Lecture 21

Secure random number generation

Nicola Laurenti December 9, 2020

Lecture 21— Contents

The need for random numbers in security

Good random sources are a most valuable resource in security. They are needed to provide

- key generation
- secrets for key agreement
- nonces in interactive protocols
- probabilistic mechanisms
- randomized algorithms
- ...

Security of real world RNGs

The ideal counterpart of a real world RNG is (rather obviously) an ideal source of independent and uniform z_n

Secure pseudo RNG

Pseudo RNGs only have access to a finite entropy source (the seed) and aim to be computationallly indistiguishable from the ideal random source
Typically, uniformity is easy; independence is impossible

True RNG

True RNGs have access to a low entropy rate source of infinite randomness, some also have access to a finite entropy source (the seed) and aim to be unconditionally indistiguishable from the ideal random source
Typically, independence and uniformity can be obtained at the price of a low rate

Many practical tests exist for checking secure randomness (e.g., the NIST test suite), mainly designed for pseudo RNGs.

The (insecure) linear congruential pseudo RNG

Let N be a large integer, M a small integer, and $a, c \in \mathbb{Z}_N$ such that $\gcd(a, N) = 1$
Then, a popular RNG can be constructed as follows

state $s_n \in \mathcal{S} = \mathbb{Z}_N$

output $z_n \in \mathcal{Z} = \mathbb{Z}_M$

seed the initial state $v = s_0$

equations

$$\begin{cases} s_{n+1} = (a s_n + c) \bmod N & \text{update} \\ z_n = s_n \bmod M & \text{output} \end{cases}$$

Typically, $M = 2$ is chosen, so z_n is a single bit
This RNG is not very secure, as each z_n leaks information about the values of a and c.
It's ok for running your own (unbiased) simulations, not for security

Blum-Blum-Shub RNG

Let p, q be two primes such that $p = q = 3 \pmod 4$, and let $N = pq$.
Then, a secure RNG can be constructed as follows

state $s_n \in \mathcal{S} = \mathbb{Z}_N$

output $z_n \in \mathcal{Z} = \{0, 1\}$

seed the initial state $v = s_0$

equations

$$\begin{cases} s_{n+1} = s_n^2 \bmod N & \text{update} \\ z_n = s_n \bmod 2 & \text{output} \end{cases}$$

RSA based RNG

Let p, q be two primes, and let $N = pq$, and $\varphi = (p-1)(q-1)$. Choose any $e \in \mathbb{Z}_N$ such that $\gcd(e, \varphi) = 1$

Then, a secure RNG can be constructed as follows

state $s_n \in \mathcal{S} = \mathbb{Z}_N$

output $z_n \in \mathcal{Z} = \{0, 1\}$

seed the initial state $v = s_0$

equations

$$\begin{cases} s_{n+1} = s_n^e \bmod N & \text{update} \\ z_n = s_n \bmod 2 & \text{output} \end{cases}$$

RNGs based on hash functions

Consider any secure cryptographic hash function $h : \mathcal{M} \mapsto \mathcal{T}$, with $\mathcal{M} = \mathbb{Z}_N$.
Then, a secure RNG can be constructed as follows

- **state** $s_n \in \mathcal{S} = \mathbb{Z}_N$
- **output** $z_n \in \mathcal{T}$
- **seed** the initial state $v = s_0$
- **equations**

$$\begin{cases} s_{n+1} = s_n + 1 & \text{update} \\ z_n = h(s_n) & \text{output} \end{cases}$$

RNGs based on symmetric encryption

Consider any secure symmetric encryption mechanism $E : \mathcal{K} \times \mathcal{M} \mapsto \mathcal{X}$.
Then, a secure RNG can be constructed as follows

state $s_n \in \mathcal{S} = \mathcal{M} = \mathbb{Z}_N$

output $z_n \in \mathcal{Z} = \mathcal{X}$

seed the key and initial state $v = (k, s_0)$

equations

$$\begin{cases} s_{n+1} = s_n + 1 & \text{update} \\ z_n = E(k, s_n) & \text{output} \end{cases}$$

RNGs based on HMAC

Consider the HMAC scheme tag computation function $T : \mathcal{K} \times \mathcal{T} \mapsto \mathcal{T}$ where $\mathcal{T} = \mathcal{A}^\ell$, $\mathcal{K} = \mathcal{A}^\Delta$, that makes use of a cryptographic hash function $h : \mathcal{A}^{\ell+\Delta} \mapsto \mathcal{A}^\ell$, and recall its definition as

$$\text{HMAC} : t = h([k + \beta_2, h[k + \beta_1, u]\,])$$

with $beta_1$ ans β_2 the input and output pad constants, respectively Then, a secure RNG can be constructed as follows

state $s_n \in \mathcal{S} = \mathcal{T}$

output $z_n \in \mathcal{Z} = \mathcal{T}$

seed the key and initial state $v = (k, s_0)$

equations

$$\begin{cases} s_{n+1} = T(k, s_n) & \text{update} \\ z_n = s_n & \text{output} \end{cases}$$

