DaTA – *Da*ta-*T*ransparent *A*uthentication Without Communication Overhead

[1]Songqing Chen, [2] Shiping Chen, [3]Xinyuan Wang, and [2]Sushil Jajodia

Dept. of Computer Science
George Mason University
sqchen@cs.gmu.edu

Center for Secure Information Systems
George Mason University
{schen3, jajodia}@gmu.edu

Dept. of Information and Software Engineering
George Mason University
xwangc@gmu.edu

***Abstract*— With the development of Internet computing techniques, continuous data streams from remote sites are commonly used in scientific and commercial applications. Correspondingly, there is increasing demand of assuring the integrity and authenticity of received data streams. Existing strategies of assuring data integrity and authenticity mainly use message authentication codes (MAC) generated on data blocks and transfer the MAC to the receiver for authentication through either out of band communication or in band communication. Transferring the MAC via out of band communication inevitably introduces communication overhead and additional complexity to synchronize the out of band communication with the data communication. Transferring the MAC via in band channel can be achieved by either appending the MAC to the original data or embedding the MAC into the original data, which would either incur communication overhead or change the original data. It would be desirable to be able to authenticate the stream data without any communication overhead and changing the original data at the same time. To deal with data packet or block loss, many of existing stream data authentication schemes rely on hash chaining, the current usage of which results in uncertainty in authenticating the subsequent data blocks once the first data packet or block loss is detected.**

In this paper, we propose a novel application layer authentication strategy called *DaTA*. This authentication scheme requires no change to the original data and causes no additional communication overhead. In addition, it can continue authenticating the rest of data stream even if some data loss has been detected. Our analysis shows that our authentication scheme is robust against packet loss and network jitter. We have implemented a prototype system to evaluate its performance. Our empirical results show that our proposed scheme is efficient and practical under various network conditions.

I. Introduction

The proliferation of the Internet has enabled more and more scientific and commercial applications to use continuous data streams sent from remote sites. For example, hurricane forecasting [1] demands continuous and timely data input from many public data archives (such as satellite images and oceanic changes from NOAA [2]) to forecast the hurricane movement accurately. ADVFN [3] streams real-time stock prices, price data, and stock charts from the New York Stock Exchange to subscribed users. Audio and video based media communications (e.g., tele-conferencing, remote education, and entertainment) are becoming ubiquitous today in our everyday life [4]. In many such network applications, the data integrity and authenticity need to be assured. For example, the receivers of NOAA oceanic data want to make sure that the received data is genuine and has not been tampered with. Subscribers to the ADVFN real-time stock streaming services need to be certain that the received stock prices are from ADVFN and they have not been modified during the transmission. In addition to the capability to detect any unauthorized change over the transmitted content, the receivers of the continuous data stream want to be able to detect any packet loss during the transmission. In such applications, confidentiality is not required as the data is meant to be public. Therefore, the continuous data stream needs to be authenticated without encryption.

Existing methods of assuring data integrity and authenticity use message authentication codes (MAC) generated through HMAC or NMAC or one-way hash functions on a block of data [5], [6], [7], [8], [9], [10], [11]. To transfer the MAC to the receiver for authentication, generally there are two approaches: via out of band or in band communication. Obviously transferring the MAC via out of band channel introduces additional communication overhead and computation complexity (i.e., synchronization between the out of band communication and the in band communication). To transfer the MAC via in band channel, the MAC can be either appended to or embedded into the original data. Embedding is to replace some bits in the original data that can eliminate the communication overhead. Thus, the in band communication for MAC either introduces additional overhead or changes the original data. It would be desirable to be able to authenticate the stream data without introducing any additional communication overhead nor changing the original data content.

To be able to detect any lost packet or data block, existing stream data authentication schemes mainly rely on hash chaining when generating the MAC. Most of these schemes [12], [13], [8] have no problem in detecting the first packet or block loss. But once the first block or packet loss is detected, it will result in uncertainty for them to continue authenticating the rest data stream. It would be desirable to be capable of detecting all packet or block loss and to continue the stream data authentication after detecting any packet or block loss.

In this paper, we propose a novel application layer stream data authentication method, *DaTA* (*Da*ta-*T*ransparent *A*uthentication without Communication Overhead), to authenticate data streams. Unlike all previous stream data authentication schemes, our *DaTA* scheme embeds the MAC into the inter-packet timing of the data stream, which does not intro-

duce any communication overhead[1] nor change the original stream content. By careful design of the MAC on the data stream, our *DaTA* approach is able to detect multiple packet or block loss, and it can continue authenticating the rest data stream even if a packet (or block) loss has been detected. This is in contrast to previous data stream authentication approaches [12], [13], [8].

