

# Protocollo Signal

Elena Tonini, Matr.727382

Università degli Studi di Brescia

A.A. 2021/2022

2022-04-25

SIGNAL

Protocollo Signal

Elena Tonini, Matr.727382

Università degli Studi di Brescia

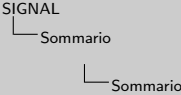
A.A. 2021/2022

# Sommario I

- 1 Sommario
- 2 Applicazione Signal
  - Storia dell'Applicazione
  - L'Applicazione e il Protocollo Signal
- 3 Crittografia End-to-End
  - Applicazioni
  - Problematiche
- 4 Signal Protocol
  - Proprietà
  - Il protocollo
  - Difetti di progettazione
  - Considerazioni
  - WhatsApp VS Signal VS Telegram
- 5 Bibliografia

- 1. Sommario
- 2. Applicazione Signal
  - Storia dell'Applicazione
  - L'Applicazione e il Protocollo Signal
- 3. Crittografia End-to-End
  - Applicazioni
  - Problematiche
- 4. Signal Protocol
  - Proprietà
  - Il protocollo
  - Difetti di progettazione
  - Considerazioni
  - WhatsApp VS Signal VS Telegram
- 5. Bibliografia

2022-04-25



- Sommario I
- 1. Sommario
- 2. Applicazione Signal
  - Storia dell'Applicazione
  - L'Applicazione e il Protocollo Signal
- 3. Crittografia End-to-End
  - Applicazioni
  - Problematiche
- 4. Signal Protocol
  - Proprietà
  - Il protocollo
  - Difetti di progettazione
  - Considerazioni
  - WhatsApp VS Signal VS Telegram
- 5. Bibliografia

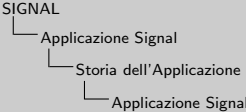
# Applicazione Signal

## Storia dell'Applicazione

L'applicazione Signal ha origine dall'unione dei due servizi di messaggistica **TextSecure** e **RedPhone**, sviluppati da **Moxie Marlinspike** e **Stuart Anderson**, che insieme fondarono la start-up **Whisper Systems** nel 2010.

Entrambe le applicazioni implementavano la crittografia end-to-end.

2022-04-25



Applicazione Signal  
Storia dell'Applicazione

L'applicazione Signal ha origine dall'unione dei due servizi di messaggistica **TextSecure** e **RedPhone**, sviluppati da **Moxie Marlinspike** e **Stuart Anderson**, che insieme fondarono la start-up **Whisper Systems** nel 2010.

Entrambe le applicazioni implementavano la crittografia end-to-end.

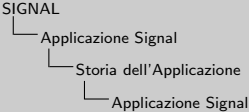
# Applicazione Signal

## Storia dell'Applicazione

L'applicazione Signal ha origine dall'unione dei due servizi di messaggistica **TextSecure** e **RedPhone**, sviluppati da **Moxie Marlinspike** e **Stuart Anderson**, che insieme fondarono la start-up **Whisper Systems** nel 2010.

Entrambe le applicazioni implementavano la crittografia end-to-end.

2022-04-25



Applicazione Signal  
Storia dell'Applicazione

L'applicazione Signal ha origine dall'unione dei due servizi di messaggistica **TextSecure** e **RedPhone**, sviluppati da **Moxie Marlinspike** e **Stuart Anderson**, che insieme fondarono la start-up **Whisper Systems** nel 2010.

Entrambe le applicazioni implementavano la crittografia end-to-end.

# Applicazione Signal

## Storia dell'Applicazione

1 Sommario

2 Applicazione Signal

3 Crittografia End-to-End

4 Signal Protocol

5 Bibliografia

A seguito del nuovo rilascio delle applicazioni nel 2011 i due servizi assumono la propria natura **open-source** che ancora oggi caratterizza l'applicazione Signal.

Nel 2013 Marlinspike fonda il progetto open-source **Open Whisper Systems**, grazie a cui rilascia la prima versione di Signal nel 2015 (anche per PC come applicazione Chrome), per poi rilasciarlo anche per Windows, Mac e Linux nel 2017.



Nel 2011 Twitter acquista Whisper Systems e Marlinspike diventa capo della cybersecurity del social media. Nel 2013 Marlinspike abbandona Twitter e fonda la OWS.

Nello stesso anno inizia a lavorare al protocollo Signal insieme al fondatore di WhatsApp Trevor Perrin.

Applicazione Signal  
Storia dell'Applicazione

A seguito del nuovo rilascio delle applicazioni nel 2011 i due servizi assumono la propria natura **open-source** che ancora oggi caratterizza l'applicazione Signal.

Nel 2013 Marlinspike fonda il progetto open-source Open Whisper Systems, grazie a cui rilascia la prima versione di Signal nel 2015 (anche per PC come applicazione Chrome), per poi rilasciarlo anche per Windows, Mac e Linux nel 2017.

# Applicazione Signal

## Storia dell'Applicazione

### 1 Sommario

### 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia



Nel 2011 Twitter acquista Whisper Systems e Marlinspike diventa capo della cybersecurity del social media. Nel 2013 Marlinspike abbandona Twitter e fonda la OWS.

Nello stesso anno inizia a lavorare al protocollo Signal insieme al fondatore di WhatsApp Trevor Perrin.

Applicazione Signal  
Storia dell'Applicazione

A seguito del nuovo rilascio delle applicazioni nel 2011 i due servizi assumono la propria natura **open-source** che ancora oggi caratterizza l'applicazione Signal.

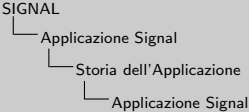
Nel 2013 Marlinspike fonda il progetto open-source **Open Whisper Systems**, grazie a cui rilascia la prima versione di Signal nel 2015 (anche per PC come applicazione Chrome), per poi rilasciarlo anche per Windows, Mac e Linux nel 2017.

# Applicazione Signal

## Storia dell'Applicazione

Nel febbraio 2018 Marlinspike e il co-fondatore di WhatsApp Brian Acton fondarono la **Signal Foundation**, il cui obiettivo è il supporto e l'accelerazione della diffusione della comunicazione privata e sicura. [Lumer]

2022-04-25



Applicazione Signal  
Storia dell'Applicazione

Nel febbraio 2018 Marlinspike e il co-fondatore di WhatsApp Brian Acton fondarono la **Signal Foundation**, il cui obiettivo è il supporto e l'accelerazione della diffusione della comunicazione privata e sicura. [Lumer]

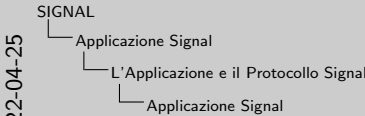
# Applicazione Signal

## L'Applicazione e il Protocollo Signal

Nel 2013, dopo la fondazione di OWS, i fondatori Marlinspike e Trevor Perrin iniziarono a lavorare al **Protocollo Signal**.

Esso rendeva il metodo crittografico end-to-end utilizzato nell'applicazione Signal implementabile anche da altri servizi.

Ogni piattaforma di messaggistica che intraprese collaborazioni con OWS al fine di integrare il protocollo Signal al proprio interno lo implementò in modalità differenti e su scala/estensione diversa.



2022-04-25

Applicazione Signal  
L'Applicazione e il Protocollo Signal

Nel 2013, dopo la fondazione di OWS, i fondatori Marlinspike e Trevor Perrin iniziarono a lavorare al **Protocollo Signal**.

Esso rendeva il metodo crittografico end-to-end utilizzato nell'applicazione Signal implementabile anche da altri servizi.

Ogni piattaforma di messaggistica che intraprese collaborazioni con OWS al fine di integrare il protocollo Signal al proprio interno lo implementò in modalità differenti e su scala/estensione diversa.

- 1 Sommario
- 2 Applicazione Signal
  - Storia dell'Applicazione
  - L'Applicazione e il Protocollo Signal
- 3 Crittografia End-to-End
  - Applicazioni
  - Problematiche
- 4 Signal Protocol
  - Proprietà
  - Il protocollo
  - Difetti di progettazione
  - Considerazioni
  - WhatsApp VS Signal
  - VS Telegram
- 5 Bibliografia



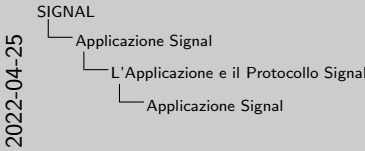
# Applicazione Signal

## L'Applicazione e il Protocollo Signal

Nel 2013, dopo la fondazione di OWS, i fondatori Marlinspike e Trevor Perrin iniziarono a lavorare al **Protocollo Signal**.

Esso rendeva il metodo crittografico end-to-end utilizzato nell'applicazione Signal implementabile anche da altri servizi.

Ogni piattaforma di messaggistica che intraprese collaborazioni con OWS al fine di integrare il protocollo Signal al proprio interno lo implementò in modalità differenti e su scala/estensione diversa.



Applicazione Signal  
L'Applicazione e il Protocollo Signal

Nel 2013, dopo la fondazione di OWS, i fondatori Marlinspike e Trevor Perrin iniziarono a lavorare al **Protocollo Signal**.

Esso rendeva il metodo crittografico end-to-end utilizzato nell'applicazione Signal implementabile anche da altri servizi.

Ogni piattaforma di messaggistica che intraprese collaborazioni con OWS al fine di integrare il protocollo Signal al proprio interno lo implementò in modalità differenti e su scala/estensione diversa.

- 1 Sommario
- 2 Applicazione Signal
  - Storia dell'Applicazione
  - L'Applicazione e il Protocollo Signal
- 3 Crittografia End-to-End
  - Applicazioni
  - Problematiche
- 4 Signal Protocol
  - Proprietà
  - Il protocollo
  - Difetti di progettazione
  - Considerazioni
  - WhatsApp VS Signal
  - VS Telegram
- 5 Bibliografia

# Applicazione Signal

## L'Applicazione e il Protocollo Signal

### 1 Sommario

### 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

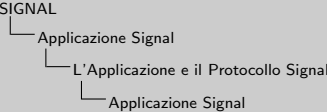
### 5 Bibliografia

Nel 2013, dopo la fondazione di OWS, i fondatori Marlinspike e Trevor Perrin iniziarono a lavorare al **Protocollo Signal**.

Esso rendeva il metodo crittografico end-to-end utilizzato nell'applicazione Signal implementabile anche da altri servizi.

Ogni piattaforma di messaggistica che intraprese collaborazioni con OWS al fine di integrare il protocollo Signal al proprio interno lo implementò in modalità differenti e su scala/estensione diversa.

2022-04-25



Applicazione Signal  
L'Applicazione e il Protocollo Signal

Nel 2013, dopo la fondazione di OWS, i fondatori Marlinspike e Trevor Perrin iniziarono a lavorare al **Protocollo Signal**.

Esso rendeva il metodo crittografico end-to-end utilizzato nell'applicazione Signal implementabile anche da altri servizi.

Ogni piattaforma di messaggistica che intraprese collaborazioni con OWS al fine di integrare il protocollo Signal al proprio interno lo implementò in modalità differenti e su scala/estensione diversa.

# Applicazione Signal

## L'Applicazione e il Protocollo Signal

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione

L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

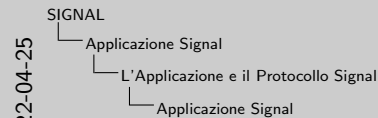
Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- **Duo:** protezione delle videochat
- **Skype:** conversazioni private dal 2018
- **WhatsApp:** tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]



- Facebook: usa Signal solo nelle Secret Conversations
- Allo: applicazione mobile di messaggistica istantanea di Google, non esiste più dal 12 marzo 2019
- Duo: applicazione per videochiamate e chat mobile di Google
- Whatsapp: introdusse Signal per la prima volta nel 2014 per utenti Android, estendendolo a tutti gli utenti nel 2016
- Google: introduce Signal di default nell'applicazione di messaggi su Android

Applicazione Signal  
L'Applicazione e il Protocollo Signal

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- Facebook: introdusse la feature Secret Conversations per gli utenti di Facebook Messenger nel luglio 2016
  - Allo: rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
  - Duo: protezione delle videochat
  - Skype: conversazioni private dal 2018
  - WhatsApp: tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)
- [Gre29], [Lumer]

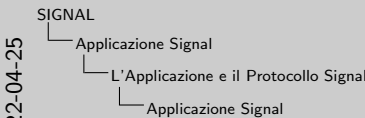
# Applicazione Signal

## L'Applicazione e il Protocollo Signal

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- **Duo:** protezione delle videochat
- **Skype:** conversazioni private dal 2018
- **WhatsApp:** tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]



- Facebook: usa Signal solo nelle Secret Conversations
- Allo: applicazione mobile di messaggistica istantanea di Google, non esiste più dal 12 marzo 2019
- Duo: applicazione per videochiamate e chat mobile di Google
- Whatsapp: introdusse Signal per la prima volta nel 2014 per utenti Android, estendendolo a tutti gli utenti nel 2016
- Google: introduce Signal di default nell'applicazione di messaggi su Android

Applicazione Signal

L'Applicazione e il Protocollo Signal

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

► Facebook: introdusse la feature Secret Conversations per gli utenti di Facebook Messenger nel luglio 2016

► Allo: rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito

► Duo: protezione delle videochat

► Skype: conversazioni private dal 2018

► Whatsapp: tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]

# Applicazione Signal

## L'Applicazione e il Protocollo Signal

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione

L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- ▶ **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- ▶ **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- ▶ **Duo:** protezione delle videochat
- ▶ **Skype:** conversazioni private dal 2018
- ▶ **WhatsApp:** tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]



- Facebook: usa Signal solo nelle Secret Conversations
- Allo: applicazione mobile di messaggistica istantanea di Google, non esiste più dal 12 marzo 2019
- Duo: applicazione per videochiamate e chat mobile di Google
- Whatsapp: introdusse Signal per la prima volta nel 2014 per utenti Android, estendendolo a tutti gli utenti nel 2016
- Google: introduce Signal di default nell'applicazione di messaggi su Android

Applicazione Signal  
L'Applicazione e il Protocollo Signal

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- ▶ **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- ▶ **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- ▶ **Duo:** protezione delle videochat
- ▶ **Skype:** conversazioni private dal 2018
- ▶ **WhatsApp:** tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]

# Applicazione Signal

## L'Applicazione e il Protocollo Signal

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione

L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- ▶ **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- ▶ **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- ▶ **Duo:** protezione delle videochat
- ▶ **Skype:** conversazioni private dal 2018
- ▶ **WhatsApp:** tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]



- Facebook: usa Signal solo nelle Secret Conversations
- Allo: applicazione mobile di messaggistica istantanea di Google, non esiste più dal 12 marzo 2019
- Duo: applicazione per videochiamate e chat mobile di Google
- Whatsapp: introdusse Signal per la prima volta nel 2014 per utenti Android, estendendolo a tutti gli utenti nel 2016
- Google: introduce Signal di default nell'applicazione di messaggi su Android

Applicazione Signal  
L'Applicazione e il Protocollo Signal

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- ▶ **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- ▶ **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- ▶ **Duo:** protezione delle videochat
- ▶ **Skype:** conversazioni private dal 2018

WhatsApp: tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]

# Applicazione Signal

## L'Applicazione e il Protocollo Signal

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione

L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- ▶ **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- ▶ **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- ▶ **Duo:** protezione delle videochat
- ▶ **Skype:** conversazioni private dal 2018
- ▶ **WhatsApp:** tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]



- Facebook: usa Signal solo nelle Secret Conversations
- Allo: applicazione mobile di messaggistica istantanea di Google, non esiste più dal 12 marzo 2019
- Duo: applicazione per videochiamate e chat mobile di Google
- Whatsapp: introdusse Signal per la prima volta nel 2014 per utenti Android, estendendolo a tutti gli utenti nel 2016
- Google: introduce Signal di default nell'applicazione di messaggi su Android

Applicazione Signal  
L'Applicazione e il Protocollo Signal

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- ▶ **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- ▶ **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- ▶ **Duo:** protezione delle videochat
- ▶ **Skype:** conversazioni private dal 2018

» WhatsApp: tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]

# Applicazione Signal

## L'Applicazione e il Protocollo Signal

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione

L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- ▶ **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- ▶ **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- ▶ **Duo:** protezione delle videochat
- ▶ **Skype:** conversazioni private dal 2018
- ▶ **WhatsApp:** tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]



- Facebook: usa Signal solo nelle Secret Conversations
- Allo: applicazione mobile di messaggistica istantanea di Google, non esiste più dal 12 marzo 2019
- Duo: applicazione per videochiamate e chat mobile di Google
- Whatsapp: introdusse Signal per la prima volta nel 2014 per utenti Android, estendendolo a tutti gli utenti nel 2016
- Google: introduce Signal di default nell'applicazione di messaggi su Android

Applicazione Signal  
L'Applicazione e il Protocollo Signal

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:

- ▶ **Facebook:** introdusse la feature *Secret Conversations* per gli utenti di Facebook Messenger nel luglio 2016
- ▶ **Allo:** rilasciata nel settembre 2016, sfruttava il Protocollo Signal se utilizzata in modalità incognito
- ▶ **Duo:** protezione delle videochat
- ▶ **Skype:** conversazioni private dal 2018
- ▶ **WhatsApp:** tra le maggiori applicazioni che implementano Signal è l'unica che garantisce di default la crittografia end-to-end delle conversazioni (da aprile 2016)

[Gre29], [Lumer]



# Applicazione Signal

## L'Applicazione e il Protocollo Signal

### 1 Sommario

### 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

Ciascuna di queste *features* richiede che le conversazioni intraprese siano dichiarate “private” affinché sia possibile applicare la crittografia end-to-end su tutto il contenuto che viene scambiato

Inoltre, conversazioni già avvenute non possono essere protette applicando il protocollo ex post.  
[Mar16]



2022-04-25

La dichiarazione delle conversazioni come “private” avviene in genere per selezione esplicita da parte dell'utente e non di default.

