Vorlesung Sicherheit

Dennis Hofheinz

ITI, KIT

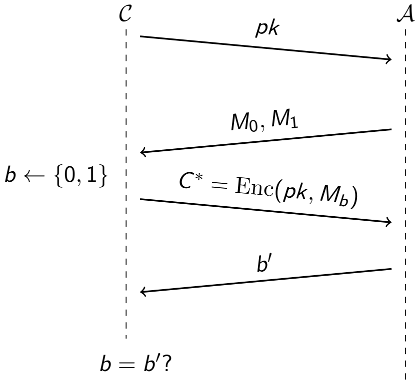
22.05.2017

Überblick

Überblick

IND-CPA für asymm. Verschlüsselung

- Herausforderer $\mathcal{C}$ erzeugt Schlüsselpaar (pk, sk).
- Kein Enc-Orakel, stattdessen erhält der Angreifer pk.



Erinnerung

- Asymmetrische (Public-Key-)Verschlüsselung:

$$\texttt{Alice}_{sk} \xleftarrow{\quad C:=\mathrm{Enc}(pk,M) \quad} \texttt{Bob}_{pk}$$

- RSA:

$$\mathrm{Enc}(pk, M) = M^e \bmod N \qquad \mathrm{Dec}(sk, C) = C^d \bmod N$$

Überblick

Sicherheit von RSA

- RSA nicht semantisch sicher
  - $f(M) = M^e \bmod N$ kann mit Chiffrat berechnet werden
  - Aber ohne Chiffrat keine Information über M
  - „Angriff“ nutzt aus, dass RSA deterministisch
- Intuitiv überzeugender: beispielsweise

$$\mathrm{Enc}(pk, \mathtt{annehmen}) \quad \text{und} \quad \mathrm{Enc}(pk, \mathtt{ablehnen})$$

bei RSA effizient unterscheidbar (keine IND-CPA-Sicherheit)

Weitere Angriffe auf RSA

- Was, wenn $e = 3$ (aus Effizienzgründen) für alle Benutzer?
  - Problem, wenn M an ≥ 3 Benutzer gesendet wird
  - Angreifer kennt Chiffrate $M^3 \bmod N_i$ für $1 \leq i \leq 3$
  - Chinesischer Restsatz $\rightsquigarrow M^3 \bmod N_1 N_2 N_3$
  - Wegen $0 \leq M \leq N_1, N_2, N_3$ ist $M^3 \bmod N_1 N_2 N_3 = M^3 \in \mathbb{Z}$
  - „Wurzelziehen“ über $\mathbb{Z}$ liefert M
- Könnte mit probabilistischem Enc behoben werden
- Weitere schlechte Idee: gleiches N für mehrere Benutzer

Homomorphie von RSA

- Es gilt (Rechnung modulo N):

$$\begin{aligned}\mathrm{Enc}(pk, M) \cdot \mathrm{Enc}(pk, M') &= M^e \cdot M'^e \\ &= (M \cdot M')^e = \mathrm{Enc}(pk, M \cdot M')\end{aligned}$$

- Problem z.B. bei Auktionen:
  - Auktionator A veröffentlicht pk, behält sk
  - Bieter B_1, B_2 senden verschlüsselte Gebote $C_i := \mathrm{Enc}(pk, M_i)$ an Auktionator
  - A entschlüsselt C_i, Bieter mit höchstem Gebot erhält Ware
  - Angriff: B_2 wartet B_1's Gebot ab, setzt

$$C_2 := C_1 \cdot \mathrm{Enc}(pk, 2) \bmod N$$

Lösungsvorschläge?

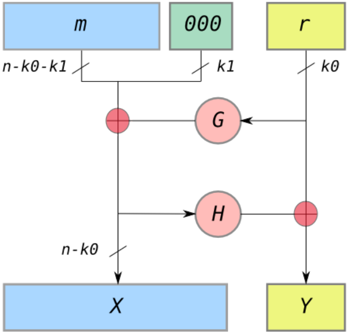
- **Diskussion:** Wie könnte man RSA „reparieren“?