Dual elliptic curve deterministic random bit generator

Consider an elliptic curve $\mathcal{E}$ on a finite field $\mathbb{F} = \mathbb{Z}_p$ with p prime
denote by $\circ$ the point operation on $\mathcal{E}$
denote by $c_1(P) \in \mathbb{F}$ the (integer) first coordinate of a point P

Let $s_n \in \mathbb{F}$ and $z_n \in \mathcal{Z} = \{0, \ldots, 2^r - 1\}$ be the state and the r-bit output of the RNG

Starting from two specific points $P, Q \in \mathcal{E}$ the update and output equations are defined via the auxiliary variable y_n as

$$\begin{cases} y_n = c_1(P \overset{s_n}{\circ}) \\ s_{n+1} = c_1(P \overset{y_n}{\circ}) \\ z_n = c_1(Q \overset{y_n}{\circ}) \bmod 2^r \end{cases}$$

Dual EC DRBG attack (1/2)

Suppose that the attacker knows $q \in \mathbb{Z}_p$ such that $Q \overset{q}{\circ} = P$
and that $2^r \geq p$, i.e. no bits are discarded from z_n.
Then, the attacker can

1. observe z_n and find the corresponding point $R = (z_n, \cdot) \in \mathcal{E}$.
   Then, it must be $R = Q \overset{y_n}{\circ}$
2. compute $S = R \overset{q}{\circ}$.
   Observe that $S = Q \overset{qy_n}{\circ} = P \overset{y_n}{\circ}$
3. extract $s_{n+1} = c_1(S)$.
   Now the attacker knows the next state of the PRNG and can predict all outputs z_m, $\forall m > n$

Dual EC DRBG attack (2/2)

Now, relax the assumption that no bits are discarded, and let $I = \lceil p/2^r \rceil$. Then, $\forall i = 0, \ldots, I-1$

- let $v_i = i \cdot 2^r + z_n$, and find $R_i = (v_i, \cdot) \in \mathcal{E}$
- repeat steps 1–3 above, extracting a guess $\hat{s}_{n+1,i}$
- compute the corresponding output value $\hat{z}_{n+1,i}$
- observe the actual output z_{n+1} and select the value of i for which $\hat{z}_{n+1,i} = z_{n+1}$. The corresponding $\hat{s}_{n+1,i}$ is the PRNG state s_{n+1}

The attack is still effective and efficient, provided the number of guesses I is not too large (i.e., few bits are discarded).

The assumption that the attacker knows q is necessary, and it is not feasible to compute q from P, Q (finite log).

However, the implementer who sets P and Q may choose Q and q and compute P

The history of Dual EC DRBG

2002-03 NSA urges NIST to include Dual EC DRBG in the future standards for secure RNG, providing explicit values for $\mathcal{E}$, P and Q and $I = 2^{16}$. Did they know q and compute $P = Q^{q}_{\circ}$?

2004 RSA makes Dual EC DRBG the default PRNG in their product BSAFE

2005 NIST standardizes Dual EC DRBG in *SP 800-90A*. The standard allows users to choose their own P adn Q, but only implementations with the suggested P and Q from NSA can get FIPS validation

2006-07 Several cryptographers and researchers point out the possible attack, observe that I is too small, and wonder if NSA inserted a backdoor into the standard on purpose

2013 NSA documents leaked by Edward Snowden describe a program aimed "to covertly introduce weaknesses into the encryption standards" used worldwide.

2013 RSA recommends its customers to stop using the Dual EC DRBG

2014 NIST removes Dual EC DRBG from the new version of the standard

True random sources

Sources for true randomness must rely on

- natural random phenomena, such as thermal currents in resistors, flickering in light sensors
- human activity, such as timing between keystrokes
- quantum measurements

Typically, the random processes describing these phenomena have memory (correlation, decreasing with time separation) and nonuniform distribution (but typically symmetric)

Unconditional security

The unconditional security measure for a true RNG that outputs a block $\boldsymbol{z} = (z_1, \ldots, z_N)$, is the variational distance between the actual and ideal output distribution

$$d_{\mathsf{V}}(p_{\boldsymbol{z}}, p_{\boldsymbol{z}^\star}) \leq \sqrt{\frac{1}{2} \mathrm{D}\left(p_{\boldsymbol{z}} \| p_{\boldsymbol{z}^\star}\right)} \quad , \qquad \text{by Pinsker inequality}$$

