Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

Il sistema TextSecure: aspetti crittografici

Tesi di Laurea in Crittografia

Relatore:
Chiar.mo Prof.
Davide Aliffi

Presentata da:
Paolo Bartolucci

Sessione unica
Anno Accademico 2015/2016

Dedico questa tesi alla mia famiglia
e ai miei compagni di corso,
che mi hanno supportato e sopportato
in questi tre anni.

Introduzione

Messaggistica istantanea e sicurezza

La messaggistica istantanea (*instant messaging*) è un particolare sistema di comunicazione che permette a due parti di scambiarsi brevi messaggi o altri tipi di file **in tempo reale**, attraverso la rete internet (o rete locale).
È sicuramente una delle innovazioni nel settore tecnologico che ha contribuito maggiormente al mutamento della società nei primi anni del XXI secolo. A livello sociale la messaggistica istantanea ha eliminato qualsiasi impedimento nel comunicare, rendendolo di gran lunga più facile e veloce.
In questo contesto si è tornati a considerare strumenti crittografici come lo scambio di chiavi Diffie-Hellmann ed il sistema di ElGamal (per entrambi: vedi Glossario) poiché si prestano meglio all'implementazione della Perfect Forward Secrecy (vedi Capitolo 1) e dei meccanismi di ratcheting (vedi Glossario).
Occorre però prestare molta attenzione al modo in cui i dati viaggiano attraverso la rete internet: molte applicazioni di messaggistica infatti cifrano i messaggi impedendo ad un attaccante esterno di leggerli, ma possono accedere a tutto il materiale scambiato da tutti i loro utenti. Questo crea la necessità di avere un sistema di crittografia end-to-end (ovvero solo i due utenti possono vedere i messaggi, il server no), che abbia tra le sue caratteristiche la Deniability (vedi in dettaglio il capitolo 1). La Deniability, in un sistema di messaggistica, è la possibilità di negare che una qualsiasi conversazione sia avvenuta, o che un particolare messaggio sia stato effettivamente

scritto da una certa persona. Questo rende una conversazione tra due utenti molto più simile ad una conversazione faccia a faccia, dal punto di vista della privacy, e rende più difficile ad applicazioni o governi il controllo su dati e conversazioni degli utenti.

Indice

Elenco delle figure

Capitolo 1

Obiettivi e protocolli precedenti

1.1 Caratteristiche importanti

Per garantire la sicurezza di un sistema di messaggistica, le caratteristiche che si vogliono ottenere sono principalmente tre: Future Secrecy, Perfect Forward Secrecy e Deniability.

1.1.1 Future Secrecy

La Future Secrecy serve a garantire che, se in una conversazione tra A e B, ad un certo punto dovessero trapelare delle chiavi segrete temporanee di sessione, il protocollo possa fare in modo che, da un certo punto in poi, i messaggi successivi tornino privati e sicuri.
È da notare che la Future Secrecy non è possibile se la chiave di lunga durata di una delle due parti trapela; infatti in questo caso l'attaccante potrebbe comportarsi da "man-in-the-middle" (cioè far credere ad A di essere B e a B di essere A), in modo da ricevere e poi inoltrare, alterati o meno, i messaggi di entrambe le parti. Per questo motivo una condizione essenziale per la Future Secrecy è che le chiavi di lunga durata rimangano segrete. In questo caso, la Future Secrecy sarà rinforzata ad ogni "ratchet" (vedi Glossario): dato che le chiavi a lunga durata sono uno degli input nella funzione di derivazione delle

chiavi, anche se vengono violate delle chiavi temporanee, nessun attaccante riuscirà a ricavare quelle successive e quindi a leggere i messaggi futuri.

1.1.2 Perfect Forward Secrecy

La Perfect Forward Secrecy è una caratteristica ricercata nei sistemi crittografici che garantisce che se una delle parti che comunicano venisse forzata a rivelare la sua chiave segreta, tutti i messaggi inviati prima di questo evento rimarrebbero sicuri, cioè non potrebbero essere decifrati con la chiave rivelata.

