Vorlesung Nr. 6

Kryptologie II - Datum: 18.10.2018

- Primzahleigenschaften (Teil 1)
- Sicherheitsmodelle (Teil 1)

Buch der Woche

Titel: Applied Cryptography (2nd ed. 1995)

Autor(en): Bruce Schneier

Verlag: Wiley

Umfang: ca. 780 Seiten

Hinweise zum Inhalt:

- dieses Buch war bei Ersterscheinung ein Politikum, da seinerzeit Kryptographie Exportbeschränkungen unterlag, das Buch jedoch Source Code enthielt
- enthält zahlreiche Protokolle für sehr verschiedene Anwendungen
- seit vielen Jahren keine Neuauflage, aber nur wenige Inhalte wurden mittlerweile obsolet; es gibt eine Neuauflage ("20th anniversary special ed.") von 2015, aber ohne ersichtliche Aktualisierung
- als pdf Download verfügbar
- sehr gut aufgebaut, sehr gut verständlich geschrieben
- eindeutige Empfehlung!

Primzahleigenschaften (1)

Primzahlen benötigen wir für verschiedene Public Key Verfahren wie beispielsweise RSA. Doch wie können wir feststellen, ob eine (pseudo-)zufällig erzeugte große Zahl Primzahl ist oder nicht?

Für Zahlen mit mehreren tausend Bit Länge ist es nicht machbar, alle möglichen Teiler durchzuprobieren. Wir benötigen daher ein effizienteres Verfahren für einen Test auf Primzahleigenschaft.

Eines davon ist der sogenannte ***Miller Rabin Test***. Wendet man diesen Test an, so lässt sich von einer Zahl sagen, ob sie mit sehr hoher Wahrscheinlichkeit eine Primzahl ist oder nicht. Absolute Gewissheit im Sinne von 100% Wahrscheinlichkeit lässt sich allerdings nicht erzielen. Mathematisch ist das unschön, für die Praxis allerdings kein Hinderungsgrund.

Es mag überraschen, dass bis vor wenigen Jahren kein praktikabler Primzahltest für große Zahlen bekannt war, der die Primzahleigenschaft zu 100% zusichert.

Bevor wir den Miller Rabin Test einführen, benötigen wir einige Vorüberlegungen.

Primzahleigenschaften (2)

Vorbemerkung: ein ***Lemma*** ist ein Sachverhalt bzw. ein Hilfssatz, dessen Aussage für sich betrachtet keine besondere Bedeutung hat, der jedoch im Beweis eines später zu beweisenden Satzes verwendet bzw. zitiert werden kann.

Lemma 1: sei p Primzahl und $a \in \mathbb{N}, 0 < a < p$. Dann gilt

$a^2 \bmod p = 1 \quad \Leftrightarrow \quad a \bmod p = \pm 1$

Beweis:

"$\Leftarrow$": sei $a \bmod p = \pm 1$, dann folgt $1 = (\pm 1)^2 = (a \bmod p)^2 = a^2 \bmod p$.

"$\Rightarrow$": sei $a^2 \bmod p = 1$, dann folgt $1 = a^2 \bmod p = (a \bmod p)^2 \bmod p$.

Aus $1 = |1| = |a \bmod p| \cdot |a \bmod p|$ folgt $a \bmod p = \pm 1$.

Primzahleigenschaften

Hinweis: wir werden in den folgenden Vorlesungen nochmals auf Primzahlen und Primzahltests zurück kommen...

Kryptosicherheit

One-Time Pad: Funktionsweise

Das One-Time Pad ist das einzige mathematisch beweisbar sichere Verschlüsselungsverfahren! Die Sicherheit fast ALLER anderen Verfahren beruht auf unbewiesenen Annahmen.

- Wie funktioniert das One-Time Pad:
  - Die Nachricht wird mittels XOR mit einem Schlüssel verknüpft, der ebenso lang wie die Nachricht ist
  - Der Schlüssel muss aus echten Zufallswerten gebildet sein
  - Jeder Schlüssel darf nur einmal verwendet werden

  Wird eine dieser drei Anforderungen verletzt, ist das One-Time Pad nicht mehr sicher

Beispiel: seien der Klartext P und Schlüssel K jeweils eine Bitfolge der Länge n:

$P, K \in \{0,1\}^n$. Dann errechnet sich der Chiffretext C mittels $C = P \oplus K$.

