РусКрипто 2013

конференция
РусКрипто'2013

http://www.ruscrypto.ru/conference/

Неравномерные распределения ключей, схемы парольной защиты и цепи Маркова

Чиликов А. А.

Пусть K — множество ключей

Тогда:
- $|K| >= |M|$
- $P(k) = Const$

=>

- $P(m|k) = P(m)$

т. е. равномерное распределение ключей — это хорошо :)

Проблема: Как получить равномерное распределение?

Пример: $k = f(pwd,salt)$ — 128 bits

Pwd — 8 chars, alnum ($62^8 \sim 2^{48}$)
Salt — 4 bytes (2^{32})
$\Rightarrow |K| \sim 2^{80}$
$p(k) \sim 2^{-80}$ или 0
Но может быть и $2 \cdot 2^{-80}$ из-за коллизий

А действительно ли нужна равномерность?

Нет равномерности — нет стойкости по Шеннону. Но может быть вычисительная!

Исследуем на примере парольных схем.

Пусть

- используются хорошие алгоритмы
- Есть данные для проверки
- T(genPwd) « T(verifyPwd)

Тогда лучший алгоритм — перебор

```
FOR i = 0; i < |K|; i++
 Pwd = Generate(i)
 Ok = VerifyPwd( Pwd )
 IF Ok return Pwd
Return False
```

Время работы минимально, если P(pwd[i]) >= P(pwd[j]) при i < j

Для построения алгоритма нужна вероятностная модель «источника» паролей

Марковские цепи:

S — множество состояний
{s(t)} — случайная последовательность состояний

$$P(s(t+1)=j \mid s(t)=i) = P(s(t+1)=j \mid s(t)=i \ \& \ s(t-1) = i' \ \& \ ...)$$

«Источник»:

$f(s) = w$ — слово над алфавитом A

$F(s(1), s(2), \ldots) = f(s(1)) \| f(s(2)) \| \ldots$

Естественные требования:

$|S|$ - конечно

$P(|F(s1, \ldots sn, ...)|) = 0$ если $|F(s1, \ldots sn, ...)|$ - бесконечна

Допустимое слово — если $P(w) > 0$

Любой регулярный (автоматный) язык порождается таким «источником»

Но — не любое распределение вероятностей!

Упрощения:

Одно терминальное состояние.
Любой процесс с вероятностью 1 оказывается в этом состоянии.
Можно считать, что $|f(s)| = 1$ или 0

$P(A^m) = c1 * x1^m + \ldots + cN * xN^m$

$|x1|, \ldots |xN| < 1$ — собственные значения матрицы переходов.

т. е.: Совокупная вероятность получения длинных слов экспоненциально убывает

Характеристики переборного алгоритма:

- Вероятность неудачи
- Среднее время работы

Оптимальный вариант — генерировать по убыванию вероятности. Но это не всегда возможно.

Варианты:

- Генерация по убыванию вероятности
- Генерация в произвольном порядке с глобальным порогом
- Генерация в произвольном порядке с локальным порогом
- Многопороговая генерация

$$T = P(w1) + 2*P(w2) + \ldots + N*P(wN)$$

Для оптимальной генерации:

$$T(p) \sim O(\exp(X*(1-Y) * \log p))$$

Для глобального порога:

$$T(p) \sim O((\exp X*Y * \log p))$$

$$N(p) \sim O(\exp(Y * \log p))$$

$T = P(w1) + 2*P(w2) + \dots + N*P(wN)$

Для локального порога:

$T(p) \sim O(\exp(X' * Y' * \log p))$

Многопороговая генерация:

Если $T(Gen) \ll T(Verify)$

Почему это не всегда работает:

- Большое число состояний
- Определение параметров
- Смешивание различных источников

Открытые вопросы

- Упрощение автомата
- Аппроксимация распределения
- Разделение различных источников

Вопросы?

chilikov@passware.com