1.1.3 Deniability

La Deniability, che è uno degli obiettivi principali nel protocollo OTR (uno dei protocolli precedenti a TextSecure che andremo ad analizzare), può essere descritta come segue: sia il mittente che il destinatario dovrebbero essere in grado di negare l'aver inviato o generato un messaggio, mostrando che ogni altro partecipante alla conversazione potrebbe aver mandato e generato quel particolare messaggio. OTR raggiunge questo obiettivo usando uno schema a cifratura malleabile (vedi Glossario) e pubblicando periodicamente la chiave MAC (Message Autentication Code) usata per l'autenticazione, subito dopo che il destinatario ha verificato l'autenticità del messaggio ricevuto. Quindi, chiunque potrebbe aver modificato i messaggi in chiaro e prodotto un MAC valido, così che l'origine del messaggio non sia più certa.

1.2 Protocollo Off-the-Record

Uno dei protocolli usati per rendere sicuro un sistema di messaggistica istantanea è Off-the-Record (OTR).

L'obiettivo di OTR differisce significativamente dai precedenti meccanismi di protezione per sistemi di messaggistica introducendo due nuove pro-

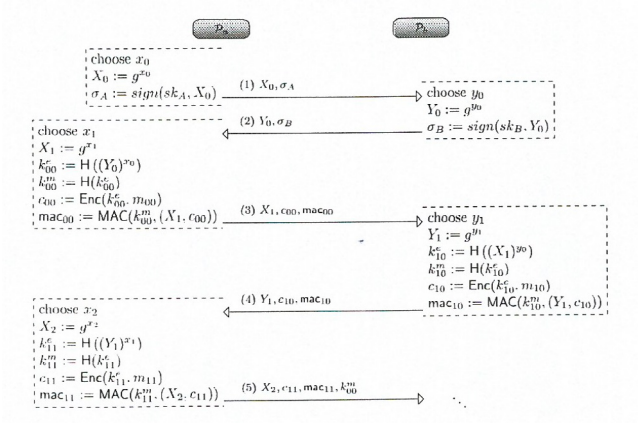

Figura 1.1: Il protocollo Off-The-Record

prietà: **Perfect Forward Secrecy** e **Deniability**.
Dopo un iniziale scambio di chiavi di Diffie-Hellman (DH), con messaggi firmati digitalmente, ad ogni messaggio viene effettuato un nuovo scambio DH (non più firmato) e la risultante chiave DH viene cambiata costantemente.
OTR usa una cifratura malleabile combinata con dei MAC, invece della firma digitale (negli scambi successivi al primo).
Il protocollo OTR rende pubblica ogni chiave MAC nel round successivo a quello in cui è stata usata; questo è essenziale per dare al protocollo la proprietà di *Deniability*: chiunque può modificare il messaggio in chiaro, dato che invertendo i bit del testo cifrato si provoca un'inversione degli stessi bit, nella stessa posizione, nel testo in chiaro. Per questo i messaggi ricevuti sono autentici solo nel momento in cui vengono ricevuti (verificando la prima firma e i MAC successivi).
Dato che le chiavi MAC sono derivate dalle chiavi di cifratura attraverso una funzione hash, unidirezionale, rivelarle non compromette la sicurezza del sistema e i messaggi scambiati rimangono confidenziali. Gli scambi DH privati x_i e y_j vengono eliminati non appena la chiave $k_{ij} = g^{x_i * y_j}$ è stata calcolata. Questo garantisce la *Perfect Forward Secrecy* perchè, senza gli scambi privati, la chiave di cifratura non può essere ricavata in seguito dagli scambi pubblici $X_i = g_i^x$ e $Y_j = g_j^y$.