Besides analyzing the quantitative tradeoffs between the authentication effectiveness and the bit error probability, we have implemented a prototype of *DaTA* and we have empirically evaluated the performance of our transparent stream data authentication scheme in a LAN and on the Internet. As with any other stream data authentication schemes, our authentication scheme is subject to malicious attacks such as packet drops and packet content modification. Our experimental results demonstrate that with appropriate redundancy, our *DaTA* scheme is robust against packet loss and network jitter, and it is able to detect packet loss at the granularity of block. Our experiments also shows that *DaTA* does not introduce any noticeable impact over the data stream application.

The rest of the paper is organized as follows. Section II describes related work. Section III presents our proposed scheme and Section IV analyzes the accuracy of the proposed scheme. Design issues are further discussed in Section V. The proposed scheme is evaluated in Section VI. Section VII concludes the paper.

II. Related Work

Existing work on stream data and multicast authentication mainly uses message authentication codes (MACs) and secret-key cryptography. The trivial solution is to generate a MAC per packet and send with the packet together. Various approaches [13], [14] are proposed to lower the communication overhead by amortizing the authentication signature (MAC) where a single digital signature is used for the authentication of multiple packets. Gennaro and Rohatgi [13] presented a scheme that uses signature amortization over a hash chain. In [14], a Merkle hash tree is used to amortize a signature over multiple packets.

Another line of work [6], [9], [11], [15] was addressing the packet loss problem. In [11], a MAC is appended to every packet and the key of the MAC is provided in some subsequent packet. To tolerate packet losses, the keys are generated by the means of a hash chain. In [6], a directed acyclic graph is constructed and the hash of a packet is repetitively embedded into multiple packets so that packet loss can be tolerated. Recently, Pannetrat and Molva [9] proposed to deal with packet loss using Erasure codes. Authentication can be performed as long as a certain number of packets are received. Karlof et al. [16] proposed distillation codes to cope with packet loss and packet pollution.

Authentication of multimedia and images has also been studied [17], [18], along the direction to protect the copyright or the ownership of multimedia objects [19], [20]. For the MPEG movie, Hartung and Girod proposed an approach [21] that leverages the DCT transformation procedure. Lu et al. [7] proposed to combine the content authentication and ownership authentication.

Nevertheless, all existing strategies either embed information to the original content, or use out-of-band channels for authentication information communication, neither of which is required in our proposed scheme, where covert channel [22] is leveraged to convey information. In addition, the hash chaining used in DaTA avoids the authentication uncertainty upon data packet/block loss detection that exists in many existing schemes.

DaTA is different from IPsec [23], [24], [25] in that it focuses on authenticating the application layer streaming data, rather than providing the data integrity and/or source authenticity at the IP layer.

III. Basic *DaTA* – *Da*ta-*T*ransparent *A*uthentication

In *DaTA*, the authentication unit is a data block and the authentication code is generated based on the content of the data block, thus called *Block Authentication Code*, abbreviated as *BAC*. *DaTA* works as follows. At the sender side, the authentication information – BAC – is generated based on a selected hash function with the packet content and a commonly agreed key as the input. Based on the value of each bit (0/1) of BAC, some packets are scheduled to be sent out with additional delays. At the receiver side, the receiver extracts the embedded BAC based on the relative packet delay and compares the extracted BAC with the BAC generated based on the received content for authentication. Thus, our proposed scheme consists of the BAC generation, BAC embedding/BAC extraction, and BAC authentication. In this section, we describe the details of these components, after which the packet boundary recognition issue is discussed with regard to packet loss, packet fragmentation, and out-of-order delivery.

To present our scheme, we use the following notations.

1) The stream packets are clustered to blocks, denoted as $block[i]$, with b packets in each block, where $0 < i < \lceil \frac{total\_packet\_number}{b} \rceil$. Padding is used when necessary to generate the last block;
2) The length (in terms of bits) of the BAC for each data block is n;
3) A hash function, denoted as $H(X)$, is a one-way hash, using an algorithm such as MD5 [26] or SHA [27].
4) X,Y represents the concatenation of X with Y.
5) A secret key, k, is only known to the communicating parties;
6) The origin of the data stream can be identified by a flag, which is f bits and only known to the communicating parties, where $0 \leq f \leq n$;

A. BAC Generation

Figure 1 sketches the BAC generation procedure. As indicated in this figure, the BAC generation for data block i involves three steps:

1) The concatenation of data block i and the secret key k is used as input to hash function H to generate a binary

[1]Note that we refer to traditional communication overhead with additional bandwidth consumption.