Was geschieht, wenn der Schlüssel K mehrfach verwendet wird:

Offensichtlich gilt für $C_1 = P_1 \oplus K$ und $C_2 = P_2 \oplus K$:

$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2 \oplus K \oplus K = P_1 \oplus P_2$

Sind P_1 und P_2 Sprachnachrichten, so lassen sich beide größtenteils wieder herstellen. Dies geht sogar (mit Einschränkungen) für mehr als zwei Überlagerungen $P_1 \oplus P_2 \oplus P_3 \oplus \ldots$ Entscheidend ist hierbei die Entropie der P_i.

Sicherheit des One-Time Pad

Wir verzichten an dieser Stelle auf einen streng formalen Sicherheitsbeweis. Auch ohne Formeln ist die Sicherheit des One-Time Pad schnell einzusehen.

Annahme: ein Angreifer soll nicht in der Lage sein, irgendeine (was auch immer) Information über den Inhalt der Nachricht zu erhalten, außer deren Länge. Hierbei gehen wir davon aus, dass der Angreifer über unbeschränkte Ressourcen (Zeit und Rechenpower) zur Berechnung verfügt und insbesondere alle möglichen Schlüssel durchprobieren kann.

Trotz dieser starken Annahme ("***Perfect Secrecy***")können wir sehen, dass der Angreifer keinen Informationsgewinn erzielt:

Angenommen, unser Schlüssel K besteht aus n echt zufälligen Bits $k_1, \ldots, k_n$. Ferner sei $P = (p_1, \ldots, p_n)$ und $C = (c_1, \ldots, c_n)$.

Da K zufällig, ist die Wahrscheinlichkeit, dass ein $k_i = 0$ ist jeweils gleich $\frac{1}{2}$.

Damit beträgt auch die Wahrscheinlichkeit, dass $k_i \oplus p_i = 0$ jeweils $\frac{1}{2}$.

Folglich sehen alle $c_i = k_i \oplus p_i$ zufällig aus und C gibt keine Information preis.

One-Time Pad und schlechte Schlüssel

Angenommen, unser One-Time Pad arbeitet auf einer Bitfolge der Länge n = 16 (wir wählen hier anschaulich kleine Zahlen, unser Argument gilt aber unverändert auch für größere Werte). Nehmen wir weiter an, unser Schlüsselraum umfasst nur eine echte Teilmenge aller möglichen 2^{16} Bitfolgen. Dies seien beispielsweise die drei Bitfolgen

K_1 = (0010 0001 0111 0011)

K_2 = (1111 1110 1100 1000)

K_3 = (1000 0000 0101 0100)

Haben wir nun einen Chiffretext C = (1010 0010 0001 0011), so gibt es nur drei Möglichkeiten P_1, P_2, P_3 für den Klartext: $P_i = C \oplus K_i$ für i = 1, 2, 3.

Wir wissen also zwar noch nicht, welches P_i der richtige Klartext ist, doch wir können alle Klartexte P ausschließen, die keinem der P_i entsprechen.

Selbst wenn die Menge der K_i bzw. P_i weit größer ist: solange diese nicht alle 2^n Bitfolgen umfasst, erhält unser Angreifer einen Informationsgewinn und unsere Chiffre erfüllt nicht mehr die ursprüngliche Anforderung (Perfect secrecy).

Varianten des One-Time Pad

Wir betrachten nun drei Varianten des One-Time Pad bezüglich folgender Frage: Erfüllt die jeweilige Lösung unsere Anforderungen an perfekte Sicherheit?

