CCA-sichere Verschlüsselung

Idee:

- Der Verschlüsseler authentisiert c mit Hilfe eines MACs t.
- D.h. nur er ist in der Lage, gültige Paare (c, t) zu erzeugen.
- Damit ist ein CCA-Entschlüsselungsorakel für Angreifer nutzlos.

Algorithmus CCA-sichere Verschlüsselung Π_{cca}

Sei $\Pi_E = (Gen_E, Enc, Dec)$ ein Verschlüsselungsverfahren und $\Pi_M = (Gen_M, Mac, Vrfy)$ ein MAC.

1. **Gen':** $k_1 \leftarrow Gen_E(1^n)$, $k_2 \leftarrow Gen_M(1^n)$.
2. **Enc':** Bei Eingabe von m und (k_1, k_2), berechne $c \leftarrow Enc_{k_1}(m)$ und $t \leftarrow Mac_{k_2}(c)$. Ausgabe des Chiffretextes (c, t).
3. **Dec':** Bei Eingabe von (c, t) und (k_1, k_2),

$$\text{Ausgabe} = \begin{cases} m := Dec_{k_1}(c) & \text{falls } Vrfy_{k_2}(c, t) = 1 \\ \perp & \text{sonst} \end{cases}.$$

Eindeutige Tags

Definition MAC mit eindeutigem Tag

Sei $\Pi_M = (Gen, Mac, Vrfy)$ ein MAC. Π_M besitzt *eindeutige Tags* falls für alle k, m genau ein t existiert mit $Vrfy_k(m, t) = 1$.

Anmerkungen:

- D.h. der *Mac*-Algorithmus ist deterministisch.
- Bsp: Π_{MAC}, CBC-MAC, NMAC, HMAC besitzen eindeutige Tags.
- Π_{MAC2} besitzt keine eindeutigen Tags.

Π_{cca} ist CCA-sicher

Satz CCA-Sicherheit von Π_{cca}

Sei Π_E CPA-sicher und Π_M ein sicherer MAC mit eindeutigen Tags. Dann ist Π_{cca} CCA-sicher.

Beweisskizze:

- Offenbar ist Π_{cca} sicher gegenüber CPA-Angreifern $\mathcal{A}$.
- Wir zeigen nun, dass ein $Dec(\cdot)$-Orakel für $\mathcal{A}$ nutzlos ist.
- Sei (c, t) eine Anfrage von $\mathcal{A}$ an $Dec(\cdot)$.

Fall 1: (c, t) kommt aus voriger $Enc(m)$-Anfrage von $\mathcal{A}$.

- Dann weiss $\mathcal{A}$ bereits, dass $Dec(c, t)$ die Antwort m liefert.
- D.h. das Entschlüsselsorakel liefert keine nützliche Information.

Fall 2: (c, t) kommt nicht aus $Enc(m)$-Anfrage.

- Falls $Vrfy_{k_2}(c, t) = 1$, hat $\mathcal{A}$ einen gültigen Tag t für ein neues c konstruiert (folgt aus der Eindeutigkeit der Tags).
- Aufgrund der MAC-Sicherheit geschieht dies mit Ws $\leq \mathrm{negl}(n)$.
- D.h. $Dec(\cdot)$ gibt mit Ws $\geq 1 - \mathrm{negl}(n)$ die nutzlose Ausgabe $\perp$.

Verfahren zur Nachrichtenübermittlung

Ziel: *Vertraulichkeit* und *Integrität* der Nachricht

Definition Nachrichtenübermittlung

Sei $\Pi_E = (Gen_E, Enc, Dec)$ ein Verschlüsselungsverfahren und $\Pi_M = (Gen_M, Mac, Vrfy)$ ein Mac. Ein *Nachrichtenübermittlungsverfahren* $\Pi' = (Gen', EncMac', Dec')$ besteht aus

