Balancing Performance and Confidentiality in Air Index

Qingzhao Tan[1] Wang-Chien Lee[1] Peng Liu[2] Dik-Lun Lee[3]

[1]Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA16801, USA {qtan,wlee}@cse.psu.edu

[2]Department of Information Science and Technology, Pennsylvania State University, University Park, PA16801, USA pliu@ist.psu.edu

[3]Department of Computer Science, Hong Kong University of Science and Technology, HK dlee@cs.ust.hk

ABSTRACT

The signature-based index technique, where a signature is generated for each data item as the retrieval index, is an efficient way for the mobile system to broadcast the data items. A lot of studies have been done for the performance issues, which is directly related to the access latency and the energy conservation for the mobile users to get their desired information. While there is few work done for the secure concerns, which is also an important issue since the system may concern whether the mobile attacker can get the information easily. This paper analyzes the signature technique for both the performance and the secure concerns. Since for the performance issues, the signature is to help user figure out the information efficiently, while for secure concerns, the signature is to prevent the hacker guessing the information easily, there is a trade-off between these two aspects. An administrator of the mobile broadcasting system should try to balance this trade-off by carefully designing the signature used in broadcasting. Our paper also provides an optimal signature design by choosing suitable parameters.

Keywords

Security, Indexing Techniques, Wireless Data Broadcast

1. INTRODUCTION

With the ever growing popularity of smart mobile devices and rapid advent of wireless technology, the vision of *pervasive information access* has come closer to a reality. Information consumers, including human beings, mobile devices, information appliances, and their applications, demand to access the much needed information from anywhere, anytime.

Today, there are many wireless technologies (e.g., Bluetooth, IEEE 802.11, UMTS, Satellite, etc) that could be integrated to construct a seamless, pervasive information access platform. Although their goals and applications are very different, information access via these wireless technologies can be logically captured by a basic model which consists of an access point (i.e., base station or satellite) and a number of wireless channels. In this model, a user may access information via two basic approaches:

- **On-demand Access.** Through an established point-to-point wireless channel, a user submits a request to the server. The server locates the appropriate data and returns it to the user.
- **Broadcast.** Data are broadcast on a wireless channel open to the public. A user tunes into the broadcast channel and filters out the data according to the request.

On-demand access employs a basic client-server model where the server is responsible for processing a query and returning the result directly to the user via a dedicate point-to-point channel. On the other hand, broadcast approach has the server actively pushing data to the users. The server determines the data and its schedule for broadcast. A user listens to a broadcast channel to retrieve data based on its queries and, thus, is responsible for query processing.

The on-demand access and broadcast can be employed individually or in combination (i.e., *on-demand broadcast*). While the pure broadcast approach does not provide an active channel for users to submit queries[1], the on-demand broadcast allows the users to submit queries via low-bandwidth uplink channels and receive the results from broadband, shared, broadcast channels. In such systems, a user is responsible for filtering out the result designated for her. The DirecWay and SpaceWay of Hughes Network Systems [1, 3, 2] are examples of on-demand broadcast systems.

In the on-demand broadcast system, the data item is broadcast only when an authorized user sends out a request. And the authorization is done when the mobile system receive the request so that the data is only sent to an authorized user. However, in a wireless data broadcast environment, any user with appropriate equipment can monitor the broadcast channel and log the data items being broadcast. If the broadcast data items are not encrypted, the broadcast data content is open to the public and any person can access them. Key-based encryption is a natural choice for ensuring secure access of data on air (i.e., only the subscribers who own the valid keys can decrypt the packets received to obtain

[1]The users can still establish their query profiles off-line that will serve as inputs to a broadcast scheduler.

the data items). Therefore, a mobile client has to receive broadcast data items off the air and decrypt them for filtering before they can be used. To help alleviate the high cost of receiving, decrypting and filtering broadcast data, auxiliary information may be provided on the broadcast channel to annotate the broadcast data items. This technique is called *air indexing*. The basic idea is that, based on index information (on indexed attribute values, arrival schedule, length of data items, etc) broadcast along with data items, mobile clients are able to selectively skip unauthorized or unwanted data items by slipping into *doze mode* and switching back to *active mode* only when the data of desire arrives. This technique, substantially reducing workload and energy consumption of mobile clients, is particularly important for encrypted data broadcast.

Nevertheless, while broadcasting the index information is important for performance, a security issue also arises, i.e., non-encrypted index information may allow unauthorized attackers to infer the data content on broadcast and therefore cause *confidentiality loss*.

Many studies have been reported in the literature to address the performance issues of wireless broadcast systems [9]. However, not much effort is put on the security problems. In this paper, we investigate the problem of *how to facilitate secure access of wireless disseminated data*. We look into a hash-based air indexing technique, called *simple signature*, which was proposed previously by the authors to address the issue of energy conservation. We show that the signature technique can avoid unprotected broadcast of index information and reduce the overhead of decrypting the index information at client side. While confidentiality loss cannot be completely eliminated by using the signature technique, we show that it can control the confidentiality loss to a low level without causing substantial performance penalty.