1.2.1 OTR applicato alla messaggistica

Le connessioni nella Messaggistica Istantanea solitamente sono di breve durata e online, mentre in altri sistemi per lo scambio di messaggi di testo queste possono durare per un tempo prolungato e gli interlocutori potrebbero essere offline temporaneamente. Inoltre, queste conversazioni potrebbero essere asincrone, cioè una delle due parti potrebbe mandare più messaggi prima di ricevere una risposta.
Il primo adattamento finalizzato a derivare un protocollo sicuro per messaggi di testo a partire da OTR è quello di rendere OTR funzionale in scenari offline. L'idea di base è presa dal protocollo di ElGamal: OTR può essere adattato a scenari offline memorizzando preventivamente un certo numero di scambi DH di chiavi temporanee per ognuna delle due parti, su un server.
Il secondo adattamento riguarda la gestione delle chiavi: in OTR, uno scambio DH di chiavi temporanee deve venire protetto da un MAC, calcolato con una chiave precedente, e deve essere ricevuto dal destinatario B prima di poter essere usato dal mittente A. Questo concatenamento di chiavi attraverso MAC ha bisogno di molta manutenzione e di sincronizzazione, come anche ne ha bisogno lo scambio DH.
TextSecure, al contrario, si adatta a questo scenario rimpiazzando la concatenazione di MAC con un valore segreto, derivato sia dallo scambio DH di lunga durata (g^a, g^b) che da quello effimero (g^{x_a}, g^{x_b}), aggiornato ad ogni passo della generazione di chiavi.

1.3 Protocollo "Silent Circle Instant Messaging"

In Silent Circle Instant Messaging Protocol (SCIMP) l'idea essenziale è quella di usare diverse generazioni di chiavi simmetriche: dopo aver mandato un messaggio, il mittente aggiorna la sua stessa chiave rimpiazzandola con il suo valore *hash*: $k_{new} := hash(k_{old})$. Facendo così, può essere raggiunta la *Perfect Forward Secrecy* tra due scambi di chiavi DH: anche se il mittente fosse costretto a rivelare le sue chiavi, un avversario non potrebbe più decifrare i messaggi già inviati, essendo state le chiavi precedenti cancellate. Anche in questo caso i due utenti che scambiano messaggi devono mantenersi sincronizzati.

Capitolo 2

Protocollo TextSecure

2.1 Il protocollo TextSecure

2.1.1 Introduzione

TextSecure è costruito a partire da un protocollo per lo scambio di chiavi a un solo round (ORKE[1]), eseguito tra le parti A e B per calcolare un segreto a lungo termine, una funzione per la derivazione delle chiavi (KDF[2]) che prende in input il segreto a lungo termine e un nuovo segreto DH, e per finire uno schema di cifratura autenticato.

Per il primo messaggio di una conversazione, una chiave effimera e segreta del destinatario (chiamata *prekey*) viene ottenuta dal server di TextSecure, insieme alla chiave pubblica a lungo termine. In seguito, nuovi scambi pubblici DH (effimeri) sono inclusi in ogni prima risposta ad un messaggio. Nella terminologia di OTR e di TextSecure, lo scambio di chiavi effimere DH è chiamato *ratcheting*[3].
Se una parte manda diversi messaggi prima di ricevere una risposta, aggiorna la chiave simmetrica usata per ogni messaggio applicando una funzione di

[1]One Round Key Exchange
[2]Key Derivation Function
[3]L'idea è quella di un meccanismo di aggiornamento mediante "scatti" successivi

derivazione delle chiavi ad un valore intermedio (chiamato "*chaining key*"), dal quale sono calcolate tutte le chiavi crittografiche del sistema. In figura 2.1 è riportato lo schema generale di comunicazione di TextSecure:

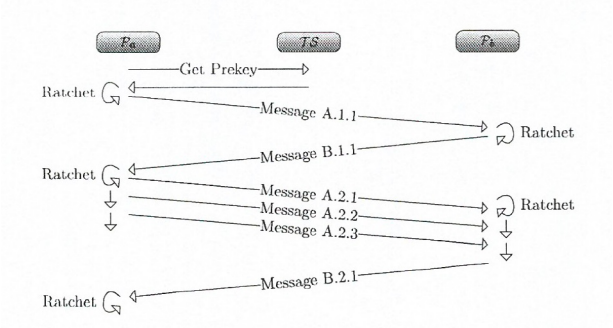

Figura 2.1: Struttura generale di comunicazione su TextSecure

2.1.2 Il protocollo

Gli scambi DH vengono eseguiti sulla curva ellittica Curve25519, essendo questa già implementata nella libreria crittografica Android di Google.
TextSecure usa crittografia simmetrica data da AES (Advanced Encryption Standard) con chiavi a 256 bit. Per l'integrità dei messaggi è usata la funzione HMAC-SHA256 (keyed-Hash Message Autentication Code - Secure Hash Algorithm a 256 bit).
Gli scambi di messaggi tra l'applicazione e il server di TextSecure (TS) utilizzano TLS[4]; il certificato del server è autofirmato ed è incluso nel codice dell'applicazione. La consegna effettiva dei messaggii avveniva in origine tramite il Google Cloud Messaging, ma dopo che il protocollo è stato incluso in WhatsApp si può presumere che venga usata la rete di distribuzione di messaggi di questa applicazione.