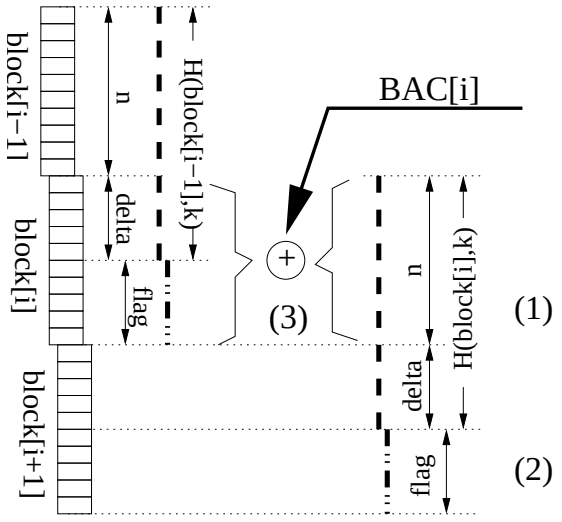

Fig. 1. Generating Block Authentication Code

string of $n+\delta$ bits as **(1)** in Figure 1, where $\delta = n - f$. We call this string the *first-digest*.

2) The source flag, denoted as f, is concatenated to the first-digest generated in the previous step, shown as **(2)** in Figure 1, to get a binary string of $2n$ bits, which we call the *second-digest*.
3) The first n bits of the second-digest is XORed with the last n bits of the block $i-1$'s second-digest, which is generated following the same procedure. The result, $BAC[i]$, is the final BAC for data block i. It will be embedded. This step is shown as **(3)** in Figure 1.

Note that for the first data block, a special second-digest (an Initial Vector - IV) must be agreed on by the communicating parties. In this BAC generation algorithm, if we vary the value of δ while keeping the $\delta + f = n$, we have different strategies.

- $\delta = 0$: when δ is 0, the strategy becomes easy and straightforward. There is no chain at all. It only demands a fixed sized buffer, of b packets, at the sender side and the receiver side. The strategy can detect packet alteration or addition and can locate changes in the granularity of a block. However, it cannot detect block deletion and block (burst packet) loss, which are very important in some stream based applications, such as streaming media delivery, since streaming media data delivery normally runs on UDP.
- $\delta = n$: when δ is n, the strategy cannot authenticate the source (unless the new BAC is XORed with f). With more bits ($2n$) in the authentication code, the strategy reduces the collision rate since the number of bits in the hash result is larger. However, it has a problem due to chaining. For example, if the verification of the current data block indicates that the current block is changed, it means the hash value of the current block cannot be used to authenticate the next block. Thus, the authentication of the next block and all its subsequent blocks will be uncertain. In addition, the protocol cannot distinguish the change of a data block and the deletion (or loss) of a data block.

The choices of δ and f have the tradeoff between authenticating the source and chaining to determine if the preceding block is lost. In most of existing hash chain based strategies, δ is n, or the hash function takes the two consecutive blocks as the input. This causes their authentication deficiency. Thus, an appropriate δ should satisfy $0 < \delta < n$.

B. BAC Embedding and Extraction

Once the BAC has been generated, it needs to be transfered to the receiver of the streaming data so that the streaming data can be authenticated. Unlike most existing streaming data authentication approaches, which either transfer the authentication code via out of band communication or embed the authentication code into the original data content, *DaTA* leverage the approach used in [28] to embed the BAC into the inter-packet timing[2]. Here we show how to embed and extract one bit of the BAC. Embedding and extracting multiple bits can easily be achieved by repeating the following process.

Assume the streaming data flow has m packets $P_1, \ldots, P_m$ with time stamps $t_1, \ldots, t_m$, respectively. To embed one bit, we first independently and randomly choose $2r$ ($r > 0$ is the redundancy used for embedding the bit) distinct packets: $P_{z_1}, \ldots, P_{z_{2r}}$ ($1 \le z_k \le m-1$), and create $2r$ packet pairs: $\langle P_{z_k}, P_{z_k+1} \rangle$ ($d \ge 1,\ k = 1, \ldots, 2r$).

We are interested in the $2r$ IPDs (Inter-Packet Delay) between P_{z_k+1} and P_{z_k}

$$ipd_{z_k} \ = \ t_{z_k+1} - t_{z_k},\ (k = 1, \ldots, 2r). \qquad (1)$$

We randomly divide the $2r$ IPDs into 2 distinct groups of equal size, and make sure that each IPD has equal probability to be in group 1 and group 2. We use $ipd_{1,k}$ and $ipd_{2,k}$ ($k = 1, \ldots, r$) to represent the IPDs in group 1 and group 2 respectively. Since each IPD has equal probability to be in group 1 and group2, IPDs in group 1 and group 2 are symmetric. Therefore, E($ipd_{1,k}$) = E($ipd_{2,k}$), and Var($ipd_{1,k}$) = Var($ipd_{2,k}$).

