Bayesian Methods for Practical Traitor Tracing

Philip Zigoris[1] and Hongxia Jin[2]

[1] University of California, Santa Cruz
Santa Cruz, CA 95060, USA
[2] IBM Research, Almaden
San Jose, CA 95120, USA

Abstract. The practical success of broadcast encryption hinges on the ability to (1) revoke the access of compromised keys and (2) determine which keys have been compromised. In this work we focus on the latter, the so-called *traitor tracing* problem. The first method utilizes a Bayesian hierarchical model to replace a crucial step in a well known tracing algorithm. Previously, this step relied on worst case bounds, which often overestimate the number of tests needed to diagnose compromised keys. The second is an adaptive tracing algorithm that selects forensic tests according to the *information gain* criteria. The results of the tests refine an explicit model of our beliefs that certain keys are compromised. In choosing tests based on this criteria, we significantly reduce the number of tests, as compared to the state-of-the-art techniques, required to identify compromised keys.

1 Introduction

Digital piracy is a burning problem for the entertainment industry. After all, digital data can be perfectly and quickly copied. If consumers may freely copy entertainment content and offer that content on the Internet, the market for entertainment content would evaporate. To solve this problem, several frameworks for broadcast encryption [3] have been devised and are in wide use in the market, like CPRM, CPPM, DTCP and AACS [1] [2]. All of these methods are based on encryption of the content: the device manufacturer is given cryptographic keys to decrypt the content, and in return is obligated by the license to follow a set of rules limiting the physical copies that can be made from a single piece of content. Maintaining the secrecy of the cryptographic keys is essential for maintaining the integrity of distribution. scheme. In the event that some keys are compromised and made public, a content protection scheme should be able to revoke their ability to access future content. To combat this eventuality, each device is assigned a set of device keys that can be indirectly used to decrypt the content. The device keys, owned by compliant devices, repeatedly encrypt the content encrypting key (called the media key) in a structure called a media key block (MKB). Each device uses a device key to decrypt the media key block to obtain a valid media key to decrypt the content. The revoked devices, on the other hand, cannot decrypt the MKB and obtain a valid media key to decrypt the content.

J. Katz and M. Yung (Eds.): ACNS 2007, LNCS 4521, pp. 194–206, 2007.

To circumvent the content protection scheme, an adversary may break a device, extract the device keys, and build a circumvention device (also known as a clone device or a clone box) comprising the extracted device keys. To identify which original devices (called traitors) have donated their keys to the circumvention device, traitor-tracing technologies [4] are used. Traitor-tracing technology uses carefully crafted media key blocks called *forensic media key blocks*. When a circumvention device is found, the license agency feeds a series of forensic media key blocks to the device. By observing the clone's responses the licensing agency can determine precisely which device keys the circumvention device comprises. The licensing agency can then produce new media key blocks that revoke those compromised device keys such that newly released content cannot be played by the circumvention device.

Since many devices can donate their keys the clone box, it is important that a traitor tracing method is resistant to collusion; this represents the real challenge in traitor tracing. The state-of-art and practice broadcast encryption and traitor-tracing technology comprises a subset-difference scheme, described in [11]. Tracing under this scheme has proven to be theoretically efficient: To determine the compromised keys, the traitor tracing method requires on the order of $T^3 \log(T)$ forensic media key blocks to defeat a circumvention device comprising T sets of compromised device keys. However, this method has not proven to be a completely practical solution. Measures can be taken by the circumvention device to slow down the testing process. For example, each testing iteration may take a minute or more. A circumvention device comprising 100 compromised keys (i.e., $T = 100$) may require over 15 years to determine the device keys the circumvention device has compromised. In effect, such a circumvention device had defeated the content protection system.

The main contribution of this paper is to present two much more efficient, thus, practical traitor tracing schemes. Efficiency is measured by the number of forensic media key blocks required to detect traitors from the circumvention device. The first method, *BayesNNLTrace*, maintains the strict black-box assumptions of previous methods, but uses a Bayesian hierarchical model for diagnosis instead of relying on worst case bounds. The second method, *IGTrace*, assumes the tracing algorithm has some prior knowledge about the behavior of the clone box. It leverages this knowledge as well as the clone box's response to forensic MKBs to infer an explicit model of which keys have been compromised. Not only does this allow for accurate diagnosis, we can also quantify how informative a potential forensic MKB is, which we then use to guide the tracing process. Overall, adaptively choosing the best forensic MKB at each step and continuously updating our beliefs about which keys are compromised allow us to substantially reduce the number of forensic MKBs needed to identify traitors.