WhatsApp implementa automaticamente la crittografia end-to-end sia per le chat private che per quelle di gruppo, tuttavia se si vuole verificare che le conversazioni siano private è necessario che entrambe le persone che partecipano alla conversazione selezionino la chat di interesse, clicchino sul nome del contatto, selezionino l'opzione “Crittografia” e scannerizzino il codice QR che viene presentato sul dispositivo dell'altro utente oppure confrontino i numeri a 60 cifre presentati.

Applicazione Signal  
L'Applicazione e il Protocollo Signal

Ciascuna di queste *features* richiede che le conversazioni intraprese siano dichiarate “private” affinché sia possibile applicare la crittografia end-to-end su tutto il contenuto che viene scambiato

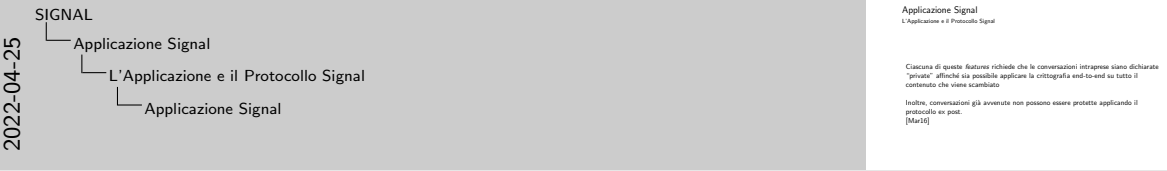
Inoltre, conversazioni già avvenute non possono essere protette applicando il protocollo ex post.  
[Mar16]

# Applicazione Signal

## L'Applicazione e il Protocollo Signal

Ciascuna di queste *features* richiede che le conversazioni intraprese siano dichiarate “private” affinché sia possibile applicare la crittografia end-to-end su tutto il contenuto che viene scambiato

Inoltre, conversazioni già avvenute non possono essere protette applicando il protocollo ex post.  
[Mar16]



La dichiarazione delle conversazioni come “private” avviene in genere per selezione esplicita da parte dell'utente e non di default.

WhatsApp implementa automaticamente la crittografia end-to-end sia per le chat private che per quelle di gruppo, tuttavia se si vuole verificare che le conversazioni siano private è necessario che entrambe le persone che partecipano alla conversazione selezionino la chat di interesse, clicchino sul nome del contatto, selezionino l'opzione “Crittografia” e scannerizzino il codice QR che viene presentato sul dispositivo dell'altro utente oppure confrontino i numeri a 60 cifre presentati.

# Applicazione Signal

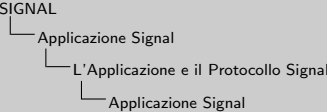
## L'Applicazione e il Protocollo Signal

La sicurezza garantita dall'implementazione del protocollo è relativa al fatto che tutti i prodotti OWS sono incentrati sulla privacy degli utenti, infatti:

- ▶ Salvano solo le informazioni strettamente necessarie
- ▶ Rendono impossibile a terze parti accedere ai messaggi o ai file scambiati tra gli utenti (grazie alla crittografia end-to-end)

[Lumer]

2022-04-25



Applicazione Signal  
L'Applicazione e il Protocollo Signal

La sicurezza garantita dall'implementazione del protocollo è relativa al fatto che tutti i prodotti OWS sono incentrati sulla privacy degli utenti, infatti:

- ▶ Salvano solo le informazioni strettamente necessarie
- ▶ Rendono impossibile a terze parti accedere ai messaggi o ai file scambiati tra gli utenti (grazie alla crittografia end-to-end)

[Lumer]

# Applicazione Signal

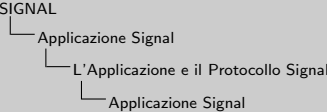
## L'Applicazione e il Protocollo Signal

La sicurezza garantita dall'implementazione del protocollo è relativa al fatto che tutti i prodotti OWS sono incentrati sulla privacy degli utenti, infatti:

- ▶ Salvano solo le informazioni strettamente necessarie
- ▶ Rendono impossibile a terze parti accedere ai messaggi o ai file scambiati tra gli utenti (grazie alla crittografia end-to-end)

[Lumer]

2022-04-25



Applicazione Signal  
L'Applicazione e il Protocollo Signal

La sicurezza garantita dall'implementazione del protocollo è relativa al fatto che tutti i prodotti OWS sono incentrati sulla privacy degli utenti, infatti:

- ▶ Salvano solo le informazioni strettamente necessarie
- ▶ Rendono impossibile a terze parti accedere ai messaggi o ai file scambiati tra gli utenti (grazie alla crittografia end-to-end)

[Lumer]

# End-to-End Encryption

## 1 Sommario

## 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

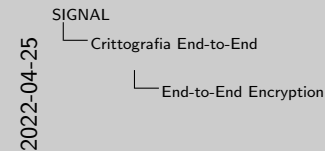
## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia



## End-to-End Encryption

La crittografia End-to-End (E2EE) è un processo di comunicazione sicura che impedisce a terze parti di accedere ai dati trasferiti da un utente a un altro.

Solamente gli utenti che sono in possesso della chiave segreta possono decifrare il testo cifrato e leggere il messaggio come *plaintext*.

In linea di massima E2EE garantisce che potenziali *eavesdroppers* non possano accedere alle chiavi necessarie per decifrare la conversazione. [Gre15]

Dati protetti da crittografia sono tali per cui solamente le persone autorizzate possono leggerne il contenuto in chiaro, mentre per tutti gli altri utenti si tratta di dati presentati in un formato non leggibile.

Grazie alla E2EE è possibile proteggere i dati trasmessi da terze parti malintenzionate che possono includere i provider dei servizi di telecomunicazione, gli Internet provider e utenti malevoli.

La E2EE si assicura inoltre che le comunicazioni tra due endpoint siano sicure.

La crittografia End-to-End (E2EE) è un processo di comunicazione sicura che impedisce a terze parti di accedere ai dati trasferiti da un utente a un altro.

Solamente gli utenti che sono in possesso della chiave segreta possono decifrare il testo cifrato e leggere il messaggio come *plaintext*.

In linea di massima E2EE garantisce che potenziali *eavesdroppers* non possano accedere alle chiavi necessarie per decifrare la conversazione. [Gre15]

# End-to-End Encryption

## 1 Sommario

## 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

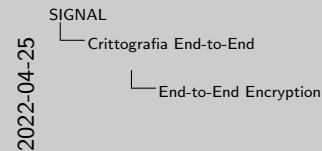
Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia

La crittografia End-to-End (E2EE) è un processo di comunicazione sicura che impedisce a terze parti di accedere ai dati trasferiti da un utente a un altro.

Solamente gli utenti che sono in possesso della chiave segreta possono decifrare il testo cifrato e leggere il messaggio come *plaintext*.

In linea di massima E2EE garantisce che potenziali *eavesdroppers* non possano accedere alle chiavi necessarie per decifrare la conversazione. [Gre15]



Dati protetti da crittografia sono tali per cui solamente le persone autorizzate possono leggerne il contenuto in chiaro, mentre per tutti gli altri utenti si tratta di dati presentati in un formato non leggibile.

Grazie alla E2EE è possibile proteggere i dati trasmessi da terze parti malintenzionate che possono includere i provider dei servizi di telecomunicazione, gli Internet provider e utenti malevoli.

La E2EE si assicura inoltre che le comunicazioni tra due endpoint siano sicure.

### End-to-End Encryption

La crittografia End-to-End (E2EE) è un processo di comunicazione sicura che impedisce a terze parti di accedere ai dati trasferiti da un utente a un altro.

Solamente gli utenti che sono in possesso della chiave segreta possono decifrare il testo cifrato e leggere il messaggio come *plaintext*.

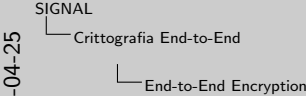
In linea di massima E2EE garantisce che potenziali *eavesdroppers* non possano accedere alle chiavi necessarie per decifrare la conversazione. [Gre15]

# End-to-End Encryption

La crittografia End-to-End (E2EE) è un processo di comunicazione sicura che impedisce a terze parti di accedere ai dati trasferiti da un utente a un altro.

Solamente gli utenti che sono in possesso della chiave segreta possono decifrare il testo cifrato e leggere il messaggio come *plaintext*.

In linea di massima E2EE garantisce che potenziali *eavesdroppers* non possano accedere alle chiavi necessarie per decifrare la conversazione. [Gre15]



Dati protetti da crittografia sono tali per cui solamente le persone autorizzate possono leggerne il contenuto in chiaro, mentre per tutti gli altri utenti si tratta di dati presentati in un formato non leggibile.

Grazie alla E2EE è possibile proteggere i dati trasmessi da terze parti malintenzionate che possono includere i provider dei servizi di telecomunicazione, gli Internet provider e utenti malevoli.

La E2EE si assicura inoltre che le comunicazioni tra due endpoint siano sicure.

La crittografia End-to-End (E2EE) è un processo di comunicazione sicura che impedisce a terze parti di accedere ai dati trasferiti da un utente a un altro.

Solamente gli utenti che sono in possesso della chiave segreta possono decifrare il testo cifrato e leggere il messaggio come *plaintext*.

In linea di massima E2EE garantisce che potenziali *eavesdroppers* non possano accedere alle chiavi necessarie per decifrare la conversazione. [Gre15]

# End-to-End Encryption

E2EE si basa sulla crittografia *asimmetrica*.

La crittografia **asimmetrica**, o a **chiave pubblica**, cifra e decifra i dati usando due chiavi distinte:

- ▶ La chiave pubblica è usata per cifrare un messaggio e inviarlo al proprietario della chiave pubblica
- ▶ In seguito, il messaggio può essere decifrato solo utilizzando la corrispondente chiave privata.

Al contrario, la crittografia **simmetrica** utilizza una sola chiave privata per cifrare il *plaintext* e decifrare il *ciphertext*.

2022-04-25

SIGNAL

Crittografia End-to-End

End-to-End Encryption

End-to-End Encryption

E2EE si basa sulla crittografia asimmetrica. La crittografia **asimmetrica**, o a **chiave pubblica**, cifra e decifra i dati usando due chiavi distinte:

- ▶ La chiave pubblica è usata per cifrare un messaggio e inviarlo al proprietario della chiave pubblica
- ▶ In seguito, il messaggio può essere decifrato solo utilizzando la corrispondente chiave privata.

Al contrario, la crittografia **simmetrica** utilizza una sola chiave privata per cifrare il *plaintext* e decifrare il *ciphertext*.

I messaggi vengono crittografati dal mittente, pertanto, anche se intercettati da una terza persona, essi non le saranno visibili in *plaintext* e saranno

dunque conservabili solo in *ciphertext*.

Al contrario, il destinatario sarà in grado di ricevere i dati e decifrarli per sé.



# End-to-End Encryption

## 1 Sommario

## 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

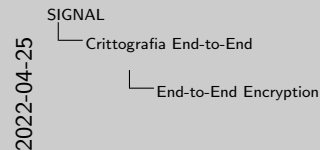
## 5 Bibliografia

E2EE si basa sulla crittografia *asimmetrica*.

La crittografia **asimmetrica**, o **a chiave pubblica**, cifra e decifra i dati usando due chiavi distinte:

- La chiave pubblica è usata per cifrare un messaggio e inviarlo al proprietario della chiave pubblica
- In seguito, il messaggio può essere decifrato solo utilizzando la corrispondente chiave privata.

Al contrario, la crittografia **simmetrica** utilizza una sola chiave privata per cifrare il *plaintext* e decifrare il *ciphertext*.



I messaggi vengono crittografati dal mittente, pertanto, anche se intercettati da una terza persona, essi non le saranno visibili in *plaintext* e saranno dunque conservabili solo in *ciphertext*.

Al contrario, il destinatario sarà in grado di ricevere i dati e decifrarli per sé.

## End-to-End Encryption

E2EE si basa sulla crittografia *asimmetrica*.  
La crittografia **asimmetrica**, o **a chiave pubblica**, cifra e decifra i dati usando due chiavi distinte:

- La chiave pubblica è usata per cifrare un messaggio e inviarlo al proprietario della chiave pubblica
- In seguito, il messaggio può essere decifrato solo utilizzando la corrispondente chiave privata.

Al contrario, la crittografia **simmetrica** utilizza una sola chiave privata per cifrare il *plaintext* e decifrare il *ciphertext*.

# End-to-End Encryption

## 1 Sommario

## 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

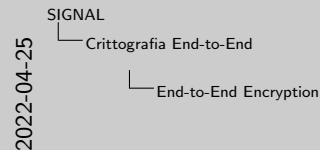
## 5 Bibliografia

E2EE si basa sulla crittografia *asimmetrica*.

La crittografia **asimmetrica**, o **a chiave pubblica**, cifra e decifra i dati usando due chiavi distinte:

- La chiave pubblica è usata per cifrare un messaggio e inviarlo al proprietario della chiave pubblica
- In seguito, il messaggio può essere decifrato solo utilizzando la corrispondente chiave privata.

Al contrario, la crittografia **simmetrica** utilizza una sola chiave privata per cifrare il *plaintext* e decifrare il *ciphertext*.



I messaggi vengono crittografati dal mittente, pertanto, anche se intercettati da una terza persona, essi non le saranno visibili in *plaintext* e saranno dunque conservabili solo in *ciphertext*.

Al contrario, il destinatario sarà in grado di ricevere i dati e decifrarli per sé.

## End-to-End Encryption

E2EE si basa sulla crittografia *asimmetrica*.  
La crittografia **asimmetrica**, o **a chiave pubblica**, cifra e decifra i dati usando due chiavi distinte:

- La chiave pubblica è usata per cifrare un messaggio e inviarlo al proprietario della chiave pubblica
- In seguito, il messaggio può essere decifrato solo utilizzando la corrispondente chiave privata.

Al contrario, la crittografia *simmetrica* utilizza una sola chiave privata per cifrare il *plaintext* e decifrare il *ciphertext*.

# End-to-End Encryption

## 1 Sommario

## 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

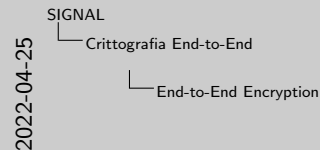
## 5 Bibliografia

E2EE si basa sulla crittografia *asimmetrica*.

La crittografia **asimmetrica**, o **a chiave pubblica**, cifra e decifra i dati usando due chiavi distinte:

- La chiave pubblica è usata per cifrare un messaggio e inviarlo al proprietario della chiave pubblica
- In seguito, il messaggio può essere decifrato solo utilizzando la corrispondente chiave privata.

Al contrario, la crittografia **simmetrica** utilizza una sola chiave privata per cifrare il *plaintext* e decifrare il *ciphertext*.



I messaggi vengono crittografati dal mittente, pertanto, anche se intercettati da una terza persona, essi non le saranno visibili in *plaintext* e saranno dunque conservabili solo in *ciphertext*.

Al contrario, il destinatario sarà in grado di ricevere i dati e decifrarli per sé.

## End-to-End Encryption

E2EE si basa sulla crittografia *asimmetrica*.  
La crittografia **asimmetrica**, o **a chiave pubblica**, cifra e decifra i dati usando due chiavi distinte:

- La chiave pubblica è usata per cifrare un messaggio e inviarlo al proprietario della chiave pubblica
- In seguito, il messaggio può essere decifrato solo utilizzando la corrispondente chiave privata.

Al contrario, la crittografia **simmetrica** utilizza una sola chiave privata per cifrare il *plaintext* e decifrare il *ciphertext*.