Let

$$Y_k \ = \ \frac{ipd_{1,k} - ipd_{2,k}}{2}\ (k = 1, \ldots, r). \qquad (2)$$

and

$$\overline{Y_r} \ = \ \frac{1}{r}\sum_{k\,=\,1}^{r} Y_k \ = \ \frac{1}{2r}[\sum_{k\,=\,1}^{r} ipd_{1,k} - \sum_{k\,=\,1}^{r} ipd_{2,k}] \qquad (3)$$

Here $\overline{Y_r}$ is the average of r normalized IPD differences. Since all the $2r$ randomly selected IPDs are equal-probable to be in group 1 and group 2, $\sum_{k\,=\,1}^{r} ipd_{1,k}$ and $\sum_{k\,=\,1}^{r} ipd_{2,k}$ must have the same distribution and mean. Therefore $\overline{Y_r}$ is symmetric, and has zero mean. In other word, $\overline{Y_r}$ has equal probability to be positive and negative.

We embed bit 1 by increasing $\overline{Y_r}$ by $a > 0$ so that its probability to be positive will be greater than 0.5. Similarly, we embed 0 by decreasing $\overline{Y_r}$ by $a > 0$ so that its probability

[2]Please note that the BAC to be embedded is very different from the watermark in [28], and the authentication based on BAC is completely new.

to be negative will be greater than 0.5. To increase or decrease $\overline{Y_r}$ by a, we can simply increase or decrease each of the r Y_k's by a. To increase Y_k by a, we can increase $ipd_{1,k}$ by a and decrease $ipd_{2,k}$ by a. Similarly, we can decrease Y_k by a by decreasing $ipd_{1,k}$ by a and increasing $ipd_{2,k}$ by a.

Apparently the embedding is probabilistic in that there is a small chance that the bit will not be successfully embedded. As shown in [28], the embedding successful rate can be made arbitrarily close to 100% with sufficiently large redundancy r.

Assuming that the receiver shares the secret of which packets are used for encoding and decoding the BAC, we can calculate $\overline{Y_r}$ at the receiver side from the timestamps of the received packets. To extract an embedded bit, we check the value of $\overline{Y_r}$. If it is greater than 1, we decode the embedded bit as 1, otherwise, we decode the embedded bit as 0.

It is true that decoding a particular bit could be wrong due to the probabilistic nature of encoding and network delay jitter. However, analysis in [28] has shown that the correct decoding rate can be made arbitrarily close to 100% with sufficiently large redundancy despite of network delay jitter.

C. Authentication

With the extracted BAC bits and received data packets, the receiver applies the same hash function (H) on the received data packets with the same secret key (k) to generate the content based BAC following the same procedure used for BAC generation at the sender side (see section III-A). Then the extracted BAC is compared with the content based BAC.

The comparisons consist of two parts: the first part is on the first δ bits, while the second is on the rest f $(= n - \delta)$ bits. Considering the possible scenarios, Table I briefly summarizes the authentication result.

Note that in our scheme, every received data block is authenticable independently, which is based on the f bits matching in the BAC comparisons. We always check this part first. The first δ bits can indicate whether the preceding data block is deleted or lost. Thus,

- in case 1, the extracted and generated BACs are completely matched, the current data block is authenticated to be genuine;
- in case 2, the second part authentication failure indicates that the current data block is changed; although the first part authentication succeeds, we cannot conclude the preceding block is genuine;
- in case 3, the success of the second part authentication indicates the current block is genuine. Therefore, the first part authentication failure indicates that the preceding data block has been changed/deleted[3];
- in case 4, the authentication failure on both parts strongly suggests that the current data block is changed. The status of the preceding block is uncertain.

Considering the situations of the preceding data block authentication, *DaTA* can distinguish the deletion/loss from alteration. For example, given two consecutively received data blocks, if the second part of the preceding data block authentication is successful, while the first part of the current block authentication fails and the second part succeeds, some data blocks between these two are lost/deleted.

Through the above analysis, *DaTA* exhibits the following features:

- It can detect any tampering, including content alteration and packet/block deletion;
- It is able to locate the change to the data block level so that in some circumstances, the application can make decisions (e.g. retransmission) accordingly;
- It is capable of ensuring that each block is authenticable even if any preceding data block is found to be changed. In another word, it is not a fragile scheme;
- Besides authenticating the content itself, the scheme is also able to authenticate the content source. This is due to the splitting of the authentication into two parts, where the second part authenticates both content and the content source.

D. Block Boundary Recognition

In our scheme, it is important that the receiver can correctly delimit packets into an appropriate data block. If a packet is mistakenly delineated at the receiver side, both the BACs extracted from the packet arrival IPDs and generated based on the received packets are not consistent with the sender side. This makes authentication impossible. Since packet loss always causes trouble with block boundary recognition, we discuss it in this section.

