Explanation of the Hancke-Kuhn Distance Bounding Protocol

by Steven Baltakatei Sandoval

Updated on 2021-03-04T01:36Z.

1 Introduction

It is possible to determine how far two computers are from eachother using the speed of light and ping time. The physical distance is, at most, the ping time multiplied by the speed of light. This documents explains the Hancke-Kuhn protocol that can calculate this upper bound for the distance between a Verifier **V** and a Prover **P** through the sending and receiving of certain bit sequences. This calculation is useful for defending against man-in-the middle attacks.

I have written this explanation in order to help solidify my own understanding of the protocol before I write my own implementation of it at my GitLab repository. It is an explanation in my own words. Any errors or misrepresentations are entirely my own.

A more detailed summary with references to academic papers was published by Cristina Onete which may be found here on her website's publication page.

2 Background

2.1 Explanation, part 1: setup

In 2005, Gerhard P. Hancke and Markus G. Kuhn proposed a distance-bounding protocol as a defense against man-in-the-middle attacks for people who use RFID tokens in order to automatically authenticate themselves for a location-based service such as the opening of a door or purchase at a specific point-of-sale device.

An example of a man-in-the-middle attack for such a building access-control could be two attackers maliciously forwarding radio traffic between an RFID token and a building RFID reader without the RFID token owner's knowledge even in the case where the token is located at a great distance from the reader. The idea to strengthen an RFID token against such an attack is to equip the building RFID reader with some means of proving the token is physically located within a specific distance.

The goal of this project is to apply this concept to the ping time between two computers in order to prove how close the computers are from eachother. A distance-bounding protocol proof uses the distance, speed, and time equation solved for distance.

$$(\text{distance}) = (\text{speed}) \cdot (\text{time}) \tag{1}$$

The speed is set to the speed of light since one conclusion from the theory of special relativity is that no information signal or material can travel faster than light in a vacuum. The time is set to half the ping time (round trip time divided by 2).

$$(\text{distance}) = (\text{speed of light}) \cdot \frac{(\text{ping time})}{2} \tag{2}$$

In the protocol, a verifier **V**, and a prover **P**, create a pair of one-time-use pseudorandom bit sequences, R^0 and R^1, each containing n elements. Each element R_i^0 or R_i^1 is a bit whose value is either 0 or 1. These sequences can be represented like so:

$$R_i^0 = R_1^0 R_2^0 R_3^0 R_4^0 R_5^0 \cdots R_n^0 \tag{3}$$
$$R_i^1 = R_1^1 R_2^1 R_3^1 R_4^1 R_5^1 \cdots R_n^1 \tag{4}$$

Regarding these bit sequences, **V** rapidly asks **P** a stream of n questions. A question may take only one of the two forms:

1. What is the ith bit of R^0, R_i^0?

2. What is the ith bit of R^1, R_i^1?

The stream of questions start with $i=1$ and end with $i=n$.

In order to decide which question **V** asks **P**, **V** generates a private random bit sequence, C, which consists of n elements. The rule **V** follows is that if $C_i=0$ then **V** requests that **P** supply R_i^0. If $C_i=1$ then **V** requests that **P** supply R_i^1. In other words, at each round, i, **V** randomly chooses which of the two questions to ask **P**.

After sending a question to **P**, **V** records the exact time and increments i by 1.

Because cause must precede effect, **P** cannot provide a correct answer to **V** until after **P** receives the question. Since the speed of light is the maximum rate at which any information can travel through space, there is a minimum ping time (or "time of flight") for any given distance between **V** and **P** which can be used by the protocol to prove an upper bound to the distance between **V** and **P**.

Immediately after receiving a question, **P** sends to **V** the value $R_i^{C_i}$ which is the requested bit from either R^0 or R^1. The set of these responses can be written as R^{C_i}.

Upon receiving each response, **V** records the exact time in order to calculate that particular question-response round-trip time (or "ping time").

2.2 Example 1: how the bit sequences are used

To help explain how this process works below is an example that sets $n=16$ and walks you through how to calculate the response bit sequence, R^{C_i}.

1. Verifier **V** and Prover **P** assemble and agree upon pseudorandom bit sequences R^0 and R^1.

$$R_i^0 = 0100101110110010 \quad (5)$$
$$R_i^1 = 1000111101101001 \quad (6)$$

2. Verifier **V** secretly produces a pseudorandom bit sequence C_i.

$$C_i = 0000101100011101 \quad (7)$$

3. **V** sends each bit of C_i , one at a time, starting from $i=1$ until $i=n$. **V** notes the exact time when it sent each value of C_i.

