Analysis on AACS' Traitor Tracing against Mix-and-Match Attacks

Bin ZHU, Min FENG
Microsoft Research Asia, Beijing 100080, China
{binzhu, minfeng}@microsoft.com

Fen LIU, Lei HU
Key Lab of Information Security, Graduate Univ. of Chinese Academy of Sciences, Beijing, China
{vanillaliu, hu}@is.ac.cn

Abstract *— In this position paper, we report the progress of our project to analyze security of the traitor tracing technology used in Advanced Access Content System (AACS). For a simplified problem that all sequence keys are statically assigned according to known Reed Solomon (RS) codes, and a colluder is to be identified with the traitor tracing, we present a mix-and-match colluding attack to victimize an innocent device such that using the highest score of matches with AACS' traitor tracing to identify a colluder will always indentify the victim as a colluder no matter how many movies are tested. Both theoretical analysis and experimental results show that a group of arbitrary 20 or more colluding devices out of a billion devices supported by AACS can always successfully victimize an innocent device. With 26 arbitrary colluders, an arbitrarily given device has a probability of 39.13% to be successfully victimized. Moreover, our attack enables everyone in a group as small as 6 arbitrary colluding devices to escape from being identified as a colluder with the traitor identification method. Analysis with more realistic assumptions will be reported in the future*[1].

1 Introduction

Intellectual property protection technologies for digital media have been actively pursued in both the academia and the industry in the past decade. In July 2004, eight companies, Disney, IBM, Intel, Matsushita, Microsoft[2], Sony, Toshiba, and Warner Bros., announced to work together on content protection for the next generation of high-definition DVD optical discs including both the HD-DVD and Blue Ray disc formats. The technology they have been developing is called the *Advanced Access Content System* (AACS) [1]. The fundamental protection in the AACS system is the broadcast encryption with a subset difference tree [2]. It allows precise revocation to disable clones and compromised devices from decrypting the content in newly released high-definition DVDs without the danger of collateral damage to innocent devices, once the keys in the circumvention devices are identified. It is a challenge to determine which keys are compromised in AACS since a broadcast encryption system is inherently anonymous. To determine which keys are compromised, the license agency obtains a copy of the circumvention device, feeds it with test forensic media key blocks, and observes if the device is able to use it or not for each fed block. Note that some of the keys will give invalid results. For the broadcast encryption with subset difference, a circumvention device can easily determine if it is under test with forensic media key blocks if it knows the keys of more than one device, and may react smartly to avoid identification of the keys it knows [2].

A trait tracing technology called *sequence keys* has been used in AACS to identify colluders (also called traitors) [3][4]. In this scheme, 15 segments are carefully selected from a movie, each segment is of 2 second duration, and has 16 variants. The total overhead is $(16-1)\times 2\times 15 = 450$ seconds of video content. The 16 variants of a segment appear the same to human eyes but can be identified with some technologies such as watermarking. Based on the key a device knows for a segment, a specific one out of the 16 variants of the segment is decrypted and displayed. Based on the observed video, the keys applied to the 15 segments are identified. By feeding a circumvention device with movies and observing the outputs, at least one colluder is identified. The traditional method is to score each individual device with the matched outputs and the device with the highest score of matches is identified as a colluder.

We have been analyzing AACS' traitor tracing technology, and have recently made some progress. As the first step, we simplify the problem by assuming that all sequence keys are statically assigned according to known RS codes, and a colluder is to be identified with the traitor tracing. For this simplified problem, we have designed a mix-and-match attack. Our theoretical analysis and experimental results show that a group as small as 20 arbitrary colluding devices out of 1 billion devices supported by AACS can always successfully victimize an innocent device which is always identified as a colluder, no matter how many movies are used in a test. With 26 arbitrary colluding devices, an arbitrarily given device has a probability of 39.13% to be successfully victimized. If we relax the requirement that none of the colluding devices is identified as a colluder by the traditional method, the size of the colluding group can be reduced to as small as 6 arbitrary colluding devices. In this position paper, we report these results. Analysis on the set cover method [3] as well as with more realistic assumptions will be reported in the future.

