Secret Sharing Shemes
Wer verdient mehr?
Skat per Telefon
Primzahlerzeugung und -test

Stephan Seigewasser

29. November 2001

Secret Sharing Shemes

Ziel

Geheimnis in mehrere Teilgeheimnisse aufteilen, so daß das Geheimnis nur aus bestimmten, vorher festgelegten Gruppen von Teilgeheimnissen wieder rekonstruiert werden kann

Beispiele

Schatzkarte zerlegen in 4 Teile, alle Teilgeheimnisse nötig
Vertragsabschluß, mind. 2 Vorstände nötig
→ flexibel - 2 beliebige Vorstände reichen aus

Threshold-Verfahren

Teilen das Geheimnis S in n verschiedene Teilgeheimnisse s_1 bis s_n, so daß $\forall\ t \in$ N gilt:

- aus je t oder mehr Teilgeheimnissen kann das Geheimnis rekonstruiert werden
- aus weniger als t Teilgeheimnissen kann man das Geheimnis nicht berechnen, sondern höchstens mit einer verschwindend geringen Wahrscheinlichkeit raten

→ t heißt dann Schwelle

Beispiel

Fall t = 3
→ betrachten reelle Ebene: Geheimnis S wird als Zahl s wie folgt dargestellt:

- suchen ein Polynom $f(x) = ax^2 + bx + c$, wobei a und b zufällig gewählte Zahlen sind
- Polynom schneidet die y-Achse in $(0, s)$, also im Punkt, der das Geheimnis S darstellt
- als Teilgeheimnisse werden n verschiedene Punkte auf dem Graph von $f(x)$ gewählt

→ Teilgeheimnisse werden dann auf verschiedene Personen verteilt

Rekonstruktion

Zur Rekonstruktion von S ist nun folgendes nötig:

- mindestens 3 Teilgeheimnisse s_i, s_j und s_k
- geometrische Struktur, d.h. Fakt, daß reelle Ebene benutzt wurde und S in $(0, s)$ liegt
- $f(x)$ kann nun mittels Lagrange-Interpolation ermittelt werden

$\rightarrow$ allg. kann aus t Geheimnissen das zugehörige Polynom $(t-1)$-ten Grades ermittelt werden wie folgt:

$$\boxed{f(x) = \sum_{i=1}^{t} \frac{(x-a_1)\ldots(x-a_{i-1})(x-a_{i+1})\ldots(a_i-a_t)}{(a_i-a_1)\ldots(a_i-a_{i-1})(a_i-a_{i+1})\ldots(a_i-a_t)}} \quad (1)$$

In der Praxis werden statt reeller Zahlen endliche Körper genommen. In diesen gelten die gleichen Rechenregeln sowie die Lagrange-Interpolation, es treten aber keine Rundungsfehler auf.
Wenn der Körper q Elemente hat und der Angreifer weniger als die t benötigten Teilgeheimnisse kennt, dann beträgt die Wahrscheinlichkeit, daß das richtige Geheimnis erraten wird, $\frac{1}{q+1}$.

Komplexe Secret Sharing Shemes

Neben den Threshold-Verfahren gibt es noch weitere, komplexere Secret Sharing Shemes. Sie werden oftmals geometrisch umgesetzt.

Beispiel

Firmentresor, der nur durch folgende Konstellationen geöffnet werden kann:

- 2 Direktoren
- 3 Vizedirektoren
- 1 Direktor + 2 Vizedirektoren

Realisierung

$\rightarrow$ nehmen 3-dim. Raum, wobei das Geheimnis durch den Punkt $(0, 0, s)$ auf der z-Achse repräsentiert wird

Erzeugung der Teilgeheimnisse wie folgt:

- wähle Ebene E durch den Punkt s, die nicht die z-Achse enthält
- wähle Gerade g in E durch s, auf der die Punkte D_1, D_2, ... für die einzelnen Direktoren angeordnet sind
- wähle Kreis c in E durch s, worauf die Punkte V_1, V_2, ... für die Vizedirektoren angeordnet sind, wobei zu beachten ist, daß keine Gerade $\overline{V_iV_j}$ durch einen Punkt D_k verläuft