4. **P** receives and uses each bit of C_i to determine whether to immediately send the bit R_i^0 or R_i^1 to **V** in response. If all bits are received and sent without error, **P** will eventually have sent the set R^{C_i}.

5. **V** receives and records the arrival time for each response bit, $R_i^{C_i}$. **V** calculates the round-trip time for each round. The resulting values of $R_i^{C_i}$ are:

$$R_i^{C_i} = 0100101110101011 \quad (8)$$

Below is a table illustrating how the example values for these bit sequences correlate. I have bolded the values of R_i^0 and R_i^1 which were sent by **P** in response to the values sent of C_i sent by **V**.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
R_i^0	**0**	**1**	**0**	**0**	1	**0**	1	1	**1**	**0**	**1**	1	0	0	**1**	0
R_i^1	1	0	0	0	**1**	1	**1**	**1**	0	1	1	**0**	**1**	**0**	0	**1**
C_i	0	0	0	0	1	0	1	1	0	0	0	1	1	1	0	1
$R_i^{C_i}$	0	1	0	0	1	0	1	1	1	0	1	0	1	0	1	1

Table 1. An example showing how C_i is used to select which bit (**bolded**) from R_i^0 or R_i^1 is used to build $R_i^{C_i}$.

2.3 Explanation, part 2: False-Acceptance Rate

At each step **V** records the round trip time required between the sending of the question and the receiving of the correct answer from **P**. Given enough correct answers from **P**, **V** can then use the average value of the round trip time, tm, of correct responses in order to calculate with some statistical certainty that **P** is physically located within a distance, d. The distance, d can be calculated using the following two equations (pg 68, Hancke 2005).

$$d = c \cdot \frac{t_m - t_d}{2} \tag{9}$$

$$t_m = 2 \cdot t_p + t_d \tag{10}$$

In the language of the Hancke paper, variables in the two equations are defined as:

> c is the propagation speed, t_p is the one way propagation time, t_m is the measured total round-trip time, and t_d is the processing delay of the remote device.

A conservative practice defines $t_d = 0$ for the processing delay variable. It is conservative because t_d is a function of the capabilities of the hardware **P** uses to process requests from **V**. If both **P** and **V** trust eachother to use specific hardware with consistent and accurate estimates for response times then t_d may be specified. However, the Hancke protocol-Kuhn does not provide a means for proving or incentivizing **P** to accurately measure and report its own hardware capability.

The highest possible propagation speed, c, according to the laws of physics is the speed of light in a vacuum. According to section 2.1.1.1 of the 8th edition of the International System of Units, a document published by the International Bureau of Weights and Measures, this speed is 299,792,458 $\frac{\text{m}}{\text{s}}$.

The statistical certainty that the round-trip time between **P** and **V** is less than t_m is $1 - p_{\text{FA}}$ where p_{FA} is the "false-accept probability". The value of p_{FA} must be a statistical estimate constrained by the possibility that prover, **P**, maliciously sends its best guesses before receiving the questions from **V**. If **P** dishonestly wishes to convince **V** that the distance is lower than it really is, then **P** can achieve a $\frac{3}{4}$ probability of guessing correctly for a given round without having yet received that round's value of C_i. This is because, on average, half of the rounds do not require guessing at all since half the time $R_i^0 = R_i^1$. The other half of the time **P**'s best strategy, assuming **V** generated C securely, is to guess 1 or 0 at random.

The false acceptance probability, or "False-Acceptance Rate", p_{FA}, of **V** accepting the distance-bounding protocol proof of **P** can be calculated using the following equation found on the sixth page of the Hancke paper. This equation calculates p_{FA} assuming **V** judges that receiving k correct responses out of n total rounds is acceptable.

$$p_{\text{FA}} = \sum_{i=k}^{n} \binom{n}{i} \cdot \left(\frac{3}{4}\right)^i \cdot \left(\frac{1}{4}\right)^{n-i} \tag{11}$$

The equation states that p_{FA} is equal to the sum of each individual probability where **P** guessed correctly k or more times (for example: one outcome exists where **P** guesses perfectly, some outcomes where **P** makes only one mistake, some outcomes where **P** makes two mistakes, etc.). The total number of terms in the sum is $n - k + 1$.

In other words, the final term (the n'th term) of the sum is the probability that **P** guesses correctly in exactly every single response (one very rare possibility). The penultimate term (the $(n-1)$'th term) is the probability that **P** guesses correctly every single time *except* for exactly *one* mistake somewhere (a slightly less rare possibility). The $(n-2)$'th term is the probability that **P** guesses all responses correctly but with *two* errors somewhere. The $(n-3)$'th term is the probability that **P** guesses all responses correctly but with *three* errors somewhere, and so forth. The first term of the sum is the probability that **P** guesses correctly exactly k times out of n responses and therefore provided incorrect responses exactly $n-k$ times. Each term of the sum is the binomial probability function (a.k.a. "binomial distribution formula" or "probability mass function") which should be part of the syllabus for any a typical Statistics course.