To enable the receiver to delineate the data block boundary at the sender side, the last packet of the previous data block and the first packet of the current data block should be specially utilized. In our proposed scheme, the BAC bit is embedded by slightly adjusting the packet delivery time, so if we intentionally add a sufficient large delay between these two packets (but still in the allowable range), this delay serves as the boundary delineating purpose.

In our scheme, the normal inter-packet delay adjustment is limited by a, where a is the *adjustment size* (see Section III). Thus, if we find the maximum inter-packet delay in the data stream (through online profiling), and add an extra a, this number is sufficient to be used as block boundary delimiter if there are no network fluctuations. In practice, some extra space can be added to allow for network fluctuations. Thus, we propose to use $2a$ and will evaluate this value in the experiments.

Packet loss is not uncommon in streaming based communications, particularly when the streaming runs on UDP. Although existing research [29] and measurement [30] find that packet loss is less than 0.1% on the Internet backbone, our strategy is very sensitive to that since the loss of a packet not only damages the current data block, but may also cause a change in the block boundary and result in incorrect BAC extraction.

A simple method for *DaTA* to deal with packet loss is to examine both the number of packets and the artificial block boundary delay. At the receiver side, whenever an artificial block boundary delay is found, the receiver considers that a

[3]Theoretically, there is false positive if the change of the current block does not alter the second part. But the probability is very small.

case number	δ bits	$n-\delta=f$ bits	Current Data Block	Other Implications
1	match	match	true	None
2	match	mis-match	false	current data block changed
3	mis-match	match	true	preceding data block loss/deletion or alteration
4	mis-match	mis-match	false	current data block changed

TABLE I
AUTHENTICATION SCENARIOS

block is completely received and the next packet belongs to the next block. If the received packet number is less than b, clearly, some packets are lost.

Regarding the packet boundary, a particular concern comes from the effect of additional delay on the application's performance in the proposed scheme. Since the additional delay introduced by our scheme is always in the granularity of several milliseconds, it could hardly affect the application's performance. In addition, it is common that at the sender and receiver sides, applications always use buffers to buffer some transmitted data before they are used. For example, for Internet video streaming, normally there is always a few second initial buffer. Taking these facts into consideration, the additional delay introduced for boundary recognition by our proposed scheme can be easily absorbed by the application buffer.

Two other factors that may raise concerns are the packet fragmentation and out-of-order delivery. Since the packet fragmentation rate is very low today [31], we don't consider the problem caused by the normal packet fragmentation. For out-of-order delivery, it is observed that only about 0.3% packets arrive out of order [32]. Thus, we don't consider its effect in this study.

IV. *DaTA* ANALYSIS

In *DaTA*, the authentication information is generated based on the content of the data block (see section III-A), which we call *content BAC*. The BAC embedded to (or extracted from) the inter-packet timing is called the *reference BAC*. At the receiver side, when the receiver receives the stream flow, the reference BAC extracted from the inter-packet timing will be used as a reference for the content BAC calculated from the packet content.

Let $C = b_1, \ldots, b_n$ be the original content BAC and the original reference BAC (they are always equal), $C_1 = b_{1,1}, \ldots, b_{1,n}$ be the reference BAC of the received stream flow, and $C_2 = b_{2,1}, \ldots, b_{2,n}$ be the content BAC of the received streaming flow. In ideal case, $C_1 = C_2 = C$. In real world, both C_1 and C_2 could deviate from C due to reasons such as packet loss, content change or network delay jitter. Our goal is to determine whether C_2 equals to C. However, the receiver of the streaming flow does not know C and he/she only knows C_1 and C_2. Therefore, the receiver has to use C_1 and C_2 to determine whether C_2 equals to C.

After getting C_1 and C_2 from the received streaming flow, the receiver uses the following rules to determine whether C_2 equals to C:

$$\begin{cases} C_2 = C & \text{if } C_1 = C_2, \\ C_2 \neq C & \text{if } C_1 \neq C_2. \end{cases} \tag{4}$$

Here we use C_1 as the reference for checking the value of C_2. Since C_1 could be different from C, it is possible that

1) C_2 is falsely determined to be equal to C; or
2) C_2 is falsely determined to be different from C.

We regard the first case as a false positive, and the later case as a false negative. The false positive rate and the false negative rate can be quantitatively represented as $\Pr[C_1 = C_2 | C_2 \neq C]$ and $\Pr[C_1 \neq C_2 | C_2 = C]$ respectively.

Assume the probability that any particular bit in C_1 and C_2 is different from the corresponding bit in C is independent from each other. Let $\Pr[b_{1,i} = b_i] = p_1$ and $\Pr[b_{2,i} = b_i] = p_2$, where $1 < i < n$.

