乱数と擬似乱数の生成技術

DS5 坂田誠也

目次

概要

- 真の乱数と擬似乱数に大まかに大別できる
- 乱数は自然の事象から生成するため予測や再現は極めて困難
- 暗号技術には乱数または暗号論的擬似乱数以外を使ってはいけない
- 弱い擬似乱数は出力される値を予測したり再現することが可能

スライドはmarkdown形式でGitHubで管理してます。
typoとか誤りの報告のPR歓迎!
スライド

テーマの動機

- 認証周りの実装で擬似乱数生成器ライブラリの選定をすることがあった
- 擬似乱数生成器をどういった基準で選べばいいのか？
- 基準を考えるには擬似乱数生成のアルゴリズムをある程度知っておく必要がある
- あまり普段考えないけど、いざいう時知っておいた方がいいかも？
- ~~CTF について発表の予定が、直近の CTF で想定以上に解けなかったのでボツにした代替案~~

お前誰よ

- 名前
  - 坂田誠也(Sakata Seiya)
- 所属
  - DS5
- 興味がある Tech 的なもの
  - サーバサイド技術全般、セキュリティ、プログラミング言語
- 好き(Tech 以外)
  - アニメ
  - ゲーム（ポケモン、TCG、ボードゲーム）

乱数とは

規則性が無く予測が不可能な数

- 性質
  - 無作為性: 統計的な偏りが無く、でたらめな数列になっていること
  - 予測不可能性: 過去の数列から次の数を予測できないこと
  - 再現不可能性: 同じ数列を再現できないこと。再現するためには、数列そのものを保存しておくしかない。
- 使用例
  - シミュレーション、ソーシャルゲームのガチャ、DNS クエリ

乱数とは

乱数をどうやって生成する？

- コンピュータは決定的な計算しか数列を生成することができない...
  - 決定的:その動作が予測可能であること
- 方法１
  - ハードウェアを用いて自然現象・物理現象から生成する（真の乱数）
- 方法２
  - ソフトウェアを用いて確定的な計算で生成する（擬似乱数）

乱数と擬似乱数の種類

	無作為性	予測不可能性	再現不可能性	暗号技術に使用
真の乱数	○	○	○	○
強い擬似乱数	○	○	×	○
弱い擬似乱数	○	×	×	×

乱数生成器

ハードウェアを用いて自然現象から真の乱数を生成する。（例：熱雑音、宇宙線など）

予測や再現は事実上不可能なので、そのようなハードウェアを「乱数生成器」と呼ぶ

- ブラックボックスなのと万が一 NSA がバックドアを仕込んでいるかもしれない？
- そのまま使うのでは無く、別の手法と合わせて使うのがベストらしい（要出典）

乱数生成器

- 実装例
  - Linux カーネルの/dev/random や/dev/urandom
    ※https://linuxjm.osdn.jp/html/LDP_man-pages/man4/random.4.html
  - Intel や AMD の CPU の RDRAND 命令
    ※https://www.isus.jp/security/drng-guide/

暗号論的擬似乱数生成器(CSPRNG)

- 性質
  - 統計的に無作為であることが保証されていること
  - その状態の一部または全部が明らかになっても、明らかにされた状態より以前に生成された乱数列は再現できないこと
    - これがただの擬似乱数生成器との違い
- 代表的なアルゴリズム
  - ANSI X9.17、ANSI X9.21、ANSI X9.62

暗号論的擬似乱数生成器(CSPRNG)

ANSI X9.13 のアルゴリズム

図が載っているので見るとわかりやすい

http://www.spiritek.co.jp/spkblog/2017/02/09/暗号技術入門12-乱数/

暗号論的擬似乱数生成器(CSPRNG)

- 実装例
  - PHP の random_int()
  - Java の java.security.SecureRandom
  - Ruby の SecureRandom.rand
  - その他多くの言語の標準ライブラリ

（弱い）擬似乱数生成器

- 周期性があり、必要十分な乱数列を集めることで内部状態を把握することが可能
- 暗号技術目的には絶対に使ってはいけない
- 代表的なアルゴリズム
  - 線形合同法、メルセンヌ・ツイスター、Xorshift

線形合同法(Linear congruential generators: LCGs)

- 70 年代〜80 年代にかけて非常によく使われた擬似乱数生成器
- 過去にポケモンの乱数生成で使用されていた（現在はメルセンヌ・ツイスターと別の擬似乱数生成器が使われているらしい）
- 実装例（純粋な LCG かは未調査）
  - C の rand()
  - Java の java.util.Random
  - PHP の rand()

線形合同法(Linear congruential generators: LCGs)

- アルゴリズム(A, C, Mは定数で$A < M, C < M$)
  $R_{n+1} = (A \times R_n + C) \bmod M$
  最後に出力した擬似乱数を内部状態で使用している。

例）$A = 3$、$C = 0$、$M = 7$、R の初期値の$seed = 6$とすると

$$
\begin{aligned}
R_0 &= (A \times seed + C) \bmod M \\
&= (3 \times 6 + 0) \bmod 7 \\
&= 18 \bmod 7 \\
&= 4
\end{aligned}
$$

線形合同法(Linear congruential generators: LCGs)

$$
\begin{aligned}
R_1 &= (3 \times R_0 + 0) \bmod 7 \\
&= (3 \times 4 + 0) \bmod 7 \\
&= 12 \bmod 7 \\
&= 5 \\
R_2 &= 15 \bmod 7 \\
&= 1 \\
R_3 &= 3
\end{aligned}
$$

