“Pseudo Random” Generators within Cryptographic Algorithms

Mihir Bellare  
Shafi Goldwasser  
Daniele Micciancio
Randomness in Cryptographic Algorithms

Input → Crypto Algorithm → Output

Secret Storage

Secret Keys, Integral Part of Algorithm
Randomness in Crypto Example (1)

Randomized Encryption

- Probabilistic Encryption [GM82, AD97]
- \( RSA'(m) = RSA(m + \text{Random Pad}) \) [BR]

If randomness used for message is revealed, can decrypt message

- Randomness is necessary for Partial Information Security
Randomness in Crypto Example (2)

Randomized Signatures

• Signatures provably secure against chosen-message-attack [GMRi84,NY,DN]

Random Choices are Revealed

• Digital Signature Standard [ElGamal, DSA]

If random choices for 1 message are revealed then secret key is compromised
Source of Randomness

- Specialized Hardware exploiting Some Physical process
- User Input: Every time random number used, user is queried
- Pseudo Random Number Generators:
  Deterministic Programs that stretch a “truly random” seed into a longer sequence of “seemingly random” bits

\[
\begin{align*}
\text{seed} & \quad \rightarrow \quad \text{PSRG} & \quad \rightarrow \quad b_1 \ b_2 \ b_3 \ \ldots
\end{align*}
\]
Pseudo-Random Generation (PSRG)

• Cryptographically Strong PSRG:
  • Cannot predict $b_i$ from $b_1...b_{i-1}$, in polynomial time
    – If there exists one-way functions, they exist
    – computationally expensive often

• Cryptographically “Not Strong”’ PSRG:
  – Often pass many statistical tests
  – Often, efficient prediction algorithm exists, or can hardly analyze tests at all
  – Much faster than strong ones

  Thus, often used anyway and the security depends on the cryptographic application
Our Work

• PSRG usually studied as stand alone algorithms
  (Statistical properties, prediction properties).
• We combine study of cryptographic algorithm and
  the PSRG used

  We analyze the use of linear congruential generators
  (LCG) within the DSS signing algorithm and show
  that it is insecure for a wide class of LCG’s

[Ba] Proved Rabin-Miller primality testing with LCG
[Kr] Prove security of MAC’s using LFSR (any e-bias seq’s)
Results

• DSS is totally breakable when used with LCG
• DSS is totally breakable when used with truncated LCG or concatenated LCG, i.e. LCG’s for which for security reasons only a fraction of the bits are used
• In general: any PSRG that can be expressed by modular linear equations in insecure when used with DSS
Digital Signature Algorithm
(DSS, variant of EG)

• Let p,q primes |p| = 512, |q| = 160,
g generator of subgroup Zp* of order q

• To sign m in Zq:
  
  Hash m
  Pick at random “nonce” k.
  Compute \( r = (g^k \mod p) \mod q^* \)
  \( s = (xr+m) k^{-1} \mod q \)

  Output (r,s)

• To verify (r,s,m):
  
  Check that \( r = (y^{r/s}g^{m/s} \mod p) \mod q \)

* Signing is cheaper than Verifying, if choices of nonces is done offline which is recommended.
Security Properties and Features

– If discrete log is easy then totally breakable
– Existential forgery with key-only attack,
  Universal forgery with chosen signature attack
– If for a single message the nonce $k$ is found, then the secret key $x$ can be found
  • where do you store the off-line generated nonces for on-line usage?
  • how do you ensure they are never reused?
  • how do you generate the nonces: DSS calls for Random or Pseudo Random
Linear Congruential Generators

\[ k_0 \text{ truly random seed} \]
\[ k_{i+1} = a \cdot k_i + b \mod M \]

(where \( a, b, M \) define the generator)

Predictable !!!

Even if \( a, b, M \) unknown [Pl]
Even if truncated [FHLK]

However, predictability insecurity within any crypto application as the pseudo random sequence of \( k_i \)'s can be hidden

( in particular: can't use prediction algorithms)
The attack on simplest LCG

• Get the signatures of two consecutive messages
  \((r_1,s_1) = DSA(x, m_1, k_1)\)
  \((r_2,s_2) = DSA(x, m_2, k_2)\)

Take signature equations
\[
\begin{align*}
k_1 s_1 - x r_1 &= m_1 \pmod{q} \\
k_2 s_2 - x r_2 &= m_2 \pmod{q} \\
k_2 &= a k_1 + b \pmod{M}
\end{align*}
\]

IGNORE all relations \(r = g^k \pmod{p}\)

Need to Address:
1. need to solve linear system in different moduli
2. assuming found solution, may not be meaningful, i.e. not right \(x, k\).

