Protocollo Signal

Elena Tonini, Matr.727382

Università degli Studi di Brescia

A.A. 2021/2022

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
4 Signal Protocol
Storia
Vulnerabilità
LFSR
Attacchi Algebrici
Attacchi possibili
Considerazioni
WhatsApp VS Signal
5 Telegram VS Signal
6 Bibliografia

Sommario I

1. Sommario
2. Applicazione Signal
   Storia dell'Applicazione
   L'Applicazione e il Protocollo Signal
3. Crittografia End-to-End
4. Signal Protocol
   Storia
   Vulnerabilità
   LFSR
   Attacchi Algebrici
   Attacchi possibili
   Considerazioni
   WhatsApp VS Signal
5. Telegram VS Signal
6. 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.

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

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

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

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.

Applicazione Signal

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.

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.

Applicazione Signal

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

- **Facebook**: introdusse la feature *Secret Conversations* agli utenti di Facebook Messenger nel luglio 2016
- **Allo**: rilasciata nel settembre 2016, utilizzava 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)

Applicazione Signal

Tra le più note implementazioni (parziali) del Protocollo Signal troviamo:
- **Facebook**: introdusse la feature *Secret Conversations* agli utenti di Facebook Messenger nel luglio 2016
- **Allo**: rilasciata nel settembre 2016, utilizzava 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)

- Facebook: solo nelle Secret Conversations
- Allo: applicazione di messaggistica di Google, non esiste più dal 12 marzo 2019
- Whatsapp: introdotto per la prima volta nel 2014 per utenti Android, esteso a tutti gli utenti nel 2016
- Google: introdotto di default nell'applicazione di messaggi su Android

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 tal modo E2EE garantisce che chi non è munito di autorizzazione non abbia la possibilità di accedere al contenuto della conversazione.

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 tal modo E2EE garantisce che chi non è munito di autorizzazione non abbia la possibilità di accedere al contenuto della conversazione.

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.

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

End-to-End Encryption

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

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

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**: se non volute, possono essere introdotte tramite attacchi cyber e poi sfruttate per violare la sicurezza del sistema

End-to-End Encryption

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**: se non volute, possono essere introdotte tramite attacchi cyber e poi sfruttate per violare la sicurezza del sistema

- Endpoint security: E2EE protegge i dati solo tra i due endpoint; ciò significa che i due endpoint possono essere soggetti ad attacchi;
- Attacchi MITM: la conversazione può essere soggetta a *eavesdropping* da parte di terzi malintenzionati in grado di intercettare i messaggi e impersonare il destinatario. Essi possono, per esempio, scambiare le chiavi con il mittente, decifrare il messaggio inviato e poi inoltrarlo al vero destinatario senza farsi notare;
- 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.

EAVESDROPPING:

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;

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.

La protezione dei dati tramite *encryption in transit* consiste nel cifrare i dati solo lungo il percorso su cui vengono trasmessi ma non alla sorgente. In queste condizioni, colui che invia i dati ha accesso al loro contenuto, cosa che si vuole spesso evitare.

Storia

CRYPTO1 è uno stream cipher sviluppato da *NXP Semiconductors* nel 1994 insieme al livello di comunicazione di MIFARE Classic [**?**]

Storia

CRYPTO1 è uno stream cipher sviluppato da *NXP Semiconductors* nel 1994 insieme al livello di comunicazione di MIFARE Classic [**?**]

Storia

CRYPTO1 è uno stream cipher sviluppato da *NXP Semiconductors* nel 1994 insieme al livello di comunicazione di MIFARE Classic [**?**]

Storia

CRYPTO1 è uno stream cipher sviluppato da *NXP Semiconductors* nel 1994 insieme al livello di comunicazione di MIFARE Classic [**?**]

Security through obscurity

La sicurezza del sistema era affidata al concetto di *Security through obscurity*

- Metodo fortemente sconsigliato da tutti gli organi normativi sulla sicurezza[?]
- Tecnica in contrasto con *Security by Design* e *Open security*