The main objective of our work is to balance the performance and the confidentiality in air index. Our specific contributions are stated as follows.

- We define several control parameters and metrics for the confidential issues in air index as well as for the performance.
- We try to find the linkage between the parameters and the metrics so that we can analyze the trade off between the metrics in terms of control parameters.
- By tuning the control parameters, we try to balance the trade off between the metrics of performance and confidentiality.

The rest of this paper will be organized as follows. In Section 2, we present some preliminaries of the technique we will study in our paper. In Section 3, we overview the whole scenario in our work. In Section 4, we analyze the false drop probability as well as the false guess probability. In Section 5, we analyze the performance and the security issues in secure mobile broadcasting. We give the concluding remarks in Section 6.

2. PRELIMINARIES

In this section, we first give an overview of the signature technique and its application in the mobile broadcasting system. Then we review some research work related to our paper.

2.1 Overview of the Signature Techniques

The signature techniques have been studied extensively in information retrieval [13]. A signature is basically an abstraction of the information in a data item, which contains a set of attributes. By examining the signature only, we can estimate whether the data item contains the desired information. Therefore, the signature technique is very suitable for quickly filtering a large set of data items.

There are a number of ways to generate a signature for a given data item. Let us consider a typical one. Given a data item, i, a signature, S_i, is formed by first hashing each attribute in the data item into a random bit string and then superimposing together all bit strings into the signature. Note that the size of the signature is the same as the size of each bit string. Therefore, no matter how many attributes are indexed into one data item, the data item's signature size will not change. During filtering, given a query, Q, a signature, S_Q, is constructed in the same way and then compared to S_i.

Obviously, there are two possible outcomes of the comparison:

- The data item doesn't match the query (i.e. $S_Q \bigwedge S_i \neq S_Q$).
- The data item matches the query; that is, for every bit set in the query signature, the corresponding bit in the data item's signature is also set (i.e. $S_Q \bigwedge S_i = S_Q$). Furthermore, this signature match has two possible implications:
  - *true match*, i.e., the data item is really what the query searches for;
  - *false drop*, i.e., even though the signature comparison indicates a match but the data item in fact does not match the search criteria.

To eliminate false drops, the record must be compared directly with the query after the record signature signifies a match. A data signature failing to match the query signature guarantees that the corresponding data item can be ignored. Figure 1 depicts the signature generation and comparison processes of a data item annotated with two attributes *Security* and *Pervasive*. In this example, the query search for data items containing the attribute *Hacker* does not match with the data item under comparison. The other two queries passed the signature comparisons but the data item needs to be retrieved again and directly evaluated with the queries to determine whether they are true matches or false drops. In our example, *Security* is the true match while *mobile* is a false match. Signature techniques are good at screening out unqualified records, such as the attribute *Hacker* in our example.

2.2 Signature-based Broadcast Scheme

Applied in indexing data in a broadcast environment, signature plays an important role to abstract a data item and to let clients filter their desired data items, thus improving tuning performance.

In a wireless data broadcast cycle, the signatures are constructed from data items and broadcast along with the data items. Intuitively, the signatures are broadcast before their corresponding data items. This is called *interleaved broadcast*.

The most straightforward approach to index the data items is to interleave a signature and a data item represented by

Data Item	Attr.1 : Security	Attr.2: Pervasive
Security	001 100 001 001	
Pervasive	101 000 100 001	
Data Signature (V)	101 100 101 001	

(a) Signature Generation

Search	Search Signatures	Results
Hacker	000 101 000 101	No Match
Security	001 100 101 000	True Match
Mobile	100 100 001 001	False Drop

(b) Signature Comparison

Figure 1: Signature Generation and Comparison.

the signature. This is called the *simple signature*. Figure 2 shows its structure.

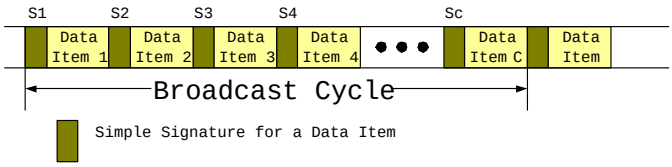


Figure 2: Simple Signature Scheme

Suppose a mobile client tunes in the broadcast cycle to retrieve information based on her query, Q. A query signature S_Q is then generated based on Q to scan and compare with the data items' signatures. When a received data item signature does not match with the query signature, the mobile client will switch into doze mode until the next signature arrives. If most of the signatures don't match with the query signature, the mobile device will stay in doze mode for the most part of a broadcast cycle, thus saving a lot of energy. When a match is found, the data item following the signature is received for further examination.

Recall that it is possible to have false drops. In this case, the client needs to perform further checking. If the data item is not a false drop, it will be retained in the results set. Otherwise, the data item is discarded. In traditional mobile broadcast system, frequent false drops consume the client's energy since the client have to stay in active mode for the false drops although they are unqualified.

