, per rispondere a un messaggio deve toccare il tasto 'reply' relativo al messaggio a cui si vuole replicare (Figura 2.30), scrivere in 'New Entry' e toccare l'icona ➤.

2.4.4 - Abbandono del gruppo privato

Ogni membro del gruppo può abbandonarlo ma se il creatore abbandona il gruppo, questo non è più utilizzabile anche per tutti gli altri partecipanti. Infatti quando il creatore abbandona il gruppo, gli altri membri possono continuare a visualizzare tutti i messaggi ricevuti fino a quel momento ma non possono più inviare nuovi messaggi o ottenere i messaggi non ancora ricevuti.

Per abbandonare un gruppo privato l'utente locale deve:

- Selezionare il gruppo privato che si vuole abbandonare dalla lista dei gruppi privati
- Toccare l'icona ☰ in alto a destra
- Selezionare la voce 'Dissolve Group' (se l'utente locale è il creatore del gruppo) o 'Leave Group' (se l'utente locale è un partecipante) nel menù (Figura 2.31)

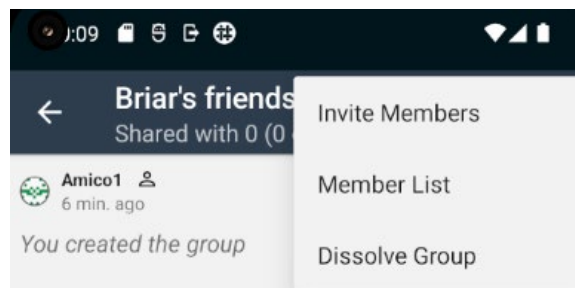


Figura 2.31: Menù del gruppo privato

- Dare conferma dell'abbandono del gruppo privato nella finestra di pop-up che si apre.

2.5 - Gestione del forum

Un forum è una conversazione pubblica in cui ogni partecipante può invitare i propri contatti.

Non vi sono differenze tra il membro creatore e i membri non creatore.

2.5.1 - Creazione forum

Per creare un forum l'utente deve:

- Aprire il menu principale di Briar toccando l'icona ☰
- Selezionare la voce 'Forums' (Figura 2.32)

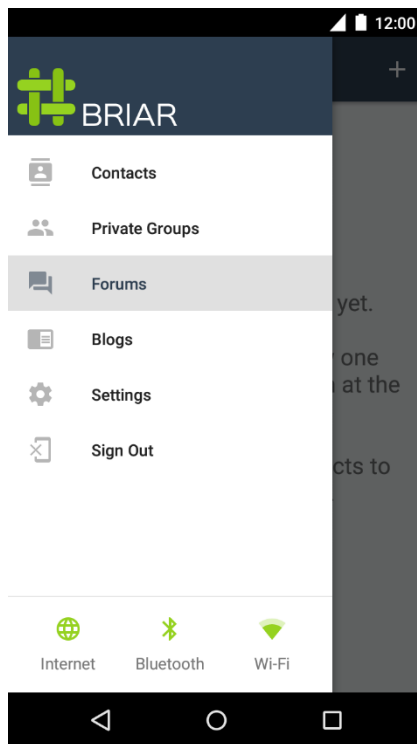


Figura 2.32: Menù principale

- Toccare l'icona in alto a destra **+** per creare un nuovo forum
- Scegliere il nome del forum e toccare il tasto 'CREATE FORUM' (Figura 2.33)

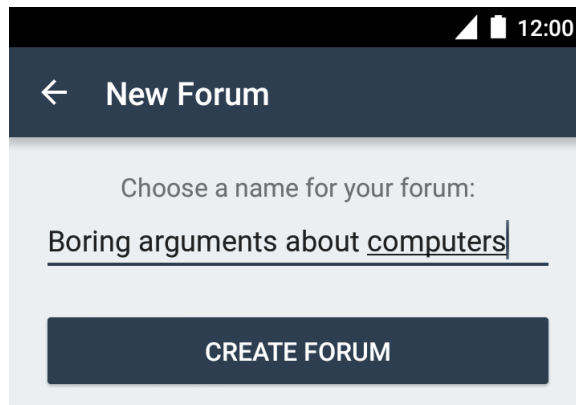


Figura 2.33: Schermata creazione forum

- Il nuovo forum è aggiunto alla lista dei forum dell'utente come mostrato in Figura 2.34.

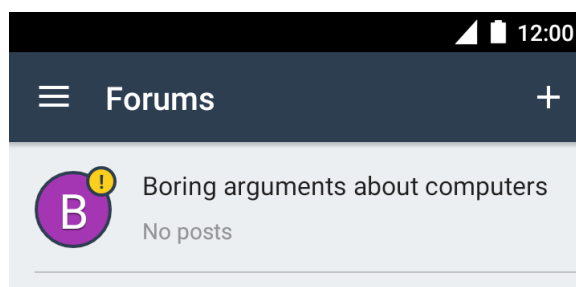



Figura 2.34: Lista dei forum

Quando un forum è stato appena creato l'unico membro del forum è il creatore.

2.5.2 - Invito al forum

Se un membro vuole invitare i propri contatti a partecipare al forum di cui è già membro deve:

- Entrare nel forum come descritto nei primi due punti della Sezione 2.5.1 - Creazione forum
- Toccare l'icona  in alto a destra
- Selezionare la voce 'Share Forum' (Figura 2.35)

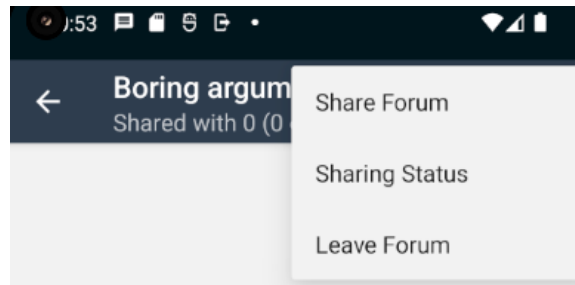


Figura 2.35: Menù del forum

- Selezionare i contatti a cui si vuole inviare un invito a partecipare al forum e premere l'icona di spunta in alto a destra (Figura 2.36)

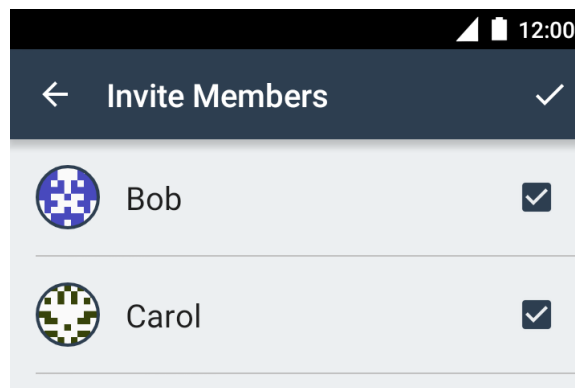


Figura 2.36: Lista dei possibili contatti da invitare

- Aggiungere un messaggio opzionale che sarà inviato insieme all'invito ai contatti selezionati e toccare il tasto 'SHARE FORUM' (Figura 2.37)

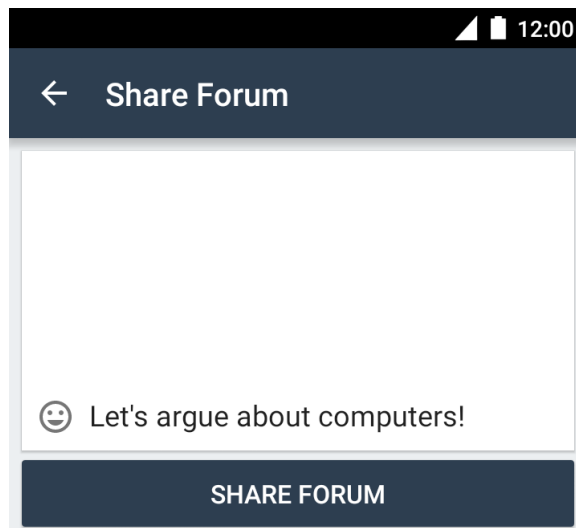


Figura 2.37: Schermata di inserimento messaggio opzionale

Una volta ricevuto l'invito, l'invitato diventa membro del forum solo ed esclusivamente se accetta l'invito ricevuto.

2.5.3 - Messaggi in forum

Come per i gruppi privati anche nei forum i messaggi sono organizzati in threads e ogni membro può rispondere a un messaggio o iniziare un nuovo thread.

La Figura 2.38 mostra un esempio di forum in cui è stato scritto un post.

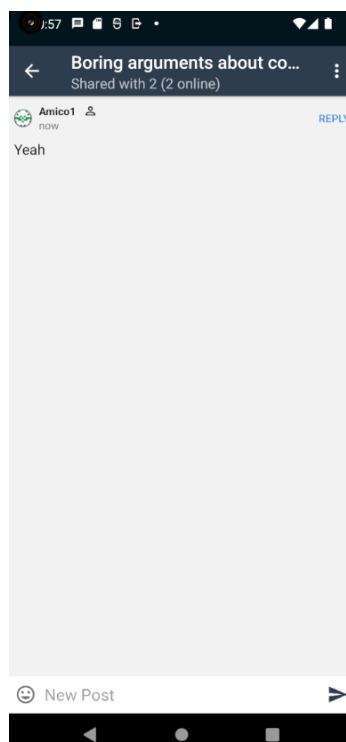


Figura 2.38: Schermata messaggi forum

Per iniziare un nuovo thread, l'utente deve scrivere nella textbox 'New Post' (Figura 2.38) e toccare l'icona ➤. Invece, per rispondere a un messaggio deve toccare il tasto reply relativo al messaggio a cui si vuole replicare (Figura 2.38), scrivere in 'New Post' e toccare l'icona ➤.

2.5.4 - Abbandono del forum

Ogni membro, compreso il creatore, può abbandonare il forum in qualsiasi momento. Tuttavia, l'abbandono di un singolo membro non influisce sulla possibilità degli altri membri di utilizzare il forum. Se, però, l'utente che ha abbandonato il forum ha precedentemente invitato altri utenti e quest'ultimi hanno accettato l'invito diventando partecipanti del forum, gli utenti invitati dal membro che ha abbandonato il forum non riceveranno più i messaggi inviati nel forum.

Si prenda come esempio l'utente A che invita gli utenti B e C a partecipare al forum. Sia B che C accettano l'invito di A e diventano membri del forum. Se A abbandona il forum, B e C possono continuare a scrivere nel forum ma B non riceverà più i messaggi postati nel forum da C e viceversa.

Per abbandonare un forum di cui è membro l'utente locale deve:

- Entrare nel forum
- Toccare l'icona ⋮ in alto a destra
- Selezionare la voce 'Leave Forum' (Figura 2.39)

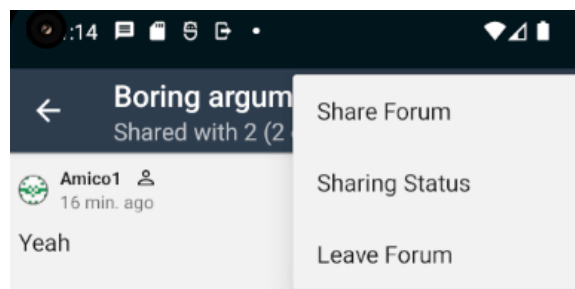


Figura 2.39: Menù del forum

- Dare conferma dell'abbandono del forum nella finestra di pop-up che si apre.

2.6 - Gestione del blog

Ogni account Briar ha un built-in blog che è creato in fase di registrazione dell'account.

Il blog può essere usato dall'utente per postare notizie o aggiornamenti sulla propria vita ed è condiviso con tutti i suoi contatti.

Ogni utente di Briar ha solo un blog ad esso associato e tale blog non può essere cancellato.

Per accedere al proprio blog o vedere i post del blog dei propri contatti l'utente deve:

- Aprire il menu principale di Briar toccando l'icona ☰
- Selezionare la voce 'Blogs' (Figura 2.40)

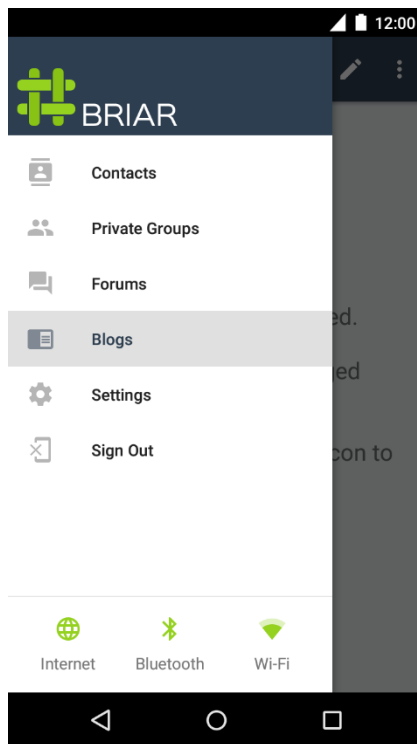


Figura 2.40: Menù principale

Entrando nella sezione 'Blogs', l'utente visualizza il feed del suo blog e quello dei suoi contatti.

2.6.1 - Scrivere un post

Una volta entrato nella sezione 'Blogs', per scrivere un post l'utente deve:

- Toccare l'icona a forma di penna in alto a destra (Figura 2.41)

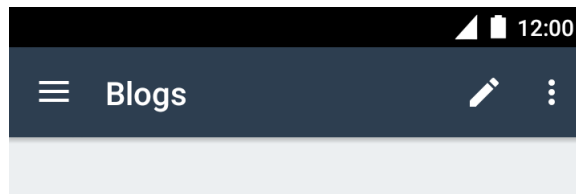


Figura 2.41: Schermata del feed del blog

- Scrivere il testo del post e toccare il tasto 'PUBLISH' (Figura 2.42)

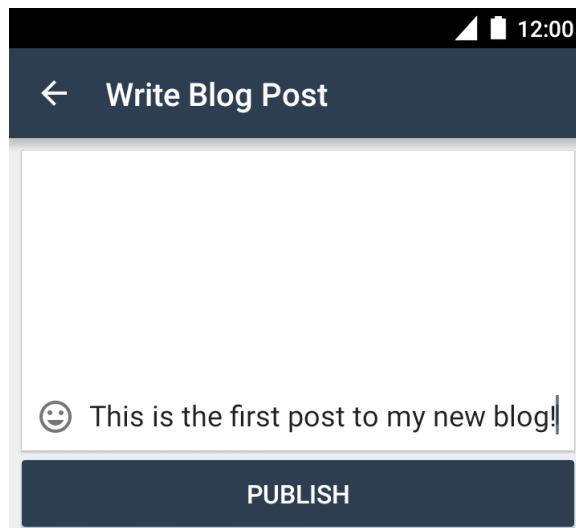



Figura 2.42: Schermata di inserimento testo post

Il nuovo post apparirà nel feed del blog dell'utente e di tutti i suoi contatti.

2.6.2 - Re-blog di un post

Una volta entrato nella sezione 'Blogs', per re-bloggare un post l'utente deve:

- Toccare l'icona di re-blog  (Figura 2.43)

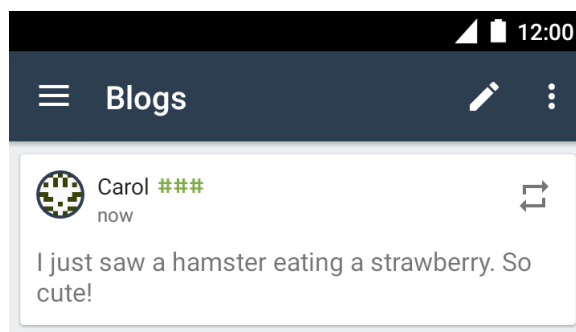


Figura 2.43: Post di un blog

- Aggiungere un commento opzionale e toccare il tasto 'REBLOG' (Figura 2.44)

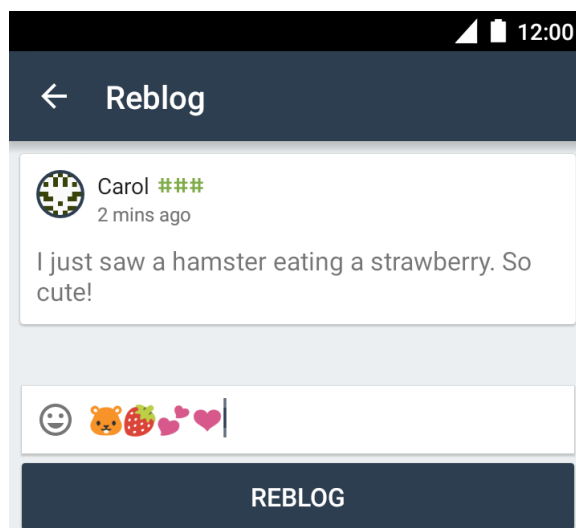


Figura 2.44: Schermata di re-blog

Il re-blog del post apparirà nel feed del blog dell'utente e in quello di tutti i suoi contatti con il commento allegato.


2.7 - Gestione del feed RSS

In Briar è possibile accedere alle notizie di qualsiasi blog o sito di notizie che abbia un feed RSS.

Gli articoli dei feed RSS importati dall'utente locale in Briar sono scaricati attraverso la rete Tor per proteggere la privacy degli utenti. Ognuno di questi articoli può essere ripubblicato e commentato esattamente come i post di un blog.

2.7.1 - Import di un feed RSS

Per visualizzare le notizie di un determinato feed RSS l'utente locale deve importare tale feed RSS in Briar. Per fare ciò l'utente locale deve:

- Aprire il menu principale di Briar toccando l'icona 
- Selezionare la voce 'Blogs' (Figura 2.45)

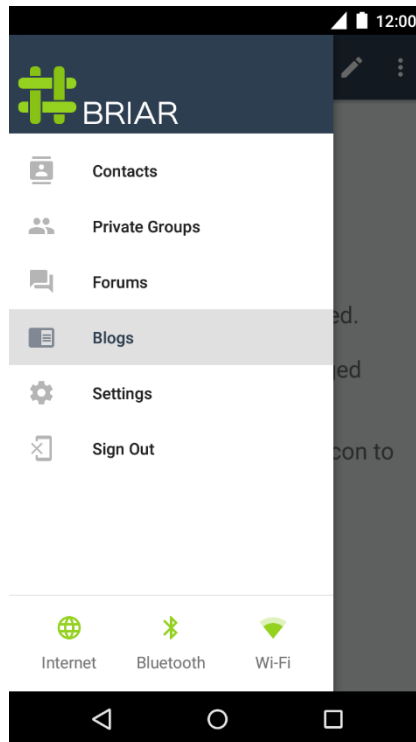



Figura 2.45: Menù principale

- Toccare l'icona  in alto a destra
- Selezionare la voce 'RSS Feeds' dal menù (Figura 2.46)

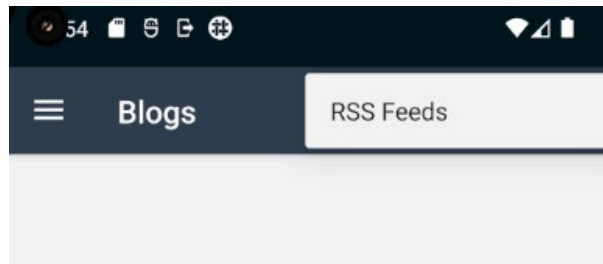


Figura 2.46: Menù del blog

- Toccare l'icona **+** in alto a destra
- Inserire l'URL del feed RSS e toccare il tasto 'IMPORT' (Figura 2.47)

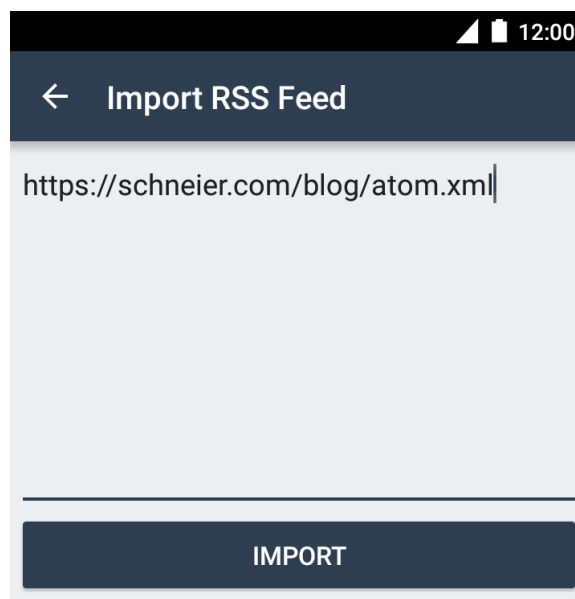


Figura 2.47: Schermata di inserimento URL feed RSS

Una volta importato il feed RSS in Briar, gli articoli già pubblicati e quelli che sono pubblicati successivamente sono scaricati e aggiunti al feed del blog dell'utente che ha importato il feed RSS.

Gli articoli del feed RSS non sono condivisi con i contatti dell'utente a meno che esso non decida di re-bloggarli o di condividere l'intero RSS feed.

2.7.2 - Condivisione di un feed RSS

Una volta importato un feed RSS nel proprio blog, l'utente può condividere tale feed RSS con i suoi contatti.

Per fare ciò l'utente locale deve:

- Entrare nella sezione Blogs dal menù principale
- Toccare l'icona **⋮** in alto a destra
- Selezionare la voce 'RSS Feeds' dal menù (Figura 2.48)

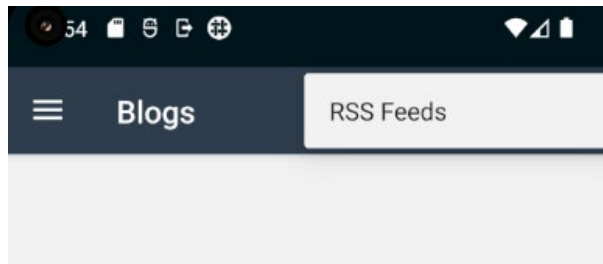


Figura 2.48: Menù del blog

- Selezionare il feed RSS che si vuole condividere (Figura 2.49)

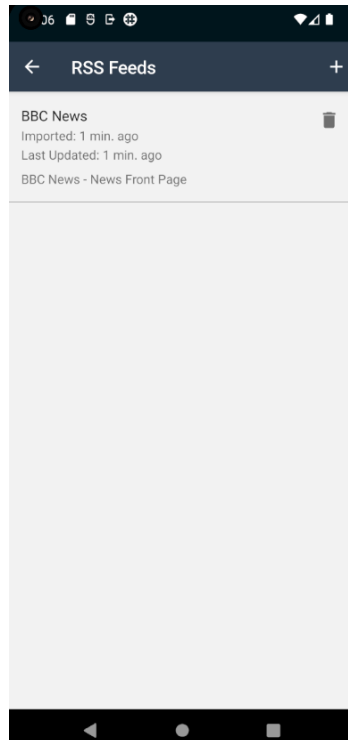



Figura 2.49: Lista dei feed RSS

- Toccare l'icona  in alto a destra
- Selezionare i contatti a cui si vuole inviare l'invito di condivisione del feed RSS e premere l'icona di spunta in alto a destra (Figura 2.50)

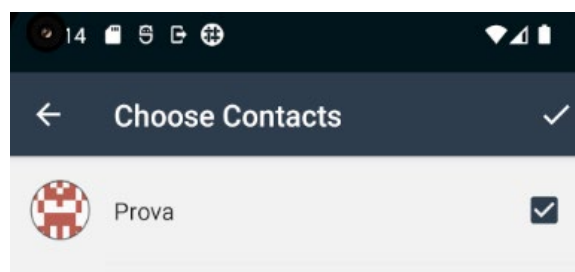


Figura 2.50: Lista dei possibili contatti con cui condividere il feed RSS

- Aggiungere un messaggio opzionale che sarà inviato insieme all'invito di condivisione del feed RSS e toccare il tasto 'SHARE BLOG' (Figura 2.51)

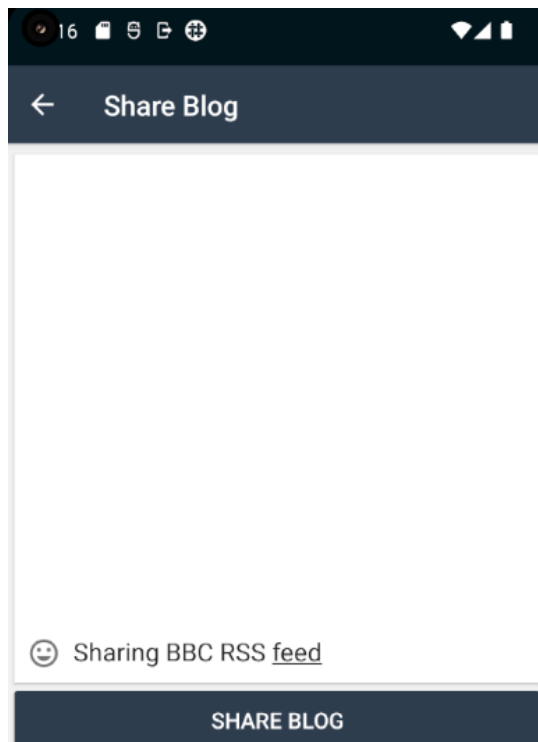



Figura 2.51: Schermata di inserimento messaggio opzionale

Il feed RSS è condiviso con l'utente che ha ricevuto l'invito di condivisione solo ed esclusivamente se quest'ultimo accetta tale invito. L'accettazione della condivisione fa sì che il feed RSS sia importato nel feed del blog dell'utente che ha accettato l'invito.

2.7.3 - Re-blog di un articolo del feed RSS

Se l'utente locale vuole re-bloggare un articolo del feed RSS che ha precedentemente importato deve:

- Entrare nella sezione Blogs dal menù principale
- Toccare l'icona di re-blog  (Figura 2.52)

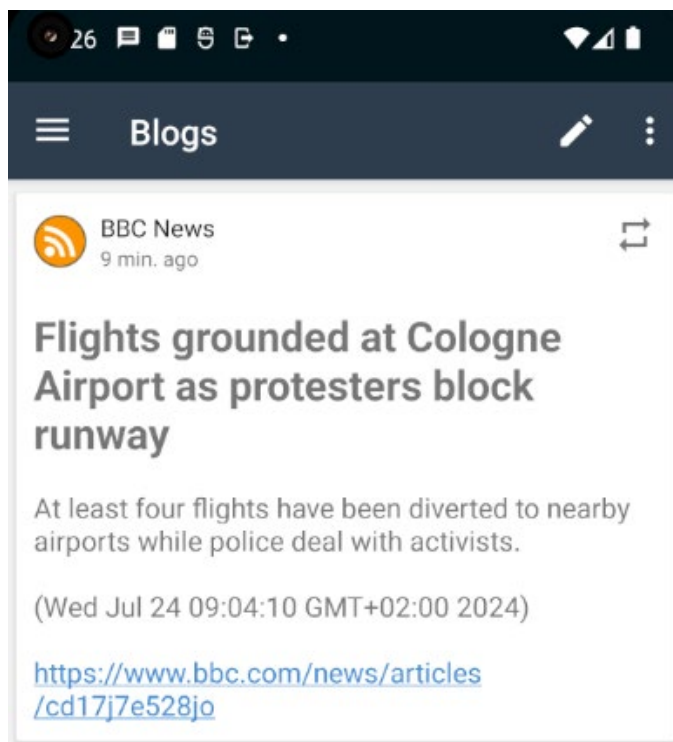


Figura 2.52: Articolo di un feed RSS

- Aggiungere un commento opzionale e toccare il tasto 'REBLOG' (Figura 2.53)

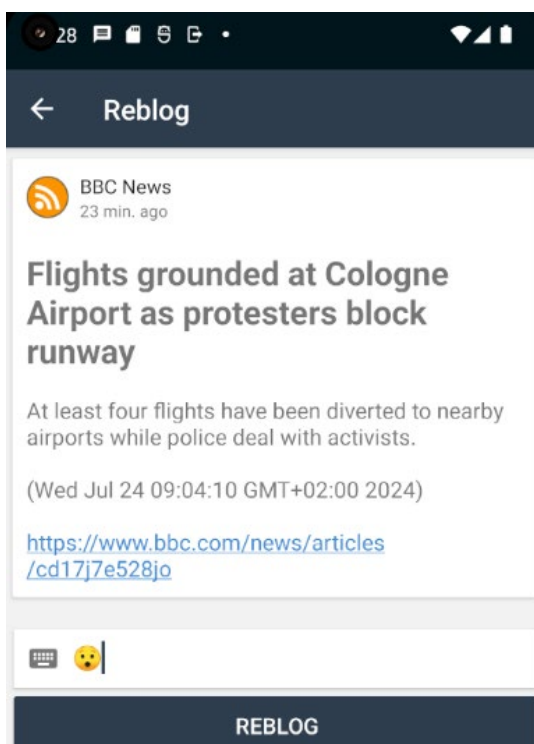


Figura 2.53: Schermata di re-blog



Il re-blog dell'articolo apparirà con il commento allegato nel feed del blog dell'utente autore del re-blog e in quello di tutti i suoi contatti.

2.7.4 - Cancellazione di un feed RSS

Ogni feed RSS importato o dall'utente o attraverso l'accettazione di un invito di condivisione può essere cancellato.

Quando avviene l'eliminazione di un feed RSS, nel feed del blog dell'utente locale sono cancellati tutti gli articoli del feed RSS tranne quelli che sono stati re-bloggati, mentre i contatti con cui l'utente locale ha precedentemente condiviso il feed RSS mantengono gli articoli del feed RSS ricevuti fino a quel momento ma smettono di ricevere gli articoli che saranno pubblicati in futuro.

Per eliminare un feed RSS l'utente deve:

- Entrare nella sezione Blogs dal menù principale
- Toccare l'icona  in alto a destra
- Selezionare la voce 'RSS Feeds' dal menù
- Toccare l'icona  di fianco al nome del feed RSS che si vuole cancellare (Figura 2.54)

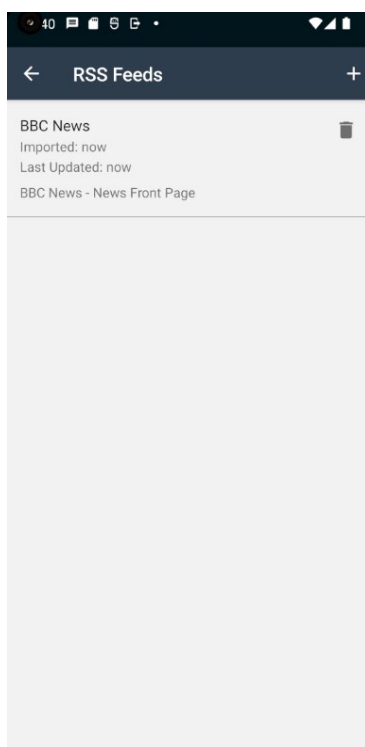


Figura 2.54: Lista dei feed RSS


- Dare conferma della cancellazione del feed RSS nella finestra di pop-up che si apre.

2.8 - Gestione delle impostazioni

Nella sezione delle impostazioni è possibile personalizzare l'esperienza di utilizzo di Briar.

Più nello specifico in questa sezione l'utente locale può cambiare la sua immagine di profilo, definire il tema e la lingua dell'app, definire quale tipologia di connessione deve essere utilizzata dall'app e settare diverse impostazioni relative alla sicurezza e alle notifiche dell'app.

Per accedere alle impostazioni l'utente deve:

- Aprire il menu principale di Briar toccando l'icona 

- Selezionare la voce 'Settings' (Figura 2.55)

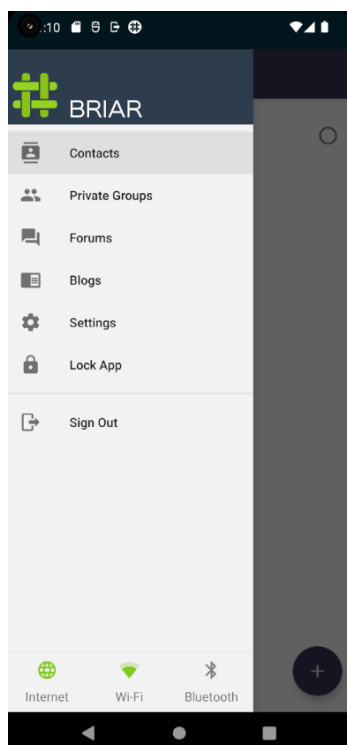


Figura 2.55: Menù principale

2.8.1 - Modifica immagine del profilo

Se l'utente vuole modificare la propria immagine del profilo deve entrare nella sezione Settings, toccare la sua attuale immagine del profilo e selezionare l'immagine che vuole utilizzare dal proprio dispositivo.

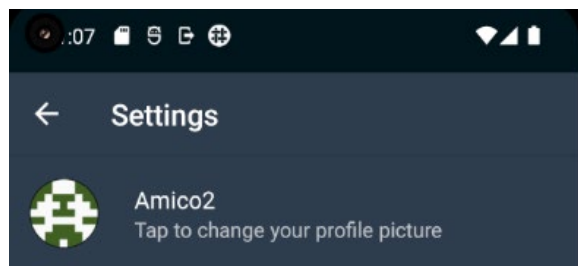


Figura 2.56: Profilo utente

2.8.2 - Impostazioni Display

Nella sezione 'Display' si può cambiare il tema utilizzato da Briar scegliendo tra il tema Light, Dark, Automatic (Briar cambia il tema in base al momento della giornata) e System default (Briar usa il tema del sistema).

Inoltre, è possibile modificare la lingua utilizzata nell'app. Di default la lingua utilizzata è quella del sistema ma è possibile settare una lingua qualsiasi tra quelle messe a disposizione.

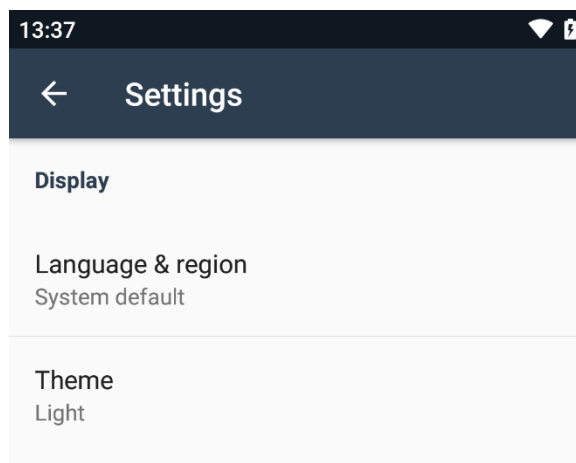


Figura 2.57: Sezione Display

2.8.3 - Impostazioni Connections

Nella sezione 'Connections' è possibile settare diverse impostazioni relative a quale tipo di connessioni possono essere utilizzate da Briar per scambiare dati con i contatti dell'utente.

Briar può utilizzare tre tipi di connessione:

- Bluetooth
- Wi-Fi
- Internet attraverso la rete Tor

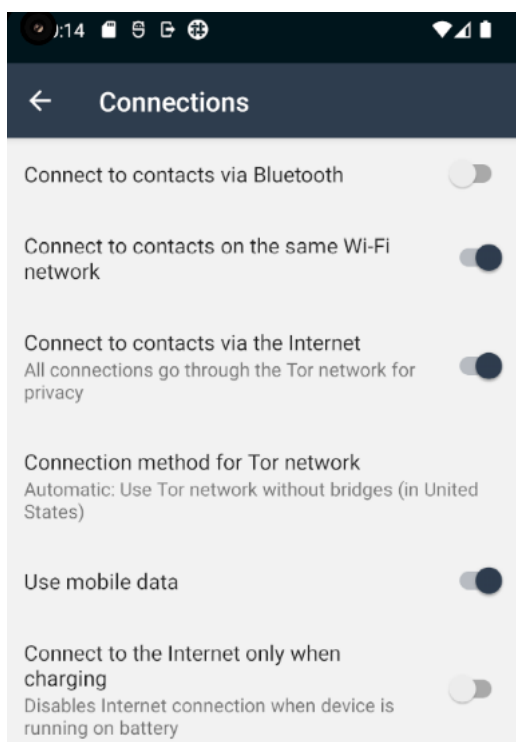


Figura 2.58: Sezione Connections

Abilitando l'impostazione 'Connect to contacts via Bluetooth' (prima voce della Figura 2.58) si dà il permesso a Briar di scambiare dati con i propri contatti attraverso il Bluetooth.

Abilitando l'impostazione 'Connect to contacts on the same Wi-Fi network' (seconda voce della Figura 2.58) si dà il permesso a Briar di scambiare dati con i contatti dell'utente che sono connessi alla stessa rete Wi-Fi dell'utente locale.

Queste due impostazioni, se abilitate, permettono lo scambio di dati tra utenti Briar se i due sono fisicamente vicini. Lo scambio di dati avviene solo ed esclusivamente se i due utenti si sono aggiunti l'un l'altro tra i propri contatti.

Abilitando l'impostazione 'Connect to contacts via the Internet' (terza voce della Figura 2.58) si dà il permesso a Briar di utilizzare la rete Tor per scambiare dati con i contatti dell'utente locale.

La sezione 'Connection method for Tor network' (quarta voce della Figura 2.58) permette di selezionare il metodo di connessione che la rete Tor deve utilizzare.

Sono disponibili tre metodi:

- Automatic based on location (Briar sceglie come connettersi in base alla posizione corrente)
- Use Tor network without Bridges (Briar si connette a Tor senza usare Bridges)
- Use Tor network with Bridges (Briar si connette a Tor usando Bridges)

I Bridges sono nodi non pubblici che aiutano a garantire l'accesso continuo alla rete Tor.

Abilitando l'impostazione 'Use mobile data' (quinta voce della Figura 2.58) si dà il permesso a Briar di utilizzare i dati mobili. Se tale impostazione è disabilitata Briar usa internet solo quando il dispositivo è collegato a una rete Wi-Fi.

Infine, abilitando l'impostazione 'Connect to the internet only when charging' (sesta voce della Figura 2.58) si dà il permesso a Briar di collegarsi alla rete solo quando il dispositivo è in carica.

2.8.4 - Impostazioni Security

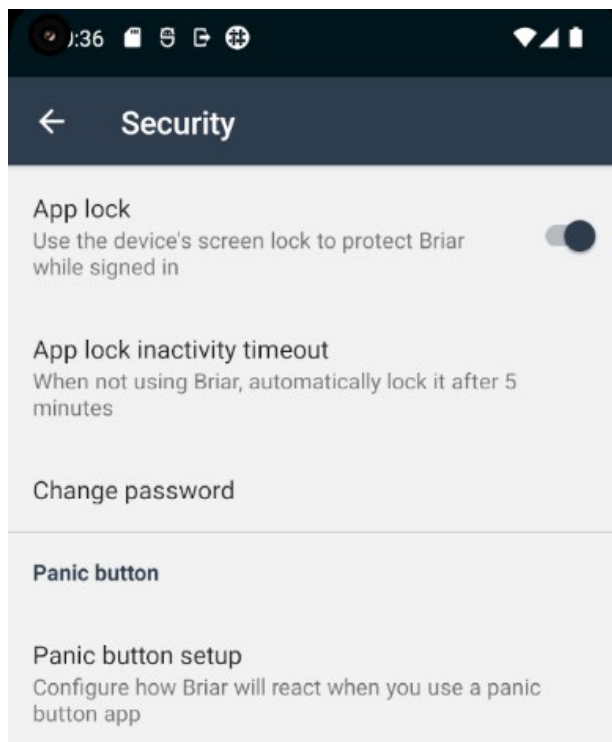


Figura 2.59: Sezione Security

Per proteggere la privacy dell'utente quando altre persone stanno utilizzando il suo dispositivo, è possibile bloccare Briar senza effettuare la disconnessione. In questo modo, Briar rimarrà inutilizzabile finché l'utente non inserisce il PIN, il pattern o la password.

Quando questa opzione è attivata, è possibile sbloccare Briar utilizzando lo stesso metodo di sblocco settato per lo sblocco del dispositivo.

Solo quando l'impostazione 'App lock' (prima voce della Figura 2.59) è abilitata è possibile settare l'impostazione 'App lock inactivity timeout' (seconda voce della Figura 2.59). Questa impostazione permette di definire quanto tempo di inattività dell'app deve passare prima che Briar si blocchi automaticamente.

L'impostazione 'Change password' (terza voce della Figura 2.59) permette all'utente di modificare la password del proprio account.

2.8.4.1 - Sezione Panic button

In Briar è possibile utilizzare un panic button che se configurato permette o di effettuare il sign out da Briar (l'opzione Sign Out nella sezione Panic Button Setup deve essere attivata) o di cancellare l'account di Briar (l'opzione Sign Out e Delete Account nella sezione Panic Button Setup devono essere attivate).

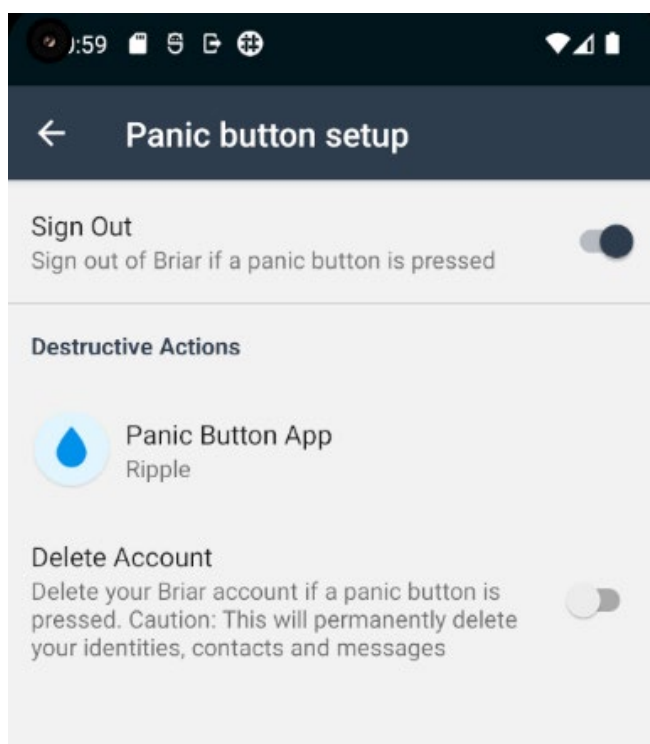


Figura 2.60: Sezione Panic button

Per poter abilitare l'opzione di cancellazione account è necessario utilizzare una app che gestisca il panic button. L'unica app di gestione del panic button compatibile con Briar è Ripple. Quest'ultima deve essere installata e configurata a parte sul dispositivo in cui è presente Briar e solo dopo ciò è possibile selezionarla come Panic Button app dentro a Briar.

Inoltre, una volta abilitata l'opzione di cancellazione account, per poterla attivare è necessario che l'opzione di sign out sia anch'essa attivata.

2.8.5 - Impostazioni Notifications

In questa sezione è possibile gestire le notifiche che si vogliono ricevere da Briar.

Più nello specifico si può:

- Scegliere se visualizzare un promemoria che ricordi all'utente di eseguire l'accesso a Briar quando il telefono si avvia o l'applicazione è stata aggiornata (Prima voce della Figura 2.61)
- Configurare le notifiche per i messaggi privati (Seconda voce della Figura 2.61)
- Configurare le notifiche per i messaggi di gruppo (Terza voce della Figura 2.61)
- Configurare le notifiche per i post dei forum (Quarta voce della Figura 2.61)
- Configurare le notifiche per i post dei blog (Quinta voce della Figura 2.61)

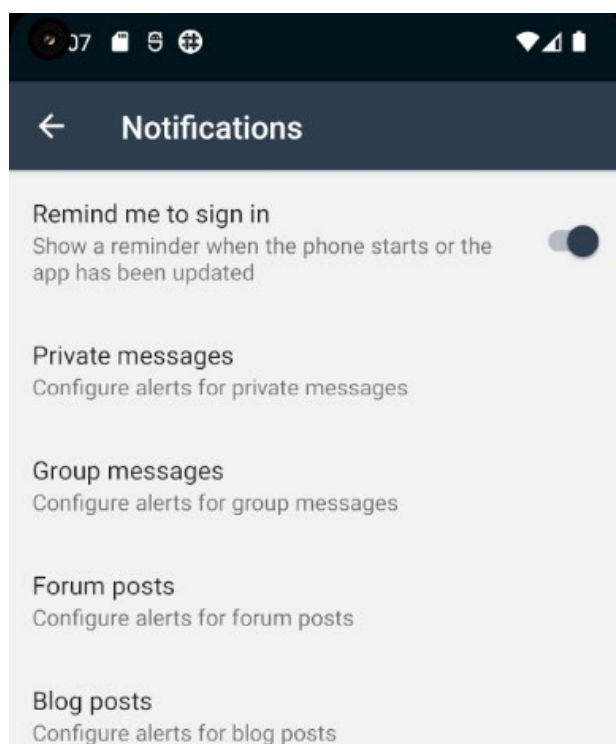


Figura 2.61: Sezione Notifications

La configurazione delle notifiche di ogni sezione in realtà utilizza la sezione Notifiche nelle Impostazioni del dispositivo per abilitarle o disabilitarle.

Capitolo 3

Organizzazione dei dati generati da Briar

Durante l'utilizzo di Briar, l'app genera molteplici file e dati, ma solo alcuni di essi posseggono un interesse forense.

In questo capitolo si individueranno questi file e questi dati andando a definire anche come decodificarli per ottenere le informazioni ricercate.

Gli artefatti forensicamente più importanti prodotti da Briar e che quindi si analizzeranno nel dettaglio sono:

- db.key: file che contiene la chiave criptata per accedere al database. È collocato nella cartella app_key in org.briarproject.briar.android (cartella dell'applicazione) (Sezione 3.2)
- db.mv.db: database h2 utilizzato dall'applicazione. È collocato nella cartella app_db in org.briarproject.briar.android (cartella dell'applicazione) (Sezione 3.3)
- File creati durante l'utilizzo della modalità di scambio messaggi via removable drive. Sono collocati in una cartella dello smartphone o del dispositivo rimuovibile scelta dall'utente in fase di salvataggio del file. (Sezione 3.4)
- File xml: sono file che contengono informazioni su preferenze o impostazioni dell'applicazione. Sono collocati nella cartella shared_prefs in org.briarproject.briar.android (cartella dell'applicazione) (Sezione 3.5)

In Figura 3.1 si riporta la struttura della cartella dell'applicazione (org.briarproject.briar.android).

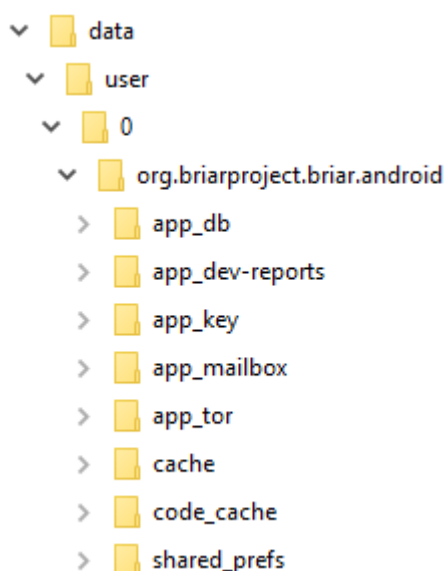


Figura 3.1: Struttura delle cartelle dove Briar salva gli artefatti

Prima di procedere con l'analisi degli artefatti d'interesse forense è necessario soffermarsi sulla descrizione di alcuni elementi fondamentali dell'architettura di Briar.

3.1 - Architettura

L'architettura di Briar si basa sui seguenti componenti:

- Protocolli
- Strutture dati
- Client

3.1.1 - Protocolli

Briar utilizza protocolli diversi in base alla tipologia di operazione richiesta.

I protocolli impiegati all'interno dell'architettura dell'applicazione sono:

- BQP (Bramble QR code Protocol) : è un protocollo di accordo sulla chiave tramite codice QR
- BHP (Bramble Handshake Protocol): è un protocollo di accordo sulla chiave per reti tolleranti ai ritardi
- BRP (Bramble Rendezvous Protocol): è un protocollo di scoperta per reti peer-to-peer
- BTP (Bramble Transport Protocol): è un protocollo di sicurezza del livello di trasporto per reti tolleranti al ritardo
- BSP (Bramble Synchronisation Protocol): un protocollo di sincronizzazione dei dati a livello applicativo per reti tolleranti al ritardo

Come si può notare in Figura 3.2, l'architettura di Briar è suddivisa in un livello applicativo e un livello di trasporto.

Il primo è a sua volta suddiviso in una parte che si occupa dell'aggiunta di contatti (Contact Establishment) e una parte che permette la comunicazione con i propri contatti (Contact Communication). La sezione di Contact Establishment contiene i protocolli BQP, BRP e BHP, mentre la parte di Contact Communication contiene il protocollo BSP.

Il livello di trasporto, invece, contiene solo il protocollo BTP che è utilizzato da tutti i protocolli del livello applicativo.

Infine, è bene notare che la procedura di aggiunta di un contatto da lontano a livello applicativo utilizza:

- il protocollo BRP per l'individuazione del contatto nella rete peer-to-peer
- Il protocollo BHP per l'aggiunta vera e propria di un contatto

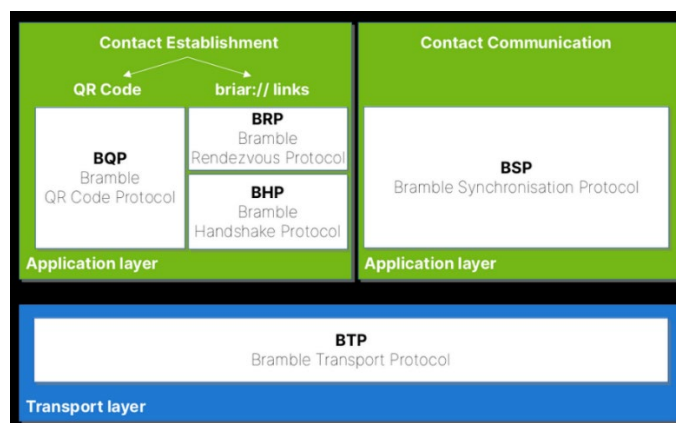


Figura 3.2: Architettura di Briar

3.1.1.1 - BQP

È un protocollo di accordo sulla chiave che stabilisce una chiave segreta condivisa tra due peer.

I peer devono essere vicini tra loro, devono avere schermi e fotocamere e devono usare un tipo di trasporto bidirezionale a corto raggio (es. Wi-Fi o Bluetooth). [\[1\]](#)

Questo protocollo è utilizzato per la comunicazione tra due peer quando quest'ultimi vogliono aggiungersi l'un l'altro con modalità di aggiunta contatti da vicino tramite QR code (Sezione 2.2.1.2).

3.1.1.2 - BHP

Il Bramble Handshake Protocol (BHP) è un protocollo di accordo sulla chiave per reti tolleranti ai ritardi. Permette a due peer che hanno precedentemente scambiato le proprie chiavi pubbliche a lungo termine di definire una chiave segreta effimera condivisa non conosciuta da nessun'altro utilizzando la connessione stabilita dal protocollo BRP. Tale chiave permette a BHP di fornire un trasporto (es. Wi-Fi, Tor o Bluetooth) che garantisca la segretezza futura (Sezione 3.1.1.4).

BHP può operare su qualsiasi protocollo di trasporto bidirezionale e orientato alla connessione. [\[2\]](#)

Questo protocollo è utilizzato per la comunicazione tra due peer quando quest'ultimi vogliono aggiungersi l'un l'altro con modalità di aggiunta contatti da lontano tramite link (Sezione 2.2.1.1).

3.1.1.3 – BRP

Il Bramble Rendezvous Protocol (BRP) è un protocollo di scoperta per reti peer-to-peer. Consente a due peer che hanno precedentemente scambiato le proprie chiavi pubbliche di connettersi tra loro senza bisogno che siano scambiate ulteriori informazioni (es. gli indirizzi di rete).

Infatti, invece di scambiarsi altre informazioni, i peer utilizzano un segreto condiviso, derivato dalle loro chiavi pubbliche e private, per generare dettagli di contatto pseudo-casuali che permettano loro di connettersi attraverso vari mezzi di trasporti.

Qualsiasi connessione stabilita dal BRP viene restituita all'applicazione chiamante, insieme a una chiave condivisa che può essere utilizzata dagli altri protocolli per garantire una comunicazione sicura tra i peer.

Il BRP è progettato per operare su protocolli di trasporto bidirezionali orientati alla connessione.

Più nello specifico, la versione attuale del BRP utilizza come trasporto il protocollo dei servizi nascosti di Tor ma le versioni future del BRP potrebbero supportare altri trasporti. [\[3\]](#)

Questo protocollo è utilizzato per effettuare la prima connessione tra due peer durante la procedura di aggiunta contatti da lontano tramite link (Sezione 2.2.1.1).

