2.9 Resümee

Die Abschnitte 2.1 bis 2.7 ergaben ein Vorhersageverfahren, das so abläuft:

1. Der Kryptoanalytiker findet durch Klartextraten ein Stück der Schlüssel-Bitfolge, so lange, bis sich eine geeignete lineare Relation aufstellen lässt (NOETHERsches Prinzip).

2. Er sagt mit Hilfe dieser linearen Relation weitere Schlüsselbits voraus.

3. Erweisen sich vorausgesagte Bits als falsch (weil der Klartext an dieser Stelle aufhört, sinnvoll zu sein), muss der Kryptoanalytiker wieder etwas Klartext raten und damit die Parameter adjustieren; dann kann er weiter vorhersagen.

Dieses Verfahren ist für die „klassischen“ Zufallsgeneratoren effizient, also für Kongruenzgeneratoren – auch bei unbekanntem Modul – und für Schieberegister – auch nichtlineare. „Effizient“ bedeutet hier auch, dass die benötigte Menge von bekanntem oder erratenem Klartext klein ist.

Das Fazit daraus ist, dass für kryptographisch sichere Zufallserzeugung niemals der Zustand des Zufallsgenerators direkt als Output verwendet werden sollte; vielmehr ist eine Transformation dazwischen zu schalten. Abschnitt 2.8 zeigt exemplarisch, dass das schlichte Unterdrücken einiger Bits, das „Stutzen“ oder die „Dezimierung“, als Output-Transformation aber auch nicht ohne weiteres ausreicht. Bessere Output-Transformationen werden in den folgenden Abschnitten behandelt.

Die Grauzone zwischen dem, was dem Kryptoanalytiker Erfolg garantiert, und dem, was den Kryptologen ruhig schlafen lässt, ist freilich sehr breit. Auf jeden Fall sollten besser beide Prozesse

- Zustandsänderung,

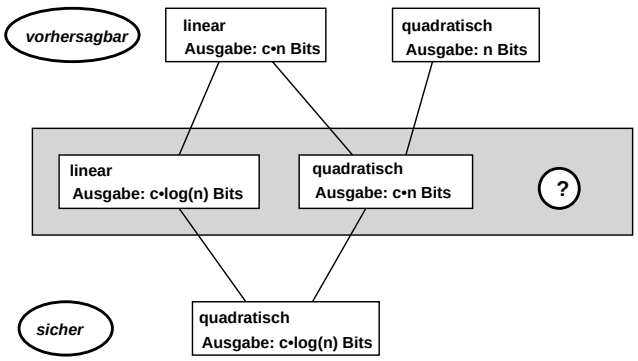
- Output-Transformation,

nichtlinear sein. In der Grauzone, wo keine nützlichen Aussagen über die Sicherheit bekannt sind, liegen unter anderem quadratische Kongruenzgeneratoren mit mäßig gestutztem Output.

Im Folgenden werden zwei Ansätze behandelt, zu sicheren Zufallsgeneratoren zu kommen:

- Kombination linearer Schieberegister mit nichtlinearer Output-Transformation,

- nichtlineare Kongruenzgeneratoren mit stark gestutztem Output.


vorhersagbar
linear
Ausgabe: c•n Bits
quadratisch
Ausgabe: n Bits
linear
Ausgabe: c•log(n) Bits
quadratisch
Ausgabe: c•n Bits
?
sicher
quadratisch
Ausgabe: c•log(n) Bits