In rest of the paper, we will show the state of art subset tracing in more details in Section 2. This section provides the foundation for all subsequent sections. We will then show the first Bayesian approach to improve the subset tracing in Section 3 when no known strategy is known/assumed. We then show another Bayesian approach for subset tracing in Section 5 with a known strategy. Our

experimental results in Section 6 shows order of magnitude of improvement in the number of forensic MKBs needed to defeat the clone box.

2 NNL Subset Tracing

Naor, et. al [11] present a broadcast encryption framework using subset covers. Let $\mathcal{D}$ be the set of devices and $\mathcal{K}$ be the set of device keys. Every device $d \in \mathcal{D}$ owns a subset of keys, denoted by $\mathcal{K}_d$. Similarly, associated with every key $k \in \mathcal{K}$ is a set of users $\mathcal{D}_k = \{d \in \mathcal{D} : k \in \mathcal{K}_d\}$.

Suppose we want to broadcast some media M, which, for all intent and purpose, is a binary string. We would like to encrypt M in such a way that a set of legitimate devices $L \subseteq \mathcal{D}$ is able to decrypt and view the media. The first step is to encrypt M with some key K, referred to as the *media key*. We will use the term *key* without a qualifier to refer to device keys. We then find a subset of device keys C such that all legitimate devices are *covered*. That is, C is chosen such that $\bigcap_{k \in C} \mathcal{D}_k = L$. Now, for every $k \in C$ we separately encrypt the media key, giving us $E_k(K)$. Ultimately, the following items are broadcast

- The encrypted media: $E_K(M)$
- The encrypted media key: $\langle E_{k_1}(K), E_{k_2}(K), \ldots, E_{k_{|C|}}(K) \rangle$
- An index of the device keys used to encrypt the media key

These items together are referred to as a *media key block (MKB)*. Every device $d \in L$ will own a key used in the MKB and every device $r \in \mathcal{D}/L$, referred to as a *traitor*, will own none. Hence, it cannot recover the content.

Naor, et.al give a black box tracing algorithm where the only means to diagnosis traitors is to submit tests, sometimes referred to as *forensic MKBs*, to the clone box and observe its response. Their tracing algorithm relies on two things:

1. For every key $k \in \mathcal{K}$ such that $|\mathcal{D}_k| > 1$, there exists keys k_1 and k_2 such that $\mathcal{D}_{k_1} \cup \mathcal{D}_{k_2} = \mathcal{D}_k$ and $D_{k_1} \cap D_{k_2} = \emptyset$. This is referred to as the *bifurcation property*. By this property, we can replace k with k_1 and k_2 and still cover the same set of devices.
2. We have access to a method that, given a set of keys F, finds at least one key in F owned by the clone box. This is referred to as *subset tracing*; the name is due to the fact that each key is associated with a subset.

The algorithm maintains a covering of all legitimate devices $\mathcal{F}$, referred to as the *frontier*. The algorithm proceeds by repeatedly identifying a compromised key $k \in \mathcal{F}$, removing it, and adding to $\mathcal{F}$ k_1 and k_2 satisfying the bifurcation property. If $|\mathcal{D}_k| = 1$ then the single device in $\mathcal{D}_k$ is a traitor. This process is reiterated until the clone box is unable to play the MKB associated with the frontier.

We can formalize subset tracing as follows: the *frontier* $\mathcal{F}$ is a set of keys and the clone box owns a subset $\mathcal{C} \subseteq \mathcal{F}$ of these keys. Let $m = |\mathcal{F}|$. The set $\mathcal{C}$ is, of course, unknown to the tracing algorithm. The task is to determine, within

Algorithm 1.

NNLTrace$(\mathcal{F}, a, b, p_{T_a}, p_{T_b})$
if $a = b - 1$ **then**
  return b
else
  $c \leftarrow \lceil \frac{a+b}{2} \rceil$
  $T_c \leftarrow \{f_{c+1}, \ldots, f_{|F|}\}$
  if $|p_{T_c} - p_{T_a}| \geq |p_{T_c} - p_{T_b}|$ **then**
    return NNLTrace$(\mathcal{F}, a, c, p_{T_a}, p_{T_c})$
  else
    return NNLTrace$(\mathcal{F}, c, b, p_{T_c}, p_{T_b})$

some specified confidence ϵ, at least one key $k \in \mathcal{F}$ that is owned by the clone box. That is, $Pr(k \in \mathcal{C}) > 1 - \epsilon$.

