Kryptographie

Boyan Hristov 14. Januar 2018 Claudia Dieckmann

Zusammenfassung

Wir werden einige Konzepten einführen, die man als die Grundbausteine der Kryptographie betrachtet. Wir werden dabei sehen, unter welchen Annahmen sichere Kryptographie existieren kann. Weiter wird der Zusammenhang mit den Klassen P und NP erleutert, als auch werden einigen interessanten Schlussfolgerungen gemacht. Am Ende werden wir auch die Funktion hinter den berühmten RSA Algorithmus sehen.

1 Motivation

1.1 Secret Codes & One-way Pads

Die naive Methode kriptierte Konversation zu haben ist, wenn Sender und Empfänger der gleiche Schlüssel benutzen, um Nachrichten zu kriptieren und dekriptieren. Dabei existiert das Problem, dass Sender und Empfänger sich auf einen Schlüssel einigen müssen. Das ist aber schwierig, da bevor sie das gemacht haben, können sie auch keine sichere Nachrichten zwischen einander senden. Weiter ist das keine gute Lösung, da für jede Konversation ein Schlüssel existieren muss, d.h. wenn jemand mit 100 Leuten kriptierte Nachrichten austauschen will, muss der auch 100 Schlüsseln speichern.

Laut Theorie ist ein Schlüssel nur dann sicher, wenn es mindestens so lang ist wie alle Nachrichten, die damit verschlüsselt werden sollen. Deswegen sind die sogenannte “One-way Pads“ entstanden. Dabei wird je Nachricht mit einem Teil des Schlüssels verschlüsselt und dieses Teil wird nachher gelöscht. So garantiert man, dass ein Schlüssel durch “Brute-Force“ schwierig zu finden ist. Wie wir aber später sehen werden, reicht es auch wenn der Schlüssel relativ lang ist, solange es keine andere Methode außer Brute-Force gibt, den Schlüssel leichter zu finden.

So kommen wir zu den Konzepten der sicheren Hashfunktionen und der asymmetrischen Verschlüsselung, wobei eine Nachricht mit einem Schlüssel verschlüsselt wird und mit einem anderen entschlüsselt.

1.2 Ziele der Public-Key Kryptosysteme

Hier werden wir einige Notationen einführen.

- $E : \Sigma^* \longrightarrow \Sigma^*$ - Funktion, die eine Nachricht kriptiert
- $D : \Sigma^* \longrightarrow \Sigma^*$ - Funktion, die eine Nachricht dekriptiert
- $M \in \Sigma^*$ - eine Nachricht

Jedes "Public-Key" Kryptosystem muss die folgenden Eigenschaften haben. Wir werden später sehen, was für Annahmen gemacht werden müssen, um diese Eigenschaften zu sichern.

1. Man muss die originelle von kriptierten Nachrichten generieren können.
   Formal: $D(E(M)) = M$
2. E und D sind polynomiellzeit berechenbar
3. Wenn ein Nutzer E veröffentlicht, veröffentlich er nicht D und auch kein Weg, D leichter (in polynomieller Zeit) zu berechnen
4. Mann muss eine Nachricht erstmal dekriptierten und dann kriptieren können, wobei wieder die originelle Nachricht entsteht.
   Formal: $E(D(M)) = M$

2 Erinnerung

Die meisten Leser können dieses Teil überspringen, es ist aber wichtig zu verstehen, was links- und rechtsinvertierbarkeit bedeuten und welche Folgen diese für die Kryptographie haben.

2.1 Injektivität

Definition 2.1. Eine Funktion $f : X \longrightarrow Y$ ist injektiv
$\Leftrightarrow (f(a) = f(b) \Leftrightarrow a = b)$

2.2 Surjektivität

Definition 2.2. Eine Funktion $f : X \longrightarrow Y$ ist surjektiv
$\Leftrightarrow \forall y \in Y \exists x \in X : f(x) = y$

2.3 Linksinvertierbarkeit

Hier ist I die Identitätsfunktion und $\odot$ die Verkettung von Funktionen.

Definition 2.3. Eine Funktion $f : X \longrightarrow Y$ ist linksinvertierbar
$\Leftrightarrow \exists g : Y \longrightarrow X : g \odot f = I_x$
$\Leftrightarrow f$ ist injektiv

2.4 Rechtsinvertierbarkeit

Hier ist I die Identitätsfunktion und $\odot$ die Verkettung von Funktionen.