3.1.1.4 - BTP

Il Bramble Transport Protocol (BTP) è un protocollo di sicurezza a livello di trasporto per reti tolleranti ai ritardi. Si occupa di tutte le comunicazioni fornendo un canale sicuro tra i peer e garantendo riservatezza, integrità, autenticità e segretezza futura.

La segretezza futura garantisce che, una volta che due peer hanno cancellato i messaggi che si sono scambiati e il periodo di rotazione delle chiavi è trascorso, nessuno, nemmeno un

attaccante, possa decriptare i messaggi cancellati, anche se quest'ultimi sono stati precedentemente raccolti e si ottiene l'accesso al dispositivo di uno dei due peer.

Questa proprietà è ottenuta definendo una chiave radice iniziale tra i due peer e utilizzando una funzione di derivazione delle chiavi one-way per derivare una serie di chiavi temporanee dalla chiave radice iniziale. Quando entrambi i peer eliminano i messaggi che si sono scambiati e quindi viene cancellata la chiave per decriptarli, essa non può essere rigenerata nemmeno se i dispositivi dei due peer vengono successivamente compromessi.

I principali componenti di BTP sono:

- un protocollo di gestione delle chiavi basato sul tempo
- un protocollo Wire per trasportare in modo sicuro i flussi di dati (messaggi)

Il protocollo Wire di BTP include un padding opzionale e non utilizza timeout, handshake o intestazioni in chiaro. Questo rende BTP compatibile con tecniche di prevenzione dell'analisi del traffico come il traffic morphing, con l'obiettivo di rendere difficile distinguere BTP da altri protocolli.

BTP può operare su qualsiasi trasporto in grado di fornire un flusso di byte da un dispositivo all'altro su base best-effort, il che significa che i flussi possono essere ritardati, persi, riordinati o duplicati.

Il trasporto sottostante non è tenuto a fornire alcuna proprietà di sicurezza. Tuttavia, siccome BTP non fornisce anonimato, non tracciabilità e non osservabilità, questo protocollo utilizza Tor come trasporto sottostante per garantire queste proprietà.[\[4\]](#)

Protocollo di gestione chiavi

BTP usa il suo protocollo di gestione chiavi basato sul tempo per derivare un insieme di chiavi temporanee da una chiave radice iniziale. La funzione utilizzata per derivare le chiavi temporanee dalla chiave radice è basata sulla funzione pseudocasuale PRF (k,m) di cui si parlerà nella Sezione Tags del Protocollo Wire.

Per ogni trasporto, BSP divide il tempo in periodi numerati (il periodo 0 parte dall'epoca Unix) e per ogni periodo viene utilizzato un set diverso di chiavi temporanee.

Grazie all'utilizzo di questo protocollo è garantita la segretezza futura in quanto le chiavi in uscita (chiavi per la criptazione dei messaggi in uscita, cioè, inviati) per un periodo P sono cancellate alla fine del periodo P, mentre le chiavi in ingresso (chiavi per la decriptazione dei messaggi in ingresso, cioè, ricevuti) per un periodo P sono cancellate alla fine del periodo P+1.

Protocollo Wire

È di particolare interesse il protocollo Wire che compone il BTP perché è proprio questo il componente utilizzato per l'invio di messaggi compreso l'invio via removable drive (Sezione 2.3.1).

Conoscere la struttura di tale protocollo è fondamentale per la decriptazione del file generato quando un utente invia uno o più messaggi via removable drive.

Ogni stream di tale protocollo è composto da tre parti:

- un tag pseudo-casuale
- un'intestazione del flusso

- uno o più frame

Tags

Ogni flusso inizia con un tag pseudo-casuale lungo TAG_LEN byte (nella versione attuale del protocollo TAG_LEN = 16 byte).

Il destinatario calcola lo stesso tag in anticipo e lo utilizza per riconoscere da quale mittente proviene il flusso e quale chiave di intestazione in ingresso deve essere utilizzata per il flusso.

Il tag per l'i-esimo flusso da un determinato mittente a un determinato destinatario in un determinato periodo di tempo è dato dai primi TAG_LEN byte della funzione pseudocasuale $PRF(k, int_16(protocol_version) || int_64(i))$ dove k è la chiave del tag in uscita del mittente (che è anche la chiave del tag in entrata del destinatario) e il secondo argomento della funzione è la concatenazione della versione del protocollo utilizzato (attualmente 4) rappresentato come intero a 16 bit con il numero del flusso (i) rappresentato come intero a 64 bit. La funzione PRF è utilizzata per derivare le chiavi temporanee dalla chiave radice. L'attuale versione di BTP utilizza come funzione pseudocasuale BLAKE2b [5] con lunghezza dell'output uguale a 32 byte.

I flussi sono numerati a partire da zero in ogni periodo di tempo.

Stream Header

Il tag pseudo-casuale è seguito dall'intestazione del flusso, che è calcolato dall'istruzione $stream_header = nonce || ENC(outgoing_header_key, nonce, stream_header_plaintext)$ che concatena il nonce con l'output di una funzione di cifratura autenticata (ENC).

Il nonce è calcolato da un generatore pseudocasuale di numeri $R(n)$ dove n è la lunghezza dell'output in byte. Nel caso delle intestazioni di flusso si avrà $nonce = R(NONCE_LEN)$ dove NONCE_LEN è la lunghezza in byte del nonce.

La funzione $ENC(k, n, m)$ è una funzione di cifratura con tre parametri in cui il primo è la chiave utilizzata dalla funzione di cifratura, il secondo è il nonce utilizzato dalla funzione di cifratura e il terzo è la vera e propria intestazione del flusso. L'output della funzione ENC ha lunghezza uguale alla lunghezza del terzo parametro più AUTH_LEN. L'attuale versione di BTP utilizza come cifrario autenticato XSalsa20/Poly1305 che ha KEY_LEN = 32 byte, NONCE_LEN = 24 byte e AUTH_LEN = 16 byte.

Il terzo parametro della funzione ENC è calcolato dall'istruzione $stream_header_plaintext = int_16(protocol_version) || int_64(stream_number) || ephemeral_cipher_key$ che concatena la versione del protocollo, espressa come intero a 16 bit, con il numero del flusso, espresso come intero a 64 bit, concatenato a sua volta con la chiave di cifratura effimera.

Quindi in totale l'intestazione del flusso è lunga $NONCE_LEN + 2 + 8 + KEY_LEN + AUTH_LEN = 82$ byte in cui, in base agli algoritmi di cifratura che l'attuale versione di BTP usa, NONCE_LEN = 24 byte, 2 è la lunghezza in byte del *protocol_version*, 8 è la lunghezza in byte del *stream_number*, KEY_LEN = 32 byte e AUTH_LEN = 16 byte.

L'intestazione contiene una chiave di cifratura effimera utilizzata per criptare e autenticare il resto del flusso.

Il nonce casuale assicura che, anche se un numero di flusso viene accidentalmente riutilizzato, la combinazione di chiave di intestazione e nonce sarà unica, il che è un requisito di sicurezza per molti cifrari.

Frame

Il resto dello stream consiste in uno o più frame.

Ogni frame ha:

- un'intestazione a lunghezza fissa
- un corpo del frame a lunghezza variabile che può contenere dati e/o padding.

La lunghezza totale dei dati e del padding non deve superare i 988 byte, dando una lunghezza massima di 1.024 byte per un frame criptato, inclusa l'intestazione. I frame in ogni stream sono numerati a partire da zero.

Uno stream non deve contenere più di 2^{63} frame.

Intestazione del Frame

L'intestazione in chiaro del frame è lunga 4 byte, con il seguente formato (dove il bit 0 è il più significativo):

- Bit 0: flag del frame finale, impostato a uno se questo è l'ultimo frame del flusso
- Bit 0 - 15: $\text{int_16}(\text{len}(\text{data}))$
- Bit 16 - 31: $\text{int_16}(\text{len}(\text{padding}))$

Il flag del frame finale si sovrappone al bit più significativo della lunghezza dei dati, che non viene utilizzato perché i dati non devono superare 988 byte di lunghezza. Il flag del frame finale consente al destinatario di rilevare la fine del flusso senza leggere fino a EOF, la qual cosa non è possibile per tutti i trasporti su tutte le piattaforme.

Cifratura e Autenticazione

L'intestazione e il corpo di ogni frame sono criptati e autenticati separatamente utilizzando la chiave di cifratura effimera e nonce deterministici.

I nonce non sono inviati sul flusso.

Ogni nonce è lungo `NONCE_LEN` (32) byte, con il seguente formato (dove il bit 0 è il più significativo):

- Bit 0: flag dell'intestazione, impostato a uno per l'intestazione del frame o a zero per il corpo del frame
- Bit 0 - 63: $\text{int_64}(\text{frame_number})$
- Bit rimanenti: Zero

Il flag dell'intestazione si sovrappone al bit più significativo del numero di frame, che non viene utilizzato perché un flusso non deve contenere più di 2^{63} frame.

Siccome per criptare sia l'intestazione che il corpo del frame si utilizza la funzione ENC (nella versione attuale di BTP è XSalsa20), l'intestazione criptata e autenticata del frame è lunga $4 + \text{AUTH_LEN}$ (16) byte, mentre il corpo criptato e autenticato del frame ha lunghezza uguale alla lunghezza dei dati più la lunghezza del padding più AUTH_LEN (16) byte. [4]

3.1.1.5 - BSP

Il Bramble Synchronisation Protocol (BSP) è un protocollo di sincronizzazione dei dati a livello applicativo adatto a reti tolleranti ai ritardi.

BSP sincronizza i dati tra insiemi di dispositivi. Ogni dispositivo ha un insieme dinamico di peer (contatti) con cui comunica.

I dati sono organizzati in gruppi i quali sono singole istanze dei client (Sezione 3.1.3).

Ogni client è responsabile di decidere con quali peer il gruppo dovrebbe essere condiviso, cosa costituisce un messaggio valido, quali messaggi dovrebbero essere condivisi e quali messaggi dovrebbero essere eliminati. BSP effettua queste azioni per conto del client.

Ogni gruppo rappresenta un contesto di sincronizzazione indipendente contenente un grafo dei messaggi i cui nodi sono i messaggi di quel gruppo.

Se un dispositivo partecipa alla sincronizzazione dei messaggi di un gruppo, si dice che il dispositivo è un membro del gruppo.

Se due peer sincronizzano i messaggi di un gruppo tra di loro, si dice che condividono il gruppo tra di loro.

L'appartenenza al gruppo e la condivisione sono dinamiche. I dispositivi che sono membri di un gruppo non sono necessariamente contatti tra di loro, e i contatti che sono membri di un gruppo non necessariamente lo condividono tra di loro.

Ogni membro di un gruppo memorizza una copia del grafo dei messaggi, che può essere incompleta. Ogni messaggio nella copia del grafo del membro può essere condiviso o eliminato. Se un messaggio è condiviso, il dispositivo sincronizzerà il messaggio con qualsiasi contatto con cui condivide il gruppo.

Se un messaggio viene eliminato, il dispositivo eliminerà la sua copia del messaggio ma conserverà alcune informazioni sulla posizione del messaggio nel grafo. [\[6\]](#)

Formati dei Dati

Identificatori del Client

Ogni client è identificato da una stringa con namespace (es. org.briarproject.briar.messaging).

Il namespace è basato su nomi di dominio e permette di prevenire collisioni accidentali tra client creati da sviluppatori diversi. Ad esempio, tutti i client sviluppati dal progetto Briar hanno identificatori che iniziano con org.briarproject.

Versioning del Client

BSP utilizza un sistema di numerazione major/minor per distinguere tra le versioni di ciascun client (Sezione 3.1.3).

Siccome nel calcolo dell'id del gruppo sono utilizzati l'identificatore del client (namespace) e la versione major utilizzata da quel client, lo stesso client (stesso namespace) con versioni major diverse genera id del gruppo diversi, mentre lo stesso client con versioni major uguali e versioni minor diverse genera id del gruppo uguali.

Per comprendere meglio il funzionamento del versioning dei client e di come ciò influisca sulla generazione dell'id del gruppo, si prenda in considerazione il client identificato dal namespace `org.briarproject.briar.messaging`.

Se questo client (`org.briarproject.briar.messaging`) utilizza la versione major A, l'id del gruppo sarà X, mentre se lo stesso client utilizza la versione major B, l'id del gruppo sarà Y.

Se, invece questo client utilizza la versione major C e versioni minor diverse, l'id del gruppo sarà sempre lo stesso.

La versione major viene utilizzata per indicare cambiamenti non retrocompatibili.

Un client può, quindi, inviare un messaggio a un altro client con identificatore (namespace) uguale e versione major diversa anche se il client destinatario non è in grado di gestire il messaggio ricevuto. Questo perché l'id del gruppo del client mittente è diverso dall'id del gruppo del client destinatario (a causa della versione major diversa). Perciò quando il client destinatario riceve il messaggio, non trovando il gruppo che deve riceverlo, lo scarta.

La versione minor è utilizzata per indicare cambiamenti compatibili con le versioni precedenti.

Un client non può inviare un messaggio a un altro client con identificatore (namespace) uguale, versione major uguale e versione minor diversa se il client destinatario non è in grado di gestire il messaggio ricevuto. Questo perché in tal caso l'id del gruppo del client mittente è uguale all'id del gruppo del client destinatario (a causa della versione major uguale). Perciò quando il client destinatario riceve il messaggio, trovando il gruppo che deve riceverlo, lo consegna nonostante non sia in grado di gestirlo.

Gruppi

Ogni gruppo ha un identificatore unico lungo `HASH_LEN` byte. L'identificatore è calcolato computando l'hash dell'identificatore del client (`client_id`) e della versione major del client (`client_major_version`) con una stringa di byte chiamata descrittore del gruppo (`group_descriptor`):

```
group_id = HASH("org.briarproject.bramble/GROUP_ID", int_8(group_format_version), client_id, int_32(client_major_version), group_descriptor)
```

Per la versione corrente di BSP la versione del formato del gruppo è 1.

Il descrittore del gruppo è fornito dal client e BSP lo tratta come una stringa di byte opaca senza interpretarne il contenuto. La lunghezza massima di un descrittore di gruppo è 16 KiB.

Messaggi

Un messaggio consiste in una stringa di byte chiamata corpo del messaggio e un timestamp che indica il momento in cui il messaggio è stato creato, rappresentato in millisecondi dall'epoca Unix.

Il corpo del messaggio è fornito dal client.

BSP lo tratta come una stringa di byte opaca senza interpretarne il contenuto. La lunghezza massima del corpo del messaggio è 32 KiB.

Ogni messaggio ha un identificatore unico lungo HASH_LEN (32) byte.

L'identificatore è calcolato computando l'hash dell'identificatore del gruppo (*group_id*), del timestamp e del corpo del messaggio (*body_hash*).

```
message_id = HASH("org.briarproject.bramble/MESSAGE_ID", int_8(message_format_version),  
group_id, int_64(timestamp), body_hash)
```

Il quinto parametro della funzione di hash è calcolato effettuando l'hash del corpo del messaggio.

```
body_hash = HASH("org.briarproject.bramble/MESSAGE_BLOCK",  
int_8(message_format_version), message_body)
```

Per la versione corrente di BSP la versione del formato del messaggio è 1 e la funzione di hash utilizzata è Blake2b [5].

Protocollo Wire

Formato del Record

I peer sincronizzano i dati scambiando record tramite il protocollo di sicurezza del trasporto. Ogni record ha il seguente formato:

- *record_header* = *int_8(protocol_version) || int_8(record_type) || int_16(len(payload))*
- *record* = *record_header || payload*

La lunghezza massima del payload è 48 KiB.

Tipi di Record

La versione attuale del protocollo è 0 e comprende cinque tipi di record (*record_type*):

- 0: ACK - Il payload del record consiste in uno o più identificatori di messaggi. Questo record informa il destinatario che il mittente ha visto i messaggi elencati.
- 1: MESSAGE - Il payload del record consiste in un messaggio (identificatore del gruppo, timestamp e corpo del messaggio).
- 2: OFFER - Il payload del record consiste in uno o più identificatori di messaggi. Questo record informa il destinatario che il mittente ha visto i messaggi elencati, li sta condividendo con il destinatario e non sa se il destinatario li ha visti.
- 3: REQUEST - Il payload del record consiste in uno o più identificatori di messaggi. Questo record chiede al destinatario di inviare i messaggi elencati al mittente.
- 4: VERSIONS - Il payload del record consiste in un massimo di dieci interi a 8 bit, ciascuno identificante una versione di BSP supportata dal mittente. Questo record sarà utilizzato per la negoziazione della versione dalle future versioni di BSP. Un dispositivo dovrebbe rifiutare qualsiasi record con una versione del protocollo (*protocol_version*) non supportata e ignorare qualsiasi record con una versione del protocollo (*protocol_version*) supportata ma un tipo di record (*record type*) non riconosciuto. Questo permette di aggiungere nuovi tipi di record senza interrompere la compatibilità.

Sincronizzazione

Stato di sincronizzazione

Un dispositivo memorizza le seguenti informazioni relative allo stato di sincronizzazione di ogni messaggio che sta condividendo con un peer:

- Flag "Seen" - A True se si sa che il peer ha visto il messaggio, altrimenti a False.
Il valore di questo flag è salvato nella colonna SEEN della tabella STATUSES (Sezione 3.3.12)
- Flag "Ack" - A True se il peer ha offerto o inviato il messaggio da quando il dispositivo l'ha riconosciuto l'ultima volta, altrimenti a False.
Il valore di questo flag è salvato nella colonna ACK della tabella STATUSES (Sezione 3.3.12)
- Flag "Request" - A True se il peer ha richiesto il messaggio da quando il dispositivo l'ha offerto o inviato l'ultima volta, altrimenti a False.
Il valore di questo flag è salvato nella colonna REQUESTED della tabella STATUSES (Sezione 3.3.12)
- Send count - Il numero di volte in cui il messaggio è stato offerto o inviato al peer.
Il valore di questo flag è salvato nella colonna TXCOUNT della tabella STATUSES (Sezione 3.3.12)
- Send time - Il momento in cui il messaggio può essere offerto o inviato nuovamente al peer, o zero se il messaggio non è mai stato offerto o inviato al peer.
Il valore di questo flag è salvato nella colonna EXPIRY della tabella STATUSES (Sezione 3.3.12)
- Max latency - La latenza massima del trasporto che è stato utilizzato l'ultima volta per inviare il messaggio.
Il valore di questo flag è salvato nella colonna MAXLATENCY della tabella STATUSES (Sezione 3.3.12)

Il dispositivo può anche memorizzare un elenco di identificatori di messaggi che sono stati offerti dal peer e non ancora richiesti dal dispositivo. Questo elenco permette di inviare richieste in modo asincrono. La lunghezza dell'elenco può essere limitata e il peer può scartare gli identificatori di messaggi offerti quando l'elenco è pieno. [\[6\]](#)

3.1.2 - Strutture dati

La principale struttura dati utilizzata in Briar è il BDF.

3.1.2.1 - BDF (Binary Data Format)

Il formato di dati binari (BDF) è un formato di dati strutturato espresso come sequenza esadecimale.

Esso è costituito da un insieme di oggetti che possono essere di tipo primitivo (null, boolean, integer, float, string, raw) o di tipo contenitore (list, dictionary).

```
byte NULL = 0x00;
byte FALSE = 0x10;
byte TRUE = 0x11;
byte INT_8 = 0x21;
byte INT_16 = 0x22;
byte INT_32 = 0x24;
byte INT_64 = 0x28;
byte FLOAT_64 = 0x38;
byte STRING_8 = 0x41;
byte STRING_16 = 0x42;
byte STRING_32 = 0x44;
byte RAW_8 = 0x51;
byte RAW_16 = 0x52;
byte RAW_32 = 0x54;
byte LIST = 0x60;
byte DICTIONARY = 0x70;
byte END = (byte) 0x80;
```

Figura 3.3: Tipi di oggetti del BDF

Come si può notare in Figura 3.3, ogni oggetto è espresso come coppia di cifre esadecimali ognuna delle quali occupa 4 bit. La prima cifra esadecimale definisce il tipo di oggetto mentre la seconda cifra esadecimale esprime informazioni diverse in base al tipo dell'oggetto.

Ogni tipo è codificato dalla prima cifra esadecimale che può assumere valori tra 0 e 8 compresi:

- 0: Null
- 1: Boolean
- 2: Integer
- 3: Float
- 4: String
- 5: Raw
- 6: List
- 7: Dictionary
- 8: End

La seconda cifra invece può essere inutilizzata o indicare il valore dell'oggetto o la lunghezza del valore dell'oggetto o la lunghezza della lunghezza del valore dell'oggetto (Tutte le lunghezze sono misurate in byte). [\[7\]](#)

Più nello specifico:

- Se l'oggetto è di tipo Null, la seconda cifra esadecimale (4 bit) non è utilizzata.
È uguale a 0.
- Se l'oggetto è di tipo Boolean, la seconda cifra esadecimale indica il valore dell'oggetto.
Può assumere i valori 0 (False) o 1 (True)
- Se l'oggetto è di tipo Integer, la seconda cifra esadecimale indica la lunghezza del valore dell'oggetto (intero).
Può assumere i valori 1, 2, 4 o 8 i quali esprimono la lunghezza dell'intero in byte.
Si noti, per esempio, che 0x21 corrisponde all'oggetto INT_8. Questo perché la seconda cifra di 0x21 esprime la lunghezza dell'intero in byte (1) mentre 8 di INT_8 esprime la lunghezza dell'intero in bit (8). Lo stesso discorso vale per i tipi Float, String e Raw
- Se l'oggetto è di tipo Float, la seconda cifra esadecimale di 0x38 indica la lunghezza del valore dell'oggetto (float).
Assume sempre valore 8 il quale esprime la lunghezza del float in byte
- Se l'oggetto è di tipo String, la seconda cifra esadecimale esprime la lunghezza in byte del numero che esprime la lunghezza del valore dell'oggetto (stringa).
Può assumere i valori 1, 2 o 4 i quali esprimono la lunghezza della lunghezza della stringa in byte.
Si prenda in considerazione 0x41 che corrisponde all'oggetto STRING_8. La seconda cifra di 0x41 definisce quanti byte sono utilizzati per rappresentare il valore della lunghezza della stringa (1) mentre 8 di STRING_8 definisce quanti bit sono usati per esprimere il valore della lunghezza della stringa.
(es. La stringa 'prova' è lunga 5. Il valore 5 è espresso con 8 bit (1 byte) quindi l'oggetto è di tipo STRING_8 = 0x41)
- Se l'oggetto è di tipo Raw, la seconda cifra esadecimale esprime la lunghezza della lunghezza del dato binario (come per String).
Può assumere i valori 1, 2 o 4 i quali esprimono la lunghezza della lunghezza del dato binario in byte
- Se l'oggetto è di tipo List, la seconda cifra non è utilizzata.
È uguale a 0.
Questo tipo di elemento può contenere da zero a più elementi di qualsiasi tipo, ad eccezione degli elementi di tipo End, e deve terminare con un elemento di tipo End
- Se l'oggetto è di tipo Dictionary, la seconda cifra non è utilizzata.
È uguale a 0.
Questo tipo di elemento può contenere da zero a più coppie chiave-valore.
In ogni coppia, la chiave è un oggetto di tipo String e il valore è un elemento di qualsiasi tipo tranne End.
L'oggetto Dictionary deve essere concluso con un oggetto di tipo End.
- Se l'oggetto è di tipo End, la seconda cifra non è utilizzata.
È uguale a 0.
Questo tipo di elemento segna la fine di un oggetto di tipo List o Dictionary

Ogni tipo di oggetto utilizzabile in un BDF segue una struttura precisa specificata di seguito:

- Oggetto Null -> "tipo"
- Oggetto Boolean -> "tipo_valore"
- Oggetto Int -> "tipo_lunghezzaValore", "valore"
(es. INT_8, 3)

- Oggetto Float -> "tipo_lunghezzaValore", "valore"
(es. FLOAT_64, 1234567891123456)
- Oggetto String -> "tipo_lunghezzaDellaLunghezzaValore", "lunghezzaDelValore",
"carattere_1", "carattere_2", "carattere_3",....., "carattere_n"
(es. STRING_8, 5, prova)
- Oggetto Raw -> "tipo_lunghezzaDellaLunghezzaValore", "lunghezzaDelValore", "valore"
(es. RAW_8, 32,
9c115635ad663bb88387612d39b6cf9b57bda92a091626661124e655a3b70d06)
- Oggetto List -> "tipo", "oggetto1", "oggetto2", "oggetto3",, "oggetto_n"
"oggettoEnd"
(es. LIST,, END)
- Oggetto Dictionary -> "tipo", "oggetto1", "oggetto2", "oggetto3",, "oggetto_n",
"oggettoEnd"
(es. DICTIONARY,, END)
- Oggetto End -> "tipo"

Vi è da aggiungere che in questa struttura dati tutte le lunghezze sono espresse in byte, che gli interi e le lunghezze sono espressi in complemento a due big-endian, che i numeri in virgola mobile seguono lo standard IEEE 754 e che le stringhe utilizzano codifica UTF-8. [\[7\]](#)

È inoltre necessario specificare che, quando tali strutture sono memorizzate in campi del database, queste sono salvate come sequenze esadecimali.

Per comprendere meglio come si presentano e come sono utilizzati i BDF facciamo un esempio.

Prendiamo il BDF 602101410550726f766180 e lo suddividiamo in coppie di cifre ottenendo 60 21 01 41 05 50 72 6f 76 61 80.

Basandoci sulla teoria appena descritta relativa alla struttura del BDF e sulla Figura 3.3, traduciamo ogni coppia di cifre e, come riportato in Figura 3.4, otteniamo che:

- 60 si traduce con LIST
- 21 si traduce con INT_8 cioè un intero a 8 bit (1 byte)
- 01 si traduce con 1 ed è il valore dell'intero
- 41 si traduce con STRING_8 cioè una stringa la cui lunghezza è espressa con 8 bit (1 byte)
- 05 si traduce con 5 ed è la lunghezza della stringa
- 50 è il primo carattere della stringa e si traduce con 'P' (traducendo il valore 50 in carattere con codifica UTF-8)
- 72 è il secondo carattere della stringa e si traduce con 'r'
- 6f è il terzo carattere della stringa e si traduce con 'o'
- 76 è il quarto carattere della stringa e si traduce con 'v'
- 61 è il quinto carattere della stringa e si traduce con 'a'
- 80 si traduce con END e chiude l'oggetto LIST (60) presente come primo byte della sequenza esadecimale.

60	LIST
21	INT_8
01	1
41	STRING_8
05	5
50	P
72	r
6f	o
76	v
61	a
80	END

Figura 3.4: Traduzione di un BDF

Questa struttura dati è utilizzata in tutti quei campi del database in cui si ha bisogno di salvare un oggetto strutturato.

3.1.3 - Client

Come già descritto nella sezione relativa ai BSP (Sezione 3.1.1.5), i client sono componenti di Briar che inviano e ricevono messaggi e che sono identificati da un namespace con formato `org.briarproject` se sviluppati dal team di sviluppo di Briar.

I client attualmente presenti nell'applicazione sono:

- Transport Key Agreement Client (`org.briarproject.bramble.transport.agreement`)
- Transport Properties Client (`org.briarproject.bramble.properties`)
- Messaging Client (`org.briarproject.briar.messaging`)
- Forum Client (`org.briarproject.briar.forum`)
- Forum Sharing Client (`org.briarproject.briar.forum.sharing`)
- Blog Client (`org.briarproject.briar.blog`)
- Blog Sharing Client (`org.briarproject.briar.blog.sharing`)
- Private Group Client (`org.briarproject.briar.privategroup`)
- Private Group Sharing Client (`org.briarproject.briar.privategroup.invitation`)
- Introduction Client (`org.briarproject.briar.introduction`)

3.1.3.1 - Transport Key Agreement Client

Il Transport Key Agreement Client è un client BSP che stabilisce le chiavi per i trasporti aggiunti di recente [8]. È bene specificare che con il termine 'trasporto' ci si riferisce a tutti i metodi che consentono l'invio di dati, quali per esempio Wi-Fi, Tor e Bluetooth.

Il client stabilisce chiavi di trasporto con ogni contatto per qualsiasi trasporto che sia stato aggiunto più recentemente rispetto all'aggiunta del contatto.

Una sessione di accordo chiavi può essere avviata tra due peer, X e Y, nei seguenti casi:

- Nessuno dei peer supportava il trasporto quando si sono aggiunti come contatti. In questo caso, ogni peer invierà un messaggio di accordo chiavi quando aggiungerà il supporto per il trasporto. Se un peer riceve un messaggio di accordo chiavi per un

trasporto che non supporta, rimanderà la gestione del messaggio fino a quando non supporterà tale trasporto.

La sessione si completerà quando entrambi i peer supporteranno quel trasporto.

- X non supportava il trasporto quando i peer si sono aggiunti come contatti, ma Y sì. In questo caso, quando X aggiunge successivamente il supporto per il trasporto, X avvierà una sessione di accordo chiavi. Sebbene Y abbia già le chiavi per il trasporto, completerà la sessione per stabilire un nuovo set di chiavi che è posseduto anche da X.

L'identificatore del client è `org.briarproject.bramble.transport.agreement` e la sua versione principale attuale è 0.

Il client utilizza un gruppo BSP separato per comunicare con ciascun contatto.

Il descrittore del gruppo è un elenco BDF che contiene gli id univoci dei contatti, ordinati in ordine crescente come stringhe di byte.

I gruppi di questo client utilizzano due tipi di messaggi:

- 0: KEY - Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - `messageType` (int)
 - `transportId` (string)
 - `publicKey` (raw) che è una chiave pubblica X25519
- 1: ACTIVATE - Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - `messageType` (int)
 - `transportId` (string)
 - `previousMessageId` (raw) che è l'ID del messaggio di chiave del mittente in questa sessione

La Figura 3.5 raffigura la macchina degli stati del client Transport Key Agreement andando a specificare anche quali azioni portano a un cambiamento dello stato del client.

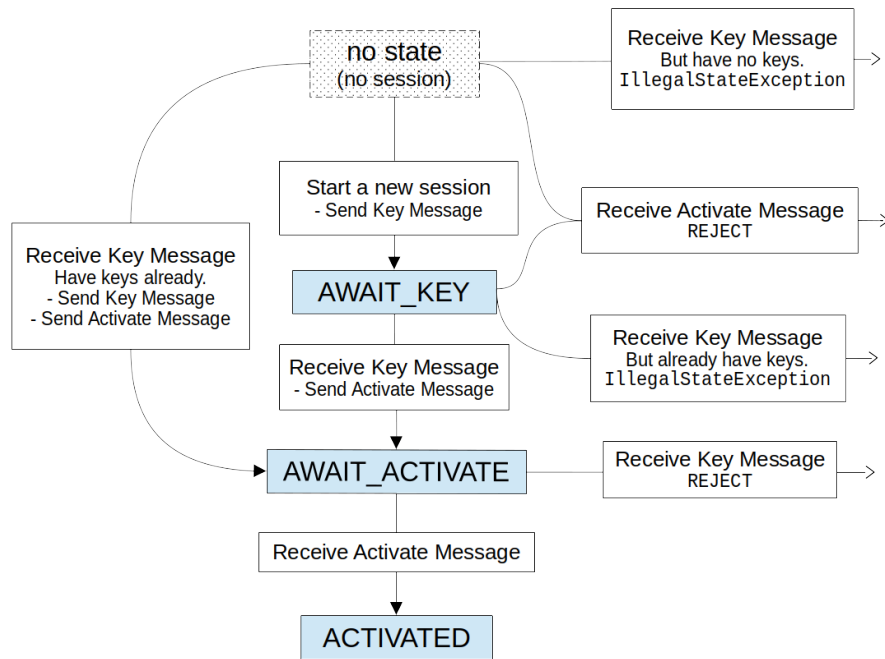


Figura 3.5: State machine di Transport Key Agreement Client

3.1.3.2 - Transport Properties Client

Il Transport Properties Client è un client BSP che sincronizza le proprietà di trasporto (es. Wi-Fi, Tor e Bluetooth) tra coppie di dispositivi. Le proprietà di trasporto descrivono come connettersi a un dispositivo tramite i vari trasporti. [9]

L'identificatore del client è org.briarproject.bramble.properties e la sua versione principale attuale è 0.

Il client utilizza un gruppo BSP separato per comunicare con ciascun contatto.

Il descrittore del gruppo è un elenco BDF contenente gli id univoci dei contatti, ordinati in ordine crescente come stringhe di byte.

Il client utilizza anche un gruppo con un descrittore vuoto per memorizzare le proprietà di trasporto locali. Tale gruppo non è condiviso con alcun contatto.

I gruppi di questo client utilizzano un solo tipo di messaggio:

- UPDATE - Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - transportId (string)
 - version (int)
 - properties (dictionary)

transportId e properties sono forniti dal plugin di trasporto.

Le chiavi e i valori di properties sono stringhe.

version è incrementata ogni volta che le proprietà cambiano.

3.1.3.3 - Messaging Client

Il Messaging Client è un client BSP che sincronizza messaggi privati tra coppie di dispositivi. [\[10\]](#)

L'identificatore del client è org.briarproject.briar.messaging e la sua versione principale attuale è 0.

Il client utilizza un gruppo BSP separato per comunicare con ciascun contatto.

Il descrittore del gruppo è un BDF composto da un oggetto di tipo List contenente gli id univoci dei contatti, ordinati in ordine crescente come stringhe di byte.

I gruppi di questo client utilizzano due tipi di messaggio:

- 0: PRIVATE_MESSAGE - Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - messageType (int)
 - text (string)
 - List:
 - Id del messaggio dell'immagine (raw) o NULL
 - formato dell'immagine (string)
 - Timer di autodelete (int)
Questo campo è valorizzato solo quando è attiva l'opzione Disappearing messages altrimenti è uguale a NULL
- 1: ATTACHMENT – Il corpo del messaggio è un BDF composto da un oggetto di tipo List con due elementi:
 - messageType (int)
 - formato dell'immagine (string)

3.1.3.4 - Forum Client

Il Forum Client è un client BSP che sincronizza i post del forum tra gruppi di dispositivi ed è utilizzato insieme al Forum Sharing Client. [\[11\]](#)

Qualsiasi utente che si iscrive a un forum può pubblicare messaggi i quali sono firmati dai loro autori.

L'identificatore del client è org.briarproject.briar.forum e la sua versione principale attuale è 0.

Ogni forum è rappresentato da un gruppo BSP separato.

Il descrittore del gruppo è un BDF composto da un oggetto di tipo List contenente due elementi: nome (string) e salt (raw). Il salt è costituito da 32 byte casuali, per prevenire collisioni tra forum con lo stesso nome.

I gruppi di questo client utilizzano un solo tipo di messaggio:

- POST - Il corpo del messaggio è un BDF composto da un oggetto di tipo List con quattro elementi:
 - parentId (raw o null)
È l'id opzionale di un post nello stesso forum a cui questo post risponde

- author (list)
 - Identifica l'autore del post ed è un oggetto di tipo List con tre elementi:
 - formatVersion (int)
 - nickname (string)
 - publicKey (raw)
- text (stringa)
- signature (raw)
 - La signature contiene un oggetto di tipo List con cinque elementi:
 - groupId (raw)
 - timestamp (int)
 - parentId (raw o null)
 - author (list)
 - text (string)

groupId e il timestamp sono presi dall'intestazione del messaggio

La chiave pubblica dell'autore viene utilizzata per verificare la firma.

L'etichetta della signature è org.briarproject.briar.forum/POST.

3.1.3.5 - Forum Sharing Client

Il Forum Sharing Client è un client BSP che consente agli utenti di condividere forum con i propri contatti, i quali possono accettare o rifiutare gli inviti ricevuti. È utilizzato insieme al Forum Client.

[\[12\]](#)

L'identificatore del client è org.briarproject.briar.forum.sharing e la sua versione principale attuale è 0.

Il client utilizza un gruppo BSP separato per comunicare con ciascun contatto.

Il descrittore del gruppo è un BDF composto da un oggetto di tipo List contenente gli id univoci dei contatti, ordinati in ordine crescente come stringhe di byte.

Tutta la comunicazione dei gruppi di questo client avviene tra due contatti, che hanno ruoli simmetrici.

I messaggi scambiati tra due contatti relativi a un dato forum costituiscono una sessione. L'id univoco del forum è utilizzato come id della sessione.

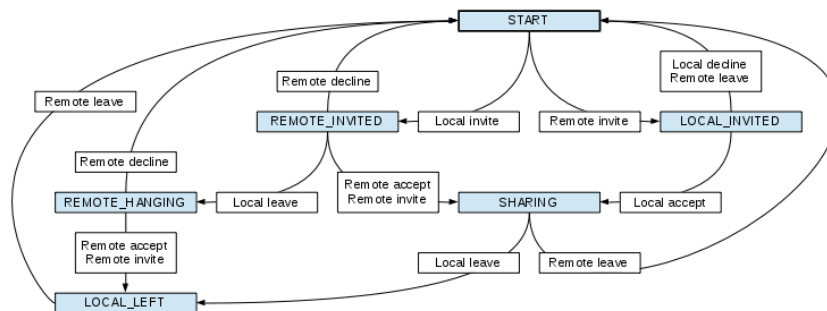


Figura 3.6: Macchina a stati Forum

Come è possibile notare in Figura 3.6, un forum può essere in uno dei seguenti stati:

- **START (0)**
In questo stato il gruppo relativo al forum (org.briarproject.briar.forum) ha visibilità **INVISIBLE**
- **LOCAL_INVITED (1)**
In questo stato il gruppo relativo al forum (org.briarproject.briar.forum) ha visibilità **INVISIBLE**
- **REMOTE_INVITED (2)**
In questo stato il gruppo relativo al forum (org.briarproject.briar.forum) ha visibilità **VISIBLE**
- **SHARING (3)**
In questo stato il gruppo relativo al forum (org.briarproject.briar.forum) ha visibilità **SHARED**
- **LOCAL_LEFT (4)**
In questo stato il gruppo relativo al forum (org.briarproject.briar.forum) ha visibilità **INVISIBLE**
- **REMOTE_HANGING (5)**
In questo stato il gruppo relativo al forum (org.briarproject.briar.forum) ha visibilità **INVISIBLE**

La visibilità di un gruppo può assumere i seguenti valori:

- **INVISIBLE:** il gruppo non è visibile
- **VISIBLE:** il gruppo è visibile e i messaggi sono stati accettati ma non inviati
- **SHARED:** il gruppo è visibile e i messaggi sono stati accettati e inviati

I gruppi di questo client utilizzano cinque tipi di messaggi:

- **0: INVITE** - Inviato da una delle due parti per avviare o riavviare una sessione.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con quattro elementi:
 - **messageType (int)**
 - **previousMessageId (raw o nullo)**
È l'ID del messaggio precedente in questa sessione, se presente
 - **descriptor (list)**
È il descrittore del forum condiviso.
L'id del forum deve essere calcolato dal descrittore, poiché è utilizzato dai messaggi successivi nella sessione.
 - **text (string o null)**
text è un messaggio opzionale dall'invitante all'invitato che accompagna l'invito.Il mittente imposta la visibilità del forum su **VISIBILE** quando invia un messaggio di invito.
- **1: ACCEPT** - Inviato in risposta a un invito per accettare la proposta.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - **messageType (int)**
 - **forumId (raw)**
 - **previousMessageId (raw)**
È l'id del messaggio precedente in questa sessione.

Il mittente imposta la visibilità del forum su SHARED quando invia un messaggio di accettazione.

Il destinatario imposta la visibilità del forum su SHARED quando riceve un messaggio di accettazione.

- 2: DECLINE - Inviato in risposta a un invito per rifiutare la proposta.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:

- messageType (int)
- forumId (raw)
- previousMessageId (raw)

È l'id del messaggio precedente in questa sessione.

Il destinatario imposta la visibilità del forum su INVISIBLE quando riceve un messaggio di rifiuto.

- 3: LEAVE - Inviato da una delle due parti quando si abbandona un forum.

Questo tipo di messaggio può essere inviato solo se la visibilità del forum è VISIBLE o SHARED.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:

- messageType (int)
- forumId (raw)
- previousMessageId (raw)

È l'id del messaggio precedente in questa sessione.

Il mittente imposta la visibilità del forum su INVISIBLE quando invia un messaggio di abbandono.

Il destinatario imposta la visibilità del forum su INVISIBLE quando riceve un messaggio di abbandono.

- 4: ABORT - Inviato da una delle due parti quando riceve un messaggio valido ma inaspettato nello stato corrente.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:

- messageType (int)
- forumId (raw)
- previousMessageId (raw)

È l'id del messaggio precedente in questa sessione.

Il mittente imposta la visibilità del forum su INVISIBLE quando invia un messaggio di abort.

Il destinatario imposta la visibilità del forum su INVISIBLE quando riceve un messaggio di abort.

3.1.3.6 - Blog Client

Il Blog Client è un client BSP che sincronizza i post del blog tra gruppi di dispositivi. Viene utilizzato insieme al Blog Sharing Client. [\[13\]](#)

Il creatore di un blog è l'unico utente che può pubblicare messaggi. I post e i commenti di altri blog possono essere ripubblicati con commenti opzionali. Un blog può consistere in post importati da un feed RSS.

L'identificatore del client è org.briarproject.briar.blog e la sua versione principale è 0.

Ogni blog è rappresentato da un gruppo BSP separato.

Il descrittore del gruppo è un BDF composto da un oggetto di tipo List con due elementi:

- author (list)
Identifica l'utente che pubblica il blog.
È un oggetto di tipo List con tre elementi:
 - formatVersion (int)
 - nickname (string)
È usato come titolo del blog
 - publicKey (raw)
- rss (boolean)
Indica se il blog contiene un feed RSS importato o un blog personale dell'utente

I gruppi di questo client utilizzano tre tipi di messaggi:

- 0: POST - Un post del blog.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - messageType (int)
 - text (string)
 - signature (raw)
La signature contiene un oggetto di tipo List con tre elementi:
 - groupId (raw)
 - timestamp (int)
 - text (string)
 groupId e il timestamp sono presi dall'intestazione del messaggio.
La chiave pubblica presa dal descrittore del gruppo viene utilizzata per verificare la signature.
L'etichetta della signature è org.briarproject.briar.blog/POST.
- 1: COMMENT - Un puntatore a un post o commento ripubblicato, con un commento opzionale.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con cinque elementi:
 - messageType (int)
 - comment (string o null)
 - parentOriginalId (raw)
È l'id di un post o commento in questo blog o in un altro blog
 - parentId (raw)
È l'id del messaggio genitore del commento, il quale può essere un post, un commento, un post ripubblicato o commento ripubblicato nel blog, e che aveva l'id parentOriginalId nel blog dove il post era stato originariamente pubblicato
 - signature (raw)
La signature contiene un oggetto di tipo List con cinque elementi:
 - groupId (raw)
 - timestamp (int)
 - comment (string o null)
 - parentOriginalId (raw)
 - parentId (raw)
 groupId e il timestamp sono presi dall'intestazione del messaggio.
La chiave pubblica presa dal descrittore del gruppo viene utilizzata per verificare la signature.
L'etichetta della firma è org.briarproject.briar.blog/COMMENT.
- 2: WRAPPED_POST - Un post ripubblicato da un altro blog.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con cinque elementi:

- messageType (int)
- copiedGroupDescriptor (list)

È il descrittore del blog dove questo post è stato originariamente pubblicato

- copiedTimestamp (int)
- copiedText (string)
- copiedSignature (raw)

La chiave pubblica presa dal descrittore del gruppo viene utilizzata per verificare la signature. L'etichetta della signature è org.briarproject.briar.blog/POST.

copiedTimestamp, copiedText e copiedSignature sono copiati dal post originale.

Il group ID originale deve essere calcolato, poiché è coperto dalla signature. L'id del messaggio originale deve anche essere calcolato, poiché è riferito dai commenti.

- 3: WRAPPED_COMMENT - Un commento ripubblicato da un altro blog.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con otto elementi:

- messageType (int)
- copiedGroupDescriptor (list)
- copiedTimestamp (int)
- copiedComment (string o null)
- copiedParentOriginalId (raw)
- copiedParentId (raw)
- copiedSignature (raw)

La chiave pubblica presa dal descrittore del gruppo viene utilizzata per verificare la signature. L'etichetta della signature è org.briarproject.briar.blog/COMMENT.

- parentId (raw)

È l'id del genitore del commento, il quale può essere un post, un commento, un post ripubblicato o un commento ripubblicato in questo blog, il quale aveva l'id copiedParentOriginalId nel blog dove il genitore è stato originariamente pubblicato e l'id copiedParentId nel blog dove questo commento è stato originariamente pubblicato.

copiedTimestamp, copiedComment, copiedParentOriginalId, copiedParentId e copiedSignature sono copiati dal commento originale.

Il group ID originale deve essere calcolato, poiché è coperto dalla signature. L'ID del messaggio originale deve anche essere calcolato, poiché è riferito dai commenti.

3.1.3.7 - Blog Sharing Client

Il Blog Sharing Client è un client BSP che consente agli utenti di condividere i blog con i loro contatti, i quali possono accettare o rifiutare gli inviti. Viene utilizzato insieme al Blog Client. [\[14\]](#)

L'identificatore del client è org.briarproject.briar.blog.sharing e la sua versione principale è 0.

Il client utilizza un gruppo BSP separato per comunicare con ciascun contatto.

Il descrittore del gruppo è un BDF composto da un oggetto di tipo List contenente gli id univoci dei contatti, ordinati in ordine crescente come stringhe di byte.

Tutte le comunicazioni avvengono tra due contatti, che hanno ruoli simmetrici.

I messaggi scambiati tra due contatti relativi a un determinato blog costituiscono una sessione. L'id univoco del blog viene utilizzato come id di sessione.

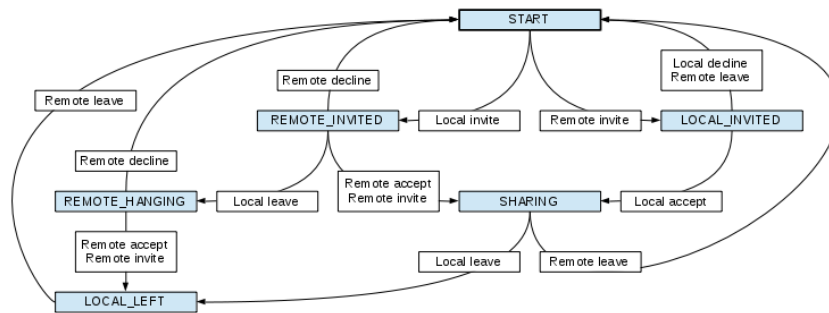


Figura 3.7: Macchina a stati Blog

Come si può notare in Figura 3.7, anche un blog, come il forum può essere in uno dei seguenti stati:

- **START (0)**
In questo stato il gruppo relativo al blog (org.briarproject.briar.blog) ha visibilità INVISIBLE
- **LOCAL_INVITED (1)**
In questo stato il gruppo relativo al blog (org.briarproject.briar.blog) ha visibilità INVISIBLE
- **REMOTE_INVITED (2)**
In questo stato il gruppo relativo al blog (org.briarproject.briar.blog) ha visibilità VISIBLE
- **SHARING (3)**
In questo stato il gruppo relativo al blog (org.briarproject.briar.blog) ha visibilità SHARED
- **LOCAL_LEFT (4)**
In questo stato il gruppo relativo al blog (org.briarproject.briar.blog) ha visibilità INVISIBLE
- **REMOTE_HANGING (5)**
In questo stato il gruppo relativo al blog (org.briarproject.briar.blog) ha visibilità INVISIBLE

Il significato delle possibili visibilità del gruppo è lo stesso illustrato nella Sezione 3.1.3.5.

I gruppi di questo client utilizzano cinque tipi di messaggio:

- **0: INVITE** - Inviato da entrambe le parti per avviare o riavviare una sessione.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con quattro elementi:
 - messageType (int)
 - previousMessageId (raw o null)
È l'id del messaggio precedente in questa sessione, se presente
 - descriptor (list)
È il descrittore del blog condiviso. L'id del blog deve essere calcolato dal descrittore, poiché viene utilizzato dai messaggi successivi nella sessione
 - text (string o null)
È un messaggio opzionale dall'invitante all'invitato che accompagna l'invito.
Il mittente imposta la visibilità del blog su VISIBLE quando invia un messaggio di invito.
- **1: ACCEPT** - Inviato in risposta a un invito per accettare la proposta.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - messageType (int)

- blogId (raw)
- previousMessageId (raw)
È l'id del messaggio precedente in questa sessione.

Il mittente imposta la visibilità del blog su SHARED quando invia un messaggio di accettazione.

Il destinatario imposta la visibilità del blog su SHARED quando riceve un messaggio di accettazione.

- 2: DECLINE - Inviato in risposta a un invito per rifiutare la proposta.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - messageType (int)
 - blogId (raw)
 - previousMessageId (raw)
È l'id del messaggio precedente in questa sessione.

L'invitante imposta la visibilità del blog su INVISIBLE quando riceve un messaggio di rifiuto.

- 3: LEAVE - Inviato da entrambe le parti quando si abbandona un blog.
Questo tipo di messaggio può essere inviato solo se la visibilità del blog è VISIBLE o SHARED.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - messageType (int)
 - blogId (raw)
 - previousMessageId (raw)
È l'id del messaggio precedente in questa sessione.

Il mittente imposta la visibilità del blog su INVISIBLE quando invia un messaggio di abbandono.

Il destinatario imposta la visibilità del blog su INVISIBLE quando riceve un messaggio di abbandono.

- 4: ABORT - Inviato da entrambe le parti quando si riceve un messaggio valido ma inaspettato nello stato corrente.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:
 - messageType (int)
 - blogId (raw)
 - previousMessageId (raw)
È l'id del messaggio precedente in questa sessione.

Il mittente imposta la visibilità del blog su INVISIBLE quando invia un messaggio di abort.

Il destinatario imposta la visibilità del blog su INVISIBLE quando riceve un messaggio di abort.

L'abort da qualsiasi stato riporta la macchina a stati nello stato START.

3.1.3.8 - Private Group Client

Il Private Group Client è un client BSP che sincronizza i messaggi tra gruppi di dispositivi. Viene utilizzato insieme al Private Group Sharing Client. [\[15\]](#)

Il creatore di ogni gruppo privato è l'unico utente che può invitare altri membri. Qualsiasi membro può inviare messaggi a un gruppo i quali sono firmati dai loro autori.

L'identificatore del client è org.briarproject.briar.privategroup e la sua versione principale è 0.

Ogni gruppo privato è rappresentato da un gruppo BSP separato.

Il descrittore del gruppo è un BDF composto da un oggetto di tipo List con tre elementi:

- creator (list)
Identifica l'utente che ha creato il gruppo privato ed è un oggetto di tipo List con tre elementi:
 - formatVersion (int)
 - nickname (string)
 - publicKey (raw)
- name (string)
- salt (raw)
È composto da 32 byte casuali, per prevenire collisioni tra gruppi privati con lo stesso creatore e nome.

I gruppi di questo client utilizzano due tipi di messaggi:

- 0: JOIN - Inviato da un nuovo membro per annunciare che si è unito al gruppo privato.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con quattro elementi:
 - messageType (int)
 - member (list)
Identifica l'utente che si è unito al gruppo privato ed è un oggetto di tipo List con tre elementi:
 - formatVersion (int)
 - nickname (string)
 - publicKey (raw)
 - invite (list o null)
È null se il membro ha creato il gruppo privato altrimenti è un oggetto di tipo List con due elementi:
 - inviteTimestamp (int)
 - inviteSignature (raw)
Contiene un oggetto di tipo List con tre elementi:
 - inviteTimestamp (int)
 - contactGroupId (raw)
È l'id del gruppo utilizzato dal creatore per gli inviti ai gruppi privati.
 - privateGroupId (raw)
Questi sono copiati dall'invito inviato dal creatore.La chiave pubblica presa dal descrittore del gruppo privato viene utilizzata per verificare inviteSignature.
L'etichetta della signature è org.briarproject.briar.privategroup.invitation/INVITE.
 - memberSignature (raw)
Contiene un oggetto di tipo List con quattro elementi:
 - privateGroupId (raw)
 - timestamp (int)
 - member (list)
 - invite (list o null)privateGroupId e il timestamp sono presi dall'intestazione del messaggio.
La chiave pubblica del membro è utilizzata per verificare memberSignature.
L'etichetta della signature è org.briarproject.briar.privategroup/JOIN.