This paper is organized as follows. In Section 2.3, we present a brief introduction to AACS' traitor tracing technology. Our mix-and-match attack is described in Section 3. Theoretical analysis and experimental results are presented in Section 4 and Section 5, respectively. The paper concludes in Section 6.

2 Background

2.1 Reed-Solomon Codes

Reed-Solomon (RS) codes [5] are linear block codes widely used for error correction. For a q-ary RS code, the RS encoder takes j source symbols of $s = log_2 q$ bits and appends parity

[1] This work was done when Fen LIU was an intern with Microsoft Research Asia.

[2] None of the authors has been involved in AACS or has access to AACS' internal documents. This work is based on the public information from the references listed at the end of this paper.

1-4244-1457-1/08/$25.00 © IEEE

symbols to make an $m = q - 1$ symbol code word. Such an RS code is represented by a triple tuple (m, j, d) in this paper, where $d = m - j + 1$ is the minimal Hamming distance between two code words. When error occurs, the received word is no longer a valid code word. The RS decoder tries to correct the error by finding the closest valid code word. An RS code (m, j, d) can correct up to $t = \lfloor (m - j)/2 \rfloor$ erroneous symbols. If a word w is within a Hamming distance of t or less from a valid code word c, then c is the code word closest to w among all the valid code words.

2.2 AACS' Threat Model

Threat model is important in analyzing a security scheme. In an attack called *key attack*, attackers use reverse engineering technologies to extract the decryption keys in some devices and decrypt a version of DVD content to avoid identification of the compromised keys. There are two well-known models on how a pirated copy can be generated [3][4]:

1. Given two variants v_1 and v_2 of a segment, a pirated copy can be either v_1 or v_2. No other variants can be generated.
2. Given two variants v_1 and v_2 of a segment, $v_2 \neq v_2$, a pirated copy can be any variant out of v_1 and v_2.

AACS focuses on attacks restricted to the first model. Although we believe that attacks of the second model are not very difficult to launch, we shall focus on attacks of the first model in our analysis in this paper.

2.3 AACS' Sequence Keys

In AACS' traitor tracing scheme, each variant of a segment, i.e., a key, is assigned according to a code called an *inner code*, which is then encoded using another code called an *outer code*. This nested code is called the *supercode*. The inner code creates effectively multiple movie versions for a movie and the outer code assigns different movie versions to a device for a sequence of movies. As a result, AACS' traitor tracing scheme is called the *sequence keys*. RS codes are used to generate both the inner and outer codes. The inner code used in AACS is (15,2,14) with $q_{in} = 16$, and the outer code is (255,4,252) with $q_{out} = 256$. As a result, each segment has $q_{in} = 16$ variants, and a movie can generate 256 different versions. Each symbol in an outer code word maps to an inner code word. Therefore an outer code word maps to 255 inner code words. An inner code word has 15 symbols. Each inner code symbol corresponds to one variant of the corresponding segment. One outer code symbol, i.e., one inner code word, is used by a device to generate a movie version. The total number of outer code words is $q_{out}^4 = 256^4$. Therefore AACS can accommodate about 4 billion devices, much larger than 1 billion devices AACS wants to support.

The key assignment in AACS is static. The keys stored in a device cannot be dynamically updated once the device is shipped. A simple method is to assign all the keys used to select variants of segments at manufacturing time. That means that each device should store 15×255 keys. To save storage space used to store sequence keys in a device, AACS assigns each device only 255 movie version keys corresponding to an outer code word. A movie version key is used to decrypt a row of 15 keys stored together with the movie in a high-definition DVD disc. Each of the 15 keys is used to decrypt a variant for the corresponding segment. For simplicity, we ignore this indirect key assignment in our current analysis, and assume that each device still stores 15×255 keys. For each movie, the 15 keys corresponding to a specific inner code word are used to decrypt the 15 segments to generate a movie version. An inner code word is determined by a symbol in the outer code word.

