Cifratura autenticata

Paolo D'Arco

Obiettivo: cifratura autenticata

Abbiamo visto che:

- schemi di cifratura $\Rightarrow$ confidenzialitá/riservatezza
- codici per l'autenticazione di messaggi $\Rightarrow$ integritá/autenticitá

Vorremmo ottenere entrambi gli obiettivi, confidenzialitá ed autenticitá, insieme.

Sia $\Pi = (Gen, Enc, Dec)$ uno schema di cifratura a chiave privata.

- la nozione di segretezza a cui siamo interessati é rispetto ad attacchi CCA
- la nozione di integritá a cui siamo interessati é "non falsificabile esistenzialmente rispetto ad attacchi di tipo chosen message"

Purtroppo Π non soddisfa la sintassi di un codice per l'autenticazione di messaggi. Pertanto, occorre una definizione specifica.

Cifratura non falsificabile

Sia $\Pi = (Gen, Enc, Dec)$, A un avversario, n parametro di sicurezza.

The unforgeable encryption experiment $\mathsf{Enc\text{-}Forge}_{\mathcal{A},\Pi}(n)$**:**

1. *Run* $\mathsf{Gen}(1^n)$ *to obtain a key* k.
2. *The adversary* $\mathcal{A}$ *is given input* 1^n *and access to an encryption oracle* $\mathsf{Enc}_k(\cdot)$. *The adversary outputs a ciphertext* c.
3. *Let* $m := \mathsf{Dec}_k(c)$, *and let* $\mathcal{Q}$ *denote the set of all queries that* $\mathcal{A}$ *asked its encryption oracle. The output of the experiment is* 1 *if and only if (1)* $m \neq \perp$ *and (2)* $m \notin \mathcal{Q}$.

Cifratura non falsificabile

Definizione 4.16. Uno schema di cifratura a chiave privata Π é non falsificabile (unforgeable) se, per ogni avversario A ppt, esiste una funzione trascurabile *negl* tale che

$$Pr[\text{Enc-forge}_{A,\Pi}(n) = 1] \leq negl(n).$$

Nota: possiamo considerare una definizione piú forte in cui A ha anche accesso all'oracolo di decifratura. La costruzione proposta nel seguito la soddisfa.

Definizione 4.17. Uno schema di cifratura a chiave privata Π é uno schema di cifratura autenticata se é CCA-sicuro e non falsificabile.

Come possiamo costruire schemi di cifratura autenticata?

Cifratura autenticata

Ogni generica combinazione di uno schema di cifratura e di un codice per l'autenticazione di messaggi sicuri funziona?

... purtroppo no!

Siano:

- $\Pi_E = (Gen, Enc, Dec) \rightarrow$ schema di cifratura CPA-sicuro
- $\Pi_M = (Mac, Vrfy) \rightarrow$ codice per l'autenticazione di messaggi sicuro

Esistono tre combinazioni naturali:

1. **Cifra ed autentica**. Il mittente trasmette il cifrato $< c, t >$ dove

$$c \leftarrow Enc_{k_E}(m) \quad \text{e} \quad t = Mac_{k_M}(m)$$

Cifratura autenticata

Il ricevente decifra c per recuperare m. Poi verifica il tag t. Restituisce m se non si verificano errori, $\perp$ altrimenti.

2. **Autentica e poi cifra**. Il mittente trasmette il cifrato $< c >$ calcolato come

$$t = Mac_{k_M}(m) \quad \text{e} \quad c \leftarrow Enc_{k_E}(m||t)$$

Il ricevente decifra c per recuperare $m||t$. Poi verifica il tag t. Restituisce m se non si verificano errori, $\perp$ altrimenti.

3. **Cifra e poi autentica**. Il mittente trasmette il cifrato $< c, t >$ calcolato come

$$c \leftarrow Enc_{k_E}(m) \quad \text{e} \quad t = Mac_{k_M}(c)$$

Cifratura autenticata

Il ricevente verifica il tag t. Se risulta corretto, decifra c e dá in output m. Altrimenti, restituisce $\perp$.