- 1: POST - Il corpo del messaggio è un BDF composto da un oggetto di tipo List con sei elementi:
 - messageType (int)
 - member (list)
 - Identifica l'autore del messaggio ed è un oggetto di tipo List con tre elementi:
 - formatVersion (int)
 - nickname (string)
 - publicKey (raw)
 - parentId (raw o null)
 - È l'id opzionale del messaggio a cui questo messaggio risponde
 - previousMessageId (raw)
 - È l'id del messaggio precedente di join o post inviato da questo membro a questo gruppo privato
 - text (string)
 - signature (raw)
 - La signature contiene un oggetto di tipo List con sei elementi:
 - groupId (raw)
 - timestamp (int)
 - member (list)
 - parentId (raw o null)
 - previousMessageId (raw)
 - text (string)
 - groupId e timestamp sono presi dall'intestazione del messaggio.
 - La chiave pubblica del membro è utilizzata per verificare la signature.
 - L'etichetta della signature è org.briarproject.briar.privategroup/POST.

3.1.3.9 - Private Group Sharing Client

Il Private Group Sharing Client è un client BSP che consente agli utenti di condividere gruppi privati con i loro contatti, i quali possono accettare o rifiutare gli inviti. Viene utilizzato insieme a Private Group Client. [\[16\]](#)

L'identificatore del client è org.briarproject.briar.privategroup.invitation e la sua versione principale è 0.

Il client utilizza un gruppo BSP separato per comunicare con ciascun contatto.

Il descrittore del gruppo è un BDF composto da un oggetto di tipo List contenente gli id univoci dei contatti, ordinati in ordine crescente come stringhe di byte.

Tutte le comunicazioni avvengono tra due contatti, che possono assumere ruoli diversi rispetto a ciascun gruppo privato. Il contatto che ha creato il gruppo privato assume il ruolo di creatore mentre qualsiasi altro contatto assume il ruolo di invitato.

I messaggi scambiati tra due contatti relativi a un determinato gruppo privato costituiscono una sessione. L'id univoco del gruppo privato viene utilizzato come id di sessione.

Gli stati che il gruppo relativo a un gruppo privato può assumere sono diversi in base al ruolo che l'utente ha all'interno del gruppo privato.

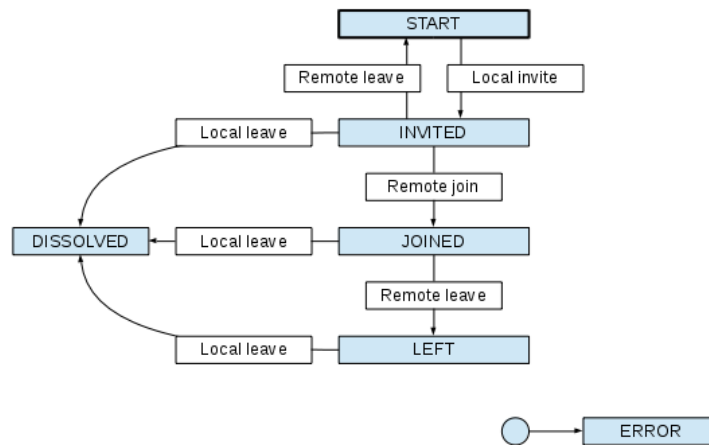


Figura 3.8: Macchina a stati Private Group (lato creatore)

Come si può osservare in Figura 3.8, se l'utente ha ruolo di creatore, il gruppo privato può assumere i seguenti stati:

- **START (0)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE
- **INVITED (1)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE
- **JOINED (2)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità SHARED
- **LEFT (3)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE
- **DISSOLVED (4)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE
- **ERROR (5)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE

Il significato delle possibili visibilità del gruppo è lo stesso illustrato nella Sezione 3.1.3.5.

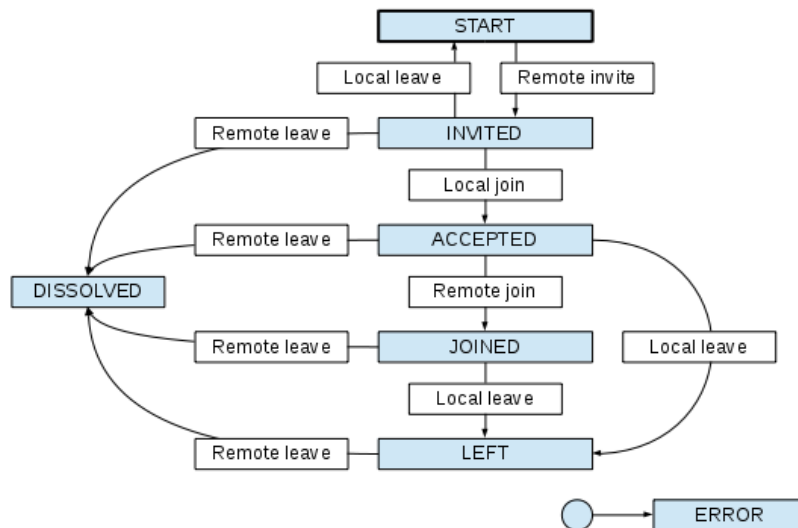


Figura 3.9: Macchina a stati Private Group (lato invitato)

Come si può osservare, invece, in Figura 3.9, se l'utente ha ruolo di invitato, il gruppo privato può assumere i seguenti stati:

- **START (0)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE
- **INVITED (1)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE
- **ACCEPTED (2)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità VISIBLE
- **JOINED (3)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità SHARED
- **LEFT (4)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE
- **DISSOLVED (5)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE
- **ERROR (6)**
In questo stato il gruppo relativo al gruppo privato (org.briarproject.briar.privategroup) ha visibilità INVISIBLE

I gruppi di questo client utilizzano quattro tipi di messaggio.

- **0: INVITE** - Inviato dal creatore all'invitato.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con sei elementi:
 - messageType (int)
 - creator (list)
È l'identità del contatto che ha inviato il messaggio

- groupName (string)
- salt (raw)
- text (string o null)

È un messaggio opzionale dal creatore all'invitato che accompagna l'invito

- signature (raw)

La signature contiene un oggetto di tipo List con tre elementi:

- timestamp (int)
- contactGroupId (raw)
- privateGroupId (raw)

timestamp e contactGroupId sono presi dall'intestazione del messaggio.

privateGroupId è calcolato a partire dal descrittore del gruppo privato.

La chiave pubblica presa dal descrittore del gruppo privato è utilizzata per validare la signature. L'etichetta della signature è

org.briarproject.briar.privategroup.invitation/INVITE.

creator, groupName e salt sono presi dal descrittore del gruppo privato.

Il creatore imposta la visibilità del gruppo privato su **VISIBLE** quando invia un messaggio di invito.

- 1: JOIN - Inviato da un invitato in risposta a un invito per accettarlo.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:

- messageType (int)
- privateGroupId (raw)
- previousMessageId (raw)

È l'id del messaggio precedente in questa sessione.

Un invitato imposta la visibilità del gruppo privato su **SHARED** quando invia un messaggio di join.

Il creatore imposta la visibilità del gruppo privato su **SHARED** quando riceve un messaggio di join.

- 2: LEAVE - Inviato da un invitato in risposta a un invito per rifiutarlo, o da un invitato quando lascia il gruppo privato, o dal creatore quando abbandona e cancella il gruppo privato.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:

- messageType (int)
- privateGroupId (raw)
- previousMessageId (raw)

È l'id del messaggio precedente in questa sessione.

Il mittente imposta la visibilità del gruppo privato su **INVISIBLE** quando invia un messaggio di leave.

Il destinatario imposta la visibilità del gruppo privato su **INVISIBLE** quando riceve un messaggio di leave.

- 3: ABORT - Inviato da entrambe le parti quando si riceve un messaggio valido ma inaspettato nello stato corrente.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con due elementi:

- messageType (int)
- privateGroupId (raw)

Il mittente imposta la visibilità del gruppo privato su **INVISIBLE** quando invia un messaggio di abort.

Il destinatario imposta la visibilità del gruppo privato su **INVISIBLE** quando riceve un messaggio di abort.

3.1.3.10 - Introduction Client

L'Introduction Client è un client BSP che consente a un utente di presentare due suoi contatti tra loro. Ogni contatto può accettare o rifiutare l'introduzione ma se entrambi i contatti accettano, diventano contatti l'uno dell'altro. [\[17\]](#)

L'identificatore del client è org.briarproject.briar.introduction e la sua versione principale è 1.

Il client utilizza un gruppo BSP separato per comunicare con ciascun contatto.

Il descrittore del gruppo è un BDF composto da un oggetto di tipo List contenente gli id univoci dei contatti, ordinati in ordine crescente come stringhe di byte.

Tutte le comunicazioni avvengono tra due contatti, che possono assumere ruoli diversi rispetto a ciascuna introduzione. Il contatto che ha avviato l'introduzione assume il ruolo di introducer. I contatti che ricevono la richiesta di introduzione assumono il ruolo di introducee. Ogni introduzione coinvolge due introducee, chiamati Alice e Bob. Alice è l'introducee il cui id univoco è inferiore, confrontando gli id come stringhe di byte.

I messaggi scambiati tra un dato introducer e una coppia (non ordinata) di introducee costituiscono una sessione.

Per l'introducer, la sessione utilizza i gruppi BSP condivisi con i due introducee mentre per ciascun introducee, la sessione utilizza il gruppo BSP condiviso con l'introducer. Gli introducee non comunicano direttamente.

L'id di sessione è dato dalla computazione dell'hash di un BDF composto da un oggetto di tipo List contenente l'id univoco dell'introducer, seguito dagli id univoci degli introducee ordinati in ordine crescente come stringhe di byte. L'etichetta dell'hash è org.briarproject.briar.introduction/SESSION_ID.

Gli stati che il gruppo relativo a un introduction può assumere sono diversi in base al ruolo che l'utente ha nell'introduction.

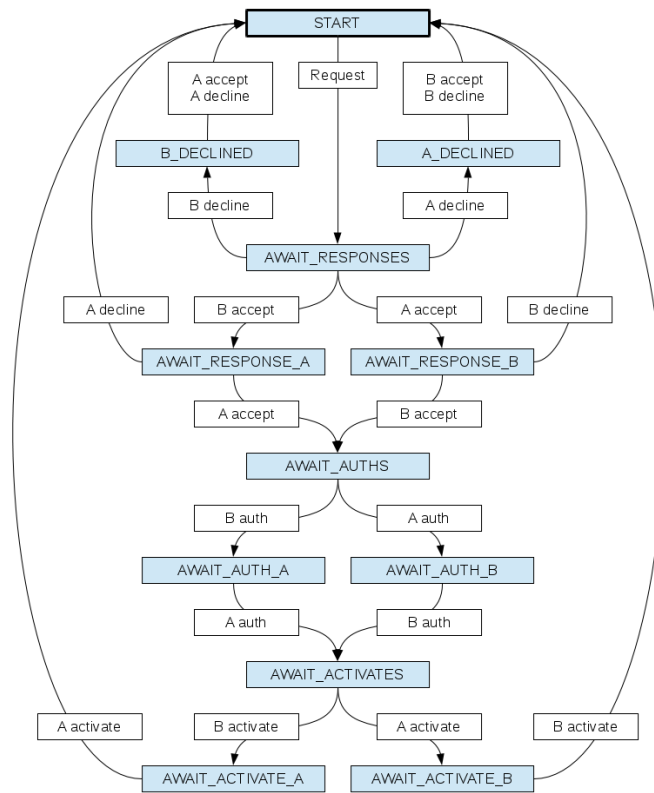


Figura 3.10: Macchina a stati Introduction (lato introducer)

Come si può notare in Figura 3.10, se l'utente è un introducer, l'introduction può assumere i seguenti stati:

- START (0)
- AWAIT_RESPONSES (1)
- AWAIT_RESPONSE_A (2)
- AWAIT_RESPONSE_B (3)
- A_DECLINED (4)
- B_DECLINED (5)
- AWAIT_AUTHS (6)
- AWAIT_AUTH_A (7)
- AWAIT_AUTH_B (8)
- AWAIT_ACTIVATES (9)
- AWAIT_ACTIVATE_A (10)
- AWAIT_ACTIVATE_B (11)

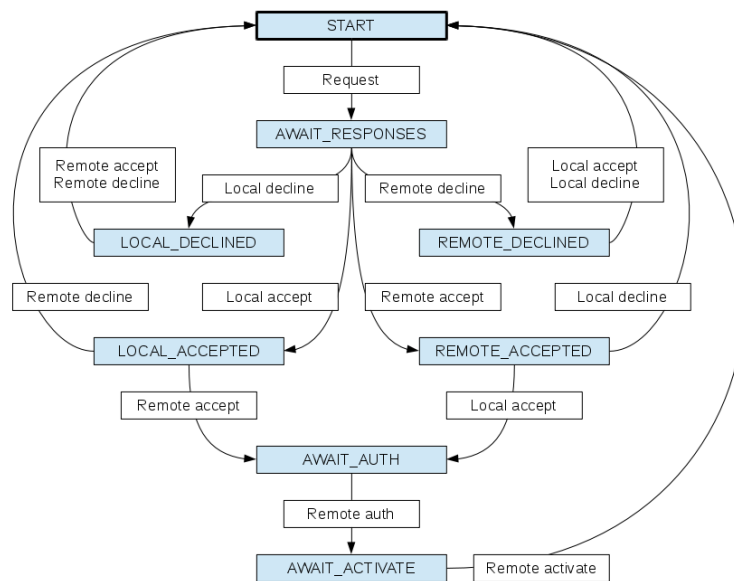


Figura 3.11: Macchina a stati Introduction (lato introducee)

Come si può notare in Figura 3.11, se l'utente è un introducee, l'introduction può assumere i seguenti stati:

- START (0)
- AWAIT_RESPONSES (1)
- LOCAL_DECLINED (2)
- REMOTE_DECLINED (3)
- LOCAL_ACCEPTED (4)
- REMOTE_ACCEPTED (5)
- AWAIT_AUTH (6)
- AWAIT_ACTIVATE (7)

A differenza degli stati dei Private Group, dei Forum e dei Blog, gli stati dell'introduction, sia se l'utente è un introducer sia se è un introducee, non impostano la visibilità del gruppo relativo all'introduction in quanto la visibilità di un gruppo relativo al client org.briarproject.briar.introduction è sempre SHARED.

I gruppi di questo client utilizzano sei tipi di messaggio:

- 0: REQUEST - Inviato dall'introducer a ciascun introducee.
Il corpo del messaggio è un BDF composto da un oggetto di tipo List con quattro elementi:
 - messageType (int)
 - previousMessageId (raw o null)
Se non è null questo campo contiene l'id univoco del messaggio precedente inviato dal mittente in questa sessione e viene utilizzato come riferimento.
 - contact (list)
È l'identità dell'altro introducee ed è un oggetto di tipo List con tre elementi:
 - formatVersion (int)
 - nickname (string)
 - publicKey (raw)
 - text (string o null)

È un messaggio opzionale dall'introducer all'introducee che accompagna l'introduzione.

- 1: ACCEPT - Inviato da un introducee all'introducer se il mittente accetta l'introduzione. È inoltrato dall'introducer all'altro introducee.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con sei elementi:

- messageType (int)
- sessionId (raw)
- previousMessageId (raw)
- ephemeralPublicKey (raw)
- timestamp (int)
- transportProperties (dictionary)

Ogni chiave del dizionario transportProperties è un id di trasporto (es. org.briarproject.bramble.lan per Wi-Fi, ecc.).

Il valore è un dizionario contenente proprietà per il rispettivo trasporto, dove le chiavi e i valori sono stringhe. Il dizionario transportProperties non può essere vuoto.

- 2: DECLINE - Inviato da un introducee all'introducer se il mittente rifiuta l'introduzione. È inoltrato dall'introducer all'altro introducee.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:

- messageType (int)
- sessionId (raw)
- previousMessageId (raw)

- 3: AUTH - Inviato da un introducee all'introducer se i messaggi di accettazione (ACCEPT) sono stati inviati e ricevuti. È inoltrato dall'introducer all'altro introducee.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con cinque elementi:

- messageType (int)
- sessionId (raw)
- previousMessageId (raw)
- mac (raw)
- signature (raw)

- 4: ACTIVATE - Inviato da un introducee all'introducer se i messaggi di autenticazione (AUTH) sono stati inviati e ricevuti. È inoltrato dall'introducer all'altro introducee.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con quattro elementi:

- messageType (int)
- sessionId (raw)
- previousMessageId (raw)
- mac (raw)

Ogni introducee elimina le chiavi MAC dopo aver inviato e ricevuto i messaggi di attivazione (ACTIVATE).

- 5: ABORT - Inviato da qualsiasi parte per interrompere la sessione.

Il corpo del messaggio è un BDF composto da un oggetto di tipo List con tre elementi:

- messageType (int)
- sessionId (raw)
- previousMessageId (raw)

Ogni introducee elimina la sua chiave privata effimera, la chiave master e le chiavi MAC prima di inviare un messaggio di abort, o dopo aver ricevuto un messaggio di abort.

3.2 - File db.key

Il db.key è un file di testo in cui l'applicazione salva la password criptata necessaria per l'accesso al database. Per ragioni di sicurezza, il contenuto di tale file è copiato in un altro file con nome db.key.bak. Entrambi i file sono salvati nella cartella app_key.

Come espresso precedentemente il contenuto del file db.key, quindi la password per accedere al database di Briar, è criptato. La procedura di decriptazione della password utilizza diversi algoritmi di cifratura che andiamo ora a descrivere brevemente.

3.2.1 - Algoritmo Scrypt

Scrypt è una funzione di derivazione di chiavi basata su password che è stata progettata per rendere più difficili gli attacchi hardware su larga scala richiedendo l'utilizzo di una grande quantità di memoria.

I ragguardevoli requisiti di memoria di Scrypt derivano dall'utilizzo all'interno dell'algoritmo di un grande vettore di stringhe di bit pseudocasuali le quali sono generate come parte dell'algoritmo.

L'algoritmo prende in input la password da criptare/decriptare e i seguenti parametri:

- salt: byte casuali generati in modo sicuro (minimo 64 bit, raccomandati 128 bit)
- N: numero di iterazioni
- r: dimensione del blocco
- p: fattore di parallelismo
- derived-key-length: lunghezza dell'output in byte [\[18\]](#)

Successivamente è generato un ampio vettore di stringhe di bit pseudocasuali i cui elementi sono acceduti in ordine pseudocasuale e combinati per produrre la chiave derivata.

In output si ottiene una chiave crittografica derivata della lunghezza definita nel parametro derived-key-length.

3.2.2 - Algoritmo HMacSHA256

HMacSHA256 è un tipo di algoritmo hash con chiave costruito dalla funzione hash SHA256 e usato come HMAC (Hash-based Message Authentication Code).

Un HMAC è un tipo specifico di codice di autenticazione del messaggio (MAC) che coinvolge una funzione crittografica di hash e una chiave crittografica segreta. L'HMAC impiegato in questo algoritmo di cifratura è utilizzato per verificare contemporaneamente l'integrità dei dati e l'autenticità di un messaggio a condizione che il mittente e il destinatario condividano una chiave privata. [\[19\]](#)

Per il calcolo di un HMAC può essere utilizzata qualsiasi funzione crittografica di hash. Nel caso dell'algoritmo HMacSHA256, l'HMAC è calcolato servendosi della funzione SHA256.

L'algoritmo HMacSHA256 prende in input una stringa su cui applicare l'algoritmo e il parametro key che contiene la chiave utilizzata dall'algoritmo.

Il mittente, quindi calcola il valore hash per i dati e invia come singolo messaggio tali dati insieme al valore hash. Il ricevitore ricalcola il valore hash nel messaggio ricevuto e verifica che l'HMAC calcolato corrisponda all'HMAC ricevuto.

Se i valori hash originali e calcolati corrispondono, il messaggio viene autenticato. [\[20\]](#)

3.2.3 - Algoritmo XSalsa20

XSalsa20 è un cifrario a flusso basato su Salsa20 ma con un nonce di 192 bit invece di 64 bit.

Salsa20 è anch'esso un cifrario a flusso basato su una funzione pseudocasuale basata su operazioni di addizione a 32 bit, rotazione dei bit e XOR.

La funzione principale di Salsa20 mappa una chiave da 256 bit, un nonce da 64 bit e un contatore da 64 bit in un blocco da 512 bit.

XSalsa20 prende in input la stringa su cui applicare l'algoritmo e i seguenti parametri:

- key: chiave da 16 o 32 byte (128 or 256 bits)
- nonce: nonce da 24 bytes (192 bit)
- counter: il contatore.
Il contatore parte da 0 all'inizio del flusso di chiavi (keystream) ed è incrementato ogni 64 byte.

Dati in input i dati e i parametri appena descritti, XSalsa20 combina i primi 128 bit del nonce con la key e genera una sottochiave.

La sottochiave appena ottenuta, insieme ai 64 bit rimanenti del nonce e al contatore, è utilizzata per generare il flusso di chiavi.

Il flusso di chiavi ricavato è infine usato per cifrare o decifrare i dati tramite le operazioni di XOR. [\[21\]](#)

3.2.4 - CyberChef

Oltre agli algoritmi di cifratura appena descritti per effettuare la decriptazione della password del database di Briar, si è utilizzato lo strumento CyberChef.

CyberChef è un'applicazione web che permette di eseguire diverse operazioni di analisi e decodifica dei dati. In questo ambito è stata utilizzata per decriptare i dati utilizzando un determinato algoritmo di cifratura.

3.2.5 - Keystore di Android

Il sistema Android Keystore consente di memorizzare chiavi crittografiche in un contenitore per renderle più difficili da estrarre dal dispositivo. Una volta che le chiavi sono nel keystore, queste possono essere utilizzate per operazioni crittografiche, mantenendo il valore della chiave non esportabile. Inoltre, il sistema keystore consente di limitare quando e come le chiavi possono essere utilizzate, ad esempio richiedendo l'autenticazione dell'utente per l'uso della chiave o limitando l'uso delle chiavi a determinati modalità crittografiche. [\[22\]](#)

Briar durante il suo utilizzo salva nel keystore una entry con chiave 'db'. Il valore di questa entry è estratto e successivamente usato dall'algoritmo HMacSHA256 durante la procedura di decriptazione della password del database.

L'estrazione del valore di una entry salvata nel keystore è possibile solo programmaticamente ricorrendo alla funzione `getEntry` della classe Java `KeyStore`. Tale funzione restituisce un oggetto `Entry` che non permette di visualizzare il valore della entry ottenuta anche se lo contiene.

3.2.6 - Processo di decriptazione della password del database

Per decriptare la password del database di Briar è necessario seguire la seguente procedura:

1. Suddividere e interpretare la stringa esadecimale contenuta nel file in questo modo:
 - A. Il primo byte è il formatversion e rappresenta la versione del formato di derivazione di chiavi basato su password che utilizza l'algoritmo Scrypt. Se formatversion vale 0, nella criptazione della password non è utilizzato un keyStrengthener (rafforzatore di chiave). Se formatversion vale 1 è utilizzato un keyStrengthener. Per la password del database di Briar il formatversion è sempre 1.
 - B. I 32 byte successivi rappresentano il valore del parametro 'salt' dell'algoritmo di cifratura Scrypt
 - C. I 4 byte successivi rappresentano il valore del parametro 'N' dell'algoritmo di cifratura Scrypt. Questo valore è espresso in esadecimale e deve essere convertito in decimale prima di poter essere utilizzato
 - D. I 24 byte successivi rappresentano il valore del parametro 'nonce' dell'algoritmo di cifratura XSalsa20
 - E. I 16 byte successivi non sono utilizzati dalla procedura di decriptazione
 - F. Gli ultimi 32 byte della stringa esadecimale sono l'input dell'algoritmo XSalsa20
2. Ottenuti tali elementi, applicare l'algoritmo di cifratura Scrypt fornendo come input la password dell'account utente di Briar e come parametri dell'algoritmo il 'salt' (punto 1.B) e 'N' (punto 1.C) individuati al punto precedente. L'output dell'algoritmo di criptazione Scrypt è una stringa esadecimale lunga 32 byte.
3. Applicare l'algoritmo HMACSHA256 fornendo come input la stringa esadecimale lunga 32 byte ottenuta al punto 2 e utilizzando il valore della entry 'db' salvata nel keystore del dispositivo. L'output di questo algoritmo è un'altra stringa esadecimale lunga 32 byte.
4. Applicare l'algoritmo XSalsa20 utilizzando come input gli ultimi 32 byte della stringa salvata nel file db.key (punto 1.F) preceduti da 32 byte di zeri (siccome la stringa in input è una stringa esadecimale 32 byte di zeri sono espressi con 64 zeri). I parametri dell'algoritmo saranno 'key' valorizzato con la stringa esadecimale da 32 byte ottenuta come output dell'applicazione dell'algoritmo HMACSHA256 (punto 3), 'nonce' valorizzato con la stringa esadecimale ottenuta dal punto 1.D e 'counter' uguale a 0. L'output di questo algoritmo è una stringa esadecimale lunga 64 byte.

I 32 byte di zeri dell'input sono necessari per il seguente motivo.

Nel codice dell'applicazione, dopo l'inizializzazione di XSalsa20, la funzione processBytes che si occupa di eseguire le operazioni dell'algoritmo XSalsa20 è richiamato due volte. La prima volta genera la sottochiave dell'algoritmo Poly1305 (algoritmo utilizzato in Briar per l'autenticazione) mentre la seconda volta decripta la password del database. Durante la prima chiamata di processBytes viene generato un flusso di chiavi che nella stessa chiamata della funzione viene totalmente utilizzato. Siccome il flusso di chiavi è stato usato completamente durante la prima chiamata, alla seconda chiamata della funzione processBytes è necessario generare un nuovo flusso di chiavi e incrementare il contatore dell'algoritmo.

Siccome la seconda chiamata di processBytes è quella d'interesse e che permette di decriptare la password del database, per far in modo che l'output dell'algoritmo XSalsa20 eseguito esternamente (in questo caso utilizzando CyberChef) sia uguale all'output

716765E1C22292716351B42B90E12402683C8BC1727C0F0DD9CEB873 ed è composto da 64 zeri (32 byte) seguiti dagli ultimi 32 byte della stringa salvata nel file db.key.

L'output di XSalsa20 è

DF970D4225F9BFDFE72B2F79E2C528F6BD99BA9DDCCFF4D50A4DDC2FBB174E363A3F48457F92D16A3D461593816643C676F3EF7B64AFDAFB5F81916531C0A04C.

5. Si estraggono gli ultimi 32 byte dell'output dell'algoritmo XSalsa20 (3A3F48457F92D16A3D461593816643C676F3EF7B64AFDAFB5F81916531C0A04C) e si ottiene la password del database che è:

3A3F48457F92D16A3D461593816643C676F3EF7B64AFDAFB5F81916531C0A04C password

Si noti che la password del database è composta dagli ultimi 32 byte dell'output dell'algoritmo XSalsa20 seguiti da uno spazio seguito a sua volta dalla parola password.

3.3 - File db.mv.db

Il file db.mv.db è il file che contiene il database h2 utilizzato da Briar. Il database è protetto da password e quindi non leggibile a meno che non si ottenga la password di accesso al database (Sezione 3.2).

Il database di Briar è composto da molteplici tabelle (Figura 3.12) ma solo alcune di esse contengono dati da cui si possono ottenere informazioni forensicamente rilevanti.

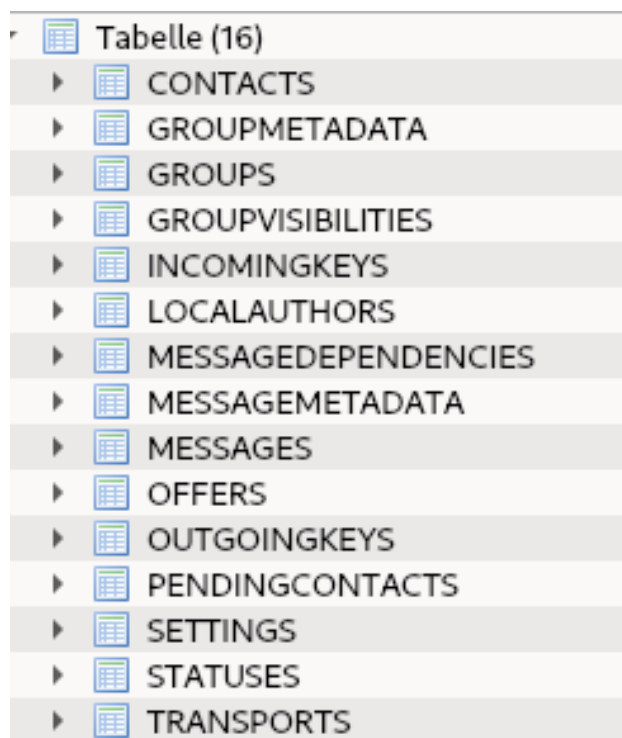


Figura 3.12: Tabelle del database

Tali tabelle sono:

- CONTACTS: contiene i dati relativi ai contatti dell'utente locale (Sezione 3.3.1)
- GROUPMETADATA: contiene i metadati di ogni gruppo di Briar (Sezione 3.3.2)
- GROUPS: contiene tutti i gruppi di Briar (Sezione 3.3.3)
- GROUPVISIBILITIES: contiene la visibilità di ogni gruppo definito in GROUPS (Sezione 3.3.4)

- INCOMINGKEYS: contiene le informazioni riguardanti le chiavi in ingresso per i vari trasporti (Sezione 3.3.5)
- LOCALAUTHORS: contiene i dati relativi all'utente locale (Sezione 3.3.6)
- MESSAGEDEPENDENCIES: contiene le dipendenze tra i messaggi se ce ne sono (Sezione 3.3.7)
- MESSAGEMETADATA: contiene i metadati relativi ai messaggi inviati e ricevuti dall'utente locale (Sezione 3.3.8)
- MESSAGES: contiene i messaggi inviati e ricevuti dall'utente locale (Sezione 3.3.9)
- OUTGOINGKEYS: contiene le informazioni riguardanti le chiavi in uscita per i vari trasporti (Sezione 3.3.10)
- PENDINGCONTACTS: contiene tutti i contatti in sospeso dell'utente locale (Sezione 3.3.11)
- STATUSES: contiene informazioni aggiuntive relative ai messaggi inviati o ricevuti dall'utente locale (Sezione 3.3.12)

Andiamo ora ad analizzarle più nel dettaglio soffermandoci nell'analisi dei campi contenuti in esse.

3.3.1 - Tabella CONTACTS

La tabella CONTACTS è la tabella che contiene tutti i contatti dell'utente.











CONTACTS	
	CONTACTID
	AUTHORID
	FORMATVERSION
	NAME
	ALIAS
	PUBLICKEY
	HANDSHAKEPUBLICKEY
	LOCALAUTHORID
	VERIFIED
	SYNCVERSIONS

Figura 3.13: Struttura tabella CONTACTS

Questa tabella, come rappresentato in Figura 3.13, è composta dai seguenti campi:

- *CONTACTID*: contiene l'id locale incrementale del contatto. Tale id identifica il contatto nel sistema locale
- *AUTHORID*: contiene l'id del contatto. Tale id identifica il contatto in modo univoco in tutto il sistema Briar
- *FORMATVERSION*
Non è stato possibile determinare l'esatto significato di questo campo.
- *NAME*: contiene il nome del contatto. Corrisponde al nome utente che il contatto ha scelto al momento della registrazione in Briar
- *ALIAS*: contiene l'alias che l'utente locale ha scelto per il contatto

- *PUBLICKEY*: contiene la chiave pubblica del contatto
- *HANDSHAKEPUBLICKEY*: contiene la chiave pubblica di handshake del contatto
- *LOCALAUTHORID*: contiene l'id globale dell'utente locale. Questo id identifica l'utente locale in modo univoco in tutto il sistema Briar.
È una chiave esterna che fa riferimento al campo *AUTHORID* della tabella *LOCALAUTHORS*.
- *VERIFIED*: è un campo booleano che vale *TRUE* se il contatto a cui fa riferimento è verificato e vale *FALSE* se il contatto non è verificato
- *SYNCVERSIONS*
Non è stato possibile determinare l'esatto significato di questo campo.

La chiave primaria di questa tabella è il campo *CONTACTID*.

3.3.2 - Tabella *GROUPMETADATA*

La tabella contiene i metadati relativi a ogni gruppo di Briar.



Figura 3.14: Struttura tabella *GROUPMETADATA*

Come raffigurato in Figura 3.14, i campi della tabella sono:

- *GROUPID*: contiene l'id del gruppo a cui il metadato fa riferimento.
È una chiave esterna che fa riferimento al campo *GROUPID* nella tabella *GROUPS*.
- *METAKEY*: contiene il nome del metadato.
- *VALUE*: contiene il valore del metadato.

La chiave primaria di questa tabella è composta dai campi *GROUPID* e *METAKEY*.

3.3.3 - Tabella *GROUPS*

La tabella contiene tutti i gruppi relativi ai diversi client di Briar.

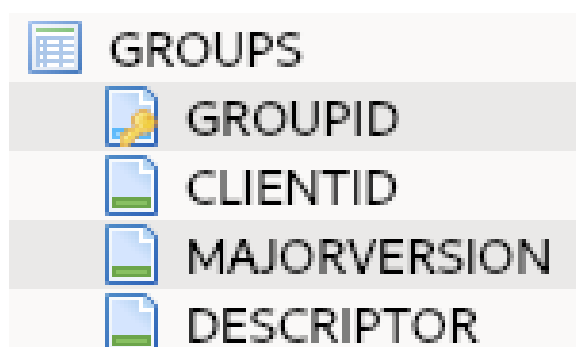


Figura 3.15: Struttura tabella *GROUPS*

Come mostrato in Figura 3.15, i campi di questa tabella sono:

- *GROUPID*: contiene l'id del gruppo
- *CLIENTID*: contiene il nome del client a cui appartiene il gruppo identificato dal campo *GROUPID*
- *MAJORVERSION*: contiene la versione major del client definito nel campo *CLIENTID* della tabella
- *DESCRIPTOR*: contiene un BDF in cui sono presenti informazioni di varia natura relative al group id specificato nel campo *GROUPID* della tabella.
Le informazioni contenute nel BDF dipendono dal client a cui ci si sta riferendo.

La chiave primaria della tabella è il campo *GROUPID*.

3.3.4 - Tabella *GROUPVISIBILITIES*

La tabella contiene la visibilità di ogni gruppo associato a un determinato contatto.



Figura 3.16: Struttura tabella *GROUPVISIBILITIES*

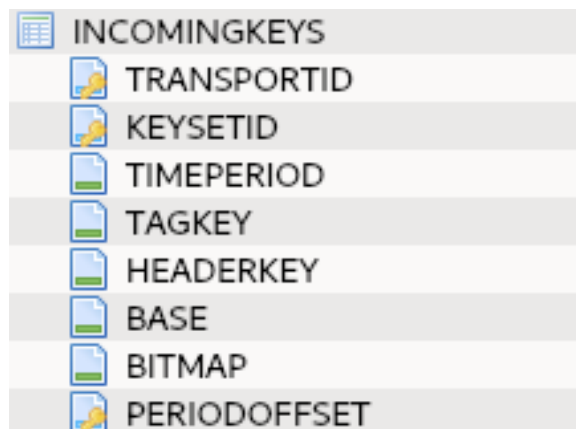
Questa tabella, come illustrato in Figura 3.16, contiene i seguenti campi:

- *CONTACTID*: contiene l'id del contatto a cui è associato l'id del gruppo.
È una chiave esterna che fa riferimento al campo *CONTACTID* nella tabella *CONTACTS*
- *GROUPID*: contiene l'id del gruppo.
È una chiave esterna che fa riferimento al campo *GROUPID* nella tabella *GROUPS*
- *SHARED*: contiene la visibilità del gruppo a cui fa riferimento.
Come descritto per i client Forum Sharing Client (Sezione 3.1.3.5) Blog Sharing Client (Sezione 3.1.3.7) e Private Sharing Client (Sezione 3.1.3.9), un gruppo può assumere tre diversi valori di visibilità:
 - *INVISIBILE*: il gruppo non è visibile
 - *VISIBILE*: il gruppo è visibile e i messaggi relativi a quel gruppo sono accettati ma non sono inviati
 - *SHARED*: il gruppo è visibile e i messaggi relativi a quel gruppo sono accettati e inviati

La chiave primaria della tabella è composta dai campi *CONTACTID* e *GROUPID*.

3.3.5 - Tabella *INCOMINGKEYS*

La tabella *INCOMINGKEYS* contiene le chiavi in ingresso usate dai vari trasporti per un dato contatto o contatto in sospeso dell'utente locale in un dato periodo di tempo (periodo precedente a quello attuale, periodo attuale e periodo successivo a quello attuale).



INCOMINGKEYS
TRANSPORTID
KEYSETID
TIMEPERIOD
TAGKEY
HEADERKEY
BASE
BITMAP
PERIODOFFSET

Figura 3.17: Struttura tabella *INCOMINGKEYS*

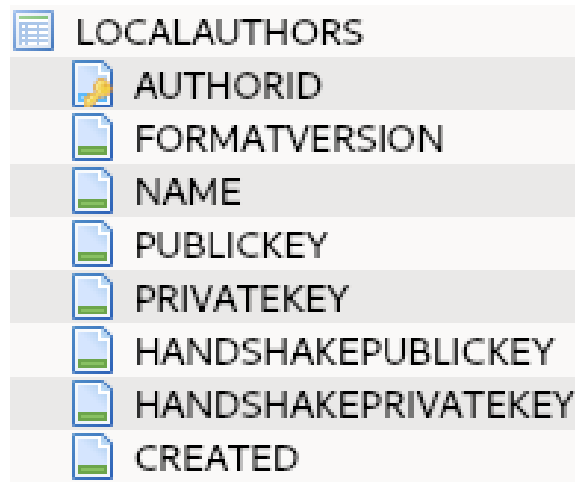
La tabella *INCOMINGKEYS*, come si può notare in Figura 3.17, è composta dai seguenti campi:

- *TRANSPORTID*: contiene l'id del trasporto (es. org.briarproject.bramble.lan per Wi-Fi).
È una chiave esterna che fa riferimento al campo *TRANSPORTID* nella tabella *TRANSPORTS*
- *KEYSETID*: contiene l'id del set di chiavi a cui appartengono la chiave del tag dello stream e la chiave dell'header dello stream.
È una chiave esterna che fa riferimento al campo *KEYSETID* nella tabella *OUTGOINGKEYS*
- *TIMEPERIOD*: contiene il periodo di tempo del set di chiavi partendo dall'epoca Unix
- *TAGKEY*: contiene la chiave del tag dello stream
- *HEADERKEY*: contiene la chiave dell'header dello stream
- *BASE*
Non è stato possibile determinare l'esatto significato di questo campo
- *BITMAP*
Non è stato possibile determinare l'esatto significato di questo campo
- *PERIODOFFET*: contiene il valore di offset del periodo di tempo.
Vale -1 per il periodo di tempo precedente a quello attuale.
Vale 0 per il periodo di tempo attuale.
Vale 1 per il periodo di tempo successivo a quello attuale.

La chiave primaria della tabella è composta dai campi *TRANSPORTID*, *KEYSETID* e *PERIODOFFSET*.

3.3.6 - Tabella LOCALAUTHORS

La tabella contiene l'account utente utilizzato sul dispositivo in cui è presente il file del database.



LOCALAUTHORS
AUTHORID
FORMATVERSION
NAME
PUBLICKEY
PRIVATEKEY
HANDSHAKEPUBLICKEY
HANDSHAKEPRIVATEKEY
CREATED

Figura 3.18: Struttura tabella LOCALAUTHORS

In Briar attualmente è possibile configurare un solo account per dispositivo quindi la tabella conterrà una sola tupla.

Questa tabella è composta dai seguenti campi (Figura 3.18):

- *AUTHORID*: contiene l'id che identifica l'utente in tutto il sistema Briar
- *FORMATVERSION*: contiene la versione di formato delle varie strutture utilizzate dall'utente locale per lo scambio dei dati
- *NAME*: contiene il nome dell'utente locale
- *PUBLICKEY*: contiene la chiave pubblica utilizzata dall'utente locale
- *PRIVATEKEY*: contiene la chiave privata utilizzata dall'utente locale
- *HANDSHAKEPUBLICKEY*: contiene la chiave pubblica di handshake utilizzata dall'utente locale
- *HANDSHAKEPRIVATE*: contiene la chiave privata di handshake utilizzata dall'utente locale
- *CREATED*: contiene il timestamp di creazione dell'account dell'utente.
Espresso in millisecondi, segue il fuso orario GMT+0.

La chiave primaria della tabella è il campo AUTHORID.

3.3.7 - Tabella MESSAGEDEPENDENCIES

La tabella contiene le dipendenze tra messaggi diversi inviati o ricevuti dall'utente locale.

Un esempio di dipendenza tra messaggi è la relazione tra il messaggio di richiesta introduction e il messaggio di risposta a tale richiesta.

Il diagramma mostra la struttura della tabella MESSAGEDEPENDENCIES con sei colonne. La prima colonna è 'MESSAGEDEPENDENCIES' con un'icona di tabella. Le altre cinque colonne sono: 'GROUPID' (icona di tabella con un'etichetta gialla), 'MESSAGEID' (icona di tabella con un'etichetta gialla), 'DEPENDENCYID' (icona di documento), 'MESSAGESTATE' (icona di documento) e 'DEPENDENCYSTATE' (icona di documento).

MESSAGEDEPENDENCIES	GROUPID	MESSAGEID	DEPENDENCYID	MESSAGESTATE	DEPENDENCYSTATE
---------------------	---------	-----------	--------------	--------------	-----------------

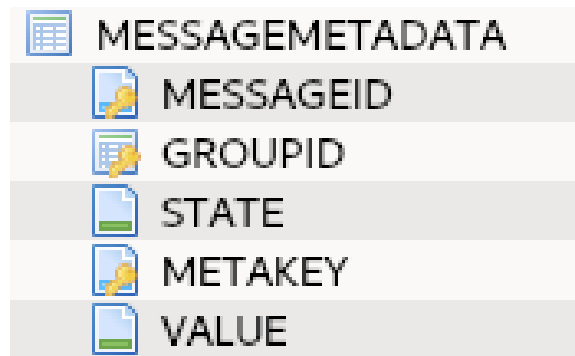
Figura 3.19: Struttura tabella MESSAGEDEPENDENCIES

Questa tabella, come rappresentato in Figura 3.19, è composta dai seguenti campi:

- **GROUPID**: contiene l'id del gruppo che ha inviato o ricevuto i due messaggi in relazione di dipendenza.
È una chiave esterna che fa riferimento al campo GROUPID nella tabella GROUPS
- **MESSAGEID**: contiene l'id del messaggio che dipende da un altro messaggio (nell'esempio dell'introduction in questo campo è contenuto l'id del messaggio di risposta alla richiesta di introduction).
È una chiave esterna che fa riferimento al campo MESSAGEID nella tabella MESSAGES
- **DEPENDENCYID**: contiene l'id del messaggio da cui MESSAGEID dipende (nell'esempio dell'introduction in questo campo è contenuto l'id del messaggio di richiesta introduction)
- **MESSAGESTATE**: contiene lo stato del messaggio con id uguale al valore contenuto nel campo MESSAGEID.
Lo stato di un messaggio può assumere i seguenti valori:
 - UNKNOWN (0): un messaggio remoto che non è stato ancora validato
 - INVALID (1): Un messaggio remoto che ha fallito la validazione
 - PENDING (2): Un messaggio remoto che ha passato la validazione ed è in attesa di essere consegnato al client locale
 - DELIVERED (3): Un messaggio locale o remoto che ha passato la validazione ed è stato consegnato al client locale
- **DEPENDENCYSTATE**: contiene lo stato del messaggio con id uguale al valore contenuto nel campo DEPENDENCYID.
Gli stati possibili di questo campo sono gli stessi descritti per il campo MESSAGESTATE.

3.3.8 - Tabella *MESSAGEMETADATA*

La tabella contiene i metadati relativi a ogni messaggio inviato o ricevuto dall'utente locale. È bene precisare che ogni messaggio ha più metadati e ogni metadato è una tupla della tabella.



MESSAGEMETADATA
MESSAGEID
GROUPID
STATE
METAKEY
VALUE

Figura 3.20: Struttura tabella *MESSAGEMETADATA*

Come si può notare dalla Figura 3.20, questa tabella contiene i seguenti campi:

- **MESSAGEID**: contiene l'id del messaggio a cui il metadato è associato.
È una chiave esterna che fa riferimento al campo MESSAGEID nella tabella MESSAGES
- **GROUPID**: contiene l'id del gruppo che ha inviato o ricevuto il messaggio.
È una chiave esterna che fa riferimento al campo GROUPID nella tabella GROUPS
- **STATE**: contiene lo stato del messaggio.
Gli stati possibili sono gli stessi descritti per il campo MESSAGESTATE nella tabella MESSAGEDEPENDENCIES.
- **METAKEY**: contiene la chiave del metadato.
È una stringa che definisce il nome del metadato
- **VALUE**: contiene il valore del metadato.
In base alla chiave del metadato questo campo può contenere un valore semplice o un BDF

La chiave primaria di questa tabella è composta dai campi MESSAGEID e METAKEY.

3.3.9 - Tabella *MESSAGES*

La tabella contiene tutti i messaggi inviati o ricevuti dai gruppi di Briar.

Il diagramma mostra la struttura della tabella *MESSAGES* con i seguenti campi:

MESSAGES
MESSAGEID
GROUPID
TIMESTAMP
STATE
SHARED
TEMPORARY
CLEANUPTIMERDURATION
CLEANUPDEADLINE
LENGTH
RAW

Figura 3.21: Struttura tabella *MESSAGES*

Come mostrato in Figura 3.21, questa tabella contiene i seguenti campi:

- *MESSAGEID*: contiene l'id univoco del messaggio
- *GROUPID*: contiene l'id del gruppo che ha ricevuto o inviato il messaggio.
È una chiave esterna che fa riferimento al campo *GROUPID* nella tabella *GROUPS*
- *TIMESTAMP*: contiene il timestamp del messaggio. Se il messaggio è stato inviato contiene il timestamp dell'invio del messaggio. Se il messaggio è stato ricevuto contiene il timestamp della ricezione del messaggio.
Espresso in millisecondi, segue l'orario GMT+0
- *STATE*: contiene lo stato del messaggio.
Gli stati possibili sono gli stessi descritti per il campo *MESSAGESTATE* nella tabella *MESSAGEDEPENDENCIES* (Sezione 3.3.7)
- *SHARED*: definisce se il messaggio è condiviso o meno.
I messaggi condivisi sono quelli inviati o ricevuti dall'utente e quindi d'interesse forense. I messaggi non condivisi sono i messaggi di servizio di Briar.
- *TEMPORARY*: definisce se il messaggio è temporaneo o meno.
Vale True solo fino a quando l'immagine che si vuole inviare non è pronta per l'invio.
- *CLEANUPTIMERDURATION*: contiene un timer, scaduto il quale è cancellato il messaggio su cui è settato il timer.
Questo campo contiene la durata del timer in millisecondi.
Questo campo è valorizzato quando l'utente locale riceve un'immagine o se l'utente locale invia o riceve un messaggio di testo (sia esso un invito o meno) con l'impostazione 'Disappearing messages' attivata.
Nel primo caso (immagine) il campo conterrà un timer con durata 28 giorni e che si ferma solo quando riceve il messaggio di testo associato all'immagine ricevuta. Se il timer scade prima di ricevere tale messaggio di testo, l'immagine ricevuta è cancellata.
Nel secondo caso (testo), il campo conterrà un timer con durata 7 giorni, scaduto il quale il messaggio è cancellato (Sezione 2.3.2).

Se l'opzione 'Disappearing messages' è attiva, i messaggi contenenti le immagini hanno questo campo non valorizzato

- **CLEANUPDEADLINE:** contiene il timestamp di scadenza del messaggio. Espresso in millisecondi, segue l'orario GMT+0. È valorizzato solo quando l'opzione di 'Disappearing messages' è abilitata
- **LENGTH:** contiene la lunghezza in byte del campo RAW
- **RAW:** contiene il messaggio.
Quando valorizzato, questo campo è composto dai seguenti valori concatenati tra loro:
 - Il valore contenuto nel campo GROUPID
 - Il timestamp del messaggio (valore contenuto nel campo TIMESTAMP) espresso in esadecimale
 - Un BDF la cui struttura dipende dal tipo di messaggio e dal client associato al gruppo che ha inviato o ricevuto il messaggio.

La chiave primaria della tabella è il campo MESSAGEID.

3.3.10 - Tabella OUTGOINGKEYS

Questa tabella contiene le chiavi in uscita usate dai vari trasporti per un dato contatto o contatto in sospenso dell'utente locale in un dato periodo di tempo.

OUTGOINGKEYS
TRANSPORTID
KEYSETID
TIMEPERIOD
CONTACTID
PENDINGCONTACTID
TAGKEY
HEADERKEY
STREAM
ACTIVE
ROOTKEY
ALICE

Figura 3.22: Struttura tabella OUTGOINGKEYS

La tabella, come mostrato in Figura 3.22, è composta dai seguenti campi:

- **TRANSPORTID:** contiene l'id del trasporto (es. org.briarproject.bramble.lan per Wi-Fi). È una chiave esterna che fa riferimento al campo TRANSPORTID nella tabella TRANSPORTS
- **KEYSETID:** contiene l'id del set di chiavi a cui appartengono la chiave del tag dello stream e la chiave dell'header dello stream
- **TIMEPERIOD:** contiene il periodo di tempo del set di chiavi partendo dall'epoca Unix
- **CONTACTID:** contiene l'id incrementale del contatto. È una chiave esterna che fa riferimento al campo CONTACTID nella tabella CONTACTS

- *PENDINGCONTACTID*: contiene l'id del contatto in sospeso.
È una chiave esterna che fa riferimento al campo *PENDINGCONTACTID* nella tabella *PENDINGCONTACTS*
- *TAGKEY*: contiene la chiave del tag dello stream
- *HEADERKEY*: contiene la chiave dell'header dello stream
- *STREAM*: contiene il numero di stream inviati su un dato trasporto
- *ACTIVE*: definisce se il set di chiavi è attivo o meno
- *ROOTKEY*: contiene la chiave radice generata dall'utente locale
- *ALICE*
Non è stato possibile determinare l'esatto significato di questo campo.

La chiave primaria della tabella è *KEYSETID*.

3.3.11 - Tabella *PENDINGCONTACTS*

La tabella contiene tutti i contatti in sospeso con cui l'utente locale si sta connettendo.

	PENDINGCONTACTS
	PENDINGCONTACTID
	PUBLICKEY
	ALIAS
	TIMESTAMP

Figura 3.23: Struttura tabella *PENDINGCONTACTS*

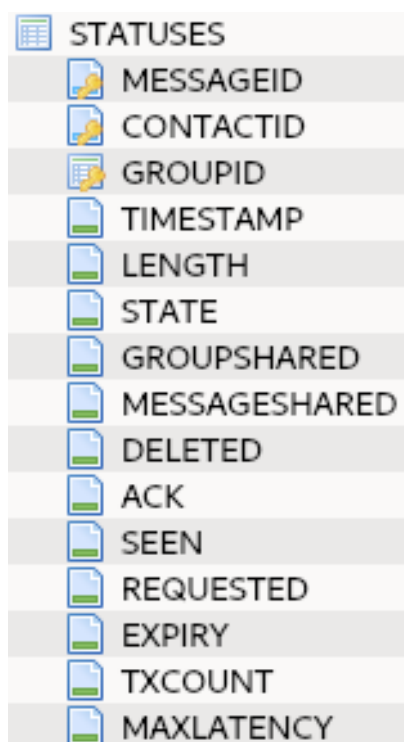
Come illustrato in Figura 3.23, questa tabella è composta dai seguenti campi:

- *PENDINGCONTACTID*: contiene l'id del contatto in sospeso.
Questo id è un id temporaneo in quanto se la connessione con il contatto va a buon fine, il contatto è cancellato da questa tabella e aggiunto nella tabella *CONTACTS*, mentre se la connessione non va a buon fine il contatto è cancellato da questa tabella
- *PUBLICKEY*: contiene la chiave pubblica utilizzata dal contatto in sospeso
- *ALIAS*: contiene il nome definito dall'utente locale per il contatto in sospeso
- *TIMESTAMP*: contiene il timestamp del momento in cui è avvenuta l'aggiunta del contatto.

La chiave primaria della tabella è il campo *PENDINGCONTACTID*.

3.3.12 - Tabella STATUSES

La tabella contiene diverse informazioni di stato per ogni messaggio inviato o ricevuto dai gruppi presenti nel database dell'utente locale.