$\Rightarrow$ dann entschlüsselt jede Konstellation, welche der obigen Form entspricht, daß Geheimnis S

Einsatzgebiete

1. Bereitstellung eines Geheimnisses:
   $\rightarrow$ Geheimnis vorher nicht im System vorhanden und wird dann z.B. als Kryptoschlüssel weiterverwendet
2. Vergleich mit systeminternen Geheimnis:
   $\rightarrow$ Geheimnis wird mittels Einwegfunktion verschlüsselt und dann mit dem systeminternen Geheimnis verglichen
   $\rightarrow$ bei Übereinstimmung wird dann irgenteine Aktivität ausgelöst

Wer verdient mehr?

Problem

Drei Personen A, B und C wollen untereinander feststellen, ob sie mit ihrem Gehalt zufrieden sein können. Dabei sollen die jeweils beiden Anderen aber nichts über das eigene Gehalt erfahren.

Lösung

Als Verfahren wählen sie sich die Berechnung eines Durchschnittsgehalts. Person A wird als Protokollführer ausgewählt. A addiert zuerst zu seinem Gehalt eine beliebige Zufallszahl r und teilt dann B das Ergebnis mit. B addiert zu dieser Zahl sein Gehalt und teilt C das Ergebnis mit. C verfährt analog und teilt dann A den erhaltenen Wert mit. A zieht davon die Zufallszahl r ab und kann leicht den Durchschnitt der 3 Gehälter ausrechnen. Anhand dieses Wertes können A, B und C dann feststellen, ob sie mit ihrem Gehalt zufrieden sein können oder nicht.

In diesem Protokoll können A, B und C nicht auf die Gehälter der jeweils Anderen schließen. Das Protokoll ist aber nur solange sicher, wie sich nicht 2 Personen absprechen. In diesem Fall können sie leicht auf das Gehalt des Dritten schließen.

Das Protokoll läßt sich auch auf beliebig viele Personen erweitern. Je mehr Personen daran teilnehmen, desto sicherer wird das Verfahren, da sich immer mehr Personen miteinander verbünden müssen, um auf das Gehalt einer einzelnen Person zu schließen.

Problem

Das oben vorgestellte Verfahren eignet sich nicht dazu, nur von 2 Personen A und B die Gehälter zu vergleichen.

Lösung

Zuerst wird vorausgesetzt, daß die Gehälter auf 100 EURO genau gerundet werden und das keiner der Beiden mehr als 10.000 EURO verdient. Mittels leichter Modifikationen des Verfahrens können diese Voraussetzungen aber einfach verändert werden.
$\Rightarrow$ Gehälter sind darstellbar als Zahlen $0 \leq a, b \leq 100$

START: A wählt Zufallszahl x und verschlüsselt sie mit $B's$ öffentlichen Schlüssel $\Rightarrow$ c:=$\mathrm{E}_B(x)$. A sendet dann die Zahl $d := c - a$ an B.

B versucht jetzt, x zu berechnen, kennt aber a nicht und kann somit nicht auf c schließen. Deshalb wendet B die Entschlüsselungsoperation $y_i := D_B(d+i)$ $\forall\ i \in \{0, ..., 100\}$ an.

Zur Kodierung der Gehaltsinformation wendet B auf die y_i eine Einwegfunktion f an und erhält $z_i := f(y_i)$. Gilt $z_i - z_j = 1$ für ein Paar $j \geq i$ (*), dann muß das Protokoll nochmal von vorne begonnen werden. Dies tritt bei sehr großem x sehr selten auf.

Danach schickt B die Folge $F : z_0, ..., z_b, z_{b+1} + 1, ..., z_{100} + 1$ an A. A berechnet jetzt $f(x)$ und $f(x)+1$ und vergleicht diese Werte mit den Elementen aus F.