Definition 2.4. Eine Funktion $f : X \longrightarrow Y$ ist rechtsinvertierbar
$\Leftrightarrow \exists g : Y \longrightarrow X : f \odot g = I_x$
$\Leftrightarrow f$ ist surjektiv

3 One way functions (Einwegfunktionen)

In diesem Abschnitt werden die Einwegfunktionen in starke und schwache Unterteilt. Es wird auch gezeigt, dass die existenz von schwachen Einwegfunktionen diese von starken impliziert. Weiter werden wir sehen, dass eine längenerhaltende Funktion aus eine nicht längenerhaltende konstruiert werden kann. Die starke Einwegfunktionen sind ein grundlegendes Baustein für die Kryptographie, Ziel ist aber hier auch alle Annahmen zu verstehen die dazu führen.
Da die meisten Definitionen auf probabilistische Algorithmen sich basieren, werden wir erstmal definieren, was vernachlässige Erfolgswahrscheinlichkeit bedeutet, um bessere die folgende Konzepte erklären zu können.

3.1 Vernachlässige Erfolgswahrscheinlichkeit

Definition 3.1. Vernachlässige Erfolgswahrscheinlichkeit

- Die Erfolgsrate eines Algorithmus ist genau dann vernachlässigbar, wenn es asymptotisch von oben polynomialbegrenzbar ist
- Funktion abhängig von der Eingabegroße

- D.h, auch wenn man polynomial lange den Algorithmus wiederholt, bekommt man wieder eine vernachlässigbare Erfolgswahrscheinlichkeit
- Damit ist vernachlässigbare Erfolgswahrscheinlichkeit $\equiv$ nicht polynomielzeit berechenbar

3.2 Starke und Schwache Einwegfunktionen

Definition 3.2. Eine Funktion $f : 0,1^* \longrightarrow 0,1^*$ ist eine starke one-way Funtkion $\Leftrightarrow$

1. f ist polynomiellzeit berechenbar
2. Für jeden polynomiellzeit Algorithmus A, für jedes Polynom p und für große n gilt:
   $Pr(A(f(w), 1^n) \in f^{-1}(f(w))) \leq \frac{1}{p(n)}$

Damit hat A per Definition eine vernachlässige Erfolgswahrscheinlichkeit. 1^n wird als Parameter übergeben um zu garantieren, dass die inverse Funktion echt polynomielzeit unberechenbar ist. Wenn man das weglässt, könnte sein, dass die Eingabegröße von f exponentiell größer ist als die Ausgabegröße und damit f nur deswegen nicht invertierbar ist, weil kein Algorithmus in polynomielle Zeit das Ergebniss auf dem Band schreiben kann.

Definition 3.3. Eine Funktion $f : 0,1^* \longrightarrow 0,1^*$ ist eine schwache one-way Funtkion $\Leftrightarrow$

1. f ist polynomiellzeit berechenbar
2. Es existiert ein Polynom p, so dass für alle polynomiellzeit Algorithmen A und große n:
   $Pr(A(f(w), 1^n) \notin f^{-1}f(w)) > \frac{1}{p(n)}$

Merke, dass f invertierbar sein kann, nur nicht in polynomieller Zeit. Weiter steht in der Definition nicht fest, ob f injektiv oder surjektiv ist, da hier f nicht unbedingt in mathematischem Sinne invertierbar ist. D.h. auch $f(x) = 2$ ist laut dieser Definition keine Einwegfunktion, da $g(x) = 1$ schon als zu f inverse Funktion gilt. Surjektivität werden wir für "Public-Key" Kryptosysteme auf jeden Fall brauchen. Ohne injektivität kann man schon eine Hashingfunktion erstellen (es existieren viele solche), aber keine mathematisch sichere, da mehrere unterschiedliche $w \in \Sigma^*$ die selbe Hashwerte bekommen werden.

Hoffnung für sichere Hashfunktionen gibt es erst mit den Einwegpermutationen, die wir bald sehen werden. Bevor müssen wir aber auch die längenerhaltende Funktionen einführen, die in der Praxis ganz hilfreich sind.

