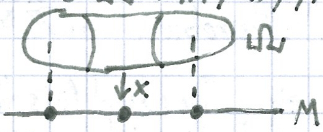
CRYPTO2-Übersicht: VL08+09: Zufall

↳VL08: Definition PRG, Distinguisher, Predictor; VL09: Blum-Blum-Shub & Beweis (b.w.)

- **Recap Wk.rechnung:**
  VL: Wk.raum: $\Omega = \{w_1, \dots, w_n\}$ Wk.verteilg.: $Pr : \Omega \to [0,1] \in \mathbb{R}$ mit $\sum_{w \in \Omega} P[w] = 1$. Wenn $A \subseteq \Omega$ Ereignis, $Pr[A] = \sum_{\alpha \in A} Pr[\alpha]$
  Gleichförmige Wk.verteilg.: $U_\Omega : \Omega \to [0,1]$, $w \mapsto 1/|\Omega|$
  de.wiki: Zufallsvariable = messbare Fkt. v. einem Wk.raum in einen Messraum
  VL: Zufallsvar. $X : \Omega \to M$ (M Menge) induziert auf M eine Wk.verteilg. Pr_X: $Pr_X[m] := Pr[X^{-1}(m)]$ „von X induzierte Wk.vert."
  de.wiki: Bsp. Würfelsumme zweier Würfelwürfe: $S : \Omega \to \mathbb{N}$; $(x,y) \mapsto x+y$, wobei $\Omega = \{(1,1), \dots, (6,6)\}$
  VL:



- **Def. ε-Distinguisher:** Seien $X : \Omega_X \to M$ und $Y : \Omega_Y \to M$ zwei Zufallsvar., $\varepsilon > 0$. Ein ε-Distinguisher ist eine in Polynomialzeit berechenbare Abbildung $D : M \to \mathbb{B}$ mit $|Pr[D(X)=1] - Pr[D(Y)=1]| \geq \varepsilon$ wobei $Pr[D(X)=1] := Pr_X[D^{-1}(1)] = Pr[X^{-1}(D^{-1}(1))]$
  ↳ wähle per X zufällig Elemente aus M und setze in D ein. D darf dabei ausdrücklich auch eine Zufallskomponente haben!

  💡 D.h. die Wk., dass D auf von X generierten Elem. aus M 1 returned ist um ε größer/kleiner als für Y.

- **Def. PRG/Pseudo Random Generator:** $G^{(l)} : \underbrace{\mathbb{B}^l}_{\text{„Saat"}} \to \mathbb{B}^{l+\underbrace{s(l)}_{\text{„Stretch"}}}$ mit $s(l) \geq 1$.
  Schön wäre es, wenn die Ergebnisse gleichförmig verteilt wären, d.h. $G(\mathbb{B}^l)$ nicht von $U_{\mathbb{B}^{l+s(l)}}$ zu unterscheiden wäre, doch das ist unmöglich! Ausweg: Fordere stattdessen, dass $G(\mathbb{B}^l)$ nicht mit polynomiellem Aufwand von $U_{\mathbb{B}^{l+s(l)}}$ zu unterscheiden ist! Formal: Zu jedem $\varepsilon > 0$ gibt es ein L, sodass für alle $l > L$ $G^{(l)}$ und $U_{\mathbb{B}^{l+s(l)}}$ nicht ε-unterscheidbar sind.
  Die Def. eines ε-Distinguishers eingesetzt bedeutet dies: Für alle polynomiellen $D : \mathbb{B}^{l+s(l)} \to \mathbb{B}$ ist $|Pr[D(G^{(l)})=1] - Pr[D(U_{\mathbb{B}^{l+s(l)}})=1]| < \varepsilon$.

- **Def. (i,ε)-Predictor:** Sei X Zufallsvariable auf $\mathbb{B}^l$. Schreibe X komponentenweise: $X = X_1 \dots X_l$ (X_i = Komponente von X). Sei $\varepsilon > 0$ und $2 \leq i \leq l$.
  Ein (i,ε)-Predictor ist eine Abbildung $P : \mathbb{B}^{i-1} \to \mathbb{B}$ mit $Pr[P(X_1 \dots X_{i-1}) = X_i] \geq \frac{1}{2} + \varepsilon$.
  💡 D.h. ein Predictor kann das i-te Bit mit Wk. $> \frac{1}{2}$ vorhersagen.

  $\boxed{1}$ Gibt es einen (i,ε)-Predictor für eine Zufallsvariable X auf $\mathbb{B}^l$, dann gibt es einen ε-Distinguisher von X und $U_{\mathbb{B}^l}$:
  $$D : \mathbb{B}^l \to \mathbb{B};\ D(b_1, \dots, b_{i-1}, b_i, \dots, b_l) := \begin{cases} 1 & \text{wenn } P(b_1, \dots, b_{i-1}) = b_i \\ 0 & \text{sonst} \end{cases}$$

  $\boxed{2}$ Gibt es einen ε-Distinguisher für X und $U_{\mathbb{B}^l}$, dann gibt es einen $(i, \varepsilon/l)$-Predictor:
  $$P(x_1, \dots, x_{i-1}) = \begin{cases} u_i & \text{wenn } D(x_1, \dots, x_{i-1}, u_i, \dots, u_l) = 1 \\ 1+u_i & \text{sonst} \end{cases}$$
  → wobei die u_i gleichförmig zufällig aus $\mathbb{B}$ gewählt!