However, in secure air index, the false drops may make the data item safer. Given a data item along with the corresponding signature, a hacker generates several queries with their corresponding signatures and compares them to the data item's signature to guess what are the attributes inside the data item. Similar to the false drops, the hacker will get some false guesses. With these false guesses, the hacker cannot obtain the accurate attributes in the data item. Thus leading to the problem we will analyze in our paper.

2.3 Related Work

There is a number of proposals on air indexing. The goal of air indexing is to conserve battery power for mobile clients. Several item-based indexing techniques, such as flexible indexing and distributed tree indexing, for broadcast channels were first introduced by Imielinski et al. [10, 11]. Based on the index tree method, work presented in [5, 16] investigated unbalanced indexes to improve performance for skewed data access. However, these studies concentrated on one-dimensional indexes for equality-based queries, and thus are inapplicable to location-dependent query processing where point queries are involved. Lee et al. recently have addressed general queries with a semantics-based broadcast approach [14]. Tan and Yu have developed a broadcast program that supports range queries [17]. Traditional index techniques, such as hashing [11] and signature file [7], were also applied in on-air storage, along with hybrid approach [8]. Besides the design of different indexing structures for different scenarios, index organization algorithms were also studied [12].

However, none of the above techniques addresses any security problems. There are also some secure wireless broadcast technique proposed. Some of them are based on secure multicast group key management in the network [15]. Others use broadcast encryption schemes to disseminate a secret to only the privileged users via broadcast channels [4]. These techniques properly encrypt the data item so that the attacker could not know the exact content of the data items broadcasted. Nevertheless, the signatures cannot be encrypted, otherwise data items cannot be differentiated by the client. In our approach, we consider take advantage of the signature-based index itself to balance the trade-off between the performance and security.

3. PROBLEM DESCRIPTION

Besides the system administrator, there are two types of potential listeners that we are interested in this work. They are the authorized clients and the unauthorized attackers.

From the standpoint of authorized clients, two crucial issues are energy efficiency and access latency. They listen to the broadcast, download identified data items based on matched index and extract required information after decrypting the downloaded data items. To those legal mobile clients, a good broadcast program should be precise enough that they do not need to download unnecessary data block for energy saving, that means the index scheme should have a high true match rate.

On the other hand, unauthorized *attackers* (or called *hackers*) eavesdrop the broadcast. They scan the broadcast, download possible data items, and try to guess the content encrypted in the downloaded data items. Hence, when an attacker downloads a signature from the broadcast channel, she might start a *dictionary attack*. She uses all the attributes in her dictionary (denoted by D_H) to generate $|D_H|$ signatures and compare each of them with the downloaded signature. Assuming that the attacker's dictionary is comprehensive, she will find a set of matches in her dictionary. Among those matches, there are *correct guesses* and *false guesses*.

Finally, the administrator wants to meet the users' performance requirements and avoid confidentiality loss to the attackers without introducing too much resource overhead (e.g., bandwidth). Thus, an important job of the the system administrator is to facilitate highly energy-efficient data access to the authorized users (maybe with a cost of some small access latency delay and bandwidth overhead), while minimizing confidentiality loss through the index. A key challenge for the system administrator is to determine con-

fidentiality loss by answering "how much information has been leaked to attackers?" It's important for system administrators to be able to estimate and minimize *information leakage*. Comes along are further concerns of the administrators, e.g., "do we have a way to control the information leaking?", and "how much performance will deteriorate if the security is to be tightened?" It is also crucial for the administrator to choose or/and fine-tune a suitable indexing scheme that meets various application and security requirements. Thus, planning and tuning tools that give a better picture of the system performance are needed. For example, a control panel (as depicted in Figure 3) that allows the administrators to estimate various performance and security metrics by setting a few buttons and turning the control knobs will be handy. To answer these critical questions and support such tools, we conduct in-depth performance and security analysis on the signature air indexes in this paper. In the following, we first describe the metrics and control parameters and then provide an overview of the analysis.

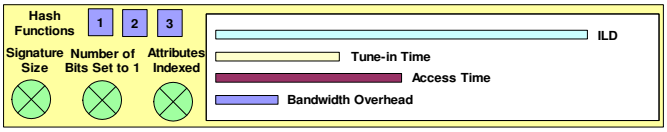


Figure 3: A Control Panel

3.1 Performance and Security Metrics

As mentioned earlier, the users have concerns about access latency and energy consumption, which are typically measured by *access time* and *tune-in time*, respectively. In addition to the above, the system administrator has to control the use of scarce wireless bandwidth (measured by *bandwidth overhead*) and minimize the information leakage (defined by a new security metric, *information leaking degree*).

- **Access time.** The period of time from the moment a query is issued until all the qualified data items are received.
- **Tune-in time.** The time required for mobile clients to stay in the active mode in order to answer a query.
- **Bandwidth overhead.** The bandwidth consumed on broadcasting the auxiliary index information (i.e., signatures). Obviously, it is closely related to the size of the signature.
- **Information leaking degree (ILD).** The average number of correct guess among all the matched guesses obtained by an attacker. Intuitively, this depends not only on the deployed air indexing schemes, but also the size and the quality of dictionaries used by the attackers.

