PSEUDO-RANDOMNESS AND STREAM CIPHERS

BURTON ROSENBERG
UNIVERSITY OF MIAMI

Contents

1. Pseudorandomness

Shannon's Perfect Secrecy definition is equivalent to a model of two interacting Turing Machines. One is a protocol machine that offers and judges challenges, and the other an adversary that responds to the challenges, attempting to win the game. The equivalent model permits a limited sort of attack called the eaves dropping attack, but permits unlimited computational power to the adversary. The result is an absolutely guarantee of secrecy at the expensive of practicality. An example of an encryptions achieving perfect secrecy is the Vernam Cipher.

Figure 1 diagrams the interaction between the protocol Π and the adversary $\mathcal{A}$. The rounds of communication occur in order from top to bottom.

(1) Π sends $\mathcal{A}$ the value n in unary.
(2) $\mathcal{A}$ responds with a pair of equal length messages, m_0 and m_1.
(3) Π then chooses one message at random and encrypts it with a randomly chosen key, and sends the result c to $\mathcal{A}$.
(4) $\mathcal{A}$ returns a bit $\tilde{b}$, hoping that it matches the message choice made by Π.

Date: September 13, 2021.

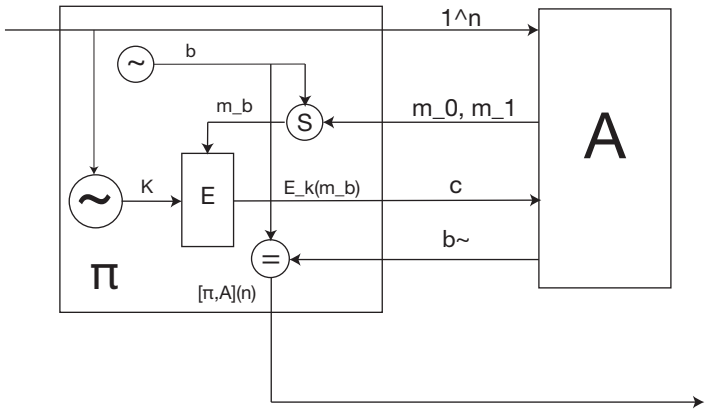

FIGURE 1. The Adversarial Indistinguishability Game.

The outcome of the interaction is a boolean valued random variable denoted $[\Pi, \mathcal{A}](n)$,

$$\begin{array}{rccc} [\,\Pi, \mathcal{A}\,](n): & K \times \Omega & \longrightarrow & \{\,T, F\,\} \\ & (k, \omega) & \mapsto & (\tilde{b} == b) \end{array}$$

where $K \in U_n$ is an n bit key chosen uniformly at random, and Ω includes all other randomness used by the PPT's as well as the bit b.

The claim of Shannon Perfect Secrecy is equivalent to the expected no-advantage to the adversary,

$$P(\{\,(k, \omega) \,|\, k \in K, \omega \in \Omega, b = \tilde{b}\,\}) = 1/2$$

when k and b are chosen uniformly, independently at random.

1.1. **PPT bound adversaries.** The restrictions to achieve perfect secrecy make it impractical. The more practical outcome results by proposing only polynomial-time bounded, probabilistic Turing Machine (PPT) as adversaries. Machines of this sort are generally considered the proper notion of *effective computation*, computations that are not only possible theoretically, but use a practical amount of resource and produce applicable results.

Having introduced an *asymptotic* complexity notion, in which run time is bounded by some notion of the problem size n, we must follow through and give all elements of the problem size bounds. The n is often called the *security parameter*. This can often be thought of as the key length. By increasing n, we out-run the resources of the attacker. For instance, no matter what our encryption, if the key is only 8 bits, with only 256 possibilities, we do not have security in the practical sense of the word. But if n is a parameter, then in practice, we can increase n until all likely opponents will be exhausted by an inefficient search for the key.

Therefore we need to bound the message sizes, and we do so by stating that for each adversary $\mathcal{A}$ it states a polynomial $l(n)$, and that,

$$l(n) = |\, m_0 \,| = |\, m_1 \,|$$

This is hardly a restriction because the PPT $\mathcal{A}$ comes with a polynomial run-time bound, which is of course a bound on the length of anything the adversary can produce. If nothing else, $l(n)$ can be that run-time bound.