[4]Transport Layer Security

2.1.3 Le fasi del protocollo

Il protocollo TextSecure è diviso in varie fasi come vediamo dalla Figura 2.2

Distinguiamo: (1) Registrazione, (2) Confronto e autenticazione recipro-

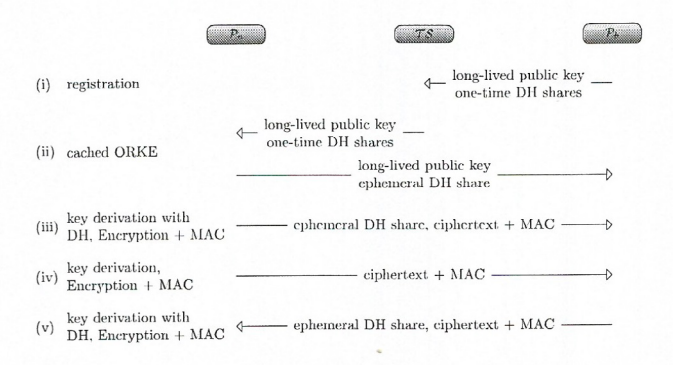

Figura 2.2: Struttura generale delle fasi in TextSecure

ca delle chiavi, (3) Invio/Ricezione del primo messaggio, (4) Invio di un messaggio successivo, (5) Invio di una risposta.

2.1.4 Registrazione

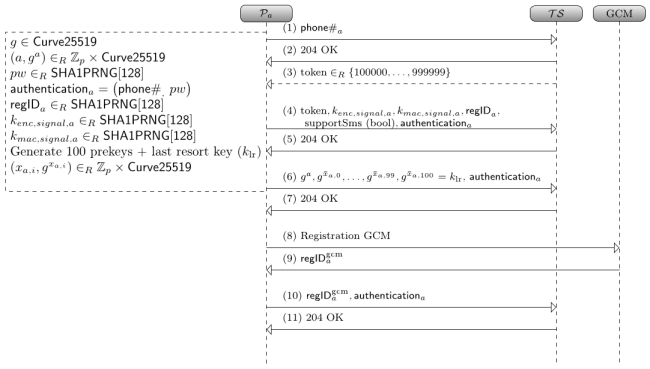

Figura 2.3: Registrazione su TextSecure

Per registrarsi con il server TS di TextSecure, una parte P_a richiede un token (simbolo) di verifica trasmettendo il suo numero di telefono ($phone\#_a$)

e il suo medoto di trasporto preferito al TS (Step 1), che quest'ultimo conferma con un messaggio HTTP 204 (Step 2).
A seconda del metodo di trasporto scelto da P_a, il TS o invia un breve messaggio o effettua una chiamata vocale contenente un token random (Step 3) al numero trasmesso nello Step 1.
L'autenticazione di P_a avviene perciò attraverso uno scambio "challenge-response". P_a mette in atto la vera registrazione nello Step 4 in cui dimostra di possedere il numero $phone\#_a$ includendo il token, registra le sue credenziali con il server tramite l'autenticazione fornita attraverso HTTP e stabilisce le sue "*signaling keys*", chiavi che verranno usate in seguito dal TS per cifrare i messaggi che verranno trasmessi tramite Google Cloud Messaging (GCM), così che il GCM non sia in grado di leggerli: $k_{mac,TS,A}$ e $k_{enc,TS,A}$. Il server accetta la registrazione solo se il token corrisponde a quello fornito nello Step 3 e il numero di telefono non è ancora registrato (Step 5).
Nello Step 6 P_a fornisce le sue 100 prekeys e k_{lr} (last resort key) a TS; il server accetta se il messaggio è corretto e se l'autenticazione HTTP avviene con successo (Step 7).
In seguito P_a si registra con il GCM nello Step 8 e riceve il suo $regID_a^{gcm}$ (identificatore di avvenuta registrazione di A con il GCM), da trasmettere al TS nello Step 10 dopo una nuova autenticazione.
Una prekey per P_a è necessaria ogni volta che inizia una nuova sessione (con qualsiasi persona) nella quale P_a è quello che risponde (quindi non scrive per primo), perciò il numero di prekeys sul server diminuisce. Per questo, quando rimangono poche prekeys sul server, P_a può depositarne di nuove. Quando non ci sono più prekeys disponibili, viene usata k_{lr}. Quest'ultima però, a differenza delle altre, non viene cancellata dal server dopo l'uso.