The false negative rate is thus

$$\begin{aligned} & \Pr[C_1 \neq C_2 | C_2 = C] \\ = \;& 1 - \Pr[C_1 = C_2 | C_2 = C] \\ = \;& 1 - \frac{\Pr[C_1 = C \wedge C_2 = C]}{\Pr[C_2 = C]} \\ = \;& 1 - \Pr[C_1 = C] \\ = \;& 1 - p_1^n \end{aligned} \tag{5}$$

Let X_k be the probability that both C_1 and C_2 (where $C_1 = C_2$) have exactly the same $0 \leq k \leq n$ bits different from C, and let Y_k be the probability such that C_2 have exactly $0 \leq k \leq n$ bits different from C, then we have

$$X_k = \left(\begin{array}{c} n \\ k \end{array} \right) p_1^{n-k}(1-p_1)^k \times p_2^{n-k}(1-p_2)^k$$

and

$$Y_k = \left(\begin{array}{c} n \\ k \end{array} \right) p_2^{n-k}(1-p_2)^k$$

When $0 < p_1 \leq 1,\ 0 < p_2 < 1$, the false positive rate is

$$\begin{aligned} & \Pr[C_1 = C_2 | C_2 \neq C] \\ = \;& \frac{\Pr[C_1 \neq C \wedge C_2 \neq C \wedge C_1 = C_2]}{\Pr[C_2 \neq C]} \\ = \;& \frac{\sum_{k=1}^{n} X_k}{\sum_{k=1}^{n} Y_k} \\ = \;& \frac{\sum_{k=1}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) p_1^{n-k}(1-p_1)^k \times p_2^{n-k}(1-p_2)^k}{\sum_{k=1}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) p_2^{n-k}(1-p_2)^k} \\ = \;& \frac{[p_1 p_2 + (1-p_1)(1-p_2)]^n - p_1^n p_2^n}{1 - p_2^n} \end{aligned} \tag{6}$$

Based on Equation 5, it is easy to see that the larger p_1 is, the smaller the false negative rate. Equation 6 indicates that when p_1 approaches 1, the false positive rate approaches 0.

Therefore, it is desirable to make the reference BAC as robust as possible so that both the false positive and false negative rates are small. This analysis result demonstrates the necessity of enhancing the robustness of our basic authentication strategy.

Given any fixed p_1 and p_2, larger n will decrease the false positive rate as implied in Equation 6. Besides the fact that the false positive rate decreases with the increase of n, there is always a maximum value for the false positive rate for different n as indicated in these figures.

Simply increasing the BAC length is not always possible. In the next section, we will introduce some other optimizations for improving p_1 and reducing delay impact to the application's performance.

V. Optimizations and Discussions

The analysis in the previous section shows that it is necessary to increase the robustness of the reference BAC. The accuracy of the reference BAC could be affected by many factors, such as network jitter, packet loss, adjustment size, and BAC length. Thus, it is necessary to optimize the basic design to make it practical. In this section, we discuss several issues and strategies for this purpose.

A. Redundancy Level

In an ideal environment, where there is no network fluctuations, embedding a BAC bit using two packet pairs (4 packets) is enough (redundancy level $r = 1$). For example, to embed a 24-bit BAC, we only need 96 packets if these packets are not repetitively used. A block size of 96 packets will make locating a changed packet easier. This provides convenience and reduces cost for a remedy: if the application decides to retransmit the tampered block, only 96 packets need to be re-transmitted.

However, due to network fluctuations, a redundancy level of 1 will make the embedded BAC error-prone since the BAC embedding in our scheme is probability based. An occasional large jitter could delay one of these packets, resulting in an error of the corresponding embedded BAC bit, and decreases the authentication accuracy of our scheme.

According to Equation 3, the larger the redundancy level r, the more centralized to 0 of the $\overline{Y_r}$ distribution. In another word, the instance of $\overline{Y_r}$ will be more likely to fall into $[-a, +a]$. Given a fixed adjustment size a, a large redundancy level leads to a higher authentication accuracy. Thus, to make our scheme robust, it is desirable to increase the redundancy level.

But the higher the redundancy level, the more packets are needed in a data block. For example, if the redundancy level is increased to 20 for the 24-bit BAC, at least 1920 packets are needed. If any of these packets is lost or tampered with, 1920 packets have to be retransmitted. Thus, we should always use an appropriate redundancy level. We further explore this via experiments in Section VI.

B. Trade-off Between Redundancy Level and BAC Length

In the above, we have shown that an appropriate redundancy level should be determined when the block size can vary. Under this condition, the block size should be kept relatively small. In other situations, if a block size is limited or fixed, a new issue arises.

Given a fixed data block size, increasing the redundancy level will increase the robustness of the extracted BAC bits, thus increasing p_1 in the above equation. However, with a limited number of packets in the block, increasing the redundancy level also reduces the number of BAC bits that can be embedded. That is the BAC length. The BAC length determines the collision rate and the false positive rate of the scheme. Thus, a longer BAC length is always desired. Clearly, increasing the redundancy level and increasing the BAC length have conflicting interests and thus must be well balanced. That is to say, when the block size is determined, a trade-off between block size and BAC length exists, and an appropriate redundancy level should be used. This will be further evaluated in Section VI.