Observe that, since $\boldsymbol{z}^\star \sim \mathcal{U}(\mathcal{Z}^N)$

$$\mathrm{D}\left(p_{\boldsymbol{z}} \| p_{\boldsymbol{z}^\star}\right) = N \log_2 |\mathcal{Z}| - H(\boldsymbol{z}) = N(\underbrace{\log_2 |\mathcal{Z}| - H(z)}_{\text{nonuniformity}}) + \underbrace{N H(z) - H(\boldsymbol{z})}_{\text{dependence}}$$

Deterministic extractors

Deterministic extractors are trasformations mapping long messages with low information efficiency to shorter messages with higher efficiency

$$\xrightarrow[T_x]{x_\ell} \boxed{\mathsf{Ext}(\cdot)} \xrightarrow[T_z]{z_n}$$

Owing to the deterministic mapping it must be

$$\frac{1}{T_z}\log_2 M_z = \frac{H(z)}{T_z} = \frac{H_{\mathsf{s}}(z)}{T_z} = R_z \leq R_x = \frac{H_{\mathsf{s}}(x)}{T_x}$$

An optimal source encoder is a good determinstic randomness extractor.

Designing determinsitic extractors requires knowledge of p_x. Otherwise, if p_x is only partially known, we must resort to seeded extractors

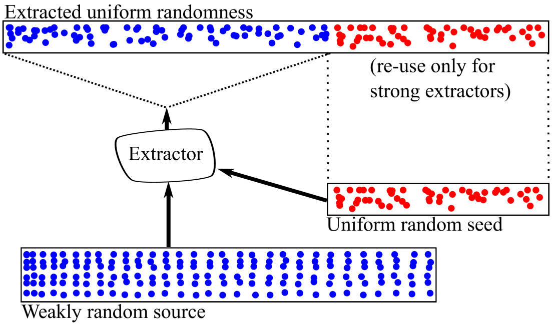
Seeded extractors

Definition

A seeded extractor is a function $f : \mathcal{X}^N \times \mathcal{V} \mapsto \mathcal{Z}^M$ such that $M \log_2 |\mathcal{Z}| \simeq H_{\min}(\boldsymbol{x})$, and if $v \sim \mathcal{U}(\mathcal{V})$, then $d_{\mathsf{V}}(p_{\boldsymbol{z}}, p_{\boldsymbol{z}^\star}) \ll 1$

Definition

A seeded extractor is said to be strong if $\boldsymbol{z}$ is independent of v



Universal hashing

A seeded extractor can be obtained from an ε-almost strongly universal$_2$ family of hash functions $T_k : \mathcal{X}^N \mapsto \mathcal{Z}^M$ where the seed is the key k

Proposition (Leftover hashing lemma)

If a strongly universal$_2$ family of hash functions is used with a uniform seed, then

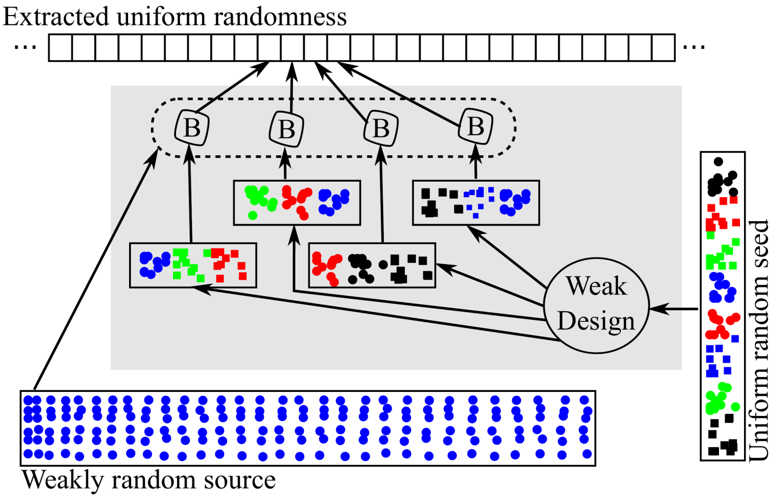
$$d_{\mathsf{V}}(p_{\boldsymbol{z}}, p_{\boldsymbol{z}^\star}) \leq \frac{1}{2}\sqrt{|\mathcal{Z}|^M / 2^{H_2(\boldsymbol{x})}} = 1/2^{(H_2(\boldsymbol{x}) - M \log_2 |\mathcal{Z}|)/2+1}$$

where $H_2(\boldsymbol{x}) = \log_{1/2} \sum_{\boldsymbol{a}} p_{\boldsymbol{x}}(\boldsymbol{a})^2$ *is the collision entropy of the input and* $M \log_2 |\mathcal{Z}|$ *is the output nominal information (number of output bits)*
If an ε*-almost strongly universal$_2$ family of hash functions is used, then*