線形合同法(Linear congruential generators: LCGs)

線形合同法の（擬似乱数生成器としての）問題点

1. 擬似乱数の値は必ず0〜$M - 1$の範囲に収まる
   - A = 0 の場合: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...（周期は 1）
   - A = 1 の場合: 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,...（周期は 1）
   - A = 2 の場合: 5, 3, 6, 4, 3, 6, 5, 3, 6, 5,...（周期は 3）
   - A = 3 の場合: 4, 5, 1, 3, 2, 6, 4, 5, 1, 3,...（周期は 6）
   - A = 4 の場合: 3, 5, 6, 3, 5, 6, 3, 5, 6, 3,...（周期は 3）
   - A = 5 の場合: 2, 3, 1, 5, 4, 6, 2, 3, 1, 5,...（周期は 6）
   - A = 6 の場合: 1, 6, 1, 6, 1, 6, 1, 6, 1, 6,...（周期は 2）

線形合同法(Linear congruential generators: LCGs)

LCG の周期は（初期 seed に関係なく！）高々$2^{32} = 4294967296$しかないので、数分で使い切ってしまう。

2. 偶数と奇数が交互に出る場合がある
   mod偶数で使った場合、最下位ビットは同じものが出続ける or 0 と 1 が交互に出る
   上位ビットだけ使えばマシになる？（要出典）

線形合同法(Linear congruential generators: LCGs)

- 使用されていた or 使用されていそうな例
  - ポケモン
    https://www.slideshare.net/Blastoise_X/ss-71991306
  - カルドセプトサーガ
    ダイスの目が偶数・奇数を繰り返すバグがあった
    https://ja.wikipedia.org/wiki/カルドセプト_サーガ
  - ファイナルファンタジータクティクスアドバンス
    https://gamefaqs.gamespot.com/gba/560436-final-fantasy-tactics-advance/faqs/26262

メルセンヌ・ツイスター（Mersenne twister: MT）

- 1996 年に松本眞と西村拓士（敬称略）によって発表された擬似乱数生成器
- 特徴
  - 周期が$2^{19937} - 1 \fallingdotseq 4.3 \times 10^{6001}$でとても長い
    - 観測可能な宇宙の中にある基本粒子の数が最大推定10^{85}
    - 1 ナノ秒ごとに 1 億台のコンピュータで同時に乱数を取得しても宇宙の寿命までに 1 周期が終わらない
  - 1 周期で 623 次元空間に均等分布することが証明されている

メルセンヌ・ツイスター（Mersenne twister: MT）

- 実装例
  - PHP の mt_rand()
  - Python の random.random()
    https://github.com/python/cpython/blob/3.8/Lib/random.py
  - Ruby の Random
- デメリット
  - 計算が比較的遅くなる

メルセンヌ・ツイスター（Mersenne twister: MT）

- アルゴリズム
  - 現在では改良された SFMT が使われていますが、ここでは元の実装(MT19937)を対象とします。
  - 確実に説明できる自信がないのと、説明するためには行列計算と XOR とビット演算から説明しないといけないので割愛（いつか…）

メルセンヌ・ツイスター（Mersenne twister: MT）

- C 言語でのオリジナルの実装(genrand_int32()参照)
  http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/CODES/mt19937ar.c
- MT の論文
  http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/ARTICLES/mt.pdf
- MT19937 のアルゴリズムについて正しそうな解説（保証はしません）
  https://narusejun.com/archives/5/

Xorshift

- フロリダ州立大学の George Marsaglia が考案
- 特徴
  - XOR とシフト演算のみで実装されており簡潔（そして高速）
  - 周期も最大で$2^{128} - 1$持っており、偏りが少ない
- 実装例
  - Google Chrome の JavaScript エンジン V8 の Math.random https://v8project.blogspot.jp/2015/12/theres-mathrandom-and-then-theres.html

Xorshift

- アルゴリズム
  - 説明するには XOR とビット演算(ry と多分時間ないので wikipedia を読んで https://ja.wikipedia.org/wiki/Xorshift

参考文献

特に参考にさせていただきました。

著者:結城 浩
書籍名:暗号技術入門 第３版　秘密の国のアリス
出版社(発行年月日):SB クリエイティブ(2015/8/25)

あなたの使っている乱数、大丈夫？-危ない標準乱数と、メルセンヌ・ツイスター開発秘話-
松本　眞
2014/11/18
第 50 回市村学術賞記念　先端技術講演会

まとめ

- 自然現象から生成した乱数列だけが（真の）乱数で他は擬似乱数
- 暗号技術には乱数生成器と暗号論的擬似乱数生成器(CSPRNG)以外を使ってはいけない
- 線形合同法(LCG)で実装された擬似乱数生成器は暗号技術以外でも使用して問題ないか検討する

Thank you for listening!

Appendix:LCG の擬似乱数列を予測する

問題

LCG で内部で使用される A,C,M,初期 seed が不明な状態で数列$R_{0...6}$が与えられる。添字は出力された順番を表し、R_0は初期 seed を使用して生成されたものとする。
この場合に次に出力される擬似乱数R_7を予測することは可能か？

考えてみてください。（条件付きで可能です）