Note that we use the ratio between the tune-in time and the whole broadcast cycle to represent the tune-in time. And it is the same for the bandwidth overhead. Therefore, all of these metrics except the access time, which is represented by bits, can be computed in percentage.

3.2 Control Parameters

As shown in the control panel (see Figure 3), there are a few parameters system administrators can use to tune the filtering power of the signatures:

- **Hash functions.** While existing research typically assume a uniform distributed hash function is used, different hash functions in combination with other parameter settings have an impact on *hash collisions* (i.e., two or more different values hashed into the same bit string), which has an impact on both performance and security metrics.
- **Signature size.** This is the number of bits in one signature (denoted as m). In addition to the obvious impact on bandwidth overhead, it also has an impact on all the other metrics.
- **Bit setting.** The number of 1's set in a bit string (denoted by w_b) is related to tune-in time and ILD.
- **Attributes indexed.** The number of attributes in a data item selected for indexing. A data signature is superimposed from the bit strings of these attributes in the data item. This parameter is application dependent, but can still be adapted based on system needs.

In traditional information retrieval applications, m is set to a large value and w_b are carefully selected to provide a large space of hashed bit strings in order to prevent hash collisions. As such, the impact of hash collision can be minimized. However, for secure wireless data broadcast systems, using a large signature size is not necessarily a good idea due to bandwidth overhead and security concern. Thus, it's important for administrators to set these control parameters properly.

3.3 Methodology

The goal of this study is to make clear the relations between performance and security requirements with respect to the simple signature air indexing technique for wireless data broadcast systems. Moreover, we want to figure out proper settings of control parameters to balance these two sets of requirements. As a first step, we conduct an analytical study towards our goals.

Our approach is to derive cost models of performance and security metric in terms of the control parameters. Based on our observation, *false drop* (for normal users) and *false guess* (for attackers) play important roles in performance and security of the system. Thus, in the next section, we first analyze the *false drop probability*, which is the probability that a mobile user thought a data item is qualified with his query but actually it is not; and *false guess probability*, which is the probability that a dictionary value showing a match with an attacked data item but actually it is not. These probabilities represent a linkage between the control parameters and the performance and security metrics. In other words, the parameter settings can control the false drop probability and false guess probability, which in turn can determine tune-in time, access time, bandwidth overhead and ILD. Thus, the cost model will be derived in the subsequent section.

Furthermore, based on the derived model, we will perform a series of analysis to show the relationship between the performance and security metrics and determine proper configuration under various system settings. For example, we can fix *bandwidth overhead* and *access Time* and analyze the tradeoff between *ILD* and *tune-in time* by varying related parameters. For the control panel shown in Figure **??**, the first step is to choose the hash function used in the signature scheme. Second, the administrator would like to set

the knobs, signature size and attributes indexed. Finally, the knob of bits set is tuned from 1 to m. Then the change and the comparison are shown on the screen of the panel. Our study, not performed previously, reveal important and practical insight on design, deployment and administration of the secure wireless data broadcast systems.

4. ANALYSIS OF FALSE DROP AND FALSE GUESS PROBABILITIES

In this section, we will analyze two probabilities, false drop probability and false guess probability.

Given an application denoted by A, finite u attributes are used to generate a signature to index a data item in the data broadcast cycle. Without loss of generality, we assume the hash function used to generate the signature is uniform. We use D_A to denote the combined domain of these u attributes. So u elements in D_A are indexed to generate a valid signature. Totally there are $|D_A|$ attributes. Here we summarize the notation of symbols and parameters used in our analysis in Table 1.

A	the application;
D_A	the combined domain in A;
D_H	the hacker's dictionary;
C	number of data items in a broadcast cycle;
C_f	number of data items received due to false drops;
C_t	number of data items received due to true drops;
G	number of values received from the hacker's dictionary;
G_f	number of values received due to false guesses;
G_t	number of values received due to true guesses;
n	average number of bits in a data value;
u	number of values in a data item;
m	number of bits in a signature;
w_b	number of 1's in an attribute's signature;
w_f	average number of 1's in a data item's signature;
w'_f	average number of 0's in a data item's signature ($w_f + w'_f = m$);
P_f	false drop probability;
P_g	false guess probability;
P_s	selectivity of a query;
$overlap$	the part of overlap between a hacker's dictionary, D_H, and the system's combined domain, D_A.

Table 1: Symbols and Parameters.

