Anmerkungen zu Rabins Fingerprintmethode

Dirk Meister

In diesem Dokument beschäftige ich mich näher mit Rabins Fingerprintingmethode. Das Verfahren wurde 1981 von Rabin entwickelt[3] und es arbeitet wie folgt[2]:

Sei S ein Bitstring mit m Bits $[b_1, b_2, ..., b_m]$. Der Bitstring S kann einem Polynom in Z_2 vom Grad $m-1$ in t wie folgt zu geordnet werden $S(t) = b_1 t^{m-1} + b_2 t^{m-2} + ... + b_{m-1} t + b_m$ Sei weiterhin ein irreduzibles Polynom P(t) vom Grad k gegeben: $P(t) = p_1 t^k + p_2 t^{k-1} + ... + p_{k-1} t + k_m$. Chan und Lu[2] geben einen Algorithmus an, mit dem ein solches Polynom gefunden werden kann.

Für ein fixes Polynom $P(t)$ ist der Fingerprint für den Bitstring S ein Polynom vom Grad $k-1$ und ist definiert als $f(t) = S(t) \, mod \, P(t)$.

Broder stellte ein effizientes Implementierungschema vor[1], das eine im Voraus berechnete Lookup-Tabelle zur Beschleunigung der Berechnung verwendet und pro Byte mit zwei XOR-Operationen und einem Index-Lookup auskommt. Chan und Lu beweisen die Korrektheit des Schemas. Allerdings basiert dieses Verfahren darauf, jeweils ein 32-Bit-Wort dem Fingerprint hinzufügen. Dieses Vorgehen ist beim Chunking nicht möglich, da dadurch das Chunking nur Verschiebungen um eine Anzahl von Worten erkennen würde.

Im weiteren stelle ich eine Implementierung mit Lookup-Tabelle vor, in der einzelne Bytes dem Fingerprint hinzugeführt werden. Außerdem zeige ich eine Implementierung als rollende Hashfunktion und führe den Korrektheitsbeweis.

Die Umsetzung des per Lookuptabelle beschleunigten Schemas mit der Möglichkeit einzelne Bytes hinzufügen, funktioniert wie folgt. Sei $P(t)$ ein zufällig gewähltes, irreduzibles Polynom in Z_2 vom Grad 64, $S = s_1, s_2, ..., s_n$ ein Bitstring der Länge n und $S(t)$ das S zugeordnete Z_2-Polynom in t.

Außerdem wird die Rabin-Fingerprinting-Funktion als f bezeichnet. Die Funktion $concat(x, y)$ verbindet zwei Bitstrings. Ich gehe im Folgenden stets davon aus, dass Fingerprints eine Länge von 64 Bit haben.

Der Fingerabdruck einer Verkettung zweiter Strings A und B kann mit $f(concat(A, B)) = f(concat(f(A), B)$ berechnet werden. Wenn $f(A)$ und $f(B)$ gegeben sind, so entspricht dies: $f(concat(A, B)) = f(f(A) \cdot f(t^l)) + f(B)$[1].

Satz 0.1. *Sei F_m der Fingerabdruck des Bitstrings $[s_1, s_2, ..., s_m]$. Der Fingerabdruck F_m besteht aus den Bytes $F_{(m,8)}, F_{(m,7)}, ..., F_{(m,1)}$.*

$F_m(t)$ wird berechnet durch

$$\begin{aligned} F_m(t) =& F_{(m-1)}(t)t^8 + T[F_{(m-1,7)}(t)t^{64}] + s_m \, mod \, P(t) && \text{wenn } 0 < m \leq n \\ F_0(t) =& 0(t) && \text{wenn } m = 0 \end{aligned}$$

T ist eine im voraus berechnete Lookup-Tabelle:

$$\forall 0 \leq x \leq 255 : T[x(t)] = x(t)t^{64} \bmod P(t)$$

Beweis von Satz 0.1. Wenn für einen Bitstring $A = [a_1, ..., a_{63}]$ gilt:

$$\begin{aligned} & F_n \\ & = f([a_1, ..., a_l]) \\ & = a_1 t^{l-1} + a_2 t^{l-2} + ... + a_{l-1}t + a_l \bmod P(t) \\ & = r_1 t^{63} + r_2 t^{62} + ... + r_{62}t + r_{63} \end{aligned}$$

dann gilt auch:

$$\begin{aligned} & F_{l+8} \\ & = f([a_1, ..., a_{l+1}]) \\ & = f(f([a_1, ..., a_l])t^8 + f([a_{l+1}, ..., a_{l+8}])) \bmod P(t) \\ & = [a_1, ..., a_l](t)t^8 + [a_{l+1}, ..., a_{l+8}](t) \bmod P(t) \\ & = [a_1, ..., a_l](t)t^8 \bmod P(t) + [a_{l+1}, ..., a_{l+8}](t) \bmod P(t) \\ & = (r_1 t^{71} + r_2 t^{70} + ... + r_{62}t^9 + r_{63}t^8 + [a_{l+1}, ..., a_{l+8}](t)) \bmod P(t) \\ & = [r_1, r_8]t^{64} \bmod P(t) + [r_9, ..., r_6 3]t^8 + [a_{l+1}, ..., a_{l+8}](t)) \\ & = F_{(l,1)}(t)t^{64} \bmod P(t) + F_{(l,2)}(t)t^{56} + ... + F_{(l,7)}(t)t^{16} + F_{(l,8)}(t)t^8 + [a_{l+1}, ..., a_{l+8}](t) \\ & = T[F_{(l,1)}(t)] + F_{(l,2)}(t)t^{56} + ... + F_{(l,7)}(t)t^{16} + F_{(l,8)}(t)t^8 + [a_{l+1}, ..., a_{l+8}](t) \end{aligned}$$