Definition 1.1. (Adversarial indistinguishablity) An encryption is adversarial indistinguishable if for any PPT adversary $\mathcal{A}$, its advantage is asymptotically negligible beyond a no–information guess of b,

$$P([\,\Pi, \mathcal{A}\,](n)) \leq 1/2 + \mathit{negl}(n)$$

1.2. **Pseudorandom Generator.** A Pseudorandom generator is a deterministic algorithm which outputs a stream of bits that cannot be distinguished by any PPT adversary from true randomness. The algorithm is deterministic, but its input is a n bit seed, chosen uniformly from U_n. If the stream is $l > n$ bits long, the 2^n possible sequences generated are a tiny fraction of the much larger 2^l sequences that result from l random bit choices.

For pseudorandomness, sampling form this small subset passes all polynomial tests with only indistinguishably different likelihood as sampling from the full space of sequences.

1.3. **Formal definition of pseudorandomness.** A formal definition of pseudorandmoness is illustrated in Figure 2. We suppose a PPT algorithm $\mathcal{D}$, called the *distinguisher*, which is given either a sample of the generator's output, or a true random string. It then judges the string as true or false. Essentially, the measure of a property of some sort. The output can be randomized, so the result is not exactly a classification predicate, such as "element of", which tend to be common functions from input to output. For each string presented we might have a probability distribution over true/false.

Note also the parameter n, and the polynomial $l(n)$ which specifies an required expansion amount for the pseudorandom generator.

Definition 1.2 (Bit generator)**.** A bit generator $G_n^l((s)$ is a deterministic polynomial time program taking an n bit input s, called the *seed* at producing the first l bits of an infinite string $G_n(s)$,

$$\begin{array}{rccc} G_n^l: & U^n & \longrightarrow & U^l \\ & s & \mapsto & G_n(s)|^l \end{array}$$

The polynomial is in inputs n and l.

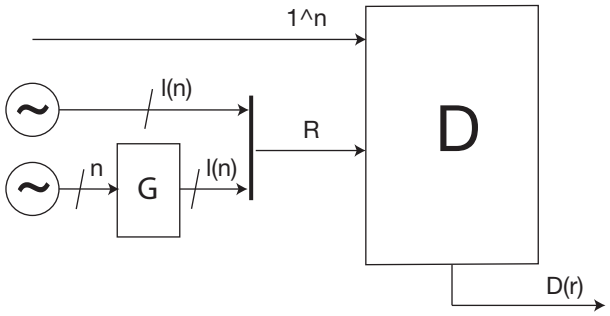

FIGURE 2. Distinguishing Pseudo-random Generators

Definition 1.3 (Indistinguishable)**.** A bit generator $G_i(n)$ is a pseudorandom generator if for all polynomials $l(n)$ and an PPT distinguishers $\mathcal{D}$,

$$\left| P(\mathcal{D}(G_n^{l(n)}(U_n))) - P(\mathcal{D}(U_{l(n)})) \right| \leq negl(n)$$

where the distinguisher receives either a uniform random string $\tilde{u} \in U_{l(n)}$, or the output of $G_n^{l(n)}(u)$ on a uniform random string from $u \in U_n$, and the includes all randomness in $\mathcal{D}$.

2. Stream Ciphers

If a pseudorandom generator exists, the pseudo random stream can be as a pad such as was done in the Vernam cipher. The key to the cipher would be the n bit input to the generator, and an unlimited amount of padding would be available for messages. Ciphers made in this way, based on pseudorandom generators, are called *stream ciphers*.

The basic mechanism of a stream cipher and its relationship to the Vernam cipher is given in Figure 3. Given an l bit message and an n bit key, the l random bits needed are the output of the function G. If G is pseudorandom, while not perfectly secret, the encryption is secret against any polynomial-time bounded attack.