3.3 Längenerhaltende Funktionen

Definition 3.4. Eine Funktion $f : \Sigma^* \longrightarrow \Sigma^*$ ist längenerhaltend $\Leftrightarrow \forall w \in \Sigma^* : |f(w)| = |w|$

Satz 3.5. *Wenn eine Einwegfunktion existiert, dann existiert auch eine längenerhaltende Einwegfunktion*

Beweis. Sei $f : \Sigma^* \longrightarrow \Sigma^*$ eine starke Einwegfunktion. Da f polynomialzeit berechenbar ist $\Rightarrow \forall x \in \Sigma^* : |f(x)| \leq p(|x|)$, d.h. die Länge von jeder Abbildung von f ist polynomiell in der Länge begrenzt.
Wir konstruieren $g : \Sigma^* \longrightarrow \Sigma^*$ und $h : \Sigma^* \longrightarrow \Sigma^*$.
$g(x) =^{def} f(x)10^{p(|x|)-|f(x)|}$.
Damit haben wir festgelegt, dass jede Abbildung von g die maximale Länge von allen Abbildungen von f haben wird.
$h(x'x'') =^{def} g(x')$, wobei $|x'x''| = p(|x'|) + 1$.
Damit konstruieren wir h so, dass je Eingabe die gleiche Länge hat. Die Eingabemenge wird wieder künstlich gefüllt, die letzte Zeichen werden aber ignoriert.

Damit ist trivialerweise h eine längenerhaltende Funktion. Jetzt müssen wir nur zeigen, dass diese auch eine starke Einwegfunktion ist. Der Beweis per Wiederspruch folgt.

Sei B ein Algorithmus, dass g in polynomielle Zeit invertiert. Wir konstruieren den Algorithmus A, der f in polynomielle Zeit invertiert.
Bei Eingabe $(y, 1^n)$ hällt dann A mit Ausgabe $B(y10^{p(n)-|y|}, 1^{p(n)+1})$.
Damit haben wir gezeigt, dass $B \geq_p A$. Wiederspruch, da A polynomiellzeit unberechenbar ist.
Analog für h. □

4 One-way Permutationen

Hier wurde die Definition aus "Introduction to the Theory of Computation" von Michael Sipser benutzt, jedoch etwas angepasst, damit die Konvention weiter ähnlich bleibt.

Definition 4.1. Eine Funktion $f : \Sigma^* \longrightarrow \Sigma^*$ ist eine Permutation $\Leftrightarrow$

1. f ist polynomiellzeit berechenbar

2. Für alle probabilistische Algorithmen A, $\forall k \in N^+$ und für große n, wir wählen ein Wort w mit $|w| = n$ und führen A mit Eingabe w aus $Pr_{A,w}(A(f(w)) = w) \leq n^{-k}$

Hier wurde auch die Definition einer starken Einwegpermutation benutzt. Wie wurde diese einer schwachen aussehen?

4.1 Bedeutung

Die starke längenerhaltende Einwegpermutationen sind das Grundbaustein der Kryptographie. Wenn solche existieren, kann man sichere Hashingalgorithmen erstellen. Mit solchen werden heutzutage z.B. die Passwörter hashiert, bevor diese in einem Datenbanksystem gespeichert werden. So kann jedes mal das System das von Nutzer eingegebenes Passwort hashieren und mit dem gespeicherten vergleichen. Wenn jemand die Passwörter aus dem Datenbank sehen kann, kann dieser nie das originelle Klartextpasswörter von diesen erstellen.

5 Trapdoor Functions (Falltürfunktionen)

Definition 5.1. Eine Funktion $f : \Sigma^* \longrightarrow \Sigma^*$ ist eine indexierte Funktion, für die ein probabilistischer polynomiellzeit Algorithmus A existiert, und eine zu f inverse Funktion $h : \Sigma^* \longrightarrow \Sigma^*$. Dabei müssen f, g und h die folgenden Eigenschaften haben.

1. f und h sind polynomiellzeit berechenbar

2. Für alle probabilistische Algorithmen A', $\forall k \in N^+$ und für große n, wir wählen eine beliebige Rückgabe (i, t) von A bei Eingabe $(w \in \Sigma^n, 1^n)$ $Pr(A'(i, f_i(w)) = y | f_i(y) = f_i(w)) \leq n^{-k}$

3. $\forall n, \forall w \in \Sigma^n$, für je Rückgabepaar (i, t) von A, dass mit nichtvernachlässige Wahrscheinlichkeit vorkommt, gilt: $h(t, f_i(w)) = y$, wobei $f_i(y) = f_i(w)$

Hier ist t das Gehemniss, womit f leicht invertierbar ist. Die zweite Eigenschaft sagt, dass f ohne t schwer invertierbar ist und die dritte, dass f mit Hilfe von t leicht invertierbar ist und h die entsprechende inverse Funktion ist.

5.1 Bedeutung

Das ist schon alles, was wir brauchen, um ein "Public-Key" Kryptosystem zu konstruieren. Die Funktion f_i können wir veröffentlichen, das wird unser öffentliches Schlüssel sein. Die Funktion h bzw. das Gehemnis t werden wir für uns behalten.