RSA-Padding

- Erster Ansatz (PKCS#1 1.5, 1993): (Randomized) Padding
  - $\mathrm{Enc}(pk, M) = (\mathrm{pad}(M, R))^e$ mod N (mit Zufall R)
  - Dec erhält und überprüft $\mathrm{pad}(M, R)$, und extrahiert M
  - Beispiel: $\mathrm{pad}(M, R) = M||0^\ell||R$ (mit $M, R \ll N$)
- PKCS#1 1.5 problematisch
  - Kein Sicherheitsbeweis, 1998 sogar Problem gefunden
  - Homomorphe Eigenschaft der RSA-Funktion erlaubt subtile Änderungen an Nachricht
- PKCS#1 2.0 nutzt „RSA-OAEP“: besseres Padding

RSA-OAEP

- pad-Funktion von RSA-OAEP (G, H Hashfunktionen):



Quelle: Wikipedia

Sicherheit von RSA-OAEP

- Heuristisch so sicher wie „RSA-Funktion invertieren“
  - Sicher heißt hier: semantisch sicher selbst gegen aktive Angriffe (im idealisierten Random Oracle Modell)
- Bester bekannter Angriff: N faktorisieren
  - $P, Q \rightsquigarrow \varphi(N) = (P-1)(Q-1) \rightsquigarrow d = e^{-1} \bmod \varphi(N)$
  - Bester bekannter Faktorisierungsalgorithmus: Zahlkörpersieb
  - Nach heutigem Stand 2048 als Bitlänge von N sicher
- Offene Forschungsfrage: N faktorisieren *notwendig*, um RSA(-OAEP) zu brechen?

Relevanz von RSA(-OAEP)

- Nachteil von RSA(-OAEP): rechenaufwändig
  - Naiver Algorithmus für Exponentiation modulo ℓ-Bit-Zahl benötigt $\mathbf{O}(\ell^3)$ Bitoperationen, schlecht parallelisierbar
  - Es existieren (asymptotisch) bessere Algorithmen
  - **Aber:** Für realistische ℓ ist naiver Algorithmus am schnellsten
- Gründe, warum RSA(-OAEP) trotzdem benutzt wird:
  - Einfach zu implementieren
    - Einfache Arithmetik
    - Ver- und Entschlüsselung sehr ähnlich
  - Mit Optimierungen ($e = 3$ bei Verschlüsselung, CRS nutzen bei Entschlüsselung) teilweise konkurrenzfähig
- RSA als „Trapdoor One-Way Permutation“ interessant

Überblick

ElGamal (1985)

- Szenario: zyklische Gruppe $\mathbb{G} = \langle g \rangle$
- $pk = (\mathbb{G}, g, g^x)$, $sk = (\mathbb{G}, g, x)$ (mit x zufällig)
- $\mathrm{Enc}(pk, M) = (g^y, g^{xy} \cdot M)$ (mit y zufällig)
- $\mathrm{Dec}(sk, (Y, Z)) = Z/Y^x \qquad (= (g^{xy} \cdot M)/(g^y)^x = M)$
- Beobachtung: Verschlüsselung probabilistisch
- **Aber:** ElGamal wie RSA homomorph

$$\begin{aligned}\mathrm{Enc}(pk, M) \cdot \mathrm{Enc}(pk, M') &= (g^y, g^{xy} \cdot M) \cdot (g^{y'}, g^{xy'} \cdot M') \\ &= (g^{y+y'}, g^{x(y+y')} \cdot M \cdot M') = \mathrm{Enc}(pk, M \cdot M')\end{aligned}$$

Mehr über ElGamal

- ElGamal unter naheliegender Annahme semantisch sicher
  (allerdings *nicht* gegen aktive Angriffe)
- Nicht-homomorphe Varianten von ElGamal existieren
- Kandidaten für geeignete Gruppen $\mathbb{G}$:
  - (Echte) Untergruppen von $\mathbb{Z}_P^*$ (mit P prim)
  - Allgemeiner: Untergruppen von $\mathbb{F}_q^*$ (mit q Primpotenz)
  - Effizienter: (Untergruppen von) elliptischer Kurve $\mathbf{E}(\mathbb{F}_q)$
- Realistische Gruppengröße:
  - $|\mathbb{G}| \approx 2^{2048}$ (für $\mathbb{G} \subset \mathbb{Z}_P^*, \mathbb{F}_q^*$)
  - $|\mathbb{G}| \approx 2^{200}$ (für $\mathbb{G} \subseteq \mathbf{E}(\mathbb{F}_q)$)

Überblick

Zusammenfassung asymmetrische Verschlüsselung

- Public-Key-Verschlüsselung löst Schlüsselverteilungsproblem
- RSA wichtig, aber ohne Padding problematisch $\rightsquigarrow$ RSA-OAEP
- ElGamal in kleineren Gruppen möglich (Effizienz)
- Beide Verfahren (ungepadded) homomorph (Vorteil/Nachteil)