# End-to-End Encryption

## 1 Sommario

## 2 Applicazione Signal

Storia  
dell'Applicazione

L'Applicazione e il  
Protocollo Signal

## 3 Crittografia End-to-End

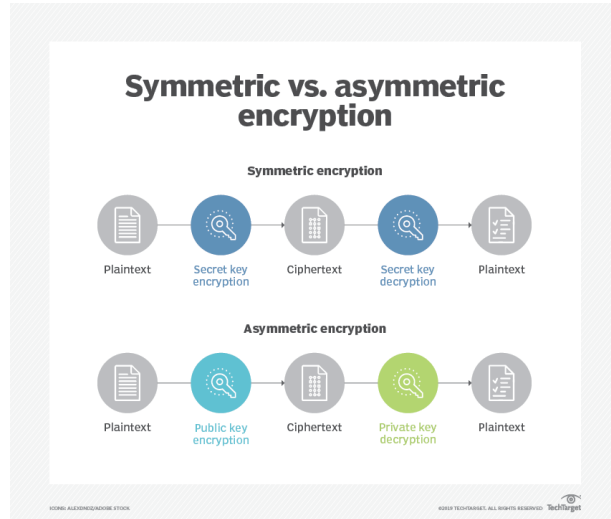
Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni

WhatsApp VS Signal  
VS Telegram

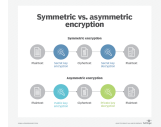
## 5 Bibliografia



2022-04-25

SIGNAL  
└─ Crittografia End-to-End  
    └─ End-to-End Encryption

End-to-End Encryption



# End-to-End Encryption

## Applicazioni

### 1 Sommario

### 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

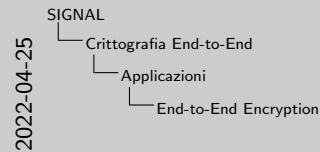
### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

- **Comunicazioni sicure:** applicazioni di messaggistica e posta elettronica per mantenere private le conversazioni degli utenti;
- **Gestione password:** in questo caso a entrambi gli endpoint della comunicazione si trova lo stesso utente, che è l'unica persona munita di chiave;
- **Data storage:** nei servizi di storage in cloud può anche essere garantita E2EE *in transit*, proteggendo i dati degli utenti anche dall'accesso da parte dei fornitori del servizio in cloud;

[IBM]



End-to-End Encryption  
Applicazioni

- **Comunicazioni sicure:** applicazioni di messaggistica e posta elettronica per mantenere private le conversazioni degli utenti;
- **Gestione password:** in questo caso a entrambi gli endpoint della comunicazione si trova lo stesso utente, che è l'unica persona munita di chiave;
- **Data storage:** nei servizi di storage in cloud può anche essere garantita E2EE *in transit*, proteggendo i dati degli utenti anche dall'accesso da parte dei fornitori del servizio in cloud;

[IBM]

Alcuni sistemi, come ad esempio Lavabit e Hushmail, hanno in passato dichiarato di implementare la crittografia end-to-end nonostante ciò non fosse vero. [Gra 7]

Lavabit, servizio email in passato ritenuto sicuro e oggi non più attivo, nel 2014 consegnò al governo americano le chiavi che utilizzava per proteggere i dati dei propri utenti in occasione delle indagini sul caso Snowden. La controversia nacque dal fatto che la compagnia aveva in precedenza dichiarato che il proprio livello di sicurezza era tale che nemmeno gli amministratori della compagnia stessa avevano accesso al contenuto delle mail scambiate dai propri utenti. [Pou16], [GM13]

Hushmail, altro email provider dichiarato sicuro, violò la privacy dei propri utenti utilizzandone le password per decrittare le email e consegnarle al governo federale in *plaintext*. [Sin 7]

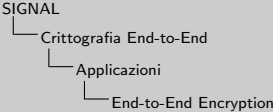
Altri sistemi, come per esempio Telegram, non implementano la crittografia end-to-end di default e sono pertanto stati criticati.

In modo particolare Telegram non la implementa né per le chat di gruppo né per i client desktop. Tra le altre critiche mosse all'applicazione c'è quella di utilizzare il protocollo di crittografia non standard MTPROTO. [EPP 1]

# End-to-End Encryption

## Applicazioni

2022-04-25



End-to-End Encryption  
Applicazioni

- **Comunicazioni sicure:** applicazioni di messaggistica e posta elettronica per mantenere private le conversazioni degli utenti;
- **Gestione password:** in questo caso a entrambi gli endpoint della comunicazione si trova lo stesso utente, che è l'unica persona munita di chiave;
- **Data storage:** nei servizi di storage in cloud può anche essere garantita E2EE *in transit*, proteggendo i dati degli utenti anche dall'accesso da parte dei fornitori del servizio in cloud;

[IBM]

Alcuni sistemi, come ad esempio Lavabit e Hushmail, hanno in passato dichiarato di implementare la crittografia end-to-end nonostante ciò non fosse vero. [Gra 7]

Lavabit, servizio email in passato ritenuto sicuro e oggi non più attivo, nel 2014 consegnò al governo americano le chiavi che utilizzava per proteggere i dati dei propri utenti in occasione delle indagini sul caso Snowden. La controversia nacque dal fatto che la compagnia aveva in precedenza dichiarato che il proprio livello di sicurezza era tale che nemmeno gli amministratori della compagnia stessa avevano accesso al contenuto delle mail scambiate dai propri utenti. [Pou16], [GM13]

Hushmail, altro email provider dichiarato sicuro, violò la privacy dei propri utenti utilizzandone le password per decrittare le email e consegnarle al governo federale in *plaintext*. [Sin 7]

Altri sistemi, come per esempio Telegram, non implementano la crittografia end-to-end di default e sono pertanto stati criticati.

In modo particolare Telegram non la implementa né per le chat di gruppo né per i client desktop. Tra le altre critiche mosse all'applicazione c'è quella di utilizzare il protocollo di crittografia non standard MTPROTO. [EPP 1]

- **Comunicazioni sicure:** applicazioni di messaggistica e posta elettronica per mantenere private le conversazioni degli utenti;
- **Gestione password:** in questo caso a entrambi gli endpoint della comunicazione si trova lo stesso utente, che è l'unica persona munita di chiave;
- **Data storage:** nei servizi di storage in cloud può anche essere garantita E2EE *in transit*, proteggendo i dati degli utenti anche dall'accesso da parte dei fornitori del servizio in cloud;

[IBM]

### 1 Sommario

### 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

# End-to-End Encryption

## Applicazioni

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

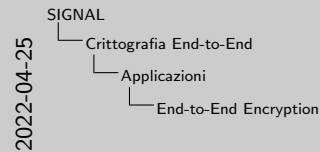
### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

- **Comunicazioni sicure:** applicazioni di messaggistica e posta elettronica per mantenere private le conversazioni degli utenti;
- **Gestione password:** in questo caso a entrambi gli endpoint della comunicazione si trova lo stesso utente, che è l'unica persona munita di chiave;
- **Data storage:** nei servizi di storage in cloud può anche essere garantita E2EE *in transit*, proteggendo i dati degli utenti anche dall'accesso da parte dei fornitori del servizio in cloud;

[IBM]



End-to-End Encryption  
Applicazioni

- **Comunicazioni sicure:** applicazioni di messaggistica e posta elettronica per mantenere private le conversazioni degli utenti;
- **Gestione password:** in questo caso a entrambi gli endpoint della comunicazione si trova lo stesso utente, che è l'unica persona munita di chiave;
- **Data storage:** nei servizi di storage in cloud può anche essere garantita E2EE *in transit*, proteggendo i dati degli utenti anche dall'accesso da parte dei fornitori del servizio in cloud;

[IBM]

Alcuni sistemi, come ad esempio Lavabit e Hushmail, hanno in passato dichiarato di implementare la crittografia end-to-end nonostante ciò non fosse vero. [Gra 7]

Lavabit, servizio email in passato ritenuto sicuro e oggi non più attivo, nel 2014 consegnò al governo americano le chiavi che utilizzava per proteggere i dati dei propri utenti in occasione delle indagini sul caso Snowden. La controversia nacque dal fatto che la compagnia aveva in precedenza dichiarato che il proprio livello di sicurezza era tale che nemmeno gli amministratori della compagnia stessa avevano accesso al contenuto delle mail scambiate dai propri utenti. [Pou16], [GM13]

Hushmail, altro email provider dichiarato sicuro, violò la privacy dei propri utenti utilizzandone le password per decrittare le email e consegnarle al governo federale in *plaintext*. [Sin 7]

Altri sistemi, come per esempio Telegram, non implementano la crittografia end-to-end di default e sono pertanto stati criticati.

In modo particolare Telegram non la implementa né per le chat di gruppo né per i client desktop. Tra le altre critiche mosse all'applicazione c'è quella di utilizzare il protocollo di crittografia non standard MTPROTO. [EPP 1]

1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

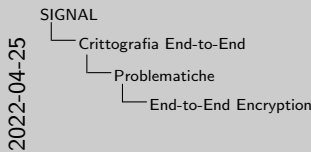
5 Bibliografia

# End-to-End Encryption

## Problematiche

La E2EE non garantisce di per sé né la sicurezza né la privacy, in quanto i dati trasmessi potrebbero essere protetti in modo poco sicuro sui dispositivi endpoint. Tuttavia, l'implementazione della E2EE consente di applicare una protezione dei dati migliore della sola crittografia “in transit”.

Per molti sistemi di messaggistica i messaggi passano attraverso un intermediario che li conserva finché non vengono recuperati dal destinatario. Anche se protetti da crittografia, essi lo sono solamente in transito, quindi possono essere letti dai provider di servizi.  
[int20], [IBM]



In questo modo è possibile monitorare il contenuto dei messaggi (per esempio in cerca di contenuti offensivi o pericolosi) ma si corre anche il rischio che utenti non autorizzati e/o malintenzionati aventi accesso allo storage dei messaggi possano fare un uso improprio dei contenuti.

Nella crittografia “in transit” è possibile o salvare direttamente i messaggi decrittati oppure salvare i dati crittografati e la chiave con cui decrittarli sullo stesso database.

End-to-End Encryption

Problematiche

La E2EE non garantisce di per sé né la sicurezza né la privacy, in quanto i dati trasmessi potrebbero essere protetti in modo poco sicuro sui dispositivi endpoint. Tuttavia, l'implementazione della E2EE consente di applicare una protezione dei dati migliore della sola crittografia “in transit”.

Per molti sistemi di messaggistica i messaggi passano attraverso un intermediario che li conserva finché non vengono recuperati dal destinatario. Anche se protetti da crittografia, essi lo sono solamente in transito, quindi possono essere letti dai provider di servizi.  
[int20], [IBM]



1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

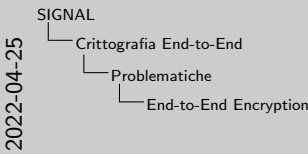
5 Bibliografia

# End-to-End Encryption

## Problematiche

La E2EE non garantisce di per sé né la sicurezza né la privacy, in quanto i dati trasmessi potrebbero essere protetti in modo poco sicuro sui dispositivi endpoint. Tuttavia, l'implementazione della E2EE consente di applicare una protezione dei dati migliore della sola crittografia **“in transit”**.

Per molti sistemi di messaggistica i messaggi passano attraverso un intermediario che li conserva finché non vengono recuperati dal destinatario. Anche se protetti da crittografia, essi lo sono solamente in transito, quindi possono essere letti dai provider di servizi.  
[int20], [IBM]



In questo modo è possibile monitorare il contenuto dei messaggi (per esempio in cerca di contenuti offensivi o pericolosi) ma si corre anche il rischio che utenti non autorizzati e/o malintenzionati aventi accesso allo storage dei messaggi possano fare un uso improprio dei contenuti.

Nella crittografia “in transit” è possibile o salvare direttamente i messaggi decrittati oppure salvare i dati crittografati e la chiave con cui decrittarli sullo stesso database.

End-to-End Encryption

Problematiche

La E2EE non garantisce di per sé né la sicurezza né la privacy, in quanto i dati trasmessi potrebbero essere protetti in modo poco sicuro sui dispositivi endpoint. Tuttavia, l'implementazione della E2EE consente di applicare una protezione dei dati migliore della sola crittografia **“in transit”**.

Per molti sistemi di messaggistica i messaggi passano attraverso un intermediario che li conserva finché non vengono recuperati dal destinatario. Anche se protetti da crittografia, essi lo sono solamente in transito, quindi possono essere letti dai provider di servizi.  
[int20], [IBM]

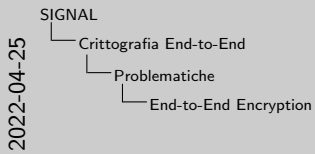
# End-to-End Encryption

## Problematiche

### Ulteriori problematiche:

- ▶ **Endpoint security:** gli endpoint sono vulnerabili se non protetti adeguatamente
- ▶ **Attacchi di tipo Man-in-the-Middle:** la conversazione può essere soggetta a *eavesdropping*
- ▶ **Backdoors:** metodi per bypassare l'autenticazione standard o la protezione crittografica di un dispositivo. Se non volute, possono essere introdotte tramite attacchi cyber e poi sfruttate per violare la sicurezza del sistema

[Gre15], [IBM]



- Endpoint security: E2EE protegge i dati solo tra i due endpoint; ciò significa che i due endpoint possono essere soggetti ad attacchi;
- Attacchi MITM: anziché forzare la crittografia dei dati, ci si può aspettare un tentativo da parte di terzi malintenzionati di impersonare il destinatario durante. Essi possono, per esempio, impersonare il destinatario durante lo scambio di chiavi con il mittente, decifrare il messaggio inviato e poi inoltrarlo al vero destinatario senza farsi notare. Una soluzione per questo tipo di attacchi è introdurre un metodo di autenticazione (per es. certification authorities, web of trust, fingerprint numeriche o come QR code)
- Backdoors: nonostante le *backdoors* non siano sempre implementate volutamente, esse possono essere introdotte grazie a *cyber-attacks* e poi essere utilizzate per la negoziazione delle chiavi o per oltrepassare la protezione crittografica.

End-to-End Encryption

Problematiche

Ulteriori problematiche:

- ▶ **Endpoint security:** gli endpoint sono vulnerabili se non protetti adeguatamente
- ▶ Attacchi di tipo Man-in-the-Middle: la conversazione può essere soggetta a *eavesdropping*
- ▶ Backdoors: metodi per bypassare l'autenticazione standard o la protezione crittografica di un dispositivo. Se non volute, possono essere introdotte tramite attacchi cyber e poi sfruttate per violare la sicurezza del sistema

[Gre15], [IBM]

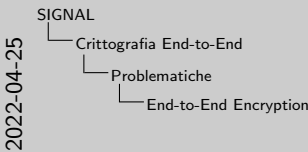
# End-to-End Encryption

## Problematiche

### Ulteriori problematiche:

- ▶ **Endpoint security:** gli endpoint sono vulnerabili se non protetti adeguatamente
- ▶ **Attacchi di tipo Man-in-the-Middle:** la conversazione può essere soggetta a *eavesdropping*
- ▶ **Backdoors:** metodi per bypassare l'autenticazione standard o la protezione crittografica di un dispositivo. Se non volute, possono essere introdotte tramite attacchi cyber e poi sfruttate per violare la sicurezza del sistema

[Gre15], [IBM]



- Endpoint security: E2EE protegge i dati solo tra i due endpoint; ciò significa che i due endpoint possono essere soggetti ad attacchi;
- Attacchi MITM: anziché forzare la crittografia dei dati, ci si può aspettare un tentativo da parte di terzi malintenzionati di impersonare il destinatario durante. Essi possono, per esempio, impersonare il destinatario durante lo scambio di chiavi con il mittente, decifrare il messaggio inviato e poi inoltrarlo al vero destinatario senza farsi notare. Una soluzione per questo tipo di attacchi è introdurre un metodo di autenticazione (per es. certification authorities, web of trust, fingerprint numeriche o come QR code)
- Backdoors: nonostante le *backdoors* non siano sempre implementate volutamente, esse possono essere introdotte grazie a *cyber-attacks* e poi essere utilizzate per la negoziazione delle chiavi o per oltrepassare la protezione crittografica.

End-to-End Encryption

Problematiche

Ulteriori problematiche:

- ▶ **Endpoint security:** gli endpoint sono vulnerabili se non protetti adeguatamente
- ▶ **Attacchi di tipo Man-in-the-Middle:** la conversazione può essere soggetta a *eavesdropping*
- ▶ **Backdoors:** metodi per bypassare l'autenticazione standard o la protezione crittografica di un dispositivo. Se non volute, possono essere introdotte tramite attacchi cyber e poi sfruttate per violare la sicurezza del sistema

[Gre15], [IBM]

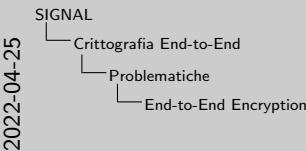
# End-to-End Encryption

## Problematiche

### Ulteriori problematiche:

- ▶ **Endpoint security:** gli endpoint sono vulnerabili se non protetti adeguatamente
- ▶ **Attacchi di tipo Man-in-the-Middle:** la conversazione può essere soggetta a *eavesdropping*
- ▶ **Backdoors:** metodi per bypassare l'autenticazione standard o la protezione crittografica di un dispositivo. Se non volute, possono essere introdotte tramite attacchi cyber e poi sfruttate per violare la sicurezza del sistema

[Gre15], [IBM]



- Endpoint security: E2EE protegge i dati solo tra i due endpoint; ciò significa che i due endpoint possono essere soggetti ad attacchi;
- Attacchi MITM: anziché forzare la crittografia dei dati, ci si può aspettare un tentativo da parte di terzi malintenzionati di impersonare il destinatario durante. Essi possono, per esempio, impersonare il destinatario durante lo scambio di chiavi con il mittente, decifrare il messaggio inviato e poi inoltrarlo al vero destinatario senza farsi notare. Una soluzione per questo tipo di attacchi è introdurre un metodo di autenticazione (per es. certification authorities, web of trust, fingerprint numeriche o come QR code)
- Backdoors: nonostante le *backdoors* non siano sempre implementate volutamente, esse possono essere introdotte grazie a *cyber-attacks* e poi essere utilizzate per la negoziazione delle chiavi o per oltrepassare la protezione crittografica.