1. **Gen':** $k_1 \leftarrow Gen_E(1^n)$, $k_2 \leftarrow Gen_M(1^n)$
2. **EncMac':** Bei Eingabe von m und (k_1, k_2), berechne mittels Enc_{k_1} und Mac_{k_2} einen authentisierten Chiffretext γ.
3. **Dec':** Bei Eingabe von γ und (k_1, k_2), berechne mittels Dec_{k_1} und $Vrfy_{k_2}$ einen Klartext m oder eine Fehlerausgabe $\perp$. Es gilt

$$Dec'_{k_1,k_2}(EncMac'_{k_1,k_2}(m)) = m \text{ für alle } (k_1, k_2) \text{ und } m \in \{0,1\}^*.$$

Sicherheitsspiel der Nachrichtenübermittlung

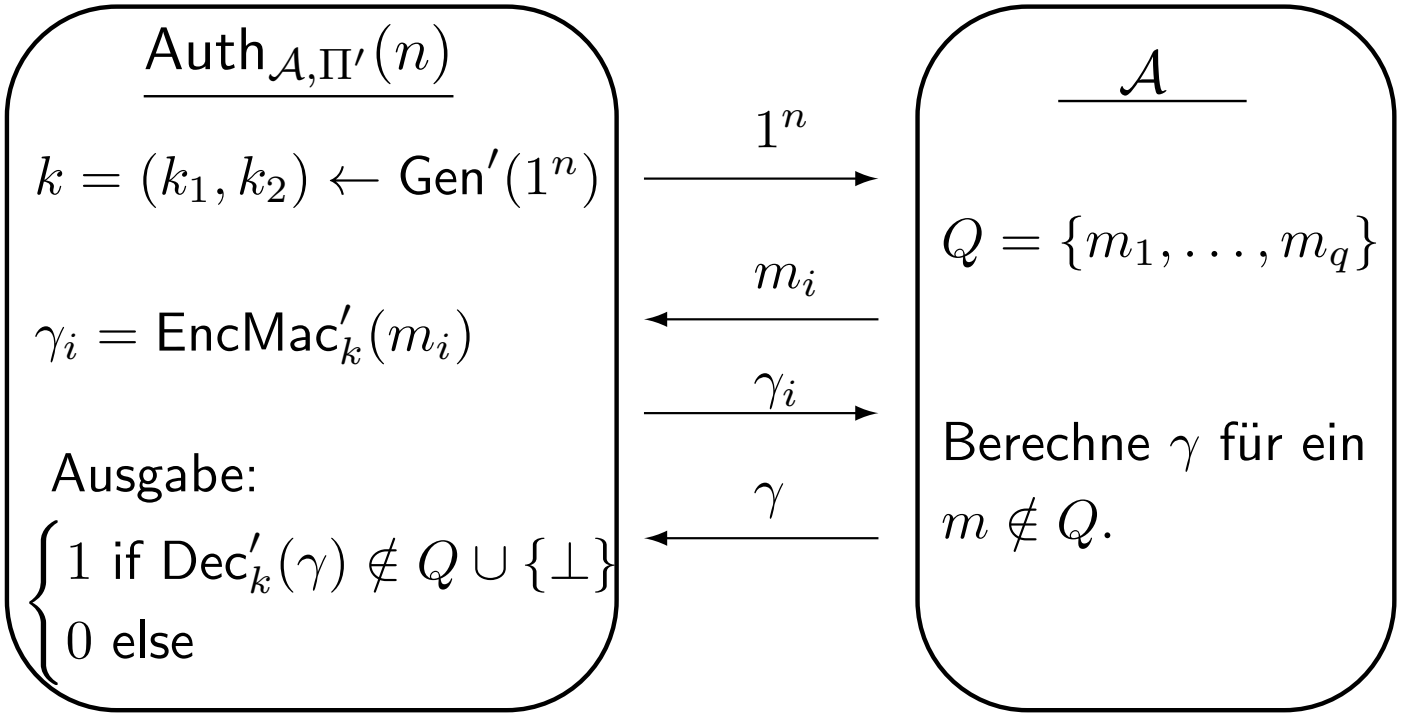
Spiel Nachrichtenübermittlung $Auth_{\mathcal{A},\Pi'}(n)$

Sei $\mathcal{A}$ ein Angreifer für $\Pi' = (Gen', EncMac', Dec')$.

