Modalitá operative di cifratura

Paolo D'Arco
pdarco@unisa.it

Universitá di Salerno

Elementi di Crittografia

Contenuti

Modalitá operative

Esistono diversi modi per cifrare in maniera sicura **messaggi lunghi** usando

- Stream cipher
- Cifrari a blocchi

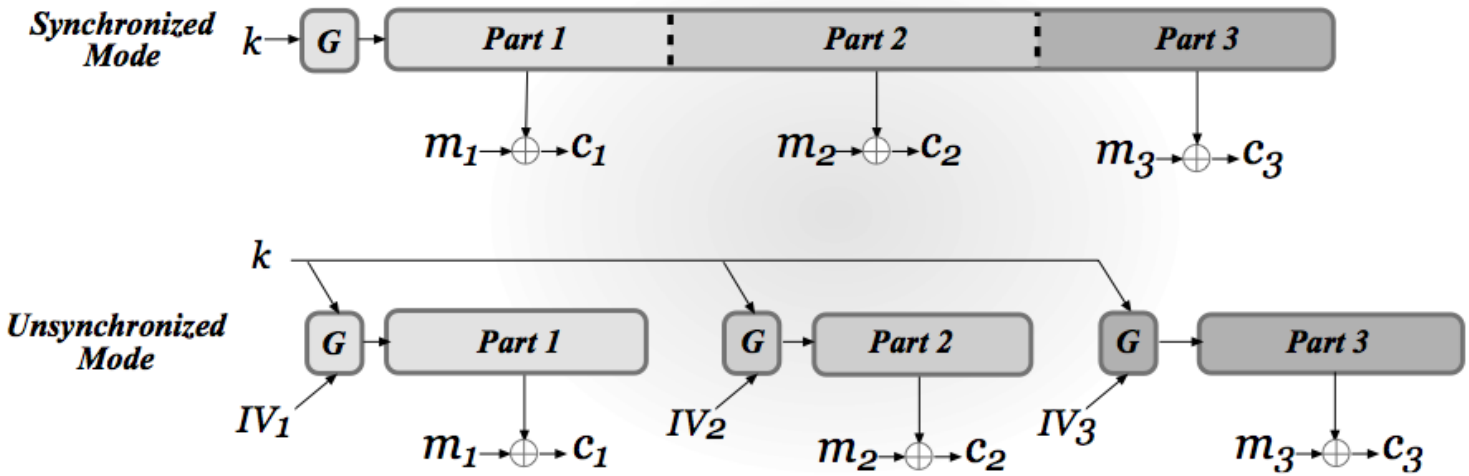
Modalitá operative per stream cipher

Abbiamo visto con la costruzione con PRG come cifrare messaggi di lunghezza fissata in modo EAV-sicuro.

Gli stream cipher possono essere utilizzati in

- modo sincrono (con stato)
- modo asincrono (senza stato)

Cifratura con stream cipher: sincrona e asincrona



Sincrono: viene generata una lunga stringa pseudocasuale e una parte diversa viene usata per ogni messaggio

Estensione del PRG

Abbiamo visto come costruire G_ℓ, con ℓ fisso, da uno stream cipher

Possiamo estendere la costruzione per ottenere un generatore a lunghezza variabile

$$G_\infty(s, 1^\ell)$$

che invoca $\mathit{Init}(s)$ e poi $\mathit{GetBits}$ esattamente ℓ volte.

Cifratura e decifratura diventano

$$c := G_\infty(k, 1^{|m|}) \oplus m \qquad\qquad m := c \oplus G_\infty(k, 1^{|c|}).$$

Se le parti mantengono uno stato, i messaggi $m_1, m_2, \ldots$, possono essere visti come un messaggio lungo.

