THE VERNAM CIPHER AND PERFECT SECRECY

BURTON ROSENBERG
UNIVERSITY OF MIAMI

Contents

1. What is the purpose of encryption

The most obvious goal of encryption is to keep secret the meaning of a message from an eavesdropper while communicating over a public channel that meaning to the intended counter-party.

There is always some sort of ceremony, preceding the use of the encryption on the communication channel, where a secret is created and shared. Say this occurs in a *secluded room*. Encryption systems bottle a moment of privacy to be reopened and refashioned later to create privacy over a public channel.

Encryption does not create privacy, it stores it and redeploys it, to suit the will of the participants and the needs of the situation.

Date: April 12, 2020.

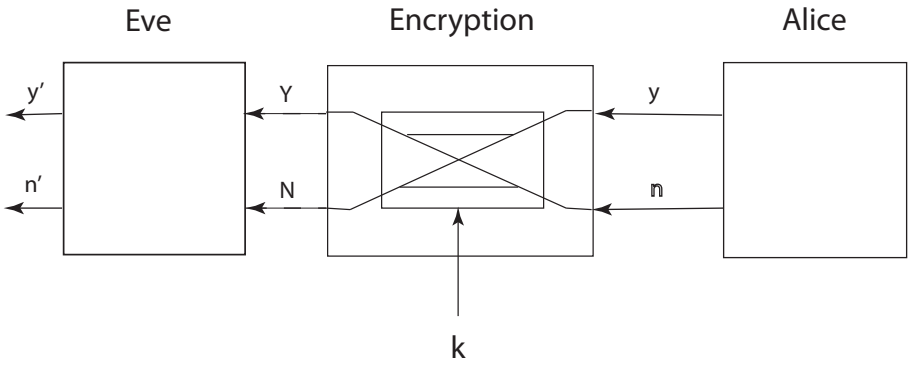

FIGURE 1. Vernam Cipher.

2. Perfect encryption

The model of communication is given in figure 1.

There is a message source, in the diagram called Alice, that selects a message from a message space randomly, with a known distribution. Alice sends symbols over the encrypted channel so to inform the intended receiver which message was selected.

In the diagram we see Eve, the eavesdropper, observing the symbols and trying to determine which message Alice has selected. Eve can know the message space, and the probability distribution of the messages, as well as the encryption machinery that created the symbols from the message. What Eve does not know is the secret key used in the encryption. In the digram, that key is shown as k.

It is possible to carry out this plan in such a way that Eve learns nothing by observing the activity in the channel. The best guess Eve can have is unchanged by observing activity in the channel. That is, guessing the most likely message according to the distribution.[1]

[1]Note that Eve does learn that a message has been sent. In our case a message is one bit, but in general Eve also learns approximately the size of the message. All of this is valuable information, but not to be discussed here.

Claude Shannon put together this model of encryption and communication, and defined perfect secrecy with the formula,

$$P(M) = P(M \,|\, C)$$

for any distribution on messages, $P(M)$. The probability that the eavesdropper believes a message $m \in M$ was sent, $P(M = m)$, is unchanged after observing the ciphertext $c \in C$ in the channel, $P(M = m \,|\, C = c)$. This needs to be true for all m and c, and for all distributions $P(M)$.

3. The Vernam cipher

The *Vernam Cipher* consumes one bit of stored privacy for every bit encrypted and sent over the public channel. It works with the very simple equation,

$$c = m \oplus k$$

Take the message bit and exclusive or it with the secret bit, and send the result. The decryption equation is,

$$c \oplus k = (m \oplus k) \oplus k = m \oplus (k \oplus k) = m \oplus 0 = m.$$

Theorem 3.1. The Vernam Cipher has Prefect Secrecy if the key k is chosen by an unbiased coin, perhaps observed by the communicating parties while in the secluded room.