Aktuelle Forschung

- Mehr/andere Funktionalität, zum Beispiel:
  - Identitätsbasierte Verschlüsselung (löst Zertifizierungsproblem)
  - Vollhomomorphe Verschlüsselung (Berechnungen delegieren[1])
  - Funktionale Verschlüsselung (viele sk_f, $\mathrm{Dec}(sk_f, C) = f(M)$)
  - Broadcast-Verschlüsselung (Beispielanwendung: Pay-TV)
- Andere Probleme, alternative mathematische Strukturen
  - Public-Key-Verschlüsselung so sicher wie Faktorisierung
  - Gitterbasierte Verschlüsselung

[1] momentan noch um Größenordnungen zu ineffizient

Überblick

Überblick

Ziel

- Authentifizierte Übermittlung auf unauthentifiziertem Kanal:

$$\texttt{Alice} \quad \xleftarrow{\quad (M,\sigma) \quad} \quad \texttt{Bob}$$

- Nachricht M soll vor Veränderungen geschützt werden
- Idee: Sende „Unterschrift“ σ mit Nachricht
- Anforderungen:
  - `Bob` muss σ (aus/für Nachricht M) berechnen können
  - `Alice` muss σ (zusammen mit M) verifizieren können
  - Außenseiter soll kein gültiges σ für neues M erzeugen können

Grundidee MACs

- Annahme: `Alice` und `Bob` besitzen gemeinsames Geheimnis K

$$\texttt{Alice}_K \xleftarrow{\quad (M,\sigma) \quad} \texttt{Bob}_K$$

- Signieren: $\sigma \leftarrow \mathrm{Sig}(K, M)$
- Verifizieren: $\mathrm{Ver}(K, M, \sigma) \in \{0, 1\}$
- Korrektheit: $\mathrm{Ver}(K, M, \sigma) = 1$ für alle K, M und $\sigma \leftarrow \mathrm{Sig}(K, M)$
- Wird „MAC“ (Message Authentication Code) genannt

Überblick

Sicherheitsmodell

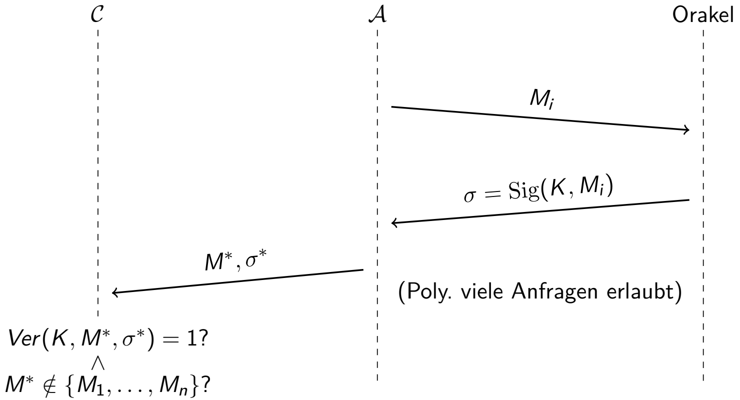
- **Diskussion:** Wünschenswerte Sicherheitseigenschaften?

Sicherheitsdefinition

- Schema EUF-CMA-sicher $\Leftrightarrow$ kein PPT-Angreifer $\mathcal{A}$ gewinnt folgendes Spiel nicht-vernachlässigbar oft:
  1. $\mathcal{A}$ erhält Zugriff auf ein $\mathrm{Sig}(K, \cdot)$-Orakel
  2. $\mathcal{A}$ gibt Ausgabe (M^*, σ^*)
  3. $\mathcal{A}$ gewinnt, wenn $\mathrm{Ver}(K, M^*, \sigma^*) = 1$ und M^* „frisch" (d.h. M^* ungleich aller Nachrichten, die $\mathcal{A}$ an das Orakel gesendet hat)

MACs – EUF-CMA: Spiel

- Herausforderer $\mathcal{C}$ wählt Schlüssel K zufällig.
- $\mathcal{C}$ stellt $\mathrm{Sig}(K, \cdot)$-Orakel für $\mathcal{A}$ bereit (aber kein Ver-Orakel!).



Sicherheitsdefinition

- Modelliert passive Angriffe ($\mathcal{A}$ erhält keinen Ver-Zugriff)
  - Für viele Verfahren (z.B. bei eindeutigem σ) äquivalent zu Definition mit Ver-Orakel für $\mathcal{A}$
  - Intuition: wenn $\mathcal{A}$ Ver-Anfrage mit $\mathrm{Ver}(K, M, \sigma) = 1$ und „frischem" (also nicht schon von Sig erzeugtem) σ generiert, ist das schon eine gefälschte Signatur