Analizziamo i tre approcci quando instanziati con "componenti generiche": Π_E CPA-sicuro e Π_M fortemente sicuro

1. **Cifra e autentica**. Purtroppo non garantisce neanche i requisiti minimali di segretezza. Infatti:
   - il Mac non garantisce alcuna segretezza di per sé
   - puó rilasciare informazioni importanti su m, e.g., CBC-mac é deterministico $\Rightarrow$ ogni volta che lo stesso messaggio viene cifrato, il tag rimane inalterato

     $$< c, t >, \qquad < c', t > \qquad \text{non é piú CPA-sicuro!}$$

     Poiché la maggior parte dei Mac é deterministica, la preoccupazione é reale.

Cifratura autenticata

2. **Autentica e poi cifra**. Abbiamo visto nelle lezioni passate uno schema di cifratura CPA-sicuro per il quale un attacco CCA permetteva di recuperare l'intero messaggio: CBC-mode con padding $\Rightarrow$ Padding-oracle attack.

   Se viene usato, la decifratura nello schema di cifratura autenticata $Dec'_{k_E,k_M}(c)$ funziona al modo seguente

   - Calcola $\overline{m} = Dec_{k_E}(Enc_{k_E}(m||t))$. Se viene riscontrato un errore nel padding di $m||t$, allora restituisce "padding scorretto" e termina
   - Divide $\overline{m} = m||t$. Se $Vrfy(m,t) = 1$, restituisce m. Altrimenti dá in output "autenticazione fallita".

Un'implementazione del genere di Dec' ha due messaggi d'errore differenti. Usando il primo, possiamo applicare il padding-oracle attack ottenendo $m||t$.

Cifratura autenticata

Patch: dare un singolo messaggio di errore.

Contro: puó essere utile mantenere messaggi diversi

- usabilitá, debugging
- complessitá d'uso nelle applicazioni
- é difficile assicurare che errori di tipo diverso non possano essere distinti attraverso side-channel attack

Alcune versioni di SSL usavano l'approccio precedente e furono rotte attraverso attacchi basati sulla misurazione del tempo (timing attack).

3. **Cifra e poi autentica**. Funziona per **ogni** generica instanziazione delle primitive, assumendo che il Mac sia **fortemente** sicuro

Cifratura autenticata

CONSTRUCTION 4.18

Let $\Pi_E = (\mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme and let $\Pi_M = (\mathsf{Mac}, \mathsf{Vrfy})$ be a message authentication code, where in each case key generation is done by simply choosing a uniform n-bit key. Define a private-key encryption scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- Gen': on input 1^n, choose independent, uniform $k_E, k_M \in \{0,1\}^n$ and output the key (k_E, k_M).
- Enc': on input a key (k_E, k_M) and a plaintext message m, compute $c \leftarrow \mathsf{Enc}_{k_E}(m)$ and $t \leftarrow \mathsf{Mac}_{k_M}(c)$. Output the ciphertext $\langle c, t \rangle$.
- Dec': on input a key (k_E, k_M) and a ciphertext $\langle c, t \rangle$, first check whether $\mathsf{Vrfy}_{k_M}(c, t) \stackrel{?}{=} 1$. If yes, then output $\mathsf{Dec}_{k_E}(c)$; if no, then output $\perp$.

Cifratura autenticata

La sicurezza forte del Mac implica due cose:

1. A non puó produrre t' su qualsiasi nuovo c'
   - $\Rightarrow (Gen', Enc', Dec')$ é non falsificabile
2. A non puó produrre $< c, t >$ di cui chiedere la decifratura all'oracolo in $Priv^{cca}_{A,\Pi}$
   - $\Rightarrow$ l'oracolo di decifratura $Dec'(\cdot)$ é inutile
   - $\Rightarrow A$ agisce esattamente come in $Priv^{cpa}_{A,\Pi}$

Formalmente, vale il seguente:

Teorema 4.19. Sia Π_E uno schema di cifratura CPA-sicuro a chiave privata e sia Π_M un codice di autenticazione di messaggi fortemente sicuro. Allora la Costruzione 4.18 produce uno schema di cifratura autenticata.