4.1 The False Drop Probability

Semantically, false drop probability means the probability that the signature of a data item matches with the query signature, yet the data item does not satisfy the query. Thus, the false drop probability can be *experimentally* obtained as follows. Given a query Q, a query signature S_Q is generated to match with all data signatures. Let C be the numbers of signatures compared (in mobile broadcast system, C is the number of signatures in a broadcast cycle), C_t be the true matches, C_f be the false drops, and C_m be the signatures that don't match the query. The relation for these parameters can be shown as:

$$C = C_t + C_f + C_m$$

Then the false drop probability is:

$$P_{f,experiment} = \frac{C_f}{C - C_t}$$

Two possible reasons for causing false drops are 1) hash collision and 2) superimpose operation. First, let's analyze the potential false drops caused by hash collisions, which is denoted by $P_{f,collision}$. We define the collision factor of the hash function as the average number of different inputs hashed into the same output because of hash collision. Given a signature scheme which is to generate a bit string for an attribute under the application A, its collision factor is denoted as $CF_{A,bstr}$. That means $CF_{A,bstr}$ attributes falls into the same bit string. Similar to $CF_{A,bstr}$, the collision factor for a signature in application A (denoted by $CF_{A,sig}$) can be defined as the average number of data items hashed into the same signature. Because there are u bit strings used in a signature, there are at least $(CF_{A,bstr})^u$ data items collide into a signature. Given the average number of 1's in a signature (denoted by w_f) which can be derived in terms of u, m and w_b, there can be as many as $(CF_{A,bstr})^{\mathsf{comb}(w_f,w_b)}$ data items colliding into a signature. Here $\mathsf{comb}(\cdot,\cdot)$ is the binomial function and $\mathsf{comb}(w_f, w_b)$ is the number of bit strings that possibly contribute to the signature. Thus, we have $(CF_{A,bstr})^{\mathsf{comb}(w_f,w_b)} \geq CF_{A,sig} \geq (CF_{A,bstr})^u$. Based on $CF_{A,sig}$, we can derive the false drop probability caused by hash collisions as follows.

$$\begin{aligned} P_{f,collision(D_A)} &= \frac{\mathsf{comb}(|D_A|-1,u-1)\cdot(CF_{A,sig}-1)}{\mathsf{comb}(|D_A|,u)-\mathsf{comb}(|D_A|-1,u-1)} \\ &= \frac{u(CF_{A,sig}-1)}{|D_A|-u} \end{aligned} \quad (1)$$

Obviously, the collision factor has a great impact on the $P_{f,collision}$. So one problem is to find out the collision factor, $CF_{A,bstr}$, for the mobile system. We try several simulation experiments based on the parameters set in Table 2. In our simulation, m is fixed since the system administrator knows the system's limitation of the bandwidth overhead. Then w_b is tuned to see the change of the collision factors. We find out that $CF_{A,bstr}$ is larger than one only when w_b is very small (closed to 1) or very large (closed to m). In other cases, it is equal to one. We use the lower bound of $CF_{A,sig}$, which is equal to $(CF_{A,bstr})^u$, to generate $P_{f,collision(D_A)}$.

$D_A = 1000$	$D_H = 10000$
$u = 10$	$overlap = 0.01$
$C = 10000$	$P_s = 0.01$
$n = 128$	

Table 2: Parameters used in the analysis.

Next, we analyze the false drop probability caused by superimposition, which is denoted by $P_{f,superimposition}$. Since the false drop probability can be estimated assuming unsuccessful search [6], a false drop occurs when each of the bits in the data signature corresponding to the 1's in the query signature is set to 1. Let α_i and β_i be the i-th bit of the query signature and a data signature, respectively. A false drop occurs when the following condition holds: "if $\alpha_i = 1$ then $\beta_i = 1, 1 \leq i \leq m$" or when "if $\beta_i = 0$ then $\alpha_i = 0, 1 \leq i \leq m$". Thus, we have

$$P_{f,superimposition} = \frac{\mathsf{comb}(w_f, w_b)}{\mathsf{comb}(m, w_b)} \approx (1 - e^{-\frac{w_b u}{m}})^{w_b} \quad (2)$$

Assuming the false drops caused by hash collision and superimposition are independent, we have

$$\begin{aligned} P_f = P_{f,superimposition} + P_{f,collision(D_A)} \\ -P_{f,superimposition} \cdot P_{f,collision(D_A)} \end{aligned} \quad (3)$$

Finally, we plot the curves of the false drop probabilities

in figure 4(a). Figure 4(a) shows the trend of the false drop probability when w_b is increased under different bandwidth overhead, i.e., different ms. From this figure we can see that the false drop probabilities are large at the beginning due to the hash collision. Then they decrease since the hash collision also decreases. They increase again because of the superimposition. We can also see that when m is getting larger, P_f would increase slower. This is because when m increases, the P_f would be smaller.

4.2 The False Guess Probability

Similar to false drop probability, *false guess probability* (from the attacker's view) can be defined as the probability that a dictionary value which does not actually in the attacked data item but showing a match. The false guess probability, used to estimate the false guesses generated by a dictionary against a signature, can be *experimentally* obtained as follows. Given a dictionary D_H, a sequence of single value queries Q_i and corresponding query signature S_{Q_i} (where $i = 1..|D_H|$) are generated to match an attacked data signature. Let G be the total number of matched guesses, G_t be the correct guess, and G_f be the false guesses (i.e., $G = G_t + G_f$).

$$P_{g,experiment} = \frac{G_f}{|D_H| - G_t}$$

While the concepts of false guess probability and false drop probability are closely related, their definitions and semantics are different. Unlike P_f which depends only on the application data domain and signature generation scheme used, P_g also needs to factor in the hash collisions resulted from the dictionary, D_H, which may be much larger than D_A. The collision factor associated with D_H (denoted by CF_H) is thus likely to be much larger than $CF_{A,bstr}$. For each true guess, there are $CF_H - 1$ false matches, respectively. Therefore, the false guess probability caused by the hash collision of D_H is

$$P_{g,collision(D_H)} = \frac{G_t \cdot (CF_H - 1)}{|D_H| - G_t} \quad (4)$$

Again, we use simulation experiments to get the approximate CF_Hs.