The structure of a forensic MKB, or simply a *test*, is quite simple: we *disable* certain keys by encrypting a random bit string instead of the media key. The remaining keys are said to be *enabled*. This way, all devices that rely on disabled keys are unable to decrypt, or play, the content. There are two caveats. The first is that a device, including a clone box, can determine if a key it owns has been disabled. In this way, its possible for a clone box to know it is under test. The second is that since a clone box may contain both enabled and disabled keys, it can still decrypt content when one of its keys is disabled as well as stop playing when some of its keys are enabled. Its response in these situation constitutes an anti-tracing strategy. We will assume that if all of the keys in a clone box are enabled then it will always play. Of course, if none of the keys are enabled then it is impossible for it to play.

From here on, a test can be simply thought of as a set $T \subseteq \mathcal{F}$ of keys. The keys in T are enabled and the keys in $\mathcal{F}/T$ are disabled. Let p_T be the probability that the clone box plays test T. Since the clone box is assumed stateless, we can treat the outcome of the test as a Bernoulli random variable. If the probability of playing two tests, T and T', are not equal then it must be case that the clone box owns a device key in the exclusive-or of the two sets. Formally,

$$p_T \neq p_{T'} \longrightarrow \exists c \in C, c \in T \otimes T'$$

This motivates *NNLTrace* (Algorithm 1), a binary-search-like method for identifying a compromised key. We initially call the procedure with the arguments $(\mathcal{F}, 0, m, p_F, 0)$. The algorithm proceeds by progressively reducing the interval (a, b) in which we know there must be a compromised device-key. It does this by determining the probability that the midpoint test T_c plays and recursing on the the side with the larger difference in the endpoint probabilities.

Technically, we can recurse on either side as long as the probability of the endpoint tests are not equal. However, this allows for the possibility that the algorithm will become trapped at an interval where the end-point probabilities are arbitrarily close to one another. The challenge in this style of tracing algorithm is that we must estimate p_{T_c} by repeatedly submitting T_c to the clone

box. The closer p_{T_a} and p_{T_b} are to one another, the more tests that are needed to confidently decide which is the larger subinterval. The authors show that $O(\log^2(m)\log(\frac{1}{\epsilon})/\Delta^2)$ tests are needed, where $|p_{T_a} - p_{T_b}| \geq \Delta$ (Claim 8). Since our algorithm recurses on the larger side, we have $\Delta > \frac{1}{m}$ and, therefore, the number of tests is upperbounded by $O(m^2 \log^2(m)\log(\frac{1}{\epsilon}))$. Since NNLTrace will recurse $\log(m)$ times, the overall number of tests required for NNLTrace to succeed with probability at least $1-\epsilon$ is upperbounded by $O(m^2 \log \frac{\log m}{\epsilon} \log^3 m)$.

In the *subset difference* key distribution scheme the final frontier size will be $O(2r)$ [11]. Since the size of the frontier can increase by at most one at every iteration of the top-level procedure, the subset tracing procedure is called $O(2r)$ times. Combining this with the bound from the previous paragraph, we have that for r traitors, the entire traitor tracing process will require $O(r^3 \log^3 r)$ tests. As discussed previously, this algorithm, though theoretically efficient, is easily defeated in practice.

3 Bayesian Approach to Subset Tracing

The NNLTrace routine requires the ability to determine, given three tests T_a, T_b, and T_c, if $|p_{T_a} - p_{T_c}| > |p_{T_b} - p_{T_c}|$ (Line 7, Algorithm 1. Henceforth we abbreviate P_{T_a} as P_a. We require that this decision is made with a probability of error less than some specified threshold ϵ. In this section we introduce a principled method for making this decision based on methods from Bayesian statistics. Instead of relying on worst-case bounds to determine the number of times each test is submitted, we model our uncertainty about p_a, p_b, and p_c using a Bayesian hierarchical model. This model is conditioned on the actual responses of the clone box and not a hypothetical worst case. Tests, therefore, are repeated only if we are unable to confidently determine if $|p_a - p_c| > |p_b - p_c|$.