STATUSES
MESSAGEID
CONTACTID
GROUPID
TIMESTAMP
LENGTH
STATE
GROUPSHARED
MESSAGESHARED
DELETED
ACK
SEEN
REQUESTED
EXPIRY
TXCOUNT
MAXLATENCY

Figura 3.24: Struttura tabella STATUSES

Come è possibile notare in Figura 3.24, questa tabella contiene i seguenti campi:

- **MESSAGEID**: contiene l'id del messaggio.
È una chiave esterna che fa riferimento al campo MESSAGEID nella tabella MESSAGES
- **CONTACTID**: contiene l'id del contatto da cui è stato ricevuto o inviato il messaggio.
È una chiave esterna che fa riferimento al campo CONTACTID nella tabella CONTACTS
- **GROUPID**: contiene l'id del gruppo che ha inviato o ricevuto il messaggio.
È una chiave esterna che fa riferimento al campo GROUPID nella tabella GROUPS
- **TIMESTAMP**: contiene il timestamp di invio o ricezione del messaggio.
Espresso in millisecondi, segue l'orario GMT+0
- **LENGTH**: contiene la lunghezza del messaggio.
Ha valore uguale al campo LENGTH nella tabella MESSAGES (Sezione 3.3.9)
- **STATE**: contiene lo stato del messaggio.
Ha valore uguale al campo STATE nella tabella MESSAGES (Sezione 3.3.9)
- **GROUPSHARED**: definisce se il gruppo identificato dall'id contenuto nel campo GROUPID è condiviso.
Se il gruppo è condiviso, questo campo è a True e l'id del gruppo contenuto nel GROUPID è presente nella tabella GROUPVISIBILITIES con campo SHARED uguale a True.
- **MESSAGESHARED**: definisce se il messaggio è stato condiviso quindi ricevuto o inviato da un utente remoto o meno
- **DELETED**: definisce se il messaggio è stato cancellato (valore uguale a True) o meno (valore uguale a False)

- *ACK*: definisce se il messaggio ha bisogno della conferma di ricezione da parte del contatto o meno [6]
- *SEEN*: definisce se il messaggio è stato visto dall'utente locale o meno [6]
- *REQUESTED*: definisce se il messaggio è stato richiesto o meno dall'utente locale [6]
- *EXPIRY*: contiene l'ora in cui il messaggio può essere nuovamente inviato al destinatario, oppure zero se il messaggio non è mai stato inviato o è stato ricevuto [6]
- *TXCOUNT*: contiene il numero di volte in cui il messaggio è stato inviato al destinatario, o zero se il messaggio non è mai stato inviato o è stato ricevuto [6]
- *MAXLATENCY*: contiene la latenza massima del trasporto che è stato utilizzato l'ultima volta per inviare il messaggio [6]

3.4 - File per modalità removable drive

Quando si utilizza la modalità di invio mediante removable drive è creato un file che contiene la struttura del protocollo BTP descritta nella Sezione 3.1.1.4 e cioè uno stream che a sua volta è composto da uno o più messaggi (i messaggi possono essere messaggi testuali, immagini o inviti a partecipare a gruppi privati o forum o alla condivisione di feed RSS).

Uno stream è una sequenza esadecimale composta da tre parti:

- Tag pseudo-randomico (16 byte)
- L'header dello stream (82 byte)
- Uno o più frame.
Ogni frame può contenere uno o più record.
Ogni record contiene un messaggio

Ogni frame è composto dal:

- header del frame a lunghezza fissa (20 byte)
- corpo del frame a lunghezza variabile che contiene un record per ogni messaggio del frame

Poiché ogni frame può avere una lunghezza massima di 1024 byte, di cui 20 byte sono occupati dall'header del frame e 16 byte dal MAC, si deduce che il corpo del frame può avere una lunghezza massima di 988 byte ($1024 - 20 - 16 = 988$).

Ogni frame è decriptato (sia header che body) separatamente e fino a quando non si raggiunge l'ultimo frame.

3.4.1 - Decriptazione file

La decriptazione del file consiste nella decriptazione dell'header dello stream e nella successiva decriptazione dei frame.

Il flusso per la decriptazione del file è stato interamente ottenuto attraverso l'analisi del codice sorgente di Briar.

3.4.1.1 - Decriptazione header dello stream

Per decriptare l'header dello stream è necessario seguire la seguente procedura:

1. Prendere il file generato
2. Utilizzare un visualizzatore di file esadecimale (es. hexyl)
3. Estrarre il contenuto del file

3.4.1.2 - Decriptazione frame

Una volta decrittato l'header dello stream è necessario decifrare ogni frame. La decrittazione di un frame è composta dalla decrittazione dell'header del frame e dalla successiva decrittazione del corpo del frame.

3.4.1.2.1 - Decriptazione header frame

La decrittazione dell'header del frame è eseguita nel seguente modo:

1. Si prendono i primi 20 byte del frame. Questi costituiscono l'header del frame.
2. Si decodifica il NONCE
Siccome il nonce che si decodifica è il nonce dell'algoritmo XSalsa20, questo ha lunghezza 24 byte.
Nei primi 8 byte del nonce si copia il frameNumber (parte da 0 ed è incrementato di 1 ogni volta che un frame è decrittato) e, siccome si sta trattando un header si esegue un OR tra il primo byte del nonce e 0x80.
I byte dalla posizione 9 alla posizione 24 sono invece settati a 0.
3. Si applica l'algoritmo XSalsa20 sugli ultimi 4 byte dell'header del frame (sempre preceduti da 64 zeri) utilizzando come 'key' dell'algoritmo il frameKey ottenuto al punto 5 della procedura di decrittazione dell'header dello stream e come 'nonce' dell'algoritmo il nonce ottenuto al punto precedente (2).
XSalsa20 si applica solo sugli ultimi 4 byte perché per eseguire l'algoritmo di decrittazione è richiamata la funzione process che a sua volta richiama la funzione processBytes che si occupa di eseguire le operazioni dell'algoritmo XSalsa20.
La funzione processBytes ha come parametri in input l'array di byte in input, l'offset dell'input, la lunghezza dell'input, l'array di byte in output e l'offset in output.
Quando la funzione process richiama processBytes al parametro offset dell'input sono aggiunti 16 byte ($0 + 16 = 16$) mentre al parametro lunghezza dell'input sono sottratti 16 byte ($20 - 16 = 4$).

Dalla stringa ottenuta in output dall'algoritmo XSalsa20 è possibile ottenere le seguenti informazioni:

- 3.1. Se il frame è il frame finale cioè l'ultimo
- 3.2. La lunghezza del payload cioè del body del frame
- 3.3. Il padding
4. Si controlla se il frame che si sta considerando è il frame finale.
Per controllare se il frame considerato è il frame finale si prende il primo byte dell'output dell'ultima applicazione dell'algoritmo XSalsa20 (punto 3) e lo si mette in AND con 0x80.
Se l'operazione restituisce 0x80 allora il frame è il frame finale altrimenti no.
5. Si recupera la lunghezza del payload (payloadLength) nel seguente modo:
 - 5.1. Si prende il primo byte dell'output dell'ultima applicazione di XSalsa20 (punto 3) e lo si trasforma in binario. Il numero in binario è messo in AND con 0xFF (11111111).
Il risultato dell'operazione è una rappresentazione del valore decimale senza segno.
 - 5.2. Si effettua la stessa operazione eseguita al punto precedente anche con il secondo byte dell'output
 - 5.3. Si mettono in OR i risultati ottenuti nei due punti precedenti (5.1 e 5.2) spostando a sinistra di 8 bit il risultato del primo byte (punto 5.1)

- 5.4. Il risultato del punto precedente (5.3) è messo in AND con 0x7FFF (0111111111111111 in binario).
Il risultato di quest'ultima operazione restituisce la lunghezza del payload.
6. Si recupera la lunghezza del padding dell'header (paddingLength) nel seguente modo:
 - 6.1. Si prende il terzo byte dell'output dell'ultima applicazione di XSalsa20 (punto 3) e lo si trasforma in binario. Il numero in binario è messo in AND con 0xFF (11111111).
 - Il risultato dell'operazione è una rappresentazione del valore decimale senza segno.
 - 6.2. Si fa effettuare stessa operazione eseguita al punto precedente anche con il quarto byte dell'output.
 - 6.3. Si mettono in OR i risultati ottenuti nei due punti precedenti (6.1 e 6.2) spostando a sinistra di 8 bit il risultato del terzo byte (punto 6.1)
7. Si calcola la lunghezza del frame come FRAME_HEADER_LENGTH (20) + payloadLength + paddingLength + MAC_LENGTH (16)

3.4.1.2.2 - Decrittazione corpo frame

Per effettuare la decrittazione del corpo del frame bisogna:

1. Leggere i byte del frame dal byte 20, escludendo i 20 byte dell'header del frame che sono già stati decrittati, ottenendo come lunghezza del corpo del frame, la lunghezza del frame calcolata precedentemente meno i 20 byte dell'header.
2. Decodificare il nonce esattamente come si è decodificato il nonce nell'header del frame ma in questo caso header sarà uguale a false e quindi il primo byte non deve essere messo in AND con 0x80
3. Applicare l'algoritmo XSalsa20 con 'key' uguale alla frameKey utilizzata per decodificare l'header e come 'nonce' il nonce ottenuto al punto precedente (2).
L'input dell'algoritmo partirà dall'offset 20 (la lunghezza dell'header) e ha lunghezza pari a payloadLength + paddingLength + MAC_LENGTH (16).
L'input deve essere preceduto da 64 zeri.
4. La stringa ottenuta in output contiene un record per ogni messaggio inviato via removable drive in quello stream.

Ogni record è composto dai seguenti elementi:

- Header del record è composto da 4 byte:
 - Il primo byte è il protocol version
 - Il secondo byte è il tipo di record
I record possono appartenere a uno dei seguenti tipi:
 - ACK = 0
 - MESSAGE = 1
 - OFFER = 2
 - REQUEST = 3
 - VERSIONS = 4
 - PRIORITY = 5
 - I due byte rimanenti rappresentano la lunghezza del payload.
- Il payload del record è composto da:
 - 32 byte di groupID

- 8 byte (successivi al groupID) di timestamp
- I byte restanti per raggiungere la lunghezza calcolata del payload contengono il BDF del messaggio

Per comprendere meglio il procedimento appena descritto facciamo un esempio.

Come descritto nei punti 1,2 e 3 della sezione 3.4.1.1, si estrae il contenuto del file e utilizzando un visualizzatore di esadecimali si ottiene il seguente contenuto del file

```
3480900672dc7ed672c569cfb0fc07134b48d87876af06d2486a18621c66471de7abe576e1
f9e0332abe1100b6e4086b7b1c7d077025c657b41e1491bae2465e3feafd42665bb63b387
ed1c805825233ac03931ce6542582b41346e80c75cdd1f0177c4dabd04cc939db7399568e
966aba4a890eaa706cdb6ea5c0682f344c140ce72c462c847b6a6788f9032e35f0d69a5d08
da84829cb35650449afa91bb44651a7983ffcabdebd4bb669fc6b5736013063df3f2496d2da
a382c7f04416222292e3d96a9613305146dff84cc3aadfd09f92860b25848bd7f3b7d0ee8e2
f5796666752f8ab908ef8f3d927be6651582026a806468c2c07f83a0a3b915c19a0f4325f40
9b640325047061413fe4b5ee600931ccafaca8a4d8865c45b7a60bcea7d93b03928f018041
f382308c52f26589830536a2925cadc11d15a35ab2ebb89964e29f2bda8f
```

Si divide, quindi, il contenuto del file come descritto nel punto 4 della sezione 3.4.1.1 ottenendo:

- Come tag del frame 3480900672dc7ed672c569cfb0fc0713
- Come header dello stream
4b48d87876af06d2486a18621c66471de7abe576e1f9e0332abe1100b6e4086b7b1c7d077025c657b41e1491bae2465e3feafd42665bb63b387ed1c805825233ac03931ce6542582b41346e80c75cdd1f017
- Come frame
7c4dabd04cc939db7399568e966aba4a890eaa706cdb6ea5c0682f344c140ce72c462c847b6a6788f9032e35f0d69a5d08da84829cb35650449afa91bb44651a7983ffcabdebd4bb669fc6b5736013063df3f2496d2daa382c7f04416222292e3d96a9613305146dff84cc3aadfd09f92860b25848bd7f3b7d0ee8e2f5796666752f8ab908ef8f3d927be6651582026a806468c2c07f83a0a3b915c19a0f4325f409b640325047061413fe4b5ee600931ccafaca8a4d8865c45b7a60bcea7d93b03928f018041f382308c52f26589830536a2925cadc11d15a35ab2ebb89964e29f2bda8f

Effettuando l'azione descritta nel punto 5 della Sezione 3.4.1.1, si applica l'algoritmo XSalsa20 fornendo come input

```
b41e1491bae2465e3feafd42665bb63b387ed1c805825233ac03931ce6542582b41346e80c75cdd1f017 (preceduto da 64 zeri) e usando come 'key'
253173A451C84177D3460B892AFF4BA343753D7D49A562A16E938DF2F0EF3175 e come 'nonce'
4b48d87876af06d2486a18621c66471de7abe576e1f9e033.
```

L'output dell'algoritmo è

```
00040000000000000001f02b2fe81dd5ba8b02684e77cb41253d93a8c69a916c6aec3327eb27dd72e95f2.
```

Si estraggono gli ultimi 32 byte dell'output dell'algoritmo di criptazione e si ottiene 02B2FE81DD5BA8B02684E77CB41253D93A8C69A916C6AEC3327EB27DD72E95F2, chiave che sarà utilizzata per la decriptazione del frame.

Come definito nel punto 3 della Sezione 3.4.1.2.1, per decriptare l'header del frame si applica l'algoritmo XSalsa20 usando come input gli ultimi 4 byte (890eaa70) dei primi 20 byte dell'header del frame 7c4dabd04cc939db7399568e966aba4a890eaa70 (preceduti da 64 zeri), come 'nonce' il nonce ottenuto effettuando le azioni descritte al punto 2 della Sezione 3.4.1.2.1 e come 'key' la chiave 02B2FE81DD5BA8B02684E77CB41253D93A8C69A916C6AEC3327EB27DD72E95F2.

L'output dell'algoritmo è 80c00000.

Il primo byte dell'output è messo in AND con 0x80 ottenendo come risultato 0x80 e quindi il frame considerato è il frame finale (punto 4 della Sezione 3.4.1.2.1).

Eseguendo le azioni descritte nel punto 5 della Sezione 3.4.1.2.1, per ottenere la lunghezza del payload (payloadLength) si effettuano le seguenti azioni:

- Si prende il primo byte dell'output (80), lo si traduce in binario 10000000 e lo si mette in AND con 11111111 (0xFF) ottenendo 10000000 (in decimale senza segno è 128)
- Si prende il secondo byte dell'output (c0), lo si traduce in binario 11000000 e lo si mette in AND con 11111111 (0xFF) ottenendo 11000000 (in decimale senza segno è 192)
- Si sposta a sinistra di 8 bit il numero binario che rappresenta il primo byte (11111111) e si mettono in OR i risultati dei due punti precedenti nel seguente modo:

```
10000000 00000000
|
00000000 11000000
```

ottenendo

10000000 11000000 (32960 in decimale senza segno)

- Si mette in AND il risultato del punto precedente con 0X7FFF (0111111111111111) in binario nel seguente modo:

```
10000000 11000000
&
01111111 11111111
ottenendo
```

00000000 11000000 (192 in decimale senza segno)

Quindi la lunghezza del payload (payloadLength) è 192.

Eseguendo le azioni descritte nel punto 6 della Sezione 3.4.1.2.1, per ottenere la lunghezza del padding (paddingLength) si effettuano le seguenti azioni:

- Si prende il terzo byte dell'output (00), lo si traduce in binario 00000000 e lo si mette in AND con 11111111 (0xFF) ottenendo 00000000 (in decimale senza segno è 0)
- Si prende il quarto byte dell'output (00), lo si traduce in binario 00000000 e lo si mette in AND con 11111111 (0xFF) ottenendo 00000000 (in decimale senza segno è 00)
- Si sposta a sinistra di 8 bit il numero binario che rappresenta il terzo byte e si mettono in OR i risultati dei due punti precedenti nel seguente modo:

00000000 00000000
|
00000000 00000000

ottenendo

00000000 00000000 (0 in decimale senza segno)

Quindi la lunghezza del padding (paddingLength) è 0.

Infine come descritto nel punto 7 della Sezione 3.4.1.2.1, si calcola la lunghezza del frame con la seguente formula:

$FRAME_HEADER_LENGTH (20) + payloadLength + paddingLength + MAC_LENGTH (16) = 20 + 192 + 0 + 16 = 228.$

Il corpo del frame ha lunghezza uguale alla lunghezza del frame compreso di header meno la lunghezza dell'header ($228 - 20 = 208$).

Come definito nel punto 3 della Sezione 3.4.1.2.2, per decriptare il corpo del frame, si applica l'algoritmo di XSalsa20 utilizzando come 'key' la frameKey utilizzata per decodificare l'header 02B2FE81DD5BA8B02684E77CB41253D93A8C69A916C6AEC3327EB27DD72E95F2 e come 'nonce' 00.

L'input dell'algoritmo invece è dato dai byte del frame

7c4dabd04cc939db7399568e966aba4a890eaa706cdb6ea5c0682f344c140ce72c462c847b6a6788f9032e35f0d69a5d08da84829cb35650449afa91bb44651a7983ffcabdebd4bb669fc6b5736013063df3f2496d2daa382c7f04416222292e3d96a9613305146dff84cc3aadfd09f92860b25848bd7f3b7d0ee8e2f5796666752f8ab908ef8f3d927be6651582026a806468c2c07f83a0a3b915c19a0f4325f409b640325047061413fe4b5ee600931ccafaca8a4d8865c45b7a60bcea7d93b03928f018041f382308c52f26589830536a2925cadc11d15a35ab2ebb89964e29f2bda8f

meno i byte dell'header del frame 7c4dabd04cc939db7399568e966aba4a890eaa70 e i primi 16 byte che seguono l'header del frame 6cdb6ea5c0682f344c140ce72c462c84.

Ottenendo che l'input dell'algoritmo è

7b6a6788f9032e35f0d69a5d08da84829cb35650449afa91bb44651a7983ffcabdebd4bb669fc6b5736013063df3f2496d2daa382c7f04416222292e3d96a9613305146dff84cc3aadfd09f92860b25848bd7f3b7d0ee8e2f5796666752f8ab908ef8f3d927be6651582026a806468c2c07f83a0a3b915c19a0f4325f409b640325047061413fe4b5ee600931ccafac8a4d8865c45b7a60bcea7d93b03928f018041f382308c52f26589830536a2925cadc11d15a35ab2ebb89964e29f2bda8f.

Tale input deve essere preceduto da 64 zeri.

L'output dell'algoritmo è

000400010000010035df87258f349e63def6db8949bf9e1510c5e65a526e6a4e9a96845e5824c97c45000001909b63f2b760210041044369616f608000800001007e529a16ba27d5f932e86acc68a56f83d82dbeb2d6200ddf463b0a057e8a1fef1b00000190963c54596041226f72672e627269617270726f6a6563742e6272616d626c652e626c7565746f6f7468210170410475756964412431333039383365392d376362392d336333662d393339322d3933616263313962656364638080

Interpretando l'output ottenuto come descritto nel punto 4 della Sezione 3.4.1.2.2 si ottiene

Record 1:

Header del record:

00 Protocol version
04 Record Type
0001 Lunghezza del payload (1)

Payload del record:

00

Record 2:

Header del record:

00 Protocol version
01 Record type

0035 Lunghezza del payload (53)

Payload del record:

df87258f349e63def6db8949bf9e1510c5e65a526e6a4e9a96845e5824c97c45 GroupID
000001909b63f2b7 Timestamp
60210041044369616f60800080 BDF che contiene il messaggio

Record 3:

Header del record:

00 Protocol version
01 Record Type
007e Lunghezza del payload (126)

Payload del record:

529a16ba27d5f932e86acc68a56f83d82dbeb2d6200ddf463b0a057e8a1fef1b GroupID
00000190963c5459 Timestamp
6041226f72672e627269617270726f6a6563742e6272616d626c652e626c7565746f6f7468210170
410475756964412431333039383365392d376362392d336333662d393339322d39336162633139
62656364638080 BDF che contiene il messaggio

Per comprendere meglio come tradurre il valore del timestamp si consideri un timestamp con valore in esadecimale uguale a 000001909b63f2b7. Per ottenere il valore reale del timestamp bisogna:

- Tradurre ogni byte esadecimale che compone il timestamp in binario a 8 byte
- Mettere ogni numero binario a 8 byte che compone il timestamp in OR con gli altri numeri binari a 8 byte del timestamp
- Tradurre in decimale il risultato ottenuto dall'operazione di OR

```
00 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 |
00 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 |
01 00000000 00000000 00000001 00000000 00000000 00000000 00000000 00000000 |
90 00000000 00000000 00000000 10010000 00000000 00000000 00000000 00000000 |
9b 00000000 00000000 00000000 00000000 10011011 00000000 00000000 00000000 |
```

```
63 00000000 00000000 00000000 00000000 00000000 01100011 00000000 00000000 |
f2 00000000 00000000 00000000 00000000 00000000 00000000 11110010 00000000 |
b7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 10110111 |
```

Risultato:

```
00000000 00000000 00000001 10010000 10011011 01100011 11110010 10110111
(1720593937079 in decimale).
```

Il timestamp con valore esadecimale 000001909b63f2b7 corrisponde al timestamp in decimale 1720593937079.

3.4.2 - Correlazione tra timestamp del file e del messaggio

Il file generato durante l'utilizzo della funzionalità removable drive ha un timestamp che però non ha alcuna correlazione con il momento di invio o di ricezione del messaggio.

Per comprendere meglio, vediamo il flusso di azioni che sono eseguite quando si utilizza la modalità di removable drive.

L'utente scrive un messaggio all'istante X e lo invia mediante removable drive. L'utente può inviare tale messaggio via removable drive minuti, ore o giorni dopo la scrittura del messaggio e il file che contiene il messaggio avrà timestamp Y (il momento di creazione del file).

Il mittente poi copia il file su un dispositivo rimovibile e il file presente su tale dispositivo avrà timestamp Z (il momento della copia del file sul dispositivo rimovibile).

Il destinatario del messaggio riceve il dispositivo contenente il file in un momento A (il momento della ricezione del dispositivo rimovibile), poi copia il file sul proprio smartphone in un momento B che può essere minuti, ore o giorni dopo la ricezione del dispositivo rimovibile.

Infine, il destinatario importa il file ricevuto in un momento C che può differire di minuti, ore o giorni rispetto al momento in cui ha copiato il file sul proprio smartphone (B).

Alla fine di tale flusso appena descritto, il messaggio importato avrà timestamp X cioè avrà come timestamp il momento in cui il mittente ha scritto il messaggio.

Ciò dimostra che il timestamp del file e quello del o dei messaggi inviati o ricevuti con questa modalità non sono in alcun modo correlati.

3.5 - File XML

Briar oltre a generare i dati e file visti finora, durante il suo funzionamento crea diversi file xml nella cartella shared_prefs dell'applicazione.

I file generati da Briar sono:

- db.xml: contiene le impostazioni del database.
Il file è sempre vuoto in quanto nella versione analizzata le impostazioni del database sono definite direttamente nel codice dell'applicazione.
- variant-emoji-manager.xml: memorizza le emoji che l'utente locale ha inviato nelle ultime conversazioni in cui è entrato. Se l'utente rientra in una delle conversazioni in cui ha inviato delle emoji, le emoji memorizzate nel file vengono cancellate.

Quando l'xml non è vuoto, nel file è presente un oggetto di tipo stringa con nome uguale a variant-emojis e con valore uguale alle emoji inviate concatenate dal carattere tilde.

- `_has_default_values.xml`: memorizza se Briar utilizza valori di default.
Nel file è presente un unico oggetto booleano con nome uguale a `_has_set_default_values` e valore uguale a `True`.
- `org.briarproject.briar.android_preferences`: memorizza diversi settaggi delle impostazioni di Briar.

Nel file sono salvate le seguenti informazioni:

- Il tema utilizzato dall'applicazione.
Questo è salvato nell'elemento con nome `pref_key_theme`.
Il valore di questo elemento può essere:
 - Light (tema chiaro)
 - Dark (tema scuro)
 - Auto (tema con colore in base al momento della giornata)
 - System (tema uguale al tema utilizzato dal sistema operativo del dispositivo)
- La lingua utilizzata nell'applicazione.
Questa è salvata nell'elemento con nome `pref_key_language`.
Il valore di questo elemento può essere `default`, nel caso in cui si utilizzi la stessa lingua utilizzata nel sistema operativo del dispositivo oppure il codice dello standard ISO 639-1 delle lingue
- L'impostazione che definisce se dopo un update o dopo il riavvio del dispositivo deve essere ricordato all'utente di loggarsi nell'applicazione.
Questa informazione è salvata nell'elemento con nome `pref_key_notify_sign_in`.
Il valore di questo elemento può essere `True` o `False`
- L'impostazione che definisce se all'utilizzo del panic button effettuare il sign out dell'applicazione.
Questa informazione è salvata nell'elemento con nome `pref_key_lock`.
Il valore di questo elemento può essere `True` o `False`.
È bene ricordare che, anche se questo elemento è settato a `True`, se non è configurata una applicazione per la gestione del panic button il sign out non avverrà.
- L'impostazione che definisce quale applicazione terza gestisce il panic button (in Briar l'unica applicazione di gestione del panic button compatibile è Ripple).
Questa informazione è salvata nell'elemento con nome `pref_key_panic_app`.
Se non vi è alcuna applicazione di gestione panic button installata questo elemento ha valore `NONE`.
- L'impostazione che definisce se l'utilizzo del panic button esegue la cancellazione dell'account.
Questa informazione è salvata nell'elemento con nome `pref_key_purge`.
Il valore di questo elemento può essere `True` o `False`.
È fondamentale sapere che se questo elemento è attivato allora deve esserlo anche l'elemento `pref_key_lock`

Di seguito si riporta un esempio di file `org.briarproject.briar.android_preference` completo.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="pref_key_theme">light</string>
  <string name="pref_key_language">default</string>
  <boolean name="pref_key_notify_sign_in" value="true" />
  <boolean name="pref_key_lock" value="true" />
  <string name="pref_key_panic_app">NONE</string>
  <boolean name="pref_key_purge" value="false" />
</map>
```

Figura 3.25: Esempio contenuto file org.briarproject.briar.android_preferences

Capitolo 4

Analisi forense di Briar

L'analisi di una app si basa sulle seguenti fasi:

- La definizione degli esperimenti delle diverse funzionalità individuate nel capitolo 2
- L'individuazione dei dati generati nel database dell'app e nel dispositivo durante gli esperimenti
- L'analisi vera e propria di tali artefatti

È bene specificare che gli esperimenti effettuati possono essere suddivisi nei seguenti macrogruppi, ognuno dei quali con un obiettivo specifico:

- Esperimenti relativi all'account utente (Sezione 4.1)
 - L'obiettivo è determinare l'account Briar usato sullo smartphone
- Esperimenti relativi ai contatti (Sezioni 4.2, 4.3, 4.4 e 4.5)
 - L'obiettivo è la ricostruzione della lista contatti dell'utente e le operazioni eseguite sulla lista dall'utente
- Esperimenti relativi ai dialoghi (Sezioni 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20 e 4.21)
 - Gli obiettivi sono la ricostruzione dei dialoghi di cui fa parte l'utente e la definizione del tipo di dialogo, del ruolo dell'utente nel dialogo e la data di creazione del dialogo
- Esperimenti relativi allo scambio di messaggi (Sezioni 4.6, 4.7, 4.8 e 4.9)
 - Gli obiettivi sono la ricostruzione della cronologia e del contenuto dei messaggi testuali e non testuali (immagini)
- Esperimenti relativi alle Impostazioni dell'applicazione (Sezioni 4.22 e 4.23)
 - L'obiettivo è il recupero delle immagini del profilo dell'utente locale e dei suoi contatti oltre alle impostazioni di configurazione del panic button

Durante gli esperimenti appena descritti sono state effettuate azioni di cancellazione che hanno l'obiettivo di determinare se è possibile recuperare le informazioni cancellate.

L'analisi forense degli artefatti ottenuti a seguito degli esperimenti è stata eseguita utilizzando i seguenti strumenti:

- il programma AnForA (Android Forensics Automator), uno strumento sviluppato dall'Università del Piemonte Orientale che automatizza le parti più ripetitive dell'analisi e che permette di ottenere risultati più riproducibili simulando l'interazione di un'utente con l'applicazione analizzata estraendo i file necessari per la successiva analisi. Ogni esperimento in AnForA è composto da un file di configurazione (.yaml) e uno o più sotto-esperimenti, ognuno dei quali ha un file delle azioni (csv) associato. Ogni file delle azioni è composto da un insieme di istruzioni che simulano le azioni dell'utente, al fine di ottenere gli artefatti di interesse per l'analisi. Nell'ambito dell'analisi forense di Briar, si è deciso che ogni sotto-esperimento contenesse tutte le azioni necessarie per utilizzare una specifica funzionalità di Briar. [\[23\]](#)

Per esempio, in Tabella 4.1, il sotto-esperimento “A si registra in Briar” conterrà tutte le azioni che devono essere effettuate per l’utente A si registri in Briar.

- Una versione leggermente modificata di Briar v 1.5.11.
La modifica è consistita nell’aggiunta all’interno del codice sorgente del metodo getH2Script che esegue la query ‘SCRIPT TO nome_file_sql’. Questa query, eseguita al sign out dell’applicazione, permette di creare un file di tipo SQL che contenga la struttura e i dati presenti nel database h2. Siccome il metodo getH2Script è richiamato dal metodo Close e quest’ultimo è a sua volta richiamato al sign out dell’applicazione, per garantire che il file SQL sia generato è necessario che, come ultima azione di ogni sotto esperimento, sia eseguito il sign out dall’applicazione.
Una volta ottenuto il file SQL, il contenuto di quest’ultimo è stato tradotto dal dialetto SQL utilizzato nei database H2 al dialetto SQL usato nei database SQLite. Lo script per SQLite è stato infine eseguito su un database SQLite creato appositamente.
Tale modifica è stata necessaria perché AnForA utilizza Diffoscope per confrontare le differenze tra i file presenti nella cartella dell’applicazione prima e dopo un esperimento e tale strumento nell’ambito dei database più conosciuti permette di gestire correttamente solo SQLite.
- Due o più emulatori di dispositivi Android (La quantità di emulatori utilizzata dipende dall’esperimento da eseguire).
Un emulatore che rappresenta il dispositivo su cui si esegue l’analisi e su cui AnForA simula l’interazione dell’utente con l’applicazione e uno o più emulatori che comunicano con l’emulatore oggetto d’analisi.
Tutti gli emulatori utilizzati hanno comunque le seguenti caratteristiche:
 - Pixel 5 6.0 1080x2340 440dpi
Il modello del dispositivo virtuale è Pixel 5 con schermo da 6 pollici, risoluzione dello schermo da 1080x2340 e densità dei pixel dello schermo uguale a 440 dpi
 - API Level 31 Google APIs Android 12.0 x86_64
L’emulatore utilizza l’API Level 31 che corrisponde ad Android 12, include i Google APIs (Application Programming Interfaces), utilizza la versione 12 di Android e l’architettura del processore è x86_64

Andiamo ora a considerare separatamente ogni funzionalità e per ognuna di esse a specificare gli esperimenti eseguiti compresi i file delle azioni utilizzati e gli artefatti generati.

È necessario precisare che negli esperimenti elencati in tutto il capitolo 4 l’utente A è l’utente oggetto d’analisi, mentre l’utente B e C sono utenti remoti non soggetti di studio.

Problematiche riscontrate

Durante il lavoro svolto sono state riscontrate diverse difficoltà legate al funzionamento stesso di AnForA o ai software o framework in esso utilizzati o causate dall’utilizzo di emulatori.

Peer-to-peer e AnForA

Wipe dei dati

La prima problematica legata al funzionamento stesso di AnForA è l’utilizzo del wipe dei dati all’avvio di un esperimento. Infatti, siccome tale strumento è stato concepito per applicazioni client-server in cui i dati sono salvati su un server centrale, per garantire che l’emulatore su cui AnForA esegue gli esperimenti parta sempre da una situazione ripristinata, il programma esegue un wipe dei dati dell’emulatore. Questa scelta, assolutamente corretta in ambito applicazioni

client-server, è risultata una grossa problematica per applicazione peer-to-peer come Briar in cui i dati sono salvati solo sui dispositivi degli utenti. Infatti, a causa di ciò, all'avvio di ogni esperimento l'utenza di Briar precedentemente creata veniva cancellata.

Per risolvere tale problematica, per ogni esperimento è stato introdotto un sotto-esperimento iniziale che si occupasse di eseguire tutte le azioni necessarie perché l'emulatore si trovasse nella situazione corretta di partenza.

Ogni esperimento eseguito è stato strutturato come segue:

- Un sotto-esperimento che esegue la creazione dell'utenza e, nel caso degli esperimenti di Introduction (Sezione 4.3) e degli esperimenti relativi ai messaggi privati (Sezioni 4.6, 4.7 e 4.8), ai gruppi privati (Sezioni 4.10, 4.11, 4.12 e 4.13), ai forum (Sezioni 4.14, 4.15, 4.16 3 4.17) e al blog (Sezioni 4.18, 4.19, 4.20 e 4.21), effettua anche l'aggiunta dei contatti
- I sotto-esperimenti necessari per simulare le azioni dell'utente e ottenere gli artefatti utili per l'analisi forense della funzionalità d'interesse. Le azioni simulate cambiano in base a qual è la funzionalità di cui si vogliono ottenere gli artefatti da analizzare (es. sotto-esperimento 1 dell'esperimento 4 in Tabella 4.3 "A introduce B a C (B e C rifiutano)" simula le azioni necessarie per inviare un'introduction agli utenti B e C da parte dell'utente A)

Definizione della durata delle istruzioni di SLEEP

La natura peer-to-peer di Briar è stata una problematica molto rilevante anche nell'esecuzione degli esperimenti automatici in AnForA in quanto in questo tipo di esperimenti il programma esegue automaticamente una dopo l'altra le istruzioni presenti nel file delle azioni non permettendo di definire trigger che si attivano in risposta a determinati eventi. Questo ha portato a dover definire manualmente, all'interno dei file delle azioni degli esperimenti, diverse istruzioni di SLEEP la cui durata (definita in secondi) doveva essere impostata durante la scrittura del file delle azioni.

Per comprendere meglio il notevole impatto di tutto ciò si riporta un caso specifico.

Durante l'esperimento di aggiunta contatti da lontano (Sezione 2.2.1.1), nei vari file delle azioni che compongono l'esperimento, è simulata la procedura di aggiunta contatto. Durante tale procedura è stato necessario aggiungere un'istruzione di SLEEP da 10 secondi per avere il tempo di effettuare lo scambio dei link manualmente. Una volta conclusa tale procedura, come illustrato in Figura 2.5 della Sezione 2.2.1.1, l'utente remoto si trova tra i contatti in sospeso. In questa fase è stata inserita un'ulteriore istruzione di SLEEP la cui durata è stata difficile da definire. Infatti, il tempo che serve all'utente locale per connettersi con l'utente remoto può variare da pochi secondi a diversi minuti. Questa variabilità ha comportato il fallimento di diverse esecuzioni dell'esperimento e la necessità di eseguirlo nuovamente richiedendo una quantità non indifferente di tempo dedicato alla riesecuzione dello stesso esperimento.

Utilizzo di Diffoscope in AnForA

Database h2 e Diffoscope

Un'ulteriore criticità è stata sollevata dall'utilizzo integrato in AnForA di Diffoscope. Quest'ultimo è un software che permette il confronto di diversi tipi di file, tra cui i database. Attualmente questo software supporta solo il calcolo delle differenze tra database SQLite. Briar al contrario utilizza un database h2, quindi per poter comunque usufruire di Diffoscope e quindi eseguire

l'analisi di Briar su dati generati da esperimenti riproducibili, si è deciso di ideare una procedura che permettesse di ottenere un database SQLite partendo da un database h2.

Tale procedura ha richiesto:

- La modifica del codice sorgente di Briar in modo che al sign out dell'applicazione venisse creato uno script SQL. Tale script utilizza un dialetto SQL per database h2.
Si è scelto di inserire la modifica al sign out dell'applicazione in modo che nello script SQL fossero presenti i dati generati dalle azioni eseguite durante il sotto-esperimento. Ciò ha comportato l'aggiunta delle istruzioni che eseguono il sign out alla fine del file delle azioni di ogni sotto esperimento.
Successivamente alla modifica del codice sorgente di Briar, è stata compilata un'APK di Briar contenente la variazione descritta e usata al posto dell'APK originale.
- L'implementazione di uno script Python che traducesse lo script SQL ottenuto al sign out dell'applicazione in uno script SQL con dialetto per database SQLite e creasse il database SQLite eseguendo lo script ottenuto dalla traduzione.
Lo script è eseguito sulla cartella dell'esperimento e per ogni sottocartella della cartella dell'esperimento, effettua la traduzione di tutti gli script SQL e la creazione del database SQLite

In seguito a queste aggiunte, il processo di esecuzione degli esperimenti in AnForA è diventato il seguente:

1. Esecuzione dell'esperimento in AnForA su emulatore in cui è installata l'APK contenente la modifica precedentemente illustrata
2. Alla conclusione dell'esperimento, avvio dello script Python che esegue la traduzione degli script SQL e la creazione del database SQLite
3. Alla fine dell'esecuzione dello script Python, click sul tasto 'Auto Differ' di AnForA. Questo tasto avvia Diffoscope per il calcolo delle differenze tra i database SQLite ottenuti al punto precedente

Lunghezza dei blocchi di differenze in Diffoscope

Sempre relativamente a Diffoscope utilizzato in AnForA, si è identificata una complicazione aggiuntiva. Infatti, il numero massimo di default delle righe (40) che Diffoscope può mostrare per ciascun blocco di differenze in una singola pagina di output è risultato insufficiente per visualizzare tutte le differenze presenti nei file del database. È stato quindi necessario modificare il codice sorgente di AnForA per aggiungere l'opzione '--max-page-diff-block-lines' al comando richiamato nel codice di AnForA e che lancia l'esecuzione di Diffoscope. Questa opzione permette di definire un numero massimo di righe diverso da quello di default. Il valore di tale opzione è stato impostato a 1000.

Letture di QR-code durante gli esperimenti in AnForA

La necessità di eseguire un esperimento che eseguisse l'aggiunta di contatti da vicino (Sezione 2.2.1.2) ha richiesto invece la risoluzione di un altro problema: la lettura del QR-code generato da un emulatore da parte di un altro emulatore e viceversa.

Per aggiungere un contatto da vicino, infatti i due utenti devono incontrarsi e ognuno di essi deve leggere il QR-Code dell'altro dispositivo. Un'azione così semplice su dispositivi fisici ha richiesto uno studio approfondito per poter essere eseguito su emulatori.

Per risolvere tale ostacolo si è configurata l'impostazione 'Camera Back' dell'emulatore d'interesse al valore 'VirtualScene'. Tale impostazione è accessibile dal Virtual Device Manager, entrando nelle impostazioni avanzate dell'emulatore.

'VirtualScene' è uno dei possibili valori configurabili per l'impostazione 'Camera Back'. Questo valore permette di visualizzare dalla fotocamera dell'emulatore una scena virtuale in cui sono presenti spazi appositi dove caricare immagini. Utilizzando questi spazi per inserire le immagini dei QR-Code, l'emulatore riesce a leggerli e a eseguire l'aggiunta di contatti da vicino.

Appium in AnForA

L'ultimo problema riscontrato è stato causato da Appium¹, un framework impiegato in AnForA per registrare le azioni eseguite durante un esperimento manuale, facilitando così la scrittura dei file delle azioni da impiegare negli esperimenti automatici. Questo strumento permette ad AnForA di identificare gli elementi dell'interfaccia dell'applicazione mediante ID o Accessibility ID o XPath. Una volta ottenute tali informazioni, per ogni elemento identificato su cui è stata eseguita un'operazione, viene costruita la descrizione dell'azione (es.

Tap,RES_ID:org.briarproject.briar.android:id/addButton). Il problema è emerso quando durante la costruzione dei file delle azioni mediante la registrazione delle azioni eseguite durante un esperimento manuale, Appium utilizzato in AnForA non è stato in grado di identificare correttamente l'elemento dell'interfaccia oggetto dell'azione.

Si è quindi proceduto tentando di identificare tali elementi usando Appium Inspector², ma anche questa strada non è stata percorribile in quanto l'APK di release di Briar ha il FLAG_SECURE attivato. Il FLAG_SECURE è un flag che, se impostato a true non permette di acquisire screenshot dell'applicazione. Poiché Appium Inspector cattura screenshot per visualizzare lo schermo dell'emulatore e la versione di release di Briar ha il FLAG_SECURE attivato, inizialmente non è stato possibile utilizzare tale strumento per identificare gli elementi dell'interfaccia di Briar.

È stato possibile risolvere tale problematica e quindi identificare gli elementi dell'interfaccia che Appium non riusciva a rilevare correttamente, avendo accesso al codice sorgente di Briar. Grazie a ciò, si è disabilitato il FLAG_SECURE lato codice e si è compilata un'APK di release con questa modifica.

Installando quest'ultima APK nell'emulatore è stato possibile utilizzare Appium Inspector per individuare gli elementi dell'interfaccia non correttamente identificati da Appium.

Gestione dei Messaggi e Metadati nel Sistema Briar

In Briar, ogni messaggio ricevuto o inviato è salvato nella tabella MESSAGES.

Il campo più importante della tabella MESSAGES è il campo RAW composto dal GROUPID seguito dal timestamp del messaggio (8 byte) a sua volta seguito dal un BDF (<GROUPID><timestamp><BDF>). Il timestamp contiene il momento di invio del messaggio da

¹ Appium è un framework open-source e multipiattaforma, ideato per facilitare l'automazione dell'interfaccia utente di molte piattaforme di app, tra cui quelle per dispositivi mobili (ad es., iOS e Android), browser (ad es., Firefox e Safari), desktop (ad es., macOS e Windows) e TV (ad es., Roku e Android TV). [\[24\]](#)

² Appium Inspector è uno strumento con interfaccia grafica per Appium. Permette di effettuare un'ispezione degli elementi di una applicazione mobile in modo più intuitivo utilizzando l'interfaccia grafica messa a disposizione. [\[25\]](#)

parte dell'utente se l'utente locale è il mittente, mentre contiene il momento della ricezione del messaggio se l'utente locale è il destinatario del messaggio.

È il BDF che contiene le informazioni forensicamente più interessanti quali per esempio il tipo di messaggio, il testo del messaggio, ecc...

Ogni messaggio è corredato da metadati salvati nella tabella MESSAGEMETADATA.

Qualsiasi tipo di messaggio ha sempre il seguente metadato associato e recuperabile nella tabella MESSAGEMETADATA:

- Il timestamp (chiave timestamp) che è un BDF in cui è presente un intero a 64 byte. Questo intero corrisponde al timestamp, espresso in esadecimale, presente nel campo RAW del messaggio. Trasformando il timestamp da esadecimale a decimale, si ottiene un timestamp che segue il fuso orario GMT+0.

Se l'utente locale e l'utente remoto sono entrambi online, ogni messaggio inviato o ricevuto è salvato nella tabella STATUSES. I messaggi ricevuti hanno i campi EXPIRY, TXCOUNT e MAXLATENCY non valorizzati e il campo SEEN a 0, mentre i messaggi inviati hanno i campi EXPIRY, TXCOUNT e MAXLATENCY valorizzati e il campo SEEN a 1.

Se l'utente locale o l'utente remoto non sono online quando l'utente locale invia un messaggio, il messaggio rimane con campi EXPIRY, TXCOUNT e MAXLATENCY non valorizzati e campo SEEN uguale a 0 fino a quando entrambi gli utenti non tornano online.

Dalla tabella STATUSES è inoltre possibile ottenere l'id del contatto a cui è stato inviato o da cui è stato ricevuto il messaggio (campo CONTACTID) e il timestamp di invio del messaggio da parte dell'applicazione o di ricezione del messaggio da parte dell'applicazione in formato decimale (campo TIMESTAMP).

È infine possibile ricostruire lo storico degli inviti ricevuti o inviati, il collegamento tra messaggio di post e re-post e il collegamento tra il messaggio di accettazione dell'invito con il messaggio di abbandono del dialogo attraverso i dati contenuti nella tabella MESSAGEDEPENDENCIES.

4.1 - Creazione dell'account utente

4.1.1 - Esperimenti

L'analisi di questa funzionalità ha richiesto un solo esperimento:

Numero esperimento	Descrizione	Sotto-esperimenti
1	Configurazione dell'account utente	1. A si registra in Briar

Tabella 4.1: Lista esperimenti per creazione account utente

Per eseguire l'esperimento, è stato utilizzato il seguente file delle azioni che specifica ad AnForA le operazioni da eseguire, corrispondenti ai passi della registrazione in Briar descritti nella Sezione 2.1.

	GROUPID	CLIENTID	MAJORVERSION	DESCRIPTOR
	Filtro	Filtro	Filtro	Filtro
1	2d0f35130fa045381b469c79f66714de1b59e0acf4468898072738d05d26b30a	org.briarproject.bramble.versioning	0	
2	9ee405152135a8a148832c27b7217eff111a25e622dbb5305dac9624dfc194be	org.briarproject.bramble.mailbox.properties	2	
3	8bc014f18e9e7491eb12bf7ec5faba537131df93351fa2b36519aec5f0cec52	org.briarproject.bramble.properties	0	
4	7668710f2df34020d39dd11642ead14e3941313e93e3037ac1a30323a0b43867	org.briarproject.bramble.transport.agreement	0	
5	fe24883ae0a51ef368a5045871da084e6740ece4e8cb5224285e7781612ae697	org.briarproject.briar.autodelete	0	
6	fc265d5878abf7040420c521f2f5dbcb9642daa25322c444a944f0d0ecd9087	org.briarproject.briar.avatar	0	35b2b79513faede53da8aa9806b8edf...
7	a29de2e92baf39bdf9e0566a9d18a7a4ec3aaecca5a85073fcbcb15c90296c	org.briarproject.briar.blog	0	60602101410550726f76615120d11dfc...
8	52b1bc1016ea05ecb646fe683f26aa45822d47193ffdf6f076937426b22156978	org.briarproject.briar.feed	0	
9	54330e6d1ac608216acd3b82ba4c35a1d2b0ea16de68b9c8222b3468da400948	org.briarproject.briar.privategroup.invitation	0	
10	7fa1cd54758509d6ebac0bc85af198cb17b1b8bcd1dba4cf4a421e1cd52d92	org.briarproject.briar.messaging	0	
11	e002af0d60195082c32798f2c58956bf646c9f30e817882fb871049e199f6b1d	org.briarproject.briar.forum.sharing	0	
12	added3edcc74faea8ca48e3b4378deee5c517f67e9877879ad994e3e087089f79	org.briarproject.briar.blog.sharing	0	
13	ff18982f1b5dba23d517c2872c839e963b958ef3dc45af42e3d232d0fc0df611	org.briarproject.briar.introduction	1	

Figura 4.2: Dati tabella GROUPS

Come si può notare nella Figura 4.2, tutti i gruppi tranne quello relativo al CLIENTID org.briarproject.briar.avatar e quello relativo al CLIENTID org.briarproject.briar.blog hanno il campo DESCRIPTOR vuoto.

Il campo DESCRIPTOR del gruppo relativo al CLIENTID org.briarproject.briar.avatar contiene l'id dell'utente locale.

Il campo DESCRIPTOR del gruppo relativo al CLIENTID org.briarproject.briar.blog contiene un BDF con la struttura definita nella Sezione 3.1.3.6. Inoltre, il nickname dell'oggetto 'author' del BDF contenuto nel descrittore del gruppo in questione avrà lo stesso valore presente nel campo NAME nella tabella LOCALAUTHORS.

	AUTHORID	FORMATVERSION	NAME	PUBLICKEY	PRIVATEKEY	HANDSHAKEPUBLICKEY	HANDSHAKEPRIVATEKEY	CREATED
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
	35b2b79513faede53da8aa9806b8edf93f852978e176833dbc968bca2ce41782	1	Prova	d11dfc06da281...	58271970a7bfac8...	b98ed26b2fee18b7653124...	70d3524787f9cb1fb169a8...	1715074748691

Figura 4.3: Dati tabella LOCALAUTHORS

Nella tabella LOCALAUTHORS sono inseriti i dati relativi all'utente locale. Di particolare interesse sono i campi NAME in cui è salvato lo username dell'utente che non deve per forza corrispondere al nome reale dell'utente e il campo CREATED che contiene il timestamp in millisecondi del momento in cui l'account dell'utente locale è stato creato quindi il momento in cui l'utente si è registrato in Briar.

4.1.3 - Query

L'unica informazione rilevante che è possibile recuperare dal database sono i dati dell'utente locale che possono essere estratti dal database mediante la seguente query.

```
SELECT * FROM LOCALAUTHORS
```

4.2 - Aggiunta contatti

4.2.1 - Esperimenti

L'analisi di questa funzionalità ha richiesto diversi esperimenti per poter individuare gli artefatti generati in ogni modalità di aggiunta di contatti. Briar, infatti, come esplicitato nella Sezione 2.2.1 ha due modalità di aggiunta contatti: una da lontano e una da vicino.

Numero esperimento	Descrizione	Sotto-esperimenti
2	Aggiunta contatti lontani	<ol style="list-style-type: none">1. A aggiunge B via link e A non si collega con B2. A cancella la Pending Contact Request di B3. A aggiunge B via link4. A si collega con B
3	Aggiunta contatti vicini	<ol style="list-style-type: none">1. A aggiunge B via Qr code

Tabella 4.2: Lista esperimenti aggiunta contatti

Gli esperimenti sono costituiti da un insieme di sotto-esperimenti. Per ogni sotto-esperimento è stato definito un file delle azioni che simula ciò che l'utente deve fare in app per effettuare il passo.

4.2.1.1 - Aggiunta contatti da lontano

L'esperimento 2 è stato pensato per ottenere tutti gli artefatti relativi all'aggiunta dei contatti da lontano.

Tale esperimento è stato suddiviso in 4 sotto-esperimenti:

1. Aggiunta contatto da lontano in sospeso
2. Aggiunta contatto da lontano fallita
3. Aggiunta contatto da lontano in sospeso
4. Aggiunta contatto da lontano riuscita

I file delle azioni utilizzati in questo esperimento sono i seguenti.

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 # Aggiunta contatto da lontano
7 Tap,RES_ID:org.briarproject.briar.android:id/fab_main
8 SLEEP,2
9 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
yout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout
[2]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.Linea-
rLayout[1]/android.widget.ImageButton[@resource-id="org.briarproject.briar.android:id/fab_mini"][1]
10 SLEEP,10
11 SendKeys,RES_ID:org.briarproject.briar.android:id/linkInput,TEXT:briar://adekc5vt1mh2as5mhgs7abra1j4f5mhr5wv2ry4xt-
tqcaqantx2
12 HideKeyboard
13 #SLEEP
14 Tap,RES_ID:org.briarproject.briar.android:id/addButton
15 #SLEEP
16 SendKeys,RES_ID:org.briarproject.briar.android:id/contactNameInput,TEXT:AMICO_1
17 HideKeyboard
18 #SLEEP
19 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.
ViewGroup[1]/android.widget.FrameLayout[2]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widg-
et.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.Button[1]
20 SLEEP,10
21 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.
ViewGroup[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navi-
gate up']
22 SLEEP,1
23 # Apertura menù laterale
24 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
on[@content-desc='Open the navigation drawer']
25 # Sign-out in Briar
26 SLEEP,5
27 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
28 SLEEP,5

```

Figura 4.4: File delle azioni sotto-esperimento1 dell'esperimento numero 2

Nelle righe 1 e 5 della Figura 4.4 è eseguito il sign in nell'applicazione. Questa azione è necessaria in quanto, come specificato nella Sezione 4.1.1, alla fine di ogni sotto esperimento bisogna eseguire il sign out dall'applicazione per poter eseguire il codice che permette di ottenere il file .sql del database. Tali istruzioni saranno quindi presenti come istruzioni iniziali di tutti i file delle azioni a partire dal secondo sotto esperimento di ogni esperimento.

Nelle righe 7 e 9 della Figura 4.4. sono definite le azioni per selezionare la modalità di aggiunta contatti da lontano (Sezione 2.2.1).

Dalla riga 10 fino alla riga 19 della Figura 4.4 sono specificate tutte le azioni necessarie per eseguire la procedura di aggiunta contatti da lontano (Sezione 2.2.1.1).

Come si può notare nella Figura 4.4, la riga 11 inserisce il link generato dall'utente remoto per poterlo aggiungere tra i contatti. Come spiegato nella Sezione 2.2.1.1 i due utenti per potersi aggiungere da lontano devono scambiarsi i link che durante ogni procedura di aggiunta contatti da lontano genera. Si è deciso di inserire hard-coded nel file delle azioni il link generato dall'istanza di Briar dell'utente remoto per automatizzare completamente il sotto-esperimento.