End-to-End Encryption

Problematiche

Ulteriori problematiche:

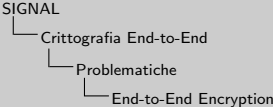
- ▶ **Endpoint security:** gli endpoint sono vulnerabili se non protetti adeguatamente
- ▶ **Attacchi di tipo Man-in-the-Middle:** la conversazione può essere soggetta a *eavesdropping*
- ▶ **Backdoors:** metodi per bypassare l'autenticazione standard o la protezione crittografica di un dispositivo. Se non volute, possono essere introdotte tramite attacchi cyber e poi sfruttate per violare la sicurezza del sistema

[Gre15], [IBM]

# End-to-End Encryption

## Problematiche

2022-04-25



End-to-End Encryption  
Problematiche

- **Complessità nel definire gli endpoint:** alcune implementazioni consentono di decodificare e ricodificare i dati lungo il percorso, quindi è necessario definire accuratamente gli estremi della trasmissione
- Privacy “eccessiva”: enti governativi non hanno modo di verificare la natura dei contenuti trasmessi dagli utenti, pertanto non sono in grado di prendere misure adeguate in caso di illeciti
- Metadati visibili
- Non vi è certezza che E2EE possa funzionare altrettanto bene con l’eventuale introduzione di *quantum computer* che rendano la crittografia obsoleta [LB21]

- **Complessità nel definire gli endpoint:** alcune implementazioni consentono di decodificare e ricodificare i dati lungo il percorso, quindi è necessario definire accuratamente gli estremi della trasmissione
- **Privacy “eccessiva”:** enti governativi non hanno modo di verificare la natura dei contenuti trasmessi dagli utenti, pertanto non sono in grado di prendere misure adeguate in caso di illeciti
- **Metadati visibili**
- Non vi è certezza che E2EE possa funzionare altrettanto bene con l’eventuale introduzione di *quantum computer* che rendano la crittografia obsoleta

[LB21]

# End-to-End Encryption

## Problematiche

### 1 Sommario

### 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

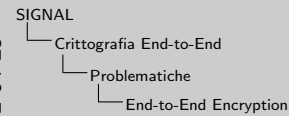
Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

- **Complessità nel definire gli endpoint:** alcune implementazioni consentono di decodificare e ricodificare i dati lungo il percorso, quindi è necessario definire accuratamente gli estremi della trasmissione
- **Privacy “eccessiva”:** enti governativi non hanno modo di verificare la natura dei contenuti trasmessi dagli utenti, pertanto non sono in grado di prendere misure adeguate in caso di illeciti
- **Metadati visibili**
- Non vi è certezza che E2EE possa funzionare altrettanto bene con l’eventuale introduzione di *quantum computer* che rendano la crittografia obsoleta

[LB21]

2022-04-25



End-to-End Encryption  
Problematiche

- **Complessità nel definire gli endpoint:** alcune implementazioni consentono di decodificare e ricodificare i dati lungo il percorso, quindi è necessario definire accuratamente gli estremi della trasmissione
- **Privacy “eccessiva”:** enti governativi non hanno modo di verificare la natura dei contenuti trasmessi dagli utenti, pertanto non sono in grado di prendere misure adeguate in caso di illeciti
- Metadati visibili
- Non vi è certezza che E2EE possa funzionare altrettanto bene con l’eventuale introduzione di *quantum computer* che rendano la crittografia obsoleta [LB21]

# End-to-End Encryption

## Problematiche

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

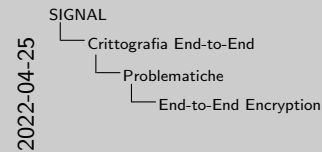
### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

- **Complessità nel definire gli endpoint:** alcune implementazioni consentono di decodificare e ricodificare i dati lungo il percorso, quindi è necessario definire accuratamente gli estremi della trasmissione
- **Privacy “eccessiva”:** enti governativi non hanno modo di verificare la natura dei contenuti trasmessi dagli utenti, pertanto non sono in grado di prendere misure adeguate in caso di illeciti
- **Metadati visibili**
  - Non vi è certezza che E2EE possa funzionare altrettanto bene con l’eventuale introduzione di *quantum computer* che rendano la crittografia obsoleta

[LB21]



## End-to-End Encryption Problematiche

- **Complessità nel definire gli endpoint:** alcune implementazioni consentono di decodificare e ricodificare i dati lungo il percorso, quindi è necessario definire accuratamente gli estremi della trasmissione
- **Privacy “eccessiva”:** enti governativi non hanno modo di verificare la natura dei contenuti trasmessi dagli utenti, pertanto non sono in grado di prendere misure adeguate in caso di illeciti
- **Metadati visibili**
  - Non vi è certezza che E2EE possa funzionare altrettanto bene con l’eventuale introduzione di *quantum computer* che rendano la crittografia obsoleta [LB21]

# End-to-End Encryption

## Problematiche

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

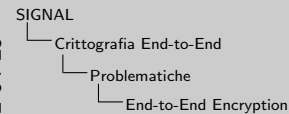
Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

- **Complessità nel definire gli endpoint:** alcune implementazioni consentono di decodificare e ricodificare i dati lungo il percorso, quindi è necessario definire accuratamente gli estremi della trasmissione
- **Privacy “eccessiva”:** enti governativi non hanno modo di verificare la natura dei contenuti trasmessi dagli utenti, pertanto non sono in grado di prendere misure adeguate in caso di illeciti
- **Metadati visibili**
- Non vi è certezza che E2EE possa funzionare altrettanto bene con l’eventuale introduzione di *quantum computer* che rendano la crittografia obsoleta

[LB21]

2022-04-25



End-to-End Encryption  
Problematiche

- **Complessità nel definire gli endpoint:** alcune implementazioni consentono di decodificare e ricodificare i dati lungo il percorso, quindi è necessario definire accuratamente gli estremi della trasmissione
- **Privacy “eccessiva”:** enti governativi non hanno modo di verificare la natura dei contenuti trasmessi dagli utenti, pertanto non sono in grado di prendere misure adeguate in caso di illeciti
- **Metadati visibili**
- Non vi è certezza che E2EE possa funzionare altrettanto bene con l’eventuale introduzione di *quantum computer* che rendano la crittografia obsoleta [LB21]



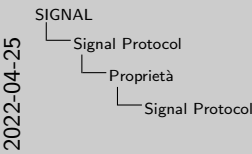
# Signal Protocol

## Proprietà

Tradizionalmente facciamo riferimento a tre proprietà come requisiti principali in ambito crittografico:

- **Confidenzialità**
- **Integrità**
- **Autenticità**

Le proprietà CIA spesso vengono accompagnate dalla **non ripudiabilità**



- **Confidenzialità:** dati trasmessi non vengono diffusi a terzi non coinvolti nella conversazione
- **Integrità:** dati trasmessi non danneggiati e/o dispersi
- **Autenticità:** possesso di una chiave da parte di due persone al fine di riconoscere e verificare l'identità dell'altro
- **Non ripudiabilità:** non deve essere possibile negare per es. la propria firma a un documento, per questo spesso è implementata tramite firme digitali

Signal Protocol

Proprietà

Tradizionalmente facciamo riferimento a tre proprietà come requisiti principali in ambito crittografico:

- **Confidenzialità**
- **Integrità**
- **Autenticità**

Le proprietà CIA spesso vengono accompagnate dalla non ripudiabilità

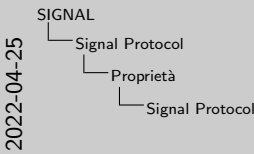
# Signal Protocol

## Proprietà

Tradizionalmente facciamo riferimento a tre proprietà come requisiti principali in ambito crittografico:

- **Confidenzialità**
- **Integrità**
- **Autenticità**

Le proprietà CIA spesso vengono accompagnate dalla **non ripudiabilità**



- **Confidenzialità:** dati trasmessi non vengono diffusi a terzi non coinvolti nella conversazione
- **Integrità:** dati trasmessi non danneggiati e/o dispersi
- **Autenticità:** possesso di una chiave da parte di due persone al fine di riconoscere e verificare l'identità dell'altro
- **Non ripudiabilità:** non deve essere possibile negare per es. la propria firma a un documento, per questo spesso è implementata tramite firme digitali

Signal Protocol

Proprietà

Tradizionalmente facciamo riferimento a tre proprietà come requisiti principali in ambito crittografico:

► **Confidenzialità**

► **Integrità**

► **Autenticità**

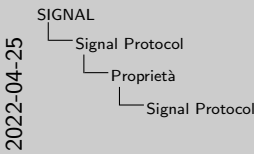
Le proprietà CIA spesso vengono accompagnate dalla **non ripudiabilità**

# Signal Protocol

## Proprietà

Ulteriori proprietà spesso richieste sono:

- **Forward Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i precedenti
- **Future Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i successivi
- **Cryptographic Deniability:** l'esistenza di un file cifrato o di un messaggio è rinne­gabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.



Cryptographic deniability in genere è più richiesta nelle applicazioni di messaggistica

Signal Protocol

Proprietà

Ulteriori proprietà spesso richieste sono:

► **Forward Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i precedenti

► **Future Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i successivi

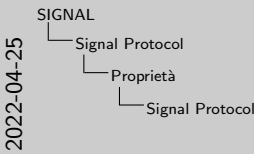
► **Cryptographic Deniability:** l'esistenza di un file cifrato o di un messaggio è rinne­gabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.

# Signal Protocol

## Proprietà

Ulteriori proprietà spesso richieste sono:

- **Forward Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i precedenti
- **Future Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i successivi
- **Cryptographic Deniability:** l'esistenza di un file cifrato o di un messaggio è rinne-gabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.



Cryptographic deniability in genere è più richiesta nelle applicazioni di messaggistica

Signal Protocol

Proprietà

Ulteriori proprietà spesso richieste sono:

► **Forward Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i precedenti

► **Future Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i successivi

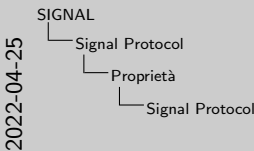
► **Cryptographic Deniability:** l'esistenza di un file cifrato o di un messaggio è rinne-gabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.

# Signal Protocol

## Proprietà

Ulteriori proprietà spesso richieste sono:

- **Forward Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i precedenti
- **Future Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i successivi
- **Cryptographic Deniability:** l'esistenza di un file cifrato o di un messaggio è rinnegabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.



Cryptographic deniability in genere è più richiesta nelle applicazioni di messaggistica

Signal Protocol

Proprietà

Ulteriori proprietà spesso richieste sono:

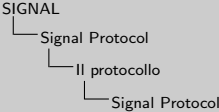
- **Forward Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i precedenti
- **Future Secrecy:** se una chiave è compromessa solo un messaggio è compromesso e non lo sono i successivi
- **Cryptographic Deniability:** l'esistenza di un file cifrato o di un messaggio è rinnegabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.

# Signal Protocol

## Il protocollo

Il protocollo Signal fornisce crittografia end-to-end a sistemi di messaggistica istantanea e di chiamate vocali, combinando l’algoritmo **“Double Ratchet”**, pre-chiavi e un triplo handshake Elliptic-curve Diffie–Hellman (3-DH).

2022-04-25



Signal Protocol  
Il protocollo

Il protocollo Signal fornisce crittografia end-to-end a sistemi di messaggistica istantanea e di chiamate vocali, combinando l’algoritmo **“Double Ratchet”**, pre-chiavi e un triplo handshake Elliptic-curve Diffie–Hellman (3-DH).

- 1 Sommario
- 2 Applicazione Signal
  - Storia dell'Applicazione
  - L'Applicazione e il Protocollo Signal
- 3 Crittografia End-to-End
  - Applicazioni
  - Problematiche
- 4 Signal Protocol
  - Proprietà
  - Il protocollo**
  - Difetti di progettazione
  - Considerazioni
  - WhatsApp VS Signal
  - VS Telegram
- 5 Bibliografia

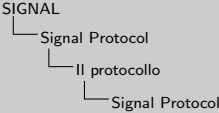
# Signal Protocol

## Il protocollo

Le specifiche di riferimento sono infatti: [sig]

- **X3DH**: protocollo di negoziazione delle chiavi Extended Triple Diffie-Hellman.
- **Double Ratchet**: algoritmo utilizzato da due parti per lo scambio di messaggi basato su una chiave segreta condivisa.
- **Sesame**: gestisce le sessioni crittografate in ambiente asincrono e multi-device.

2022-04-25



- Double Ratchet: le due parti derivano nuove chiavi per ogni messaggio in modo tale che chiavi usate in precedenza non possano essere ricavate dalle chiavi successive (grazie alla non invertibilità della funzione ratchet)
- X3DH: stabilisce una chiave segreta condivisa da due parti che si autenticano a vicenda basandosi su chiavi pubbliche. X3DH fornisce *forward secrecy* e *cryptographic deniability*

*Cryptographic deniability*: l'esistenza di un file cifrato o di un messaggio è rinnegabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.

Signal Protocol  
Il protocollo

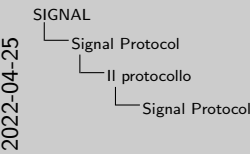
Le specifiche di riferimento sono infatti: [sig]  
► **X3DH**: protocollo di negoziazione delle chiavi Extended Triple Diffie-Hellman.  
» Double Ratchet: algoritmo utilizzato da due parti per lo scambio di messaggi basato su una chiave segreta condivisa.  
» Sesame: gestisce le sessioni crittografate in ambiente asincrono e multi-device.

# Signal Protocol

## Il protocollo

Le specifiche di riferimento sono infatti: [sig]

- ▶ **X3DH**: protocollo di negoziazione delle chiavi Extended Triple Diffie-Hellman.
- ▶ **Double Ratchet**: algoritmo utilizzato da due parti per lo scambio di messaggi basato su una chiave segreta condivisa.
- ▶ **Sesame**: gestisce le sessioni crittografate in ambiente asincrono e multi-device.



- Double Ratchet: le due parti derivano nuove chiavi per ogni messaggio in modo tale che chiavi usate in precedenza non possano essere ricavate dalle chiavi successive (grazie alla non invertibilità della funzione ratchet)
- X3DH: stabilisce una chiave segreta condivisa da due parti che si autenticano a vicenda basandosi su chiavi pubbliche. X3DH fornisce *forward secrecy* e *cryptographic deniability*

*Cryptographic deniability*: l'esistenza di un file cifrato o di un messaggio è rinnegabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.

Signal Protocol  
Il protocollo

Le specifiche di riferimento sono infatti: [sig]

- ▶ **X3DH**: protocollo di negoziazione delle chiavi Extended Triple Diffie-Hellman.
- ▶ **Double Ratchet**: algoritmo utilizzato da due parti per lo scambio di messaggi basato su una chiave segreta condivisa.
- ▶ **Sesame**: gestisce le sessioni crittografate in ambiente asincrono e multi-device.



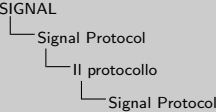
# Signal Protocol

## Il protocollo

Le specifiche di riferimento sono infatti: [sig]

- ▶ **X3DH**: protocollo di negoziazione delle chiavi Extended Triple Diffie-Hellman.
- ▶ **Double Ratchet**: algoritmo utilizzato da due parti per lo scambio di messaggi basato su una chiave segreta condivisa.
- ▶ **Sesame**: gestisce le sessioni crittografate in ambiente asincrono e multi-device.

2022-04-25



- Double Ratchet: le due parti derivano nuove chiavi per ogni messaggio in modo tale che chiavi usate in precedenza non possano essere ricavate dalle chiavi successive (grazie alla non invertibilità della funzione ratchet)
- X3DH: stabilisce una chiave segreta condivisa da due parti che si autenticano a vicenda basandosi su chiavi pubbliche. X3DH fornisce *forward secrecy* e *cryptographic deniability*

*Cryptographic deniability*: l'esistenza di un file cifrato o di un messaggio è rinnegabile, nel senso che un altro utente non può dimostrare che i dati in *plaintext* esistono. Gli utenti possono negare che dei dati siano cifrati o anche negare di essere in grado di decifrarli, indipendentemente dal fatto che ciò sia vero o meno.

Signal Protocol  
Il protocollo

Le specifiche di riferimento sono infatti: [sig]

- ▶ **X3DH**: protocollo di negoziazione delle chiavi Extended Triple Diffie-Hellman.
- ▶ **Double Ratchet**: algoritmo utilizzato da due parti per lo scambio di messaggi basato su una chiave segreta condivisa.
- ▶ **Sesame**: gestisce le sessioni crittografate in ambiente asincrono e multi-device.