C. Bidirectionally Embedding A Bit

So far to embed a BAC bit, we always increase the IPD value of two packets. Intuitively, to embed bit 1, we always delay the first and the second packet of the first and second packet pair, respectively. That means those two packets are sent a later than their original scheduled delivery time, while the other two packets are sent as usual. To embed bit 0, the second and the first packet of the first and second packet pairs are sent with a delay. Based on the previous analysis, the larger the adjustment size a, the higher the accuracy. However, if a is large, it may adversely affect the performance of some real-time applications. For example, for online video data authentication, a very large delay of the packet may cause the packet to be dropped, or cause the client to experience playback jitter. Neither of these situations is pleasant.

To alleviate the delay effect, an optimization can be done as follows: to embed bit 1, we send the first packet in the first pair $\frac{a}{2}$ later than its originally scheduled delivery time and the second packet $\frac{a}{2}$ earlier than its originally scheduled delivery time. For the second packet pair, the first packet is delivered $\frac{a}{2}$ earlier while the second is delivered $\frac{a}{2}$ later. Similarly we can embed bit 0.

This optimization is constrained by buffer availability at the sender side and is thus advantageous only for some applications. If there is a non-zero-second buffer (such as encoding buffer) at the sender side, this optimization can be applied.

VI. Performance Evaluation

To study the effectiveness of our proposed scheme, we implemented a real-time Linux kernel based packet-level authentication prototype system. The system runs at the precision of 100 microseconds (the normal Linux system runs at the 10 milliseconds) and is capable of online embedding and extracting authentication information. Based on the prototype system, we first performed 5 sets of experiments in a local

(and more controllable) network to study various aspects of our proposed scheme with artificially introduced network jitter and packet loss. The experimental results over the Internet are omitted due to page limit. Interested readers can refer to our technical report [33].

A. Experiment Setup

In local experiments, the workload we used is a 160 second streaming video, which is encoded at 300 Kbps with a rate of 15 frames/second, with a total of 10299 packets.

The experimental platform consists of three machines. A Linux box of 2.4 GHz CPU with 1 GB memory running Fedoral Core 2 is used for Darwin Streaming server [34] or TCP Replayer, and a second Linux machine of CPU 2.4 GHz and 1 GB memory running our prototype system is set up as a router, where the BAC bits are added before the data is streamed to the destination machine, which runs Windows XP with 1.8 GHz CPU 1.8 and 512 MB memory.

To generate the content based BAC, we use MD5 with the payload of the data packets as the input. The output hash value consists of 128 bits, of which the first and the last 12 bits are cut off and concatenated as the BAC for that data block. In these experiments, the number of BAC bits is always 24 and the block length is 1000 if not explicitly noted. We set the packet distance as 1 and the distance between the selected packet pair (for embedding) is 2. With different redundancy levels, the required number of packets varies. For example, with a redundancy level of 6, a total of 576 packets is used for adjusting the inter packet delay, since we use one packet only once in a packet pair.

One particular important factor affecting our proposed scheme is the network jitter. In local experiments, we run experiments when two types of jitter distribution (uniform and normal) present. For each distribution, we also test the jitter with different probabilities to simulate the normal network jitter (1%) and the extreme case (100%). A jitter probability of 1% means among 100 packets only 1 packet is randomly selected to be delayed and this emulates the normal network. A 100% jitter probability is to emulate the network upon network congestion (and thus packet burst, or a spike), where all packets are delayed. The delay of selected packet is always limited by the maximum jitter value in each experiment.

B. Block Boundary Detection

In our proposed scheme, the correct block boundary detection is the basis for authentication. The data block boundary is determined by examining and looking for the artificially introduced large inter-packet delay. A sufficiently large delay indicates that a boundary is met. With the dynamics of network fluctuations, it is possible that the network fluctuations may affect the correctness of the block boundary detection. In addition, a packet loss or deletion may introduce large inter-packet delays.

First, in our experiments we study the impact of network fluctuations. In these experiments, we set the block size as 100 and have 103 data blocks in total. Figure 2 and Figure 3 show the boundary detection error rates when the network fluctuations follow a uniform distribution for the video stream. In the figures, x-axis represents the network *jitter* in *ms*, while y-axis denotes the sum of *false positive* and *false negative* (due to page limit). We simply refer to them as the *error rate* in the following context. *False positive* is defined as the error when a non-boundary is recognized as a boundary, while *false negative* means the error when a boundary is not recognized as boundary. Each experiment is repeated 100 times. Since the maximum inter-packet delay in the original packet flow is 69 ms, and the average is 15.221 ms, we apply the adjustment size a, from 1 to 10 ms and shown on the figures with all even values.