1. $k = (k_1, k_2) \leftarrow Gen'(1^n)$
2. $\gamma \leftarrow \mathcal{A}^{EncMac'_k(\cdot)}(1^n)$, d.h. $\mathcal{A}$ darf $EncMac_k(m)$ für beliebige m anfragen.
3. Sei Q die Menge der $EncMac_k(\cdot)$-Anfragen und $m := Dec'_k(\gamma)$.

$$Auth_{\mathcal{A},\Pi'}(n) = \begin{cases} 1 & \text{falls } m \neq \perp \text{ und } m \notin Q \\ 0 & \text{sonst} \end{cases}.$$

Sicherheitsspiel der Nachrichtenübermittlung



Sicherheit eines Nachrichtenübertragungsverfahrens

Definition Authentisierte Kommunikation

Ein Nachrichtenübertragungsverfahren Π' liefert *authentisierte Kommunikation* falls für alle ppt Angreifer $\mathcal{A}$ gilt

$$\mathrm{Ws}[\mathit{Auth}_{\mathcal{A},\Pi'}(n) = 1] \leq \mathrm{negl}(n).$$

Definition Sicherheit eines Nachrichtenübertragungsverfahrens

Sei Π' ein Nachrichtenübertragungsverfahren. Π' heißt *sicher*, falls es CCA-sicher ist und authentisierte Kommunikation liefert.

Sicherheit von Encrypt-then-authenticate Π_{cca}

Satz Sicherheit von Π_{cca}

Sei Π_E CPA-sicher und Π_M ein sicherer MAC mit eindeutigen Tags. Dann ist Π_{cca} ein sicheres Nachrichtenübermittlungsverfahren.

Beweis:

- Die CCA-Sicherheit von Π_{cca} wurde bereits gezeigt.
- Sei $\mathcal{A}$ ein Angreifer im Spiel $Auth_{\mathcal{A},\Pi_{cca}}(n)$ mit Erfolgsws $\epsilon(n)$.
- Wir konstruieren daraus einen Angreifer $\mathcal{A}'$ für Π_M.

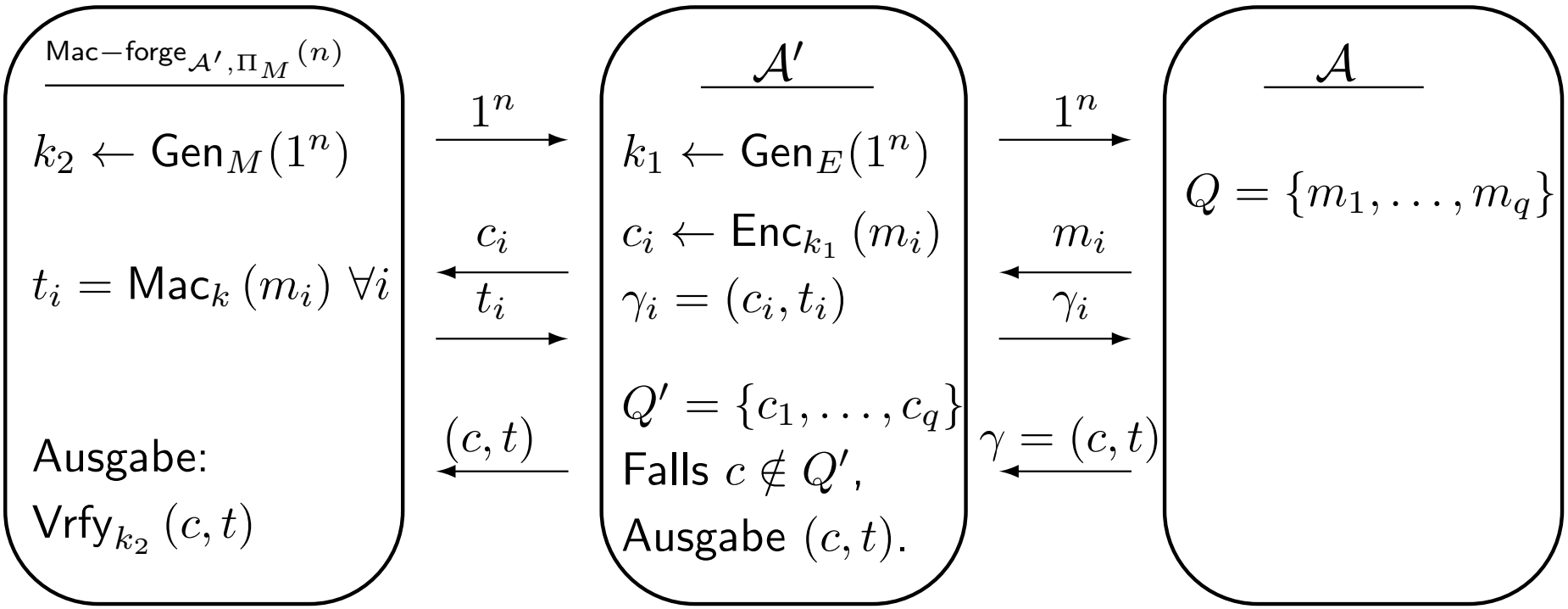
Algorithmus Angreifer $\mathcal{A}'$ für Π_M

