

Attacchi chosen ciphertext

Paolo D'Arco
pdarco@unisa.it

Università di Salerno

Elementi di Crittografia

- 1 Attacchi di tipo chosen ciphertext
- 2 Padding-Oracle attack

Attacchi di tipo chosen ciphertext

L'Adv ha anche l'abilità di ottenere i messaggi in chiaro corrispondenti a cifrati di propria scelta. Modelliamo questa abilità dando ad Adv accesso ad un oracolo per la decifratura

$PrivK_{A,\Pi}^{cca}(n)$

- ❶ Il Challenger sceglie $k \leftarrow Gen(1^n)$
- ❷ A riceve in input 1^n ed ha accesso agli oracoli $Enc_k(\cdot)$ e $Dec_k(\cdot)$. Dá in output m_0 ed m_1 della stessa lunghezza
- ❸ Il Challenger sceglie $b \leftarrow \{0, 1\}$ e calcola $c^* \leftarrow Enc_k(m_b)$
- ❹ $A(1^n)$ riceve c^* , continua ad accedere agli oracoli $Enc_k(\cdot)$ e $Dec_k(\cdot)$. Non può chiedere la decifratura di c^* . Alla fine dá in output $b' \in \{0, 1\}$.
- ❺ Se $b = b'$, l'output dell'esperimento é 1 ($A(1^n)$ vince); altrimenti, 0.

Definizione 3.33. Uno schema di cifratura a chiave privata $\Pi = (Gen, Enc, Dec)$ ha cifrature indistinguibili rispetto ad un attacco di tipo chosen ciphertext (CCA-sicuro) se, per ogni Adv A PPT, esiste una funzione trascurabile $negl$ tale che:

$$Pr[PrivK_{A,\Pi}^{cca}(n) = 1] \leq \frac{1}{2} + negl(n),$$

dove la probabilità é calcolata su

- randomness usata da A
- randomness usata nell'esperimento
 - scelta della chiave
 - scelta del bit b
 - random bit usati da $Enc_k(\cdot)$

Si può dimostrare che:

Sicurezza CCA per un singolo messaggio \Rightarrow Sicurezza CCA per messaggi multipli

Sono gli attacchi CCA realistici?

- 1 US NAVY 1942: "manda messaggi cifrati ed osserva il comportamento del nemico"
- 2 Server on-line: "manda messaggi cifrati ed osserva la reazione"
e.g., servizio rifiutato \Rightarrow cifrato scorretto
- 3 Protocolli crittografici: le parti oneste in alcuni casi possono inconsapevolmente agire come oracoli di decifratura

Tutti gli schemi che abbiamo studiato sono insicuri rispetto ad attacchi di tipo CCA

$$Enc_k(m) = \langle r, F_k(r) \oplus m \rangle$$

F Pseudocasuale \Rightarrow CPA-sicuro

Schemi di cifratura introdotti e attacchi CCA

L'Adv A che segue, di tipo CCA, é in grado di distinguere le cifrature con prob. 1.

- A sceglie $m_0 = 0^\ell$ ed $m_1 = 1^\ell$.
- riceve il cifrato $c = \langle r, s \rangle$
- calcola s' invertendo il primo bit di s e chiede all'oracolo $Dec_k(\cdot)$ di decifrare $c' = \langle r, s' \rangle$
(nota che $c' = \langle r, s' \rangle \neq c = \langle r, s \rangle$, quindi la query é lecita)
- Se $Dec_k(c') = 10^{\ell-1}$ allora $b' = 0$; altrimenti ($Dec_k(c') = 01^{\ell-1}$), $b' = 1$
- Dá in output b' .

Schemi di cifratura introdotti e attacchi CCA

Osservazione: ogni schema di cifratura in cui i cifrati possono essere "manipolati" in modo controllato *non* possono essere CCA-sicuri.

Sicurezza CCA \Rightarrow "Non malleabilità"

Non malleabilità = Informalmente significa che un cifrato modificato, quando decifrato, comporta

- la restituzione di un messaggio di errore \perp , oppure
- la restituzione di un messaggio m' *scorrelato* dal messaggio m originariamente cifrato

Costruire schemi di cifratura CCA-sicuri richiede strumenti aggiuntivi.

Padding-Oracle attack

Un attacco CCA utilizzato contro protocolli reali

L'attacco richiede soltanto l'abilità di determinare **se un cifrato modificato viene decifrato correttamente oppure no**

Nella cifratura CBC un msg deve avere lunghezza multipla della lunghezza del blocco. Pertanto, uno *schema di padding* (completamento) viene usato per ottenere il requisito.

Sia L la lunghezza del blocco in byte.

Sia b il numero di byte che devono essere aggiunti al messaggio. Risulta $1 \leq b \leq L$.

Se il messaggio é già "multiplo" di L , allora $b = L$

Strategia di padding: aggiungiamo al messaggio la stringa contenente b (un byte) esattamente b volte.

Padding-Oracle attack

1 byte necessario \Rightarrow aggiungiamo il byte con valore 0x01 (rappr. esadecimale)

2 byte necessari \Rightarrow aggiungiamo la stringa 0x0202 (rappr. esadecimale)

4 byte necessari \Rightarrow aggiungiamo la stringa 0x04040404 (rappr. esadecimale)

...