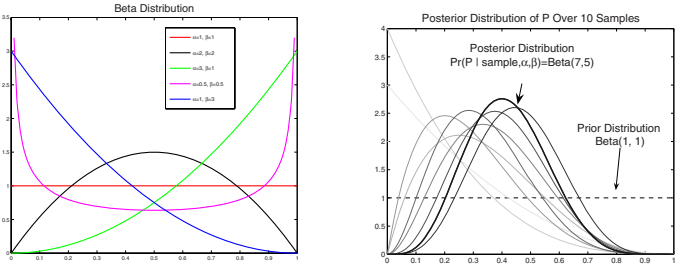
To do this, we think of the probability of playing as a random variable, which we will denote by P_T in order to distinguish it from the actual probability the test plays p_T. For now, assume we have some fixed probability density function (pdf) specifying $Pr(P_T = p)$, abbreviated as $Pr(p)$. This is referred to as the *prior* distribution of P_T since it is not conditioned on any observations. After observing the outcome of a test, $T = t$, we would like to refine our belief about P_T; $t = 1$ implies the clone box was able to decrypt the content, $t = 0$ implies it wasn't. This is provided by Bayes Theorem:

$$Pr(P_T = p|T = t) = \frac{Pr(T = t|P_T = p)Pr(P_T = p)}{Pr(T = t)}$$

We refer to $Pr(P_t = p|T = t)$ as the *posterior distribution*. When there is no risk of confusion we will abbreviate the notation by only writing the observation, i.e. $Pr(P_T = p|T = t) = Pr(p|t)$.

We will model P_T with the Beta distribution, which is defined for p in the interval $[0, 1]$:

$$\text{Beta}(p; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} p^{\alpha-1}(1-p)^{\beta-1}$$



(a) (b)

Fig. 1. (a) Beta distribution for different values of (α, β). (b) Example of posterior distribution of P. The gray curves show the posterior distribution of P after receiving each of 10 observations: $S = \{0, 0, 0, 1, 0, 1, 0, 1, 1, 0\}$.

where $B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha)+\Gamma(\beta)}$ is the Beta *function* and $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}dt$ is the Gamma function. The $B(\alpha, \beta)$ term acts as a normalizing constant. The shape of the distribution is controlled by two parameters, α and β. Various settings of these parameters are illustrated in Figure 1(a). The expectation and variance of $P \sim \text{Beta}(\alpha, \beta)$ are, respectively, $\frac{\alpha}{\alpha+\beta}$ and $\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$.

The choice to use the Beta distribution comes from the fact that it is the *conjugate prior* of the Bernoulli and, implicitly, the Binomial distribution. That is, if the prior distribution for P_T is $\text{Beta}(\alpha, \beta)$ then the posterior distribution $Pr(P_T \mid T = t)$ is simply $\text{Beta}(\alpha + t, \beta + (1 - t))$. Given observations $\{T_i = t_i\}$ for $i = 1$ to n, where $\sum_i t_i = k$, the posterior distribution of P_T is $\text{Beta}(\alpha + k, \beta + (n - k))$. These convenient facts make updating the posterior distribution, illustrated in Figure 1(b), extremely simple. For a more general introduction to Bayesian statistics see, e.g., [6].

Ultimately we need to determine if $|p_a - p_c| > |p_b - p_c|$. Let $\mathcal{P} = \{(p, q, r) \in [0, 1]^3 : |p - r| > |q - r|\}$. Then,

$$\begin{aligned} D(\bar{t}) &= Pr(|P_a - P_c| > |P_b - P_c| \mid \bar{t}) \\ &= \int_{(p_a,p_b,p_c)\in\mathcal{P}} Pr(p_a \mid \bar{t})Pr(p_b \mid \bar{t})Pr(p_c \mid \bar{t}). \end{aligned} \tag{1}$$

It will also be useful to calculate the probability that the two gaps are equal, within some tolerance δ:

$$\begin{aligned} E_\delta(\bar{t}) &= Pr(\,|\,|P_a - P_c| > |P_b - P_c|\,| < \delta) \\ &= \int_{(p_a,p_b,p_c)\in\mathcal{Q}_\delta} Pr(p_a \mid \bar{t})Pr(p_b \mid \bar{t})Pr(p_c \mid \bar{t}). \end{aligned} \tag{2}$$

where $\mathcal{Q}_\delta = \{(p, q, r) \in [0, 1]^3 : |\,|p - r| > |q - r|\,| < \delta\}$. Finding a closed form solution for Equation 1 and 2 is unlikely. Instead, we use a simple *sample-based* approach to inference. The basic idea is that if we have a large enough

sample from a distribution, we can approximate the above integrals. Sampling from a Beta distribution can be done quickly and we do not require approximate sampling methods such as Markov Chain Monte Carlo [10].