As shall be discussed below, this scheme shares with the Vernam certain imperfections as well. Notably, the key is one time use. In either case, since encryption is incremental, the random stream can be long enough, or the expansion great enough, that a sequence of messages can be encrypted, as long as a fresh region of the random

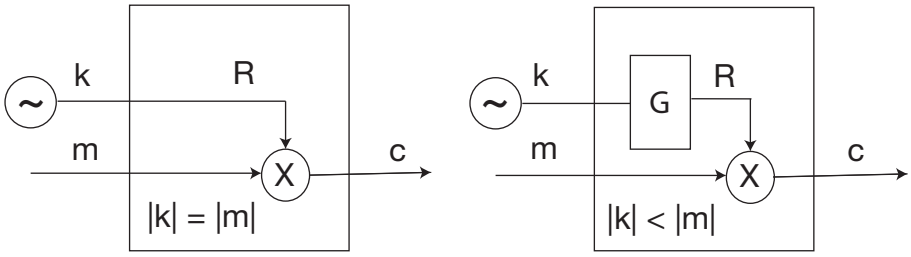

FIGURE 3. Ideal and practical Stream Ciphers

stream is used with each encryption. Synchronizing sender and receiver on which bits are are being used is a problem in itself.

The security of the encryption scheme is proven relative to the security of the generator. If P equals NP, there can be no pseudorandom generators. As we have not yet an answer for the P versus NP question, or results must be contingent, with conclusions depending upon assumptions. In this case, we will assume that pseudorandom generators exist and prove under that assumption that our stream cipher is secure.

The method is called a *reduction* because we reduce the breaking of a pseudorandom generator to the breaking of a stream cipher. We proposed a distinguisher $\mathcal{D}$ that uses an adversary $\mathcal{A}$ as a subcomponent. The transformation of the problem of generators to encryptions is such that an $\mathcal{A}$ that can distinguish messages yields a $\mathcal{D}$ that distinguishes probability distributions. Hence if no such $\mathcal{D}$ is assumed, not such $\mathcal{A}$ can exists.

See Figure 4 for the construction reducing an pseudorandom distinguisher to a encryption adversary.

2.1. **Proof of stream ciphers under the assumption of a generator.** First consider the diagram as an example of a $[\Pi, \mathcal{A}]$ adversarial indistinguishability game. The light shaded box is a composite of two possible encryption boxes. The heavy bar is meant to compress two diagrams into one. Either the top source is used, in which case we exclusive or m_b with $l(n)$ random bits; or the bottom source is used, and we exclusive or m_b with the output of the generator $G_n^{l(n)}$.

In either case, the output measures the ability of $\mathcal{A}$ to distinguish between m_0 and m_1.

Next consider the diagram as an example of an $\mathcal{D}$ distinguisher. The apparatus surrounded by the heavy dotted line is $\mathcal{D}$. Taken as a black box, fed with a sample of either $U_{l(n)}$ or $G_n^{l(n)}(U_n)$, the output measures the ability of $\mathcal{D}$ to distinguish the pseudorandom sequence from a uniformly generated sequence.

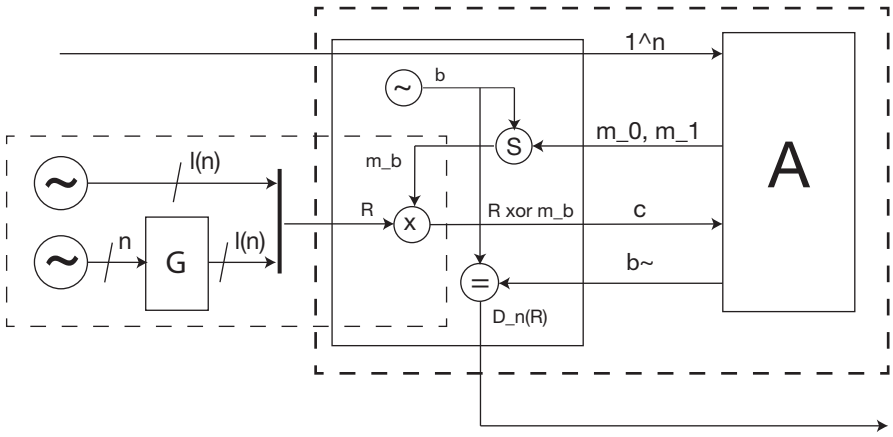

FIGURE 4. Distinguishing Reduced to Stream Cipher

Hence if the output is constrained by the assumption that G_n is a pseudorandom sequence, then $\mathcal{A}$ is constrained not to be able to distinguish between a perfect cipher and the stream cipher using G_n.

Theorem 2.1. In the stream cipher constructed in Figure 4, if G_n is a pseudorandom function then the stream cipher has adversarial indistinguishability.

Proof. For any adversary $\mathcal{A}$, choose an $l(n)$ to bound the message length, this could be the run time of $\mathcal{A}$.