# Signal Protocol

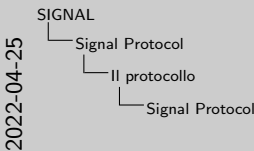
## Il protocollo: fasi di funzionamento [VD19]

► **KEY REGISTRATION:** invio di numerose chiavi pubbliche al server per consentire di iniziare una conversazione mentre l'altro utente non è online

► **KEY AGREEMENT:** Alice riceve le chiavi pubbliche di Bob dal server e le usa, insieme alle proprie chiavi private, per generare una chiave segreta condivisa. Invia a Bob un messaggio criptato con questa chiave. Bob, ricevutolo, recupera le chiavi pubbliche di Alice dal server e calcola la stessa chiave segreta condivisa. La negoziazione delle chiavi avviene tramite X3DH

► **CONVERSATION:** Alice e Bob possiedono la chiave segreta condivisa e possono conversare.

- DH ratchet phase
- Symmetric ratchet phase



- KEY REGISTRATION: se Alice vuole iniziare una conversazione con Bob può chiedere al server le sue chiavi pubbliche
- KEY AGREEMENT
- CONVERSATION

N.B. Symmetric ratchet phase e DH ratchet phase verranno meglio analizzati più avanti nel parlare dell'algorithm Double Ratchet. Ad ogni modo, a

livello generale, la loro combinazione permette di generare una nuova chiave.

Signal Protocol

Il protocollo: fasi di funzionamento [VD19]

- **KEY REGISTRATION:** invio di numerose chiavi pubbliche al server per consentire di iniziare una conversazione mentre l'altro utente non è online
- **KEY AGREEMENT:** Alice riceve le chiavi pubbliche di Bob dal server e le usa, insieme alle proprie chiavi private, per generare una chiave segreta condivisa. Invia a Bob un messaggio criptato con questa chiave. Bob, ricevutolo, recupera le chiavi pubbliche di Alice dal server e calcola la stessa chiave segreta condivisa. La negoziazione delle chiavi avviene tramite X3DH
- **CONVERSATION:** Alice e Bob possiedono la chiave segreta condivisa e possono conversare.
  - DH ratchet phase
  - Symmetric ratchet phase

# Signal Protocol

Il protocollo: fasi di funzionamento [VD19]



- KEY REGISTRATION: invio di numerose chiavi pubbliche al server per consentire di iniziare una conversazione mentre l'altro utente non è online
- KEY AGREEMENT: Alice riceve le chiavi pubbliche di Bob dal server e le usa, insieme alle proprie chiavi private, per generare una chiave segreta condivisa. Invia a Bob un messaggio criptato con questa chiave. Bob, ricevutolo, recupera le chiavi pubbliche di Alice dal server e calcola la stessa chiave segreta condivisa. La negoziazione delle chiavi avviene tramite X3DH
- CONVERSATION: Alice e Bob possiedono la chiave segreta condivisa e possono conversare.
  - DH ratchet phase
  - Symmetric ratchet phase

- KEY REGISTRATION: se Alice vuole iniziare una conversazione con Bob può chiedere al server le sue chiavi pubbliche
- KEY AGREEMENT
- CONVERSATION

N.B. Symmetric ratchet phase e DH ratchet phase verranno meglio analizzati più avanti nel parlare dell'algorithm Double Ratchet. Ad ogni modo, a

livello generale, la loro combinazione permette di generare una nuova chiave.

# Signal Protocol

Il protocollo: fasi di funzionamento [VD19]

## 1 Sommario

## 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

## 3 Crittografia End-to-End

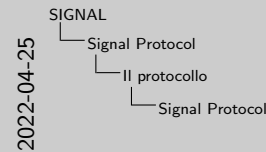
Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia

- ▶ **KEY REGISTRATION:** invio di numerose chiavi pubbliche al server per consentire di iniziare una conversazione mentre l'altro utente non è online
- ▶ **KEY AGREEMENT:** Alice riceve le chiavi pubbliche di Bob dal server e le usa, insieme alle proprie chiavi private, per generare una chiave segreta condivisa. Invia a Bob un messaggio criptato con questa chiave. Bob, ricevutolo, recupera le chiavi pubbliche di Alice dal server e calcola la stessa chiave segreta condivisa. La negoziazione delle chiavi avviene tramite X3DH
- ▶ **CONVERSATION:** Alice e Bob possiedono la chiave segreta condivisa e possono conversare.
  - ▶ DH ratchet phase
  - ▶ Symmetric ratchet phase



Signal Protocol  
Il protocollo: fasi di funzionamento [VD19]

- ▶ **KEY REGISTRATION:** invio di numerose chiavi pubbliche al server per consentire di iniziare una conversazione mentre l'altro utente non è online
- ▶ **KEY AGREEMENT:** Alice riceve le chiavi pubbliche di Bob dal server e le usa, insieme alle proprie chiavi private, per generare una chiave segreta condivisa. Invia a Bob un messaggio criptato con questa chiave. Bob, ricevutolo, recupera le chiavi pubbliche di Alice dal server e calcola la stessa chiave segreta condivisa. La negoziazione delle chiavi avviene tramite X3DH
- ▶ **CONVERSATION:** Alice e Bob possiedono la chiave segreta condivisa e possono conversare.
  - ▶ DH ratchet phase
  - ▶ Symmetric ratchet phase

- **KEY REGISTRATION:** se Alice vuole iniziare una conversazione con Bob può chiedere al server le sue chiavi pubbliche
- **KEY AGREEMENT**
- **CONVERSATION**

N.B. Symmetric ratchet phase e DH ratchet phase verranno meglio analizzati più avanti nel parlare dell'algoritmo Double Ratchet. Ad ogni modo, a

livello generale, la loro combinazione permette di generare una nuova chiave.

# Signal Protocol

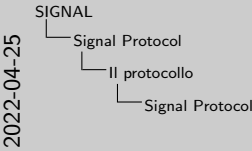
## Il protocollo: fasi di funzionamento

### Symmetric ratchet phase

Derivazione di una nuova chiave dalla chiave segreta condivisa.

Se Alice invia più messaggi a Bob senza ricevere risposta ogni messaggio sarà criptato con una nuova chiave calcolata in funzione della precedente.

In questo modo solo Alice e Bob possono calcolarla (escludendo casi in cui la chiave sia stata diffusa)



Signal Protocol  
Il protocollo: fasi di funzionamento

#### Symmetric ratchet phase

Derivazione di una nuova chiave dalla chiave segreta condivisa.

Se Alice invia più messaggi a Bob senza ricevere risposta ogni messaggio sarà criptato con una nuova chiave calcolata in funzione della precedente.

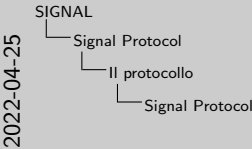
In questo modo solo Alice e Bob possono calcolarla (escludendo casi in cui la chiave sia stata diffusa)

# Signal Protocol

## Il protocollo: fasi di funzionamento

### Symmetric ratchet phase

Derivazione di una nuova chiave dalla chiave segreta condivisa.  
Se Alice invia più messaggi a Bob senza ricevere risposta ogni messaggio sarà criptato con una nuova chiave calcolata in funzione della precedente.  
In questo modo solo Alice e Bob possono calcolarla (escludendo casi in cui la chiave sia stata diffusa)



Signal Protocol  
Il protocollo: fasi di funzionamento

#### Symmetric ratchet phase

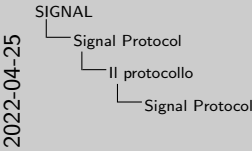
Derivazione di una nuova chiave dalla chiave segreta condivisa.  
Se Alice invia più messaggi a Bob senza ricevere risposta ogni messaggio sarà criptato con una nuova chiave calcolata in funzione della precedente.  
In questo modo solo Alice e Bob possono calcolarla (escludendo casi in cui la chiave sia stata diffusa)

# Signal Protocol

## Il protocollo: fasi di funzionamento

### Symmetric ratchet phase

Derivazione di una nuova chiave dalla chiave segreta condivisa.  
Se Alice invia più messaggi a Bob senza ricevere risposta ogni messaggio sarà criptato con una nuova chiave calcolata in funzione della precedente.  
In questo modo solo Alice e Bob possono calcolarla (escludendo casi in cui la chiave sia stata diffusa)



Signal Protocol  
Il protocollo: fasi di funzionamento

#### Symmetric ratchet phase

Derivazione di una nuova chiave dalla chiave segreta condivisa.  
Se Alice invia più messaggi a Bob senza ricevere risposta ogni messaggio sarà criptato con una nuova chiave calcolata in funzione della precedente.  
In questo modo solo Alice e Bob possono calcolarla (escludendo casi in cui la chiave sia stata diffusa)

1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
**Il protocollo**  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

5 Bibliografia

# Signal Protocol

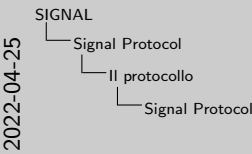
## Il protocollo: fasi di funzionamento

### Diffie–Hellman ratchet phase

Generazione di una nuova chiave segreta condivisa.

Se Bob invia un nuovo messaggio ad Alice genera una nuova coppia di chiavi effimere. Bob usa questa chiave per calcolarne una nuova condivisa, inviando poi la propria chiave effimera ad Alice per farle calcolare la chiave condivisa.

La chiave così calcolata verrà usata in una nuova *symmetric ratchet phase* per generare nuove chiavi per i messaggi.



Signal Protocol

Il protocollo: fasi di funzionamento

Diffie–Hellman ratchet phase

Generazione di una nuova chiave segreta condivisa.  
Se Bob invia un nuovo messaggio ad Alice genera una nuova coppia di chiavi effimere. Bob usa questa chiave per calcolarne una nuova condivisa, inviando poi la propria chiave effimera ad Alice per farle calcolare la chiave condivisa.  
La chiave così calcolata verrà usata in una nuova *symmetric ratchet phase* per generare nuove chiavi per i messaggi.



1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
**Il protocollo**  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

5 Bibliografia

# Signal Protocol

## Il protocollo: fasi di funzionamento

### Diffie–Hellman ratchet phase

Generazione di una nuova chiave segreta condivisa.  
Se Bob invia un nuovo messaggio ad Alice genera una nuova coppia di chiavi effimere. Bob usa questa chiave per calcolarne una nuova condivisa, inviando poi la propria chiave effimera ad Alice per farle calcolare la chiave condivisa.  
La chiave così calcolata verrà usata in una nuova *symmetric ratchet phase* per generare nuove chiavi per i messaggi.

2022-04-25

SIGNAL

Signal Protocol

Il protocollo

Signal Protocol

Signal Protocol

Il protocollo: fasi di funzionamento

Diffie–Hellman ratchet phase

Generazione di una nuova chiave segreta condivisa.  
Se Bob invia un nuovo messaggio ad Alice genera una nuova coppia di chiavi effimere. Bob usa questa chiave per calcolarne una nuova condivisa, inviando poi la propria chiave effimera ad Alice per farle calcolare la chiave condivisa.  
La chiave così calcolata verrà usata in una nuova *symmetric ratchet phase* per generare nuove chiavi per i messaggi.

# Signal Protocol

## Il protocollo: fasi di funzionamento

### Diffie–Hellman ratchet phase

Generazione di una nuova chiave segreta condivisa.

Se Bob invia un nuovo messaggio ad Alice genera una nuova coppia di chiavi effimere. Bob usa questa chiave per calcolarne una nuova condivisa, inviando poi la propria chiave effimera ad Alice per farle calcolare la chiave condivisa.

La chiave così calcolata verrà usata in una nuova *symmetric ratchet phase* per generare nuove chiavi per i messaggi.

2022-04-25

SIGNAL

Signal Protocol

Il protocollo

Signal Protocol

Signal Protocol

Il protocollo: fasi di funzionamento

Diffie–Hellman ratchet phase

Generazione di una nuova chiave segreta condivisa.  
Se Bob invia un nuovo messaggio ad Alice genera una nuova coppia di chiavi effimere. Bob usa questa chiave per calcolarne una nuova condivisa, inviando poi la propria chiave effimera ad Alice per farle calcolare la chiave condivisa.  
La chiave così calcolata verrà usata in una nuova *symmetric ratchet phase* per generare nuove chiavi per i messaggi.

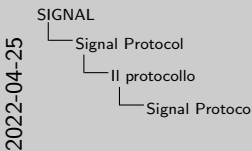
# Signal Protocol

Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- ▶ Identity key: chiave pubblica
- ▶ Ephemereal key: chiave utilizzabile una sola volta
- ▶ Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- ▶ One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- ▶ Signed pre-key: pre-key firmata con l'esponente dell'Identity key



STANDARD DIFFIE-HELLMAN: Alice e Bob generano ognuno una chiave pubblica  $pk$  basata su un generatore comune  $g$  modulo  $m$  e le proprie chiavi private (*secret keys*)  $sk$ .

Dopodiché scambiano le chiavi pubbliche attraverso un canale (potenzialmente non sicuro) e da esse possono derivare una chiave segreta condivisa  $ssk$ . [DH76], [JD16]

Signal Protocol

Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- ▶ Identity key: chiave pubblica
- ▶ Ephemereal key: chiave utilizzabile una sola volta
- ▶ Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- ▶ One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- ▶ Signed pre-key: pre-key firmata con l'esponente dell'Identity key

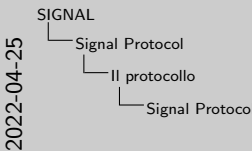
# Signal Protocol

Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- Identity key: chiave pubblica
- Ephemereal key: chiave utilizzabile una sola volta
- Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- Signed pre-key: pre-key firmata con l'esponente dell'Identity key



STANDARD DIFFIE-HELLMAN: Alice e Bob generano ognuno una chiave pubblica  $pk$  basata su un generatore comune  $g$  modulo  $m$  e le proprie chiavi private (*secret keys*)  $sk$ .

Dopodiché scambiano le chiavi pubbliche attraverso un canale (potenzialmente non sicuro) e da esse possono derivare una chiave segreta condivisa  $ssk$ . [DH76], [JD16]

Signal Protocol

Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- Identity key: chiave pubblica
- Ephemereal key: chiave utilizzabile una sola volta
- Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- Signed pre-key: pre-key firmata con l'esponente dell'Identity key

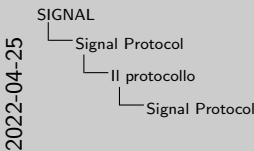
# Signal Protocol

Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- Identity key: chiave pubblica
- Ephemereal key: chiave utilizzabile una sola volta
- Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- Signed pre-key: pre-key firmata con l'esponente dell'Identity key



STANDARD DIFFIE-HELLMAN: Alice e Bob generano ognuno una chiave pubblica  $pk$  basata su un generatore comune  $g$  modulo  $m$  e le proprie chiavi private (*secret keys*)  $sk$ .

Dopodiché scambiano le chiavi pubbliche attraverso un canale (potenzialmente non sicuro) e da esse possono derivare una chiave segreta condivisa  $ssk$ . [DH76], [JD16]

Signal Protocol

Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- Identity key: chiave pubblica
- Ephemereal key: chiave utilizzabile una sola volta
- Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- Signed pre-key: pre-key firmata con l'esponente dell'Identity key

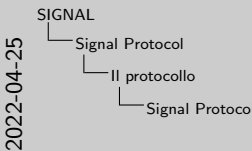
# Signal Protocol

Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- Identity key: chiave pubblica
- Ephemereal key: chiave utilizzabile una sola volta
- Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- Signed pre-key: pre-key firmata con l'esponente dell'Identity key



STANDARD DIFFIE-HELLMAN: Alice e Bob generano ognuno una chiave pubblica  $pk$  basata su un generatore comune  $g$  modulo  $m$  e le proprie chiavi private (*secret keys*)  $sk$ .

Dopodiché scambiano le chiavi pubbliche attraverso un canale (potenzialmente non sicuro) e da esse possono derivare una chiave segreta condivisa  $ssk$ . [DH76], [JD16]

Signal Protocol

Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- Identity key: chiave pubblica
- Ephemereal key: chiave utilizzabile una sola volta
- Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- Signed pre-key: pre-key firmata con l'esponente dell'Identity key

# Signal Protocol

Il protocollo: X3DH

## 1 Sommario

## 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

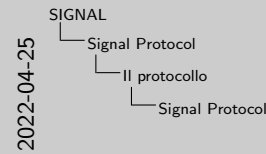
Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- ▶ Identity key: chiave pubblica
- ▶ Ephemereal key: chiave utilizzabile una sola volta
- ▶ Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- ▶ One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- ▶ Signed pre-key: pre-key firmata con l'esponente dell'Identity key



STANDARD DIFFIE-HELLMAN: Alice e Bob generano ognuno una chiave pubblica  $pk$  basata su un generatore comune  $g$  modulo  $m$  e le proprie chiavi private (*secret keys*)  $sk$ .