2.1.5 Confronto e autenticazione reciproca delle chiavi

Per provare a stabilire l'effettiva appartenenza di una data chiave pubblica a una certa parte, TextSecure offre la possibilità di visualizzare l'impronta digitale della chiave pubblica a lungo termine di un utente. Questo è molto

utile per evitare attacchi del tipo "Man In The Middle". Due parti quindi possono confrontare le impronte usando un canale esterno, come una chiamata telefonica o un incontro di persona. Se due parti si incontrano di persona, TextSecure offre la possibilità di rendere l'impronta della propria chiave pubblica a lungo termine come codice QR usando un'altra applicazione, che è la stessa che l'altra parte usa per confrontare le impronte.

2.1.6 Invio/Ricezione del primo messaggio

Invio del primo messaggio

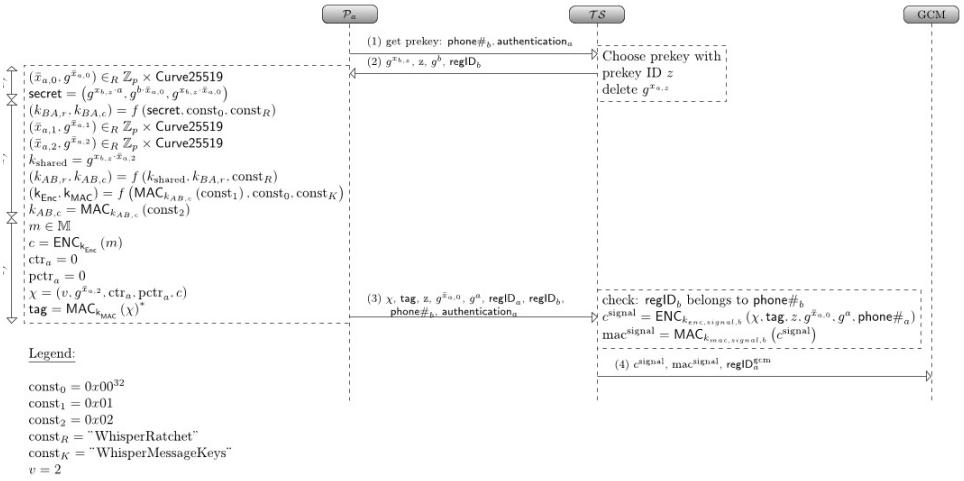

Figura 2.4: Invio del primo messaggio con TextSecure

Prima che il primo messaggio possa essere inviato da P_a a P_b devono essere completati 3 passi principali:

(a) Un protocollo di scambio di chiavi, per stabilire un segreto condiviso, simmetrico e a lungo termine, rk_{BA} (chiamata anche root key, chiave principale)

(b) Una derivazione di chiave e l'aggiornamento del protocollo (chiamato *Axolotl-ratchet*), che aggiorna la root key e genera una chaining key (valore dal quale si derivano altre chiavi) dalla quale sono derivate la chiave di cifratura e la chiave MAC.

(c) Uno schema di cifratura autenticata.