Cifratura in modalitá sincrona

$st_0 := Init(k)$

Per cifrare m_1, il trasmittente

$$\text{invoca } GetBits\ \ell_1 \text{ volte e calcola } c_1 := pad_1 \oplus m_1$$

Per decifrare c_1, il ricevente

$$\text{invoca } GetBits\ \ell_1 \text{ volte e calcola } m_1 := pad_1 \oplus c_1$$

Per cifrare m_2, le parti riprendono le computazioni dallo stato st_{ℓ_1}, e procedono allo stesso modo.

Per i messaggi successivi si procede analogamente, partendo dallo stato raggiunto all'ultima cifratura.

Cifratura in modalitá sincrona

Questa modalitá é utile per una sessione tra le parti.

Come abbiamo visto se, per ogni ℓ, $G_\infty(s, 1^\ell)$ é un PRG, allora lo schema di cifratura é EAV-sicuro.

Non funziona bene per comunicazioni sporadiche o quando una parte puó comunicare da diversi dispositivi.

Cifratura in modalitá asincrona

Nella modalitá asincrona la funzione $Init(\cdot)$ accetta un vettore di inizializzazione IV, cioé

$$st_0 = Init(s, IV).$$

In questo caso abbiamo 3 input: seme, vettore di inizializzazione, lunghezza output desiderata

$$G_\infty(s, IV, 1^\ell)$$

Per cifrare m di lunghezza ℓ usando la chiave k, il trasmittente

sceglie $IV \leftarrow \{0,1\}^n$ e calcola $c =< c_1, c_2 >:=< IV, G_\infty(k, IV, 1^\ell) \oplus m >$

Per decifrare c usando la chiave k, il ricevente

$$\text{calcola } m := G_\infty(k, c_1, 1^\ell) \oplus c_2$$

Sicurezza

Se, per ogni ℓ, la funzione

$$F_k(IV) \stackrel{def}{=} G_\infty(k, IV, 1^\ell)$$

é una PRF, allora lo schema di cifratura é CPA-sicuro.

Osservazione: i vettori di inizializzazione vengono scelti *uniformemente a caso*

$$\Downarrow$$

F pertanto deve solo essere *debolmente* pseudocasuale, cioé pseudocasuale rispetto ad input scelti uniformemente a caso.

(Ricordo che nella definizione di funzione pseudocasuale gli input possono essere scelti in accordo a *qualsiasi* distribuzione)

Funzioni pseudocasuali e stream cipher

Abbiamo visto che possiamo costruire uno stream cipher a partire da una *PRF*.

CONSTRUCTION 3.29

Let F be a pseudorandom function. Define a stream cipher (Init, GetBits), where each call to GetBits outputs n bits, as follows:

- Init: on input $s \in \{0,1\}^n$ and $IV \in \{0,1\}^n$, set $\mathrm{st}_0 := (s, IV)$.
- GetBits: on input $\mathrm{st}_i = (s, IV)$, compute $IV' := IV + 1$ and set $y := F_s(IV')$ and $\mathrm{st}_{i+1} := (s, IV')$. Output (y, st_{i+1}).

Alternativa: $y = F_s(IV || < i >)$, con $IV \in \{0,1\}^{3/4n}$ e $i \in \{0,1\}^{n/4}$.

Modalitá operative per cifrari a blocchi

Nello schema di cifratura CPA-sicuro basato su PRF (cifrario a blocchi)

- la lunghezza del cifrato é doppia rispetto alla lunghezza del messaggio

Le modalitá operative per i cifrari a blocchi sono metodi per cifrare messaggi di lunghezza arbitraria con cifrati "corti".