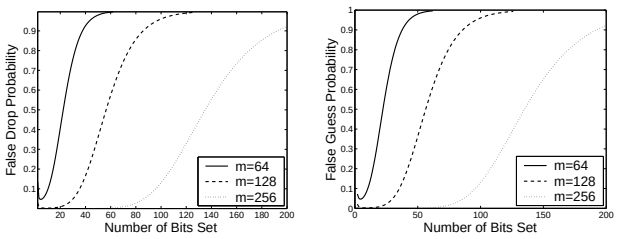
The false guesses caused by superimposition and hash collisions in D_H are assumed to be the same as the false drops. Thus, the total false guess probability is

$$P_g = P_{g,collision(D_H)} + P_{f,superimposition} - P_{g,collision(D_H)} \cdot P_{f,superimposition} \quad (5)$$

Figure 4(b) shows the trend of the false guess probability when w_b is increased under different bandwidth overhead. From this figure we can see that the trend of the false guess probability is similar with that of the false drop probability.

4.3 Observations

From the Formula (4) and (6), we can get the following observations. First, $P_{d,superimposition}$ is related to the size of the signature, the average number of 1's in the signature, and the number of signatures superimposed into one signature for the frame. Therefore, by tuning these parameters, the system can also tune the $P_{d,superimposition}$. More specifically, the $P_{d,superimposition}$ would be increased if we decrease m or increase u. Obviously, when m is decreased or u is increased, the percentage of the 1's in the signature increase.



(a) False drop probability vs Bits set (b) False guess probability vs Bits set

Figure 4: False drop probability and False guess probability vs Bits set

The superimposed signature can match more words than it originally can. Then there would be more false matches, thus increasing the false drop probability. However, the tuning of w_b is more complicated. The analysis is similar for the case of $P_{g,superimposition}$.

Second, $P_{d,collision}$ is related to the size of the system's dictionary, D_A, the number of words superimposed into one signature, u, and the collision factor, $CF_{A,sig}$. Intuitively, the larger the collision factor, the larger the $P_{d,collision}$. The changes of D_A and u can tune the number of matches and the total number of frames the system can generate. $P_{d,collision}$ should be made as small as possible. Sometimes in traditional IR, it can even be ignored when we assume the hash function is ideal. Then in that case, the false drop probability is caused only by superimposition.

Third, $P_{g,collision}$ cannot be ignored since the attacker's dictionary is much larger than the system's combined domain. Using a hash function that is ideal for a smaller input size, there would be more collisions when using a much larger input. Fortunately, this is the advantage we can take good use since it can hold the attacker's work back. The attacker would get more matches. Then it will be much harder for him to guess the right answer from some many matches.

5. COST MODEL

Based on the two probabilities analyzed in the last section, now we derive the the cost model for our performance metrics (i.e., *access time*, *tuning time*, and *bandwidth overhead*) and security metric (i.e., *ILD*).

5.1 Performance Metrics

Given a general query which may return multiple data items in the result set, the mobile clients typically need to go through a whole broadcast cycle in order to complete the query processing. Thus, the access time is mainly determined by the size of a broadcast cycle which in turn is affected by the size of the signatures. In addition, the access time needs to take into account the *initial probe time* which is the period of time from the moment a user tunes in until the first signature is received. In the simple signature scheme, it is half of the average size of a data item and its signature item. Thus, the access time is the sum of the initial probe time and the broadcast cycle. Using the number of bits visited as the physical unit, the following shows the access time for the simple signature scheme:

$$\begin{aligned} ACCESS &= PROBE + CYCLE \\ &= \frac{m+n}{2} + C \cdot (m+n) = (C + 0.5) \cdot (m+n) \end{aligned} \quad (6)$$

On the other hand, the tune-in time is basically the initial prob time plus the time used to scan the signatures, the true drops, and the false drops. Let PT denote the period in which the mobile device is active during the initial prob time and SIG denote the total number of data signatures in a broadcast cycle. The tune-in time for the simple signature scheme can be defined as follows.

$$\begin{aligned} TUNE = &\quad PT + SIG + C_t \cdot n + C_f \cdot n \\ = &\quad \tfrac{m^2+n^2}{2(m+n)} + C \cdot m + C \cdot n \cdot P_s \\ + &\quad C \cdot n \cdot P_d - C \cdot n \cdot P_s \cdot P_d \end{aligned} \tag{7}$$

Finally, the bandwidth overhead is dependent on the size of the signature, m, and the size of a broadcast cycle, C. Thus, we have:

$$Bandwidth\ Overhead = C \cdot m \tag{8}$$

We assume that C and n are fixed in a mobile system. Then the administrator can set the bandwidth overhead according to the hardware limitation. Both m and the access time are set once the bandwidth overhead is set. They are shown in the Table 3. From this table, we can see that both metrics increase when m increases. Based on this relation, the administrator can tune the access time according to the mobile system's bandwidth ability.