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,20
7 # Rimozione del contatto in sospenso dalla pending list
8 Tap,XY:957-2080
9 SLEEP,2
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[2]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.recyclerview.widget.RecyclerView[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Remove']
11 SLEEP,2
12 Tap,RES_ID:android:id/button2
13 SLEEP,2
14 # Apertura menù laterale
15 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Open the navigation drawer']
16 # Sign-out in Briar
17 SLEEP,5
18 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
19 SLEEP,5

```

Figura 4.5: File delle azioni sotto-esperimento 2 dell'esperimento numero 2

Nelle righe 1 e 5 della Figura 4.5 è eseguito il sign in nell'applicazione.

Nella riga 8 della Figura 4.5 si esegue il Tap sul tasto che permette di accedere alla lista dei contatti in attesa (Figura 2.5 della Sezione 2.2.1.1).

Infine, nella riga 10 della Figura 4.5 è effettuato il Tap sul tasto che esegue la cancellazione del contatto dalla lista dei contatti in attesa.

Il file delle azioni che esegue il sotto-esperimento 3 dell'esperimento 2 è identico al file delle azioni utilizzato per il passo 1 dell'esperimento 2 (Figura 4.4).

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 # Aggiunta del contatto in sospenso
7 SLEEP,20
8 Tap,XY:957-2080
9 SLEEP,120
10 # Apertura menù laterale
11 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Open the navigation drawer']
12 # Sign-out in Briar
13 SLEEP,5
14 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
15 SLEEP,5

```

Figura 4.6: File delle azioni sotto-esperimento 4 dell'esperimento numero 2

Nella Figura 4.6, le istruzioni che seguono il sotto-esperimento 4 sono quelle presenti a riga 8 e a riga 9.

Nella riga 8 si esegue il Tap sul tasto che permette di accedere alla lista dei contatti in attesa (Figura 2.5 della Sezione 2.2.1.1).

Mentre nella riga 9, si mette in attesa l'esecuzione dell'esperimento in modo da dare il tempo all'applicazione di connettere i due utenti cioè di aggiungere l'utente B come contatto dell'utente A e viceversa.

4.2.1.2 - Aggiunta contatti da vicino

```
1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 # Aggiunta contatto da vicino
7 Tap,RES_ID:org.briarproject.briar.android:id/fab_main
8 SLEEP,5
9 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout
  [2]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.Line-
  arLayout[2]/android.widget.ImageButton[@resource-id='org.briarproject.briar.android:id/fab_mini']
10 SLEEP,10
11 Tap,RES_ID:org.briarproject.briar.android:id/continueButton
12 SLEEP,2
13 Tap,RES_ID:com.android.permissioncontroller:id/permission_allow_foreground_only_button
14 SLEEP,2
15 Tap,RES_ID:com.android.permissioncontroller:id/permission_allow_button
16 SLEEP,2
17 Tap,RES_ID:android:id/button1
18 SLEEP,120
19 # Apertura menù laterale
20 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
  on[@content-desc='Open the navigation drawer']
21 # Sign-out in Briar
22 SLEEP,5
23 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
24 SLEEP,5
```

Figura 4.7: File delle azioni sotto-esperimento 1 dell'esperimento numero 3

Nelle righe 7 e 9 della Figura 4.7 sono definite le azioni per selezionare la modalità di aggiunta contatti da vicino (Sezione 2.2.1).

Dalla riga 11 fino alla riga 18 della Figura 4.7 sono specificate tutte le azioni necessarie per eseguire la procedura di aggiunta contatti da vicino (Sezione 2.2.1.2).

Più nello specifico alla riga 18, si mette in attesa l'esecuzione dell'esperimento in modo da avere il tempo di eseguire manualmente la lettura del Qr code da parte di entrambi gli utenti e di dare il tempo all'applicazione di connettere i due utenti, cioè di aggiungere l'utente B come contatto dell'utente A e viceversa. La procedura di lettura Qr code è stata eseguita manualmente in quanto, come spiegato nella Sezione '[Problematiche riscontrate](#)', la lettura di un Qr code presente su un emulatore da parte di un altro emulatore richiede l'utilizzo della Virtual Scene nell'opzione della Camera posteriore.

4.2.2 - Artefatti generati

Ogni passaggio degli esperimenti illustrati nella sezione precedente crea artefatti dai quali è possibile ottenere informazioni di rilevanza forense, quali se il contatto è già stato aggiunto o meno alla lista contatti, la modalità di aggiunta del contatto e il nome del contatto. Tutto ciò può essere dedotto dai dati presenti nel database.

Quando un contatto è aggiunto da lontano attraverso il link, prima di essere aggiunto tra i contatti rimane per una certa quantità di tempo come contatto in attesa. In questa situazione il contatto non è ancora salvato nella tabella CONTACTS ma è salvato nella tabella PENDINGCONTACTS (Figura 4.8) dove è possibile recuperare i dati dell'utente in attesa quali l'alias del contatto e il timestamp del momento in cui si è iniziata la procedura di aggiunta contatto.

PENDINGCONTACTID	PUBLICKEY	ALIAS	TIMESTAMP
Filtro	Filtro	Filtro	Filtro
ae358c837afaf682ec11d4a3ef417eaae0d05d90e185b44cc19bd13460d66206	c8a176b35b0f...	AMICO_1	1716452718193

Figura 4.8: Dati tabella PENDINGCONTACTS

Sia che l'aggiunta avvenga da lontano (attraverso il link) sia che avvenga da vicino (attraverso il QR-code), una volta che il contatto è aggiunto nella lista contatti non vi è modo di recuperare il timestamp di quando il contatto è stato aggiunto.

Se un contatto è stato aggiunto da lontano (Figura 4.9), il contatto nella tabella CONTACTS risulterà non verificato (campo VERIFIED uguale a 0 = False) mentre se il contatto è aggiunto da vicino (Figura 4.10), il contatto nella tabella CONTACTS risulterà verificato (campo VERIFIED uguale a 1 = True).

Il campo ALIAS contiene il nome che l'utente ha dato a quel contatto durante la procedura di aggiunta contatto e può essere valorizzato (nel caso l'utente abbia dato al contatto un nome diverso dal nome inserito dal contatto al momento dell'iscrizione a Briar) oppure NULL nel caso in cui l'utente non definisca un alias diverso dal nome del contatto. Nel caso di aggiunta di contatto da lontano attraverso il link, durante la procedura di aggiunta contatto da lontano, l'utente deve obbligatoriamente definire un alias per il contatto.

L'alias può essere modificato in un qualsiasi momento dopo l'aggiunta del contatto.

CONTACTID	AUTHORID	NAME	ALIAS	LOCALAUTHORID	VERIFIED
1	cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e	Amico1	AMICO_1	a7e38e165d4a5c5e261d852543b518d371f5f4b13c5d98f179692897c71b9ae7	0

Figura 4.9: Dati tabella CONTACTS (aggiunta contatto da lontano)

CONTACTID	AUTHORID	NAME	ALIAS	LOCALAUTHORID	VERIFIED
1	cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e	Amico1	NULL	bf8eb23f6ef342c432598d59f15a2043734b083f7919936d7c9d87957e6a0229	1

Figura 4.10: Dati tabella CONTACTS (aggiunta contatto da vicino)

Per ogni contatto aggiunto alla lista contatti è creato un gruppo per ogni client BSP. Ognuno di questi gruppi è aggiunto nella tabella GROUPS. A differenza dei gruppi generati in fase di creazione account dell'utente locale, nel campo DESCRIPTOR della tabella GROUPS i gruppi creati per un contatto contengono dati diversi.

È possibile, infatti, definire a quale contatto fa riferimento un gruppo nel seguente modo:

- I client BSP che fanno riferimento all'utente locale hanno il campo DESCRIPTOR della tabella GROUPS vuoto (tranne i gruppi org.briarproject.briar.avatar e org.briarproject.briar.blog che nel campo DESCRIPTOR hanno rispettivamente l'id dell'utente locale e un BDF che contiene il nome dell'utente locale e la sua chiave pubblica)
- I client BSP che fanno riferimento a un contatto della lista dei contatti nel campo DESCRIPTOR della tabella GROUPS hanno un BDF che contiene l'id dell'utente locale (campo AUTHORID in LOCALAUTHORS) e l'id remoto del contatto (campo AUTHORID in CONTACTS).

I client BSP org.briarproject.briar.avatar e org.briarproject.briar.autodelete relativi al contatto aggiunto, invece, riportano nel campo DESCRIPTOR l'id remoto del contatto aggiunto (corrispondente al campo AUTHORID in CONTACTS).

Infine, il client BSP org.briarproject.briar.blog riporta nel campo DESCRIPTOR il nome del proprietario del blog, la sua chiave pubblica e l'indicazione se il blog è un feed RSS o meno.

Come è possibile dedurre dalla Figura 4.11, in base a ciò che è stato appena descritto, le tuple dalla 1 alla 13 comprese contengono i gruppi relativi all'utente locale mentre le tuple dalla 14 alla 25 comprese contengono i gruppi relativi a un contatto nella lista contatti.

	GROUPID	CLIENTID	DESCRIPTOR
1	2d0f35130fa045381b469c79f66714de1b...	org.briarproject.bramble.versioning	
2	9ee405152135a8a148832c27b7217eff11...	org.briarproject.bramble.mailbox.properties	
3	8bc014f18e9e7491eb12bf7ec5faba5371...	org.briarproject.bramble.properties	
4	766810f2df34020d39dd11642ead14e3...	org.briarproject.bramble.transport.agreement	
5	fe24883ae0a51ef368a5045871da084e67...	org.briarproject.briar.autodelete	
6	7f8dc0fe7a9e8b7dbecaac931c3048f658...	org.briarproject.briar.avatar	a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae7
7	03711033481a3545226be82da3994f1c...	org.briarproject.briar.blog	606021014106416d69636f315120e60d2823aba5fc8a849f5ae97234258eb47adce928d7de2d034a8f393deec8801080
8	52b1bc1016ea05ecb646fe683f26aa4582...	org.briarproject.briar.feed	
9	54330e6d1ac608216acd3b82ba4c35a1d...	org.briarproject.briar.privategroup.invitation	
10	7fa1cd547585509d6ebac0bc8f5af198cb...	org.briarproject.briar.messaging	
11	e002af06d0195082c32798f2c58956bf64...	org.briarproject.briar.forum.sharing	
12	fd43edc74faea8ca48e3b4378deec5c51...	org.briarproject.briar.blog.sharing	
13	ff18982f1b5dba23d517c2872c839e963b...	org.briarproject.briar.introduction	
14	4aac5bd190be8c5a67ce57e10a308aaf2f...	org.briarproject.bramble.versioning	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80
15	e65f6ea21617a52cfc1c0e12eb196b95bc...	org.briarproject.bramble.mailbox.properties	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80
16	96b54773182bcab2a56f8e51189e8d7c5...	org.briarproject.bramble.properties	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80
17	3b5e3c1f9d975f3c72eb0795e6c260a6...	org.briarproject.bramble.transport.agreement	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80
18	0dca5b6311504195e8521acda243c4a24f...	org.briarproject.briar.autodelete	cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e
19	ea2cabfee3673c87748b87eb18774fd7af...	org.briarproject.briar.avatar	cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e
20	e6609b9f814ee668601741a55010dae8a...	org.briarproject.briar.privategroup.invitation	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80
21	450d9233edb8a619f7e7b56cb589a73abb...	org.briarproject.briar.messaging	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80
22	288080cfc58d2f24c0568808de84b6e9ce...	org.briarproject.briar.forum.sharing	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80
23	2de779a23a9db97cf0b478445c5e3fd1a...	org.briarproject.briar.blog.sharing	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80
24	4fb5a4e8ba36ce43626b95352d272f2be1...	org.briarproject.briar.blog	606021014106416d69636f315120e60d2823aba5fc8a849f5ae97234258eb47adce928d7de2d034a8f393deec8801080
25	d61e31452b4801d51d28ce072fd1cdc9...	org.briarproject.briar.introduction	605120a7e38e1654da5c5e261d852543b518d371f54b13c5d98f179692897c71b9ae75120cc5d913f45959e68c9ee1cca7dbf50336054ebd1085a5ce5671cc4cd9829017e80

Figura 4.11: Dati tabella GROUPS dopo l'aggiunta di un contatto

È possibile determinare a quale contatto è associato un gruppo anche esaminando il valore del campo VALUE associato al GROUPID d'interesse e alla METAKEY con valore 'contactid' nella tabella GROUPSMETADATA.

Il campo VALUE contiene un BDF che quindi deve essere interpretato come descritto nella Sezione 3.1.2.1.

Prendendo come esempio la tupla 3 della Figura 4.12, si può notare che il valore del campo VALUE relativo a una qualsiasi METAKEY con valore 'contactid' è 2101. Essendo a conoscenza che VALUE coincide con il valore del campo CONTACTID nella tabella CONTACTS.

	GROUPID	METAKEY	VALUE
	Filtro	Filtro	Filtro
1	9ee405152135a8a148832c27b7217eff11a25e622dbb5305dac9624dfc194be	sentClientSupports	6060210121008080
2	52b1bc1016ea05ecb646fe683f26aa45822d47193ffd6f076937426b22156978	feeds	6080
3	4aac5bd190be8c5a67ce57e10a308aaf2f1a2d2f91bfaca5126c209bc3df6219	contactId	2101
4	e65f6ea21617a52cfc1c0e12eb196b95bc539402fdc6316475d9be4e128cb229	contactId	2101
5	3b5e3c1f9d975f3c72eb0795e6c260a644f02252ffdbf04fc7037dbbe4a64	contactId	2101
6	0dca5b6311504195e8521acda243c4a24fd0f91da209057659fd5f9c5815d500	contactId	2101
7	ea2cabfee3673c87748b87eb18774fd7afc48c11a9367bab0ac8107b14d734b4	contactId	2101
8	e6609b9f814ee668601741a55010dae8a10ef5b6a93642273bd1a3ec18a7aca1	contactId	2101
9	450d9233edb8a619f7e7b56cb589a73abb48e7e7e9d7e3457e57aecef9b0b06c	contactId	2101
10	450d9233edb8a619f7e7b56cb589a73abb48e7e7e9d7e3457e57aecef9b0b06c	messageCount	2100
11	450d9233edb8a619f7e7b56cb589a73abb48e7e7e9d7e3457e57aecef9b0b06c	latestMessageTime	280000018fa490ecd4
12	450d9233edb8a619f7e7b56cb589a73abb48e7e7e9d7e3457e57aecef9b0b06c	unreadCount	2100
13	288080cfc58d2f24c0568808de84b6e9ce87f52fd9ed0fb6d513b09747843e	contactId	2101
14	2de779a23a9db97cf0b478445c5e3fd1a267ffe634e575ce5897d9f534b9bee	contactId	2101
15	d61e31452b4801d51d28ce072fd1cdc9a4e1a1fa614e8d922c0145a40ed86fa	contactId	2101

Figura 4.12: Dati tabella GROUPSMETADATA

4.2.3 - Query

È possibile ottenere tutti i metadati relativi a tutti i gruppi eseguendo la seguente query

```
SELECT *
FROM GROUPS JOIN GROUPMETADATA USING (GROUPID)
WHERE 1
```

Se, invece, si vogliono ottenere i metadati relativi a tutti i gruppi di un determinato client basta aggiungere nella clausola WHERE la condizione CLIENTID = 'nome del client id'

```
SELECT *
FROM GROUPS JOIN GROUPMETADATA USING (GROUPID)
WHERE CLIENTID = "org.briarproject.briar.blog.sharing"
```

Infine, se si vogliono ottenere tutti i gruppi associati a un contatto è necessario eseguire la seguente query

```
SELECT *
FROM GROUPS JOIN GROUPMETADATA USING(GROUPID)
WHERE METAKEY = "contactId" AND VALUE = "21" seguito dal valore CONTACTID del contatto
```

4.3 - Introduction

4.3.1 - Esperimenti

Gli esperimenti eseguiti per analizzare questa funzionalità sono stati molteplici in quanto era necessario analizzare l'introduction sia lato introducer (utente che effettua l'introduction tra due suoi contatti) sia lato introducee (utente che riceve la richiesta di introduction).

Numero esperimento	Descrizione	Sotto-esperimenti
4	'Introduction' tra contatti aggiunti da lontano	<ol style="list-style-type: none"> 1. A introduce B a C (B e C rifiutano) 2. A introduce B a C (B rifiuta e C accetta) 3. A introduce B a C (B e C accettano)
5	'Introduction' tra contatti aggiunti da vicino	<ol style="list-style-type: none"> 1. A introduce B a C (B e C rifiutano) 2. A introduce B a C (B rifiuta e C accetta) 3. A introduce B a C (B e C accettano)
6	'Introduction' come introducee	<ol style="list-style-type: none"> 1. B introduce A a C (A e C rifiutano) 2. B introduce A a C (A rifiuta e C accetta) 3. B introduce A a C (A accetta e C rifiuta)

		4. B introduce A a C (A e C accettano)
--	--	--

Tabella 4.3: Lista esperimenti introduction

Come si può notare nella Tabella 4.3, gli esperimenti 4 e 5 effettuano gli stessi sotto-esperimenti per poter controllare se l'introduction effettuata tra contatti aggiunti da lontano o tra contatti aggiunti da vicino creano gli stessi artefatti o meno.

Introducer

In ognuno di questi sotto-esperimenti le azioni eseguite dall'utente A sono identiche e per questo motivo ognuno dei sotto-esperimenti usa lo stesso file delle azioni che è riportato di seguito (Figura 4.13).

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,60
7 # Invio Introduction
8 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[2]/android.view.ViewGroup[1]/androidx.recyclerview.widget.RecyclerView[1]/android.view.ViewGroup[1]/android.widget.TextView[@resource-id='org.briarproject.briar.android:id/trustIndicatorDescription']
9 SLEEP,2
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.appcompat.widget.LinearLayoutCompat[1]/android.widget.ImageView[@content-desc='More options']
11 SLEEP,2
12 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.RelativeLayout[1]/android.widget.TextView[@resource-id='org.briarproject.briar.android:id/title']
13 SLEEP,2
14 Tap,RES_ID:org.briarproject.briar.android:id/nameView
15 SLEEP,2
16 SendKeys,RES_ID:org.briarproject.briar.android:id/input_text,TEXT:Conoscetevi
17 HideKeyboard
18 SLEEP,1
19 Tap,RES_ID:org.briarproject.briar.android:id/compositeSendButton
20 # Attesa risposta introduction
21 SLEEP,120
22 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
23 SLEEP,1
24 # Apertura menù laterale
25 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Open the navigation drawer']
26 # Sign-out in Briar
27 SLEEP,5
28 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
29 SLEEP,5

```

Figura 4.13: File delle azioni per i sotto-esperimenti 1, 2 e 3 dell'esperimento 4 e dell'esperimento 5

Nella Figura 4.13 la riga 8 esegue il Tap sul nome del contatto che si vuole introdurre. Questo permette di entrare nella conversazione privata con tale contatto.

Nelle righe 10 e 12 della Figura 4.13 si seleziona la funzionalità di introduction (Figura 2.8 della Sezione 2.2.2).

Nella riga 14 della Figura 4.13 è selezionato il secondo contatto (Figura 2.9 della Sezione 2.2.2).

Infine, nelle righe 16 e 19 della Figura 4.13 si eseguono rispettivamente l'azione di scrittura del messaggio di introduction e l'invio dell'invito all'introduction (Figura 2.10 della Sezione 2.2.2).

Come riportato nella didascalia delle Figura 4.13, le azioni effettuate dall'utente A (introducer) nei sotto-esperimenti 1, 2 e 3 negli esperimenti 4 e 5 sono identiche. Ciò è stato necessario per controllare se un'introduzione effettuata tra contatti aggiunti da vicino o tra contatti aggiunti da lontani generasse artefatti differenti.

Quello che cambia tra i sotto-esperimenti dell'esperimento 4 sono le azioni degli utenti B e C (introducee). Infatti, il rifiuto o l'accettazione dell'introduction da parte degli introducee generano dati differenti nel database dell'introducer. Stesso discorso vale per i sotto-esperimenti dell'esperimento 5.

Introducee

Nell'esperimento 6, l'utente A è uno degli introducee che ricevono l'invito all'introduction. Ogni introducee può rifiutare (eseguendo le azioni descritte nel file delle azioni in Figura 4.14) o accettare (eseguendo le azioni descritte nel file d'azione in Figura 4.15).

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,60
7 # Rifiuto alla richiesta di introduction
8 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
9 SLEEP,2
10 #SLEEP
11 Tap,RES_ID:org.briarproject.briar.android:id/declineButton
12 SLEEP,30
13 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android
  .widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
14 SLEEP,1
15 # Apertura menù laterale
16 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
  et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
  on[@content-desc='Open the navigation drawer']
17 # Sign-out in Briar
18 SLEEP,2
19 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
20 SLEEP,5

```

Figura 4.14: File delle azioni per i sotto-esperimenti 1 e 2 dell'esperimento numero 6

Nella riga 11 della Figura 4.14 è eseguita l'azione di rifiuto dell'invito di introduction.

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,60
7 # Accettazione della richiesta di introduction
8 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
9 SLEEP,2
10 Tap,RES_ID:org.briarproject.briar.android:id/acceptButton
11 SLEEP,30
12 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
  et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android
  .widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
13 SLEEP,1
14 # Apertura menù laterale
15 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
  et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
  on[@content-desc='Open the navigation drawer']
16 # Sign-out in Briar
17 SLEEP,2
18 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
19 SLEEP,5

```

Figura 4.15: File delle azioni per i sotto-esperimenti 3 e 4 dell'esperimento numero 6

Nella riga 10 della Figura 4.15 è eseguita l'azione di accettazione dell'invito di introduction.

4.3.2 - Artefatti generati

Gli artefatti generati durante un'introduction tra contatti aggiunti da lontano sono uguali a quelli generati durante un'introduction tra contatti aggiunti da vicino.

Invece, gli artefatti generati da un'introduction differiscono tra loro se l'utente oggetto d'analisi ha il ruolo di Introdncer o di Introducee, per ciò si tratteranno le due situazioni separatamente.

4.3.2.1 - Introdncer

Ogni utente che abbia nella propria lista contatti due o più contatti può effettuare un'introduction tra due dei suoi contatti.

È possibile dedurre se un utente ha effettuato un'introduction tra due suoi contatti controllando se nella tabella MESSAGES vi è un messaggio associato al GROUPID relativo al CLIENTID org.briarproject.briar.introduction con campo DESCRIPTOR vuoto (relativo all'utente locale).

Nella tabella MESSAGEMETADATA tale messaggio ha i seguenti metadati associati (Figura 4.16):

- role: definisce il ruolo dell'utente locale nell'introduction (introdncer– 0, introducee – 1). Il valore del campo è un BDF che contiene un intero il cui valore può essere 0 o 1. In questo caso, il valore del metadato è uguale a '2100' (21 = INT_8 e 00 = 0) quindi possiamo concludere che l'utente locale è un introdncer.
- introduceeA: contiene un BDF da cui è possibile estrarre il nome dell'introducee selezionato per primo durante la procedura di invio richiesta introduction (Tale metadato è presente solo nel database dell'introdncer)
- introduceeB: contiene un BDF da cui è possibile estrarre il nome dell'introducee selezionato per secondo (Tale metadato è presente solo nel database dell'introdncer)

Si prenda il valore del campo VALUE relativo alla METAKEY introduceeA nella Figura 4.16-

Tale valore è

```
704106617574686f726021014106414d49434f32512082713b21eb143ce313f878a4e00205e1f112
6638e2e8707a4e491b46a7a8533d80410767726f757049645120f2b02577af82e18e1716f7a2af88a
2b44a2c05066a2214e2a00010b3f1e28b3641126c6173744c6f63616c4d65737361676549645120d
b79b271c038e3d7a27be756e56ecef8c23e073b9c5991efb78d66e966ef7db541136c61737452656
d6f74654d65737361676549645120a3d33370c3e5142ae2b596ee1d96bae50bec5ca897930a6f238
790558a7d2cd1410e6c6f63616c54696d657374616d70280000018fa5e9c93480
```

Traducendo il BDF qui sopra come descritto nella Sezione 3.1.2.1 si ottiene:

- 70 corrisponde al tipo Dictionary
- 41 corrisponde al tipo STRING_8
- 06 corrisponde alla lunghezza della stringa (6 caratteri)
- 617574686f72 corrisponde alla stringa "author"
- 60 corrisponde al tipo LIST
- 21 corrisponde al tipo INT_8
- 01 corrisponde all'id del contatto
- 41 corrisponde al tipo STRING_8
- 06 corrisponde alla lunghezza della stringa (6 caratteri)
- 414d49434f32 corrisponde alla stringa "AMICO2"
- 51 corrisponde al tipo RAW_8

- 20 corrisponde alla lunghezza del valore di tipo RAW_8
- 82713b21eb143ce313f878a4e00205e1f1126638e2e8707a4e491b46a7a8533d corrisponde alla Public key di AMICO2
- 80 corrisponde al tipo END (chiude l'oggetto di tipo LIST)
- 41 corrisponde al tipo STRING_8
- 07 corrisponde alla lunghezza della stringa (7 caratteri)
- 67726f75704964 corrisponde alla stringa groupid
- 51 corrisponde al tipo RAW_8
- 20 corrisponde alla lunghezza del valore di tipo RAW_8 (32 caratteri)
- F2b02577af82e18e1716f7a2af88a2b44a2c05066a2214e2a00010b3f1e28b36 corrisponde al gruppo relativo al client id org.briarproject.briar.introduction riguardante l'introducee A
- 41 corrisponde al tipo STRING_8
- 12 corrisponde alla lunghezza della stringa (18 caratteri)
- 6c6173744c6f63616c4d6573736167654964 corrisponde alla stringa "lastLocalMessageId"
- 51 corrisponde al tipo RAW_8
- 20 corrisponde alla lunghezza del valore di tipo RAW_8 (32 caratteri)
- 3b59661f7bcdabb5da629803b323a0a8a243c5ce97f9a004872ecf0f8f63b37c corrisponde all'id dell'ultimo messaggio locale inviato dal gruppo relativo al client id org.briarproject.briar.introduction inerente l'introducee A
- 41 corrisponde al tipo STRING_8
- 13 corrisponde alla lunghezza della stringa (19 caratteri)
- 6c61737452656d6f74654d6573736167654964 corrisponde alla stringa "lastRemoteMessageId"
- 51 corrisponde al tipo RAW_8
- 20 corrisponde alla lunghezza del valore di tipo RAW_8 (32 caratteri)
- 68f03eadb5664f2d1058c4f210eedd5ba68022cf9b7eb7604be58c7d5cee714c corrisponde all'id dell'ultimo messaggio remoto ricevuto dal gruppo relativo al client id org.briarproject.briar.introduction inerente l'introducee A
- 41 corrisponde al tipo STRING_8
- 0e corrisponde alla lunghezza della stringa (14 caratteri)
- 6c6f63616c54696d657374616d70 corrisponde alla stringa "localTimestamp"
- 28 corrisponde al tipo INT_64
- 0000018fa5e31569 corrisponde al timestamp di invio dell'ultimo messaggio locale inviato dal gruppo relativo al client id org.briarproject.briar.introduction inerente l'introducee A
- 80 corrisponde al tipo END (chiude l'oggetto di tipo DICTIONARY)

	MESSAGEID	GROUPID	STATE	METAKEY	VALUE
1	900429195093e5ffaa74dd136d42cc843037f45...	ff18982f1b5dba23d517c2872c839e963b958ef...	3	introduceeA	704106617574686f726021014106414d49434f32...
2	900429195093e5ffaa74dd136d42cc843037f45...	ff18982f1b5dba23d517c2872c839e963b958ef...	3	introduceeB	704106617574686f726021014106416d69636f31...
3	900429195093e5ffaa74dd136d42cc843037f45...	ff18982f1b5dba23d517c2872c839e963b958ef...	3	requestTimestamp	280000018fa5e96656
4	900429195093e5ffaa74dd136d42cc843037f45...	ff18982f1b5dba23d517c2872c839e963b958ef...	3	role	2100
5	900429195093e5ffaa74dd136d42cc843037f45...	ff18982f1b5dba23d517c2872c839e963b958ef...	3	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997a1...
6	900429195093e5ffaa74dd136d42cc843037f45...	ff18982f1b5dba23d517c2872c839e963b958ef...	3	state	2100

Figura 4.16: Dati relativi al messaggio di gestione introduction in tabella MESSAGEMETADATA

Con "messaggio di gestione introduction" si intende un messaggio generato da Briar nel database locale e che permette di memorizzare le informazioni di una introduction quali quelle riportate in Figura 4.16.

Inoltre, in questo caso per la stessa richiesta di introduction ci saranno due messaggi, uno per ogni introducee, con metadato 'messageType' uguale a '2100' (richiesta di introduction) e tali messaggi risulteranno locali (chiave local uguale a 11).

Come si può notare nelle Figure 4.17 e 4.18 il GROUPID associato ai due messaggi è diverso. Infatti il messaggio i cui metadati sono riportati in Figura 4.17 è inviato dal gruppo associato al client org.briarproject.briar.introduction relativo al contatto che è introducee A, mentre il messaggio i cui metadati sono riportati in Figura 4.18, è inviato dal gruppo associato al client org.briarproject.briar.introduction relativo al contatto che è introducee B.

Entrambi i messaggi hanno però il valore del metadato "sessionId" uguale e questo ci segnala che tutt'e due i messaggi appartengono alla stessa introduction.

	MESSAGEID	GROUPID	STATE	METAKEY	VALUE
1	a71fa672dad7b4c01f5e81d595057555062af5...	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3	local	11
2	a71fa672dad7b4c01f5e81d595057555062af5...	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3	messageType	2100
3	a71fa672dad7b4c01f5e81d595057555062af5...	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3	read	11
4	a71fa672dad7b4c01f5e81d595057555062af5...	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997...
5	a71fa672dad7b4c01f5e81d595057555062af5...	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3	timestamp	280000018fa5e2da1a
6	a71fa672dad7b4c01f5e81d595057555062af5...	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3	visibleInUi	11

Figura 4.17: Dati relativi al messaggio di richiesta introduction in tabella MESSAGEMETADATA (Introducee A)

	MESSAGEID	GROUPID	STATE	METAKEY	VALUE
1	71c41c50c3125b3d7faad2bedd844df188bec4...	d16d7800daed208be02daa6b360a07d244e51...	3	local	11
2	71c41c50c3125b3d7faad2bedd844df188bec4...	d16d7800daed208be02daa6b360a07d244e51...	3	messageType	2100
3	71c41c50c3125b3d7faad2bedd844df188bec4...	d16d7800daed208be02daa6b360a07d244e51...	3	read	11
4	71c41c50c3125b3d7faad2bedd844df188bec4...	d16d7800daed208be02daa6b360a07d244e51...	3	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997...
5	71c41c50c3125b3d7faad2bedd844df188bec4...	d16d7800daed208be02daa6b360a07d244e51...	3	timestamp	280000018fa5e2da1a
6	71c41c50c3125b3d7faad2bedd844df188bec4...	d16d7800daed208be02daa6b360a07d244e51...	3	visibleInUi	11

Figura 4.18: Dati relativi al messaggio di richiesta introduction in tabella MESSAGEMETADATA (introducee B)

Come mostrato in Figura 4.19, se l'introduction è stata rifiutata, nella tabella MESSAGES del database dell'introducer è presente il messaggio di rifiuto dell'introducee con i seguenti metadati associati:

- messageType: 2102 (messaggio di rifiuto)
- local: 10 (False)

	GROUPID	MESSAGEID	METAKEY	VALUE
1	f2b02577af82e18e1716f7a2af88a2b44a2c050...	68f03eadb5664f2d1058c4f210eedd5ba68022c...	local	10
2	f2b02577af82e18e1716f7a2af88a2b44a2c050...	68f03eadb5664f2d1058c4f210eedd5ba68022c...	messageType	2102
3	f2b02577af82e18e1716f7a2af88a2b44a2c050...	68f03eadb5664f2d1058c4f210eedd5ba68022c...	read	11
4	f2b02577af82e18e1716f7a2af88a2b44a2c050...	68f03eadb5664f2d1058c4f210eedd5ba68022c...	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997...
5	f2b02577af82e18e1716f7a2af88a2b44a2c050...	68f03eadb5664f2d1058c4f210eedd5ba68022c...	timestamp	280000018fa5e2f4e2
6	f2b02577af82e18e1716f7a2af88a2b44a2c050...	68f03eadb5664f2d1058c4f210eedd5ba68022c...	visibleInUi	11

Figura 4.19: Dati relativi al messaggio di rifiuto dell'introduction da parte di un introducee in tabella MESSAGEMETADATA

Come si può vedere in Figura 4.20, se l'introduction è stata accettata, nella tabella MESSAGES è presente il messaggio di accettazione dell'introducee con i seguenti metadati associati:

- messageType: 2101 (messaggio di accettazione)
- local: 10 (False)

	GROUPID	MESSAGEID	METAKEY	VALUE
1	f2b02577af82e18e1716f7a2af88a2b44a2c050...	b68c2bc5b9139d3052220ef5da9ea3f281ee22...	local	10
2	f2b02577af82e18e1716f7a2af88a2b44a2c050...	b68c2bc5b9139d3052220ef5da9ea3f281ee22...	messageType	2101
3	f2b02577af82e18e1716f7a2af88a2b44a2c050...	b68c2bc5b9139d3052220ef5da9ea3f281ee22...	read	11
4	f2b02577af82e18e1716f7a2af88a2b44a2c050...	b68c2bc5b9139d3052220ef5da9ea3f281ee22...	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997...
5	f2b02577af82e18e1716f7a2af88a2b44a2c050...	b68c2bc5b9139d3052220ef5da9ea3f281ee22...	timestamp	280000018fa5e9936b
6	f2b02577af82e18e1716f7a2af88a2b44a2c050...	b68c2bc5b9139d3052220ef5da9ea3f281ee22...	visibleInUi	11

Figura 4.20: Dati relativi al messaggio di accettazione dell'introduzione da parte di un introducee in tabella MESSAGEMETADATA

Siccome è l'introducer che si occupa di inoltrare il rifiuto o l'accettazione di un introducee all'altro introducee, per ogni messaggio di risposta dell'introducee sarà presente un altro messaggio di inoltro della risposta dell'introducee A all'introducee B e viceversa. Questi messaggi avranno i seguenti metadati associati:

- messageType: 2102 se l'introducee di cui si inoltra la risposta ha rifiutato l'introduzione (Figura 4.21).
2101 se l'introducee di cui si inoltra la risposta ha accettato l'introduzione (Figura 4.22)
- local: 11 (True)

	GROUPID	MESSAGEID	METAKEY	VALUE
1	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3b59661f7bcdabb5da629803b323a0a8a243c5...	local	11
2	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3b59661f7bcdabb5da629803b323a0a8a243c5...	messageType	2102
3	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3b59661f7bcdabb5da629803b323a0a8a243c5...	read	11
4	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3b59661f7bcdabb5da629803b323a0a8a243c5...	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997...
5	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3b59661f7bcdabb5da629803b323a0a8a243c5...	timestamp	280000018fa5e31569
6	f2b02577af82e18e1716f7a2af88a2b44a2c050...	3b59661f7bcdabb5da629803b323a0a8a243c5...	visibleInUi	10

Figura 4.21: Dati relativi al messaggio d'inoltro messaggio di rifiuto di un introducee all'altro introducee in tabella MESSAGEMETADATA

	GROUPID	MESSAGEID	METAKEY	VALUE
1	d16d7800daed208be02daa6b360a07d244e51...	42a08171643bef688459697163c245cf11c390a...	local	11
2	d16d7800daed208be02daa6b360a07d244e51...	42a08171643bef688459697163c245cf11c390a...	messageType	2101
3	d16d7800daed208be02daa6b360a07d244e51...	42a08171643bef688459697163c245cf11c390a...	read	11
4	d16d7800daed208be02daa6b360a07d244e51...	42a08171643bef688459697163c245cf11c390a...	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997...
5	d16d7800daed208be02daa6b360a07d244e51...	42a08171643bef688459697163c245cf11c390a...	timestamp	280000018fa5e9a517
6	d16d7800daed208be02daa6b360a07d244e51...	42a08171643bef688459697163c245cf11c390a...	visibleInUi	10

Figura 4.22: Dati relativi al messaggio d'inoltro messaggio di accettazione di un introducee all'altro introducee in tabella MESSAGEMETADATA

Anche se l'utente locale è l'introducer, è possibile ricostruire se un'introduzione è andata a buon fine andando a recuperare per ognuno dei due contatti coinvolti (introducee), tutti i messaggi associati al gruppo relativo al client org.briarproject.briar.introduction, con lo stesso sessionId e con metadato messageType o uguale a '2103' (AUTH) o uguale a '2104' (ACTIVATE) (per il significato di AUTH e ACTIVATE si rimanda alla Sezione 3.1.3.10).

	GROUPID	MESSAGEID	METAKEY	VALUE
1	d16d7800daed208be02daa6b360a07d244e51...	81346cfe58a3dfbdcf7cfda2ba6af7d19d6884f0...	local	10
2	d16d7800daed208be02daa6b360a07d244e51...	81346cfe58a3dfbdcf7cfda2ba6af7d19d6884f0...	messageType	2103
3	d16d7800daed208be02daa6b360a07d244e51...	81346cfe58a3dfbdcf7cfda2ba6af7d19d6884f0...	read	10
4	d16d7800daed208be02daa6b360a07d244e51...	81346cfe58a3dfbdcf7cfda2ba6af7d19d6884f0...	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997...
5	d16d7800daed208be02daa6b360a07d244e51...	81346cfe58a3dfbdcf7cfda2ba6af7d19d6884f0...	timestamp	280000018fa5e9b02c
6	d16d7800daed208be02daa6b360a07d244e51...	81346cfe58a3dfbdcf7cfda2ba6af7d19d6884f0...	visibleInUi	10

Figura 4.23: Dati relativi al messaggio AUTH ricevuto da uno degli introducee in tabella MESSAGEMETADATA

L'introducer si occupa anche dell'inoltro dei messaggi di questo tipo (AUTH o ACTIVATE). Infatti, quando l'introducer riceve questi tipi di messaggi da un introducee (Figura 4.23) li inoltra all'altro introducee e viceversa (Figura 4.24).

	GROUPID	MESSAGEID	METAKEY	VALUE
1	d16d7800daed208be02daa6b360a07d244e51...	718cb5971a3a7ad047eedd15e3c9dcea4de583...	local	11
2	d16d7800daed208be02daa6b360a07d244e51...	718cb5971a3a7ad047eedd15e3c9dcea4de583...	messageType	2103
3	d16d7800daed208be02daa6b360a07d244e51...	718cb5971a3a7ad047eedd15e3c9dcea4de583...	read	11
4	d16d7800daed208be02daa6b360a07d244e51...	718cb5971a3a7ad047eedd15e3c9dcea4de583...	sessionId	5120d6efc7b0990a494549e36976ce2e9f95997...
5	d16d7800daed208be02daa6b360a07d244e51...	718cb5971a3a7ad047eedd15e3c9dcea4de583...	timestamp	280000018fa5e9ae5f
6	d16d7800daed208be02daa6b360a07d244e51...	718cb5971a3a7ad047eedd15e3c9dcea4de583...	visibleInUi	10

Figura 4.24: Dati relativi al messaggio d'inoltro di un messaggio AUTH ricevuto da un'introducee all'altro introducee in tabella MESSAGEMETADATA

Dagli id dei messaggi presenti nella tabella MESSAGEDEPENDENCIES e relativi al GROUPID corrispondente al CLIENTID org.briarproject.briar.introduction dell'utente locale è possibile ricostruire lo scambio di messaggi (dal più recente al più vecchio) che compongono la procedura di una introduction con uno specifico id di session (sessionId).

È necessario precisare che più introduction effettuate tra gli stessi due contatti (introducee) hanno lo stesso sessionId.

In Figura 4.25, è riportato un esempio di messaggi associati a una introduction con uno specifico sessionId.

	GROUPID	MessageID	MessageTimestamp	MessageMeta	DependencyID	DependencyTimestamp	DependencyMeta
1	d16d7800daed208be02...	9a0f0ca8263f63ab78f75...	1716475285022	local=10...	b9f28def257528b43a37b6e...	1716475068473	local=10...
2	d16d7800daed208be02...	b5ac42e906ccc89fff4b1...	1716475497424	local=10...	9a0f0ca8263f63ab78f75f40...	1716475285022	local=10...
3	d16d7800daed208be02...	81346cfe58a3dfbdcf7cf...	1716475506732	local=10...	b5ac42e906ccc89fff4b1283...	1716475497424	local=10...
4	d16d7800daed208be02...	afa4ad73963a1c5a5944...	1716475508535	local=10...	81346cfe58a3dfbdcf7cfda2...	1716475506732	local=10...
5	f2b02577af82e18e1716f...	3e8520766de861684e6...	1716475287480	local=10...	68f03eadb5664f2d1058c4f2...	1716475065570	local=10...
6	f2b02577af82e18e1716f...	b68c2bc5b9139d30522...	1716475499371	local=10...	3e8520766de861684e653cc...	1716475287480	local=10...
7	f2b02577af82e18e1716f...	52fef0c3e5def53f6b147...	1716475503553	local=10...	b68c2bc5b9139d305220ef...	1716475499371	local=10...
8	f2b02577af82e18e1716f...	a3d33370c3e5142ae2b...	1716475510850	local=10...	52fef0c3e5def53f6b147cfc2...	1716475503553	local=10...

Figura 4.25: Dati relativi ai messaggi scambiati durante la procedura di introduction in tabella MESSAGEDEPENDENCIES

Per fare ciò, per ogni GROUPID, partendo dal MESSAGEID con timestamp più recente (campo MessageTimestamp), bisogna:

1. Recuperare il DEPENDENCYID associato al MESSAGEID con timestamp più recente
2. Cercare il valore ottenuto al punto 1 nel campo MESSAGEID
3. Recuperare il DEPENDENCYID associato al MESSAGEID del punto 2
4. Cercare il valore ottenuto al punto 2 nel campo MESSAGEID

e così via. La ricostruzione termina quando la ricerca del DEPENDENCYID nel campo MESSAGEID fallisce.

4.3.2.2 - Introducee

Se l'aggiunta del contatto è avvenuta attraverso l'introduction da parte di un contatto terzo, nella tabella CONTACTS il contatto aggiunto risulterà non verificato (campo VERIFIED a False/0) esattamente come nel caso dell'aggiunta da lontano con procedura standard ma con ALIAS e HANDSHAKEPUBLICKEY non valorizzati.

	CONTACTID	AUTHORID	NAME	ALIAS	HANDSHAKEPUBLICKEY	LOCALAUTHORID	VERIFIED
1	1	12520c39b4fd62d2cbfc28c8d0d87d56ae89be1...	Amico1	AMICO_1	4f5ccab4506bb447e326106775b4fd5a1099a...	ee421ee3eb47eef9f483b43178331ead7a6f398...	0
2	2	7cce42ae2a59afb84af3d53d77ed509dbf9c2c...	Amico2	NULL	NULL	ee421ee3eb47eef9f483b43178331ead7a6f398...	0

Figura 4.26: Dati tabella CONTACTS

Le tuple presenti nella Figura 4.26 contengono tutti i contatti dell'utente locale ma il contatto con NAME 'Amico1' è stato aggiunto da lontano attraverso la procedura standard (attraverso un link) mentre il contatto con NAME 'Amico2' è stato aggiunto dall'utente locale attraverso l'accettazione di una richiesta di introduction.

Inoltre, nella tabella MESSAGES sarà presente un messaggio che fa riferimento al gruppo org.briarproject.briar.introduction relativo all'utente locale,

Come illustrato in Figura 4.27, tra i metadati (tabella MESSAGEMETADATA) di questo messaggio sarà presente la voce 'role' con valore '2101'.

Questo è un BDF in cui il '21' definisce il tipo di dato (INT) e '01' definisce il ruolo dell'utente locale nell'introduction (01 per introducee e 00 per introducer). Questo messaggio ha ulteriori metadati forensicamente importanti quali:

- requestTimestamp: contiene il momento in cui l'utente locale ha ricevuto la richiesta di introduzione dall'introducer
- introducer: contiene un BDF da cui è possibile estrarre il nome dell'introducer
- remote: contiene un BDF da cui è possibile ottenere il nome dell'altro introducee coinvolto in questa procedura di introduction

	CLIENTID	MESSAGEID	METAKEY	VALUE
1	org.briarproject.briar.introduction	e379d048971c7db9dc40708f1c4f1d3393e26a7...	introducer	6021014106416d69636f3151201b01d52c62b2...
2	org.briarproject.briar.introduction	e379d048971c7db9dc40708f1c4f1d3393e26a7...	local	70410f61636365707454696d657374616d7021...
3	org.briarproject.briar.introduction	e379d048971c7db9dc40708f1c4f1d3393e26a7...	remote	70410f61636365707454696d657374616d7021...
4	org.briarproject.briar.introduction	e379d048971c7db9dc40708f1c4f1d3393e26a7...	requestTimestamp	28000001902f57d2f0
5	org.briarproject.briar.introduction	e379d048971c7db9dc40708f1c4f1d3393e26a7...	role	2101
6	org.briarproject.briar.introduction	e379d048971c7db9dc40708f1c4f1d3393e26a7...	sessionId	51209effcdc74da175fed36dfd180ecf1bc46202...
7	org.briarproject.briar.introduction	e379d048971c7db9dc40708f1c4f1d3393e26a7...	state	2100

Figura 4.27: Dati relativi al messaggio di gestione introduction lato introducee in tabella MESSAGEMETADATA

Quando un utente riceve una richiesta di introduction, riceve un messaggio di richiesta introduction (Figura 4.28) con metadato 'messageType' uguale a '2100'. Tale messaggio ha anche il metadato availableToAnswer che può dare indicazioni sul fatto che l'utente locale abbia risposto o meno alla richiesta. Se availableToAnswer è uguale a 10 (False) allora l'utente locale ha risposto alla richiesta, se invece è uguale a 11 (True), l'utente locale non ha ancora risposto alla richiesta.

Inoltre, deserializzando il BDF presente nel campo RAW del messaggio di richiesta è possibile recuperare il nome dell'altro introducee e il messaggio opzionale che l'introducer ha scritto come invito dell'introduction (Sezione 3.1.3.10).

	CLIENTID	MESSAGEID	METAKEY	VALUE
1	org.briarproject.briar.introduction	897b430bb405fa84a181580422fe69862bcc4a...	availableToAnswer	10
2	org.briarproject.briar.introduction	897b430bb405fa84a181580422fe69862bcc4a...	local	10
3	org.briarproject.briar.introduction	897b430bb405fa84a181580422fe69862bcc4a...	messageType	2100
4	org.briarproject.briar.introduction	897b430bb405fa84a181580422fe69862bcc4a...	read	11
5	org.briarproject.briar.introduction	897b430bb405fa84a181580422fe69862bcc4a...	sessionId	51209effcdc74da175fed36dfd180ecf1bc46202...
6	org.briarproject.briar.introduction	897b430bb405fa84a181580422fe69862bcc4a...	timestamp	28000001902f52bed0
7	org.briarproject.briar.introduction	897b430bb405fa84a181580422fe69862bcc4a...	visibleInUi	11

Figura 4.28: Dati relativi al messaggio di richiesta di introduction lato introducee in tabella MESSAGEMETADATA

È inoltre possibile dedurre se l'utente locale e l'altro introducee hanno accettato o meno la richiesta di introduction recuperando rispettivamente il metadato 'messageType' del messaggio di risposta dell'utente locale e il metadato 'messageType' del messaggio d'inoltro della risposta dell'altro introducee da parte dell'introducer. Il messaggio di risposta dell'utente locale e quello dell'altro introducee possono essere distinti in quanto il primo ha metadato 'local' uguale a True (11) mentre il secondo ha metadato 'local' uguale a False (11).

Se il messageType ha valore '2101' allora l'introduction è stata accettata. Se, invece, il messageType ha valore '2102' allora l'introduction è stata rifiutata. Questi messaggi hanno gli stessi metadati dei messaggi di accettazione o rifiuto che sono stati trattati nella Sezione 4.3.2.1

Esattamente come per l'introducer, anche lato Introducee è di particolare interesse la tabella MESSAGESDEPENDENCIES. Dai dati contenuti in essa è possibile ricreare uno storico di tutte le richieste di introduction che l'utente ha ricevuto da ogni suo contatto e come l'altro introducee ha risposto a tali richieste.

Infatti partendo dall'ultima tupla a risalire è possibile ricostruire lo storico delle varie richieste di introduction che l'utente locale ha ricevuto (per la ricostruzione dello storico fare riferimento alla Sezione 4.3.2.1).

4.3.3 - Query

Le informazioni rilevanti per la funzionalità di introduction sono sicuramente:

- il recupero dei metadati dei messaggi inviati o ricevuti durante l'introduction
- La ricostruzione dello storico dei messaggi scambiati durante le procedure di introduction

Per recuperare i metadati dei messaggi relativi alla procedura di introduction è necessario eseguire la seguente query

```
SELECT g.CLIENTID, gm.VALUE, m.MESSAGEID, m.TIMESTAMP, md.METAKEY, md.VALUE
FROM
    GROUPS g JOIN GROUPEMETADATA gm USING(GROUPID)
    JOIN MESSAGES m USING(GROUPID) JOIN MESSAGEMETADATA md
USING(MESSAGEID)
WHERE CLIENTID = "org.briarproject.briar.introduction" AND gm.METAKEY = "contactId"
```

Dal risultato della query è possibile capire il tipo di messaggio, se il messaggio è locale (inoltro) o meno (messaggio ricevuto) e quale contatto è associato al messaggio.

È importante tenere conto che questa query non recupera il messaggio di gestione dell'introduction dove è definito il ruolo dell'utente relativamente all'introduction (per introducer Figura 4.16, per introducee Figura 4.27).

Per recuperare tale messaggio basterà:

- eseguire la query *SELECT * FROM MESSAGEMETADATA WHERE METAKEY = "role"*
- recuperare l'id del messaggio dal risultato della query del punto precedente
- Eseguire la query *SELECT * FROM MESSAGEMETADATA WHERE MESSAGEID = "id del messaggio recuperato al punto precedente"*

Per ricostruire invece lo storico dei messaggi scambiati durante le varie procedure di introduction effettuate dall'introducer è necessario eseguire la seguente query:

```
SELECT G.CLIENTID, gm.VALUE, md.MESSAGEID AS MessageID, msg_meta.TIMESTAMP
AS MessageTimestamp,msg_meta.MessageMeta,md.DEPENDENCYID AS
DependencyID,dep_meta.TIMESTAMP AS DependencyTimestamp,
dep_meta.DependencyMeta
FROM
GROUPS G JOIN GROUPMETADATA gm USING(GROUPID)
JOIN MESSAGEDEPENDENCIES md USING(GROUPID)
LEFT JOIN (
    SELECT MESSAGEID, TIMESTAMP,
        GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
    FROM MESSAGEMETADATA JOIN MESSAGES USING(MESSAGEID)
    GROUP BY MESSAGEID
) AS msg_meta ON md.MESSAGEID = msg_meta.MESSAGEID
LEFT JOIN (
    SELECT MESSAGEID, TIMESTAMP,
        GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS
    DependencyMeta
    FROM MESSAGEMETADATA JOIN MESSAGES USING(MESSAGEID)
    GROUP BY MESSAGEID
) AS dep_meta ON md.DEPENDENCYID = dep_meta.MESSAGEID
WHERE CLIENTID = "org.briarproject.briar.introduction" and gm.METAKEY = "contactId"
ORDER BY G.GROUPID, MessageTimestamp
```

Dal risultato di questa query si ottengono i vari messaggi inviati o ricevuti dai gruppi relativi al CLIENTID org.briarproject.briar.introduction e il contatto associato a ognuno di essi.

Partendo dal risultato della query ed eseguendo la procedura spiegata nella Sezione 4.3.2.1 relativamente alla tabella MESSAGEDEPENDENCIES è possibile ottenere lo storico di ogni richiesta di introduction.

Si tenga presente che se l'introducer invia più richieste di introduction tra gli stessi contatti, le varie richieste avranno un rapporto di dipendenza nella tabella MESSAGEDEPENDENCIES.

4.4 - Modifica dei contatti

4.4.1 - Esperimenti

Numero esperimento	Descrizione	Sotto-esperimenti
7	Modifica nome contatti lontani	1. A modifica il nome del contatto B
8	Modifica nome contatti vicini	1. A modifica il nome del contatto B

Tabella 4.4: Lista esperimenti modifica nome contatti

Come descritto nella Tabella 4.4, nel sotto-esperimento 1 dell'esperimento 7 e nel sotto-esperimento 1 dell'esperimento 8 è stata simulata rispettivamente la modifica del nome di un

contatto aggiunto da lontano facente parte della lista contatti dell'utente locale e la modifica del nome di un contatto aggiunto da vicino facente parte della lista contatti dell'utente locale.

Per entrambe le casistiche è stato utilizzato lo stesso file delle azioni.