Sia F un cifrario a blocchi con lunghezza di blocco n. Sia

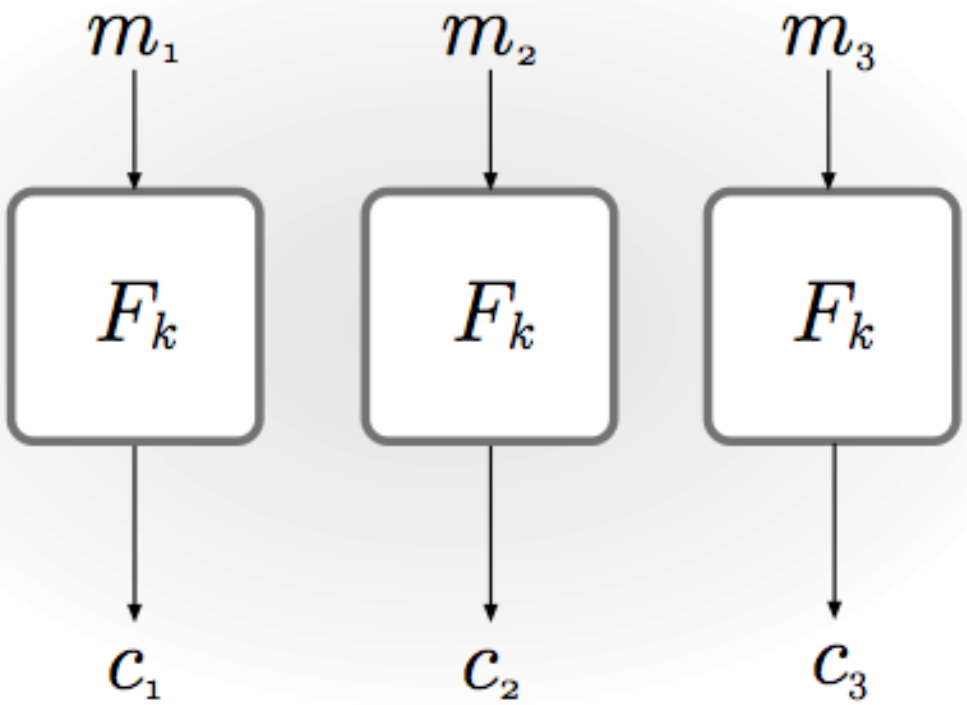
$$m = m_1 m_2 \dots m_\ell, \qquad m_i \in \{0,1\}^n \qquad \text{per ogni } i = 1, \dots, \ell.$$

Se necessario l'ultimo blocco viene "completato" (padded) in modo tale che $|m_\ell| = n$

La modalitá operativa piú semplice é l'Electronic Code Book (ECB) mode

$$c := < F_k(m_1), F_k(m_2), \dots, F_k(m_\ell) >$$

ECB mode



ECB mode - Analisi

ECB é deterministico

- Non puó dare sicurezza CPA
- Non ha neanche cifrature indistinguibili rispetto ad un eavesdropper
  - se un blocco si ripete nel messaggio in chiaro, allora il blocco si ripeterá nel cifrato

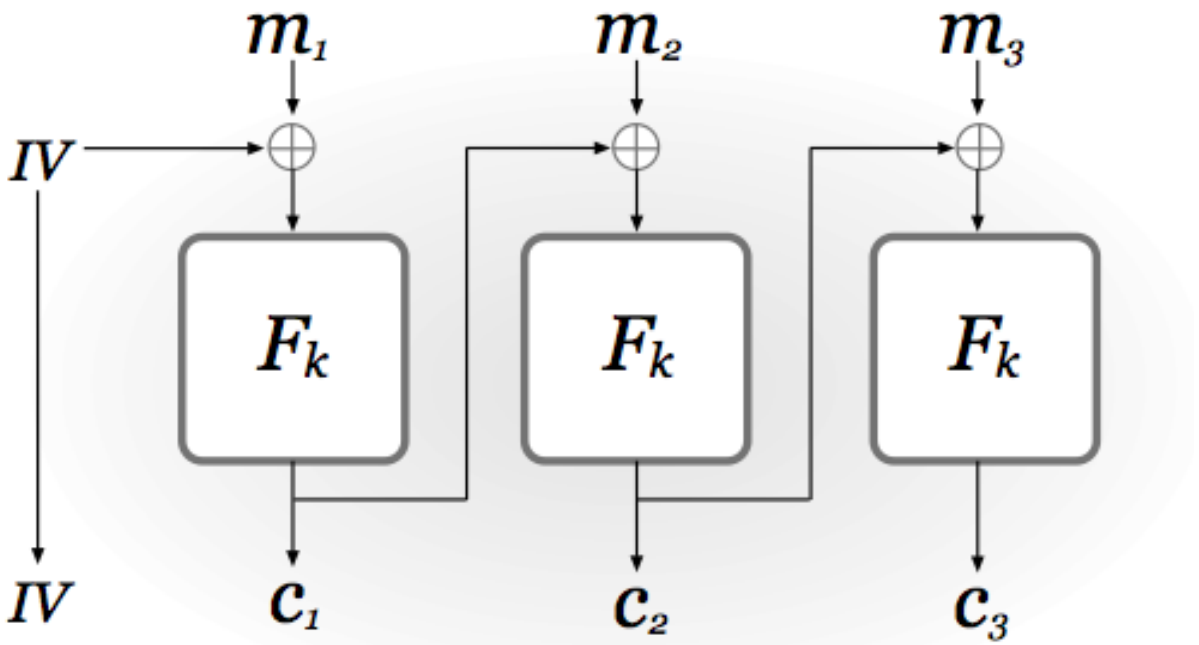
$$m_1 m^* m_3 m_4 m^* \ldots \quad \Rightarrow \quad c_1 c^* c_3 c_4 c^* \ldots$$