After every test, if $D(\bar{t}) > 1 - \epsilon$, $D(\bar{t}) < \epsilon$, or $E_\delta(\bar{t}) > 1 - \epsilon$ then we can recurse on the appropriate subinterval. If not, then we require further testing. In our approach, we repeat the test T with the highest variance in the posterior distribution of P_T.

4 Clone-Box Strategy

It has been conjectured [9] that the following strategy will force the above algorithms to submit the most tests: the clone box tries to decode the media key with one of its device keys chosen uniformly at random. If the key chosen is enabled then the clone box successfully plays the test; if it disabled then it does not. That is,

$$p_T = \frac{|T \cap C|}{|C|}$$

We refer to this as the *uniform choice* strategy. The motivation for this strategy is that it is advantageous for the clone box to minimize the difference between p_a and p_b. However, the subset tracing procedure always recurses on the larger subinterval and so at every step the gap is reduced by, at most, a factor of 2. This gives us the following lower bound on the gap on the last iteration:

$$|p_a - p_{a+1}| \geq \frac{1}{2^{\log_2 |C|}} = \frac{1}{|C|}$$

This lower bound is achieved by the uniform choice strategy.

If we assume that the clone box implements this strategy, then its reasonable to ask if we can leverage this information to reduce the number of tests. In the next section we introduce *IGTrace* which does exactly this. We will see in Section 6 that this method offers a dramatic reduction in the testing time.

5 Subset Tracing with a Known Strategy

In this section, we will be more specific about what defines a *clone-box strategy*. As before, $\mathcal{F}$ and $\mathcal{C}$ denote the frontier and the set of keys in the clone box, respectively. Let $m = |\mathcal{F}|$ and $n = |\mathcal{C}|$. Denote by T the random variable associated with the outcome of a test or a set of enabled keys comprising a test. The intended interpretation should be clear from the context. Let R be the set of possible responses to a test. In previous sections the set of responses R was limited to the set $\{0, 1\}$; it either plays or it does not. The approach we present here can accommodate a richer set of responses, a possibility we will study more closely in the experimental section. Associated with each key $k \in \mathcal{F}$ is a binary random variable F_k; $F_k = 1$ is meant to imply that $k \in \mathcal{C}$. We refer to these

random variables, collectively, as F. In many contexts it is useful to treat F as the set of keys for which $F_k = 1$.

For our purposes, a strategy is a *conditional probability distribution* (CPD) specifying the probability the box plays a test conditioned on the fact that it contains a specific subset of keys. Formally, it specifies

$$Pr(T = t \mid F)$$

for all tests $T \subseteq \mathcal{F}$, all possible valuations of F, and all responses $t \in R$. The CPD is also subject to the same constraints as before

$$\text{if } T \cap F = \emptyset \text{ then } Pr(T = 0 \mid F) = 1$$

$$\text{if } T \cap F = F \text{ then } Pr(T = 0 \mid F) = 0$$

where the response 0 indicates the clone box did not play.

The question we would like to answer is: provided with a set of test-response pairs $T_1 = t_1, \ldots, T_N = t_N$, abbreviated as $\bar{T} = \bar{t}$, what is the posterior probability that the clone box contains a particular set of keys F. Applying Bayes rule we have:

$$\begin{aligned} Pr(F \mid \bar{T} = \bar{t}) &= \frac{Pr(\bar{T} = \bar{t} \mid F) Pr(F)}{Pr(\bar{T} = \bar{T})} \\ &= \frac{\prod_i Pr(T_i = t_i \mid F) Pr(F)}{Pr(\bar{T} = \bar{T})} \end{aligned} \tag{3}$$

where the second equality derives from the fact that the results of tests are independent conditioned on F. The goal of the subset tracing procedure is to find a key that is, with high probability, contained by the clone box. Formally, for some threshold ϵ, does there exist $k \in \mathcal{F}$ such that

$$Pr(F_k = 1 \mid \bar{T} = \bar{t}) = \sum_{F \subseteq \mathcal{F} : k \in F} Pr(F \mid \bar{T} = \bar{t}) > 1 - \epsilon$$

The term $Pr(F)$ specifies the prior probability that the clone box contains a set of keys F. Without any background knowledge we choose the uniform distribution. It is possible to embed domain specific knowledge in the prior to give the process a "head start". For example, if we knew that a particular subset of devices were more likely to have their keys compromised then we could increase the prior probability that the clone box contained one of those keys.