(git: bbs.py)
- **Blum-Blum-Shub-Generator (BBS-Generator):** → BBS's Sicherheit lässt sich auf das Faktorisierungsproblem reduzieren 💡 (Äquivalenz) (en) (de)
  Seien p, q zwei große Primzahlen mit $p,q \equiv 3 \pmod 4$ und geheim.
  de.wiki: Es sollte $2 < p/q < 1000$ gelten und $p \pm 1$ und $q \pm 1$ sollten jew. einen Primfaktor $> \sqrt[4]{n}$ haben.
  [1] Sei $n = p \cdot q$ (öffentlich) und $s_0 \in \mathbb{Z}_n^*$ der Seed/Startwert (d.h. $ggT(s_0, n) \stackrel{!}{=} 1$). Es sollen L Zufallsbits generiert werden.
  [2] Generiere $s_{i+1} = s_i^2 \bmod n$ für $i = 1, \dots, L$.
  [3] Behalte nur die jeweiligen least significant bits: $b_i = s_i \bmod 2$
  [4] Output: $(b_1, b_2, \dots, b_L) = BBS_L(s_0)$ (💡 Achtung: keine Rückgabe von b_0 💡)
  de.wiki: Bsp.: $p=7$, $q=11$, $n=77$, $s_0=64$: $64 \to 15, 71, 36, 64, \dots$ (ab hier wiederholt es sich!)
  $BBS_4(64) = (1, 1, 0, 0)$

- **Beweis: BBS ist sicher:**

Faktorisierungsproblem $\leq_{[2]}$ Quadratisches Resteproblem (QRP) $\leq_{[1]}$ BBS-"Problem"

reduziere auf · reduziere auf

[1] Den.wiki

leicht lösbar, wenn man $\sqrt{\ }$ ziehen kann

Dabei sei das Quadratische Resteproblem wie folgt definiert:
Geg. n und $x \in \mathbb{Z}_n^*$, gibt es ein $y \in \mathbb{Z}_n^*$ mit $y^2 = x$?
D.h.: Ist x ein quadratischer Rest modulo n, oder nicht?

Red. QRP auf BBS: [1] Wenn man einen ε-Distinguisher D konstruieren könnte, der $BBS(\mathbb{Z}_n^*)$ und $U_{\mathbb{B}^L}$ unterscheidet, so könnte man das Quadratische Resteproblem in $\mathbb{Z}_n^*$ lösen:

Frage: Ist $x \in \mathbb{Z}_n^*$ ein Quadrat(-ischer Rest), d.h. ist $x \in (\mathbb{Z}_n^*)^2$?
Angenommen, D sei ein ε-Distinguisher von $BBS(\mathbb{Z}_n^*)$ und $U_{\mathbb{B}^L}$.
D.h. $D(b_1, \dots, b_L) = 1$ mit Wk. $\frac{1}{2} + \varepsilon$, wenn $\exists x$: $BBS(x) = (b_1, b_2, \dots, b_L)$. $\approx$ iterate.^{2}
Baue einen Distinguisher $\hat{D}$ mit $\hat{D}(w_1, \dots, w_L) = D(w_L, \dots, w_1)$. $\approx$ iterate$^{2^{-1}}$.
Wir wissen: Dann gibt es auch einen $(i, \varepsilon/L)$-Predictor P für $\hat{D}$, für den gilt: $P(w_1, \dots, w_{i-1}) = w_i$ mit Wk. $\frac{1}{2} + \varepsilon/L$.
! P kann jetzt quasi Wurzeln ziehen, d.h.:
Wenn $P(\underbrace{reverse(BBS(x))}_{\text{(in VL: "....")}}) = \text{least-sign-bit}(\sqrt[\#]{x^2})$ ← „berechnet" uns P
$= \text{least-sign-bit}(x \text{ oder } -x)$
$= x \bmod 2 \underbrace{\neq}_{\text{und nicht...}} -x \bmod 2$ (da n ungerade), $(3 \cdot 3 = 9 \equiv 1)$
dann ist x ein Quadrat
(mit Wk. $\frac{1}{2} + \varepsilon/L$ versteht sich).
Warum ist das so?:
Das Quadrieren auf $(\mathbb{Z}_n^*)^2$ ist ein Gruppenisomorphismus,
d.h. wenn $x \in (\mathbb{Z}_n^*)^2$, existiert ein $y \in (\mathbb{Z}_n^*)^2$ mit $x = y^2$.
Wir schreiben: $y = \sqrt[\#]{x}$, Hauptwurzel.
P berechnet uns genau diese Hauptwurzel, da es ja auf Mustern der Form $(\dots, x^{32}, x^{16}, x^8, \dots)$, sprich auf Hauptwurzeln "trainiert" wurde (meine Worte).

[2] Wenn man mod n Wurzeln ziehen könnte, so könnte man n faktorisieren:

Bsp. aus der VL: $2 \xrightarrow{\text{quadr.}} 4 \xrightarrow{\text{Wurzel}} 5$ ↯ Sage gibt wieder 2 zurück ↯
$2^2 \equiv 4 \equiv 5^2 \pmod{21} \Rightarrow 5^2 - 2^2 \equiv 0 \pmod{21}$
$\Rightarrow 21 \mid (5^2 - 2^2)$
$\Rightarrow 21 \mid (5-2) \cdot (5+2)$ ← „habe 21 faktorisiert" (in Anführungszeichen!)
$\Rightarrow ggT(21, 5-2) = 3 = p$
$\wedge\ ggT(21, 5+2) = 7 = q$ $\blacksquare$
(vorausgesetzt: $21 \nmid (5-2) \wedge 21 \nmid (5+2)$,
vgl. CRYPTO1: Faktorisierung)