É facile allora distinguere tra due messaggi

- Adv sceglie m_0 con un blocco ripetuto ed m_1 con blocchi tutti distinti e analizza i blocchi del cifrato

In conclusione: ECB non dovrebbe mai essere usato

Modalitá Cipher Block Chaining - CBC mode



Viene scelto (uniformemente) il vettore di inizializzazione $IV \in \{0,1\}^n$. Poi,

$$c_0 = IV, \qquad c_i = F_k(c_{i-1} \oplus m_i), \qquad \text{per } i = 1, \ldots, \ell.$$

Modalitá Cipher Block Chaining - CBC mode

La decifratura del cifrato $< c_0, c_1, \ldots, c_\ell >$ viene effettuata calcolando

$$m_i = F_k^{-1}(c_i) \oplus c_{i-1}, \qquad \text{per } i = 1, \ldots, \ell.$$

CBC é una modalitá *probabilistica*

Se F é una permutazione pseudocasuale $\Rightarrow$ la modalitá CBC é CPA-sicura.

Inconveniente: la cifratura deve essere effettuata sequenzialmente.

Nota: se il vettore di inizializzazione IV *non* viene scelto uniformemente in $\{0,1\}^n$ ma semplicemente in modo che i valori siano distinti

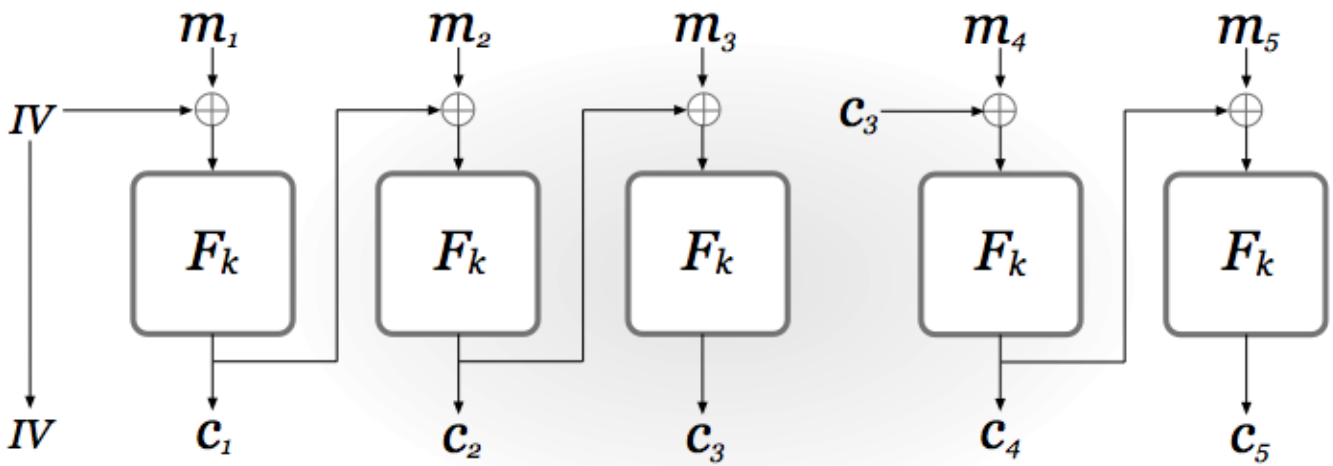
$$\Downarrow$$

la variante *non* é piú sicura.

Chained CBC mode

Una variante di CBC utilizzata é la modalitá con concatenazione.

Idea: invece di usare un nuovo *IV* per ciascuna cifratura, usa *l'ultimo blocco* c_ℓ del cifrato precedente come *IV*.



Chained CBC mode - Analisi

É una variante con stato.

Usata in SSL 3.0 e in TLS 1.0.

Sembra *tanto* sicura *quanto* CBC.

Purtroppo **non** é cosí!

É vulnerabile ad attacchi di tipo chosen-plaintext.

L'attacco si basa sul fatto che il vettore *IV* é noto *prima* che la seconda cifratura abbia luogo.

Chained CBC mode - Analisi

Adv sa che $m_1 \in \{m_1^0, m_1^1\}$

- Osserva il primo cifrato $c :=< IV, c_1, c_2, c_3 >$ di $m_1 m_2 m_3$
- Chiede una cifratura all'oracolo di $m_4 m_5$ dove $m_4 = IV \oplus m_1^0 \oplus c_3$
- Ottiene $< c_4, c_5 >$
- Se $c_4 = c_1$ allora dá in output $b' = 0$ (i.e., $m_1 = m_1^0$). Altrimenti, $b' = 1$.

Chained CBC mode - Analisi

É facile verificare che $c_4 = c_1$ se e solo se $m_1 = m_1^0$. Infatti:

$$c_4 = F_k(m_4 \oplus c_3) = F_k((IV \oplus m_1^0 \oplus c_3) \oplus c_3) = F_k(IV \oplus m_1^0)$$