- **Variante 1**: angenommen, wir möchten beliebige Bitfolgen P der Länge 128 verschlüsseln. Als Elemente unseres Schlüssels K wählen wir die Menge M aller druckbaren ASCII Zeichen (es gibt 95 solcher Zeichen) in ihrer 8-Bit Darstellung: z.B. Space = 00100000, …, A = 01000001, B = 01000010 usw. Ein 128 Bit langer Schlüssel K besteht also aus 16 druckbaren ASCII Zeichen. Die Verschlüsselung erfolgt durch bitweise XOR-Verknüpfung der beiden Bitfolgen P und K.
- **Variante 2**: wie sieht es aus, wenn wir obige Variante 1 beibehalten, als Klartexte jedoch ebenfalls nur druckbare ASCII-Zeichen zulassen?
- **Variante 3**: was ändert sich gegenüber Variante 2, wenn Klartext und Schlüssel beide aus druckbaren ASCII-Zeichen bestehen, die Verknüpfung diesmal jedoch nicht durch bitweises XOR realisiert wird, sondern wie folgt: wir ordnen jedem der 95 druckbaren ASCII-Zeichen einen Wert zwischen 0 und 94 zu und addieren dann Klartext und Schlüssel Zeichen für Zeichen modulo 95.

Der Sicherheitsbegriff in der Kryptographie

Was ist Sichere Verschlüsselung?

Frage in die Runde: was ist sichere Verschlüsselung?

Sichere Verschlüsselung

Wir sprechen immer wieder von sicherer Verschlüsselung, ohne dass wir genauer spezifiziert haben, was damit gemeint ist. Umgangssprachlich formuliert, soll eine Verschlüsselung nicht von Unbefugten geknackt werden können. Was dies genau bedeutet, hängt allerdings vom Kontext und von den Spielregeln ab, denen die beteiligten Parteien unterworfen sind.

Zu diesem Zweck formulieren wir zum einen Sicherheitsziele (Security goals) und zum anderen Angriffsmodelle (Attack models).

- **Security goals** legen fest, welchen Anforderungen hinsichtlich unterschiedlicher Sicherheitsmerkmale ein Algorithmus oder Protokoll genügen soll.
- **Attack models** legen dar, unter welchen Rahmenbedingungen ein kryptographisches Verfahren auf bestimmte Security goals hin überprüft wird.
- **Security notions** (notion = Vorstellung, Idee, Auffassung, ..) kombinieren individuelle Security goals und Attack models und erlauben so eine präzisere und differenziertere Betrachtung dessen, wieviel und welche Sicherheit in einem gegebenen Szenario wirklich realisierbar ist.

Kerckhoff's Prinzip

Wir kennen die heute größtenteils akzeptierte Forderung von Kerckhoff, die seinerzeit im Kontext militärischer Verschlüsselungsverfahren und –maschinen formuliert wurde:

Die Sicherheit eines Kryptosystems darf nur von der Geheimhaltung des Schlüssels abhängig sein. Die Details der Funktionsweise hingegen dürfen dem Gegner bekannt sein ohne dass dies die Sicherheit beeinträchtigt.

Die Forderung bezog sich damals offenkundig auf den algorithmischen Teil, also die Frage "wie wird verschlüsselt".

Heute stellt sich ein weiteres Problem in der Form speziell geschützter Hardware, die eine Analyse des Verschlüsselungsprozesses zum Zweck der Schlüsselextrahierung verhindern soll (z.B. Timing Attacken).

Frage in die Runde: sollen wir Kerckhoff's Forderung ergänzen und konkretisieren, so dass auch die Beobachtung und Analyse von Verschlüsselungsprozessen berücksichtigt wird? Machen Sie einen Vorschlag!

Attack Models

Zweck eines Attack Models ist es, die Spielregeln genauer zu spezifizieren, unter denen die beteiligten Akteure, insbesondere die Angreifer, agieren.

- Welche Möglichkeiten gestehen wir einem Angreifer zu:
  - darf er Verschlüsselungen mit selbstgewählten Werten durchführen?
  - wieviele Plaintexts, Ciphertexts, Keys bzw. Kombinationen aus solchen sind dem Angreifer bekannt?
  - unterliegt er vorgegebenen Grenzen hinsichtlich Speicher und Rechenkapazität?
  - kennt er den Algorithmus oder ist ihm das Verfahren gänzlich unbekannt?
  - darf er Messungen vornehmen, beispielsweise an der Kryptohardware?
  - hat er Einfluss auf manche der verwendeten Parameter?
- Welchen Nutzen haben Attack models:
  - Entwickler neuer Kryptosysteme wissen genauer, gegen welche Art Angriffe ihr System gewappnet sein soll
  - Kryptoanalytiker wissen im Vorfeld, ob sich ein von ihnen betrachteter Angriff im Rahmen dessen befindet, was zuvor als auszuschließendes Szenario identifiziert wurde
  - Anwender eines Kryptosystems können besser abschätzen, welche Art von Angriffen das Verfahren abzuwenden imstande ist bzw. sein soll und ob das System ihren Anforderungen gerecht wird oder nicht