But the tune-in time also depends on the change of w_b. In order to observe the tune-in time more clearly, we tune w_b from 1 to m. The results are shown in Figure 5. We set the y-axis as the ratio between the actual tune-in time and the size of the whole broadcast cycle in bits. Based on this figure, we can see that the trend of tune-in time is similar to that of the P_d. That means the tune-in time, thus the power conservation, is affected mainly by P_d. This is consistent with the fact that larger P_d leads to more false drops and then the mobile clients would waste more energy. From this point of view, the mobile system's administrator should try to minimize P_d to save the mobile client's energy.

m	Bandwidth overhead (%)	Access time (10^6 bits)
64	33.3	1.92
128	50.0	2.56
256	66.7	3.84

Table 3: Access time and Bandwidth overhead

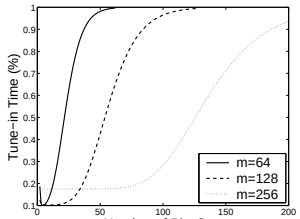


Figure 5: Tune-in time vs Bits set

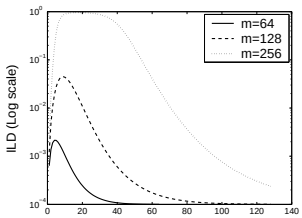


Figure 6: ILD vs Bits set

5.2 Security Metric

We now derive the cost model for ILD, which is the percentage of the information that is guessed correctly by the attacker. Suppose the attacker finds G values in his dictionary matching the signature and that G_t attributes are in the attacker's dictionary. Obviously, ILD is affected by the *false guess probability*, P_g. We can represent ILD in terms of the false guess probability as:

$$ILD = \frac{G_t}{G} = \frac{G_t}{G_t + P_g \cdot (|D_H| - G_t)} \tag{9}$$

In the above, $P_g \cdot (|D_H| - G_t)$ is the number of false guesses. This equation reveals some interesting insights. When $P_g = 0$, the information leaking degree is 100%. To reduce information leaking to 50%, P_g needs to be raised up to $G_t/(|D_H| - G_t)$. This finding indicates a dependency between information leaking degree and the false guess probability and points out an observation, i.e., having a reasonable false drop probability is not such a bad idea for security reasons, even though low false drop probability is preferred from the performance perspective.

Now we assume that the attributes in the broadcast cycle are uniformly distributed. So once the overlap between the combined domain and the hacker's dictionary is set, the number of true guesses is fixed. That is,

$$G_t = u \cdot \frac{overlap}{A}$$

Here, u is the number of bit strings superimposed into a signature.

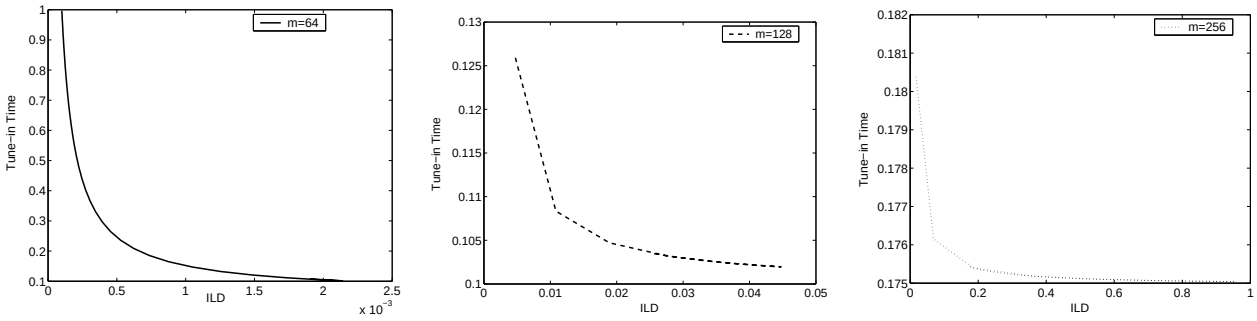
Figure 6 shows the trend of ILD when w_b increases under different bandwidth overheads. Basically, the ILD increases to the maximal and then decreases again as w_b increases. The trend of ILD is reversed to the that of P_g. This further shows that having a false guess probability, which leads to a false drop probability, have benefit on the security issues, though the false drop probability may affect the mobile system's performance.