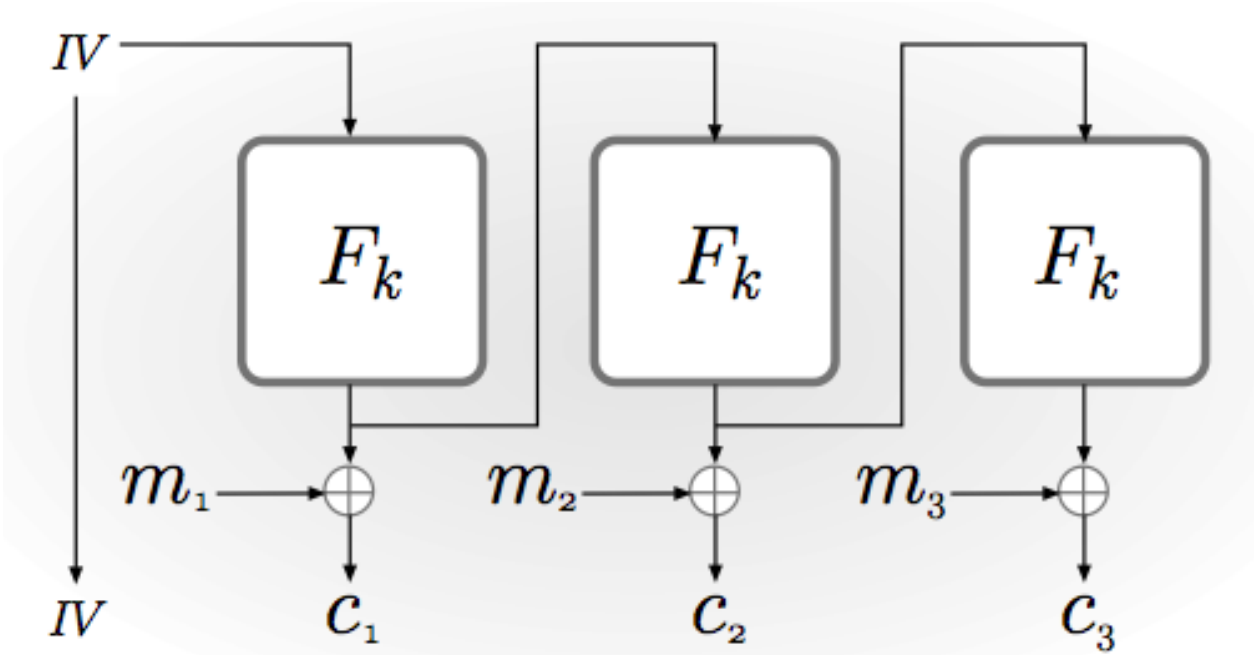
mentre

$$c_1 = F_k(IV \oplus m_1).$$

Lezione: ogni modifica ad uno schema crittografico puó essere pericolosa, anche se questa modifica sembra ragionevole ed innocua.

Modalitá Output Feedback Mode - OFB mode

Realizza uno stream cipher asincrono



Modalitá Output Feedback Mode - OFB mode

Viene scelto uniformemente il vettore di inizializzazione $IV \in \{0,1\}^n$.

Si genera la stringa pseudocasuale

$$y_0 = IV, \qquad y_i = F_k(y_{i-1}), \qquad \text{per } i = 1, \ldots, \ell.$$