The traitor tracing problem is traditionally defined as to find a single member in a group of colluders. The method is to score each individual device based on how many matches with what the device has been assigned. The device with the highest score is identified as a colluder. This traitor identification method is used in our current analysis. In addition, we simplify the problem by assuming that all the sequence keys are assigned statically according to publicly known RS codes. AACS also applies the set cover method to identify the whole group of colluders [3]. The set cover method and dynamical key assignment will be analyzed in the future.

3 Our Mix-and-Match Attack

The security analysis on AACS' sequence keys given in [3][4] is based on movie-by-movie attacks in which movie versions from colluders are mixed, i.e., the variants v_1 and v_2 in the threat model described in Section 2.3 are movie versions instead of segments. According to this threat model, AACS allows a mix of segments from different colluders for a movie version. By exploiting that there are only 256 movie versions for a movie, our mix-and-match attack mixes segments from different colluders to forge a movie version for each test movie that an innocent device is consistently identified as a colluder with the scoring method, no matter how many movies are used in a test. In this attack, the forged movie version may be an invalid movie version, i.e., the movie version may not correspond to any individual device's movie version. The license agency can detect that the output version is a colluding result, and applies the scoring method to identify a single colluder.

In our mix-and-match attack, colluders want to find an innocent device as the victim which has consistently the highest score of matches among all the devices for each test movie for an arbitrary sequence of test movies. If we express the requirement in the supercode, it is to find an innocent outer code word that the colluders can use their corresponding inner code words to forge a word that has the shortest Hamming distance to the victim's inner code word among all valid inner code words for each dimension of the outer code.

How can colluders find such a victim device? The trick for a successful attack is to exploit the properties of the RS codes used to assign sequence keys in AACS. The inner code is a 16-ary RS code (15,2,14), which implies that if a device has 9 or more matched segments out of the 15 segments in total for a test movie, then the device is guaranteed to have the highest score among all the devices for the test movie. The goal of our mix-and-match attack is therefore to find an outer code word such that for the inner code word corresponding to each

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.

dimension of the outer code, the colluders can construct a word from their inner code words to match the victim's inner code word for at least 9 symbols, i.e., the Hamming distance from the forged word to the victim's inner code word is equal to or less than 6. This can be formulated as follows.

Assume there are N colluding devices $\{d_i^{old} | i \in [1, N]\}$. Our mix-and-match attack is to find a victim device d^{vtm} such that for an arbitrary integer $k \in [1,255]$, there exists a word v_k constructed from the inner code words $\{c_k^{old}\}$ of the colluding devices corresponding to their respective k-th outer code symbols such that $\|v_k - c_k^{vtm}\| \leq 15 - n$, where $\|\cdot\|$ means Hamming distance, and $15 - n \leq 6$, i.e., $n \geq 9$, for each k-th outer code dimension, $1 \leq k \leq 255$. Each symbol in the word v_k is selected from the symbols of the colluding inner code words at the same position. Therefore n can be interpreted as the minimum number of the inner code symbols in the victim's inner code word that each symbol matches at least one colluder's inner code symbol at the same position. For the AACS inner code, two distinct valid inner code words share at most one symbol. If the victim's inner code word is different from colluder's inner code words, then at least n colluders are needed to construct a word that has n inner code symbols matched with the victim's inner code word.

We can relax the above requirement by requesting the colluders to find a victim whose Hamming distance to the forged word is consistently shorter than or equal to that of the closest inner code word of the colluders for every outer code dimension. This relaxed requirement cannot guarantee that the victim is consistently identified as a colluder by the scoring method but can guarantee that the victim is identified as a colluder before any one of the colluders is identified.

In the next two sections, we give theoretical analysis and experimental results, respectively, on the following problem: given a set of N colluding devices $\{d_i^{old} | i \in [1, N]\}$ *randomly* selected from the 2^{32} available devices, and given an arbitrary device d, what is the probability that d has, for each of its inner code word, at least n inner code symbols that each symbol matches at least one of the inner code words of the N colluders. If $n \geq 9$, this probability is the probability that the N colluding devices can successfully victimize the specific device. In Section 5, we also describe the experiment to actually find a victim device by a brute force search as well as the result for the relaxed requirement.