Proof. The formula for conditional probability is,

$$P(M = m \,|\, C = c) = \frac{P(M = m \wedge C = c)}{P(C = c)}.$$

The case of $M = m \wedge C = c$ is the same as $M = m \wedge K = k$, since for any m and c, there is exactly one k that fits the encipherment equation. And since the coin flip was independent of the message,

$$P(M = m \,|\, C = c) = \frac{P(M = m \wedge K = k)}{P(C = c)} = \frac{P(M = m)P(K = k)}{P(C = c)}.$$

The calculation of $P(C = c)$ requires the *Law of Total Probability*,

$$\begin{aligned} P(C = c) &= P(C = c \,|\, M = 0)\, P(M = 0) + P(C = c \,|\, M = 1)\, P(M = 1) \\ &= P(K = c \wedge 0)\, P(M = 0) + P(K = c \wedge 1)\, P(M = 1) \\ &= P(K = k)\, (P(M = 0) + P(M = 1)) = 1/2 \end{aligned}$$

event	code	prob.
1 1	0	9/16
1 0	10	3/16
0 1	110	3/16
0 0	111	1/16

FIGURE 2. Encoding Events

Therefore,

$$P(M = m \,|\, C = c) = \frac{P(M = m)P(K = k)}{P(C = c)} = \frac{P(M = m)\,(1/2)}{1/2} = P(M = m).$$

QED □

4. Imperfect encryption

As an exercise to help understand perfect encryption, we can experiment with imperfect encryption. Suppose the secret key bit is not entirely random, but chosen as 0 with probability 1/4 and 1 with probability 3/4.

4.1. **The randomness of an imperfect coin flip.** Entropy is the number of bits that must sent in order to indicate a choice. Because of the heavy bias towards 1, it can almost be assumed that the choice is the value 1. In order to get things fully right, the attacker only needs to be sent "corrections" to this assumption, which occur with less density, so to speak, than the bit flow.

To demonstrate this, we will send two choices with less than two bits.

Call the pair of choices to be sent an *event*. Firgure 2 shows the bits sent through the channel in order to inform the receiver which event took place.

If as expected, the event was 1 1, the sender just sends a 0 to confirm that. Only one bit was sent to indicate two choices. However, the event is not always true, so corrections might need be sent. A correction message begins with a 1 and is followed by one or two bits more that fully describe correction. In the rare case the even was 0 0, the number of bits sent is actually more than the number of choices. But this does not happen often.

We calculate the average number of bits sent in order to indicate precisely the event,

$$\begin{aligned} E_c &= l_{00}p_{00} + l_{01}p_{01} + l_{10}p_{10} + l_{11}p_{11} \\ &= 3\left(p_{00} + p_{01}\right) + 2p_{10} + p_{11} \\ &= 3(1/16 + 3/16) + 2(3/16) + 9/16 \\ &= 1.6875 \end{aligned}$$

Or 0.84375 bits transmitted per choice.

We can also calculate the formula Shannon gives for the entropy for the toss of a coin toss with bias $p = 3/4$,

$$\begin{aligned} E_s &= -(p\log_2 p + (1-p)\log_2(1-p)) \\ &= -(3/4\log_2 3/4 + 1/4\log_2 1/4) \\ &= 0.81128 \text{ bits} \end{aligned}$$

The code described was created by a procedure called *Huffman coding*, and creates a code whose *bit rate* (number of "bits of symbol" transmitted per "bits of knowledge" communicated) almost equals the entropy.

To do: To follow up on this language of knowledge and symbol to make clearer. The knowledge in this case is the knowledge of the event, and the symbol is the code for that event.

4.2. **Message leakage in the ciphertext.** Because of the coin bias, the encryption generally just flips the message bit and sends it out. Therefore, what is usually correct is for the eavesdropper to just flip the ciphertext bit back again. Some 3 out of 4 message bits will be correct, in this way.[2]

However, the eavesdropper must both take into consideration the distribution $P(M)$ and the observed bit. Suppose the message 0 is much more likely and that the observed bit is 0. Either the more likely message 0 was selected and sent through the less likely key 0, or the less likely message 1 was selected and sent through the more likely key 1.

[2]With a perfect encryption, only 1 out of 2 message bits are correct by guessing either the bit is the same or flipped. And the eavesdropper could achieve the same result by having no interest in either what is seen in the channel or what is intended by the sender, and just flipping coins, writing down the sequence of coin flips.