Ciascun blocco m_i del messaggio in chiaro viene cifrato calcolando

$$c_0 = y_0, \qquad c_i = m_i \oplus y_i, \qquad \text{per } i = 1, \ldots, \ell.$$

La decifratura del cifrato $< c_0, c_1, \ldots, c_\ell >$ viene effettuata calcolando

$$y_0 = c_0, \qquad y_i = F_k(y_{i-1}), \quad \text{e} \qquad m_i = c_i \oplus y_i, \qquad \text{per } i = 1, \ldots, \ell.$$

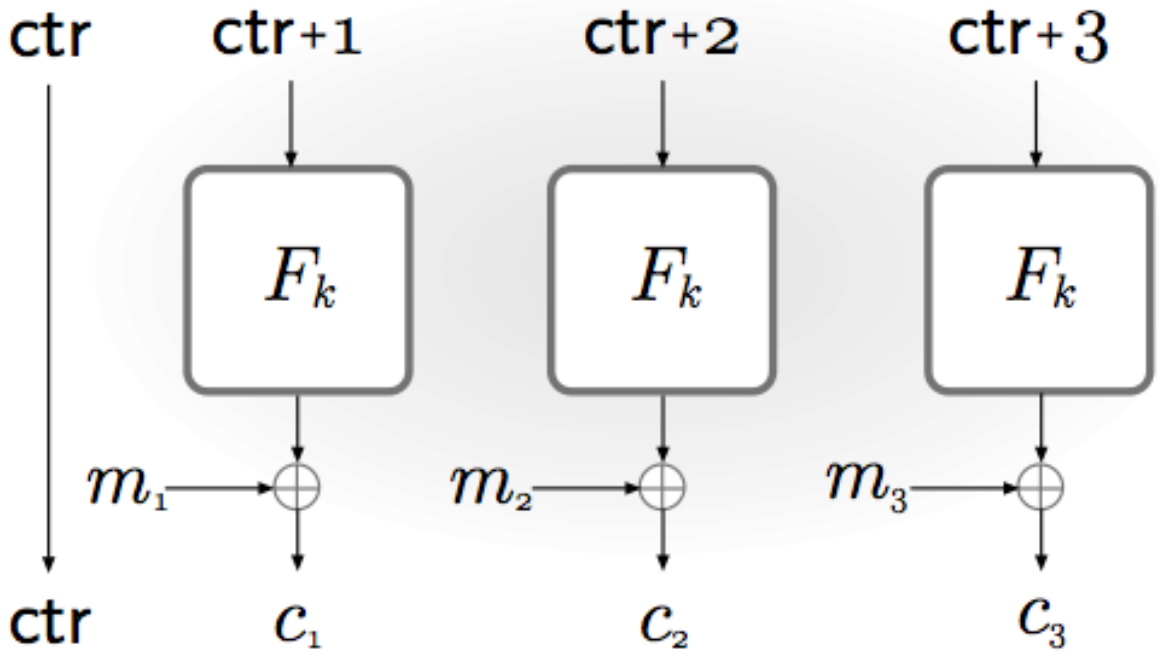
Modalitá Output Feedback Mode - OFB mode