EINGABE: 1^n, Orakelzugriff auf $Mac_{k_2}(\cdot)$

1. $k_1 \leftarrow Gen_E(1^n)$
2. $\gamma = (c', t') \leftarrow \mathcal{A}^{EncMac'_{k_1,k_2}(\cdot)}(1^n)$, bei $EncMac'_{k_1,k_2}(m)$-Anfrage berechne $c \leftarrow Enc_{k_1}(m)$, $t = Mac_{k_2}(c)$ und antworte mit (c, t).

AUSGABE: $\gamma = (c', t')$

Sicherheit von Encrypt-then-authenticate Π_{cca}



Sicherheit von Encrypt-then-authenticate Π_{cca}

Beweis: Fortsetzung

- Sei $Q = \{m_1, \ldots, m_q\}$ die Menge der $EncMac_{k_1,k_2}(\cdot)$-Anfragen.
- Seien c_i die Verschlüsselungen von m_i für $i = 1, \ldots, q$ in Schritt 2.
- Falls $\mathcal{A}$ Erfolg hat, so gilt $m := Dec_{k_1,k_2}(c) \notin Q$.
- Daraus folgt $c' \notin \{c_1, \ldots, c_q\} = Q'$. D.h. t' ist ein Tag für eine nicht an das $Mac_{k_2}(\cdot)$-Orakel angefragte Nachricht c.
- Es folgt

$$\mathrm{Ws}[\text{Mac-forge}_{\mathcal{A}',\Pi_M}(n) = 1] \geq \mathrm{Ws}[Auth_{\mathcal{A},\Pi_{cca}}(n) = 1] = \epsilon(n).$$

- Aus der Sicherheit von Π_M folgt $\epsilon(n) \leq \mathrm{negl}(n)$.

Anmerkung:

- Π_{cca} ist sicher für *jedwede* sichere Instantiierung von Π_E und Π_M.

Die Notwendigkeit zweier Schlüssel k_1, k_2

Faustregel Verwendung verschiedener Schlüssel

Verschiedene Sicherheitsziele sollten durch Wahl verschiedener Schlüssel realisiert werden.