We need to calculate,

$$\begin{aligned}
P(M=0\,|\,C=0) &= \frac{P(M=0 \wedge C=0)}{P(C=0)} \\
&= \frac{P(M=0)P(K=0)}{P(C=0\,|\,M=0)\,P(M=0)+P(C=0\,|\,M=1)\,P(M=1)} \\
&= \frac{P(M=0)(1/4)}{(1/4)\,P(M=0)+(3/4)P(M=1)} \\
&= \frac{(1/4)P(M=0)}{(1/4)\,P(M=0)+(3/4)(1-P(M=0)} \\
&= \frac{P(M=0)}{3-2\,P(M=0)}
\end{aligned}$$

$$\begin{aligned}
P(M=1\,|\,C=0) &= \frac{P(M=1 \wedge C=0)}{P(C=0)} \\
&= \frac{P(M=1)P(K=1)}{(1/4)\,P(M=0)+(3/4)(1-P(M=0)} \\
&= \frac{(1-P(M=0))(3/4)}{(3/4)-(1/2)(1-P(M=0)} \\
&= \frac{1-P(M=0)}{1-2/3\,P(M=0)}
\end{aligned}$$

I graph these as a function of $P(M=0)$ in graph 3. These are the individual graphs of a the probability that the message is 1 (red line) or zero (blue line) given the observed bit is 0. At the left edge, where message 0 is unlikely, and the observed bit 0 makes the message 1 likely, the probability that the message is a 1 is higher than it is a 0,

$$P(M=1\,|\,C=0) > P(M=0\,|\,C=0) \text{ for } P(M=0) < 3/4.$$

To the right, as it becomes very likely that the message is 0, it is better to assume that the key was 0, even though that is only true 1/4 of the time,

$$P(M=1\,|\,C=0) < P(M=0\,|\,C=0) \text{ for } P(M=0) > 3/4.$$

And at the crossover between these two guessing strategies,

$$P(M=1\,|\,C=0) = P(M=0\,|\,C=0) = 1/2 \text{ for } P(M=0) = 3/4.$$

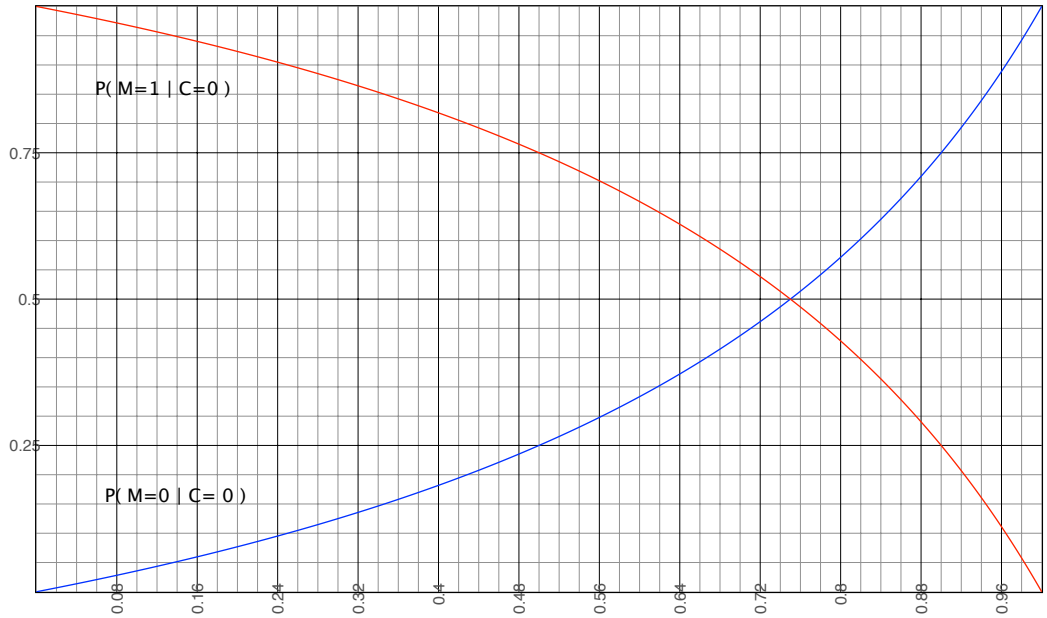

FIGURE 3. Likelihood graphs

5. Odds Ratio and Bias

I very nice way to think of this *greatest likelihood decision* is to think in terms of *odds*. The odds of an event is the ratio between the event and not the event. So for a biased coin introduced in the previous section,

$$P(K = 1) : P(K = 0) = (3/4) : (1/4) = 3 : 1$$

or "3 to 1 odds".

We can manipulate our probability equations into an a format that intuitively says: I have the a priori odds, (before seeing the ciphertext), now I see the ciphertext, what are now the odds. And it turns out that this format as a simple structure with where the old and new odds are connected by a multiplier.