4 Theoretical Analysis

To find the probability of the problem described at the end of Section 3, let's look at an inner code word corresponding to an outer code symbol at an arbitrary position $k \in [1,255]$. There are two cases.

Case 1: The inner code word c of the targeted device d is the same as one of the inner code words of the N colluding devices. The probability is $1 - (255/256)^N$.

Case 2: The inner code word c of the targeted device d is different from any of the inner code words of the N colluding devices. We can construct a word v from the N colluding inner code words which shares with the targeted inner code word for at least n inner symbols. The following fact is used in calculating the probability in this case.

Fact: Any two distinct inner code words can share at most one dimension (i.e., one symbol) in AACS' sequence keys.

The fact is true since the inner code is the RS code with Hamming distance of 14, and the length of an inner code word is 15 symbols.

To find the probability in Case 2, let's count how many N inner code word tuples that can construct a word v with n symbols shared with c. There are C_{15}^n different choices to choose n symbols out of 15 symbols. Order the indexes of the selected n symbols from low to high as $(b_1, b_2, \ldots, b_n)$. Let $x_i, 0 < i < n$, be the number of N colluding inner code words having the same symbol as c in position b_i. Let x_0 be the number of N colluders which does not share any symbol with c. Then we have the following equation:

$$\begin{cases} x_0 + x_1 + \cdots + x_n = N \\ x_0 \geq 0; x_i > 0 \ (0 < i \leq n) \end{cases} \quad \ldots\ldots (E)$$

Define another equation:

$$\begin{cases} x_0 + x_1 + \cdots + x_n = N \\ x_i \geq 0 \ (0 \leq i \leq n) \end{cases} \quad \ldots\ldots (E_n)$$

For every solution of Eq. (E), there are $C_N^{x_0,x_1,x_2,\ldots,x_n}$ ways to satisfy the requirement. If the j-th colluding inner code word shares a symbol with c, then one symbol of j-th colluding inner code word is decided. This leaves only 15 choices to fill the remaining symbols, since each inner code word has a freedom of two symbols, and a colluding inner code word cannot have more than more symbols shared with c, otherwise the colluding inner code word is the same as c. If the j-th colluding inner code word has no symbol shared with c, then there are 30 possibilities to choose. This is calculated as follows. An inner code word has 15 symbols. There is one inner code word equal to c. There are 15×15 inner code words which share one symbol with c. The total number of different inner code words is 256. Therefore there are $256 - 1 - 15 \times 15 = 30$ inner code words that don't share anything with c. For every solution of Eq. (E), there are $C_N^{x_0,x_1,x_2,\ldots,x_n} 15^{N-x_0} 30^{x_0} = C_N^{x_0,x_1,x_2,\ldots,x_n} 15^N 2^{x_0}$ N inner code word tuples that can construct a word v which shared n symbols with the inner code word c.

As a result, the probability of case 2 is

$$\left(\frac{1}{256}\right)^N \sum_{(x_0,x_1,\cdots x_n)\in E} \binom{N}{x_0, x_1, \cdots x_n} 15^{\mathrm{N}} 2^{\mathrm{x_0}}$$

$$= \left(\frac{15}{256}\right)^N \sum_{(x_0,x_1,\cdots x_n)\in E} \binom{N}{x_0, x_1, \cdots x_n} 2^{\mathrm{x_0}}$$

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.