Plaintext-Ciphertext Angriffstypen (1)

Wir unterscheiden die nachfolgend genannten Angriffstypen auf Verschlüsselungen, in Abhängigkeit von vorhandenen Wahl- und Zugriffsmöglichkeiten auf Klartexte und Chiffretexte.

- ***COA Ciphertext-only attacks***: der Angreifer erhält lediglich (irgendwelche) Ciphertexts ohne Kenntnis der zugehörigen Plaintexts.
- ***KPA Known-plaintext attacks***: der Angreifer erhält (irgendwelche) Paare aus Plaintext und Ciphertext.
- ***CPA Chosen-plaintext attacks***: der Angreifer erhält für von ihm selbst gewählte Plaintexts die zugehörigen Paare aus Plaintext und Ciphertext.
- ***CCA Chosen-ciphertext attacks***: der Angreifer erhält für von ihm selbst gewählte Ciphertexts und/oder Plaintexts die zugehörigen Paare aus Plaintext und Ciphertext. (Anmerkung: Die Bezeichnung CCA ist etwas irreführend, da hier eben auch CPA enthalten ist.)

Plaintext-Ciphertext Angriffstypen (2)

Betrachten wir kurz zwei Spezialfälle.

- Public Key Kryptographie und KPA/CPA

  Im Zusammenhang mit Public Key Verfahren macht es nur in Ausnahmen Sinn, über KPA versus CPA zu sprechen. Die Verfahren haben ja gerade die Eigenschaft, dass jeder mit Kenntnis des öffentlichen Schlüssels beliebige Texte verschlüsseln und das Ergebnis betrachten kann.

- Praxisnutzen von CCA

  Es mag nicht unmittelbar einleuchten, welchen Vorteil es bringen soll, zu gegebenen Chiffretexten den entschlüsselten Klartext zu erhalten. Tatsächlich finden sich jedoch unterschiedlichste Beispiele, die auf diesem Angriffsmuster aufsetzen. Siehe beispielsweise “Random Padding Oracle” u.v.m.

  In der Praxis haben wir z.B. bei der Verschlüsselung von TV-Signalen den Fall, dass Decoder die Möglichkeit zu Chosen ciphertext Attacken bieten. Ebenso gilt dies für Verschlüsselungshardware mit intern verborgenem Schlüssel, wie dies etwa bei Smartcards oft der Fall ist: diese erlauben sowohl die Verschlüsselung als auch die Entschlüsselung selbst gewählter Daten, ohne jedoch den Schlüssel preiszugeben. Möchte man solche Decoder oder Smartcards klonen, so braucht man den Schlüssel.

Sicher oder Unsicher?

Sicherheitsziele in der Kryptographie unterscheiden sich von allgemeineren Anforderungen der IT Security. Innerhalb der Kryptographie haben wir wiederum ganz verschiedene Forderungen, je nach Anwendungsfall.

- Ist ein Hashverfahren schon dann unsicher, wenn mit massivem Rechenaufwand irgendeine Kollission gefunden werden kann?
- Ist ein Verschlüsselungsverfahren geknackt, wenn man nicht mehr ALLE möglichen Schlüssel durchprobieren muss, aber immer noch mehr als in der Praxis machbar?
- Bezeichnen wir ein Public Key Verfahren als unsicher, wenn eine kleine Teilmenge möglicher Parameter (z.B. einzelne Schlüssel oder RSA-Exponenten) zur Schwächung des Verfahrens führt?
- Ist ein Verfahren nicht auch dann noch praktikabel, wenn es zwar anfällig ist gegenüber CCA, aber wenn es KPA widersteht?

Frage in die Runde: haben Sie weitere Beispiele für den Einsatz von Kryptosystemen, analog zu oben genannten, in denen eine Abwägung zu unterschiedlichen Sicherheitsbewertungen führen kann?