5.3 Relating Performance Metrics to Security Metrics

We have got the formula for the metrics and the probabilities. Obviously, in order to optimize the performance metrics, P_d should be as small as possible. In order to optimize the security metric, P_g should be as large as possible. However, both P_d and P_g would change proportional to the change of w_b. Now we try to find the relation between the performance metric, tuning time, and the security metric, ILD according to the change of w_b.

From figure 7, we can get the conclusion that there is a trade off between the tune-in time and ILD. They change reversely when w_b increases. In this case, the administrator can try to find the optimal case with minimal tune-in time and minimal ILD.

We can also see that the optimal cases for the tune-in time. The w_bs corresponding to the minimal tune-in time and minimal ILD of three different bandwidths are listed in Table 4. The table show the best choices for the parameters used. Although Table 4 gives us the best choices of w_b for three bandwidths, it is not suitable for every application. For example, some people may concern much more on security metrics than performance metrics. They may be willing to spend some more tune-in time to get less confidentiality loss. In such case, the mobile system can tune the control parameters on the control panel and get the suitable signature design to satisfy users' different preferences.



(a) $m = 64$ (b) $m = 128$ (c) $m = 256$

Figure 7: ILD vs Tune-in time

Signature Size	$m = 64$	$m = 128$	$m = 256$
w_b	15	3	4
ILD (%)	0.00045	0.0108	0.1859
Tune-in Time (%)	0.2648	0.1083	0.1754

Table 4: Optimal Parameters

6. CONCLUSION

The technique of indexing is very effective in mobile broadcast system since most queries select only a small number of data items. And the signature scheme is excellent for filtering the information in the broadcast cycle.

We argue in this paper that not only performance issues but also security issues should be taken into consideration in mobile broadcast. In such case, unlike the traditional signature technique that is evaluated using the performance metrics, such as access time, tune-in time, and bandwidth overhead, the signature should also be evaluated using the security metric, such as ILD. In our paper, the cost models for these metrics have been developed and compared based on various factors.

The results show that there is a trade off between the performance and the security issues. By tuning the control parameters, such as the size of the signature and the number of bits set to 1 in the signature, we can observe the optimal design of a signature scheme with the minimal energy consumption, which can be indicated by tune-in time, and minimal confidentiality loss, which can be indicated by the information leakage degree.

As for the future work, we are trying to do some further analysis in terms of different schemes in the mobile broadcasting. Also, we will do some more simulation to evaluate our analysis. And another practical research direction is to find out some new signature schemes so that the system can balance the tradeoff between the performance issues and the security concerns more effectively.

7. REFERENCES

[1] Direcway. Website at http://www.direcway.com/.

[2] Hughes network system. direcway/spaceway. Website at http://www.hns.com.

[3] Spaceway. Website at http://www.spaceway.com/.

[4] S. Berkovit. How to broadcast a secret. In *Proc. of Eurocrypt'91*, 1991.

[5] M.S. Chen, P. S. Yu, and K.L. Wu. Indexed sequential data broadcasting in wireless mobile computing. In *ICDCS '97: Proc. of the 17th ICDCS*, page 124. IEEE Computer Society, 1997.

[6] C. Faloutsos and S. Christodoulakis. Signature files: An access method for documents and its analytical performance evaluation. *ACM Trans. of Information Systems*, 2(4):267–288, 1984.

[7] Q. Hu, W.C. Lee, and D.L. Lee. Indexing techniques for wireless data broadcast under data clustering and scheduling. In *CIKM '99: Proc. of the 8th ICIKM*, pages 351–358, 1999.

[8] Q. Hu, W.C. Lee, and D.L. Lee. A hybrid index technique for power efficient data broadcast. *Distrib. Parallel Databases*, 9(2):151–177, 2001.

[9] Q. L. Hu, D. L. Lee, and W.C. Lee. A comparison of indexing methods for data broadcast on the air. In *Proc. of the 12th ICOIN*, pages 656–659, January 1998.

[10] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Energy efficient indexing on air. pages 25–36, 1994.

[11] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Power efficient filtering of data on air. In *EDBT '94: Proc. of the 4th ICEDT*, pages 245–258. Springer-Verlag New York, Inc., 1994.

[12] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Data on air: Organization and access. *Knowledge and Data Engineering*, 9(3):353–372, 1997.

[13] D. L. Lee and C.W. Leng. A partitioned signature file structure for multiattribute and text retrieval. In *Proc. of the 6th ICDE*, pages 389–397. IEEE Computer Society, 1990.

[14] K. Lee, H. V. Leong, and A. Si. A semantic broadcast scheme for a mobile environment based on dynamic chunking. In *ICDCS '00: Proc. of the The 20th ICDCS*, page 522. IEEE Computer Society, 2000.

[15] S. Mittra. A framework for scalable secure multicasting. In *Proc. of ACM SIGCOMM*, 1997.

[16] N.Shivakumar and S.Venkatasubramanian. Energy efficient indexing for information dissemination in wireless systems. *ACM-Baltzer Journal of NOMAD*, 1995.

[17] K.L. Tan and J. X. Yu. Generating broadcast programs that support range queries. *IEEE TKDE*, 10(4):668–672, 1998.