```
1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,2
7 # Modifica del nome del contatto
8 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
9 #SLEEP
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.ViewGroup[1]/androidx.appcompat.widget.LinearLayoutCompat[1]/android.widget.ImageView[@content-desc='More options']
11 #SLEEP
12 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget.LinearLayout[5]/android.widget.LinearLayout[1]/android.widget.ImageView[@resource-id='org.briarproject.briar.android:id/submenuarrow']
13 #SLEEP
14 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.RelativeLayout[1]/android.widget.TextView[@resource-id='org.briarproject.briar.android:id/title']
15 #SLEEP
16 SendKeys,RES_ID:org.briarproject.briar.android:id/aliasEditText,TEXT:Piero
17 HideKeyboard
18 #SLEEP
19 Tap,RES_ID:org.briarproject.briar.android:id/setButton
20 #SLEEP
21 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
22 SLEEP,2
23 # Apertura menù laterale
24 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Open the navigation drawer']
25 # Sign-out in Briar
26 SLEEP,5
27 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
28 SLEEP,5
```

Figura 4.29: File delle azioni per il passo 6 dell'esperimento 2 e il passo 2 dell'esperimento 3

Nelle righe 10 e 12 della Figura 4.29 si apre il menù 'More options' e si seleziona la voce 'Contact' (Figura 2.12 della Sezione 2.2.3).

Nella riga 14 della Figura 4.29 si esegue il Tap sulla voce 'Change contact name' del menù (Figura 2.13 della Sezione 2.2.3).

Infine, nelle righe 16 e 19 della Figura 4.29 si inserisce il nuovo nome del contatto e si esegue il Tap sul tasto che permette di rendere effettiva la modifica del nome del contatto.

4.4.2 - Artefatti generati

La modifica del nome di un contatto presente nella lista dei contatti dell'utente locale ha la sola conseguenza di modificare il valore del campo ALIAS della tabella CONTACTS. Questo vale sia se il contatto di cui si è modificato il nome è stato aggiunto da lontano sia se il contatto è stato aggiunto da vicino.

4.4.3 - Query

Per ottenere il valore del campo ALIAS basta eseguire la query

```
SELECT * FROM CONTACTS
```


4.5.2 - Artefatti generati

La cancellazione di un contatto non genera nessun artefatto. In realtà comporta:

- La cancellazione del contatto nella tabella CONTACTS
- La cancellazione di tutti i gruppi associati al contatto che si sta cancellando nella tabella GROUPS
- La cancellazione di tutti i metadati relativi ai gruppi associati a quel contatto nella tabella GROUPEMETADATA
- La cancellazione di tutte le tuple relative ai gruppi associati a quel contatto nella tabella GROUPVISIBILITIES
- La cancellazione di tutti i messaggi inviati o ricevuti dai gruppi associati a quel contatto nella tabella MESSAGES
- La cancellazione di tutti i metadati dei messaggi inviati o ricevuti dai gruppi associati a quel contatto nella tabella MESSAGEMETADATA
- Se presenti, la cancellazione delle dipendenze dei messaggi inviati o ricevuti dai gruppi associati a quel contatto nella tabella MESSAGEDEPENDENCIES
- La cancellazione di tutti i messaggi inviati o ricevuti dai gruppi associati a quel contatto nella tabella STATUSES

Ciò avviene sia se il contatto eliminato era stato aggiunto nella lista contatti attraverso la procedura di aggiunta contatto da lontano sia se era stato aggiunto attraverso la procedura di aggiunta contatto da vicino.

4.6 - Messaggi privati

4.6.1 - Esperimenti

L'analisi della funzionalità di invio messaggi in chat privata tra due utenti ha richiesto diversi esperimenti in quanto nelle chat private di Briar è possibile inviare messaggi di testo, immagini ed emoji (Sezione 2.3). Si è quindi deciso di dedicare a ogni tipo di messaggio un esperimento a parte.

Inoltre, si è ritenuto fondamentale analizzare gli artefatti generati in caso di situazioni diverse di connettività (esperimento 11).

Infine, per ogni tipo di messaggio inviato (testo, immagine o emoji) si sono indagati anche gli artefatti generati dalle operazioni di cancellazione dei messaggi.

Numero esperimento	Descrizione	Sotto-esperimenti
11	Messaggio testuale inviato, con mittente e destinatario a turno offline	<ol style="list-style-type: none">1. A invia un messaggio a B quando B è offline2. B torna online e riceve il messaggio3. A prova a inviare un messaggio a B quando A è offline4. A torna online
12	Messaggio testuale inviato, entrambi gli utenti online	<ol style="list-style-type: none">1. A invia un messaggio testuale a B (A e B online)2. B invia un messaggio testuale ad A (A e B online)3. B cancella tutti i messaggi4. A cancella tutti i messaggi
13	Messaggio non testuale inviato, entrambi gli utenti online	<ol style="list-style-type: none">1. A invia un'immagine a B (A e B online)2. B invia un'immagine ad A (A e B online)3. B cancella tutti i messaggi4. A cancella tutti i messaggi
14	Messaggio con emoji inviato, entrambi gli utenti online	<ol style="list-style-type: none">1. A invia un'emoji a B (A e B online)2. B invia un'emoji ad A (A e B online)3. B cancella tutti i messaggi4. A cancella tutti i messaggi

Tabella 4.6: Lista esperimenti scambio messaggi in chat privata

Gli esperimenti 11, 12, 13 e 14 della Tabella 4.6 si concentrano sulle azioni di invio e cancellazione di messaggi di diverso tipo in condizioni diverse di connettività del mittente e del destinatario.

Sono stati utilizzati file delle azioni diversi per l'invio dei diversi tipi di messaggi da parte dell'utente oggetto di analisi (A).

4.6.1.1 - Messaggi di testo

Nei sotto-esperimenti 1 e 3 dell'esperimento 11 e nel sotto-esperimento 1 dell'esperimento 12, per simulare le azioni di invio di messaggi di testo, è stato utilizzato il seguente file delle azioni.


```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,50
7 # Invio immagine con testo
8 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
9 #SLEEP
10 Tap,RES_ID:org.briarproject.briar.android:id/material_target_prompt_view
11 SLEEP,2
12 Tap,XY:1017-2151
13 SLEEP,5
14 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Show roots']
15 SLEEP,2
16 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.ScrollView[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.TextView[1]
17 SLEEP,2
18 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[2]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.recyclerview.widget.RecyclerView[1]/android.widget.LinearLayout[1]/android.widget.RelativeLayout[1]/android.view.View[1]
19 SLEEP,2
20 SendKeys,XY:235-2155,TEXT:Immagine inviata da A
21 HideKeyboard
22 #SLEEP
23 Tap,XY:1015-2155
24 SLEEP,30
25 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
26 SLEEP,1
27 # Apertura menù laterale
28 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Open the navigation drawer']
29 # Sign-out in Briar
30 SLEEP,2
31 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
32 SLEEP,5

```

Figura 4.32: File delle azioni per l'invio di un'immagine con testo associato

Nella riga 8 della Figura 4.32 si esegue l'azione per poter entrare nella chat privata con il contatto a cui si vuole inviare l'immagine.

Dalla riga 12 alla riga 18 della Figura 4.32 si effettuano le azioni di Tap necessarie per la procedura di selezione dell'immagine dal proprio dispositivo.

Nelle righe 20 e 23 della Figura 4.32 si attuano le azioni che permettono di scrivere il testo associato all'immagine e l'invio dell'immagine con il testo.

Se non si volesse inviare un testo collegato all'immagine è sufficiente non inserire l'istruzione presente a riga 20 della Figura 4.32.

4.6.1.3 - Messaggi con emoji

Infine, per eseguire il sotto-esperimento 1 dell'esperimento 14 è stato necessario creare un ulteriore file delle azioni.

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,120
7 # Invio messaggio di testo con emoji
8 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
9 #SLEEP
10 Tap,RES_ID:org.briarproject.briar.android:id/material_target_prompt_view
11 SLEEP,2
12 Tap,XY:59-2143
13 SLEEP,2
14 Tap,XY:67-1685
15 HideKeyboard
16 SLEEP,2
17 Tap,XY:1015-2155
18 SLEEP,30
19 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
    et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android
    .widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
20 SLEEP,1
21 # Apertura menù laterale
22 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
    et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
    yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
    on[@content-desc='Open the navigation drawer']
23 # Sign-out in Briar
24 SLEEP,2
25 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
26 SLEEP,5

```

Figura 4.33: File delle azioni per l'invio di un messaggio di testo con emoji

Le righe 12 e 14 della Figura 4.33 si occupano di simulare le azioni di selezione dell'emoji da inserire nel messaggio.

La riga 17 della Figura 4.33, esattamente come per l'invio di un messaggio di testo o di un'immagine, esegue l'azione di Tap per l'invio del messaggio.

4.6.2 - Artefatti generati

4.6.2.1 - Messaggi di testo

Ogni messaggio inviato o ricevuto dall'utente locale è salvato nella tabella MESSAGES e associato al gruppo relativo al CLIENTID org.briarproject.briar.messaging collegato al contatto a cui è stato inviato il messaggio o da cui si è ricevuto il messaggio.

Ogni GROUPID associato al CLIENTID org.briarproject.briar.messaging ha dei metadati (Figura 4.34) forensicamente interessanti:

- contactId: contiene l'id del contatto a cui fa riferimento quel gruppo
- latestMessageTime: contiene il timestamp dell'ultimo messaggio inviato dal gruppo
- messageCount: contiene il numero di messaggi inviati o ricevuti dal gruppo
- unreadCount: contiene il numero di messaggi non ancora letti inviati o ricevuti dal gruppo

	CLIENTID	GROUPID	DESCRIPTOR	METAKEY	VALUE
1	org.briarproject.briar.messaging	4e98a8c143f3bbf3331f7dffe8b3e0d2da9a273...	605120115757c20dd384869137669e478b2ee6...	contactId	2101
2	org.briarproject.briar.messaging	4e98a8c143f3bbf3331f7dffe8b3e0d2da9a273...	605120115757c20dd384869137669e478b2ee6...	latestMessageTime	280000018ff1a81bd9
3	org.briarproject.briar.messaging	4e98a8c143f3bbf3331f7dffe8b3e0d2da9a273...	605120115757c20dd384869137669e478b2ee6...	messageCount	2102
4	org.briarproject.briar.messaging	4e98a8c143f3bbf3331f7dffe8b3e0d2da9a273...	605120115757c20dd384869137669e478b2ee6...	unreadCount	2100

Figura 4.34: Dati relativi al CLIENTID org.briarproject.briar.messaging tabella GROUPMETADATA

È importante ricordare che il campo VALUE della GROUPMETADATA contiene sempre un BDF e che quindi per ottenere i dati contenuti in esso è necessario tradurre il BDF secondo quanto spiegato nella Sezione 3.1.2.1.

Nella Figura 4.35 sono rappresentati due messaggi. Uno (riga 1) che contiene i dati di un messaggio inviato dall'utente locale e l'altro (riga 2) che contiene i dati di un messaggio ricevuto dall'utente locale.

MESSAGEID	GROUPID	TIMESTAMP	CLEANUPDEADLINE	CLEANUPTIMERDURATION	LENGTH	RAW
1 ab6a9fd25561...	4e98a8c143f3bbf3331f7dffe8b3e0d2da9a2730b9bf42226cc68c54f80afee9	1717746277337	NULL	NULL	98	4e98a...
2 bed0dab2a2a...	4e98a8c143f3bbf3331f7dffe8b3e0d2da9a2730b9bf42226cc68c54f80afee9	1717746203122	NULL	NULL	98	4e98a...

Figura 4.35: Dati tabella MESSAGES

Come si può notare entrambi i messaggi sono associati allo stesso GROUPID.

Di particolare interesse forense è il campo RAW di ogni messaggio, infatti il campo RAW è composto da tre informazioni concatenate:

- Il GROUPID del messaggio (32 byte)
- Il timestamp del messaggio espresso in esadecimale (8 byte)
È uguale al valore presente nel campo TIMESTAMP della tabella MESSAGES. L'unica differenza è che nel campo TIMESTAMP, la marca temporale è espresso in decimale mentre nel campo RAW è espressa in esadecimale
- Il BDF contenente diversi dati relativi al messaggio e la cui struttura è stata precedentemente spiegata nella Sezione 3.1.3.3

Dal BDF contenuto nel campo RAW è possibile recuperare:

- il tipo di messaggio (PRIVATE_MESSAGE = 0, ATTACHMENT = 1)
Il tipo di messaggio è 0 se il messaggio è un messaggio di testo ed è 1 se il messaggio è un messaggio che contiene un'immagine
- il testo del messaggio
- l'allegato del messaggio
Questo campo è valorizzato se il messaggio contiene un'immagine, altrimenti è NULL
- il timer di autodelete
Questo campo è valorizzato solo quando è stata attivata l'opzione 'Disappearing messages', altrimenti è NULL

Per comprendere meglio come estrarre queste informazioni facciamo un esempio.

Si prenda come valore del campo RAW di una tupla della tabella MESSAGES il valore

```
4e98a8c143f3bbf3331f7dffe8b3e0d2da9a2730b9bf42226cc68c54f80afee90000018ff1a6f9f260210041314d657373616767696f206469204120636f6e206d697474656e746520652064657374696e61746172696f206f6e6c696e6560800080
```

I primi 32 byte cioè 4e98a8c143f3bbf3331f7dffe8b3e0d2da9a2730b9bf42226cc68c54f80afee9 corrispondono al GROUPID che ha inviato o ricevuto il messaggio.

I successivi 8 byte cioè 0000018ff1a6f9f2 rappresentano il timestamp del messaggio espresso in esadecimale.

I restanti byte cioè

```
60210041314d657373616767696f206469204120636f6e206d697474656e746520652064657374696e61746172696f206f6e6c696e6560800080
```

costituiscono il BDF che contiene i dati del messaggio.

Traducendo il BDF secondo quanto spiegato nella Sezione 3.1.2.1 si ottengono i dati del messaggio.

Come riportato nelle Figure 4.36 e 4.37, ogni messaggio inviato o ricevuto in una chat privata ha i seguenti metadati ad esso associati:

- attachmentHeaders: contiene l'intestazione dell'allegato (se il messaggio è un messaggio di testo questo metadato avrà valore 6080 cioè un BDF vuoto, se il messaggio contiene un'immagine questo metadato conterrà l'intestazione dell'immagine cioè l'id del messaggio che contiene l'immagine e il formato dell'immagine)
- hasText: definisce se il messaggio contiene del testo (11 = True) o meno (10 = False)
- local: definisce se il messaggio è stato inviato dall'utente locale (11 = True) o è stato ricevuto (10 = False)
- messageType: esprime il tipo di messaggio (0 se è un messaggio di tipo PRIVATE_MESSAGE quindi di testo, 1 se il messaggio è di tipo ATTACHMENT quindi contiene un'immagine)
- read: definisce se il messaggio è stato letto (11 = True) o meno (10 = False)
- timestamp: il timestamp del messaggio

	MESSAGEID	METAKEY	VALUE
1	bed0dab2a2a736fafdd7c73d1802afdb42d6e3...	attachmentHeaders	6080
2	bed0dab2a2a736fafdd7c73d1802afdb42d6e3...	hasText	11
3	bed0dab2a2a736fafdd7c73d1802afdb42d6e3...	local	11
4	bed0dab2a2a736fafdd7c73d1802afdb42d6e3...	messageType	2100
5	bed0dab2a2a736fafdd7c73d1802afdb42d6e3...	read	11
6	bed0dab2a2a736fafdd7c73d1802afdb42d6e3...	timestamp	280000018ff1a6f9f2

Figura 4.36: Dati relativi a un messaggio inviato dall'utente locale in tabella MESSAGEMETADATA

	MESSAGEID	METAKEY	VALUE
1	ab6a9fd25561c79635fd79efb1574c0bc9d7887...	attachmentHeaders	6080
2	ab6a9fd25561c79635fd79efb1574c0bc9d7887...	hasText	11
3	ab6a9fd25561c79635fd79efb1574c0bc9d7887...	local	10
4	ab6a9fd25561c79635fd79efb1574c0bc9d7887...	messageType	2100
5	ab6a9fd25561c79635fd79efb1574c0bc9d7887...	read	11
6	ab6a9fd25561c79635fd79efb1574c0bc9d7887...	timestamp	280000018ff1a81bd9

Figura 4.37: Dati relativi a un messaggio ricevuto dall'utente locale in tabella MESSAGEMETADATA

È importante ricordare che il campo VALUE della MESSAGEMETADATA contiene sempre un BDF.

Ogni messaggio inviato o ricevuto dall'utente locale comporta l'aggiunta di una tupla nella tabella STATUSES.

Come spiegato nella Sezione 3.3.10, in questa tabella per ogni messaggio sono salvati dati che possono fornire informazioni sulla reale ricezione del messaggio da parte dell'utente remoto e sul fatto che il messaggio sia stato inviato o ricevuto dall'utente locale.

Infatti, i messaggi ricevuti hanno i campi EXPIRY, TXCOUNT e MAXLATENCY non valorizzati e il campo SEEN a 1, mentre i messaggi inviati hanno i campi EXPIRY, TXCOUNT e MAXLATENCY valorizzati e il campo SEEN a 1.

Inoltre, se l'utente locale o l'utente remoto non sono online quando l'utente locale invia un messaggio, il messaggio rimane con campi EXPIRY, TXCOUNT e MAXLATENCY non valorizzati e campo SEEN uguale a 0 fino a quando entrambi gli utenti non tornano online.

Nel momento in cui entrambi gli utenti tornano online, se il messaggio in questione è stato inviato dall'utente locale allora i campi EXPIRY, TXCOUNT e MAXLATENCY relativi al messaggio saranno valorizzati e il valore del campo SEEN sarà settato a 1.

Se invece, entrambi gli utenti tornano online ma il messaggio in questione è stato inviato dall'utente remoto allora i campi EXPIRY, TXCOUNT e MAXLATENCY relativi al messaggio rimarranno non valorizzati e il valore del campo SEEN sarà settato a 1.

	MESSAGEID	CONTACTID	GROUPID	TIMESTAMP	LENGTH	STATE	GROUPSHARED	MESSAGESHARED	DELETED	ACK	SEEN	REQUESTED	EXPIRY	TXCOUNT	MAXLATENCY
1	bed0dab2a2a736fafdd7c73d1...	1	4e98a8c143f3bbf3331f...	1717746203122	98	3	1	1	0	0	1	0	1717746264547	1	30000
2	ab6a9fd25561c79635fd79efb1...	1	4e98a8c143f3bbf3331f...	1717746277337	98	3	1	0	0	0	1	0	0	0	NULL

Figura 4.38: Dati tabella STATUSES

Nella Figura 4.38 si possono osservare un esempio di messaggio inviato dall'utente locale (riga 1) e un esempio di messaggio ricevuto dall'utente locale (riga 2).

4.6.2.2 - Immagini

4.6.2.2.1 - Gestione immagini in database h2

Le immagini inviate o ricevute dall'utente locale sono salvate nel database h2 come sequenze di esadecimali.

Per inserire nel database le immagini inviate o ricevute dall'utente, Briar utilizza la classe ScriptCommand.java che gestisce i blob nel seguente modo:

1. Crea una tabella SYSTEM_LOB_STREAM che è composta dai seguenti campi:
 - ID che contiene l'id auto incrementale dell'immagine
 - PART che contiene il numero della parte dell'immagine con un certo ID
 - CDATA che è uguale a NULL
 - BDATA che contiene parte del blob.
2. Crea un ALIAS SYSTEM_COMBINE_BLOB per la funzione org.h2.command.dml.ScriptCommand.combineBlob. Questa funzione concatena tutte le parti del blob che hanno campo ID uguale al parametro passato alla funzione combineBlob.
3. Suddivide l'immagine in sotto parti e salva ognuna di queste nel campo BDATA di una tupla della tabella SYSTEM_LOB_STREAM. Tutte le tuple che contengono sotto parti di una stessa immagine hanno lo stesso valore nel campo ID.
4. Nel campo RAW della tabella MESSAGES relativamente al messaggio che contiene l'immagine richiama l'alias SYSTEM_COMBINE_BLOB associato alla funzione java org.h2.command.dml.ScriptCommand.combineBlob con parametro uguale all'id dell'immagine (corrisponde al valore presente nel campo ID della tabella SYSTEM_LOB_STREAM). La funzione richiamata dall'alias ricompone il blob nella sua interezza salvandolo nel campo RAW del messaggio.
5. Cancella la tabella SYSTEM_LOB_STREAM e richiama l'alias SYSTEM_COMBINE_BLOB con parametro -1 così da pulire il buffer utilizzato per ricomporre il blob.
6. Cancella gli alias associati alle funzioni java.

Per comprendere meglio la procedura appena analizzata facciamo un esempio in cui andiamo ad associare a ogni punto dell'elenco precedente le query SQL eseguite nel database.

Esempio

1. CREATE TABLE IF NOT EXISTS SYSTEM_LOB_STREAM(ID INT NOT NULL, PART INT NOT NULL, CDATA VARCHAR, BDATA BINARY);
CREATE PRIMARY KEY SYSTEM_LOB_STREAM_PRIMARY_KEY ON SYSTEM_LOB_STREAM(ID, PART);
2. CREATE ALIAS IF NOT EXISTS SYSTEM_COMBINE_CLOB FOR "org.h2.command.dml.ScriptCommand.combineClob";
CREATE ALIAS IF NOT EXISTS SYSTEM_COMBINE_BLOB FOR "org.h2.command.dml.ScriptCommand.combineBlob";
3. INSERT INTO SYSTEM_LOB_STREAM VALUES(0, 0, NULL, 'eb54fff1192fd13');
INSERT INTO SYSTEM_LOB_STREAM VALUES(0, 1, NULL, 'eb54fff1192fd13');
INSERT INTO SYSTEM_LOB_STREAM VALUES(0, 2, NULL, 'eb54fff1192fd13');
INSERT INTO SYSTEM_LOB_STREAM VALUES(0, 3, NULL, 'eb54fff1192fd13');
INSERT INTO SYSTEM_LOB_STREAM VALUES(0, 4, NULL, 'eb54fff1192fd13');
INSERT INTO SYSTEM_LOB_STREAM VALUES(0, 5, NULL, 'eb54fff1192fd13');
INSERT INTO SYSTEM_LOB_STREAM VALUES(0, 6, NULL, 'eb54fff1192fd13');
INSERT INTO SYSTEM_LOB_STREAM VALUES(0, 7, NULL, 'eb54fff1192fd13');
4. INSERT INTO PUBLIC.MESSAGES(MESSAGEID, GROUPID, TIMESTAMP, STATE, SHARED, TEMPORARY, CLEANUPTIMERDURATION, CLEANUPDEADLINE, LENGTH, RAW) VALUES (X'd35c890171633ac373034ca67c097c98d9d2d0d2a2bb35f05c2b9fd0971a9087', X'eb54fff1192fd13fc2cd0581aa292106b2dfdd55c43d039d1cfe393a7bb0b462', 1717754129935, 3, TRUE, FALSE, NULL, NULL, 31616, SYSTEM_COMBINE_BLOB(0))

Questa funzione prende tutte le tuple della tabella SYSTEM_LOB_STREAM che hanno valore del campo ID uguale a 0 (perché nel campo RAW della tabella MESSAGES SYSTEM_COMBIN_BLOB è stato richiamato con il parametro 0). Successivamente concatena il valore del campo BDATA di tutte le tuple della tabella SYSTEM_LOB_STREAM con l'indice 0 e ricompone l'immagine completa.

5. DROP TABLE IF EXISTS SYSTEM_LOB_STREAM;
CALL SYSTEM_COMBINE_BLOB(-1);
6. DROP ALIAS IF EXISTS SYSTEM_COMBINE_CLOB;
DROP ALIAS IF EXISTS SYSTEM_COMBINE_BLOB;

4.6.2.2.2 - Analisi dati immagini

Quando un'utente invia un messaggio con un'immagine sono inseriti due messaggi nella tabella MESSAGES:

- uno che gestisce l'immagine
- uno che contiene il possibile testo opzionale che accompagna l'immagine.

	MESSAGEID	GROUPID	TIMESTAMP	CLEANUPTIMERDURATION	CLEANUPDEADLINE	LENGTH	RAW
1	d35c890171633ac373034ca67c097c98d9d2d0...	eb54fff1192fd13fc2cd0581aa292106b2dfdd55c...	1717754129935	NULL	NULL	31616	eb54fff1192fd13fc2cd0581aa292106b2dfdd55...
2	0a3ec1399fa25251096659992c850c41c5093db...	eb54fff1192fd13fc2cd0581aa292106b2dfdd55c...	1717754134117	NULL	NULL	118	eb54fff1192fd13fc2cd0581aa292106b2dfdd55...
3	2e9920c8dd98d10d538cad91fe2786ab34b383...	eb54fff1192fd13fc2cd0581aa292106b2dfdd55c...	1717754255181	2419200000	NULL	31649	eb54fff1192fd13fc2cd0581aa292106b2dfdd55...
4	4c717021794058c3ab30f9af66f4774c6af48fa1...	eb54fff1192fd13fc2cd0581aa292106b2dfdd55c...	1717754269814	NULL	NULL	118	eb54fff1192fd13fc2cd0581aa292106b2dfdd55...

Figura 4.39: Dati nella tabella MESSAGES

Nella Figura 4.39, il secondo e il quarto messaggio (riga 2 e 4) sono i messaggi di testo che accompagnano rispettivamente i messaggi con immagine che sono salvati nella riga 1 e 3.

Il messaggio che contiene il possibile testo è identico a un qualsiasi messaggio di testo visto nella Sezione 4.6.2.1.

Il campo RAW del messaggio che contiene l'immagine, anche se, come un messaggio di testo, è composto da GROUPLD, timestamp e BDF, ha alcune differenze sostanziali.

In primis, come spiegato nella Sezione 3.1.3.3, il BDF del messaggio che contiene l'immagine include il tipo di messaggio uguale a 1 (ATTACHMENT perché è un allegato) e il formato del file di immagine. In secondo luogo, concatenato al BDF vi è l'immagine inviata espressa come sequenza di esadecimali.

In linea generale è possibile distinguere un messaggio che contiene un'immagine da un messaggio di testo basandosi sul valore del campo LENGTH. Infatti un messaggio che contiene un'immagine è probabile che abbia un valore del campo LENGTH molto più grande del valore del campo LENGTH di un messaggio di testo.

Ogni messaggio che contiene un'immagine ha i seguenti metadati associati:

- contentType: contiene il formato dell'immagine
- descriptorLength: contiene la lunghezza del descrittore dell'immagine che è composto dal tipo di messaggio e dal formato dell'immagine
- local: definisce se il messaggio è locale (11 = True) o meno (10 = False)
- messageType: contiene il tipo di messaggio che in caso di messaggio con immagine sarà sempre ATTACHMENT = 1,
- timestamp: il timestamp del messaggio

In Figura 4.40 si riporta un esempio di metadati relativi a un'immagine inviata dall'utente locale, mentre in Figura 4.41 è illustrato un esempio di metadati relativi a un'immagine ricevuta dall'utente locale.

	MESSAGEID	METAKEY	VALUE
1	d35c890171633ac373034ca67c097c98d9d2d0...	contentType	410a696d6167652f6a706567
2	d35c890171633ac373034ca67c097c98d9d2d0...	descriptorLength	2110
3	d35c890171633ac373034ca67c097c98d9d2d0...	local	11
4	d35c890171633ac373034ca67c097c98d9d2d0...	messageType	2101
5	d35c890171633ac373034ca67c097c98d9d2d0...	timestamp	280000018ff21fee0f

Figura 4.40: Dati relativi a un messaggio con immagine inviato in tabella MESSAGEMTADATA

	MESSAGEID	METAKEY	VALUE
1	2e9920c8dd98d10d538cad91fe2786ab34b383...	contentType	410a696d6167652f6a706567
2	2e9920c8dd98d10d538cad91fe2786ab34b383...	descriptorLength	2110
3	2e9920c8dd98d10d538cad91fe2786ab34b383...	local	10
4	2e9920c8dd98d10d538cad91fe2786ab34b383...	messageType	2101
5	2e9920c8dd98d10d538cad91fe2786ab34b383...	timestamp	280000018ff221d74d

Figura 4.41: Dati relativi a un messaggio con immagine ricevuto in tabella MESSAGEMETADATA

Come già enunciato, ogni messaggio contenente un'immagine è accompagnato da un messaggio di testo.

Ogni messaggio di testo associato a un messaggio con immagine ha gli stessi metadati associati a un messaggio di testo (Sezione 4.6.2.1). A differenza dei messaggi di testo analizzati nella Sezione 4.6.2.1, questi messaggi di testo hanno il metadato 'attachmentHeaders' valorizzato.

In Figura 4.42 e in Figura 4.43 si riportano rispettivamente un esempio di messaggio di testo associato a un messaggio con immagine inviato dall'utente locale e un esempio di messaggio di testo associato a un messaggio con immagine ricevuto dall'utente locale.

	MESSAGEID	METAKEY	VALUE
1	0a3ec1399fa25251096659992c850c41c5093db...	attachmentHeaders	60605120d35c890171633ac373034ca67c097c98d9d2d0d2a2bb35f05c2b9fd0971a9087410a696d6167652f6a7065678080
2	0a3ec1399fa25251096659992c850c41c5093db...	hasText	11
3	0a3ec1399fa25251096659992c850c41c5093db...	local	11
4	0a3ec1399fa25251096659992c850c41c5093db...	messageType	2100
5	0a3ec1399fa25251096659992c850c41c5093db...	read	11
6	0a3ec1399fa25251096659992c850c41c5093db...	timestamp	280000018ff21ffe65

Figura 4.42: Dati relativi a un messaggio di testo associato all'immagine inviata in tabella MESSAGEMETADATA

	MESSAGEID	METAKEY	VALUE
1	4c717021794058c3ab30f9af66f4774c6af48fa1...	attachmentHeaders	606051202e9920c8dd98d10d538cad91fe2786ab34b383422a3fa87e01fb6666b3615658410a696d6167652f6a7065678080
2	4c717021794058c3ab30f9af66f4774c6af48fa1...	hasText	11
3	4c717021794058c3ab30f9af66f4774c6af48fa1...	local	10
4	4c717021794058c3ab30f9af66f4774c6af48fa1...	messageType	2100
5	4c717021794058c3ab30f9af66f4774c6af48fa1...	read	11
6	4c717021794058c3ab30f9af66f4774c6af48fa1...	timestamp	280000018ff2221076

Figura 4.43: Dati relativi a un messaggio di testo associato all'immagine ricevuta in tabella MESSAGEMETADATA

Se l'immagine è stata ricevuta dall'utente analizzato, per il messaggio dell'immagine è settato un timer di 28 giorni (campo CLEANUPTIMERDURATION in MESSAGES); scaduto il quale l'immagine è cancellata.

Il timer definisce per quanto tempo conservare l'immagine in attesa di ricevere il messaggio di testo associato all'immagine (il messaggio di testo è inviato anche se non è inserito nessun testo di accompagnamento all'immagine).

Il timer si ferma appena l'utente locale riceve il messaggio privato associato a quell'immagine.

Anche i due messaggi inviati quando l'utente locale invia o riceve un messaggio contenente un'immagine sono associati a un gruppo relativo al CLIENTID org.briarproject.briar.messaging a cui sono collegati gli stessi metadati visti per il gruppo nella Sezione 4.6.1.

Esattamente come per l'invio di un messaggio di testo, anche nel caso di invio o ricezione di un'immagine, per ogni messaggio inviato o ricevuto (sia per i messaggi contenenti l'immagine sia per i messaggi di testo associati all'immagine) è aggiunta una tupla nella tabella STATUSES. Ogni messaggio se inviato o ricevuto avrà le stesse caratteristiche illustrate rispettivamente per i messaggi inviati o per i messaggi ricevuti nella Sezione 4.6.1.

4.6.2.3 - Messaggi con emoji

I messaggi che contengono emoji sono gestiti esattamente come i messaggi di testo analizzati nella Sezione 4.6.1. Più nello specifico l'emoji è salvata in esadecimale come testo del messaggio nel BDF presente nel campo RAW della tabella MESSAGES.

Per poterla tradurre basta eseguire la conversione dell'emoji da codice esadecimale a UTF-8.

4.6.3 - Query

Per poter ottenere tutti i messaggi inviati e ricevuti da tutti i CLIENTID org.briarproject.briar.messaging bisogna eseguire la seguente query:

```

SELECT g.CLIENTID, g_meta.GROUPID, g_meta.GroupMeta, m.MESSAGEID AS
MessageID,
msg_meta.TIMESTAMP AS MessageTimestamp,msg_meta.MessageMeta
FROM GROUPS g
JOIN (
    SELECT GROUPID,GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) as
GroupMeta
    FROM GROUPS JOIN GROUPMETADATA gm USING(GROUPID)
    GROUP BY GROUPID
) AS g_meta ON g.GROUPID = g_meta.GROUPID
JOIN MESSAGES m USING(GROUPID)
JOIN (
    SELECT MESSAGEID, TIMESTAMP,
    GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
    FROM MESSAGES JOIN MESSAGEMETADATA USING(MESSAGEID)
    GROUP BY MESSAGEID
) AS msg_meta ON m.MESSAGEID = msg_meta.MESSAGEID
WHERE CLIENTID = "org.briarproject.briar.messaging"
ORDER BY G.GROUPID

```

Come si può osservare in Figura 4.44, tale query restituisce una tupla per ogni messaggio associato a un gruppo relativo al CLIENTID org.briarproject.briar.messaging e, per ogni messaggio, i metadati ad esso associati. Inoltre restituisce i metadati del gruppo associato a ogni messaggio.

	CLIENTID	GROUPID	GroupMeta	MessageID	MessageTimestamp	MessageMeta
1	org.briarproject.briar.messaging	4e98a8c143f3bbf3331f7...	contactId=2101 latestMessageTime=280000018ff1a81bd9 messageCount=2102 unreadCount=2100	ab6a9fd25561c79635fd79efb1574c0bc9d7887...	1717746277337	attachmentHeaders=6080 hasText=11 local=10 messageType=2100 read=11 timestamp=280000018ff1a81bd9
2	org.briarproject.briar.messaging	4e98a8c143f3bbf3331f7...	contactId=2101 latestMessageTime=280000018ff1a81bd9 messageCount=2102 unreadCount=2100	bed0dab2a2a736fafdd7c73d1802afdb42d6e3...	1717746203122	attachmentHeaders=6080 hasText=11 local=11 messageType=2100 read=11 timestamp=280000018ff1a6f9f2

Figura 4.44: Risultato query

È possibile ottenere i messaggi relativi a uno specifico gruppo o i messaggi con uno specifico metadato o uno specifico timestamp andando semplicemente ad aggiungere una condizione nella clausola WHERE.

4.6.4 - Correlazione dati database con altri file

Quando un utente invia un messaggio contenente un'emoji il file variant-emojimanager.xml (Sezione 3.5) è modificato. Se il file non esiste viene creato.

Prima di procedere bisogna precisare che si definisce sessione di una chat privata la procedura composta dall'entrata nella chat privata con un determinato contatto seguita dalla scrittura o meno di uno o più messaggi seguita a sua volta dall'uscita dalla chat privata con quel contatto.

Quando un utente locale durante una sessione invia un'emoji, nel file xml è aggiunto un elemento con chiave "variant-emojis" e valore uguale all'emoji inviata.

Se l'utente invia o più emoji nella stessa sessione con un contatto o una o più emoji in sessioni diverse con contatti diversi (cioè sessioni relative a contatti diversi), il valore della chiave "variant-

emojis" sarà composto da tutte le emoji inviate (nella stessa sessione o in sessioni diverse) separate dalla tilde.

Se l'utente locale entra nella conversazione con un contatto a cui durante la sessione precedente aveva inviato un'emoji, l'elemento "variant-emojis" è eliminato. Questo avviene anche se il valore della chiave "variant-emojis" contiene emoji inviate in sessioni con altri contatti.

Quindi da questo file è possibile recuperare le emoji inviate solo se l'utente dopo l'invio non è più rientrato nelle conversazioni private in cui ha inviato le emoji.

4.7 - Opzione disappearing messages

4.7.1 - Esperimenti

Lo studio della funzionalità di 'Disappearing messages' ha richiesto un solo esperimento in modo da fornire gli artefatti generati dall'utilizzo della chat privata mentre l'opzione è attiva. Lo studio degli artefatti generati mentre l'opzione è disattivata è effettuabile utilizzando gli artefatti generati durante gli esperimenti riportati in Tabella 4.6 della Sezione 4.6.1.

Numero esperimento	Descrizione	Sotto-esperimenti
15	Messaggi inviati in chat con opzione disappearing messages attivata	<ol style="list-style-type: none"> 1. A setta l'opzione di disappearing messages nella chat privata con B 2. A invia un messaggio testuale a B 3. B invia un messaggio testuale ad A 4. A invia un'immagine a B 5. B invia un'immagine ad A

Tabella 4.7: Lista esperimenti messaggi con opzione disappearing messages abilitata

Per simulare le azioni descritte nei sotto-esperimenti 2 e 4 dell'esperimento 15 si sono utilizzate le stesse istruzioni presenti nei file delle azioni in Figura 4.31 della Sezione 4.6.1.1 e 4.32 della Sezione 4.6.1.2, mentre per simulare le azioni del sotto-esperimento 1 si sono utilizzate le seguenti istruzioni.

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,50
7 # Abilitazione opzione 'disappearing messages'
8 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
9 #SLEEP
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
    dget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/an-
    droid.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.appcompat.widget.LinearLayoutCompat[1]/android.wi-
    dget.ImageView[@content-desc='More options']
11 #SLEEP
12 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget.
    LinearLayout[2]/android.widget.LinearLayout[1]/android.widget.RelativeLayout[1]/android.widget.TextView[@resource-
    id='org.briarproject.briar.android:id/title']
13 #SLEEP
14 Tap,RES_ID:org.briarproject.briar.android:id/switchDisappearingMessages

```

Figura 4.45: Istruzioni per abilitare l'opzione di disappearing messages

Nella riga 8 della Figura 4.45 si esegue l'azione che permette di entrare nella chat privata con il contatto.

Nelle righe 10 e 12 della Figura 4.45 sono eseguite le azioni di Tap necessarie per accedere alla configurazione dell'opzione 'Disappearing messages' (Figura 2.21 della Sezione 2.3.2).

Infine, nella riga 14 della Figura 4.45 si esegue il Tap che abilita l'opzione se questa è disabilitata o disabilita l'opzione se questa è abilitata.

4.7.2 - Artefatti generati

Quando l'utente locale abilita l'opzione Disappearing messages in una conversazione con un suo contatto sono creati diversi artefatti all'interno del database dell'utente locale.

Innanzitutto, vengono impostati i seguenti metadati del gruppo associato al CLIENTID org.briarproject.briar.autodelete, relativo al contatto della conversazione in cui l'opzione è abilitata:

- autoDeletePreviousTimer: rappresenta il timer precedente di autodelete
- autoDeleteTimer: contiene il timer in millisecondi scaduto il quale i messaggi di quel gruppo saranno cancellati
- autoDeleteTimestamp: contiene il timestamp di quando è stato settato l'autodelete

Per comprendere meglio il metadato autoDeleteTimer supponiamo che tale metadato abbia valore '24240c8400'. Questo dato è un BDF in cui 24 corrisponde a INT_32 e i restanti byte in esadecimale rappresentano il valore del timer in millisecondi. Basta convertire gli ultimi 4 byte (8 caratteri esadecimali) da esadecimale in decimale per ottenere i millisecondi del timer.

Quando l'opzione 'Disappearing messages' è attivata, ogni messaggio inviato o ricevuto viene comunque salvato nella tabella MESSAGES, esattamente come avviene quando l'opzione è disattivata. Tuttavia, in questo caso, i messaggi di testo (Figura 4.46) presentano i campi CLEANUPTIMERDURATION e CLEANUPDEADLINE della tabella MESSAGES valorizzati. Inoltre, l'elemento autoDeleteTimer, presente nel campo RAW del messaggio, sarà anch'esso valorizzato.

MESSAGEID	GROUPID	TIMESTAMP	CLEANUPTIMERDURATION	CLEANUPDEADLINE	LENGTH	RAW
1 d748a50765d671a410a...	0b711d20a96fd5dc007091...	1718002300881	604800000	1718607108765	109	0b711d20a96fd5dc0070911a...

Figura 4.46: Messaggio di testo con opzione 'Disappearing messages' attiva

I messaggi che contengono immagini hanno il campo CLEANUPDEADLINE non valorizzato e il campo CLEANUPTIMERDURATION valorizzato se si tratta di immagini ricevute (Figura 4.47), oppure non valorizzato se si tratta di immagini inviate (Figura 4.48).

MESSAGEID	GROUPID	TIMESTAMP	CLEANUPTIMERDURATION	CLEANUPDEADLINE	LENGTH	RAW
1 b35107af4745865ac783...	0b711d20a96fd5dc0070...	1718002355240	2419200000	NULL	31649	0b711d20a96fd5dc0070...

Figura 4.47: Messaggio contenente immagine ricevuto con opzione 'Disappearing messages' attiva

MESSAGEID	GROUPID	TIMESTAMP	CLEANUPTIMERDURATION	CLEANUPDEADLINE	LENGTH	RAW
1 6955b74df08f8c46558...	0b711d20a96fd5dc00709...	1718002344269	NULL	NULL	31616	0b711d20a96fd5dc0...

Figura 4.48: Messaggio contenente immagine inviata con opzione 'Disappearing messages' attiva

Il campo CLEANUPTIMERDURATION contiene il tempo in millisecondi scaduto il quale il messaggio è cancellato. È uguale al valore del metadato autoDeleteTimer.

Il campo CLEANUPDEADLINE, valorizzato solo nei messaggi di testo quando l'opzione 'Disappearing messages' è attiva, contiene il timestamp di scadenza del messaggio. Una volta raggiunto quel timestamp il messaggio è cancellato.

La struttura del body dei messaggi di testo o contenenti immagini, così come la gestione di un messaggio con immagine e testo associato, rimane invariata rispetto a quanto descritto nella Sezione 4.6.2.

I metadati dei messaggi di testo corrispondono a quelli già osservati nella Sezione 4.6.2.1, includendo campi come read, messageType, attachmentHeaders, hasText, local e timestamp. Analogamente, i metadati dei messaggi contenenti immagini sono identici a quelli analizzati nella Sezione 4.6.2.2, comprendendo descriptorLength, messageType, contentType, local e timestamp.

Quando l'opzione 'Disappearing messages' è attiva, ai messaggi di testo normali e a quelli con immagini viene aggiunto il metadato autoDeleteTimer, che corrisponde al valore del metadato autoDeleteTimer del gruppo che ha inviato o ricevuto il messaggio.

4.7.3 - Query

È possibile recuperare i messaggi di testo (siano essi messaggi di testo normali o messaggi di testo di accompagnamento a un'immagine) con l'opzione 'Disappearing messages' attivata eseguendo la seguente query:

```
SELECT m.*, msg_meta.MessageMeta
FROM MESSAGES m JOIN (
    SELECT MESSAGEID, TIMESTAMP,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
    FROM MESSAGES JOIN MESSAGEMETADATA USING(MESSAGEID)
    GROUP BY MESSAGEID
) AS msg_meta ON m.MESSAGEID = msg_meta.MESSAGEID
WHERE msg_meta.MessageMeta LIKE "%autoDeleteTimer%"
```

4.8 - Invio via removable drive

4.8.1 - Esperimenti

Per analizzare la funzionalità di invio mediante removable drive si è deciso di effettuare due esperimenti:

- uno che effettuasse l'invio mediante removable drive (esperimento 9.6)
- uno che effettuasse la ricezione mediante removable drive (esperimento 9.7)

Numero esperimento	Descrizione	Sotto-esperimenti
16	Messaggi inviati via removable drive	1. Disconnessione di A dalla rete wifi e dalla rete tor, A invia un messaggio a B, A invia un'immagine a B, A crea un gruppo privato e invita B, A crea un forum e invita B, A scrive sul suo blog e A invia tutto via removable drive a B
17	Messaggi ricevuti via removable drive	1. B invia un messaggio ad A, B invia un'immagine ad A, B crea un gruppo privato e invita A, B crea un forum e invita A, B scrive sul suo blog, B invia tutto via removable drive ad A e A riceve tutto via removable drive

Tabella 4.8: Lista esperimenti scambio messaggi in chat privata via removable drive

Negli esperimenti 16 e 17 sono stati effettuati gli inviti a un gruppo privato, a un forum e la scrittura sul proprio blog perché gli inviti a gruppi privati e forum e i post pubblicati sul blog dell'utente locale sono messaggi che possono essere inviati o ricevuti mediante removable drive.

Come anticipato prima, nell'esperimento 16 ci si è concentrati sulla funzionalità di invio mediante removable drive. Più nello specifico si sono simulate le azioni di invio messaggio di testo, di invio immagine, di invito a un gruppo privato, di invito a un forum, di scrittura post sul proprio blog e infine invio di tutti i messaggi via removable drive.

Nell'esperimento 17 l'utente remoto ha effettuato lo stesso sotto-esperimento eseguito nell'esperimento 16 dall'utente locale e quest'ultimo ha solo eseguito l'import del file che contiene i messaggi inviati dall'utente remoto.

Per inviare tutti i messaggi non ancora inviati in una conversazione mediante removable drive, nel sotto-esperimento 1 dell'esperimento 16 sono state simulate le azioni che l'utente deve effettuare. Tali azioni sono riportate nella Figura 4.49.

Si specifica che nelle Figure 4.49 e 4.50 è riportata solo la parte del file delle azioni che contiene le istruzioni per effettuare rispettivamente l'invio e la ricezione dei messaggi mediante removable drive.

```
1 # Invio dei messaggi via removable drive
2 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
3 #SLEEP
4 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.w
  idget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/
  android.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.appcompat.widget.LinearLayoutCompat[1]/android.
  widget.ImageView[@content-desc='More options']
5 SLEEP,2
6 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget
  .LinearLayout[4]/android.widget.LinearLayout[1]/android.widget.ImageView[@resource-id='org.briarproject.briar.an-
  droid:id/submenuarrow']
7 SLEEP,2
8 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget
  .LinearLayout[2]/android.widget.LinearLayout[1]/android.widget.RelativeLayout[1]/android.widget.TextView[@resour-
  ce-id='org.briarproject.briar.android:id/title']
9 SLEEP,2
10 Tap,RES_ID:org.briarproject.briar.android:id/sendButton
11 SLEEP,2
12 Tap,RES_ID:org.briarproject.briar.android:id/fileButton
13 SLEEP,5
14 Tap,XY:920-2147
15 SLEEP,2
16 Tap,RES_ID:org.briarproject.briar.android:id/button
```

Figura 4.49: Istruzioni per invio messaggi via removable drive

Nella riga 2 della Figura 4.49 è eseguita l'azione di accesso alla conversazione con il contatto a cui si vogliono inviare i messaggi via removable drive.

Nella riga 4 della Figura 4.49 si apre il menù della conversazione.

Nelle righe 6 e 8 della Figura 4.49 si accede alla sezione di invio o ricezione dei messaggi mediante removable drive (Figura 2.18 e 2.19 della Sezione 2.3.1).

Nella riga 10 della Figura 4.49 si esegue il Tap sul tasto di invio dati (Figura 2.20 della Sezione 2.3.1) in modo da procedere con l'invio mediante removable drive.

Infine, nelle righe 12, 14 e 16 della Figura 4.49 è creato il file contenente i messaggi che si inviano via removable drive, suddetto file è salvato sul dispositivo ed è data conferma di conclusione procedura di invio mediante removable drive.

Per simulare, invece la ricezione dei messaggi mediante removable drive nel sotto-esperimento 1 dell'esperimento 17 sono state eseguite le azioni riportate nella Figura 4.50.

```
1 # Ricezione di messaggi via removable drive
2 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
3 SLEEP,5
4 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.appcompat.widget.LinearLayoutCompat[1]/android.widget.ImageView[@content-desc='More options']
5 #SLEEP
6 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget.LinearLayout[4]/android.widget.LinearLayout[1]/android.widget.ImageView[@resource-id='org.briarproject.briar.android:id/submenuarrow']
7 SLEEP,2
8 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget.LinearLayout[2]/android.widget.LinearLayout[1]/android.widget.RelativeLayout[1]/android.widget.TextView[@resource-id='org.briarproject.briar.android:id/title']
9 SLEEP,2
10 Tap,RES_ID:org.briarproject.briar.android:id/receiveButton
11 SLEEP,2
12 Tap,RES_ID:org.briarproject.briar.android:id/fileButton
13 SLEEP,2
14 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Show roots']
15 SLEEP,2
16 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.ScrollView[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.TextView[@resource-id='android:id/title']
17 SLEEP,2
18 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[2]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.recyclerview.widget.RecyclerView[1]/androidx.cardview.widget.CardView[1]/android.view.View[1]
19 SLEEP,2
20 Tap,RES_ID:org.briarproject.briar.android:id/button
```

Figura 4.50: Istruzioni per ricezione messaggi via removable drive

Nella riga 2 della Figura 4.50 è eseguita l'azione di accesso alla conversazione con il contatto da cui si vogliono ricevere i messaggi via removable drive.

Nella riga 4 della Figura 4.50 si apre il menù della conversazione.

Nelle righe 6 e 8 della Figura 4.50 si accede alla sezione di invio o ricezione dei messaggi mediante removable drive (Figura 2.18 e 2.19 della Sezione 2.3.1).

Nella riga 10 della Figura 4.50 si esegue il Tap sul tasto di invio dati (Figura 2.20 della Sezione 2.3.1) in modo da procedere con la ricezione dei messaggi mediante removable drive.

Dalla riga 12 alla riga 18 della Figura 4.50 si eseguono le azioni necessarie per selezionare il file contenente i messaggi da importare.

Nella riga 20 della Figura 4.50 si dà conferma di conclusa procedura di ricezione messaggi mediante removable drive.

4.8.2 - Artefatti generati

Quando si esegue un invio o una ricezione via removable drive, gli artefatti generati nel database sono gli stessi che si generano in caso di invio o ricezione mediante rete Tor o Wi-Fi. Quindi gli artefatti generati dall'invio di un messaggio sono gli stessi visti nella Sezione 4.6, gli artefatti generati dall'invio di un invito o dalla ricezione dell'invito a un gruppo privato sono gli stessi che si

vedranno nella Sezione 4.9, gli artefatti generati dall'invio di un invito o dalla ricezione di un invito a un forum sono gli stessi che si tratteranno nella Sezione 4.10 e infine, gli artefatti generati dalla pubblicazione di un post in un blog sono gli stessi analizzeranno nella Sezione 4.11.

A livello di database, l'unica differenza tra l'invio tramite rete Tor o Wi-Fi e l'invio tramite removable drive è che, nel secondo caso, nella tabella OUTGOINGKEYS, la tupla con il campo TRANSPORTID uguale a org.briarproject.bramble.drive e con il campo CONTACTID corrispondente all'ID del contatto a cui sono stati inviati i messaggi, presenta il campo STREAM diverso da 0. Tale campo contiene la quantità di stream che l'utente locale ha inviato via removable drive. Inoltre, la tupla considerata ha i campi ROOTKEY e ALICE impostati a NULL (Figura 4.51).

	TRANSPORTID	CONTACTID	STREAM	ACTIVE	ROOTKEY	ALICE
1	org.briarproject.bramble.drive	1	1	1	NULL	NULL

Figura 4.51: Dati in OUTGOINGKEYS con informazione del numero di stream inviati via removable drive

Per quanto riguarda invece la ricezione di uno stream via removable drive, non è possibile ottenere tale informazione dai dati presenti nel database.

4.8.3 - Query

Per sapere se l'utente locale ha inviato messaggi via removable drive è necessario eseguire la seguente query

```
SELECT
TRANSPORTID, CONTACTID, STREAM, ACTIVE, ROOTKEY, ALICE
FROM
OUTGOINGKEYS
WHERE
TRANSPORTID = "org.briarproject.bramble.drive" AND CONTACTID = 1 AND ROOTKEY IS
NULL
```

4.8.4 - Correlazione dati database con altri file

Purtroppo, utilizzando solo i dati presenti nel database non vi è modo di collegare lo stream ai messaggi che erano contenuti in esso. L'unico modo per ottenere ciò è, una volta visto che nel database il valore del campo STREAM relativo al TRANSPORTID org.briarproject.bramble.drive della tabella OUTGOINGKEYS è maggiore di 0, andare a individuare e recuperare il/i file generati da Briar quando si utilizza la funzionalità di invio via removable drive. Questi file, se non cancellati dall'utente, possono essere ovunque nel file system del dispositivo.

Una volta recuperati tali file bisognerà semplicemente applicare la procedura di decodifica del file presentata nella Sezione 3.4.1. Una volta ottenuti i messaggi contenuti nel file, basta controllare se il payload del messaggio corrisponde al contenuto del campo RAW nella tabella MESSAGES. Se tale corrispondenza esiste è possibile ottenere tutte le informazioni relative al messaggio esattamente come se il messaggio fosse stato inviato via Tor o via Wi-Fi.

Purtroppo, non vi è modo di sapere con certezza se l'utente locale ha ricevuto dei messaggi via removable drive, è solo possibile cercare nel file system del dispositivo file che potrebbero essere stati generati dalla modalità di scambio messaggi via removable drive, provare a decodificarli e, una volta ottenuti i messaggi contenuti, controllare ogni messaggio per vedere se il payload del

messaggio corrisponde a un campo RAW dei messaggi salvati nella tabella MESSAGES. Se c'è questa corrispondenza è possibile recuperare i metadati associati a tale messaggio e vedere se il messaggio ha metadato local uguale a true (messaggio inviato dall'utente locale) o ha metadato local uguale a false (messaggio ricevuto dall'utente locale).

4.9 - Cancellazione messaggi

4.9.1 - Esperimenti

Numero esperimento	Descrizione	Sotto-esperimenti
18	Cancellazione messaggi di testo	1. B cancella tutti i messaggi 2. A cancella tutti i messaggi
19	Cancellazione messaggi con immagini	1. B cancella tutti i messaggi 2. A cancella tutti i messaggi
20	Cancellazione messaggi di testo con emoji	1. B cancella tutti i messaggi 2. A cancella tutti i messaggi

Tabella 4.9: Lista esperimenti cancellazione messaggi in chat privata

La cancellazione di messaggi in chat private da parte dell'utente locale è stata simulata nel sotto-esperimento 2 degli esperimenti 18, 19 e 20.

Sia per l'esperimento 18, sia per il 19 sia per il 20, la cancellazione dei messaggi da parte dell'utente A (utente oggetto d'analisi) è stata eseguita attraverso lo stesso file delle azioni (Figura 4.52).

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,50
7 # Cancellazione di tutti i messaggi della conversazione
8 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
9 SLEEP,2
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.appcompat.widget.LinearLayoutCompat[1]/android.widget.ImageView[@content-desc='More options']
11 #SLEEP
12 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget.LinearLayout[3]/android.widget.LinearLayout[1]/android.widget.RelativeLayout[1]/android.widget.TextView[@resource-id='org.briarproject.briar.android:id/title']
13 #SLEEP
14 Tap,RES_ID:android:id/button2
15 SLEEP,2
16 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
17 SLEEP,1
18 # Apertura menù laterale
19 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Open the navigation drawer']
20 # Sign-out in Briar
21 SLEEP,2
22 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
23 SLEEP,5

```

Figura 4.52: File delle azioni per cancellazione di tutti i messaggi di una chat privata

Nella riga 8 della Figura 4.52 è eseguita l'azione di accesso alla conversazione da cui si vogliono cancellare i messaggi.