FALL 1: $f(x) \in F \Leftrightarrow a \leq b$, da gilt: $z_i = f(y_i) = f(D_B(d+i)) = f(D_B(c-a+i))$
$\rightarrow$ da $i = a$ gilt $a \leq b$ wegen $f(x) = z_i$

FALL 2: $f(x) + 1 \in F \Leftrightarrow a \geq b$, da gilt: $z_i + 1 = f(y_i) + 1 = f(D_B(d+i)) + 1$
$= f(D_B(c - a + i)) + 1$
$\rightarrow$ da $i = a$ gilt $a \geq b$ wegen $f(x) + 1 = z_i + 1$

(*) wird deshalb geprüft, damit ausgeschlossen ist, daß sowohl $f(x)$ als auch $f(x) + 1$ in F vorkommen, denn dann wäre kein Vergleich von a und b möglich. Die Einwegfunktion wird benutzt, um A daran zu hindern, b zu berechnen. A könnte dann nämlich alle y_i bzw. $y_i + 1$ berechnen und ermitteln, ab welchem i die $y_i + 1$ auftreten.

Skatspielen über's Telefon

Das Verfahren beruht auf dem Shamir's No-Key-Protokoll.

Vorbereitung

Zuerst werden eine Primzahl p sowie paarweise verschiedene Zahlen a, a', b, b', c und c' gewählt, so daß Folgendes gilt: $aa' \equiv bb' \equiv cc' \equiv 1(mod\ p-1)$

Durchführung

Die Karten werden als Paare (i, x_i) dargestellt mit

- i - Position der Karte im Stapel
- x_i - Wert der Karte

Ausgangspunkt

Stapel $S = \{(1, x_1), ..., (32, x_{32})\}$

1. Schritt

Spieler A mischt den Stapel, indem er eine Permutation α der Menge $\{1, ..., 32\}$ auswählt
$\Rightarrow S_A = \{(\alpha(1), x_1^a mod\ p), ..., (\alpha(32), x_{32}^a mod\ p)\}$
Danach werden die Karten nach der ersten Komponente sortiert, also zuerst $(1, x_k^\alpha mod\ p)$ mit $k = \alpha^{-1}(1)$ usw.

2. Schritt

Spieler B mischt den Stapel analog zu Spieler A und erhält
$\Rightarrow S_B = \{(\beta(\alpha(1)), (x_1^a)^b mod\ p), ..., (\beta(\alpha(32)), (x_{32}^a)^b mod\ p)\}$
Danach wieder sortieren nach der ersten Komponente.

3. Schritt

Spieler C mischt die Karten und erhält
$\Rightarrow S_C = \{(\gamma(\beta(\alpha(1))), ((x_1^a)^b)^c mod\ p), ..., (\gamma(\beta(\alpha(32))), ((x_{32}^a)^b)^c mod\ p)\}$
Danach wieder sortieren nach der ersten Komponente

Austeilen der Karten

Karten für Spieler C werden nach folgendem Schema verteilt:

- A berechnet $(x_i)^{abc})^{a'} \equiv x_i^{bc}\ (mod\ p)$
- B berechnet $(x_i)^{bc})^{b'} \equiv x_i^{c}\ (mod\ p)$
- C entschlüsselt die Karte mit $x_i^{cc'} \equiv x_i\ (mod\ p)$

Analog zu diesem Schema werden die Karten für A und B verteilt.

Danach können die Karten ganz normal ausgespielt werden. Beim Aufnehmen des Skats werden dann analog zum eben beschriebenen Verfahren die Verschlüsselungen von den Karten entfernt.

Durch Bekanntgabe von a, a', b, b', c, c' sowie α, β, γ kann nach dem Spiel nachvollzogen werden, ob auch jeder Spieler die 'richtigen' Karten gespielt hat.

Primzahlerzeugung und -tests

Die Kryptographie beruht sehr stark auf der Zahlentheorie und kryptographische Verfahren bzw. -protokolle greifen oftmals auf Gesetzmäßigkeiten der Zahlentheorie im Zusammenhang mit Primzahlen zurück. Dabei ist ein ganz wesentlicher Sicherheitsfaktor, daß man möglichst große Primzahlen von 1000 Bit Länge und mehr benutzt. Dies erfordert aber zum einen Kenntnisse zur schnellen und einfachen Berechnung von Primzahlen als auch zum Test von natürlichen Zahlen auf Primalität.