Dopodiché scambiano le chiavi pubbliche attraverso un canale (potenzialmente non sicuro) e da esse possono derivare una chiave segreta condivisa  $ssk$ . [DH76], [JD16]

Signal Protocol  
Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- ▶ Identity key: chiave pubblica
- ▶ Ephemereal key: chiave utilizzabile una sola volta
- ▶ Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- ▶ One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire

▶ Signed pre-key: pre-key firmata con l'esponente dell'Identity key

# Signal Protocol

Il protocollo: X3DH

## 1 Sommario

## 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

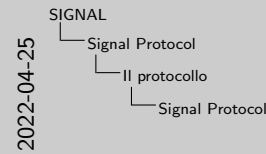
Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- Identity key: chiave pubblica
- Ephemereal key: chiave utilizzabile una sola volta
- Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- Signed pre-key: pre-key firmata con l'esponente dell'Identity key



STANDARD DIFFIE-HELLMAN: Alice e Bob generano ognuno una chiave pubblica  $pk$  basata su un generatore comune  $g$  modulo  $m$  e le proprie chiavi private (*secret keys*)  $sk$ .

Dopodiché scambiano le chiavi pubbliche attraverso un canale (potenzialmente non sicuro) e da esse possono derivare una chiave segreta condivisa  $ssk$ . [DH76], [JD16]

Signal Protocol  
Il protocollo: X3DH

X3DH è stato sviluppato da OWS per supportare lo scambio asincrono delle chiavi. [MP16b]

Definizioni:

- Identity key: chiave pubblica
- Ephemereal key: chiave utilizzabile una sola volta
- Pre-keys: chiavi condivise col server prima dell'attivazione del protocollo
- One-time pre-keys: insiemi di pre-keys condivisi col server prima dell'attivazione del protocollo. Il server condivide una chiave ogni volta che un utente vuole iniziare una conversazione e ne richiede un nuovo insieme quando stanno per finire
- Signed pre-key: pre-key firmata con l'esponente dell'Identity key



# Signal Protocol

## Il protocollo: X3DH

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

[VD19]

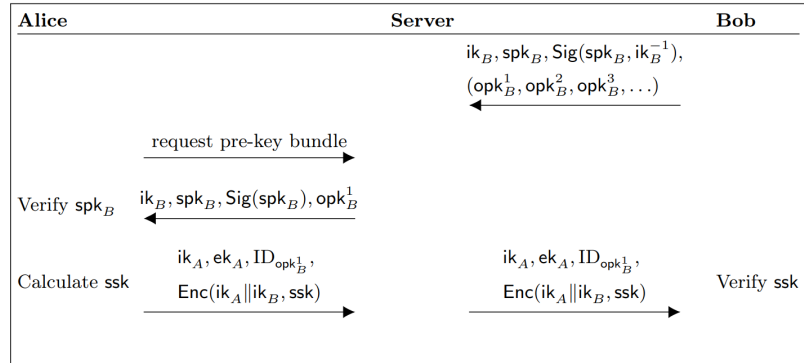


Figure: Funzionamento di X3DH semplificato

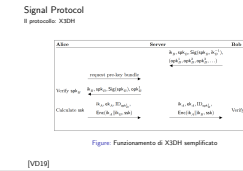
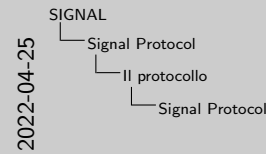


Figure: Funzionamento di X3DH semplificato

[VD19]

Perché il protocollo possa funzionare offline ogni utente deve inviare le proprie chiavi pubbliche al server, inviando cioè  $ik, spk, Sig(spk, ik^{-1})$  e  $(opk^1, opk^2, opk^3, \dots)$ .

La  $ik$  va inviata una sola volta, la  $spk$  va rinnovata periodicamente.

Se Alice vuole iniziare una conversazione richiede al server  $ik_B, spk_B, Sig(spk_B, ik_B^{-1})$  e una delle one-time pre-keys  $opk_B^x$  di Bob. Il server poi elimina  $opk_B^x$ . Una volta finite le  $opk_B$  ad Alice verranno inviate solo le altre chiavi senza  $opk_B$ .

Ricevute le chiavi Alice verifica la firma di  $spk_B$  e se va a buon fine genera una coppia di chiavi effimere; poi calcola la  $ssk$  usando una Key Derivation Function (KDF). Alla fine cancella  $ek_A^{-1}$  e tutti i valori  $k_i$  generati.

N.B.  $DH(x, y)$  è una funzione DH su curva ellittica che calcola una  $ssk$  basandosi su due chiavi, mentre  $KDF(x)$  è una funzione basata su RFC5869 [KE10].

Alice invia un messaggio iniziale a Bob contenente  $ik_A, ek_A, ID_{opk_B^x}$  (per fargli sapere quale  $opk_B^x$  ha usato) e  $Enc(ik_A || ik_B, ssk)$ .

Bob riceve il messaggio e calcola la  $ssk$  nello stesso modo di Alice. Bob decrittografa il messaggio inviato da Alice e controlla se il valore di  $ssk$  è corretto e in questo caso cancella la  $opk$  utilizzata.

Alice e Bob possono ora riutilizzare la stessa  $ssk$  per messaggi futuri oppure usare chiavi da essa derivate. [VD19]

# Signal Protocol

## Il protocollo: Double Ratchet

Algoritmo utilizzato per il proseguimento della conversazione, dopo averla iniziata tramite X3DH.

[MP16a], [VD19]

A differenza di X3DH usa una catena KDF, come mostrato in figura 2.

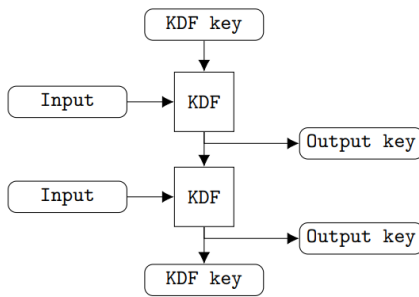
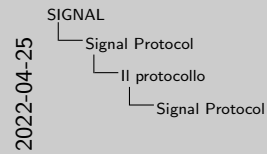
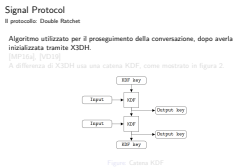


Figure: Catena KDF



2022-04-25



La catena KDF usa l'output di una KDF come input per un'altra applicazione della KDF. Ogni *output key* può essere utilizzata per cifrare un messaggio. Tale catena garantisce:

- Resilienza: l'output appare randomico
- Forward secrecy: garantita dalla non-invertibilità della KDF
- Future secrecy: garantita se l'input della KDF  $i + 1$  non è il solo output della KDF  $i$ . Per garantire ciò è necessario usare un **DH ratchet**

### 1 Sommario

### 2 Applicazione Signal

Storia dell'Applicazione  
L'Applicazione e il Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

# Signal Protocol

## Il protocollo: Double Ratchet

Algoritmo utilizzato per il proseguimento della conversazione, dopo averla iniziata tramite X3DH.

[MP16a], [VD19]

A differenza di X3DH usa una catena KDF, come mostrato in figura 2.

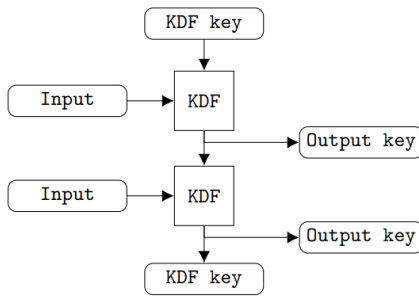
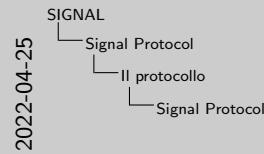


Figure: Catena KDF



2022-04-25

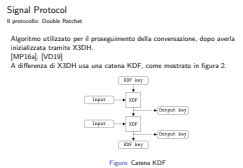


Figure: Catena KDF

La catena KDF usa l'output di una KDF come input per un'altra applicazione della KDF. Ogni *output key* può essere utilizzata per cifrare un messaggio. Tale catena garantisce:

- Resilienza: l'output appare randomico
- Forward secrecy: garantita dalla non-invertibilità della KDF
- Future secrecy: garantita se l'input della KDF  $i + 1$  non è il solo output della KDF  $i$ . Per garantire ciò è necessario usare un **DH ratchet**

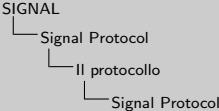
# Signal Protocol

## Il protocollo: Double Ratchet

Combinando una catena KDF con un DH ratchet otteniamo un algoritmo Double ratchet che garantisce sia *forward secrecy* che *future secrecy*.

DH ratchet infatti modifica gli input delle KDF in modo tale che, se anche una chiave venisse recuperata da terzi, si sia in grado di ristabilire la segretezza dall'applicazione successiva di una KDF.

2022-04-25



Signal Protocol  
Il protocollo: Double Ratchet

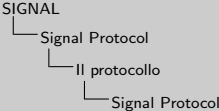
Combinando una catena KDF con un DH ratchet otteniamo un algoritmo Double ratchet che garantisce sia *forward secrecy* che *future secrecy*.  
DH ratchet infatti modifica gli input delle KDF in modo tale che, se anche una chiave venisse recuperata da terzi, si sia in grado di ristabilire la segretezza dall'applicazione successiva di una KDF.

# Signal Protocol

Il protocollo: Double Ratchet

Combinando una catena KDF con un DH ratchet otteniamo un algoritmo Double ratchet che garantisce sia *forward secrecy* che *future secrecy*.  
DH ratchet infatti modifica gli input delle KDF in modo tale che, se anche una chiave venisse recuperata da terzi, si sia in grado di ristabilire la segretezza dall'applicazione successiva di una KDF.

2022-04-25



Signal Protocol  
Il protocollo: Double Ratchet

Combinando una catena KDF con un DH ratchet otteniamo un algoritmo Double ratchet che garantisce sia *forward secrecy* che *future secrecy*.  
DH ratchet infatti modifica gli input delle KDF in modo tale che, se anche una chiave venisse recuperata da terzi, si sia in grado di ristabilire la segretezza dall'applicazione successiva di una KDF.

# Signal Protocol

## Il protocollo: Double Ratchet

Sia Alice che Bob hanno tre catene KDF da utilizzare:

- ▶ DH ratchet
- ▶ Sending ratchet
- ▶ Receiving ratchet

Ogni volta che Alice vuole inviare un messaggio a Bob aggiornerà la catena *sending ratchet* producendo una nuova chiave di output e invierà a Bob il proprio messaggio cifrato con essa.

2022-04-25

SIGNAL

Signal Protocol

Il protocollo

Signal Protocol

Signal Protocol

Il protocollo: Double Ratchet

Sia Alice che Bob hanno tre catene KDF da utilizzare:

▶ DH ratchet

▶ Sending ratchet

▶ Receiving ratchet

Ogni volta che Alice vuole inviare un messaggio a Bob aggiornerà la catena *sending ratchet* producendo una nuova chiave di output e invierà a Bob il proprio messaggio cifrato con essa.

# Signal Protocol

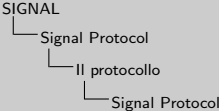
## Il protocollo: Double Ratchet

Sia Alice che Bob hanno tre catene KDF da utilizzare:

- ▶ DH ratchet
- ▶ Sending ratchet
- ▶ Receiving ratchet

Ogni volta che Alice vuole inviare un messaggio a Bob aggiornerà la catena *sending ratchet* producendo una nuova chiave di output e invierà a Bob il proprio messaggio cifrato con essa.

2022-04-25



Signal Protocol

Il protocollo: Double Ratchet

Sia Alice che Bob hanno tre catene KDF da utilizzare:

- ▶ DH ratchet
- ▶ Sending ratchet
- ▶ Receiving ratchet

Ogni volta che Alice vuole inviare un messaggio a Bob aggiornerà la catena *sending ratchet* producendo una nuova chiave di output e invierà a Bob il proprio messaggio cifrato con essa.

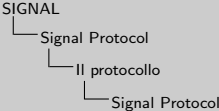
# Signal Protocol

Il protocollo: Double Ratchet

La catena di invio di Alice deve essere sincronizzata con quella di ricezione di Bob e viceversa e devono iniziare nella stessa posizione (caratteristica garantita da X3DH applicato prima del Double ratchet).

In queste condizioni tuttavia non è ancora garantita la *future secrecy*

2022-04-25



Signal Protocol  
Il protocollo: Double Ratchet

La catena di invio di Alice deve essere sincronizzata con quella di ricezione di Bob e viceversa e devono iniziare nella stessa posizione (caratteristica garantita da X3DH applicato prima del Double ratchet).

In queste condizioni tuttavia non è ancora garantita la future secrecy

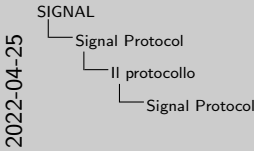


# Signal Protocol

## Il protocollo: Double Ratchet

La catena di invio di Alice deve essere sincronizzata con quella di ricezione di Bob e viceversa e devono iniziare nella stessa posizione (caratteristica garantita da X3DH applicato prima del Double ratchet).

In queste condizioni tuttavia non è ancora garantita la *future secrecy*



Signal Protocol

Il protocollo: Double Ratchet

La catena di invio di Alice deve essere sincronizzata con quella di ricezione di Bob e viceversa e devono iniziare nella stessa posizione (caratteristica garantita da X3DH applicato prima del Double ratchet).

In queste condizioni tuttavia non è ancora garantita la *future secrecy*

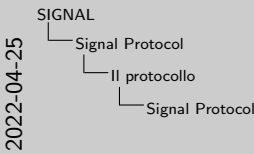
# Signal Protocol

Il protocollo: Double Ratchet

Per garantire *future secrecy* Bob invierà come parte di uno dei suoi messaggi una nuova chiave pubblica Diffie-Hellman.

Alice userà questa chiave per far avanzare la catena DH-ratchet, imponendo così il reset delle catene di ricezione e invio. In parallelo anche la catena DH-ratchet di Bob verrà aggiornata.

Un comportamento analogo verrà applicato da Bob sulle proprie catene. Questo scambio di chiavi DH può avvenire ogniqualvolta necessario ma normalmente avviene a ogni messaggio scambiato. [Pou]



Questo sistema di catene garantisce dunque:

- Forward security: grazie alle catene di invio e ricezione, caratterizzate da funzioni non invertibili, non è possibile retrocedere ai messaggi inviati in precedenza
- Future secrecy: grazie alla catena DH-ratchet se anche si riesce a intercettare un messaggio e decrittarlo si resettano le catene di invio e ricezione, rendendo impossibile generare in anticipo le chiavi che verranno utilizzate in futuro
- Funzionamento asincrono: se Alice invia dieci messaggi a Bob e lui non risponde man mano, egli comunque sarà in grado di ricostruire la sequenza di chiavi utilizzata da Alice e dunque leggere i messaggi
- In ogni messaggio viene indicato il numero di messaggi già inviati sulla stessa catena quindi se un messaggio va perso si può o aspettarne l'arrivo o far avanzare la catena di tante posizioni quanti messaggi sono andati persi.

N.B. Le chiavi vengono eliminate non appena utilizzate per decifrare un messaggio, se delle chiavi non vengono utilizzate vengono conservate finché non arriverà il messaggio corrispondente.

Signal Protocol

Il protocollo: Double Ratchet

Per garantire future secrecy Bob invierà come parte di uno dei suoi messaggi una nuova chiave pubblica Diffie-Hellman.

Alice userà questa chiave per far avanzare la catena DH-ratchet, imponendo così il reset delle catene di ricezione e invio. In parallelo anche la catena DH-ratchet di Bob verrà aggiornata.

Un comportamento analogo verrà applicato da Bob sulle proprie catene. Questo scambio di chiavi DH può avvenire ogniqualvolta necessario ma normalmente avviene a ogni messaggio scambiato. [Pou]

1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
**Il protocollo**  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

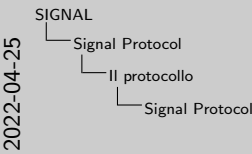
5 Bibliografia

# Signal Protocol

## Il protocollo: Double Ratchet

Per garantire *future secrecy* Bob invierà come parte di uno dei suoi messaggi una nuova chiave pubblica Diffie-Hellman. Alice userà questa chiave per far avanzare la catena DH-ratchet, imponendo così il reset delle catene di ricezione e invio. In parallelo anche la catena DH-ratchet di Bob verrà aggiornata.

Un comportamento analogo verrà applicato da Bob sulle proprie catene. Questo scambio di chiavi DH può avvenire ogniqualvolta necessario ma normalmente avviene a ogni messaggio scambiato. [Pou]



Questo sistema di catene garantisce dunque:

- Forward security: grazie alle catene di invio e ricezione, caratterizzate da funzioni non invertibili, non è possibile retrocedere ai messaggi inviati in precedenza
- Future secrecy: grazie alla catena DH-ratchet se anche si riesce a intercettare un messaggio e decrittarlo si resettano le catene di invio e ricezione, rendendo impossibile generare in anticipo le chiavi che verranno utilizzate in futuro
- Funzionamento asincrono: se Alice invia dieci messaggi a Bob e lui non risponde man mano, egli comunque sarà in grado di ricostruire la sequenza di chiavi utilizzata da Alice e dunque leggere i messaggi
- In ogni messaggio viene indicato il numero di messaggi già inviati sulla stessa catena quindi se un messaggio va perso si può o aspettarne l'arrivo o far avanzare la catena di tante posizioni quanti messaggi sono andati persi.