(a) Stabilire un segreto condiviso. Nel primo passo, P_a richiede a TS una prekey per P_b e riceve un identificatore della prekey z, una prekey $g^{x_{b,z}}$ e la chiave a lungo termine di P_b, g^b. P_a riceve anche un identificatore di registrazione di P_b. Sceglie poi una nuova chiave temporanea $\bar{x}_{a,0}$ per calcolare un segreto intermedio sec_{int} che deriva dalla concatenazione di tre operazioni DH, combinando la prekey di P_b, la chiave pubblica a lungo termine di P_a, la chiave pubblica a lungo termine di P_b e la chiave temporanea appena scelta da P_a: $sec_{int} = (g^{x_{b,z}*a}, g^{b*\bar{x}_{a,0}}, g^{x_{b,z}*\bar{x}_{a,0}})$.
Da questo valore viene derivata la prima root key rk_{BA} tramite una funzione di derivazione chiavi (chiamata HKDF, Hash-Based Key Derivation Function).

(b) Gestione delle chiavi (Axolotl-ratchet). Dopo il calcolo del segreto condiviso da parte di P_a, viene derivato un nuovo segreto k_{shared} a partire dalla prekey ricevuta e viene eseguito uno scambio DH con una chiave effimera $\bar{x}_{a,2}$: $k_{shared} := (g^{x_{b,z}})^{\bar{x}_{a,2}}$. Da k_{shared} e rk_{BA} vengono derivate (con la stessa funzione di derivazione chiavi usata per rk_{BA}) una nuova root key rk_{AB} e una chaining key ck_{AB}.
La chaining key viene infine usata per derivare (sempre con la stessa funzione HKDF) la chiave di cifratura (K_{Enc}) e la chiave MAC (K_{MAC}) per cifrare e proteggere l'integrità del primo messaggio.

(c) Cifratura autenticata. Un messaggio $m \in \mathbb{N}$ viene cifrato usando AES: $c = ENC_{K_{Enc}}(m)$. In seguito P_a calcola $tag = MAC_{K_{MAC}}(\chi)$, dove $\chi = (v, g^{\bar{x}_{a,2}}, ctr_a, pctr_a, c)$. v rappresenta la versione del protocollo usata, *ctr* e *pctr* invece sono usati per ordinare i messaggi all'interno della conversazione: entrambi posti inizialmente 0, ctr viene incrementato ad ogni messaggio inviato da una parte, pctr ha il valore che ctr aveva nel messaggio a cui si sta rispondendo. A questo punto il messaggio (3) (vedi figura 2.4) viene inviato al TS. È chiaro che ctr e pctr permettono di ordinare i messaggi (altrimenti

potrebbero arrivare in ordine casuale).

Inoltrare il messaggio al GCM

Subito dopo aver ricevuto il messaggio, il TS controlla se $regID_b$ corrisponde al numero $phone\#_b$. Esso poi cifra le parti del messaggio destinate a P_b con la signaling key di quest'ultimo ($k_{enc,TS,B}$), usando AES (denotiamo il messaggio cifrato con c^{signal}).
Il TS inoltre calcola un MAC del risultato, che denotiamo con mac^{signal} e manda questo e c^{signal} al server del GCM, insieme al $regID_b^{gcm}$ come ricevente. Il risultato di questa cifratura aggiuntiva è che il Cloud Messaging di Google sarà solo in grado di vedere il ricevente del messaggio ma non il mittente (caratteristica importante per la Deniability). TS invece conosce l'identità di entrambi.

Ricezione del primo messaggio

Appena P_b ha ricevuto il messaggio, verifica mac^{signal} e, se la verifica ha successo, decifra c^{signal}. Prende la sua chiave privata a lungo termine, la chiave privata "usa e getta" che corrisponde all'identificatore della prekey, z, e calcola sec_{int} e la prima root key rk_{BA}. Da questo punto in poi ogni azione è identica a quelle compiute dal mittente, tenendo presente che P_b ha anche ricevuto $\bar{x}_{a,z}$, fino a quando vengono derivate le chiavi K_{Enc} e K_{MAC}. P_b ora verifica il MAC e, se la verifica ha successo, decifra il messaggio.