□

Die Operationen über Polynome können einfach auf Hardwareoperationen umgesetzt werden: Die Addition entspricht dem XOR, die Multiplikation mal t entspricht einem Linksshift. Damit entspricht die letzte Zeile $F_{l+8} = (F_l << 8 | [a_{l+1}, ..., a_{l+8}]) \wedge T[F_l >> 56]$. $[a_{l+1}, ..., a_{l+8}]$ entspricht dem Byte, das dem Fingerabdruck hinzugefügt wird.

Es ist nicht direkt ersichtlich, dass Rabin-Fingerprinting als rollendes Hash-Verfahren eingesetzt werden kann. Dazu zeige ich erst wie das erste Element vom Anfang des Fensters aus dem Fingerabdruck herausgerechnet werden kann.

Satz 0.2. *Sei $F_{(i,j)}$ der Fingerprint des Bitstrings $[a_i, a_2, ..., a_j]$. $F_{(i,j)}$ besteht aus den Bytes $f_{(i,j,8)}, f_{(i,j,7)}, ..., f_{(i,j,1)}$. Sei w die Größe des Fensters von $F_{(i,j)}$. $F_{(i,j)}(t)$ wird berechnet aus $F_{(i-8,j)}$ durch*

$$F_{(i,j)} = F_{(i-8,j)} + U[[a_{i-8}, ..., a_{i-1}]]$$

wobei U eine im vorausberechnete Lookup-Tabelle:

$$\forall 0 \leq x \leq 255 : U[x(t)] = x(t)t^w \, mod \, P(t)$$

Beweis von Satz 0.2. Wenn für einen Bitstring $A = [a_1, ..., a_{63}]$ gilt:

$$\begin{aligned} & F_{(i,j)} \\ & = f([a_i, ..., a_j]) \\ & = a_i t^{w-1} + a_{i+1} t^{w-2} + ... + a_1 t + a_j \bmod P(t) \\ & = r_1 t^{63} + r_2 t^{62} + ... + r_{62} t + r_{63} \end{aligned}$$

dann gilt auch:

$$\begin{aligned} & = F_{(i-8,j)} + U[[a_{i-8}, ..., a_{i-1}]]. \\ & = a_{i-8} t^{w+7} + a_{i-7} t^{w+6} + ... + a_{i-1} t^{(w)} + a_i t^{w-1} + ... + a_{j-1} t + a_j \bmod P(t) + \\ & \quad U[[a_{i-8}, ..., a_{i-1}]] \\ & = a_i t^{w-1} + a_{i+1} t^{w-2} + ... + a_{j-1} t + a_j \bmod P(t) + ([a_{i-8}, ..., a_{i-1}](t) t^w) \bmod P(t) \\ & = r_1 t^{63} + r_2 t^{62} + ... + r_{62} t + r_{63} + ([a_{i-8}, ..., a_{i-1}](t) t^w) \bmod P(t) + U[[a_{i-8}, ..., a_{i-1}]] \\ & = F_{(i,j)} + ([a_{i-8}, ..., a_{i-1}](t) t^w) \bmod P(t) + U[[a_{i-8}, ..., a_{i-1}]]. \\ & = F_{(i,j)} + U[[a_{i-8}, ..., a_{i-1}]] + U[[a_{i-8}, ..., a_{i-1}]]. \\ & = F_{(i,j)} \end{aligned}$$

□

Kombiniert ergibt dies den abschließenden Korrolar:
Korrolar 0.1. *Sei $F_{(i,j)}$ der Fingerprint des Bitstring $[s_i, s_2, ..., s_j]$ und w die Größe des Fensters von $F_{(i,j)}$. $F_{(i,j)}$ wird berechnet aus $F_{(i-8,j-8)}$ durch*

$$F_{(i,j)} = concat(F_{(i-8,j-8)} + U[[s_{i-8}, ..., s_{i-1]}], [s_{j-7}, ..., s_{j]}])$$

Zuerst wird mit Hilfe von Satz 0.2 das älteste Byte des aktuellen Fensters aus dem Fingerabdruck herausgerechnet, dann wird mit 0.1 das nächste Byte des Bitstrings hinzugeführt. Mit Hilfe dieses Korrolars können Rabin Fingerprints zum Chunking eingesetzt werden.

Literatur

[1] A. Broder. *Some applications of Rabin's fingerprinting method*, pages 143–152. Springer Verlag, 1993.

[2] C. Chan and H. Lu. Fingerprinting using poloynomial (rabin's method). CMPUT690 Term Projekt, December 2001.

[3] M. O. Rabin. Fingerprinting by random polynomials. Technical report, Center for Research in Computing Technology, 1981.