1. Lattice Reduction
2. Uniqueness Lemma
The Uniqueness Lemma

Let G be any pseudo-random number generator and M a bound on the number of seeds.

Note: G is not necessarily a LCG.

Lemma:

Fix secret key x of DSS, and seed k of the generator G. Pick at random \( m_1 \ldots m_n \) messages from \( \mathbb{Z}_q \). Let \((r_i,s_i)\) be DSS signatures for \( m_i \) with the nonces computed by G on seed k. Then

\[
\text{Prob}_m[\text{there exists } (x',k') \text{ s.t. } x' \not\equiv x \text{ and yet } (r_i,s_i) \text{ are legal signatures of } m_i \text{ with the nonces computed by G on } k'] < M/(q^n)
\]

Recall: that in DSS messages are hashed before usage...
Lattice Reduction

- **Def:** Let $B = (b_1,\ldots,b_n)$ be a set of vectors in $\mathbb{R}^n$. Let the lattice $L(B) =$ integer combinations of $B$.
- **Closest Lattice Vector Problem:** Given $B$ and vector $y$ in $\mathbb{R}^n$, find nearest vector $w$ to $y$ in $L(B)$.
- **Babai (CVP):** Polynomial time Approximation Algorithm which on input $B$ and $y$ finds $w$ in $L(B)$ such that $\|w-y\| < f \min_v \|v-y\|$ where $f = 2^{n/2}$.
Solving the Equations

• We want to solve modular equations
  \[ k_1s_1 - xr_1 = m_1 \mod q \]
  \[ k_2s_2 - xr_2 = m_2 \mod q \]
  \[ k_2 = ak_1 + b \mod M \]

• We set up a lattice \( L(B) \) and a vector \( y \) such that any lattice vector in \( L(B) \) close to \( y \) yields a solution \( (x, k_1, k_2) \) to the modular equations
The Lattice

- Consider the lattice generated by the columns of

\[
B = \begin{pmatrix}
-r_1 & s_1 & 0 & q & 0 & 0 \\
-r_2 & 0 & s_2 & 0 & q & 0 \\
0 & -a & 1 & 0 & 0 & M \\
1/gx & 0 & 0 & 0 & 0 & 0 \\
0 & 1/gk_1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/gk_2 & 0 & 0 & 0
\end{pmatrix}
\]

where \( gx = M/2, gk_1 = gk_2 = q/2 \) are guesses for \( x, k_1, k_2 \)

- Notice that \( L(B) \) contains the vector

\[
X = (m_1, m_2, b, x/gx, k_1/gk_1, k_2/gk_2)^T
\]
The Target Vector

- Apply Babai’s algorithm to lattice B and vector
  \[ Y = (m_1, m_2, b, gx/(q-gx), \frac{gk_1}{M-gk_1}, \frac{gk_2}{M-gk_2}) \]
  and hope to find lattice vector
  \[ X = (m_1, m_2, b, \frac{x}{gx}, \frac{k_1}{gk_1}, \frac{k_2}{gk_2})^T \]
  so can recover \( x \).

- If guesses were close then will, else try all guesses in a set which is \( \text{poly}(\log q, \log M) \), WHP over messages will find \( x' \) that solves the linear system and within two trials \( x' = x \).
In general: Solving Simultaneous Modular Equations

• Previous Technique generalized to work on arbitrary linear equations in different moduli. *Where do these arise in Crypto?*

• Truncated LCG in DSS can be set up as a system of equations in different moduli,

• Thus using truncated LCG in DSS is totally breakable.
Other PSRG in DSS?

- Linear Congruential Generators with truncation
- Linear Congruential Generators with Concatenation
- Any combination of the above (see unix pseudo random number generation of 32 bits)
Open Problems

• LCG with unknown parameters $a, b, M$
  – Still predictable

• Generators defined by non-linear modular recurrences
  – Sequences produced by polynomial modular recurrence relations are also predictable

• Can our attack be extended to these more general PRNG?