2.1.7 Invio di un messaggio successivo

Se P_a vuole mandare un altro messaggio **prima** che P_b abbia risposto al primo, per prima cosa deriva una nuova chaining key $ck_{AB} := MAC_{ck_{AB}^{old}}(const_2)$
e deriva un nuovo paio $(K_{Enc}, K_{MAC}) = HKDF(MAC_{ck_{AB}}(const_1), const_0, const_K)$,
dove $const_0$, $const_1$ e $const_2$ sono costanti fissate, mentre $const_K$ è una costante derivata dalla chiave usata per il messaggio precedente. Queste nuove chiavi vengono usate per inviare il nuovo messaggio e il procedimento viene

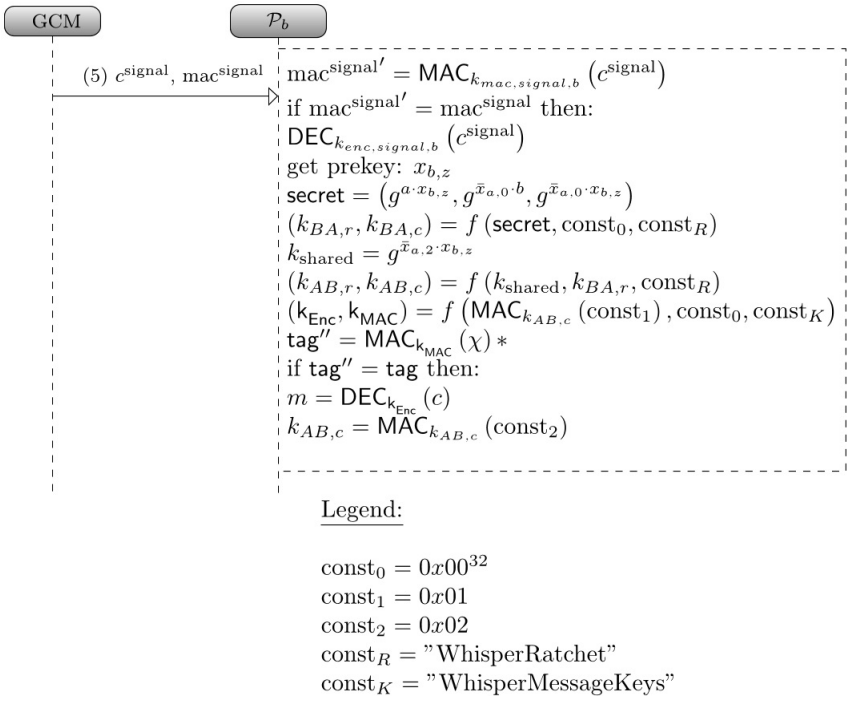

Figura 2.5: Ricezione del primo messaggio con TextSecure

ripetuto per tutti i messaggi successivi, fino a quando P_a riceve una risposta da P_b; questa contiene il materiale necessario per un nuovo scambio DH e, in questo modo, il ratcheting può eseguire uno "scatto" in avanti.

2.1.8 Invio di una risposta

Se P_b vuole inviare una risposta in una sessione già esistente con P_a, come prima cosa sceglie un nuovo paio di chiavi temporanee $(\bar{x}_{b,0}, g^{\bar{x}_{b,0}})$ e calcola una nuova k_{shared} come output di un'operazione DH che prende l'ultima chiave pubblica temporanea di P_a, $g^{\bar{x}_{a,2}}$ e la propria chiave temporanea privata appena scelta $\bar{x}_{b,0}$ come input: $k_{shared} = g^{\bar{x}_{a,2} * \bar{x}_{b,0}}$.
In seguito P_b aggiorna la sua root key e deriva una nuova chaining key calcolando HKDF della coppia formata dal nuovo k_{shared} e dalla root key "vecchia" rk_{AB}: $(rk_{BA}, ck_{BA}) = HKDF(k_{shared}, rk_{AB}, const_R)$, dove $const_R$ è una costante di ratchet. Dalla chaining key viene derivata una nuova coppia

"chiave di cifratura - chiave MAC", con le quali si protegge il messaggio, e infine vengono mandati il messaggio cifrato e $g^{\bar{x}_{b,0}}$ a P_a.