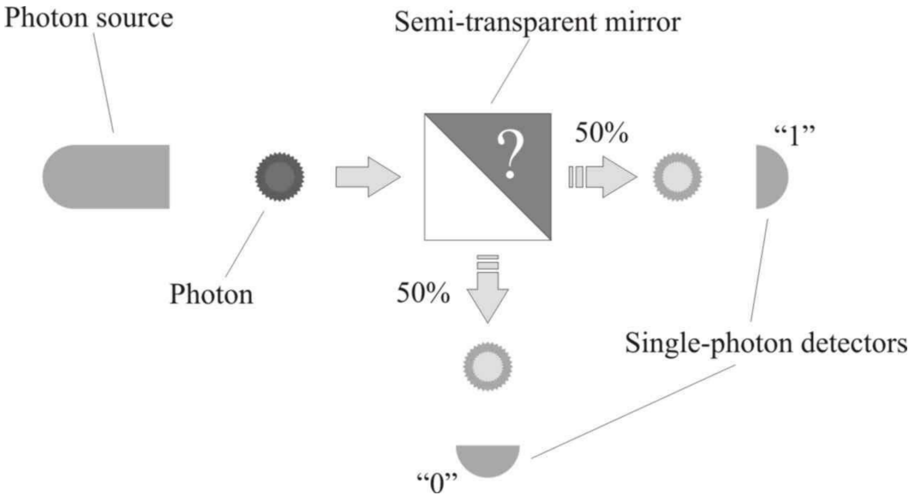
$$d_{\mathsf{V}}(p_{\boldsymbol{z}}, p_{\boldsymbol{z}^\star}) \leq \frac{1}{2}\sqrt{|\mathcal{Z}|^M}\sqrt{\varepsilon - 1/|\mathcal{Z}|^M + 1/2^{H_2(\boldsymbol{x})}}$$

Trevisan's extractor

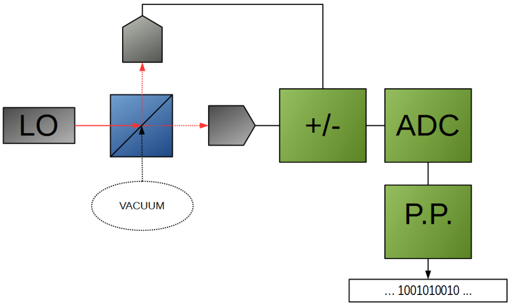
- binary extractor
- each output bit obtained by combining a different subset of t seed bits
- subsets have minimum overlap

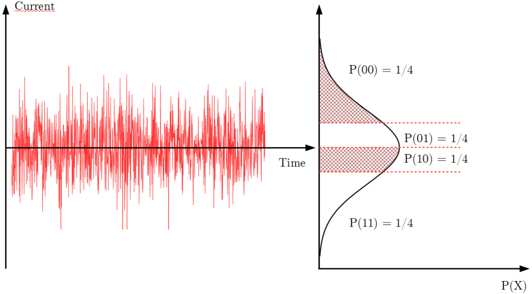


Discrete variable Quantum sources



Continuous variable Quantum sources





Summary

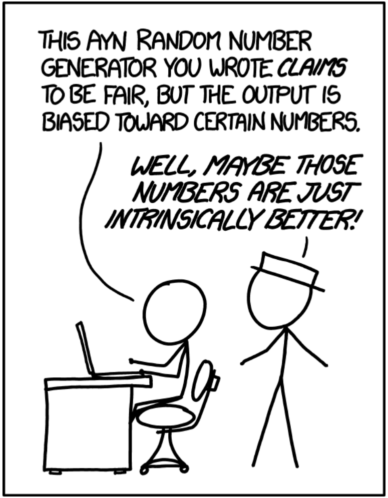
In this lecture we have:

- introduced the problem of secure randomness generation, distinguishing between pseudo- and true RNG
- presented several examples of pseudo-RNG
- introduced the unconditional security metric for true-RNG
- described two classes of randomness extractors
- presented the principles behind quantum RNGs

Assignment

- class notes
- textbook, §B.1–B.3

End of lecture



Ayn Random, reproduced from xkcd URL: xkcd.com/1277