N.B. Le chiavi vengono eliminate non appena utilizzate per decifrare un messaggio, se delle chiavi non vengono utilizzate vengono conservate finché non arriverà il messaggio corrispondente.

Signal Protocol

Il protocollo: Double Ratchet

Per garantire future secrecy Bob invierà come parte di uno dei suoi messaggi una nuova chiave pubblica Diffie-Hellman. Alice userà questa chiave per far avanzare la catena DH-ratchet, imponendo così il reset delle catene di ricezione e invio. In parallelo anche la catena DH-ratchet di Bob verrà aggiornata. Un comportamento analogo verrà applicato da Bob sulle proprie catene. Questo scambio di chiavi DH può avvenire ogniqualvolta necessario ma normalmente avviene a ogni messaggio scambiato. [Pou]

1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

5 Bibliografia

# Signal Protocol

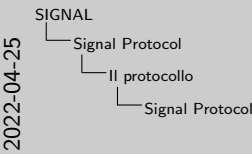
## Il protocollo: Double Ratchet

Per garantire *future secrecy* Bob invierà come parte di uno dei suoi messaggi una nuova chiave pubblica Diffie-Hellman.

Alice userà questa chiave per far avanzare la catena DH-ratchet, imponendo così il reset delle catene di ricezione e invio. In parallelo anche la catena DH-ratchet di Bob verrà aggiornata.

Un comportamento analogo verrà applicato da Bob sulle proprie catene.

Questo scambio di chiavi DH può avvenire ogniqualvolta necessario ma normalmente avviene a ogni messaggio scambiato. [Pou]



Questo sistema di catene garantisce dunque:

- Forward security: grazie alle catene di invio e ricezione, caratterizzate da funzioni non invertibili, non è possibile retrocedere ai messaggi inviati in precedenza
- Future secrecy: grazie alla catena DH-ratchet se anche si riesce a intercettare un messaggio e decrittarlo si resettano le catene di invio e ricezione, rendendo impossibile generare in anticipo le chiavi che verranno utilizzate in futuro
- Funzionamento asincrono: se Alice invia dieci messaggi a Bob e lui non risponde man mano, egli comunque sarà in grado di ricostruire la sequenza di chiavi utilizzata da Alice e dunque leggere i messaggi
- In ogni messaggio viene indicato il numero di messaggi già inviati sulla stessa catena quindi se un messaggio va perso si può o aspettarne l'arrivo o far avanzare la catena di tante posizioni quanti messaggi sono andati persi.

N.B. Le chiavi vengono eliminate non appena utilizzate per decifrare un messaggio, se delle chiavi non vengono utilizzate vengono conservate finché non arriverà il messaggio corrispondente.

Signal Protocol

Il protocollo: Double Ratchet

Per garantire future secrecy Bob invierà come parte di uno dei suoi messaggi una nuova chiave pubblica Diffie-Hellman.  
Alice userà questa chiave per far avanzare la catena DH-ratchet, imponendo così il reset delle catene di ricezione e invio. In parallelo anche la catena DH-ratchet di Bob verrà aggiornata.  
Un comportamento analogo verrà applicato da Bob sulle proprie catene.  
Questo scambio di chiavi DH può avvenire ogniqualvolta necessario ma normalmente avviene a ogni messaggio scambiato. [Pou]

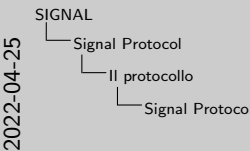
# Signal Protocol

Il protocollo: Double Ratchet

Per garantire *future secrecy* Bob invierà come parte di uno dei suoi messaggi una nuova chiave pubblica Diffie-Hellman.

Alice userà questa chiave per far avanzare la catena DH-ratchet, imponendo così il reset delle catene di ricezione e invio. In parallelo anche la catena DH-ratchet di Bob verrà aggiornata.

Un comportamento analogo verrà applicato da Bob sulle proprie catene. Questo scambio di chiavi DH può avvenire ogniqualvolta necessario ma normalmente avviene a ogni messaggio scambiato. [Pou]



Questo sistema di catene garantisce dunque:

- Forward security: grazie alle catene di invio e ricezione, caratterizzate da funzioni non invertibili, non è possibile retrocedere ai messaggi inviati in precedenza
- Future secrecy: grazie alla catena DH-ratchet se anche si riesce a intercettare un messaggio e decrittarlo si resettano le catene di invio e ricezione, rendendo impossibile generare in anticipo le chiavi che verranno utilizzate in futuro
- Funzionamento asincrono: se Alice invia dieci messaggi a Bob e lui non risponde man mano, egli comunque sarà in grado di ricostruire la sequenza di chiavi utilizzata da Alice e dunque leggere i messaggi
- In ogni messaggio viene indicato il numero di messaggi già inviati sulla stessa catena quindi se un messaggio va perso si può o aspettarne l'arrivo o far avanzare la catena di tante posizioni quanti messaggi sono andati persi.

N.B. Le chiavi vengono eliminate non appena utilizzate per decifrare un messaggio, se delle chiavi non vengono utilizzate vengono conservate finché non arriverà il messaggio corrispondente.

Signal Protocol  
Il protocollo: Double Ratchet

Per garantire future secrecy Bob invierà come parte di uno dei suoi messaggi una nuova chiave pubblica Diffie-Hellman.  
Alice userà questa chiave per far avanzare la catena DH-ratchet, imponendo così il reset delle catene di ricezione e invio. In parallelo anche la catena DH-ratchet di Bob verrà aggiornata.  
Un comportamento analogo verrà applicato da Bob sulle proprie catene. Questo scambio di chiavi DH può avvenire ogniqualvolta necessario ma normalmente avviene a ogni messaggio scambiato. [Pou]

# Signal Protocol

## Il protocollo: Double Ratchet

- 1 Sommario
- 2 Applicazione Signal
  - Storia dell'Applicazione
  - L'Applicazione e il Protocollo Signal
- 3 Crittografia End-to-End
  - Applicazioni
  - Problematiche
- 4 Signal Protocol
  - Proprietà
  - Il protocollo
  - Difetti di progettazione
  - Considerazioni
  - WhatsApp VS Signal
  - VS Telegram
- 5 Bibliografia

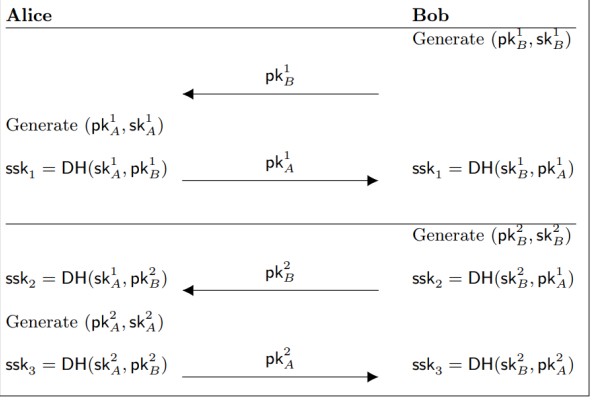
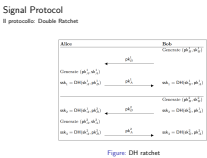
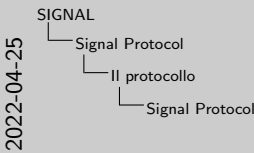


Figure: DH ratchet



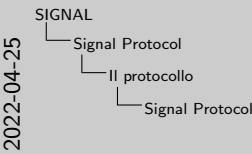
# Signal Protocol

## Il protocollo: Sesame

L’algoritmo **Sesame** gestisce la creazione, eliminazione e utilizzo delle sessioni di comunicazione.

Ogni dispositivo degli utenti deve tenere traccia di una sessione *attiva* per ogni altro dispositivo con cui sta comunicando e utilizzare quella sessione quando comunica con esso. Quando si riceve un messaggio su una sessione *inattiva*, essa diventa *attiva*.

In questo modo ogni dispositivo utilizza una sola sessione per ogni altro dispositivo remoto con cui comunica. [MP17]



Problemi che si possono generare:

- Alice e Bob possono avere ognuno più dispositivi quindi l'invio di un messaggio da parte di Alice richiede di inviarne una copia a tutti i dispositivi di Bob e a tutti i dispositivi di Alice per conoscenza.
- Alice e Bob possono aggiungere o rimuovere dispositivi, iniziando o terminando delle sessioni.
- Alice e Bob possono iniziare una nuova sessione allo stesso tempo, così che si ottengano due sessioni. Per un funzionamento ottimale del Double Ratchet Alice e Bob devono usare la stessa sessione, quindi devono concordare su quella da utilizzare.
- Alice può resettare lo stato della sessione sul suo dispositivo o impostarla tramite backup, richiedendo a Bob di concordare nuovamente sulla sessione da utilizzare.

Signal Protocol  
Il protocollo: Sesame

L'algoritmo **Sesame** gestisce la creazione, eliminazione e utilizzo delle sessioni di comunicazione.

Ogni dispositivo degli utenti deve tenere traccia di una sessione attiva per ogni altro dispositivo con cui sta comunicando e utilizzare quella sessione quando comunica con esso. Quando si riceve un messaggio su una sessione inattiva, essa diventa attiva.

In questo modo ogni dispositivo utilizza una sola sessione per ogni altro dispositivo remoto con cui comunica. [MP17]

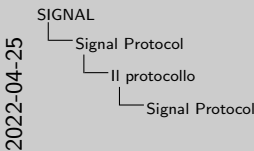
# Signal Protocol

## Il protocollo: Sesame

L’algoritmo **Sesame** gestisce la creazione, eliminazione e utilizzo delle sessioni di comunicazione.

Ogni dispositivo degli utenti deve tenere traccia di una sessione *attiva* per ogni altro dispositivo con cui sta comunicando e utilizzare quella sessione quando comunica con esso. Quando si riceve un messaggio su una sessione *inattiva*, essa diventa *attiva*.

In questo modo ogni dispositivo utilizza una sola sessione per ogni altro dispositivo remoto con cui comunica. [MP17]



Problemi che si possono generare:

- Alice e Bob possono avere ognuno più dispositivi quindi l'invio di un messaggio da parte di Alice richiede di inviarne una copia a tutti i dispositivi di Bob e a tutti i dispositivi di Alice per conoscenza.
- Alice e Bob possono aggiungere o rimuovere dispositivi, iniziando o terminando delle sessioni.
- Alice e Bob possono iniziare una nuova sessione allo stesso tempo, così che si ottengano due sessioni. Per un funzionamento ottimale del Double Ratchet Alice e Bob devono usare la stessa sessione, quindi devono concordare su quella da utilizzare.
- Alice può resettare lo stato della sessione sul suo dispositivo o impostarla tramite backup, richiedendo a Bob di concordare nuovamente sulla sessione da utilizzare.

Signal Protocol  
Il protocollo: Sesame

L'algoritmo **Sesame** gestisce la creazione, eliminazione e utilizzo delle sessioni di comunicazione. Ogni dispositivo degli utenti deve tenere traccia di una sessione attiva per ogni altro dispositivo con cui sta comunicando e utilizzare quella sessione quando comunica con esso. Quando si riceve un messaggio su una sessione inattiva, essa diventa attiva.

In questo modo ogni dispositivo utilizza una sola sessione per ogni altro dispositivo remoto con cui comunica. [MP17]



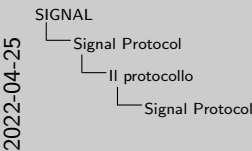
# Signal Protocol

Il protocollo: Sesame

L'algoritmo **Sesame** gestisce la creazione, eliminazione e utilizzo delle sessioni di comunicazione.

Ogni dispositivo degli utenti deve tenere traccia di una sessione *attiva* per ogni altro dispositivo con cui sta comunicando e utilizzare quella sessione quando comunica con esso. Quando si riceve un messaggio su una sessione *inattiva*, essa diventa *attiva*.

In questo modo ogni dispositivo utilizza una sola sessione per ogni altro dispositivo remoto con cui comunica. [MP17]



Problemi che si possono generare:

- Alice e Bob possono avere ognuno più dispositivi quindi l'invio di un messaggio da parte di Alice richiede di inviarne una copia a tutti i dispositivi di Bob e a tutti i dispositivi di Alice per conoscenza.
- Alice e Bob possono aggiungere o rimuovere dispositivi, iniziando o terminando delle sessioni.
- Alice e Bob possono iniziare una nuova sessione allo stesso tempo, così che si ottengano due sessioni. Per un funzionamento ottimale del Double Ratchet Alice e Bob devono usare la stessa sessione, quindi devono concordare su quella da utilizzare.
- Alice può resettare lo stato della sessione sul suo dispositivo o impostarla tramite backup, richiedendo a Bob di concordare nuovamente sulla sessione da utilizzare.

Signal Protocol  
Il protocollo: Sesame

L'algoritmo **Sesame** gestisce la creazione, eliminazione e utilizzo delle sessioni di comunicazione.  
Ogni dispositivo degli utenti deve tenere traccia di una sessione *attiva* per ogni altro dispositivo con cui sta comunicando e utilizzare quella sessione quando comunica con esso. Quando si riceve un messaggio su una sessione *inattiva*, essa diventa *attiva*.  
In questo modo ogni dispositivo utilizza una sola sessione per ogni altro dispositivo remoto con cui comunica. [MP17]

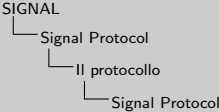
# Signal Protocol

## Il protocollo: Sesame

Sesame è stato progettato per l'uso attraverso sessioni Double Ratchet create attraverso scambio di chiavi X3DH.

- ▶ I dispositivi comunicano al server le proprie *one-time pre keys*, *signed pre keys* e *identity public key*.
- ▶ Il dispositivo mittente recupera dal server la *identity public key* del dispositivo destinatario, *signed pre keys* e una *one-time pre key* se disponibile.
- ▶ X3DH usa queste chiavi per creare sia una chiave segreta che apre una sessione Double Ratchet sia un messaggio iniziale X3DH.

2022-04-25



Signal Protocol  
Il protocollo: Sesame

Sesame è stato progettato per l'uso attraverso sessioni Double Ratchet create attraverso scambio di chiavi X3DH.

- ▶ I dispositivi comunicano al server le proprie *one-time pre keys*, *signed pre keys* e *identity public key*.
- ▶ Il dispositivo mittente recupera dal server la *identity public key* del dispositivo destinatario, *signed pre keys* e una *one-time pre key* se disponibile.
- ▶ X3DH usa queste chiavi per creare sia una chiave segreta che apre una sessione Double Ratchet sia un messaggio iniziale X3DH.

# Signal Protocol

## Il protocollo: Sesame

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

### 4 Signal Protocol

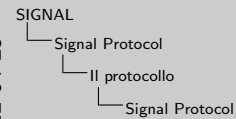
Proprietà  
**Il protocollo**  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

Sesame è stato progettato per l'uso attraverso sessioni Double Ratchet create attraverso scambio di chiavi X3DH.

- ▶ I dispositivi comunicano al server le proprie *one-time pre keys*, *signed pre keys* e *identity public key*.
- ▶ Il dispositivo mittente recupera dal server la *identity public key* del dispositivo destinatario, *signed pre keys* e una *one-time pre key* se disponibile.
- ▶ X3DH usa queste chiavi per creare sia una chiave segreta che apre una sessione Double Ratchet sia un messaggio iniziale X3DH.

2022-04-25



Signal Protocol  
Il protocollo: Sesame

Sesame è stato progettato per l'uso attraverso sessioni Double Ratchet create attraverso scambio di chiavi X3DH.

- ▶ I dispositivi comunicano al server le proprie *one-time pre keys*, *signed pre keys* e *identity public key*.
- ▶ Il dispositivo mittente recupera dal server la *identity public key* del dispositivo destinatario, *signed pre keys* e una *one-time pre key* se disponibile.

► X3DH usa queste chiavi per creare sia una chiave segreta che apre una sessione Double Ratchet sia un messaggio iniziale X3DH.

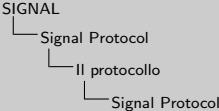
# Signal Protocol

## Il protocollo: Sesame

Sesame è stato progettato per l'uso attraverso sessioni Double Ratchet create attraverso scambio di chiavi X3DH.

- ▶ I dispositivi comunicano al server le proprie *one-time pre keys*, *signed pre keys* e *identity public key*.
- ▶ Il dispositivo mittente recupera dal server la *identity public key* del dispositivo destinatario, *signed pre keys* e una *one-time pre key* se disponibile.
- ▶ X3DH usa queste chiavi per creare sia una chiave segreta che apre una sessione Double Ratchet sia un messaggio iniziale X3DH.

2022-04-25



Signal Protocol  
Il protocollo: Sesame

Sesame è stato progettato per l'uso attraverso sessioni Double Ratchet create attraverso scambio di chiavi X3DH.

- ▶ I dispositivi comunicano al server le proprie *one-time pre keys*, *signed pre keys* e *identity public key*.
- ▶ Il dispositivo mittente recupera dal server la *identity public key* del dispositivo destinatario, *signed pre keys* e una *one-time pre key* se disponibile.
- ▶ X3DH usa queste chiavi per creare sia una chiave segreta che apre una sessione Double Ratchet sia un messaggio iniziale X3DH.