Osservazioni:

- Non é necessario che F sia invertibile (neanche che sia una permutazione).
- Non é necessario che il messaggio abbia una lunghezza multipla di n. La stringa pseudocasuale puó essere troncata dove serve.
- La variante con stato (senza IV per cifrature successive) é sicura. Equivale ad uno stream cipher sincronizzato.
- La modalitá OFB é CPA-sicura se F é una funzione pseudocasuale.
- Il "pad" pseudocasuale puó essere pre-computato.

Modalitá Counter Mode - CTR mode

Puó essere vista come uno stream cipher asincrono costruito a partire da un cifrario a blocchi.



Alternativa: $y = F_k(ctr||< i >)$, con $ctr \in \{0,1\}^{3/4n}$ e $i \in \{0,1\}^{n/4}$.

Modalitá Counter Mode - CTR mode

Viene scelto uniformemente $ctr \in \{0,1\}^n$.

Si procede poi come segue

$$y_0 = ctr, \qquad y_i = F_k((ctr + i) \bmod 2^n), \qquad \text{per } i = 1, \ldots, \ell.$$

Ciascun blocco m_i del messaggio in chiaro viene cifrato calcolando

$$c_0 = y_0, \qquad c_i = m_i \oplus y_i, \text{ per } i = 1, \ldots, \ell.$$

La decifratura del cifrato $< c_0, c_1, \ldots, c_\ell >$ viene effettuata calcolando

$$ctr = c_0, \quad y_i = F_k((ctr+i) \bmod 2^n), \quad \text{e} \quad m_i = c_i \oplus y_i, \quad \text{per } i = 1, \ldots, \ell.$$

CTR mode - Analisi

Osservazioni

- La decifratura non richiede che F sia invertibile (né tantomeno che sia una permutazione)
- Lo stream generato puó essere troncato alla lunghezza del messaggio in chiaro e puó essere generato in anticipo.
- La variante "con stato" della modalitá CTR é sicura.
- CTR puó essere *parallelizzata*.
- La decifratura puó essere selettiva: decifrare direttamente l'i-esimo blocco del cifrato.

CTR mode - Analisi

Teorema 3.32 Se F é una funzione pseudocasuale, allora la modalitá CTR é CPA-sicura.

Dim. Utilizziamo la stessa strategia dimostrativa usata in precedenza. Sia $\widetilde{\Pi}= (\widetilde{Gen}, \widetilde{Enc}, \widetilde{Dec})$ costruito a partire da $\Pi = (Gen, Enc, Dec)$, tale che

- $\widetilde{\Pi}$ usa $f \in Func_n$ scelta uniformemente a caso
- Π usa F_k, dove k é scelta uniformemente a caso

Lo schema é naturalmente un'ipotesi di lavoro.

Per ogni Adv A PPT sia $q(n)$ un limite superiore al numero di query che $A(1^n)$ rivolge al suo oracolo per la cifratura.