$$= \left(\frac{15}{256}\right)^N \left\{ \sum_{(x_0,x_1,\cdots x_n)\in E_n} \binom{N}{x_0, x_1, \cdots x_n} 2^{x_0} + \sum_{j=1}^{n} (-1)^j \sum_{0<i_1<\cdots<i_j\le n} \sum_{\substack{(x_0,x_1,\cdots x_n)\in E_n \\ x_{i_1}=x_{i_2}=\cdots=x_{i_j}=0}} \binom{N}{x_0, x_1, \cdots x_n} 2^{x_0} \right\}$$

$$= \left(\frac{15}{256}\right)^N \left\{ \sum_{(x_0,x_1,\cdots x_n)\in E_n} \binom{N}{x_0, x_1, \cdots x_n} 2^{x_0} + \sum_{j=1}^{n} (-1)^j \binom{n}{j} \sum_{(x_0,x_1,\cdots x_{n-j})\in E_{n-j}} \binom{N}{x_0, x_1, \cdots x_{n-j}} 2^{x_0} \right\}$$

$$= \left(\frac{15}{256}\right)^N \left\{ \left(2 + \underbrace{1+1+\cdots+1}_{n \text{ times}}\right)^N + \sum_{j=1}^{n} (-1)^j \binom{n}{j} \left(2 + \underbrace{1+1+\cdots+1}_{n-j \text{ times}}\right)^N \right\}$$

$$= \left(\frac{15}{256}\right)^N \sum_{j=0}^{n} \binom{n}{j} (-1)^j (2+n-j)^N$$

Therefore the probability to have at least n symbols shared with N colluding inner code words of colluders is

$$1 - \left(\frac{255}{256}\right)^N + \left(\frac{15}{256}\right)^N \sum_{i=n}^{15} \sum_{k=0}^{i} \binom{i}{k} (-1)^k (2+i-k)^N$$

The probability to have at least n symbols shared with the inner code words of colluders for every outer code symbol is

$$P = \left\{1 - \left(\frac{255}{256}\right)^N + \left(\frac{15}{256}\right)^N \sum_{i=n}^{15} \sum_{k=0}^{i} \binom{i}{k} (-1)^k (2+i-k)^N\right\}^{255} \quad \ldots\ldots\ldots\ldots\ldots \quad (1)$$

5 Experimental Analysis

We have also implemented our mix-and-match attack to actually find out an outer code word d with a brute force search for each group of N colluding outer code words $\{d_i^{old} | i \in [1, N]\}$ randomly selected from the available outer code words such that d shares at least n inner code symbols with the colluders for every outer code dimension, where $n \geq 9$. The innocent device associated with the outer code word d has then the highest score consistently and is thus identified as a colluder for an arbitrary sequence of test movies. When running on a Dell OptiPlex GX280 with 3.2 GHz Intel Pentium 4 CPU and 1GB memory, our program could find a victim device quickly, usually within a few second for $N > 22$, less than a minute for $N = 22$, and less than a few minutes for $N = 21$, all with $n = 9$. It took about 10 minutes or less to find one victim device for $N = 20$ with $n = 9$. It took a substantially long time to find a victim for $N < 20$ with $n = 9$. As a result, we have not collected any data for $N < 20$ when $n \geq 9$. From our experiments, for any randomly selected N devices as colluders, we could always find a victim device for $N \geq 20$ when $n = 9$.

As we mentioned previously, when $n \geq 9$, it is guaranteed that the victim device is consistently identified as a colluder. If we use the relaxed requirement that the Hamming distance from the victim to the forged word is no shorter than that of the inner code word of colluders closest to forged word, denoted as $n = *$, our experiments show that we could always find a victim for $N \geq 6$, usually within a minute or less. It took a substantially long time when $N \leq 5$. As a result, we have not collected any data when $N \leq 5$ and $n = *$.

We have also conducted an experiment to find out the probability to successfully victimize an arbitrarily given device for a group of colluding devices randomly selected from all the available devices. In this experiment, 50,000 outer code words are randomly selected for each group of N colluders also arbitrarily selected. For each selected outer code word, we check if the N colluding outer code words can construct an inner word that shares at least n inner code symbols with the inner code word of the selected outer code word for every outer