# Signal Protocol

## Il protocollo: Sesame

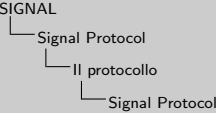
► Il messaggio iniziale X3DH è aggiunto a ogni messaggio di apertura di sessione, in modo che il destinatario lo usi per creare una sessione Double Ratchet corrispondente.

► Ricevuto il messaggio di conferma di apertura della sessione il mittente smette di inviare il messaggio iniziale X3DH.

► I dispositivi comunicano solo con Double Ratchet.

[MP17], [VD19]

2022-04-25



Signal Protocol

Il protocollo: Sesame

► Il messaggio iniziale X3DH è aggiunto a ogni messaggio di apertura di sessione, in modo che il destinatario lo usi per creare una sessione Double Ratchet corrispondente.

► Ricevuto il messaggio di conferma di apertura della sessione il mittente smette di inviare il messaggio iniziale X3DH.

► I dispositivi comunicano solo con Double Ratchet.

[MP17], [VD19]

# Signal Protocol

## Il protocollo: Sesame

### 1 Sommario

### 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

### 3 Crittografia End-to-End

Applicazioni  
Problematiche

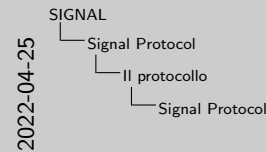
### 4 Signal Protocol

Proprietà  
**Il protocollo**  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

### 5 Bibliografia

- ▶ Il messaggio iniziale X3DH è aggiunto a ogni messaggio di apertura di sessione, in modo che il destinatario lo usi per creare una sessione Double Ratchet corrispondente.
- ▶ Ricevuto il messaggio di conferma di apertura della sessione il mittente smette di inviare il messaggio iniziale X3DH.
- ▶ I dispositivi comunicano solo con Double Ratchet.

[MP17], [VD19]



Signal Protocol  
Il protocollo: Sesame

- ▶ Il messaggio iniziale X3DH è aggiunto a ogni messaggio di apertura di sessione, in modo che il destinatario lo usi per creare una sessione Double Ratchet corrispondente.
  - ▶ Ricevuto il messaggio di conferma di apertura della sessione il mittente smette di inviare il messaggio iniziale X3DH.
- \* I dispositivi comunicano solo con Double Ratchet.  
[MP17], [VD19]

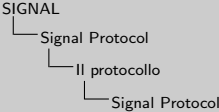
# Signal Protocol

## Il protocollo: Sesame

- Il messaggio iniziale X3DH è aggiunto a ogni messaggio di apertura di sessione, in modo che il destinatario lo usi per creare una sessione Double Ratchet corrispondente.
- Ricevuto il messaggio di conferma di apertura della sessione il mittente smette di inviare il messaggio iniziale X3DH.
- I dispositivi comunicano solo con Double Ratchet.

[MP17], [VD19]

2022-04-25



Signal Protocol  
Il protocollo: Sesame

- Il messaggio iniziale X3DH è aggiunto a ogni messaggio di apertura di sessione, in modo che il destinatario lo usi per creare una sessione Double Ratchet corrispondente.
- Ricevuto il messaggio di conferma di apertura della sessione il mittente smette di inviare il messaggio iniziale X3DH.
- I dispositivi comunicano solo con Double Ratchet.

[MP17], [VD19]

1	Sommario
2	Applicazione Signal
3	Crittografia End-to-End
4	Signal Protocol
5	Bibliografia

Storia dell'Applicazione

L'Applicazione e il Protocollo Signal

Applicazioni

Problematiche

Proprietà

Il protocollo

Difetti di progettazione

Considerazioni

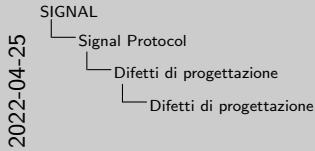
WhatsApp VS Signal

VS Telegram

# Difetti di progettazione

## Frosch analysis [FMB<sup>+</sup>16]

- *Export function* [fixed]: la funzione esportava la password necessaria per inviare le *one-time pre-keys* al server in *plaintext* su dispositivi Android
- *Vulnerabilità a UKS - Unknown key-share attack*



Difetti di progettazione

Frosch analysis [FMB<sup>+</sup>16]

- *Export function* [fixed]: la funzione esportava la password necessaria per inviare le *one-time pre-keys* al server in *plaintext* su dispositivi Android
- *Vulnerabilità a UKS - Unknown key-share attack*

I difetti di implementazione individuati in questo paper sono stati ricavati da un'analisi di X3DH e di Double Ratchet separatamente, ma non da un'analisi dell'interazione dei due algoritmi come invece avviene nel protocollo Signal.

UKS

Esempio:

Bob sa che Charlie lo inviterà a una festa. Per fare uno scherzo a Charlie, Bob sostituisce la propria chiave con quella di Dave. Quando Charlie invita Bob alla festa, Bob inoltrerà il messaggio a Dave. Dal punto di vista di Dave, sembrerà che Charlie abbia inviato il messaggio.

Charlie penserà di aver invitato Bob, ma avrà in effetti invitato Dave.



1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
Il protocollo  
**Difetti di  
progettazione**  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

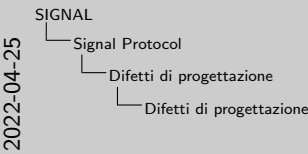
5 Bibliografia

# Difetti di progettazione

Cohn-Gordon analysis [CGCD<sup>+</sup>17]

L'analisi effettuata è valida presupponendo che tutte le KDF si comportino come oracoli che restituiscono un output simile a un output casuale.

Definiamo questa condizione come *random oracle model*.



Il paper [CGCD<sup>+</sup>17] è il primo studio scientifico a riportare un'analisi formale dell'interazione tra X3DH e Double Ratchet.

Difetti di progettazione

Cohn-Gordon analysis [CGCD<sup>+</sup>17]

L'analisi effettuata è valida presupponendo che tutte le KDF si comportino come oracoli che restituiscono un output simile a un output casuale.

Definiamo questa condizione come *random oracle model*.

1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
Il protocollo  
**Difetti di  
progettazione**  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

5 Bibliografia

# Difetti di progettazione

Cohn-Gordon analysis [CGCD<sup>+</sup>17]

L'analisi effettuata è valida presupponendo che tutte le KDF si comportino come oracoli che restituiscono un output simile a un output casuale.

Definiamo questa condizione come *random oracle model*.



Il paper [CGCD<sup>+</sup>17] è il primo studio scientifico a riportare un'analisi formale dell'interazione tra X3DH e Double Ratchet.

Difetti di progettazione

Cohn-Gordon analysis [CGCD<sup>+</sup>17]

L'analisi effettuata è valida presupponendo che tutte le KDF si comportino come oracoli che restituiscono un output simile a un output casuale.

Definiamo questa condizione come *random oracle model*.

1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
Il protocollo  
**Difetti di  
progettazione**  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

5 Bibliografia

# Difetti di progettazione

Cohn-Gordon analysis [CGCD<sup>+</sup>17]

L'attacco UKS individuato in [FMB<sup>+</sup>16] non è valido in questo modello perché esso, come in effetti anche il protocollo Signal, non differenzia gli identificatori di sessione (i.e. l'ID della sessione Alice-Bob è lo stesso ID della sessione Alice-Charlie).

Non sono stati individuati altri rilevanti difetti di progettazione che generino punti di debolezza nel protocollo.



UKS è un attacco prevenibile a livello di applicazione (introducendo degli identificativi per gli utenti, e.g. il numero di telefono).

N.B. Gli attacchi possono variare a seconda dell'implementazione fornita del protocollo, ma il protocollo in sé non presenta difetti rilevanti.

Difetti di progettazione  
Cohn-Gordon analysis [CGCD<sup>+</sup>17]

L'attacco UKS individuato in [FMB<sup>+</sup>16] non è valido in questo modello perché esso, come in effetti anche il protocollo Signal, non differenzia gli identificatori di sessione (i.e. l'ID della sessione Alice-Bob è lo stesso ID della sessione Alice-Charlie).

Non sono stati individuati altri rilevanti difetti di progettazione che generino punti di debolezza nel protocollo.

## Difetti di progettazione

Cohn-Gordon analysis [CGCD<sup>+</sup>17]

## 1 Sommario

## 2 Applicazione Signal

## Storia dell'Applicazione

## L'Applicazione e il Protocollo Signal

### 3 Crittografia End-to-End

## Applicazioni

## Problematiche

#### 4 Signal Protocol

Proprietà

## Il protocollo

Difetti di  
progettazione

### Considerazioni

## WhatsApp VS Signal VS Telegram

## 5 Bibliografia

L'attacco UKS individuato in [FMB<sup>+</sup>16] non è valido in questo modello perché esso, come in effetti anche il protocollo Signal, non differenzia gli identificatori di sessione (i.e. l'ID della sessione Alice-Bob è lo stesso ID della sessione Alice-Charlie).

Non sono stati individuati altri rilevanti difetti di progettazione che generino punti di debolezza nel protocollo.



UKS è un attacco prevenibile a livello di applicazione (introducendo degli identificativi per gli utenti, e.g. il numero di telefono).

N.B. Gli attacchi possono variare a seconda dell'implementazione fornita del protocollo, ma il protocollo in sé non presenta difetti rilevanti.

# Considerazioni

## 1 Sommario

## 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

## 3 Crittografia End-to-End

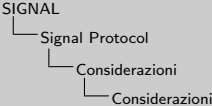
Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
**Considerazioni**  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia

2022-04-25



Considerazioni

# Bibliografia I

1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

5 Bibliografia



Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila.

A formal security analysis of the signal messaging protocol.

In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 451–466, 2017.



W Diffie and M Hellman.

New directions in cryptography.

In *IEEE Transactions on Information Theory*, pages 644–654, 1976.

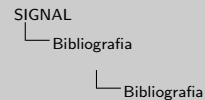


CPJ Middle East, North Africa Program, and CPJ Technology Program.

Why telegram's security flaws may put iran's journalists at risk - committee to protect journalists.

2016, May 1.

2022-04-25



Bibliografia I

-  Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila.  
A formal security analysis of the signal messaging protocol.  
In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 451–466, 2017.
-  W Diffie and M Hellman.  
New directions in cryptography.  
In *IEEE Transactions on Information Theory*, pages 644–654, 1976.
-  CPJ Middle East, North Africa Program, and CPJ Technology Program.  
Why telegram's security flaws may put iran's journalists at risk - committee to protect journalists.  
2016, May 1.

# Bibliografia II

## 1 Sommario

## 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia



Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz.

How secure is textsecure?

*In 2016 IEEE European Symposium on Security and Privacy (EuroS P), pages 457–472, 2016.*



Barton Gellman and Jerry Markon.

Edward snowden says motive behind leaks was to expose ‘surveillance state’.

*The Washington Post*, June 10, 2013.

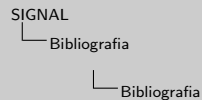


Yael Grauer.

Mr. robot uses protonmail, but it still isn’t fully secure.

2015, October 7.

2022-04-25



## Bibliografia II

-  Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz.  
How secure is textsecure?  
*In 2016 IEEE European Symposium on Security and Privacy (EuroS P), pages 457–472, 2016.*
-  Barton Gellman and Jerry Markon.  
Edward snowden says motive behind leaks was to expose ‘surveillance state’.  
*The Washington Post*, June 10, 2013.
-  Yael Grauer.  
Mr. robot uses protonmail, but it still isn’t fully secure.  
2015, October 7.

# Bibliografia III

1 Sommario

2 Applicazione  
Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

3 Crittografia  
End-to-End

Applicazioni  
Problematiche

4 Signal  
Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

5 Bibliografia



Andy Greenberg.  
Hacker lexicon: What is end-to-end encryption?  
2014, November 15.



Andy Greenberg.  
Hacker lexicon: What is the signal encryption protocol?  
2020, November 29.



Cos'è la e2ee (end-to-end encryption)?



Cryptography concepts - fundamentals - e3kit — virgil security.  
2020.



B Jacobs and J Daemen.  
Computer security: Public key crypto.  
2016.

2022-04-25

SIGNAL

Bibliografia

Bibliografia


Bibliografia III


- Andy Greenberg.  
Hacker lexicon: What is end-to-end encryption?  
2014, November 15.
- Andy Greenberg.  
Hacker lexicon: What is the signal encryption protocol?  
2020, November 29.
- Cos'è la e2ee (end-to-end encryption)?
- Cryptography concepts - fundamentals - e3kit — virgil security.  
2020.
- B Jacobs and J Daemen.  
Computer security: Public key crypto.  
2016.





# Bibliografia IV

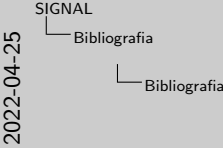
- 1 Sommario
- 2 Applicazione Signal
  - Storia dell'Applicazione
  - L'Applicazione e il Protocollo Signal
- 3 Crittografia End-to-End
  - Applicazioni
  - Problematiche
- 4 Signal Protocol
  - Proprietà
  - Il protocollo
  - Difetti di progettazione
  - Considerazioni
  - WhatsApp VS Signal
  - VS Telegram
- 5 Bibliografia

 [Hugo Krawczyk and Pasi Eronen.](#)  
Hmac-based extract-and-expand key derivation function (hkdf)'.  
2010.

 [Ben Lutkevich and Madelyn Bacon.](#)  
end-to-end encryption (e2ee).  
June 2021.

 [David J Lumb.](#)  
The story of signal.  
*Increment*, (7), 2018, October.

 [Moxie Marlinspike.](#)  
Whatsapp's signal protocol integration is now complete.  
Apr. 5, 2016.



Bibliografia IV

 [Hugo Krawczyk and Pasi Eronen.](#)  
Hmac-based extract-and-expand key derivation function (hkdf)'.  
2010.

 [Ben Lutkevich and Madelyn Bacon.](#)  
end-to-end encryption (e2ee).  
June 2021.

 [David J Lumb.](#)  
The story of signal.  
*Increment*, (7), 2018, October.

 [Moxie Marlinspike.](#)  
Whatsapp's signal protocol integration is now complete.  
Apr. 5, 2016.

# Bibliografia V

## 1 Sommario

## 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal

## 3 Crittografia End-to-End

Applicazioni  
Problematiche

## 4 Signal Protocol

Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia



Moxie Marlinspike and Trevor Perrin.

The double ratchet algorithm.

Technical report, Open Whisper Systems, 2016.



Moxie Marlinspike and Trevor Perrin.

The x3dh key agreement protocol.

Technical report, Open Whisper Systems, 2016.

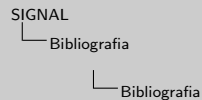


Moxie Marlinspike and Trevor Perrin.

The sesame algorithm: Session management for asynchronous message encryption.

Technical report, Open Whisper Systems, 2017.

2022-04-25



Bibliografia V

Moxie Marlinspike and Trevor Perrin.  
The double ratchet algorithm.  
Technical report, Open Whisper Systems, 2016.

Moxie Marlinspike and Trevor Perrin.  
The x3dh key agreement protocol.  
Technical report, Open Whisper Systems, 2016.

Moxie Marlinspike and Trevor Perrin.  
The sesame algorithm: Session management for asynchronous message encryption.  
Technical report, Open Whisper Systems, 2017.

# Bibliografia VI

## 1 Sommario

## 2 Applicazione Signal

Storia  
dell'Applicazione  
L'Applicazione e il  
Protocollo Signal


## 3 Crittografia End-to-End


Applicazioni  
Problematiche

## 4 Signal Protocol


Proprietà  
Il protocollo  
Difetti di  
progettazione  
Considerazioni  
WhatsApp VS Signal  
VS Telegram

## 5 Bibliografia

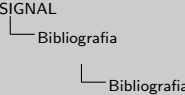
 [Mike Pound.](#)  
Double ratchet messaging encryption - computerphile.  
[Video.](#)

 [K Poulsen.](#)  
Snowden's email provider loses appeal over encryption keys.  
[2014, April 16.](#)

 [Signal documentation.](#)

 [Ryan Singel.](#)  
Encrypted e-mail company hushmail spills to feds.  
[2007, November 7.](#)

2022-04-25



Bibliografia VI

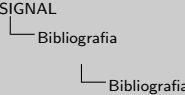
-  [Mike Pound.](#)  
Double ratchet messaging encryption - computerphile.  
[Video.](#)
-  [K Poulsen.](#)  
Snowden's email provider loses appeal over encryption keys.  
[2014, April 16.](#)
-  [Signal documentation.](#)
-  [Ryan Singel.](#)  
Encrypted e-mail company hushmail spills to feds.  
[2007, November 7.](#)

# Bibliografia VII



Dion Van Dam.  
Analysing the signal protocol - a manual and automated analysis of the signal  
protocol.  
Master's thesis, Radboud University, 2019.

2022-04-25



Bibliografia VII

 **Dion Van Dam.**  
Analysing the signal protocol - a manual and automated analysis of the signal  
protocol.  
Master's thesis, Radboud University, 2019.