CTR mode - Analisi

É facile far vedere (come prima) che esiste una funzione trascurabile $negl$ tale che

$$|Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] - Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n) = 1]| \leq negl(n).$$

Mostriamo quindi che

$$Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n) = 1] \leq 1/2 + \frac{2 \cdot q(n)^2}{2^n},$$

da cui risulta che

$$Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] \leq 1/2 + \frac{2 \cdot q(n)^2}{2^n} + negl(n).$$

Vedi il libro di testo per i dettagli.

Ulteriori commenti

- Le modalitá operative di cifratura vengono spesso comparate anche rispetto alla capacitá di protezione rispetto alla contraffazione dell'informazione.
  - Tuttavia la verifica dell'integritá richiede un tool differente
- La lunghezza del blocco nei cifrari a blocchi deve essere scelta con cura
  - Per esempio, nel CTR-mode, se il *ctr* é di ℓ bit, dopo $2^{\ell/2}$ *ctr* uniformi lo stesso valore ricompare. Il parametro ℓ non puó essere troppo corto.
- Se il vettore *IV* non é scelto uniformemente la sicurezza non é garantita
  - Se *IV* si ripete, con OFB e CTR un Adv puó recuperare informazioni importanti sul plaintext semplicemente calcolando l'xor dei due cifrati.
- Le versioni "con stato" di OFB e CTR sono sicure.

Schemi di cifratura nonce-based

Definizione 3.34. Uno schema di cifratura a chiave privata *nonce-based* è una tripla $\Pi = (Gen, Enc, Dec)$ di algoritmi PPT tale che

1. $k \leftarrow Gen(1^n)$, algoritmo probabilistico di generazione della chiave k
   - dove la chiave $k \in K$ è tale che $|k| \geq n$
2. $c := Enc_k(nonce, m)$, algoritmo deterministico di cifratura
   - dove $m \in \{0,1\}^*$, $nonce \in \{0,1\}^*$, $k \in K$ e $c \in \{0,1\}^*$
3. $m := Dec_k(nonce, c)$, algoritmo deterministico di decifratura
   - dove $c \in \{0,1\}^*$, $k \in K$, $nonce \in \{0,1\}^*$ ed $m \in \{0,1\}^*$
   - $Dec(nonce, c)$ restituisce $\perp$ in caso di errore

Correttezza. Per ogni n, per ogni k restituito da $Gen(1^n)$, per ogni $nonce \in \{0,1\}^*$ e per ogni $m \in \{0,1\}^*$, risulta

$$Dec_k(nonce, Enc_k(nonce, m)) = m$$

Schemi di cifratura nonce-based

Condizione: il nonce **deve** essere diverso per ogni messaggio.

La sicurezza si definisce come per i normali schemi di cifratura, modificando opportunamente gli esperimenti

- l'avversario A sceglie il *nonce*, che deve essere sempre diverso

La modalitá CTR, settando $crt = nonce$, produce uno schema di cifratura nonce-based CPA-sicuro.

Nota: Ogni schema nonce-based puó essere convertito in uno schema tradizionale scegliendo il nonce uniformemente a caso.

Vantaggi degli schemi nonce-based:

- utili nei contesti in cui generare randomness é difficile o troppo costoso
- utili per cifrare pochi messaggi, un nonce piccolo é sufficiente

Qualche esercizio su PRG, PRF e modalitá operative

1. Sia $G : \{0,1\}^n \to \{0,1\}^\ell$, con $\ell > n$, un PRG e sia $G' : \{0,1\}^{2n} \to \{0,1\}^{n+\ell}$ definito come

$$G'(r||s) = r||G(s),$$

con $r, s \in \{0,1\}^n$. É G' un PRG?

2. Relativamente all'Esempio 3.6, vi viene in mente un altro modo per costruire un distinguisher D efficiente ed altrettanto efficace?

3. Sia $F(k,x) = k \oplus x^c$, per qualche costante $c > 1$. É una PRF? E se sostituisco x^c con un certo polinomio $p(x)$?

4. Provate da soli la prima parte del Teorema 3.32 (rivedendo i passi della dimostrazione del Teorema 3.31, se necessario).