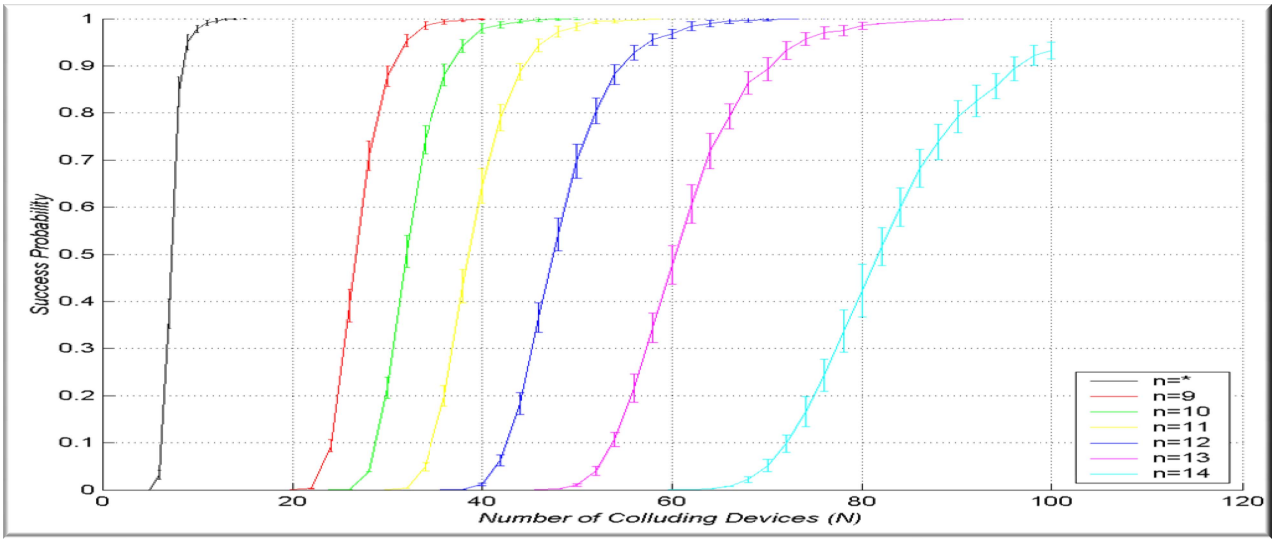

Figure 1: Success probabilities and standard deviations for different number of colluders and *n*.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.

code dimension. The count of success divided by 50,000 is the probability of success. This procedure was repeated 100 times. The resulting average success probability and the corresponding standard deviation to victimize an arbitrarily given device for a group of N randomly selected colluding devices are shown in *Figure 1* for different values of N with n in the range of $[9, 15]$ as well as $*$ (i.e., the relaxed requirement case). From the figure, the success probability with $n = 9$ increases quickly from about 0.0001% for $N = 20$ to 98.60% for $N = 34$. It shows that 34 arbitrary colluders can almost victimize any device, and 26 arbitrary colluders have a probability of 39.13% to victimize an arbitrarily given device.

When $n = *$, i.e., for the relaxed requirement, the success probability to victimize an arbitrarily given device is 3.18% for $N = 6$, 37.45% for $N = 7$, and close to 1 for $N \geq 9$.

Table 1: The minimum number of colluders (N_{min}) for different number of matched inner code symbols (n) for success probability no less than 0.02%.

n	*	9	10	11	12	13	14	15
$N_{\min}$	6	22	26	31	38	47	62	95

We have also calculated the success probability given by Eq. (1) for the same parameters as in the experiments. The theoretical results are shown in *Figure 2* together with the experimental results. As we can see from the figure, the theoretical curves almost overlap with their corresponding experimental curves. It shows that the experimental results agree well with the theoretical results given in Eq. (1).

Table *1* shows the minimum number of colluders required for different n if the success probability is no less than 0.02%.