Security through obscurity

La sicurezza del sistema era affidata al concetto di *Security through obscurity*
- Metodo fortemente sconsigliato da tutti gli organi normativi sulla sicurezza[?]
- Tecnica in contrasto con *Security by Design* e *Open security*

Security through obscurity

La sicurezza del sistema era affidata al concetto di *Security through obscurity*

- Metodo fortemente sconsigliato da tutti gli organi normativi sulla sicurezza[**?**]
- Tecnica in contrasto con *Security by Design* e *Open security*

Security through obscurity

La sicurezza del sistema era affidata al concetto di *Security through obscurity*
- Metodo fortemente sconsigliato da tutti gli organi normativi sulla sicurezza[?]
- Tecnica in contrasto con *Security by Design* e *Open security*

Security through obscurity

La sicurezza del sistema era affidata al concetto di *Security through obscurity*

- Metodo fortemente sconsigliato da tutti gli organi normativi sulla sicurezza[**?**]
- Tecnica in contrasto con *Security by Design* e *Open security*

Security through obscurity

La sicurezza del sistema era affidata al concetto di *Security through obscurity*
- Metodo fortemente sconsigliato da tutti gli organi normativi sulla sicurezza[?]
- Tecnica in contrasto con *Security by Design* e *Open security*

Contrariamente alle due politiche *Security by Design* e *Open security* la sicurezza tramite offuscazione è fortemente sconsigliata, in quanto affida la sicurezza del sistema al fatto che nessuno riesca a comprenderlo.
Questa pratica rende quindi il sistema vulnerabile a qualsiasi attacco di tipo reverse engeneering, oltre che a possibili fughe di informazioni.
L'utilizzo di ideologie "open" permette la validazione del sistema da parte di un maggior numero di enti e di membri di una comunità, permettendo così l'individuazione di falle in minor tempo.

Il metodo più efficiente, però, consiste sempre nell'utilizzo di sistemi già esistenti e ritenuti sicuri (p.e. tritium)

La caduta di CRYPTO1

Nel 2008/2009 più ricercatori in contemporanea hanno trovato vulnerabilità e sono stati rilasciati attacchi sul crittosistema CRYPTO1 che ne hanno interamente distrutto la sicurezza.[?][?][?]

Il sistema presenta falle nella sicurezza in più settori:

- Random Number Generator
- Proprietà algebriche
- Complessità delle chiavi (Che rendono il crittosistema vulnerabile a attacchi di tipo Bruteforce[?])

La caduta di CRYPTO1

Nel 2008/2009 più ricercatori in contemporanea hanno trovato vulnerabilità e sono stati rilasciati attacchi sul crittosistema CRYPTO1 che ne hanno interamente distrutto la sicurezza.[?][?][?]
Il sistema presenta falle nella sicurezza in più settori:
Random Number Generator
Proprietà algebriche
Complessità delle chiavi (Che rendono il crittosistema vulnerabile a attacchi di tipo Bruteforce[?])

La caduta di CRYPTO1

Nel 2008/2009 più ricercatori in contemporanea hanno trovato vulnerabilità e sono stati rilasciati attacchi sul crittosistema CRYPTO1 che ne hanno interamente distrutto la sicurezza.[?][?][?]
Il sistema presenta falle nella sicurezza in più settori:

- Random Number Generator
- Proprietà algebriche
- Complessità delle chiavi (Che rendono il crittosistema vulnerabile a attacchi di tipo Bruteforce[?])

La caduta di CRYPTO1

Nel 2008/2009 più ricercatori in contemporanea hanno trovato vulnerabilità e sono stati rilasciati attacchi sul crittosistema CRYPTO1 che ne hanno interamente distrutto la sicurezza.[?][?][?]
Il sistema presenta falle nella sicurezza in più settori:
- Random Number Generator
- Proprietà algebriche
- Complessità delle chiavi (Che rendono il crittosistema vulnerabile a attacchi di tipo Bruteforce[?])