Nella riga 10 della Figura 4.52 si accede alla funzionalità di cancellazione di tutti i messaggi della conversazione in cui si è.

Nella riga 14 della Figura 4.52 si dà conferma della cancellazione di tutti i messaggi della conversazione.

4.9.2 - Artefatti generati

Sia che si tratti di messaggi di testo sia che si tratti di messaggi contenenti un'immagine, se l'utente locale cancella tutti i messaggi della conversazione con un proprio contatto avvengono le seguenti modifiche ai dati del database:

- Nella tabella GROUPMETADATA il metadato messageCount relativo al gruppo del CLIENTID org.briarproject.briar.messaging associato al contatto è settato a 0
- Nella tabella MESSAGES è cancellato il contenuto del campo RAW di ogni messaggio che faceva parte della conversazione con il contatto di cui si stanno cancellando tutti i messaggi quindi tutti i messaggi inviati o ricevuti dal gruppo relativo a quel contatto e associato al CLIENTID org.briarproject.briar.messaging.

Come si può osservare in Figura 4.53 in cui si riporta un esempio di messaggi di una conversazione cancellati, il valore del campo LENGTH che contiene la lunghezza del messaggio rimane invariato.

	MESSAGEID	GROUPID	TIMESTAMP	LENGTH	RAW
1	d35c890171633ac373034ca67c097c98d9d2d0...	eb54fff1192fd13fc2cd0581aa292106b2dfdd55...	1717754129935	31616	NULL
2	0a3ec1399fa25251096659992c850c41c5093db...	eb54fff1192fd13fc2cd0581aa292106b2dfdd55...	1717754134117	118	NULL
3	2e9920c8dd98d10d538cad91fe2786ab34b383...	eb54fff1192fd13fc2cd0581aa292106b2dfdd55...	1717754255181	31649	NULL
4	4c717021794058c3ab30f9af66f4774c6af48fa1...	eb54fff1192fd13fc2cd0581aa292106b2dfdd55...	1717754269814	118	NULL

Figura 4.53: Dati relativi ai messaggi cancellati in tabella MESSAGES

- Nella tabella MESSAGEMETADATA sono cancellati tutti i metadati relativi ai messaggi della conversazione
- Nella tabella STATUSES il campo DELETED relativo ai messaggi della conversazione di cui si stanno cancellando i messaggi è settato a True (Figura 4.54)

Perciò per capire se sono stati cancellati dei messaggi basterà controllare se nella tabella STATUSES sono presenti messaggi con campo DELETED a True (1). Non sarà però possibile recuperare il contenuto di tali messaggi in quanto il contenuto del campo RAW dei messaggi cancellati è eliminato.

	MESSAGEID	GROUPID	TIMESTAMP	LENGTH	LENGTH	RAW	CONTACTID	DELETED	ACK	SEEN	REQUESTED	EXPIRY	TXCOUNT	MAXLATENCY
1	cd3cca02cdb35bb8b08ab...	5ca7eb4f0620a2f166...	1717754023626	579	579	NULL	1	1	0	1	0	1717754087803	1	30000
2	6f286e9e1a620d72ab8dd...	5ca7eb4f0620a2f166...	1717754024722	579	579	NULL	1	1	0	1	0	0	0	NULL
3	6df26d76806276d0648d0...	989b214b03c280656...	1717754023666	95	95	NULL	1	1	0	0	0	0	0	NULL
4	27de25b80f41db7ad0814...	989b214b03c280656...	1717754023664	126	126	NULL	1	1	0	0	0	0	0	NULL
5	942daa6ed55efbfc41f424...	989b214b03c280656...	1717754023668	155	155	NULL	1	1	0	0	0	0	0	NULL
6	432f822a6da2efbaa8d649...	989b214b03c280656...	1717754023667	142	142	NULL	1	1	0	0	0	0	0	NULL
7	d35c890171633ac373034...	eb54fff1192fd13fc2c...	1717754129935	31616	31616	NULL	1	1	0	1	0	1717754195371	1	30000
8	0a3ec1399fa25251096659...	eb54fff1192fd13fc2c...	1717754134117	118	118	NULL	1	1	0	1	0	1717754195372	1	30000
9	2e9920c8dd98d10d538ca...	eb54fff1192fd13fc2c...	1717754255181	31649	31649	NULL	1	1	0	1	0	0	0	NULL
10	4c717021794058c3ab30f9...	eb54fff1192fd13fc2c...	1717754269814	118	118	NULL	1	1	0	1	0	0	0	NULL

Figura 4.54: Dati relativi ai messaggi cancellati in tabella STATUSES

Se invece è l'utente remoto a cancellare tutti i messaggi della conversazione con l'utente locale, nel database dell'utente locale non avviene nessuna modifica. Questo significa che nel database dell'utente locale rimangono salvati tutti i messaggi scambiati con l'utente remoto.

4.9.3 - Query

Per recuperare i messaggi cancellati dall'utente locale è necessario eseguire la seguente query

Nella riga 14 della Figura 4.55 si entra nella procedura di aggiunta gruppo privato.

Nelle righe 16 e 19 della Figura 4.55 si definisce il nome del gruppo privato e lo si crea (Figura 2.25 della Sezione 2.4.1).

4.10.2 - Artefatti generati

Quando un utente crea un gruppo privato, questo è salvato nella tabella GROUPS in cui è presente l'id del gruppo privato, il nome del BSP client relativo al gruppo privato (org.briarproject.briar.privategroup) e un campo DESCRIPTOR al cui interno è presente un BDF che contiene il nome del creatore del gruppo privato, la public key del creatore del gruppo privato e il nome del gruppo privato.

Come si può vedere in Figura 4.56, ogni gruppo privato creato ha i seguenti metadati associati presenti nella tabella GROUPMETADATA:

- creatorId: contiene l'id del creatore del gruppo.
Corrisponde al valore del campo AUTHORID nella tabella LOCALAUTHORS se il creatore del gruppo è l'utente locale.
Corrisponde al valore del campo AUTHORID nella tabella CONTACTS se il creatore del gruppo è un contatto dell'utente locale.
- dissolved: definisce se il gruppo è stato cancellato dal creatore.
Se il valore di questo metadato è False (10) vuol dire che il gruppo privato non è stato cancellato, mentre se il valore è uguale a True (11) vuol dire che il gruppo privato è stato cancellato
- latestMessageTime: contiene il timestamp dell'ultimo messaggio inviato nel gruppo
- members: contiene la lista di tutti i membri del gruppo privato
- messageCount: contiene il numero di messaggi scambiati nel gruppo privato
- ourGroup: definisce se il gruppo privato è stato creato dall'utente locale o meno.
Se il valore del metadato ourGroup è uguale a False (10) allora il gruppo è stato creato da un altro utente, mentre se il valore è uguale a True (11) allora il gruppo è stato creato dall'utente locale.
- previousMsgId: contiene l'id del messaggio precedente inviato nel gruppo privato
- storedMessageId: contiene l'id dell'ultimo messaggio salvato nel gruppo
- unreadCount: contiene il numero di messaggi locali non letti nel gruppo privato dall'utente locale

Il valore di ogni metadato appena elencato è memorizzato in un BDF.

CLIENTID	GROUPID	DESCRIPTOR	GroupMeta
1 org.briarproject.briar.privategroup	f297d2624c2d3aa8e6add0b225cf214468ba07...	60602101410550726f766151207c29e1c17b8d...	creatorId=5120525c42bb2920a2fc9c85072ee8... dissolved=10 latestMessageTime=280000018fe262c071 members=607041066d656d626572602101410... messageCount=2101 ourGroup=11 previousMsgId=5120e58b13ba721540e14932... storedMessageId=5120e58b13ba721540e149... unreadCount=2100

Figura 4.56: Dati relativi a un gruppo privato e metadati ad esso associati

Inoltre, appena l'utente locale crea il gruppo privato, in automatico ne diventa membro e quindi nel proprio database sarà presente anche un messaggio di tipo JOIN con diversi metadati associati (Figura 4.57).

Tali metadati sono:

- initialJoinMsg: definisce se il messaggio associato a questo metadato è il primo messaggio di join di quel gruppo privato. Vale True (11) se l'utente locale è il creatore del gruppo privato, altrimenti vale False (10)
- member: contiene le informazioni del membro del gruppo
- read: definisce se il messaggio a cui fa riferimento è stato letto o meno. Vale True (11) se il messaggio è stato letto, altrimenti vale False (10)
- timestamp: contiene il timestamp del messaggio a cui fa riferimento
- type: contiene il tipo di messaggio.
Nel caso del gruppo privato, i tipi possibili sono quelli illustrati nella Sezione 3.1.3.8

Anche in questo caso il valore di ogni metadato è memorizzato in un BDF.

	GROUPID	MESSAGEID	MessageMeta
1	f297d2624c2d3aa8e6add0b225cf214468ba07...	e58b13ba721540e1493266dc9429d8c0b0a666...	initialJoinMsg=11 member=602101410550726f766151207c29e1... read=11 timestamp=280000018fe262c071 type=2100

4.57: Dati relativi a un messaggio di join del creatore del gruppo privato e metadati ad esso associati

4.10.3 - Query

Per ottenere i dati presenti in Figura 4.56, quindi per recuperare i metadati di tutti i gruppi privati creati dall'utente locale bisogna eseguire la seguente query.

```
SELECT g.CLIENTID, g.GROUPID, g.DESRIPTOR, group_meta.GroupMeta
FROM GROUPS g
JOIN (
    SELECT gm.GROUPID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS GroupMeta
    FROM GROUPS JOIN GROUPMETADATA gm USING(GROUPID)
    GROUP BY gm.GROUPID
) AS group_meta ON g.GROUPID = group_meta.GROUPID
WHERE GroupMeta LIKE "%ourGroup=11%"
```

Invece per recuperare i metadati di tutti i messaggi di join iniziale (Figura 4.57) è necessario eseguire la seguente query.

```
SELECT m.GROUPID, m.MESSAGEID, message_meta.MessageMeta
FROM MESSAGES m
JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE MessageMeta LIKE "%initialJoinMsg%"
```

4.11 - Invito al gruppo privato

4.11.1 - Esperimenti

La funzionalità di invito a un gruppo privato è stata analizzata sia quando l'utente locale invita un suo contatto sia quando un suo contatto invita l'utente locale. Per questo motivo sono stati necessari due esperimenti relativi a questa funzionalità (esperimento 22 e esperimento 23).

Numero esperimento	Descrizione	Sotto-esperimenti
22	Invito a partecipare a un gruppo privato da parte dell'utente locale verso l'utente remoto	<ol style="list-style-type: none">1. A invita B e B rifiuta l'invito2. A invita B e B accetta l'invito
23	Invito a partecipare a un gruppo privato da parte dell'utente remoto verso l'utente locale	<ol style="list-style-type: none">1. B invita A e A rifiuta2. B invita A e A accetta

Tabella 4.11: Lista esperimenti invito a gruppo privato

Relativamente alla funzione di invito, i sotto-esperimenti 1 e 2 dell'esperimento 22 in Tabella 4.11 eseguono le azioni che l'utente locale deve effettuare per invitare un suo contatto a un gruppo privato da lui creato, mentre i sotto-esperimenti 1 e 2 dell'esperimento 23 in Tabella 4.11 permettono di eseguire le azioni necessarie perché un utente remoto inviti l'utente locale in un gruppo privato creato dall'utente remoto.

Per simulare l'invio dell'invito a partecipare al gruppo privato da parte dell'utente locale nonché creatore è stato utilizzato un file delle azioni di cui parte è riportata nella Figura 4.58.


```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,100
7 # Rifiuto dell'invito a entrare a far parte di un gruppo privato
8 Tap,RES_ID:org.briarproject.briar.android:id/trustIndicatorDescription
9 #SLEEP
10 Tap,RES_ID:org.briarproject.briar.android:id/declineButton
11 SLEEP,10
12 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android
  .widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
13 SLEEP,2
14 # Apertura menù laterale
15 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
  on[@content-desc='Open the navigation drawer']
16 # Sign-out in Briar
17 SLEEP,2
18 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
19 SLEEP,5

```

Figura 4.59: File delle azioni per rispondere al messaggio di invito a partecipare a un gruppo privato

L'istruzione alla riga 10 della Figura 4.59 declina l'invito ricevuto. Per simulare l'azione di accettazione dell'invito basta sostituire l'istruzione alla riga 8 con *Tap,RES_ID:org.briarproject.briar.android:id/acceptButton*.

4.11.2 - Artefatti generati

Quando l'utente locale invia un invito a partecipare a un gruppo privato a un suo contatto o riceve un invito da un suo contatto sono creati due messaggi:

- Un messaggio di gestione del gruppo relativo al CLIENTID *org.briarproject.briar.privategroup.invitation*
- Un messaggio di invito a partecipare al gruppo privato.
Nel BDF che compone il campo RAW della tabella MESSAGES, questo messaggio contiene diverse informazioni.
Nel caso di utente locale creatore, il dato forensicamente più interessante recuperabile dal BDF è il testo opzionale che accompagna il messaggio di invito.
Nel caso, invece di utente locale invitato, i dati interessanti sono il nome del contatto che ha inviato l'invito e il testo opzionale che accompagna il messaggio di invito.

I metadati di questi due messaggi permettono di distinguerli e individuarli. Infatti il primo messaggio (messaggio di gestione del gruppo relativo al CLIENTID *org.briarproject.briar.privategroup.invitation*) ha i seguenti metadati:

- *inviteTimestamp*: contiene il timestamp del messaggio di invito a far parte del gruppo privato
- *isSession*: definisce se è una sessione. Vale True (11) se è una sessione altrimenti vale False (10)
- *lastLocalMessageId*: contiene l'id dell'ultimo messaggio locale inviato dal gruppo identificato dal GROUPID associato
- *lastRemoteMessageId*: contiene l'id dell'ultimo messaggio remoto ricevuto dal gruppo identificato dal GROUPID associato
- *localTimestamp*: contiene il timestamp locale del messaggio
- *privateGroupId*: contiene l'id del gruppo privato cioè il GROUPID associato al CLIENTID *org.briarproject.briar.privategroup*

- **role:** definisce il ruolo dell'utente locale nel gruppo privato. Il ruolo può essere CREATOR (0) o INVITEE (1).
Se l'utente locale è il creatore del gruppo privato e quindi anche colui che ha inviato l'invito questo metadato avrà valore CREATOR (0).
Se l'utente locale invece è stato invitato a partecipare a un gruppo privato questo metadato avrà valore INVITEE (1).
- **sessionId:** contiene l'id della sessione. Corrisponde all'id contenuto in privateGroupId
- **state:** definisce lo stato dell'utente nel gruppo privato.
Come illustrato nella Figura 3.8 della Sezione 3.1.3.9, se l'utente è il creatore del gruppo privato, questo metadato può assumere i valori START (0), INVITED (1), JOINED (2), LEFT (3), DISSOLVED (4) e ERROR (5).
Se, diversamente, l'utente è un invitato, questo metadato può assumere i valori START (0), INVITED (1), ACCEPTED (2), JOINED (3), LEFT (4), DISSOLVED (5) e ERROR (6) (Figura 3.9 della Sezione 3.1.3.9)

Nelle Figure 4.60 e 4.61 sono mostrati, rispettivamente, un esempio di messaggio di gestione presente nel database del creatore del gruppo privato e un esempio di messaggio di gestione presente nel database dell'invitato al gruppo privato.

GROUPID	MESSAGEID	MessageMeta
1 b31ea16ed55d783a0d5062fe686f13cc49eb8a...	62c681ff274cd2e3ac19e560e53dd3d08a11126...	inviteTimestamp=280000018fe2655d66 isSession=11 lastLocalMessageId=512036b5a6ba80d042f4f... lastRemoteMessageId=5120f9f8bf0c39b5faab... localTimestamp=280000018fe265bf95 privateGroupId=5120f297d2624c2d3aa8e6ad... role=2100 sessionId=5120f297d2624c2d3aa8e6add0b22... state=2102

Figura 4.60: Dati relativi a un messaggio di gestione del gruppo relativo al CLIENTID *org.briarproject.briar.privategroup.invitation* (creatore) e metadati ad esso associati

GROUPID	MESSAGEID	MessageMeta
1 2eba079490d274c4793006845a9aaa142c7de...	b90f69d9b21c9391d272d8b1cd9aa64ef0a827...	inviteTimestamp=280000018fe37e2424 isSession=11 lastLocalMessageId=51204d7a4cc7354207855... lastRemoteMessageId=5120a5ab2efd7db40aa... localTimestamp=280000018fe37f1040 privateGroupId=51203d7791149c390f233847... role=2101 sessionId=51203d7791149c390f23384772891f... state=2103

Figura 4.61: Dati relativi a un messaggio di gestione del gruppo relativo al CLIENTID *org.briarproject.briar.privategroup.invitation* (invitato) e metadati ad esso associati

Il messaggio di invito invece ha i seguenti metadati:

- **availableToAnswer:** definisce se l'invito deve ancora ricevere risposta. Vale True (11) se l'invitato deve ancora rispondere all'invito, altrimenti vale False (10)
- **invitationAccepted:** definisce se l'invito è stato accettato dall'utente locale. Vale True (11) se l'utente locale che ha ricevuto l'invito a partecipare al gruppo privato ha accettato l'invito, altrimenti vale False (10)
- **local:** definisce se il messaggio è locale. Vale True (11) se il messaggio è stato inviato dall'utente locale, vale False (10) se il messaggio è stato ricevuto dall'utente locale.
- **messageType:** contiene il tipo di messaggio.

Come spiegato nella Sezione 3.1.3.9 i messaggi per il CLIENTID `org.briarproject.briar.privategroup.invitation` possono essere di tipo INVITE (0), JOIN (1), LEAVE (2) e ABORT (3)

- `privateGroupId`: contiene l'id del gruppo privato, cioè il GROUPID associato al CLIENTID `org.briarproject.briar.privategroup`
- `read`: definisce se il messaggio è stato letto dall'utente locale. Vale True (11) se l'utente locale ha letto il messaggio, altrimenti vale False (10). Se il messaggio è locale (chiave `local` uguale a 11) allora il messaggio è stato sicuramente letto dall'utente locale in quanto è stato inviato da lui stesso
- `timestamp`: contiene il timestamp del messaggio
- `visibleInUi`: definisce se il messaggio è visibile nell'interfaccia grafica dell'applicazione. Vale True (11) se il messaggio è visibile nell'interfaccia grafica dell'applicazione altrimenti vale False (10)

La Figura 4.62 presenta un esempio di messaggio di invito a un gruppo privato inviato dall'utente locale a un suo contatto.

In Figura 4.63, invece è riportato un esempio di messaggio di invito a un gruppo privato inviato da un contatto dell'utente locale all'utente locale.

	GROUPID	MESSAGEID	MessageMeta
1	b31ea16ed55d783a0d5062fe686f13cc49eb8a...	88a5bac36da8f1aa157fd8ff8ce4b4c41853c8c5...	availableToAnswer=10 invitationAccepted=10 local=11 messageType=2100 privateGroupId=5120f297d2624c2d3aa8e6ad... read=11 timestamp=280000018fe263da5c visibleInUi=11

Figura 4.62: Dati relativi a un messaggio di invito invitato dall'utente locale a un suo contatto e metadati ad esso associati

	GROUPID	MESSAGEID	MessageMeta
1	2eba079490d274c4793006845a9aaaa142c7de...	60612f0ebf577636fd29a32153915ed21efce60...	availableToAnswer=10 invitationAccepted=11 local=10 messageType=2100 privateGroupId=51203d7791149c390f233847... read=11 timestamp=280000018fe37e2424 visibleInUi=11

Figura 4.63: Dati relativi a un messaggio di invito inviato da un contatto dell'utente locale all'utente locale e metadati ad esso associati

Il valore di ogni metadato di entrambi i messaggi (Figura 4.62 e Figura 4.63) è memorizzato in un BDF.

L'utente che riceve un invito a partecipare a un gruppo privato da parte del creatore del gruppo può declinare o accettare l'invito.

La risposta dell'invitato è salvata nel database dell'utente creatore come messaggio con metadati da cui si ottiene che il messaggio è remoto (chiave `local` uguale a 10) e che il tipo del messaggio (chiave `messageType`) è LEAVE (2) o JOIN (1).

Se l'invitato ha declinato l'invito il tipo di messaggio (chiave `messageType`) ha valore LEAVE (2).

In Figura 4.64 è riportato un messaggio di LEAVE inviato dall'invitato (utente remoto) all'invitante nonché creatore del gruppo privato (utente locale).

In Figura 4.65 è invece presentato un esempio di messaggio di LEAVE inviato dall'invitato all'invitante ma in cui l'invitato è l'utente locale mentre l'invitante (creatore) è un utente remoto.

	GROUPID	MESSAGEID	MessageMeta
1	b31ea16ed55d783a0d5062fe686f13cc49eb8a...	75aa01e8512717fe30d38a22fbb59eaf0cea7d0...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2102 privateGroupId=5120f297d2624c2d3aa8e6ad... read=10 timestamp=280000018fe263e369 visibleInUi=11

Figura 4.64: Dati relativi a un messaggio di rifiuto dell'invito (LEAVE) da parte dell'utente remoto e metadati ad esso associati

	GROUPID	MESSAGEID	MessageMeta
1	2eba079490d274c4793006845a9aaaa142c7de...	581d46b57e309c87d979a9d37183f686df0471...	availableToAnswer=10 invitationAccepted=10 local=11 messageType=2102 privateGroupId=51203d7791149c390f233847... read=11 timestamp=280000018fe37d0e2e visibleInUi=11

Figura 4.65: Dati relativi a un messaggio di rifiuto dell'invito (LEAVE) da parte dell'utente locale e metadati ad esso associati

Come è possibile notare, la differenza tra i metadati in Figura 4.64 e i metadati in Figura 4.65 è principalmente il metadato local che in Figura 4.64 vale False (10) mentre in Figura 4.65 vale True (11).

Questo metadato è molto utile in quanto se nel database dell'utente locale è presente un messaggio di risposta a un invito con metadato local uguale a False (10), si può dedurre che l'utente locale è il creatore del gruppo privato e ha inviato l'invito, mentre se il metadato local è uguale a True (11), si può ottenere l'informazione che l'utente locale è stato invitato a partecipare a un gruppo privato.

Se l'invitato ha accettato l'invito il tipo di messaggio (chiave messageType) ha valore JOIN (1).

Nelle Figure 4.66 e 4.67 sono riportati corrispondentemente un esempio di messaggio di tipo JOIN inviato dall'utente remoto (invitato) all'utente locale (creatore-invitante) e un esempio di messaggio di tipo JOIN inviato dall'utente locale (invitato) all'utente remoto (creatore-invitante).

	GROUPID	MESSAGEID	MessageMeta
1	b31ea16ed55d783a0d5062fe686f13cc49eb8a...	f9f8bf0c39b5faabbc2b2d69768e00e0b6252a1...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2101 privateGroupId=5120f297d2624c2d3aa8e6ad... read=10 timestamp=280000018fe265ab54 visibleInUi=11

Figura 4.66: Dati relativi a un messaggio di accettazione dell'invito (JOIN) da parte dell'utente remoto e metadati ad esso associati

GROUPID	MESSAGEID	MessageMeta
1 2eba079490d274c4793006845a9aaaa142c7de...	4d7a4cc7354207855be8f8fa5a4493e4637d735...	availableToAnswer=10 invitationAccepted=10 local=11 messageType=2101 privateGroupId=51203d7791149c390f233847...

Figura 4.67: Dati relativi a un messaggio di accettazione dell'invito (JOIN) da parte dell'utente locale e metadati ad esso associati

Siccome nel database del creatore sono salvati i messaggi di join del creatore e di tutti i partecipanti al gruppo privato, l'accettazione dell'invito a partecipare al gruppo privato da parte dell'invitato (sia esso l'utente locale o un utente remoto) implica che il messaggio di join dell'invitato che ha accettato l'invito sia salvato nel database dell'utente creatore.

Infatti, nel caso in cui l'invitato abbia accettato l'invito, lato database del creatore-invitante, sarà presente un messaggio relativo al gruppo associato al CLIENTID org.briarproject.briar.privategroup e identificabile dai metadati:

- initialJoinMsg uguale a False (10)
- type uguale a JOIN (0) (tipi di messaggi in Sezione 3.1.3.8)
- member che contiene diverse informazioni sull'utente che ha accettato l'invito

Nelle Figura 4.68 e 4.69 si riportano un esempio di messaggi di join presenti nel database del creatore quando il creatore è rispettivamente l'utente locale o l'utente remoto.

GROUPID	MESSAGEID	MessageMeta
1 f297d2624c2d3aa8e6add0b225cf214468ba07...	0b2aef5867bb2507fca2be5e19950e8df7d8f19...	initialJoinMsg=10 member=6021014106416d69636f315120eeff2... read=10 timestamp=280000018fe265ab58 type=2100
2 f297d2624c2d3aa8e6add0b225cf214468ba07...	e58b13ba721540e1493266dc9429d8c0b0a666...	initialJoinMsg=11 member=602101410550726f766151207c29e1... read=11 timestamp=280000018fe262c071 type=2100

Figura 4.68: Dati relativi ai messaggi di join di tutti i partecipanti (2 partecipanti) del gruppo privato (utente locale creatore) e metadati ad essi associati

GROUPID	MESSAGEID	MessageMeta
1 3d7791149c390f23384772891fefe34adb93f86...	1bc7cb1f996a13091a7d2aab641bfe34f07ef0a...	initialJoinMsg=11 member=6021014106416d69636f315120eeff2... read=10 timestamp=280000018fe379db3e type=2100
2 3d7791149c390f23384772891fefe34adb93f86...	b017747d542a84362e90d899d49acc2af73544...	initialJoinMsg=10 member=602101410550726f76615120dda2cc... read=11 timestamp=280000018fe37f1068 type=2100

Figura 4.69: Dati relativi ai messaggi di join di tutti i partecipanti (2 partecipanti) del gruppo privato (utente locale invitato) e metadati ad essi associati

Se l'invitato è l'utente locale, all'accettazione dell'invito:

- Nella tabella GROUPS è aggiunto il gruppo di cui l'utente locale ha accettato l'invito
- Nella tabella GROUPMETADATA sono aggiunti tutti i metadati del gruppo (Sezione 4.10.2)

Inoltre, sempre in caso di accettazione dell'invito, il gruppo privato diventerà condiviso e quindi il campo SHARED nella tabella GROUPVISIBILITIES sarà valorizzato a 1.

4.11.3 - Query

Per ottenere tutti i messaggi relativi ai CLIENTID org.briarproject.briar.privategroup e org.briarproject.briar.privategroup.invitation ordinati per TIMESTAMP è possibile eseguire la seguente query

```
SELECT g.CLIENTID, g.GROUPID, gm.METAKEY, gm.VALUE, m.MESSAGEID, TIMESTAMP,
       message_meta.MessageMeta
FROM GROUPS g JOIN GROUPMETADATA gm USING(GROUPID)
  JOIN MESSAGES m USING (GROUPID)
  JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS
           MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
  ) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE
(g.CLIENTID = "org.briarproject.briar.privategroup.invitation" AND gm.METAKEY =
"contactId")
OR (g.CLIENTID = "org.briarproject.briar.privategroup" AND gm.METAKEY = "ourGroup")
ORDER BY TIMESTAMP
```

4.12 - Messaggi in gruppo privato

4.12.1 - Esperimenti

Per analizzare questa funzionalità si è deciso di eseguire due esperimenti. Uno in cui l'utente locale scrive un post nel gruppo privato (esperimento 24) e uno in cui è l'utente remoto a scrivere il post nel gruppo privato (esperimento 25).

Inoltre, si sono voluti analizzare i possibili artefatti generati in situazioni di connettività diverse.

Numero esperimento	Descrizione	Sotto-esperimenti
24	Post dell'utente locale in gruppo privato, con mittente e destinatario offline a turno o entrambi gli utenti online	<ol style="list-style-type: none">1. A crea un post (A offline e B online)2. A crea un post (A online e B offline)3. A crea un post (A e B online)4. B replica al post di A
25	Post dell'utente remoto in gruppo privato, con mittente e destinatario offline a turno o entrambi gli utenti online	<ol style="list-style-type: none">1. B crea un post (B offline e A online)2. B crea un post (B online e A offline)3. B crea un post (A e B online)

Tabella 4.12: Lista esperimenti post in gruppo privato

Per simulare la pubblicazione di un post nel gruppo privato da parte dell'utente locale (sotto-esperimenti 1, 2 e 3 dell'esperimento 24 in Tabella 4.12) è stato utilizzato il seguente file delle azioni.

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,50
7 # Scrittura di un post nel gruppo privato
8 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
on[@content-desc='Open the navigation drawer']
9 #SLEEP
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
yout[1]/android.widget.ScrollView[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/androidx.recyclerview.
widget.RecyclerView[1]/androidx.appcompat.widget.LinearLayoutCompat[2]/android.widget.CheckedTextView[@resource-id='
org.briarproject.briar.android:id/design_menu_item_text']
11 #SLEEP
12 Tap,RES_ID:org.briarproject.briar.android:id/creatorView
13 SLEEP,5
14 SendKeys,XY:235-2155,TEXT:Post gruppo privato con mittente e destinatario online
15 HideKeyboard
16 SLEEP,2
17 Tap,XY:1015-2155
18 SLEEP,2
19 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android
.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
20 SLEEP,2
21 # Apertura menù laterale
22 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widg-
et.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
on[@content-desc='Open the navigation drawer']
23 # Sign-out in Briar
24 SLEEP,2
25 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
26 SLEEP,5

```

Figura 4.70: File delle azioni per pubblicare un post nel gruppo privato

Nella riga 12 della Figura 4.70 è selezionato il gruppo privato in cui si vuole pubblicare il post.

Nelle righe 14 e 17 della Figura 4.70 si eseguono le istruzioni per scrivere il testo del post e pubblicarlo nel gruppo privato (Figura 2.30 della Sezione 2.4.3).

4.12.2 - Artefatti generati

Ogni post è un messaggio il cui campo RAW nella tabella MESSAGES è composto dall'id del BSP client org.briarproject.briar.privategroup che ha inviato o ricevuto il messaggio seguito dal timestamp del messaggio a sua volta seguito da un BDF da cui è possibile ottenere il nome del mittente del post e il testo del post.

Il messaggio di post ha i seguenti metadati:

- member: contiene il nome dell'autore del post
- read: definisce se il messaggio è stato letto o meno
- timestamp: contiene il timestamp del messaggio
- type: definisce il tipo di messaggio.

In caso di messaggio di post questo metadato ha valore uguale a POST (1)

	GROUPID	MESSAGEID	MessageMeta
1	a6ae17a9a41ec87ba325749ee9949536d1e0ec...	56eb5c7b86f732b4c269ede9e1be99b54832f9...	member=602101410550726f76615120e86465... read=11 timestamp=280000018fe284d5dc type=2101

Figura 4.71: Dati relativi a un post pubblicato in gruppo privato e metadati ad esso associati

I messaggi di re-post di un post hanno le stesse caratteristiche di un post ma, a differenza loro, nel BDF contenuto nel campo RAW, oltre al nome dell'autore del re-post e al testo del re-post vi è anche l'informazione dell'id del messaggio del post che si sta ripostando.

Oltre ai metadati del messaggio di post, un messaggio di re-post ha i seguenti metadati:

- parentMsgId: contiene l'id del messaggio del post che si sta ripostando
- previousMsgId: contiene l'id del messaggio precedente inviato nel gruppo privato.

	GROUPID	MESSAGEID	MessageMeta
1	a6ae17a9a41ec87ba325749ee9949536d1e0ec...	97365bc945d0badfe66aa1b61e38d072afb049...	member=6021014106416d69636f315120eff2... parentMsgId=512056eb5c7b86f732b4c269ed... previousMsgId=5120f28ca28469a2e27521ff70... read=11 timestamp=280000018fe285501f type=2101

Figura 4.72: Dati relativi a un re-post di un post pubblicato in un gruppo privato e metadati ad esso associati

4.12.3 - Query

Per recuperare tutti i messaggi scambiati in tutti i gruppi privati a cui l'utente partecipa, è necessario eseguire la seguente query

```
SELECT g.CLIENTID, g.GROUPID, m.MESSAGEID, m.RAW, TIMESTAMP,
       message_meta.MessageMeta
FROM GROUPS g JOIN MESSAGES m USING (GROUPID)
JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE g.CLIENTID = "org.briarproject.briar.privategroup"
ORDER BY TIMESTAMP
```

4.13 - Abbandono del gruppo privato

4.13.1 - Esperimenti

Lo studio della funzionalità di abbandono del gruppo privato ha richiesto che si prendessero in considerazione due situazioni differenti:

- L'abbandono da parte del creatore del gruppo privato (esperimenti 26 e 28)
- L'abbandono da parte di un semplice membro del gruppo privato (esperimenti 27 e 29)

Numero esperimento	Descrizione	Sotto-esperimenti
26	Abbandono del gruppo privato da parte dell'utente locale creatore (A) (database creatore A)	1. A abbandona/cancella il gruppo privato
27	Abbandono del gruppo privato da parte dell'utente remoto membro (B) (database creatore A)	1. B abbandona il gruppo privato 2. A abbandona/cancella il gruppo privato
28	Abbandono del gruppo privato da parte dell'utente remoto creatore (B) (database membro A)	1. B abbandona/cancella il gruppo privato
29	Abbandono del gruppo privato da parte dell'utente locale membro A (database membro A)	1. A abbandona il gruppo privato 2. B abbandona/cancella il gruppo privato

Tabella 4.13: Lista esperimenti di abbandono gruppo privato

Negli esperimenti 26 e 28 si è eseguito rispettivamente l'abbandono del gruppo privato da parte dell'utente locale creatore e dell'utente remoto creatore. Mentre per indagare l'abbandono del gruppo privato da parte di un membro, sia esso l'utente locale o l'utente remoto, si sono presi in considerazione i sotto-esperimenti 1 dell'esperimento 27 e dell'esperimento 29.

Questi sotto-esperimenti hanno permesso di individuare e definire gli artefatti generati o cancellati dall'azione di abbandono del gruppo privato da parte del creatore o di un membro in situazioni diverse e da punti di vista diversi (sia dal punto di vista dell'utente locale creatore sia dal punto di vista dell'utente locale membro).

Per simulare le azioni necessarie per effettuare l'abbandono di un gruppo privato da parte dell'utente locale (A), sia esso creatore del gruppo privato o membro, è stato utilizzato il file delle azioni riportato di seguito.

I messaggi relativi a questo CLIENTID sono i messaggi di gestione del gruppo privato, di invito e di risposta all'invito.

Per abbandonare il gruppo privato, l'utente locale (sia esso creatore o membro del gruppo privato) invia un messaggio di LEAVE (2) con chiave 'local' uguale a True (11) (Figura 4.74).

	GROUPID	MESSAGEID	MessageMeta
1	b31ea16ed55d783a0d5062fe686f13cc49eb8a...	0ffed5e5921ec9e19c2875a177a3e02df9393be...	availableToAnswer=10 invitationAccepted=10 local=11 messageType=2102 privateGroupId=5120f297d2624c2d3aa8e6ad... read=11 timestamp=280000018fe266f7b9 visibleInUi=10

Figura 4.74: Dati relativi a un messaggio di abbandono di un gruppo privato e metadati ad esso associati

Ciò che però più di ogni altra cosa indica che l'utente locale ha abbandonato il gruppo privato è il valore del metadato 'state' del messaggio di gestione del gruppo privato in combinazione con il valore del metadato 'role' (entrambi metadati del messaggio di gestione del gruppo privato).

Come mostrato in Figura 4.75, se l'utente locale che ha abbandonato il gruppo è il creatore, il messaggio di gestione del gruppo privato avrà metadato 'role' uguale a CREATOR (0) e metadato 'state' uguale a DISSOLVED (4) (Figura 3.8 della Sezione 3.1.3.9).

Se l'utente locale che ha abbandonato il gruppo è un invitato, il messaggio di gestione del gruppo privato avrà metadato 'role' uguale a INVITEE (1) e metadato state uguale a LEFT (4) (Figura 3.8 della Sezione 3.1.3.9).

	GROUPID	MESSAGEID	MessageMeta
1	b31ea16ed55d783a0d5062fe686f13cc49eb8a...	62c681ff274cd2e3ac19e560e53dd3d08a11126...	inviteTimestamp=280000018fe2655d66 isSession=11 lastLocalMessageId=51200ffed5e5921ec9e19c... lastRemoteMessageId=5120f9f8bf0c39b5faab... localTimestamp=280000018fe266f7b9 privateGroupId=5120f297d2624c2d3aa8e6ad... role=2100 sessionId=5120f297d2624c2d3aa8e6add0b22... state=2104

Figura 4.75: Dati relativi a un messaggio di gestione del gruppo privato dopo l'abbandono e metadati ad esso associati

Se invece è l'utente remoto invitato a lasciare il gruppo privato, nel database dell'utente locale creatore saranno presenti:

- Un messaggio di tipo LEAVE (2) e 'local' uguale a False (10)
- Il messaggio di gestione del gruppo privato che avrà metadato 'state' uguale a 3. In questo caso il metadato 'role' è CREATOR (0) quindi 3 significa LEFT.
- Una tupla nella tabella MESSAGEDEPENDENCIES (Figura 4.76) che mette in relazione il messaggio di abbandono dell'utente remoto con il messaggio di accettazione dell'invito a partecipare al gruppo privato da parte dell'utente remoto

CLIENTID	MessageID	MessageMeta	DependencyID	DependencyMeta
org.briarproject.briar.privategroup.invitation	3bd02a63d136ae...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2102 privateGroupId=5120e518307411e0fffd41091... read=10 timestamp=280000018fe36f6e66 visibleInUi=10	c7bbd808f6dcae...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2101 privateGroupId=5120e518307411e0fffd41091... read=10 timestamp=280000018fe36d56b9 visibleInUi=11

Figura 4.76: Metadati dei messaggi in relazione in MESSAGEDEPENDENCIES

Se l'utente locale è un invitato al gruppo privato e l'utente remoto creatore abbandona il proprio gruppo privato, nel database dell'utente locale dell'invitato saranno presenti i seguenti artefatti:

- Il metadato 'dissolved' associato al gruppo privato cancellato sarà uguale a True (11)
- Un messaggio di tipo LEAVE (2) e 'local' uguale a False (10)
- Il messaggio di gestione del gruppo privato avrà metadato state uguale a 5. In questo caso il metadato 'role' è INVITEE (1) quindi 5 significa DISSOLVED.
- Una tupla nella tabella MESSAGEDEPENDENCIES che mette in relazione il messaggio di abbandono dell'utente remoto con il messaggio di accettazione dell'invito a partecipare al gruppo privato da parte dell'utente remoto

4.13.3 - Query

Per controllare se ci sono dei gruppi privato di cui l'utente locale faceva parte o come creatore o come invitato si può eseguire la seguente query

```
SELECT m.GROUPID, m.MESSAGEID, message_meta.MessageMeta
FROM MESSAGES m
JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE
    (MessageMeta LIKE "%role=2100%" AND MessageMeta LIKE "%state=2104%")
    OR (MessageMeta LIKE "%role=2101%" AND MessageMeta LIKE
"%state=2104%")
```

4.14 - Creazione del forum

4.14.1 - Esperimenti

Lo studio degli artefatti generati dalla creazione di un forum ha richiesto solamente l'esecuzione del sotto-esperimento 1 dell'esperimento 30.

Numero esperimento	Descrizione	Sotto-esperimenti
30	Creazione del forum	1. A crea un forum

Tabella 4.14: Lista esperimenti creazione forum

Per simulare le azioni di creazione del forum è stato utilizzato il seguente file delle azioni

```
1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,2
7 # Creazione forum
8 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
  on[@content-desc='Open the navigation drawer']
9 SLEEP,2
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.widget.ScrollView[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/androidx.recyclerview.
  widget.RecyclerView[1]/androidx.appcompat.widget.LinearLayoutCompat[3]/android.widget.CheckedTextView[@resource-id='
  org.briarproject.briar.android:id/design_menu_item_text']
11 #SLEEP
12 Tap,RES_ID:org.briarproject.briar.android:id/action_create_forum
13 #SLEEP
14 SendKeys,RES_ID:org.briarproject.briar.android:id/createForumNameEntry,TEXT:Forum di prova
15 HideKeyboard
16 #SLEEP
17 Tap,RES_ID:org.briarproject.briar.android:id/createForumButton
18 SLEEP,2
19 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android
  .widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']
20 SLEEP,1
21 # Apertura menù laterale
22 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
  on[@content-desc='Open the navigation drawer']
23 # Sign-out in Briar
24 SLEEP,2
25 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
26 SLEEP,5
```

Figura 4.77: File delle azioni per il passo 1 dell'esperimento 7.1

Nella riga 10 della Figura 4.77 si effettua l'entrata nella sezione Forums dell'applicazione (Figura 2.32 della Sezione 2.5.1).

Nella riga 12 della Figura 4.77 si entra nella procedura di creazione forum

Nelle righe 14 e 17 della Figura 4.77 si definisce il nome del forum e lo si crea (Figura 2.33 della Sezione 2.5.1).

4.14.2 - Artefatti generati

Quando un utente crea un forum, questo è salvato nella tabella GROUPS in cui è presente l'identificativo del forum, il nome del BSP client relativo al forum (org.briarproject.briar.forum) e un campo DESCRIPTOR al cui interno è presente un BDF che contiene il nome del forum.

Non vi sono ulteriori artefatti generati da questa funzionalità.

4.14.3 - Query

La seguente query può essere usata per recuperare tutti i forum presenti nel database dell'utente locale.

```
SELECT * FROM GROUPS WHERE CLIENTID = "org.briarproject.briar.forum"
```


Questo messaggio è creato solo al primo invito relativo al gruppo `org.briarproject.briar.forum.sharing` associato al contatto a cui si sta inviando l'invito.

	GROUPID	MESSAGEID	MessageMeta
1	bd2ff9a396118851aa78bbf322318702b45794...	6a3b7b47d27c95734a363e2dcb31902bea488...	<pre>inviteTimestamp=2100 isSession=11 lastLocalMessageId=51202f3c1fe7fb2e299e83... lastRemoteMessageId=512084c7b874e79730... localTimestamp=280000018fe7f483e6 sessionId=5120314e490eca66a0e766a5e9116... shareableId=5120314e490eca66a0e766a5e91... state=2103</pre>

Figura 4.80: Dati relativi a un messaggio di gestione del gruppo relativo al CLIENTID `org.briarproject.briar.forum.sharing` e metadati ad esso associati

- un messaggio di invito a partecipare al forum (Figura 4.81 e Figura 4.82).
 Nel BDF che compone il campo RAW di questo messaggio sono contenuti il nome del forum a cui si è invitati, la public key del forum e il testo del messaggio di invito al forum. Se l'invito inviato non è il primo invito a quel forum mandato a quell'utente remoto, nel campo RAW sarà presente anche il riferimento all'id del messaggio d'invito precedente. Il messaggio d'invito ha i seguenti metadati:
 - `availableToAnswer`: definisce se il messaggio deve ancora ricevere risposta. Vale True (11) se l'invitato deve ancora rispondere all'invito, altrimenti vale False (10)
 - `invitationAccepted`: definisce se l'invito è stato accettato dall'utente locale. Vale True (11) se l'utente locale che ha ricevuto l'invito a partecipare al forum ha accettato l'invito, altrimenti vale False (10)
 - `local`: definisce se il messaggio è locale. Vale True (11) se il messaggio è stato inviato dall'utente locale, vale False (10) se il messaggio è stato ricevuto dall'utente locale.
 - `messageType`: contiene il tipo di messaggio. Come spiegato nella Sezione 3.1.3.5 i messaggi per il CLIENTID `org.briarproject.briar.forum.sharing` possono essere di tipo INVITE (0), ACCEPT (1), DECLINE (2), LEAVE (3) e ABORT (4)
 - `read`: definisce se il messaggio è stato letto dall'utente locale. Vale True (11) se l'utente locale ha letto il messaggio, altrimenti vale False (10). Se il messaggio è locale (chiave `local` uguale a 11) allora il messaggio è stato sicuramente letto dall'utente locale in quanto è stato inviato dall'utente locale stesso
 - `shareableId`: contiene l'id del forum
 - `timestamp`: contiene il timestamp del messaggio
 - `visibleInUi`: definisce se il messaggio è visibile nell'interfaccia grafica dell'applicazione. Vale True (11) se il messaggio è visibile nell'interfaccia grafica dell'applicazione altrimenti vale False (10).

GROUPID	MESSAGEID	MessageMeta
1 bd2ff9a396118851aa78bbf322318702b45794...	cb94df025a451ac55380df8fa0c25a07f6246920...	availableToAnswer=10 invitationAccepted=10 local=11 messageType=2100 read=11 shareableId=5120314e490eca66a0e766a5e91... timestamp=280000018fe7f2df36 visibleInUi=11

Figura 4.81: Dati relativi a un messaggio di invito inviato dall'utente locale e metadati ad esso associati

GROUPID	MESSAGEID	MessageMeta
1 8e219b29b40679846680fe141155dde3965830...	a1272bcd4c839d06022185123d757d20e50269...	availableToAnswer=10 invitationAccepted=11 local=10 messageType=2100 read=11 shareableId=51208c6b73eba2eb39fd0157ddc... timestamp=280000018fec655976 visibleInUi=11

Figura 4.82: Dati relativi a un messaggio di invito inviato da un utente remoto e metadati ad esso associati

Il valore di ogni metadato di entrambi i messaggi è memorizzato in un BDF.

Proprio come per l'invito a un gruppo privato, l'utente invitato a un forum, sia esso l'utente locale o meno, può rispondere all'invito accettando o declinando l'invito di condivisione del forum. Se l'invitato accetta l'invito allora il metadato 'messageType' sarà uguale a ACCEPTED (1) (Figura 4.84), se invece l'invito è stato rifiutato, il messaggio di risposta avrà 'messageType' uguale a DECLINE (2) (Figura 4.83).

GROUPID	MESSAGEID	MessageMeta
1 bd2ff9a396118851aa78bbf322318702b45794...	9fe09a8e33dc67a20b26a1fefe3ba4a087e125d...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2102 read=10 shareableId=5120314e490eca66a0e766a5e91... timestamp=280000018fe7f2f556 visibleInUi=11

Figura 4.83: Dati relativi a un messaggio di rifiuto dell'invito da parte dell'utente remoto e metadati ad esso associati

GROUPID	MESSAGEID	MessageMeta
1 bd2ff9a396118851aa78bbf322318702b45794...	84c7b874e797308fc7898254d194c3dab4a962...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2101 read=10 shareableId=5120314e490eca66a0e766a5e91... timestamp=280000018fe7f4c722 visibleInUi=11

Figura 4.84: Dati relativi a un messaggio di accettazione dell'invito da parte dell'utente remoto e metadati ad esso associati

L'unica differenza tra i metadati dei messaggi di rifiuto da parte dell'utente remoto o da parte dell'utente locale è il valore del metadato 'local' che nel primo caso vale False (10) mentre nel secondo caso vale True (11). Stesso discorso per il messaggio di accettazione.

Il messaggio di risposta all'invito nel campo RAW contiene l'id del forum e, se la risposta all'invito non è la prima risposta a un invito a quel forum da parte di quell'utente remoto, il riferimento all'id del messaggio di risposta precedente.

Se l'invitato accetta l'invito (sia che l'utente locale sia l'invitato sia che sia l'invitante), oltre alle varie modifiche specificate fino ad ora, è salvata la visibilità del forum (CLIENTID org.briarproject.briar.forum) relativamente al contatto che ha accettato l'invito.

Più nello specifico nella tabella GROUPVISIBILITIES il campo SHARED relativo al gruppo associato al CLIENTID org.briarproject.briar.forum e al contatto che ha accettato l'invito o che ha inviato l'invito accettato dall'utente locale è settato a True.

Inoltre, se l'utente locale è l'invitato nella tabella GROUPS è aggiunto un gruppo relativo al CLIENTID org.briarproject.briar.forum.

Infine, se l'utente ha ricevuto più inviti allo stesso forum, i messaggi di risposta agli inviti sono messi in relazione tra loro nella tabella MESSAGEDEPENDENCIES.

Come si può notare in Figura 4.85, il precedente messaggio di rifiuto dell'invito a partecipare a un forum A da parte di un contatto B è in relazione di dipendenza con il messaggio di accettazione del successivo invito a partecipare allo stesso forum A da parte sempre del contatto B. Questo permette di creare uno storico preciso di tutti gli inviti a partecipare a uno stesso forum ricevuti dallo stesso utente.

CLIENTID	MessageID	MessageMeta	DependencyID	DependencyMeta
1 org.briarproject.briar.forum.sharing	84c7b874e797308fc789825...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2101 read=10 shareableId=5120314e490eca66a0e766a5e91... timestamp=280000018fe7f4c722 visibleInUi=11		9fe09a8e33dc67a20b26a1f...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2102 read=10 shareableId=5120314e490eca66a0e766a5e91... timestamp=280000018fe7f2f556 visibleInUi=11

Figura 4.85: Dati relativi ai messaggi di risposta a inviti allo stesso forum in MESSAGEDEPENDENCIES e metadati ad essi associati

4.15.3 - Query

Per ottenere tutti i messaggi relativi ai CLIENTID org.briarproject.briar.forum e org.briarproject.briar.forum.sharing ordinati per TIMESTAMP è possibile eseguire la seguente query.