The denominator of Equation 3, $Pr(\bar{T} = \bar{t})$, is the marginal probability of observing the responses to the set of tests; it also acts as a normalizing constant and is defined as:

$$Pr(\bar{T} = \bar{t}) = \sum_{F \subseteq \mathcal{F}} Pr(\bar{T} = \bar{t} \mid F).$$

Note that the sum is taken over a set of size 2^m, making this a difficult quantity to calculate directly. Related work on adaptive diagnosis in distributed systems

approximate this quantity using the *mini-bucket* algorithm [12]. For our application, there exists methods for efficiently calculating $Pr(\bar{T} = \bar{t})$ exactly, but they are outside the scope of this paper [13].

There is an important difference between this approach and the previous algorithms: here, we directly model our belief that the clone box contains a key. In (Bayes)NNLTrace, this belief was indirectly modeled by the probability that the probability of tests playing were different. In a sense, there was no negative evidence. Two tests playing with different probabilities only supported the existence of certain keys in the clone box. Now, the response of the box can indicate that certain keys are *not* contained in the box. This is one reason why this new approach offers such an improvement over (Bayes)NNLTrace. The other advantage to explicitly modeling the clone box's contents is that we can make a more informed choice about the next test to submit to the clone box. The details of this are described in Section 5.1.

The entire procedure is specified in Algorithm 2. Note that this procedure returns, in addition to the compromised key, the updated beliefs about the clone box. This information can be propagated by the top-level traitor tracing algorithm to subsequent calls to the subset tracing procedure. This, too, can have a significant impact on performance, although we do not document it here.

Algorithm 2. Strategy based subset tracing procedure

```
1: IGTrace(F, Pr(F))
2: if response of clone-box to T = F is 0 then
3:     return [∅, Pr(F)] //in this case, C = ∅
4: loop
5:     for all k ∈ F do
6:         if Pr(F_k = 1) > 1 − ε then
7:             return [k, Pr(F)]
8:     select a test T(see Section 5.1)
9:     submit test to clone-box and get response t
10:    Pr(F) ← Pr(F | T = t)
```

5.1 Test Selection with Information Gain

We can imagine the testing process as a *decision tree* where every node specifies the next test to submit to the clone box. The response of the clone box dictates the subtree on which to recurse. Every leaf in the decision tree will have a key associated with it and reaching it in the recursion implies the associated key has been compromised. Ideally, we could minimized the expected number of tests needed by mapping out an entire testing strategy. However, this task was shown to be NP-Complete [7].

Instead of devising such a decision tree at once, we take a greedy approach to test selection. With a direct model of our beliefs about F, we can ask: how informative is a single test T? That is, how much would our uncertainty about

F decrease provided with a response by the clone-box to T? At every step we simply choose the test that maximizes the decrease in uncertainty.

The first step is to quantify our uncertainty about F. We will measure uncertainty as the *entropy*:

$$H(F \mid \bar{T} = \bar{t}) = -\sum_{F \subseteq \mathcal{F}} Pr(F \mid \bar{T} = \bar{t}) \log_2 Pr(F \mid \bar{T} = \bar{t})$$

Note that if we are certain of the contents of the clone-box then $H(F) = 0$. The entropy is maximized at m when the distribution of $Pr(F)$ is uniform. We measure the quality of a new test T as the *mutual information* between the T and F. In the machine learning literature it is often referred to as the *information gain* of T. It is defined as

$$I(F; T \mid \bar{T} = \bar{t}) = H(F \mid \bar{T} = \bar{t}) - \sum_{t \in R} Pr(T = t \mid \bar{T} = \bar{t}) H(F \mid \bar{T} = \bar{t}, T = t)$$

This is equal to the expected reduction in entropy by seeing the result of test T, taken with respect to the marginal probability of each response $t \in R$. For a more detailed explanation of entropy and mutual information, see, e.g., [10].