Dimostrazione

Sia $\Pi' = (Gen', Enc', Dec')$ lo schema risultante dalla Costruzione 4.18. Dobbiamo provare che é

- non falsificabile
- *CCA*-sicuro

Diremo $< c, t >$ *valido* rispetto a (k_E, k_M) se $Vrfy_{k_M}(c, t) = 1$

Sia A un Adv ppt che sferra un attacco di tipo *CCA* contro Π'.

Diremo che $< c, t >$ é *nuovo* se A non ha ricevuto $< c, t >$ dall'oracolo di cifratura.

Sia $ValidQuery =$ "A sottomette un cifrato nuovo $< c, t >$ all' oracolo di decifratura che risulta valido".

Dimostrazione

Claim 4.20. La $Pr[ValidQuery]$ é trascurabile.

Intuizione: una query valida $\Rightarrow$ A ha falsificato una nuova coppia $< c, t >$ valida in $Mac\text{-}sforge(n)$.

Sia $q(n)$ un limite superiore polinomiale al numero di query fatte da A all'oracolo.

Consideriamo una Adv A_M che attacca Π_M, sfruttando la capacitá di A di produrre $< c, t >$ valido.

A_M "gioca" all'interno di $Mac\text{-}sforge(n)$ e simula l'esperimento $Priv^{cca}_{A,\Pi'}(n)$.

Precisamente, A_M riceve in input 1^n e accesso all'oracolo $Mac_{k_M}(\cdot)$, e agisce come segue:

Dimostrazione

Adversary $\mathcal{A}_M$:
$\mathcal{A}_M$ is given input 1^n and has access to a MAC oracle $\mathsf{Mac}_{k_M}(\cdot)$.

1. Choose uniform $k_E \in \{0,1\}^n$ and $i \in \{1,\ldots,q(n)\}$.
2. Run $\mathcal{A}$ on input 1^n. When $\mathcal{A}$ makes an encryption-oracle query for the message m, answer it as follows:
   (i) Compute $c \leftarrow \mathsf{Enc}_{k_E}(m)$.
   (ii) Query c to the MAC oracle and receive t in response. Return $\langle c,t\rangle$ to $\mathcal{A}$.

   The challenge ciphertext is prepared in the exact same way (with a uniform bit $b \in \{0,1\}$ chosen to select the message m_b that gets encrypted).

   When $\mathcal{A}$ makes a decryption-oracle query for the ciphertext $\langle c,t\rangle$, answer it as follows: If this is the ith decryption-oracle query, output (c,t). Otherwise:

   (i) If $\langle c,t\rangle$ was a response to a previous encryption-oracle query for a message m, return m.
   (ii) Otherwise, return $\perp$.

Dimostrazione

A_M "scommette" che la i-esima richiesta di cifratura all'oracolo che A effettua nell'esperimento simulato $PrivK^{cca}_{A,\Pi'}(n)$ é la **prima** nuova e valida query che A invia.

Se indovina, A_M dá in output un tag valido su un messaggio c che non é stato mai sottoposto all'oracolo $Mac_{k_M}(\cdot)$. Pertanto A_M vince!

A_M computa in ppt (se A computa in ppt).

Qual é la probabilitá di successo di A_M?

La vista di A quando eseguito come subroutine di A_M é distribuita esattamente come in $PrivK^{cca}_{A,\Pi'}(n)$ fino a quando occorre *ValidQuery*.

- le query di cifratura sono perfettamente simulate da A_M
- le query di decifratura, fino a *ValidQuery*, sono perfettamente simulate
  - se $< c, t >$ é nuovo e *ValidQuery* non si verifica, la decifratura é $\perp$

Dimostrazione

Pertanto, $Pr[ValidQuery]$ in $Mac\text{-}sforge(n)$ é identica a $Pr[ValidQuery]$ in $PrivK^{cca}_{A,\Pi'}(n)$. Poiché:

$$Pr[A_M \text{ indovina } i] = \frac{1}{q(n)}$$

risulta:

$$Pr[Mac\text{-}sforge_{A_M,\Pi_M}(n) = 1] \geq \frac{Pr[ValidQuery]}{q(n)}.$$

Ma, per ipotesi, Π_M é fortemente sicuro. Quindi:

$$Pr[Mac\text{-}sforge_{A_M,\Pi_M}(n) = 1] \leq negl(n).$$

Discende che:

$$Pr[ValidQuery] \leq Pr[Mac\text{-}sforge_{A_M,\Pi_M}(n) = 1] \cdot q(n) \leq negl'(n). \quad \square$$

Dimostrazione

Il Claim 4.20 puó ora essere usato per provare che Π' é sicuro.