Bsp: Unsicheres Encrypt-then-authenticate durch einen Schlüssel

- Wir verwenden denselben Schlüssel k für Π_E und Π_M.
- Sei F eine starke Pseudozufallspermutation auf n Bits.
- D.h. F^{-1} ist ebenfalls eine starke Pseudozufallspermutation.
- Wir konstruieren ein CPA-sicheres Π_E (Übung) mittels
$$Enc_k(m) = F_k(m||r) \text{ für } m \in \{0,1\}^{\frac{n}{2}}, r \in_R \{0,1\}^{\frac{n}{2}}.$$
- Wir konstruieren einen sicheren MAC Π_M mittels
$$Mac_k(c) = F_k^{-1}(c) \text{ für } c \in \{0,1\}^n.$$
- Encrypt-then-authenticate liefert
$$\gamma = (c,t) = (F_k(m||r), F_k^{-1}(F_k(m||r))) = (c, m||r).$$
- D.h. γ gibt die Nachricht m preis.

Encrypt-and-authenticate kann unsicher sein

Encrypt-and-authenticate:

$$\gamma = EncMac(m) := (c, t) = (Enc_{k_1}(m), Mac_{k_2}(m)).$$

- D.h. der Tag wird für die Nachricht m berechnet, nicht für c.
- Sei Π_E CPA-sicher und $\Pi_M = (Gen_M, Mac, Vrfy)$ ein sicherer Mac.
- Dann ist Π'_M mit $Mac'_{k_2}(m) = (m, Mac_k(m))$ ebenfalls sicher, denn gültige Tags (m, t) für Π'_M liefern gültige Tags t für Π_M.
- Instantiierung von Encrypt-and-authenticate mit Π_E, Π'_M liefert
$$\gamma = (c, (m, Mac_{k_2}(m))).$$
- D.h. γ gibt die Nachricht m preis, obwohl Π_E und Π'_M sicher sind.

Authenticate-then-encrypt kann unsicher sein

Authenticate-then-encrypt: $\gamma = EncMac(m) := Enc_{k_1}(m||Mac_{k_2}(m))$

- Kodieren Nachricht $m \in \{0,1\}^*$ vor Verschlüsselung:
  Jede 0 wird als 00 oder 11 kodiert, jede 1 als 01 oder 10.
- Dekodierung in Zweierblöcken: 00 oder 11 zu 0, 01 oder 10 zu 1.
- Wir verschlüsseln $Enc_k(m) = Enc'_k(\text{Kodierung}(m))$, wobei Enc' die CPA-sichere Verschlüsselung im Counter-Modus ist. Erinnerung:
  $Enc'(m_1 \dots m_\ell) = (ctr, m_1 \oplus r_1, \dots, m_\ell \oplus r_\ell)$ mit $r_i = F_k(ctr + i \bmod 2^n)$.

Algorithmus CCA-Angreifer

EINGABE: $c = Enc'_k(\text{Kodierung}(m||Mac_{k_2}(m)))$, $Dec_k(\cdot)$-Orakel

Für alle Zweierblöcke in c, die eine Kodierung von m enthalten:

1. Berechne c' durch Flippen eines Zweierblocks in c.
2. Falls $Dec_k(c') \in \{00, 11\}$, entschlüssele den Zweierblock zu 0.
3. Falls $Dec_k(c') \in \{01, 10\}$, entschlüssele den Zweierblock zu 1.

AUSGABE: m

Authenticate-then-encrypt kann unsicher sein

Fall 1: Zweierblock entspricht Kodierung 00 oder 11 eines Bits $m_j = 0$

- Flippen wechselt zwischen den zwei Kodierungen von $m_j = 0$.
- Damit erhält man bei der Dekodierung noch immer eine 0.
- $Mac_{k_2}(m)$ bleibt gültig, da nur die Kodierung von m geändert wird, nicht m selbst. Mac_{k_2} wird nicht auf $\mathrm{Kodierung}(m)$ angewendet.

Fall 2: Zweierblock entspricht Kodierung 01 oder 10 eines Bits $m_j = 1$

- Argumentation ist analog zum obigen Fall.

Anmerkungen:

- Bsp. zeigt, dass Authenticate-then-encrypt i. allg. nicht sicher ist.
- Das SSL (Secure Sockets Layer) Protokoll im Internet verwendet eine sichere Variante von Authenticate-then-encrypt.