Since no factor of the equation for p_{FA} can be made exactly equal to zero it is impossible for Verifier **V** to completely eliminate the possibility that **P** could forge this distance-bounding proof. The best **V** can do to strengthen confidence in the proof's validity is to set the parameters k and n to values that produce an acceptably low value for p_{FA}, the probability of falsely accepting a maliciously constructed proof by Prover **P**.

2.4 Example 2: Calculating False-Acceptance Rate

Below is a copy of the previous example table, table 1, but with values of R_i^0 and R_i^1 bolded when $R_i^0 = R_i^1$. From inspection it should be clear that **P** does not have to guess roughly half of the rounds since a quarter of the time $R_i^0 = R_i^1 = 0$ and a quarter of the time $R_i^0 = R_i^1 = 1$.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
R_i^0	0	1	**0**	**0**	**1**	0	**1**	**1**	1	0	**1**	1	0	**0**	1	0
R_i^1	1	0	**0**	**0**	**1**	1	**1**	**1**	0	1	**1**	0	1	**0**	0	1
C_i	0	0	0	0	1	0	1	1	0	0	0	1	1	1	0	1
$R_i^{C_i}$	0	1	0	0	1	0	1	1	1	0	1	0	1	0	1	1

Table 2. An example showing instances when guessing the correct bit of $R_i^{C_i}$ without C_i is less difficult due to $R_i^0 = R_i^1$.

Side note: I believe the inefficiency of allowing the protocol to have instances where $R_i^0 = R_i^1$ is due to Hancke designing the protocol to be simple in order to accomodate implementation in RFID tags with limited computatioinal ability and over noisy communication channels. The scope of this project doesn't include attempting to improve the protocol but to simply implement it as described in the Hancke paper.

In order to illustrate how the False-Acceptance Rate, p_{FA}, is calculated, let us say that **V** was programmed to accept 14 correct responses out of 16 ($k = 14$, $n = 16$). In this case p_{FA} could be calculated as described below.

The binomial coefficient factor in the p_{FA} equation can be expanded out, with ! signifying the factorial operation (for example, $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$).

$$p_{\text{FA}} = \sum_{i=k}^{n} \frac{n!}{i!\,(n-i)!} \cdot \left(\frac{3}{4}\right)^i \cdot \left(\frac{1}{4}\right)^{n-i} \tag{12}$$

The sum consists of a total of $n - k + 1 = 16 - 14 + 1 = 3$ terms.

The last term ($i = n = 16$) is:

$$\frac{16!}{16!\,(16-16)!} \cdot \left(\frac{3}{4}\right)^{16} \cdot \left(\frac{1}{4}\right)^{16-16} = 1.00226 \cdot 10^{-2} \tag{13}$$

The penultimate term ($i = 15$) is:

$$\frac{16!}{15!\,(16-15)!} \cdot \left(\frac{3}{4}\right)^{15} \cdot \left(\frac{1}{4}\right)^{16-15} = 5.34538 \cdot 10^{-2} \tag{14}$$

The first term ($i = k = 14$) is:

$$\frac{16!}{14!\,(16-14)!} \cdot \left(\frac{3}{4}\right)^{14} \cdot \left(\frac{1}{4}\right)^{16-14} = 1.33635 \cdot 10^{-1} \tag{15}$$

The sum of these three terms is:

$$1.00226 \cdot 10^{-2} + 5.34538 \cdot 10^{-2} + 1.33635 \cdot 10^{-1} = 1.97111 \cdot 10^{-1} \tag{16}$$

Therefore, the False-Acceptance Rate, p_{FA}, can be written as:

$$p_{\mathrm{FA}} = \sum_{i=k=14}^{n=16} \frac{n!}{i!\,(n-i)!} \cdot \left(\frac{3}{4}\right)^{i} \cdot \left(\frac{1}{4}\right)^{n-i} = 1.97111 \cdot 10^{-1} = 19.7111\% \tag{17}$$

In other words, if **V** decides to accept only $k = 14$ or more correct bits from from **P** out of a possible $n = 16$ bits in the bit sequences they exchange, then there is about a 19.7% chance that **P** could fool **V** into accepting that the distance between them was lower than it physically is. **P** could do this by completely disregarding **V**'s questions, C, and sending only best guesses for bit sequence R^{C_i} given the structure of R^0 and R^1.

Written in TeXmacs.