Now we can view test selection as solving the following optimization problem:

$$T^* = \text{argmax}_{T \subseteq \mathcal{F}} I(F; T \mid \bar{T} = \bar{t})$$

Of course, this obscures the fact that there are 2^m possible tests to consider. We know of no algorithm for efficiently solving this problem. Therefore, its necessary to approximate this by only considering a small subset of the possible tests. The simplest approach is to pick a set at random. Another approach is described in Algorithm 3. In this approach we maintain a set of tests $\mathcal{S}$ called the *retention set*. At every iteration we try adding a new key to each test in S, creating a new set of tests S'. We then update S to be the top s tests in S'. If after an iteration the retention set does not change, then we return the top test in S. The total number of tests evaluated by this procedure is $O(s^2 m^2)$. The proof is as follows: Denote by T_i^j the jth most informative test at iteration i. There are at most sm iterations since $|T_i^j| \leq m$ and at every iteration, at least one of the tests must increase in size. Let $\mathcal{T}_i^j = \{T_i^j \cup \{k\} : k \notin T_i^j\}$ be the set of tests generated in lines 6- 8. Clearly, $|\mathcal{T}_i^j| < m$ so at every iteration we need to evaluate at most sm tests.

Note that there are many alternatives to using entropy to measure uncertainty. For instance, the *Gini impurity*(a generalization of variance) and *misclassification impurity* are popular measures for learning decision trees [5]. We chose entropy because it has proven effective for selecting tests in adaptive diagnosis tasks. Rish, et al. [12] apply a method similar to ours to the task of diagnosing faults in distributed system. A similar approach has been used to play a probabilistic version of the game of Mastermind [8]. Mastermind is a popular board game created in the 70's where two players engage in a query-response loop. In the original formulation, game play is completely deterministic. The goal is for

Algorithm 3. Test selection procedure

```
1: InfoGainTestSelect
2: S ← {T = ∅}
3: S_old ← ∅
4: while S ≠ S_old do
5:    S_old ← S
6:    for all T ∈ S do
7:       for all K ∉ T do
8:          add T' = T ∪ {K} to S
9:    sort S by descending information gain
10:   remove all but the top s tests from S
11: return the first test in S
```

one player, the code-breaker, to identify the code put down by the code-maker. In many ways, it is similar to the task we are face with. We can imagine the clone box as a secret code that we are trying to reveal. In our game, though, we are only required to find one bit in the 'code' and the code-maker is under fewer constraints.

6 Experiments

In this section we present empirical work comparing the number of tests needed by the described methods. All experiments were repeated 20 times for random choices of compromised keys. In all experiments, we use $\epsilon = 0.001$ and $R = \{0, 1\}$ Preliminary experiments showed that the number of tests is not particularly sensitive to s, the retention set size, so in all experiments $s = 2$. We only report results for the uniform choice strategy since it is, theoretically, the most challenging strategy to defeat. Preliminary experiments also confirmed this was the case.

All experiments were run using Matlab running under Linux, with 3GB of memory and a 2Ghz Pentium 4 processor. The posterior distribution and information gain were calculated exactly. Both operations scale exponentially with the size of the frontier so we did not evaluate our method on very large frontiers. Elsewhere we describe an efficient algorithm [13], but it is beyond the scope of this paper. The code is optimized to take advantage of Matlab's efficient matrix operations and for a frontier of size 18, test selection takes around 45 seconds.

Table 1 highlights the results of the experiments. It is immediately clear that IGTrace outperforms both other methods in all cases where there is more than 1 compromised key. For larger frontiers and only 1 compromised key, BayesNNLTrace seems to need slightly fewer tests. But for larger numbers of compromised keys, which is of more interest to us, there is an order of magnitude improvement. Note, too, that there is substantially less variance in the number of tests needed by IGTrace.

BayesNNLTrace compares extremely well with NNLTrace in all cases, as well. One obvious shortcoming of NNLTrace is that it does not automatically take

Table 1. Comparative performance of the three tracing methods for a variety of frontier sizes and clone box sizes. Each entry lists the mean and standard deviation of the number of tests. Column headers indicate the number of compromised keys.