Parte facile: dal claim segue che Π' é *non falsificabile*.

A' in $\mathit{Enc\text{-}forge}_{A',\Pi'}(n)$ é una versione "ristretta" di A in $\mathit{PrivK}^{cca}_{A,\Pi'}(n)$

- usa soltanto l'oracolo di cifratura

Pertanto, quando A' dá in output $< c, t >$, ha successo solo se é valido e nuovo.

$\Rightarrow$ il Claim 4.20 ha mostrato che accade con prob. trascurabile.

Parte difficile da mostrare: Π' é CCA-sicuro. Sia A un Adv ppt che attacca Π' con un attacco CCA. Risulta: $Pr[\mathit{PrivK}^{cca}_{A,\Pi'}(n) = 1]$

$$
\begin{aligned}
&= Pr[\mathit{PrivK}^{cca}_{A,\Pi'}(n) = 1 \wedge \mathit{ValidQuery}] + Pr[\mathit{PrivK}^{cca}_{A,\Pi'}(n) = 1 \wedge \overline{\mathit{ValidQuery}}] \\
&\leq Pr[\mathit{ValidQuery}] + Pr[\mathit{PrivK}^{cca}_{A,\Pi'}(n) = 1 \wedge \overline{\mathit{ValidQuery}}].
\end{aligned}
$$

Dimostrazione

Claim 4.21. Esiste una funzione trascurabile $negl(n)$ tale che

$$Pr[PrivK^{cca}_{A,\Pi'}(n) = 1 \wedge \overline{ValidQuery}] \leq 1/2 + negl(n).$$

Proviamo il claim usando la sicurezza CPA di Π_E. Definiamo A_E che usa A contro Π' per vincere $PrivK^{cpa}_{A_E,\Pi_E}(n)$.

Adversary $\mathcal{A}_E$:
$\mathcal{A}_E$ is given input 1^n and has access to $\mathsf{Enc}_{k_E}(\cdot)$.

1. Choose uniform $k_M \in \{0,1\}^n$.
2. Run $\mathcal{A}$ on input 1^n. When $\mathcal{A}$ makes an encryption-oracle query for the message m, answer it as follows:
   (i) Query m to $\mathsf{Enc}_{k_E}(\cdot)$ and receive c in response.
   (ii) Compute $t \leftarrow \mathsf{Mac}_{k_M}(c)$ and return $\langle c, t\rangle$ to $\mathcal{A}$.

   When $\mathcal{A}$ makes a decryption-oracle query for the ciphertext $\langle c, t\rangle$, answer it as follows:

Dimostrazione

- If $\langle c, t \rangle$ was a response to a previous encryption-oracle query for a message m, return m. Otherwise, return $\perp$.

3. When $\mathcal{A}$ outputs messages (m_0, m_1), output these same messages and receive a challenge ciphertext c in response. Compute $t \leftarrow \mathsf{Mac}_{k_M}(c)$, and return $\langle c, t \rangle$ as the challenge ciphertext for $\mathcal{A}$. Continue answering $\mathcal{A}$'s oracle queries as above.
4. Output the same bit b' that is output by $\mathcal{A}$.

A_E esegue in tempo polinomiale

A_E non ha bisogno di un oracolo di decifratura: assume che ogni decifratura che non corrisponde ad una query di cifratura precedente sia invalida.

La vista di A quando esegue come subroutine di A_E é identicamente distribuita alla vista in $PrivK^{cca}_{A,\Pi'}(n)$ *fino a quando ValidQuery* non si verifica.