6 Beispiel

6.1 Konstruktion der Trapdoor Funktion hinter RSA

1. Wähle 2 (echt) große Primzahlen p und q
2. Berechne $N = pq$
3. Berechne $\phi(N) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$
4. Wähle $1 < e < N$ zufällig, so dass es semiprim zu $\phi(N)$ ist. Je Primzahl $> \phi(N)$, die die bisher beschriebene Eigenschaften hat ist möglich.
5. Berechne d, so dass $ed \equiv 1 mod \phi(N)$. Das ist immer möglich, da alle Zahlen zwischen 1 und $\phi(N)$ formen eine Gruppe, die abgeschlossen bezüglich Multiplikation Modulo $\phi(N)$ ist. Dazu bietet sich das euclidische Algorithmus für die Suche nach $GGT(\phi(N), e)$.

6.2 Anwendung von RSA

Aus dem oben beschriebenen Algorithmus werden wir e und N veröffentlichen und d für uns behalten. Damit gilt:

- $E(M) = M^e mod N$
- $D(M) = M^d mod N$
- Damit $D(E(M)) = (M^e mod N)^d mod N = M^{ed} mod N = M$
- Und $E(D(M)) = M^{ed} mod N = M$ analog

So können wir Nachrichten beides kryptieren (mit dem öffentlichen Schlüssel von jemandem, nur der kann das öffnen), oder signieren, wobei wir eine Nachricht mit unserem privaten Entschlüsselungsalgorithmus kriptieren. Wenn die Nachricht sich mit unserem öffentlichen Schlüssel sich entschlüsseln lässt, dann wurde diese sicherlich mit unserem privaten Schlüssel verschlüssel und wurde damit sicherlich von uns geschrieben.

6.3 Mathemaschen Hintergrund

7 Konsequenzen von der Existenz der Einwegfunktionen

Obwohl es eine starke Vermutung gibt, dass $P \neq NP$, macht es immer Sinn nachzudenken, was für Bedeutung $P = NP$ für die Kryptographie hat, bzw. wie sicher wir sind, wenn $P \neq NP$ bewiesen wird.

Satz 7.1. $P = NP \Rightarrow$ *es existieren keine Einwegfunktionen*

Beweis. Sei $f : \Sigma^* \longrightarrow \Sigma^*$, $\Sigma = 0, 1$, eine Einwegfunktion.
Sei L eine Sprache von Paaren, so dass $(x^*, y) \in L \Leftrightarrow \exists x : f(x) = y$ und x^* ist ein Präfix von x.
$L \in NP$, da bei ein Eingabe (x^*, y) und Zeuge (x, y) können wir in polynomieller Zeit überprüfen, ob $f(x) = y$ (da f polynomiellzeit berechenbar nach der Definition einer Einwegfunktion) und ob x^* ein Präfix von x ist.

Da aber nach Annahme $P = NP \Rightarrow L \in P$. Sei A das Algorithmus, dass L in polynomieller Zeit entscheidet. Deswegen können wir L benutzen um f in polynomieller zu invertieren, also um aus je Ausgabe y das entsprechende Eingabewert x zu generieren.
Wir fangen mit (ϵ, y). Dann raten wir das nächste Zeichen von x und benutzen A um zu überprüfen, ob wir richtig geraten haben. Da A in polynomieller Zeit läuft, haben wir am Ende eine Laufzeit von O($|y|p$), wobei p ein Polynom ist. Damit haben wir in polynomieller Zeit f invertiert, deswegen ist per Definition f keine Einwegfunktion. Wiederspruch! □

8 Literatur

- Introduction To The Theory Of Computation, Michael Sipser
- Foundations of Cryptography - Basic Tools, Oded Goldreich
- https://crypto.stackexchange.com/questions/39878/how-to-show-that-a-one-way-function-proves-that-p-
  - Beweis für $P = NP \Rightarrow$ Einwegfunktionen existieren nicht
- Construct a length preserving one way function, Stackexchange

  - https://cs.stackexchange.com/questions/10639/length-preserving-one-way-functions
  - Oben wurde den Beweis von Oded Goldreich benutzt, das war aber für das Verständniss hilfreich

- A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, R.L. Rivest, A. Shamir, und L. Adleman
  - Wie die Namen der Authoren schon zeigt, ist das das Paper zu RSA

- https://en.wikipedia.org/wiki/One-way_function