To do this, recall *Bayes' Law*,

$$P(M = m \,|\, C = c) = \frac{P(C = c \,|\, M = m)P(M = m)}{P(C = c)}$$

and apply it to the case of learning $C = 0$,

$$\begin{aligned}\frac{P(M=1\,|\,C=0)}{P(M=0\,|\,C=0)} &= \frac{P(C=0\,|\,M=1)P(M=1)}{P(C=0)}\frac{P(C=0)}{P(C=0\,|\,M=0)P(M=0)} \\ &= \frac{P(C=0\,|\,M=1)}{P(C=0\,|\,M=0)}\,\frac{P(M=1)}{P(M=0)} \\ &= \frac{P(K=1)}{P(K=0)}\,\frac{P(M=1)}{P(M=0)}\end{aligned}$$

Now we can see a bit clearer what is going on. If we want to know what should be the a priori odds on messages, so that, given 3:1 odds that the encryption flips the message, bit, so that the new odds on the message appear to be 1:1.

$$\begin{aligned}\frac{P(M=0)}{P(M=1)} &= \frac{P(K=1)}{P(K=0)}\,\frac{P(M=0\,|\,C=0)}{P(M=1\,|\,C=0)} \\ &= \frac{3}{1}\,\frac{1}{1}\end{aligned}$$

and that is when $P(M = 0) = 3/4$, confirming the crossover point in graph 3.

6. Longer messages and pseudo-randomness

With the exception of a brief investigation of the entropy of a biased coin, we have discussed one bit messages. The communicating partners meet in the the secluded room, observe a coin flip, and now have the right and ability to communicate one bit secretly over a public channel.

One bit, and only one bit.

If they shared k bits then they can securely transmit k bits. The vernam cipher could be used one bit at a time, using the bits in an order agreed upon either in secret or by public declaration. The k bit message would be perfectly encrypted. However, this requirement of a massive amount of random bits, and the dependence on the secluded room, makes this an impractical solution.

A cryptographically pseudo-random generator (PRG) is a deterministic function that can produce a stream of bits that "seem" random; and the meaning of "seeming random" is that no polynomial time algorithm can discriminate between this stream and a stream of truly random bits. It is not random, but no practical computing

algorithm can make use of this defect. An encryption then picks a PRG using the shared secret key and uses the pseudo-random stream of bits as if they were random, to encrypt the stream of message bits.

Our Radius protocol does something similar. It uses a cryptographically strong hash function in place of a PRG. It samples the hash function on inputs that are a combination of the shared secret and a public random *nonce*,[3] called in the protocol the *request authenticator*, to produce 128 pseudo-random bits per hash. These bits are used to encrypt the password.

The encryption, if not theoretically perfect, is perfect if the adversary is powerful only up to polynomial time computations.

7. Concluding Remarks

The consideration of a simple encryption for the Radius protocol motivated a presentation of modern cryptography and its use to achieve practical security. Shannon's definitions of entropy and perfect secrecy were introduced. A perfect cipher was described — then Vernam, also known as the One Time Pad (OTP).

We discussed the realities of encryption, in its limitation on how the all import shared secret is created and shared. To make the secret usable beyond its own bit length, the pseudo-random generator (PRG) was presented. The discussion was then summarized by indicating how the Radius protocol used a PRG formed from the MD5 hash function, the (short) shared secret and the fresh 128 bits of entropy of the access request, to generate the random key to be used in a Vernam cipher.

Certain concepts were left for future consideration. Interested readers can look at the notion of *Adversarial Indistingishability*, which is taught at length in my cryptography course, as a next step in understanding.

The crucial question is whether or not pseudo-randomness exists. It is important to emphasize that there is no claim that an "ad hoc" hash construction, such as MD5, or SHA-2, is pseudo-random. PRG's do exist *relative* to the assumption that other problems are in the gap between P and NP, such as Discrete Logs or Factorization. However, the hardness of those number theory problems are conjectured, not established.

[3]A nonce is a random number but selected for each use but never repeated.

And certainly if P is collapses to NP then no pseudo-randomness as described cannot exist.

The number theoretic problems which are the basis of sample PRG constructions, are not generally NP-complete. So they can be shown to be inadequate basis for pseudo-randomness without the theory of pseudo-randomness collapsing. And in recent years, some of these number theoretic problems have been solved on quantum computers.