		1	2	4	8
$\|F\| = 8$	*IGTrace*	6±0.8	19±8.0	41±12.6	42±16.1
	BayesNNLTrace	22±3.1	109±19.9	539±168.5	823±231.4
	NNLTrace	271±0.0	605±181.5	919±423.8	1424±160.6
$\|F\| = 10$	*IGTrace*	13±2.0	24±8.4	52±11.1	86±38.7
	BayesNNLTrace	25±4.4	142±34.4	518±270.1	1173±438.1
	NNLTrace	380±54.7	948±299.2	1408±601.2	2632±607.4
$\|F\| = 12$	*IGTrace*	19±2.0	32±9.4	57±18.6	113±36.5
	BayesNNLTrace	28±4.6	148±41.9	784±221.4	1322±602.7
	NNLTrace	481±55.4	1263±415.1	1686±949.3	3562±1223.1
$\|F\| = 14$	*IGTrace*	29±1.0	38±9.5	64±14.7	129±26.1
	BayesNNLTrace	28±3.9	176±34.7	855±244.7	1510±624.5
	NNLTrace	555±51.4	1366±399.4	2326±1221.7	4506±1547.4
$\|F\| = 16$	*IGTrace*	38±1.1	47±9.0	68±14.0	159±41.6
	BayesNNLTrace	29±3.0	188±46.4	914±346.3	2258±923.2
	NNLTrace	637±0.0	1493±478.0	3660±1239.4	4278±1588.7
$\|F\| = 18$	*IGTrace*	49±1.6	57±7.3	86±15.9	150±30.3
	BayesNNLTrace	32±3.0	197±52.0	1203±320.4	1518±705.1
	NNLTrace	728±69.2	1845±488.9	4525±1457.6	6229±3199.4

advantage of situations where the clone box is deterministic. BayesNNLTrace, however, naturally accommodates this scenario, as seen in the first column, where the number of compromised keys is 1. BayesNNLTrace only repeats each test about 7 times.

7 Conclusion

Traitor tracing is an essential component in any broadcast encryption scheme. Previous algorithms, while theoretically efficient, do not offer a practical solution. In this work we have presented two methods, BayesNNLTrace and IGTrace, that significantly improve the number of tests required by the tracing algorithm to defeat a clone box.

In situations where some information about the clone box strategy is obtained, IGTrace can be applied. However, this begs the question: how does one obtain such information? In the future, it is worth investigating methods for learning this strategy along the way, perhaps by representing the clone box strategy as a latent variable.

Similarly, it would also be interesting to investigate the sensitivity of IGTrace to errors in the clone box strategy. We would like it to be such that if the actual strategy deviates from the modeled strategy we are still guaranteed to make accurate diagnoses.

References

1. 4Centity.
2. AACS.
3. A.Fiat and M. Naor. Broadcast encryption. In CRYPTO 93: International Cryptology Conference on Advances in Cryptology, volume 773, pages 480491, London, UK, 1993. Springer-Verlag.
4. A.Fiat B. Chor and M. Naor. Tracing traitors. In CRYPTO 94: International Cryptology Conference on Advances in Cryptology, volume 839, pages 480491, London, UK, 1994. Springer-Verlag.
5. R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification. Wiley-Interscience Publication, 2000.
6. Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. Bayesian Data Analysis, Second Edition. Chapman & Hall/CRC, July 2003.
7. Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. Inf. Process. Lett., 5(1):1517, 1976.
8. Vomlel J. Bayesian networks in mastermind. Proceedings of the 7th Czech-Japan Seminar on Data Analysis and Decision Making under Uncertainty., pages 185 190., 2004.
9. Jeff Lotspiech. Personal communication, 2006.
10. David J. C. MacKay. Information Theory, Inference, and Learning Algorithms. Cambridge University Press, 2003. available from http://www.inference.phy.cam.ac.uk/mackay/itila/.
11. Dalit Naor, Moni Naor, and Jeffrey B. Lotspiech. Revocation and tracing schemes for stateless receivers. In CRYPTO 01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, pages 4162, London, UK, 2001. Springer-Verlag.
12. I. Rish,M. Brodie, S.Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez. Adaptive diagnosis in distributed systems. IEEE Transactions on Neural Networks, 16:10881109, 2005.
13. Philip Zigoris. Improved traitor tracing algorithms. Technical report, UC Santa Cruz, 2006.