Satz 1

Wenn n eine zusammengesetzte natürliche Zahl ist, dann hat n einen Primteiler, der nicht größer als $\sqrt{n}$ ist.

Probedivision

Eine natürliche Zahl n wird auf Primalität getestet, indem sie durch alle Primzahlen $p \leq \sqrt{n}$ dividiert wird.
→ Verfahren wird in der Praxis nur mit Primzahlen bis 10^6 durchgeführt
→ für $n \geq 17$ sind dafür mindestens $\sqrt{n}/log(\sqrt{n})$ Probedivisionen nötig

Bei RSA werden Primzahlen verwendet, die größer als 10^{75} sind. Dafür wären dann $\sqrt{10^{75}}/log(\sqrt{10^{75}}) \geq 0.36 \times 10^{36}$ Probedivisionen nötig, was aber nicht durchführbar ist.

Fermat-Test

Ist n eine Primzahl, so gilt $\forall a \in Z$ mit $ggt(a,n) = 1$: $a^{n-1} \equiv 1 \; mod \; n$
Der Fermat-Test kann nur feststellen, ob eine Zahl keine Primzahl ist.

Pseudoprimzahl

Ist n eine ungerade zusammengesetzte Zahl und gilt für ein $a \in Z$:
$a^{n-1} \equiv 1 \; mod \; n$, dann heißt n Pseudoprimzahl.

Carmichaelzahl

Ist n eine Pseudoprimzahl $\forall a \in Z$ mit $ggt(a,n) = 1$ und $a^{n-1} \equiv 1 \; mod \; n$, dann heißt n Carmichaelzahl. Diese Zahlen kann der Fermat-Test nicht erkennen.

→ sei $s = max\{r \in N \mid 2^r \; teilt \; n-1\}$
→ setze $d = (n-1)/2^r$

Miller-Rabin-Test

Ist n eine Primzahl und $a \in Z$ mit $ggt(a,n) = 1$, dann gilt mindestens eine der beiden Bedingungen:

1. $a^d \equiv 1 \; mod \; n$
2. $\exists r \in \{0, 1, ..., s-1\}$ mit $a^{2^r d} \equiv -1 \; mod \; n$

⇒ findet man also ein $a \in Z$ mit $ggt(a,n) = 1$, für das weder (1) noch (2) gilt, dann ist bewiesen, daß n keine Primzahl ist.

Zeuge

a heißt dann Zeuge gegen die Primalität von n.

Erzeugung von Primzahlen

Sei n eine k-bit lange Binärzahl. Zuerst werden das erste und das letzte Bit von n auf 1 gesetzt. Die restlichen k-2 Bits werden gemäß der Gleichverteilung zufällig gesetzt.
Dann wird geprüft, ob n durch eine Primzahl unterhalb einer Schranke B teilbar ist. Üblicherweise ist $B = 10^6$. Die Primzahlen werden für diesen Test aus entsprechenden Tabellen entnommen.
Wurde kein Teiler für n gefunden, dann wird der Miller-Rabin-Test auf n mit t Wiederholungen angewendet. Wenn dabei kein Zeuge gegen die Primalität von n gefunden wurde, dann gilt n als Primzahl. Andernfalls ist bewiesen, daß n zusammengesetzt ist und der Test muß mit einer neuen Zufallszahl gemacht werden.
Die Wiederholungszahl t wird so gewählt, daß die Fehlerwahrscheinlichkeit des Miller-Rabin-Tests hinreichend klein ist. Bei der Suche nach einer Primzahl mit mehr als 1000 Bits reicht es für eine Fehlerwahrscheinlichkeit von weniger als $(1/2)^{80}$ aus, $t = 3$ zu wählen.
Die Auswahl der Primzahlschranke B hängt davon ab, wie schnell bei der verwendeten Hard- und Software eine Probedivision im Verhältnis zu einem Miller-Rabin-Test ausgeführt werden kann.