6 Conclusion

We have presented an efficient mix-and-match colluding attack to victimize an innocent device which will be identified as a colluder with AACS' traitor tracing scheme and the scoring method to identify a colluder if we assume that all sequence keys are assigned statically according to known RS codes. Both theoretical analysis and experimental results were presented. The experimental results agree with the theoretical analysis well. Our results indicate that a group of as small as 20 arbitrary colluding devices can always successfully victimize an innocent device that is consistently identified as a colluder. With 26 arbitrary colluders, there is a probability of 39.13% to victimize an arbitrarily given device. With relaxed requirement that the victim is identified before any colluder without guaranteeing that the victim is always identified as a colluder no matter how test movies are used, the group of randomly selected colluders can be as small as 6 colluders to find a victim device. Analysis with more realistic assumptions will be reported in the future.

References

[1] Advanced Access Content System, http://www.aacsla.com/home.

[2] J. Lotspiech, "Broadcast Encryption," in *Multimedia Security Technologies for Digital Rights Management*, W. Zeng, H. Yu, and C.-Y. Lin, Eds., Elsevier, Amsterdam, Boston, London, chap. 12, pp. 303-322, 2006.

[3] H. Jin and J. Lotspiech, "Practical 'Traitor Tracing'," in *Multimedia Security Technologies for Digital Rights Management*, W. Zeng, H. Yu, and C.-Y. Lin, Eds., Elsevier, Amsterdam, Boston, London, chap. 12, pp. 323-347, 2006.

[4] H. Jin, J. Lotspiech, and N. Megiddo, "Efficient Traitor Tracing," IBM Technical Report RJ10390 (A0610-018), Oct. 2006.

[5] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice Hall, New York, 1983.

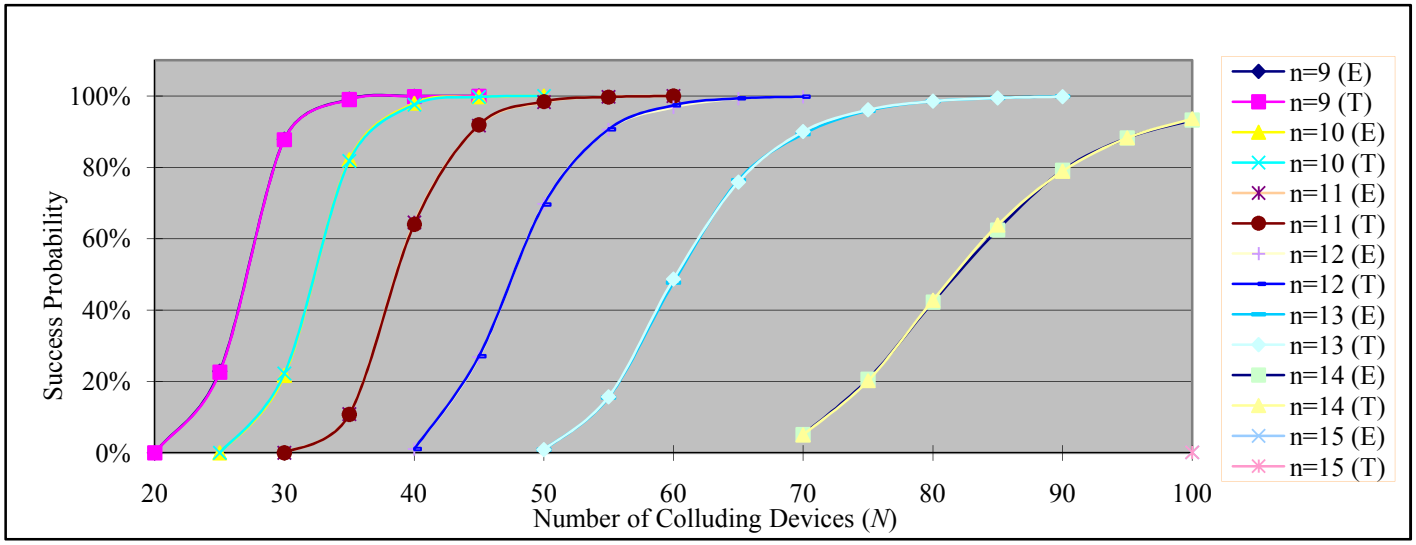

Figure 2: Comparison of theoretical results (T) with the experimental results (E) for different *N* and *n*.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.