If we choose the top of two random source of bits, the construction implements the Vernam cipher. For clarity, this will be protocol $\tilde{\Pi}$, and the interaction $[\tilde{\Pi}, \mathcal{A}]$. Since the Vernam cipher has perfect secrecy,

$$P(\,[\tilde{\Pi}, \mathcal{A}](n)\,) = 1/2.$$

Since this configuration is also a sample of $\mathcal{D}(U_{l(n)})$,

$$P(\,[\tilde{\Pi}, \mathcal{A}](n)\,) = P(\mathcal{D}(U_{l(n)})).$$

Likewise, this configuration when used with the source G_n shows that,

$$P([\Pi, \mathcal{A}](n)) = P(\mathcal{D}(G_n^{l(n)}(U_n))),$$

Assuming that G_n is a pseudorandom function,

$$\left|\, P(\mathcal{D}(G_n^{l(n)}(U_n))) - P(\mathcal{D}(U_{l(n)}))\,\right| \leq \mathit{negl}(n)$$

we have,

$$\begin{aligned}
\left| P(\mathcal{D}(G_n^{l(n)}(U_n))) - P(\mathcal{D}(U_{l(n)})) \right| &= \left| P(\mathcal{D}(G_n^{l(n)}(U_n))) - P(\,[\tilde{\Pi}, \mathcal{A}](n)\,)) \right| \\
&= \left| P(\mathcal{D}(G_n^{l(n)}(U_n))) - 1/2 \right| \\
&= \left| P(\,[\Pi, \mathcal{A}](n)\,) - 1/2 \right| \\
&\leq \mathit{negl}(n).
\end{aligned}$$

Hence,

$$P(\,[\Pi, \mathcal{A}](n)\,) \leq 1/2 + \mathit{negl}(n),$$

Therefore the stream cipher is adversarial indistinguishable. □

3. Practical Considerations

Stream ciphers are widely used for their speed and simplicity. However, as proposed they require two important cautions.

3.1. **Multiple messages.** If multiple messages are encrypted with the same key, the encryption schemes presented are completely vulnerable. As a model of a multiple-message protocol, suppose the adversary can present a list of messages,

$$\{\,(m_0^1, m_1^1), (m_0^2, m_1^2), \dots\}, \ |\,m_0^i\,| = |\,m_1^i\,| \text{ for all } i,$$

and the protocol responds, after choice of k and b with,

$$\{\,E_k(m_b^1), E_k(m_b^2), \dots\}$$

followed by the adversary's guess $\tilde{b}$ for b.

We could put a bound on the number of messages, however even if only two messages are allowed, the scheme will not have indistinguishability. The adversary chooses two messages m_0 and m_1 and requests,

$$\{\,(m_0, m_0), (m_0, m_1), \dots\}$$

and calculates,

$$\tilde{b} = \begin{cases} 0, \text{ if } c_0 == c_1 \\ 1, \text{ else} \end{cases}$$

This answer is always correct.

3.2. **Malleability.** It is also cautioned that this encryption is *malleable*. in our example, when the adverasary is given $c = M_k(m_b)$, although the adversary does not know b, it knows $M_k(m_{1-b})$,

$$\begin{aligned} M_k(m_{1-b}) &= r \oplus m_{1-b} \\ &= r \oplus (m_b \oplus m_b) \oplus m_{1-b} \\ &= M_k(m_b) \oplus (m_b \oplus m_{1-b}) \\ &= c \oplus (m_0 \oplus m_1) \end{aligned}$$

since the adversary knows m_0 and m_1.

Therefore learning c teaches other things. We do not know the use this extra knowledge can be put to, but we would rather not worry about that.

3.3. **Key diversity.** The practical need for encrypting multiple messages under a single key is pressing. Since a stream cipher proceeds incrementally on the next bit of message, it is possible to consider multiple messages as in their concatenation as one long message. However, the example of encrypting files shows the problem with this approach. Each file is encrypted without a notion of how much encryption as gone before. A more standard approach is a single key along with a randomly chosen diversifier that is made public, and combined with the secret master key, to create a per file key.

This is also called the *initial vector*. But the same idea is used in other contexts and called the *salt* and *key diversity*. One solution to the advice of separate keys for each application is to mix a publicly derivable tag from the application, perhaps its name, and salt the one master key diversifying in a systematic way the panoply of keys.