La caduta di CRYPTO1

Nel 2008/2009 più ricercatori in contemporanea hanno trovato vulnerabilità e sono stati rilasciati attacchi sul crittosistema CRYPTO1 che ne hanno interamente distrutto la sicurezza.[?][?][?]
Il sistema presenta falle nella sicurezza in più settori:

- Random Number Generator
- Proprietà algebriche
- Complessità delle chiavi (Che rendono il crittosistema vulnerabile a attacchi di tipo Bruteforce[?])

La caduta di CRYPTO1

Nel 2008/2009 più ricercatori in contemporanea hanno trovato vulnerabilità e sono stati rilasciati attacchi sul crittosistema CRYPTO1 che ne hanno interamente distrutto la sicurezza.[?][?][?]
Il sistema presenta falle nella sicurezza in più settori:
- Random Number Generator
- Proprietà algebriche
- Complessità delle chiavi (Che rendono il crittosistema vulnerabile a attacchi di tipo Bruteforce[?])

La caduta di CRYPTO1

Nel 2008/2009 più ricercatori in contemporanea hanno trovato vulnerabilità e sono stati rilasciati attacchi sul crittosistema CRYPTO1 che ne hanno interamente distrutto la sicurezza.[**?**][**?**][**?**]
Il sistema presenta falle nella sicurezza in più settori:

- Random Number Generator
- Proprietà algebriche
- Complessità delle chiavi (Che rendono il crittosistema vulnerabile a attacchi di tipo Bruteforce[**?**])

La caduta di CRYPTO1

Nel 2008/2009 più ricercatori in contemporanea hanno trovato vulnerabilità e sono stati rilasciati attacchi sul crittosistema CRYPTO1 che ne hanno interamente distrutto la sicurezza.[?][?][?]
Il sistema presenta falle nella sicurezza in più settori:
- Random Number Generator
- Proprietà algebriche
- Complessità delle chiavi (Che rendono il crittosistema vulnerabile a attacchi di tipo Bruteforce[?])

Le vulnerabilità trovate negli anni nel crittosistema sono le seguenti.
Alcune di queste sono di facile soluzione perchè riguardano il lato hardware del lettore, per cui, a fronte di un costo maggiore, è possibile migliorarne le capacità.

Le vulnerabilità trovate negli anni nel crittosistema sono le seguenti.
Alcune di queste sono di facile soluzione perchè riguardano il lato hardware del lettore, per cui, a fronte di un costo maggiore, è possibile migliorarne le capacità.

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

- Utilizzo di chiavi a 48 bit: Possibili attacchi BruteForce [?]
- RNG del tag non è crittograficamente sicuro. Infatti è un LFSR con condizione iniziale costante.
  - Consegue che lo stato è prevedibile e dipende dal tempo trascorso dal poweron [?][?]
  - Inoltre i numeri casuali sono generati a partire da 16 bit del registro
  - in particolare i numeri sono generati ad ogni ciclo di clock del tag, quindi la precisione del attaccante deve limitarsi a quanti di 10 microsecondi (106kHz) e la sequenza di numeri i ripete ogni 65535 iterazioni (0.6s)
- L'RNG dei lettori viene aggiornato solamente ad ogni nuova autenticazione [?]
- La funzione di filtraggio del LFSR usa 20bit del registro e sono solo bit in posizione dispari
- LFSR State Recovery
- LFSR Rollback
- i bit di parità sono computati sul plaintext e poi inviati non cifrati
- nested authentication attacks

Dettagli sul LFSR

todo: reverse engeneering [**?**] del chip per trovare il circuito di autenticazione
attacchi per trovare la funzione [**?**] attacchi dati dai odd nested auth e drschottky

Dettagli sul LFSR

todo: reverse engeneering [?] del chip per trovare il circuito di autenticazione attacchi per trovare la funzione [?] attacchi dati dai odd nested auth e drschottky

Attacchi Algebrici

Bibliografia I

Bibliografia I