```
SELECT g.CLIENTID, g.GROUPID, gm.METAKEY, gm.VALUE, m.MESSAGEID, TIMESTAMP,
       message_meta.MessageMeta
FROM
  GROUPS g LEFT JOIN GROUPMETADATA gm ON g.GROUPID = gm.GROUPID
  LEFT JOIN MESSAGES m ON g.GROUPID = m.GROUPID
  JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS
           MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
  ) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE
  (g.CLIENTID = "org.briarproject.briar.forum.sharing" AND gm.METAKEY =
  "contactId")
  OR (g.CLIENTID = "org.briarproject.briar.forum" and gm.METAKEY =
  "messageCount")
ORDER BY TIMESTAMP
```

4.16 - Messaggi nel forum

4.16.1 - Esperimenti

Per analizzare gli artefatti generati dai messaggi inviati in un forum si è reso necessario definire un esperimento in cui l'utente locale (A) pubblica un post nel forum (sotto-esperimenti 1, 2 e 3 dell'esperimento 33) e un esperimento in cui è l'utente remoto (B) che scrive un post nel forum (sotto-esperimenti 1, 2 e 3 dell'esperimento 34).

Numero esperimento	Descrizione	Sotto-esperimenti
33	Post dell'utente locale in un forum, con mittente e destinatario offline a turno o entrambi gli utenti online	<ol style="list-style-type: none">1. A crea un post (A offline e B online)2. A crea un post (A online e B offline)3. A crea un post (A e B online)4. B replica al post di A
34	Post dell'utente remoto in forum, con mittente e destinatario offline a turno o entrambi gli utenti online	<ol style="list-style-type: none">1. B crea un post (B offline e A online)2. B crea un post (B online e A offline)3. B crea un post (A e B online)

Tabella 4.16: Lista esperimenti post in forum

Per emulare la pubblicazione di un post in un forum da parte dell'utente locale (A) (sotto-esperimenti 1, 2 e 3 dell'esperimento 33) è stato impiegato il sottostante file delle azioni.

che ha inviato o ricevuto il messaggio, dal timestamp del messaggio e da un BDF che racchiude il nome dell'autore del post, la public key dell'autore e il testo del post (Il post è salvato come messaggio solo nel momento in cui entrambi i contatti sono connessi a briar nello stesso momento).

Come ogni messaggio, anche questo ha dei metadati associati da cui è possibile ottenere diverse informazioni. Più nello specifico, questo messaggio ha associati i seguenti metadati:

- **author:** contiene il nome dell'autore del post e la sua public key
- **local:** definisce se il messaggio è locale.
Questo metadato è presente e vale True (11) se il post è stato scritto dall'utente locale. Se invece il post è stato scritto da un utente remoto, il messaggio del post non avrà questo metadato
- **read:** definisce se il post è stato letto dall'utente locale.
Vale True (11) se il post è stato letto altrimenti vale False (10)
Se il messaggio è locale sicuramente risulterà letto dall'utente locale.
- **timestamp:** contiene il timestamp del messaggio

Di seguito si riportano un esempio di messaggio di post in un forum da parte dell'utente locale (Figura 4.88) e un esempio di messaggio di post in un forum da parte dell'utente remoto (Figura 4.89).

	GROUPID	MESSAGEID	MessageMeta
1	9f34aa50e48813a0de262b0d500149b1b2daac...	411171957210443b248641706d03ea4494c028...	author=602101410550726f766151205190bb2... local=11 read=11 timestamp=280000018fe8a13c3e

Figura 4.88: Dati relativi a un post dell'utente locale in un forum e metadati ad esso associati

	GROUPID	MESSAGEID	MessageMeta
1	3f543da7b635d35f80aaf94e7a7d8887b9d54c...	c655b5a0d3b11c65e65a268e5ee4fb1b6ceb2a...	author=6021014106416d69636f315120eeff26... read=11 timestamp=280000018fe8dd99b1

Figura 4.89: Dati relativi a un post di un utente remoto in un forum e metadati ad esso associati

In caso il post pubblicato nel forum sia un re-post di un post precedente, il messaggio di re-post (Figura 4.90), nel suo campo RAW conterrà l'id del messaggio di post a cui il re-post fa riferimento e tutte le altre informazioni presenti in un qualsiasi messaggio di post (il nome dell'autore del re-post, la sua public key e il testo del re-post).

	GROUPID	MESSAGEID	MessageMeta
1	9f34aa50e48813a0de262b0d500149b1b2daac...	bc14a4757675d8b08de4c0c59d9e8bd4adc43f...	author=6021014106416d69636f315120eeff26... parent=512041576df6c5b2ef7f4349d7ed05ef0... read=10 timestamp=280000018fe8a4629e

Figura 4.90: Dati relativi a un messaggio di un re-post e metadati ad esso associati

I metadati del messaggio di re-post sono gli stessi di un messaggio di post (compreso il fatto che local è presente come metadato solo se il re-post è scritto dall'utente locale) con l'aggiunta di un metadato che permette di ottenere l'id del messaggio di post a cui il re-post fa riferimento ('parent').

È possibile recuperare l'associazione tra messaggio di post e messaggio di re-post dalla tabella MESSAGEDEPENDENCIES (Figura 4.91).

	CLIENTID	MessageID	MessageTimestamp	MessageMeta	DependencyID	DependencyTimestamp	DependencyMeta
1	org.briarproject.briar.forum	bc14a4757675d8...	1717595038366	author=60210141... parent=512041576... read=10 timestamp=28000...	41576df6c5b2ef7f434...	1717594916360	author=602101410550... local=11 read=11 timestamp=28000001...

Figura 4.91: Associazione tra messaggio di post e messaggio di re-post di un forum

Nella Figura 4.91 è possibile notare come in caso di re-post in un forum, nella tabella MESSAGEDEPENDENCIES sia aggiunta una tupla che mette in relazione i due messaggi. In questa tupla, il campo MessageID contiene l'id del messaggio di post, mentre nel campo DependencyID è memorizzato l'id del messaggio di re-post.

4.16.3 - Query

Per recuperare tutti i messaggi sia di post che di re-post scambiati in tutti i forum di cui l'utente fa parte, è necessario eseguire la query successiva.

```
SELECT g.CLIENTID, g.GROUPID, m.MESSAGEID, m.RAW, TIMESTAMP,
       message_meta.MessageMeta
FROM GROUPS g JOIN MESSAGES m USING (GROUPID)
JOIN (
  SELECT m.MESSAGEID,
         GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
  FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
  GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE g.CLIENTID = "org.briarproject.briar.forum"
ORDER BY TIMESTAMP
```

Per ottenere, invece le associazioni tra post e re-post basta eseguire la seguente query

```
SELECT G.CLIENTID, md.MESSAGEID AS MessageID,
       msg_meta.TIMESTAMP AS MessageTimestamp, msg_meta.MessageMeta,
       md.DEPENDENCYID AS DependencyID, dep_meta.TIMESTAMP AS
DependencyTimestamp,
       dep_meta.DependencyMeta
FROM GROUPS G JOIN MESSAGEDEPENDENCIES md USING(GROUPID)
LEFT JOIN (
  SELECT MESSAGEID, TIMESTAMP,
         GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
  FROM MESSAGEMETADATA JOIN MESSAGES USING(MESSAGEID)
  GROUP BY MESSAGEID
) AS msg_meta ON md.MESSAGEID = msg_meta.MESSAGEID
LEFT JOIN (
  SELECT MESSAGEID, TIMESTAMP,
         GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS
DependencyMeta
  FROM MESSAGEMETADATA JOIN MESSAGES USING(MESSAGEID)
  GROUP BY MESSAGEID
) AS dep_meta ON md.DEPENDENCYID = dep_meta.MESSAGEID
WHERE CLIENTID = "org.briarproject.briar.forum"
```

4.17 - Abbandono del forum

4.17.1 - Esperimenti

Per effettuare uno studio completo di questa funzionalità sono stati pensati principalmente due esperimenti che permettessero di prendere in considerazione l'abbandono del forum da due punti di vista differenti:

- L'abbandono da parte dell'utente che ha creato il forum (esperimenti 35 e 37)
- L'abbandono da parte dell'utente che è stato invitato a partecipare al forum (esperimenti 36 e 38)

Numero esperimento	Descrizione	Sotto-esperimenti
35	Abbandono del forum da parte dell'utente locale creatore (A) (database creatore A)	1. A abbandona il forum
36	Abbandono del forum da parte dell'utente remoto membro (B) (database creatore A)	1. B abbandona il forum 2. A abbandona il forum
37	Abbandono del forum da parte dell'utente remoto creatore (B) (database membro A)	1. B abbandona il forum
38	Abbandono del forum da parte dell'utente locale membro A (database membro A)	1. A abbandona il forum 2. B abbandona il forum

Tabella 4.17: Lista esperimenti di abbandono forum

Gli esperimenti 35 e 37 sono stati eseguiti per ottenere rispettivamente gli artefatti generati quando l'utente locale che ha creato il forum lo abbandona e quando l'utente remoto, creatore del forum, lo abbandona. Mentre gli esperimenti 36 e 38 hanno permesso di ottenere gli artefatti generati a seguito dell'abbandono del forum da parte di un semplice membro rispettivamente nel database dell'utente che ha creato il forum e nel database di un utente che è solo un membro.

Per simulare le azioni essenziali per eseguire l'abbandono di un forum da parte dell'utente locale (A), sia esso il creatore del forum o meno, è stato usato il seguente file delle azioni.

Se, al contrario è l'utente remoto ad abbandonare il forum, nel database dell'utente locale, oltre ai messaggi associati al CLIENTID org.briarproject.briar.forum.sharing, rimangono memorizzati anche tutti i messaggi di post e i re-post (messaggi associati al CLIENTID org.briarproject.briar.forum) del forum in quanto l'utente locale non ha abbandonato il forum e quindi deve ancora poter visualizzare tali dati.

Se è l'utente locale ad abbandonare il forum, viene generato un messaggio di abbandono il cui campo RAW contiene informazioni diverse a seconda che l'utente locale abbia invitato qualcuno o sia stato invitato nel forum.

Nel primo caso (utente locale invitante), nel BDF del campo RAW del messaggio saranno presenti l'id del forum che l'utente locale sta abbandonando e l'id dell'ultimo messaggio di invito a partecipare a quel forum inviato precedentemente dall'utente locale e accettato da quello specifico utente.

Nel secondo caso (utente locale invitato), invece, nel BDF del campo RAW del messaggio si troveranno l'id del forum che si sta abbandonando e l'id dell'ultimo messaggio di accettazione a partecipare al forum.

Il metadato associato al messaggio di abbandono e che permette di riconoscerlo è il metadato 'messageType' con valore uguale a LEAVE (3). Se il messaggio è inviato dall'utente locale avrà metadato 'local' uguale a True (11) altrimenti avrà valore uguale a False (10).

Di seguito si riporta un esempio di messaggio di abbandono inviato dall'utente locale (Figura 4.93).

GROUPID	MESSAGEID	MessageMeta
1 bd2ff9a396118851aa78bbf322318702b45794...	bb861a307eea059518bdb37c60c065c225809b...	availableToAnswer=10 invitationAccepted=10 local=11 messageType=2103 read=11 shareableId=5120314e490eca66a0e766a5e91... timestamp=280000018fe7f61965 visibleInUi=10

Figura 4.93: Dati relativi a un messaggio di abbandono del forum e metadati ad esso associati

Se il primo ad abbandonare il forum è un utente remoto, nel database dell'utente locale sono generati due messaggi:

- Un messaggio di abbandono del forum da parte dell'utente locale e associato al gruppo org.briarproject.briar.forum.sharing relativo al contatto remoto che ha abbandonato il gruppo.
Il BDF del campo RAW di questo messaggio contiene l'id del forum insieme o all'id dell'ultimo messaggio di accettazione a partecipare al forum (se l'utente locale è stato invitato) o all'id dell'ultimo messaggio di invito accettato dall'utente remoto (se l'utente locale ha invitato l'utente remoto).
Questo messaggio ha gli stessi metadati del messaggio di abbandono del forum in Figura 4.93.
- Un messaggio di abbandono del forum da parte dell'utente remoto.
Il BDF del campo RAW permette di ottenere le informazioni dell'id del forum che si sta abbandonando insieme o all'id dell'ultimo messaggio di accettazione a partecipare al forum (se l'utente remoto è stato invitato) o all'id dell'ultimo messaggio di invito accettato dall'utente remoto (se l'utente remoto ha invitato l'utente remoto).

Questo messaggio ha gli stessi metadati del messaggio di abbandono del forum in Figura 4.93.

Se, al contrario, il primo ad abbandonare il forum è l'utente locale, nel database dell'utente locale è generato solo l'ultimo dei due messaggi precedenti.

Come illustrato nella Figura 3.6 della Sezione 3.1.3.5, a seguito dell'abbandono del forum da parte di un utente, lo stato del client di condivisione (org.briarproject.briar.forum.sharing) cambia il proprio valore del seguente modo:

- Se l'utente locale abbandona il forum, lato utente locale, lo stato del client di condivisione del forum diventa LOCAL_LEFT (4).
- Se l'utente remoto abbandona il forum, lato utente locale, lo stato del client di condivisione del forum diventa START (0). Lo stato rimane tale se l'utente locale successivamente abbandona il forum.

4.17.3 - Query

Per controllare se ci sono forum che l'utente ha abbandonato si può eseguire la query seguente

```
SELECT m.GROUPID, m.MESSAGEID, message_meta.MessageMeta
FROM MESSAGES m
JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE
    (MessageMeta LIKE "%state=2104%")
    OR ( MessageMeta LIKE "%state=2100%")
```

4.18 - Import di un feed RSS

4.18.1 - Esperimenti

Per analizzare la funzionalità di import di un feed RSS nel blog dell'utente locale è bastato il sotto-esperimento 1 dell'esperimento 39.

Numero esperimento	Descrizione	Sotto-esperimenti
39	Aggiunta di un feed RSS nel blog dell'utente locale	<ol style="list-style-type: none"> 1. A importa un feed RSS 2. A cancella il feed RSS

Tabella4.18: Lista esperimenti di import di un feed RSS in blog

Per simulare le azioni necessarie per eseguire il sotto-esperimento 1 dell'esperimento 39 è stato utilizzato il seguente file delle azioni

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,2
7 # Import di un feed RSS nel blog dell'utente
8 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
  on[@content-desc='Open the navigation drawer']
9 #SLEEP
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.widget.ScrollView[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/androidx.recyclerview.
  widget.RecyclerView[1]/androidx.appcompat.widget.LinearLayoutCompat[4]/android.widget.CheckedTextView[@resource-id='
  org.briarproject.briar.android:id/design_menu_item_text']
11 #SLEEP
12 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.appcompat.widget
  .LinearLayoutCompat[1]/android.widget.ImageView[@content-desc='More options']
13 #SLEEP
14 Tap,RES_ID:org.briarproject.briar.android:id/title
15 #SLEEP
16 Tap,RES_ID:org.briarproject.briar.android:id/action_rss_feeds_import
17 #SLEEP
18 SendKeys,RES_ID:org.briarproject.briar.android:id/urlInput,TEXT:https://feeds.bbc.co.uk/news/rss.xml?edition=uk
19 HideKeyboard
20 SLEEP,2
21 Tap,RES_ID:org.briarproject.briar.android:id/importButton
22 SLEEP,60
23 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.
  ViewGroup[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navi-
  gate up']
24 SLEEP,1
25 # Apertura menù laterale
26 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wid-
  get.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButt-
  on[@content-desc='Open the navigation drawer']
27 # Sign-out in Briar
28 SLEEP,2
29 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
30 SLEEP,5

```

Figura 4.94: File delle azioni per l'aggiunta di un feed RSS nel blog dell'utente locale

Nella riga 10 della Figura 4.94 si entra nella sezione 'Blogs' dell'applicazione (Figura 2.45 della Sezione 2.7.1).

Nelle righe 12 e 14 della Figura 4.94 si apre il menù del blog e si accede alla sezione per gestire i feed RSS (Figura 2.46 della Sezione 2.7.1).

Nella riga 16 della Figura 4.94 si esegue il Tap sul tasto per aggiungere un feed RSS.

Nelle righe 18 e 21 della Figura 4.94 si inserisce l'url del feed RSS e si esegue il Tap sul tasto di import (Figura 2.47 della Sezione 2.7.1).

Come si può notare, nella Figura 4.94, è stato aggiunto un preciso feed RSS (<https://feeds.bbc.co.uk/news/rss.xml?edition=uk>) che è stato utilizzato come feed di test per analizzare gli artefatti generati a seguito dell'aggiunta.

4.18.2 - Artefatti generati

Quando un feed RSS è importato in Briar, il feed è salvato nella tabella GROUPS come se fosse il blog di un utente Briar (CLIENTID org.briarproject.briar.blog). Dal campo DESCRIPTOR della tabella è possibile recuperare l'autore del feed (in questo caso BBC) e se il gruppo è un feed RSS o meno.

L'aggiunta di un feed RSS implica, inoltre, che nella tabella GROUPMETADATA sia aggiunto al metadato feeds relativo al CLIENTID org.briarproject.briar.feed diverse informazioni sul feed appena inserito. Infatti, il valore di questo metadato è un BDF che per ogni feed RSS dell'utente locale contiene le seguenti informazioni:

- il timestamp di aggiunta del feed RSS
- l'autore del feed
- la descrizione del feed
- il timestamp dell'ultima notizia del feed
- il link del feed
- il titolo del feed
- l'url del feed
- il timestamp dell'aggiornamento del feed

All'import di un feed RSS sono scaricate tutte le notizie del feed, ognuna delle quali è gestita come un messaggio di post del blog (Sezione 4.20.2).

4.18.3 - Query

Per visualizzare e ottenere tutti i feed RSS importati da un utente bisogna eseguire la seguente query

```
SELECT * FROM GROUPMETADATA WHERE METAKEY = "feeds"
```

Una volta ottenuto il risultato della query, basta tradurre il BDF (Sezione 3.1.2.1) presente nel campo VALUE e si otterranno tutte le informazioni di tutti i feed dell'utente locale.

4.19 - Condivisione di un feed RSS

4.19.1 - Esperimenti

Per la funzionalità di condivisione di un feed RSS è stato utilizzato un solo esperimento in cui si è simulato dapprima il rifiuto (sotto-esperimento 1) e successivamente l'accettazione (sotto-esperimento 2) della richiesta di condivisione del feed RSS.

Numero esperimento	Descrizione	Sotto-esperimenti
40	Condivisione di un feed RSS da parte dell'utente locale (A)	<ol style="list-style-type: none"> 1. A condivide RSS feed con B e B rifiuta la condivisione dell'RSS feed 2. A condivide RSS feed con B e B accetta la condivisione dell'RSS feed 3. A cancella RSS feed

Tabella 4.19: Lista esperimenti di condivisione del feed RSS

Per eseguire la condivisione di un feed RSS è stato necessario utilizzare un file delle azioni di cui si riportano solo le istruzioni che effettuano le azioni di condivisione del feed RSS.

```

1 # Condivisione di un feed RSS con un proprio contatto
2 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
dget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.Draw-
erLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/androidx.appcompat.
widget.LinearLayoutCompat[1]/android.widget.ImageView[@content-desc='More options']
3 #SLEEP
4 Tap,RES_ID:org.briarproject.briar.android:id/title
5 #SLEEP
6 Tap,RES_ID:org.briarproject.briar.android:id/importedView
7 #SLEEP
8 Tap,RES_ID:org.briarproject.briar.android:id/action_blog_share
9 #SLEEP
10 Tap,RES_ID:org.briarproject.briar.android:id/checkbox
11 #SLEEP
12 Tap,RES_ID:org.briarproject.briar.android:id/action_contacts_selected
13 #SLEEP
14 SendKeys,RES_ID:org.briarproject.briar.android:id/input_text,TEXT:Condivisione dell'intero RSS feed
15 HideKeyboard
16 #SLEEP
17 Tap,RES_ID:org.briarproject.briar.android:id/compositeSendButton
18 SLEEP,30

```

Figura 4.95: Parte del file delle azioni per la condivisione di un feed RSS da parte dell'utente locale

Nelle righe 2 e 4 della Figura 4.95 si apre il menù del blog e si accede alla sezione per gestire i feed RSS (Figura 2.48 della Sezione 2.7.2 e Figura 2.49 della Sezione 2.7.2).

Nella riga 6 della Figura 4.95 si seleziona il feed RSS che si vuole condividere.

Nelle righe 10 e 12 della Figura 4.95 si selezionano i contatti con cui si vuole condividere il feed RSS e si dà conferma (Figura 2.50 della Sezione 2.7.2).

Nelle righe 14 e 17 della Figura 4.95 si inserisce il testo che accompagna la richiesta di condivisione del feed RSS e si invia la richiesta.

4.19.2 - Artefatti generati

Quando un feed RSS è condiviso con un utente remoto è generato:

- un messaggio di configurazione del gruppo relativo al BSP client `org.briarproject.briar.blog.sharing` con metadato `state` uguale a `START (0)` (Figura 4.96)

GROUPID	MESSAGEID	MessageMeta
1 b04f8719bc42446a78a1bfedf4103ea5739b777...	716ce11a9a081573a2ec8f681a1e08f5b4af3cc...	inviteTimestamp=2100 isSession=11 lastLocalMessageId=5120a7fcd237e7692e287... lastRemoteMessageId=5120e205169e46235d... localTimestamp=280000018fed09d752 sessionId=51204f504e7d3a8aebc2fd4ff6520c7... shareableId=51204f504e7d3a8aebc2fd4ff652... state=2100

Figura 4.96: Dati relativi a un messaggio di configurazione del gruppo relativo al CLIENTID `org.briarproject.briar.blog.sharing` e messaggi ad esso associati

- Un messaggio di richiesta di condivisione del feed RSS con metadato `'messageType'` uguale a `INVITE (0)` (Figura 4.97)
Dal BDF presente nel campo RAW di questo messaggio è possibile ottenere il tipo di messaggio, l'id del messaggio precedente della sessione, il nome dell'autore del blog e se è un feed RSS o meno.

	GROUPID	MESSAGEID	MessageMeta
1	b04f8719bc42446a78a1bfedf4103ea5739b777...	a7fcd237e7692e287f7ad47cb8a0d6257a418b...	availableToAnswer=10 invitationAccepted=10 local=11 messageType=2100 read=11 shareableId=51204f504e7d3a8aebc2fd4ff652... timestamp=280000018fed09d752 visibleInUi=11

4.97: Dati relativi a un messaggio di invito a condividere il feed RSS e metadati ad esso associati

L'utente che riceve l'invito di condivisione, esattamente come per gli inviti ai gruppi privati e ai forum, può accettare o declinare l'offerta.

La risposta dell'invitato è salvata in un messaggio che ha metadato 'messageType' uguale a ACCEPT (1) se l'utente ha accettato l'offerta (Figura 4.99) altrimenti ha metadato 'messageType' uguale a DECLINE (2) (Figura 4.98).

Dal BDF presente nel campo RAW del messaggio di risposta è possibile ottenere il tipo di messaggio, l'id del blog condiviso e, se presente, l'id del messaggio di risposta all'invito precedente di condivisione dello stesso feed.

	GROUPID	MESSAGEID	MessageMeta
1	b04f8719bc42446a78a1bfedf4103ea5739b777...	e205169e46235de3dcef61c7b6773eabb5c2e7...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2102 read=10 shareableId=51204f504e7d3a8aebc2fd4ff652... timestamp=280000018fed09ea13 visibleInUi=11

Figura 4.98: Dati relativi a un messaggio di rifiuto dell'invito da parte dell'utente remoto e metadati ad esso associati

	GROUPID	MESSAGEID	MessageMeta
1	b04f8719bc42446a78a1bfedf4103ea5739b777...	55e419c0dc498e79900bb0be577c0d67295909...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2101 read=10 shareableId=51204f504e7d3a8aebc2fd4ff652... timestamp=280000018fed0bb705 visibleInUi=11

Figura 4.99: Dati relativi a un messaggio di accettazione dell'invito da parte dell'utente remoto e metadati ad esso associati

Inoltre, se l'utente accetta la richiesta di condivisione feed RSS lo stato del gruppo associato al BSP client org.briarproject.briar.blog.sharing relativo a quel contatto remoto assume valore SHARING (3).

Infine, quando un feed RSS è condiviso con un altro utente remoto l'id del blog condiviso relativo al CLIENTID org.briarproject.briar.blog è presente nella tabella GROUPVISIBILITIES con valore del campo SHARED uguale a True (1).

4.19.3 - Query

Per ottenere tutte le richieste di condivisione di feed RSS inviate dall'utente locale basta eseguire la seguente query

```
SELECT g.CLIENTID, g.GROUPID, gm.METAKEY, gm.VALUE, m.MESSAGEID, TIMESTAMP,
       message_meta.MessageMeta
FROM GROUPS g LEFT JOIN GROUPMETADATA gm ON g.GROUPID = gm.GROUPID
JOIN MESSAGES m ON g.GROUPID = m.GROUPID
```

```

JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS
    MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE
    (g.CLIENTID = "org.briarproject.briar.blog.sharing" AND gm.METAKEY =
    "contactId")
ORDER BY TIMESTAMP

```

4.20 - Messaggi nel blog

4.20.1 - Esperimenti

Questa funzionalità ha richiesto l'esecuzione di due esperimenti per poter ottenere gli artefatti generati nel database dell'utente locale sia quando è quest'ultimo (A) a scrivere un post sul proprio blog sia quando è un utente remoto (B) a pubblicare un post sul proprio blog.

Inoltre, si è reputato necessario controllare quali artefatti fossero generati in situazioni diverse di connettività degli utenti interessati.

Numero esperimento	Descrizione	Sotto-esperimenti
41	Post dell'utente locale (A) nel proprio blog con mittente e destinatario offline a turno o entrambi gli utenti online e re-blog dell'utente remoto (B)	<ol style="list-style-type: none"> 1. A scrive un post sul suo blog (A offline e B online) 2. A scrive un post sul suo blog (A online e B offline) 3. A scrive un post sul suo blog (A e B online) 4. B re-blogga il post di A
42	Post dell'utente remoto (B) nel proprio blog con mittente e destinatario offline a turno o entrambi gli utenti online e re-blog dell'utente locale (A)	<ol style="list-style-type: none"> 1. B scrive un post sul suo blog 2. A re-blogga il post di B (A offline e B online) 3. A re-blogga il post di B (A online e B offline) 4. A re-blogga il post di B (A e B online)

Tabella 4.20: Lista esperimenti post nel blog

Siccome i feed RSS, in Briar sono considerati come blog e le loro notizie sono considerate come post di un blog, in questa Sezione si prenderà in considerazione anche il re-blog di una notizia di un feed RSS.

Per analizzare tale funzionalità si è eseguito un solo esperimento in cui si è effettuato il re-blog di una notizia di un feed RSS in situazioni di connettività differenti.

Numero esperimento	Descrizione	Sotto-esperimenti
--------------------	-------------	-------------------

Nel caso di messaggio di post questo metadato ha valore POST (0)

GROUPID	MESSAGEID	MessageMeta
1 9b78c1de555f9b78bdb8bf92134d314ab8b84d...	40dc5452b7af6eb82291ee6a192f5d4a7dfdb55...	author=602101410550726f7661512077b9bce... read=11 rssFeed=10 timestamp=280000018fed168d11 type=2100

Figura 4.102: Dati relativi a un messaggio di post e metadati ad esso associati

Dal BDF presente nel campo RAW relativo al messaggio di post è, inoltre possibile recuperare il testo del messaggio di post.

Se l'utente re-blogga un post di un blog sono generati due messaggi:

- Un messaggio di tipo WRAPPED_POST (2) che contiene il post che si è commentato. Questo messaggio è generato solo al primo re-blog di un post
- Un messaggio di commento al post

Il primo messaggio (Figura 4.103) ha i seguenti metadati:

- author: contiene un BDF in cui è presente il nome dell'autore del post
- originalMessageId: contiene l'id del messaggio originale, cioè, l'id del messaggio di post
- rssFeed: definisce se il messaggio è un feed RSS o meno. Vale False (10) se il messaggio è un post di un blog, mentre vale True (11) se il messaggio è una notizia pubblicata da un feed RSS importato nel blog dell'utente locale
- timeReceived: contiene il timestamp di ricezione del messaggio. Se il messaggio è stato inviato dall'utente locale questo metadato non è presente
- type: contiene il tipo di messaggio. Nel caso di re-post il messaggio che contiene il post commentato ha il valore di questo metadato settato a WRAPPED_POST (2)

GROUPID	MESSAGEID	MessageMeta
1 cfde068e7e890757f1d0b71e48101639e8548d...	ff911ffcf65cdfb49fcef5f9c8781dab683d671...	author=6021014106416d69636f315120c2ae6c... originalMessageId=5120400521fcfd66106d2... rssFeed=10 timeReceived=280000018feda34bf0 timestamp=280000018feda33cbb type=2102

Figura 4.103: Dati relativi a un messaggio di tipo WRAPPED_POST e metadati ad esso associati

Il messaggio di commento al post (Figura 4.104) invece ha i seguenti metadati:

- author: contiene un BDF in cui è presente il nome dell'autore del commento
- comment: contiene un BDF in cui è salvato il testo del commento
- originalMessageId: contiene l'id del messaggio originale, cioè, l'id del commento stesso
- originalParentMessageId: contiene un BDF in cui è presente l'id originale del precedente messaggio di commento a quel post o l'id del messaggio di post che si sta commentando
- parentMessageId: contiene un BDF in cui è presente l'id del precedente messaggio di commento al post o l'id del messaggio di tipo WRAPPED_POST
- read: definisce se il post è stato letto dall'utente locale
- timestamp: contiene il timestamp del messaggio
- type: contiene il tipo di messaggio.

Nel caso di messaggio di post questo metadato ha valore COMMENT (1)

	GROUPID	MESSAGEID	MessageMeta
1	cfde068e7e890757f1d0b71e48101639e8548d...	6f5ed78f680f5749f3385b23cbd9d99ca5d7b6d...	author=602101410550726f76615120e2134eb... comment=412f52652d706f737420646920412... originalMessageId=51206f5ed78f680f5749f33... originalParentMessageId=512082a4bfc9011cc... parentMessageId=512082a4bfc9011ccde8898... read=11 timestamp=280000018feda8f84c type=2101

Figura 4.104: Dati relativi a un messaggio di re-post e metadati ad esso associati

Esattamente come per il re-blog di un post del blog anche quando un utente re-blogga una notizia del feed RSS è generato un messaggio relativo alla notizia re-bloggata (messageType WRAPPED_POST) e un messaggio relativo al commento del post (messageType uguale a COMMENT).

All'interno del BDF contenuto nel campo RAW relativo al primo messaggio, sia che si parli di re-blog di un post sia che si parli di un re-blog di una notizia di un feed RSS, è possibile recuperare il tipo di messaggio, il nome dell'autore del feed, l'anteprima della notizia re-bloggata e il timestamp di quando è stata pubblicata la notizia sul dispositivo.

4.20.3 - Query

Per ottenere tutti i messaggi di post di un blog o di re-blog di un post o di una notizia di un feed RSS basta eseguire la seguente query

```
SELECT g.CLIENTID, g.GROUPID, m.MESSAGEID, TIMESTAMP,
message_meta.MessageMeta
FROM GROUPS g LEFT JOIN GROUPMETADATA gm ON g.GROUPID = gm.GROUPID
JOIN MESSAGES m ON g.GROUPID = m.GROUPID
JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS
           MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE (g.CLIENTID = "org.briarproject.briar.blog")
ORDER BY TIMESTAMP
```

Il risultato di questa query permette di ottenere anche uno storico di tutti i commenti a un determinato post.

4.21 - Cancellazione di un feed RSS

4.21.1 - Esperimenti

Per analizzare tale funzionalità è stato necessario eseguire un solo esperimento che effettuasse la cancellazione del feed RSS importato dall'utente locale.

Numero esperimento	Descrizione	Sotto-esperimenti
44	Cancellazione feed RSS da parte dell'utente locale	1. A cancella il feed RSS importato

Tabella 4.22: Lista esperimenti cancellazione di un feed RSS importato

Per il messaggio di abbandono dell'utente che ha condiviso il feed RSS quest'ultimo dato contiene l'id dell'ultimo messaggio di richiesta di condivisione. Per il messaggio di abbandono dell'utente che ha accettato la condivisione del feed RSS quest'ultimo dato contiene l'id dell'ultimo messaggio di risposta alla richiesta di condivisione.

Come si può notare dalle figure sottostanti, i metadati del messaggio di abbandono sono uguali, indipendentemente dal fatto che l'utente che sta abbandonando la condivisione del feed RSS sia l'utente locale (Figura 4.106) o l'utente remoto (Figura 4.107).

	GROUPID	MESSAGEID	MessageMeta
1	b04f8719bc42446a78a1bfedf4103ea5739b777...	267b918bc015a734f49867ef4b8f5624d8ae8f5...	availableToAnswer=10 invitationAccepted=10 local=11 messageType=2103 read=11 shareableId=51204f504e7d3a8aebc2fd4ff652... timestamp=280000018fed0d33f8 visibleInUi=10

Figura 4.106: Dati relativi a un messaggio di abbandono della condivisione del feed RSS da parte dell'utente locale e metadati ad esso associati

	GROUPID	MESSAGEID	MessageMeta
1	b04f8719bc42446a78a1bfedf4103ea5739b777...	be629e1d7c510e2c85112d517817937bd8e9f9...	availableToAnswer=10 invitationAccepted=10 local=10 messageType=2103 read=10 shareableId=51204f504e7d3a8aebc2fd4ff652... timestamp=280000018fed0d4a36 visibleInUi=10

Figura 4.107: Dati relativi a un messaggio di abbandono della condivisione del feed RSS da parte dell'utente remoto e metadati ad esso associati

Quando un feed RSS condiviso è cancellato lo stato del BSP client con l'utente remoto va in stato START (Figura 3.7 della Sezione 3.1.3.7).

4.21.3 - Query

Per controllare se ci sono dei feed RSS che l'utente locale ha abbandonato basta eseguire la seguente query.

```
SELECT g.CLIENTID, g.GROUPID, gm.METAKEY, gm.VALUE, m.MESSAGEID, TIMESTAMP,
       message_meta.MessageMeta
FROM GROUPS g LEFT JOIN GROUPMETADATA gm ON g.GROUPID = gm.GROUPID
JOIN MESSAGES m ON g.GROUPID = m.GROUPID
JOIN (
    SELECT m.MESSAGEID,
           GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS
           MessageMeta
    FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
    GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE
    (g.CLIENTID = "org.briarproject.briar.blog.sharing" AND gm.METAKEY =
    "contactId")
    AND message_meta.MessageMeta LIKE "%messageType=2103%"
ORDER BY TIMESTAMP
```

4.22 - Modifica immagine del profilo

4.22.1 - Esperimenti

Nell'ambito della sezione delle Impostazioni si è reputato forensicamente interessante analizzare gli artefatti generati dalla modifica dell'immagine di profilo degli utenti. Su tale funzionalità si è deciso di eseguire due esperimenti che permettessero di valutare gli artefatti generati nel database dell'utente locale sia se è quest'ultimo a modificare la propria immagine del profilo sia se è uno dei contatti dell'utente locale a modificare la propria immagine del profilo.

Numero esperimento	Descrizione	Sotto-esperimenti
45	Modifica dell'immagine di profilo	<ol style="list-style-type: none">1. A modifica la sua immagine del profilo di Briar2. B modifica la sua immagine del profilo di Briar

Tabella 4.23: Lista esperimenti modifica immagine del profilo

Per simulare le azioni di cambio immagine del profilo sia lato utente locale (A) sia lato utente remoto (B), è stato utilizzato il seguente file delle azioni.

```

1 # Sign-in in Briar
2 SendKeys,RES_ID:org.briarproject.briar.android:id/edit_password,TEXT:1234+'
3 HideKeyboard
4 #SLEEP
5 Tap,RES_ID:org.briarproject.briar.android:id/btn_sign_in
6 SLEEP,150
7 # Cambio immagine del profilo di A (utente locale)
8 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
  dget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.Draw-
  erLayout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.view.ViewGroup[1]/android.widget.Ima-
  geButton[@content-desc='Open the navigation drawer']
9 #SLEEP
10 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
  dget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.Draw-
  erLayout[1]/android.widget.ScrollView[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/androidx.recycle-
  rview.widget.RecyclerView[1]/androidx.appcompat.widget.LinearLayoutCompat[5]/android.widget.CheckedTextView[@reso-
  urce-id='org.briarproject.briar.android:id/design_menu_item_text']
11 #SLEEP
12 Tap,RES_ID:org.briarproject.briar.android:id/avatarImage
13 SLEEP,5
14 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
  dget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewG-
  roup[1]/android.widget.ImageButton[@content-desc='Show roots']
15 SLEEP,2
16 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
  dget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.Scroll-
  view.widget.RecyclerView[1]/android.widget.LinearLayout[1]/android.widget.LinearLayout[1]/android.widget.TextView[1]
17 SLEEP,2
18 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
  dget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/androidx.drawerlayout.widget.DrawerLa-
  yout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[2]/android.widget.Lin-
  earLayout[1]/android.view.ViewGroup[1]/androidx.recyclerview.widget.RecyclerView[1]/android.widget.LinearLayout[1]/
  android.widget.RelativeLayout[1]/android.view.View[1]
19 SLEEP,2
20 Tap,RES_ID:android:id/button1
21 SLEEP,10
22 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
  ew.ViewGroup[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='
  Navigate up']
23 # Cambio immagine di profilo di B (utente remoto)
24 SLEEP,20
25 # Apertura menù laterale
26 Tap,XPATH://android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.wi-
  dget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/androidx.drawerlayout.widget.Draw-
  erLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.view.ViewG-
  roup[1]/android.widget.ImageButton[@content-desc='Open the navigation drawer']
27 # Sign-out in Briar
28 SLEEP,2
29 Tap,RES_ID:org.briarproject.briar.android:id/nav_btn_signout
30 SLEEP,5

```

Figura 4.108: File delle azioni per la modifica dell'immagine del profilo

Nelle righe 8 e 10 della Figura 4.108 si accede alla sezione delle Impostazioni di Briar.

Nella riga 12 della Figura 4.108 si esegue il Tap sull'attuale immagine del profilo. Questa azione permette di accedere alla procedura di modifica dell'immagine del profilo (Figura 2.56 della Sezione 2.8.1).

Dalla riga 14 alla riga 20 della Figura 4.108 si effettua la selezione della nuova immagine del profilo dalle immagini presenti nel dispositivo.

4.22.2 - Artefatti generati

La modifica dell'immagine del profilo da parte dell'utente locale implica l'aggiunta di un messaggio nel database dell'utente locale e nel database di ognuno dei suoi contatti.

Il campo RAW di questo messaggio è composto dai seguenti dati tra loro concatenati:

- Il GROUPID associato al messaggio
- Il timestamp di invio/ricezione del messaggio
- Il BDF che contiene il descrittore dell'immagine

- L'immagine del profilo espressa come sequenza di esadecimali

Se l'utente locale cambia la propria immagine del profilo tale messaggio è associato al gruppo relativo al CLIENTID org.briarproject.briar.avatar che fa riferimento all'utente locale. Se, invece è l'utente remoto che cambia la propria immagine del profilo il messaggio in questione è associato al gruppo relativo al CLIENTID org.briarproject.briar.avatar che fa riferimento all'utente remoto.

In entrambi i casi il messaggio (Figura 4.109 e Figura 4.110) ha i seguenti metadati:

- contentType: contiene un BDF in cui è specificato il formato dell'immagine
- descriptorLength: contiene un BDF in cui è specificata la lunghezza del descrittore dell'immagine (il tipo di messaggio, la versione del formato del messaggio e il formato del messaggio). In questo caso il descrittore dell'immagine è 6021002100410a696d6167652f6a70656780 e quindi la lunghezza è 18 (12 in esadecimale).
- version: contiene un BDF con la versione del formato del messaggio

	GROUPID	MESSAGEID	MessageMeta
1	3bc520989ec0db980f0b0eea07412565833407...	23be973c1f932aeebe92c2f4070625eb06f0a5c...	contentType=410a696d6167652f6a706567 descriptorLength=2112 version=2100

Figura 4.109: Dati relativi a un messaggio di cambio immagine del profilo dell'utente locale e metadati ad esso associati

	GROUPID	MESSAGEID	MessageMeta
1	b73a82f4899311449db283ca5da9482d7a277b...	066c13b46c5a3c524da07b139992a00e6d946c...	contentType=410a696d6167652f6a706567 descriptorLength=2112 version=2100

Figura 4.110: Dati relativi a un messaggio di cambio immagine del profilo dell'utente remoto e metadati ad esso associati

Le immagini di profilo hanno la stessa gestione delle immagini vista nella Sezione 4.6.2.2.1.

4.22.3 - Query

È possibile recuperare tutti messaggi di cambio immagine profilo e le relative immagini di profilo eseguendo la seguente query.

```
SELECT g.CLIENTID, g.GROUPID,
CASE WHEN gm.METAKEY IS null THEN "Utente locale" ELSE gm.METAKEY || "==" ||
gm.VALUE END AS GroupMeta,
m.MESSAGEID, TIMESTAMP, message_meta.MessageMeta,
RAW
FROM GROUPS g LEFT JOIN GROUPMETADATA gm USING(GROUPID)
JOIN MESSAGES m USING (GROUPID)
JOIN (
SELECT m.MESSAGEID,
GROUP_CONCAT(METAKEY || '=' || VALUE, CHAR(10)) AS MessageMeta
FROM MESSAGES m JOIN MESSAGEMETADATA mm USING(MESSAGEID)
GROUP BY mm.MESSAGEID
) AS message_meta ON m.MESSAGEID = message_meta.MESSAGEID
WHERE g.CLIENTID = "org.briarproject.briar.avatar"
ORDER BY TIMESTAMP
```

È necessario recuperare anche il campo RAW per poter recuperare la sequenza esadecimale che rappresenta l'immagine di profilo vera e propria.

4.23 - Security

4.23.1 - Esperimenti

L'unico altro aspetto che si è ritenuto forensicamente interessante nella sezione delle Impostazioni è il panic button.

Numero esperimento	Descrizione	Sotto-esperimenti
46	Gestione panic button	<ol style="list-style-type: none"> 1. A configura il panic button per effettuare il sign out 2. A configura il panic button per effettuare la cancellazione dell'account

Tabella 4.24: Lista esperimenti configurazione panic button

Di seguito si riportano le istruzioni che permettono di configurare il panic button per la cancellazione dell'account.

```

1 # Configurazione del panic button per cancellazione account
2 Tap, XPATH: //android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[2]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/androidx.recyclerview.widget.RecyclerView[1]/android.widget.LinearLayout[4]/android.widget.RelativeLayout[1]/android.widget.TextView[@resource-id='android:id/title']
3 SLEEP,2
4 Tap, XPATH: //android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[2]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/androidx.recyclerview.widget.RecyclerView[1]/android.widget.LinearLayout[4]/android.widget.RelativeLayout[1]/android.widget.TextView[@resource-id='android:id/title' and @text='Panic button setup']
5 SLEEP,2
6 Tap, XPATH: //android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[2]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[2]/android.widget.RelativeLayout[1]/android.widget.TextView[@resource-id='android:id/title' and @text='Panic Button App']
7 SLEEP,2
8 Tap, XPATH: //android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.ListView[1]/android.widget.CheckedTextView[@resource-id='android:id/text1' and @text='Ripple']
9 SLEEP,2
10 Tap, XPATH: //android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[2]/android.widget.FrameLayout[1]/android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/androidx.recyclerview.widget.RecyclerView[1]/android.widget.LinearLayout[3]/android.widget.LinearLayout[1]/android.widget.Switch[@resource-id='org.briarproject.briar.android:id/switchWidget']
11 SLEEP,2
12 Tap, XPATH: //android.widget.FrameLayout[1]/android.widget.LinearLayout[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.FrameLayout[1]/android.view.ViewGroup[1]/android.widget.ImageButton[@content-desc='Navigate up']

```

Figura 4.111: Parte del file delle azioni per la configurazione del panic button per la cancellazione dell'account

Nella riga 2 della Figura 4.111 si accede alla sezione 'Security' delle Impostazioni di Briar (Figura 2.59 della Sezione 2.8.4).

Nella riga 4 della Figura 4.111 si accede alla sezione di configurazione del panic button (Figura 2.60 della Sezione 2.8.4.1).

Nelle righe 6 e 8 della Figura 4.111 si setta l'applicazione che si occupa di gestire il panic button.

Nella riga 10 della Figura 4.111 si abilita la cancellazione dell'account all'utilizzo del panic button.

4.23.2 - Artefatti generati

L'operazione di settaggio e configurazione del panic button non genera alcun dato all'interno del database ma aggiunge diverse voci nel file `org.briarproject.briar.android_preferences.xml` nella cartella `shared_prefs`.

Più nello specifico nel caso di configurazione del panic button per effettuare il sign out, nel file xml sono aggiunti:

- Un elemento con chiave `pref_key_lock` e valore `True`.
Questo elemento, se ha valore `True`, definisce che il panic button è configurato per effettuare il sign out
- Un elemento con chiave `pref_key_panic_app` e valore `info.guardianproject.ripple`.
Questo elemento definisce se è stata configurata un'applicazione per la gestione del panic button. In Briar è possibile utilizzare solo l'applicazione Ripple per gestire il panic button quindi se la voce `pref_key_panic_app` è settata ha valore `info.guardianproject.ripple`.
- Un elemento con chiave `panicResponderTriggerPackageName` e valore `info.guardianproject.ripple`. Questo elemento definisce quale applicazione scatena l'evento all'utilizzo del panic button cioè l'applicazione Ripple
- Un elemento con chiave `pref_key_purge`.
Questo elemento ha valore `False` se il panic button è configurato solo per effettuare il sign out, mentre ha valore `True` se il panic button è configurato per cancellare l'account Briar configurato presente nel dispositivo.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <boolean name="pref_key_lock" value="true" />
  <string name="pref_key_panic_app">info.guardianproject.ripple</string>
  <string name="panicResponderTriggerPackageName">info.guardianproject.ripple</string>
  <boolean name="pref_key_purge" value="false" />
</map>
```

Figura 4.112: File `org.briarproject.briar.android_preferences.xml` con configurazione panic button per sign out

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <boolean name="pref_key_lock" value="true" />
  <string name="pref_key_panic_app">info.guardianproject.ripple</string>
  <string name="panicResponderTriggerPackageName">info.guardianproject.ripple</string>
  <boolean name="pref_key_purge" value="true" />
</map>
```

Figura 4.113: File `org.briarproject.briar.android_preferences.xml` con configurazione panic button per cancellazione account

Le rimanenti funzionalità presenti nel menù Impostazioni dell'app non generano artefatti con valore forense e quindi non saranno trattati.

Capitolo 5

Conclusioni

Questo lavoro di tesi si è concentrato sullo studio di Briar per dispositivi Android ponendosi come obiettivo quello di delineare la correlazione tra le azioni eseguite dall'utente e gli artefatti generati da tali azioni.

Come è stato descritto nei capitoli precedenti, tale correlazione è stata individuata per ogni funzionalità dell'applicazione, fornendo informazioni fondamentali per gli analisti forensi che si troveranno a dover ricostruire le azioni eseguite in Briar da un indagato partendo dagli artefatti presenti sul dispositivo oggetto d'analisi.

All'inizio della mia attività sono state affrontate diverse problematiche riconducibili in primis alla necessità di utilizzare uno strumento di esecuzione di esperimenti (AnForA) appositamente pensato per le applicazioni mobile client-server su un'applicazione peer-to-peer quale è Briar. In secondo luogo, all'esigenza di effettuare tali esperimenti su emulatori invece che su dispositivi reali.

Superate queste difficoltà iniziali, i risultati ottenuti dai vari esperimenti dimostrano una forte relazione tra le azioni eseguite nell'applicazione dall'utente e i dati presenti nel database. Infatti, in quasi ogni funzionalità analizzata le azioni effettuate dall'utente comportano un'aggiunta o una modifica o una cancellazione di dati all'interno del database dell'applicazione.

Vista l'importanza assoluta dei dati presenti nel database, è risultato fondamentale lo studio del processo di decrittazione della chiave del database salvata in un file all'interno della cartella dell'applicazione.

Il lavoro di analisi svolto in questa trattazione apre la strada a futuri approfondimenti che potrebbero estendere i risultati ottenuti. In particolare, questo studio può essere considerato il punto di partenza per esplorare alcune componenti del progetto Briar esterne all'applicazione principale quali Briar MailBox e Briar Desktop.

Al momento della stesura di questo elaborato, la versione di Briar è la 1.5.11. Tuttavia, vi è un costante lavoro da parte del team di sviluppo per aggiungere nuove funzionalità e migliorare quelle già presenti. Pertanto, sarà fondamentale aggiornare periodicamente i risultati ottenuti in questa dissertazione, così da offrire un elaborato sempre in linea con le evoluzioni dell'applicazione agli analisti forensi che avranno necessità di eseguire l'esame forense di un dispositivo in cui è presente Briar.

Bibliografia

- [1] Wiki Briar – BQP, URL: <https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BQP.md>
- [2] Wiki Briar – BHP, URL: <https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BHP.md>
- [3] Wiki Briar – BRP, URL: <https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BRP.md>
- [4] Wiki Briar – BTP, URL: <https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BTP.md>
- [5] Wikipedia – Blake2b, URL - [https://en.wikipedia.org/wiki/BLAKE_\(hash_function\)](https://en.wikipedia.org/wiki/BLAKE_(hash_function))
- [6] Wiki Briar – BSP, URL: <https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BSP.md>
- [7] Wiki Briar - BDF, URL: <https://code.briarproject.org/briar/briar-spec/blob/master/BDF.md>
- [8] Wiki Briar – Transport Key Agreement Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Transport-Key-Agreement-Client.md>
- [9] Wiki Briar – Transport Properties Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Transport-Properties-Client.md>
- [10] Wiki Briar – Messaging Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Messaging-Client.md>
- [11] Wiki Briar – Forum Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Forum-Client.md>
- [12] Wiki Briar – Forum Sharing Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Forum-Sharing-Client.md>
- [13] Wiki Briar – Blog Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Blog-Client.md>
- [14] Wiki Briar – Blog Sharing Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Blog-Sharing-Client.md>
- [15] Wiki Briar – Private Group Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Private-Group-Client.md>
- [16] Wiki Briar – Private Group Sharing Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Private-Group-Sharing-Client.md>
- [17] Wiki Briar – Introduction Client, URL - <https://code.briarproject.org/briar/briar-spec/blob/master/clients/Introduction-Client.md>
- [18] Practical Cryptography for Developers – Scrypt, URL - <https://cryptobook.nakov.com/mac-and-key-derivation/scrypt>
- [19] Wikipedia – HMAC, URL - <https://en.wikipedia.org/wiki/HMAC>

- [20] Microsoft – HMACSHA256 Class, URL - <https://learn.microsoft.com/it-it/dotnet/api/system.security.cryptography.hmacsha256?view=net-8.0>
- [21] Wikipedia – XSalsa20, URL - https://en.wikipedia.org/wiki/Salsa20#XSalsa20_with_192-bit_nonce
- [22] Android Developers – Android Keystore system, URL - <https://developer.android.com/privacy-and-security/keystore#java>
- [23] AnForA Documentation – Automatic Experiments, URL - <https://dsdf.pages.di.unipmn.it/anfora/anfora-docs/experiments/automatic-experiments/creation.html>
- [24] Appium, URL - <https://appium.io/docs/en/latest/>
- [25] Appium Inspector, URL - <https://github.com/appium/appium-inspector>

Ringraziamenti

L'avventura è finita, e non riesco a non pensare a tutti i momenti e le persone che ne hanno fatto parte e che ora voglio ringraziare per il sostegno che mi hanno dato.

Ringrazio innanzitutto mio marito, Giulio, che per questi due anni mi ha aiutato in ogni modo possibile, interrogandomi ogni sera sugli argomenti dell'esame su cui mi stavo preparando e credendo in me più di quanto facessi io.

Ringrazio tutti i miei compagni di corso, in particolare Riccardo, Gaia e Gloria per aver alleggerito le lunghe giornate di lezione e aver reso possibile una rete di collaborazione e aiuto reciproco che non è per niente scontata.

Ringrazio i miei genitori per aver sponsorizzato l'ennesima mia folle avventura e avermi sempre sostenuto.

Ringrazio i miei amici per esserci stati e per aver fatto quelle piccole cose che contano.

Infine, mi sento di ringraziare l'università per avermi ricordato come funziona il mondo lì fuori, e i miei relatori per essere stati sempre presenti durante tutto il mio lavoro di Tirocinio e di stesura di questo elaborato, e soprattutto per avermi ricordato che l'importante non è arrivare primi, ma portare a termine un progetto di cui si va orgogliosi.

Insomma, grazie per ogni singolo momento, bello o brutto che sia stato. Questa sicuramente è stata un'avventura che non dimenticherò mai.