In fase di decifratura, il pad viene controllato e rimosso, prima di restituire il plaintext.

Se il pad *non* è corretto \Rightarrow viene generato e restituito un errore \perp



Usando questa informazione, Adv riesce a recuperare *completamente* il plaintext

Padding-Oracle attack

Consideriamo l'attacco su un cifrato di 3 blocchi

$$c = \langle c_0, c_1, c_2 \rangle = \langle IV, c_1, c_2 \rangle$$

Avendo applicato il CBC-mode e lo schema di padding descritto,

$$m_2 = F_k^{-1}(c_2) \oplus c_1 \quad \text{dove } m_2 \text{ termina con } 0xbb \dots b \text{ (} b \text{ volte)}$$

Sia c'_1 identico a c_1 a parte una modifica nel byte finale. Nota che:

$$c' = \langle IV, c'_1, c_2 \rangle \quad \Rightarrow \quad m'_2 = F_k^{-1}(c_2) \oplus c'_1$$

Risulta m'_2 uguale ad m_2 eccetto che per una modifica nel byte finale.

Padding-Oracle attack

Lo stesso ragionamento vale per qualsiasi byte scelto, i.e., se modifichiamo l' i -esimo byte di c_1 , allora m'_2 differisce da m_2 nell' i -esimo byte soltanto.

In generale,

$$c'_1 = c_1 \oplus \Delta \quad \Rightarrow \quad m'_2 = m_2 \oplus \Delta$$

Come possiamo usare questa relazione per calcolare il valore di b ?

Un Adv A può agire come segue:

A modifica il *primo* byte di c_1 e chiede la decifratura di $c' = \langle IV, c_1^1, c_2 \rangle$

Se la decifratura

- *fallisce* significa che, verificando il pad, l'algoritmo di decifratura è arrivato al primo byte e l'ha trovato scorretto $\Rightarrow b = L$
- *non fallisce* significa che il primo byte è del messaggio (non del pad) $\Rightarrow b < L$

Padding-Oracle attack

A ripete l'attacco modificando il *secondo* byte di c_1 e chiedendo la decifratura di $c'' = \langle IV, c_1^2, c_2 \rangle$

L'attacco viene ripetuto al più L volte ed alla fine A ottiene il valore di b .

Disponendo di b , A può calcolare i byte del messaggio uno dopo l'altro al modo seguente:

A sa che m_2 termina con $0xB0xb \dots b$ dove

- B è il byte sconosciuto del messaggio in chiaro che A vuole conoscere
- $0xb \dots b$ è il pad, con il byte b ripetuto b volte

A definisce, per ogni i tale che $0 \leq i < 2^8$,

$$\Delta_i \stackrel{\text{def}}{=} \begin{array}{c} 0x00 \dots 0 \ 0xi \ 0x(b+1) \dots (b+1) \\ \oplus \ 0x00 \dots 0 \ 0x0 \ 0xb \qquad \dots \quad b \end{array}$$

Padding-Oracle attack

Si noti che gli ultimi $b + 1$ byte di Δ_i contengono

l'intero i seguito da $(b + 1) \oplus b \dots (b + 1) \oplus b$

A questo punto, quando A chiede la decifratura di

$$c_i = \langle IV, c_1 \oplus \Delta_i, c_2 \rangle$$

l'ultima parte del messaggio sottostante al cifrato é

$$0x(B \oplus i)0x(b + 1) \dots (b + 1).$$

Nota che il pad $0xb \dots b$ é stato sostituito con $0x(b + 1) \dots (b + 1)$.

Padding-Oracle attack

Pertanto, se la decifratura é corretta

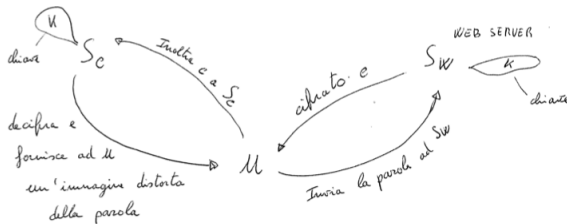
$$0x(B \oplus i) = 0x(b + 1) \quad \Rightarrow \quad B = (b + 1) \oplus i.$$

A ha quindi calcolato il byte B del messaggio!

L'attacco viene ripetuto per $i = 0, \dots, 2^8 - 1$, fino a quando la decifratura non risulta corretta e B viene calcolato

L'attacco può essere esteso per recuperare gli altri byte del messaggio in chiaro.

Un attacco di tipo Padding-Oracle contro un CAPTCHA



Il punto chiave di questa organizzazione é che S_C decifra *qualsiasi* cifrato riceva e dá un errore se il cifrato é scorretto.



\mathcal{U} può implementare un attacco di tipo Padding-oracle per calcolare la parola cifrata "senza intervento umano"!

Un attacco di tipo Padding-Oracle contro un CAPTCHA

Attacco CCA

- Manipola opportunamente il cifrato c
- Invia la versione manipolata e osserva la risposta del server S_c
- Ripete fino a quando non riesce a calcolare la parola

Pertanto, uno schema di cifratura CCA-sicuro é necessario, a meno di rendere il CAPTCHA inutile.