Dimostrazione

Pertanto:

$$Pr[PrivK^{cpa}_{A_E,\Pi_E}(n) = 1 \wedge \overline{ValidQuery}] = Pr[PrivK^{cca}_{A,\Pi'}(n) = 1 \wedge \overline{ValidQuery}]$$

Discende che:

$$\begin{aligned} Pr[PrivK^{cpa}_{A_E,\Pi_E}(n) = 1] &\geq Pr[PrivK^{cpa}_{A_E,\Pi_E}(n) = 1 \wedge \overline{ValidQuery}] \\ &= Pr[PrivK^{cca}_{A,\Pi'}(n) = 1 \wedge \overline{ValidQuery}]. \end{aligned}$$

Poiché, per ipotesi, Π_E é CPA-sicuro, $\exists$ $negl(n)$ tale che

$$Pr[PrivK^{cpa}_{A_E,\Pi_E}(n) = 1] \leq \frac{1}{2} + negl(n).$$

$$\Rightarrow \quad Pr[PrivK^{cca}_{A,\Pi'}(n) = 1 \wedge \overline{ValidQuery}] \leq \frac{1}{2} + negl(n). \qquad \square$$

Necessitá di chiavi indipendenti

"Istanze differenti di primitive crittografiche dovrebbero sempre usare chiavi indipendenti"

Consideriamo un esempio in cui applichiamo l'approccio "cifra e poi autentica" in cui la **stessa** chiave viene usata

$$\text{Sia } F \text{ una PRP forte} \quad \Rightarrow \quad F^{-1} \text{ é una PRF forte}$$

Si consideri lo schema definito da

- $F_k : \{0,1\}^n \to \{0,1\}^n$
- $Enc_k(m) = F_k(m||r)$, dove $m \leftarrow \{0,1\}^{n/2}, r \leftarrow \{0,1\}^{n/2}$
- $Mac_k(c) = F_k^{-1}(c)$

Necessitá di chiavi indipendenti

É possibile mostrare che:

1. lo schema di cifratura é CPA-sicuro
2. il Mac é fortemente sicuro
3. la combinazione usando la stessa chiave k **non** lo é!

Infatti,

$$[Enc_k(m) = F_k(m||r), Mac_k(Enc_k(m)) = F_k^{-1}(F_k(m||r))] \quad = \quad [F_k(m||r), m||r]$$

Nella costruzione 4.18 si richiede che le chiavi siano scelte indipendentemente l'una dall'altra ed in modo uniforme.

Sessioni di comunicazione sicure

"Un periodo di tempo durante il quale le parti che comunicano mantengono uno stato e desiderano comunicare in modo sicuro"

Uno schema di cifratura autenticata da solo **non** é sufficiente.

- **Attacco di riordino.**
  - *Alice* trasmette c_1, c_2, Adv intercetta e trasmette c_2, c_1
- **Attacco di replay.**
  - Adv invia piú volte c
- **Attacco di riflessione.**
  - Adv invia ad *Alice* un cifrato c che in precedenza *Alice* ha inviato a *Bob*

Soluzione

Bit di direzionalitá: nelle comunicazioni, il bit $b_{X,Y}$ indica il verso, da X verso Y

Contatori: il valore $ctr_{X,Y}$ conta i messaggi che X ha inviato a Y.

- $A \rightarrow B \qquad c \leftarrow Enc_k(b_{A,B}||ctr_{A,B}||m)$
- $B \rightarrow A \qquad c \leftarrow Enc_k(b_{B,A}||ctr_{B,A}||m)$
- $\text{Stato}_A = \text{Stato}_B = (ctr_{A,B}, ctr_{B,A})$

Cifratura CCA-sicura

Uno schema di cifratura autenticata é CCA-sicuro

Esistono schemi di cifratura a chiave privata che risultano CCA-sicuri ma non infalsificabili?

Sí!

$$F_k\{0,1\}^n \leftarrow \{0,1\}^n$$

$$Enc_k(m) = F_k(m||r), \quad m \in \{0,1\}^{\frac{n}{2}}, \quad r \leftarrow \{0,1\}^{\frac{n}{2}}$$

Sono utili? In alcune applicazioni in cui l'integritá non é una preoccupazione, potrebbero.

In realtá, poiché con le costruzioni attualmente note non c' é alcun guadagno in termini di efficienza, conviene usare uno schema di cifratura autenticata.

Da un punto di visto concettuale é peró utile distinguere.