Conclusioni

Il protocollo TextSecure ha introdotto novità importanti nell'implementazione della crittografia end-to-end, tant'è vero che ha avuto una collaborazione (annunciata nel novembre 2014 e completata non molto tempo fa) con l'applicazione di messaggistica più popolare al mondo: WhatsApp. Più in dettaglio, il protocollo TextSecure è stato incorporato in quello di WhatsApp per procurare a tutti gli utilizzatori di quest'ultimo la crittografia end-to-end di default.
Oltre a questo tipo di crittografia, il protocollo TextSecure ha raggiunto anche altri obiettivi riguardanti la sicurezza.
La Future Secrecy, come descritto nella definizione del capitolo 1, dipende dalla quantità e dal tipo di dati che riesce ad ottenere un attaccante, ma nel caso in cui le chiavi di lunga durata rimangano segrete, questa proprietà viene raggiunta e rinforzata ad ogni ratchet, essendo tali chiavi parte degli input della funzione di derivazione per le chiavi da usare nei messaggi successivi.
Anche la Perfect Forward Secrecy dipende dal tipo di chiavi che ottiene l'attaccante, ma in questo caso un aspetto da non sottovalutare è il fatto che il server TS non memorizza nessuna chiave privata; per questo tali chiavi possono trapelare solo dall'App TextSecure nel dispositivo dell'utente. Inoltre la compromissione delle sole chiavi a lungo termine non compromette la sicurezza di alcun messaggio in TextSecure; questo perché esse sono solo uno dei parametri che influiscono nella derivazione delle chiavi usate per i messaggi. Se però venissero rivelate altre chiavi (di breve o media durata) la situazione si farebbe più complessa e, in alcuni casi, potrebbe venire meno la Perfect

Forward Secrecy.

Per la Deniability, infine, TextSecure cifra e autentica tutti i messaggi in modo simmetrico: assumendo che il protocollo di scambio di chiavi procuri chiavi firmate, entrambe le parti hanno la certezza che il messaggio ricevuto provenga dall'altra parte con cui si sta comunicando. Dato che entrambe le parti calcolano le stesse chiavi per cifratura e MAC, nessuno di loro può attribuire crittograficamente l'appartenenza di un messaggio all'altro. Analizzando il protocollo si può vedere che le parti possono simulare l'intero scambio di chiavi da sole, dato che la diretta interazione di un'altra parte non è necessaria, essendo le prechiavi accessibili sul server. Quindi si ha che TextSecure ha Deniability a livello di protocollo ma nella pratica si possono verificare alcune complicazioni: quando una parte manda un messaggio, deve inviarlo attraverso il server di TextSecure. Il messaggio è cifrato ma per garantire una corretta consegna al destinatario, le identità del mittente ($regID_a$) e del destinatario ($regID_b$) vengono trasmesse al server. Inoltre la richiesta di invio viene autenticata col numero di telefono del mittente ($phone\#_a$) e la sua password. Quindi a livello pratico la Deniability non è completamente raggiunta da TextSecure.

Glossario

Scambio di chiavi Diffie-Hellman: protocollo crittografico che permette a due parti di stabilire una chiave condivisa e segreta, attraverso un canale insicuro, senza che la chiave passi attraverso il canale o che le due parti si incontrino.

ElGamal: sistema di cifratura a chiave pubblica basato sulla difficoltà di calcolo del logaritmo discreto.

Axolotl Ratchet: algoritmo di gestione delle chiavi usato in alcuni protocolli crittografici per sistemi di messaggistica istantanea per ottenere una cifratura end-to-end.

Ratcheting: meccanismo di aggiornamento delle chiavi di sessione ad ogni messaggio: nel caso in cui una parte invii più di un messaggio prima di ottenere una risposta, per ognuno di questi messaggi la chiave si aggiornerà a una nuova, ottenuta direttamente dalla precedente (senza eseguire uno scambio DH).

Malleabilità: caratteristica di un sistema crittografico consistente nel permettere di modificare il testo in chiaro intervenendo sul testo cifrato, senza conoscere la chiave.

Bibliografia

[1] How Secure is TextSecure? - T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, T. Holz.
Disponibile all'indirizzo: https://eprint.iacr.org/2014/904.pdf

[2] Open Whisper System Blog.
Indirizzo: https://whispersystems.org/blog/

[3] Cryptography Theory and Practice - Douglas R. Stinson, terza edizione, editore Chapman & Hall/CRC, anno 2003.

Ringraziamenti

Ringrazio il professor Davide Aliffi per avermi fatto appassionare alla crittografia, materia che continuerò a studiare nel corso della laurea magistrale e che approfondirò negli anni a venire.