In Figure 2 and Figure 3, packet jitter occurs with probability 1% and 100%, respectively. As shown in both figures, given certain network jitter, the larger of the adjustment size, the smaller the error rate. If the maximum jitter is smaller than the adjustment size, the error rate is always 0 or close to 0 in both cases. Note in Figure 3, the error rate is larger than 1 when the adjustment size is 2 ms and maximum jitter is larger than 7 ms. This is reasonable since when each packet is delayed and the jitter is much larger than the adjustment size, a boundary could be found between each packet pair. These results suggest that an adjustment size of 4 ms is good to consider average network jitter (less than 1 ms) as reported by [30]. We will further test this in the Internet experiments.

Experiments have also been performed when the jitter follows a normal distribution with a mean value in the range of 2-8 ms. *Compared to the uniform distribution, the jitter following a normal distribution has less impact on the correctness of boundary detection. We omit the figures for brevity.*

Second, we study the impact of packet loss to boundary detection. We artificially delete packets from the data block randomly with the deletion probability of $1/100$, $1/1000$, $1/10000$, $1/100000$, and $1/1000000$ in the entire packet flow, respectively. Experiments are repeated 100 times to check if a false boundary is found. Figure 4 shows the result. Roughly, adjustment sizes do not make much difference. As reported in [30], the regular packet loss rate on the Internet is less than 0.1%. So a a of 4 ms can detect the boundary with more than 99% accuracy.

C. Redundancy Level/BAC Length

In a regular network environment, network fluctuations are common and may cause the extracted BAC bits to be wrong. Thus, it is important to increase the redundancy of the embedded BAC so that our scheme can survive with dramatic network jitter.

First, experiments are conducted to find an appropriate redundancy level for embedding authentication code to our data streams. The basic BAC bits are 24 bits, and we set the adjustment size as 4 ms. We only add BAC to a data block with 2000 packets, so that different redundancy levels can be tested. Again, we artificially introduce data flows with the uniform and normal jitter with probabilities of 1% and 100%.

We define the *error bit* as the total number of error bits in the experiments in average. Figure 5 and Figure 6 show the number of error bits when the redundancy level varies

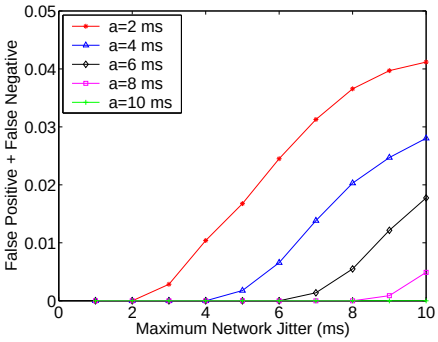

Fig. 2. Boundary Detection: Uniform Jitter Distribution (1%)

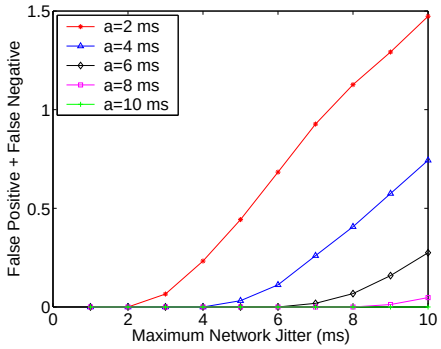

Fig. 3. Boundary Detection: Uniform Jitter Distribution (100%)

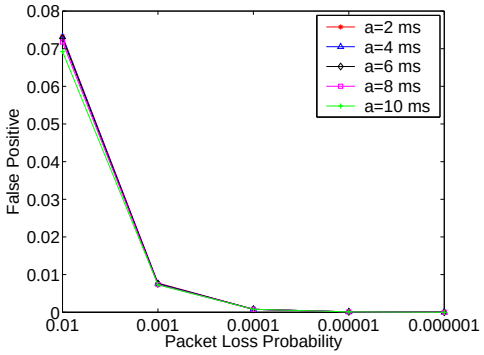

Fig. 4. Boundary Detection: Packet Loss

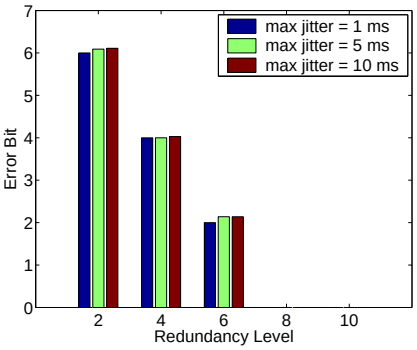

Fig. 5. Redundancy Level: Uniform Jitter Distribution (1%)

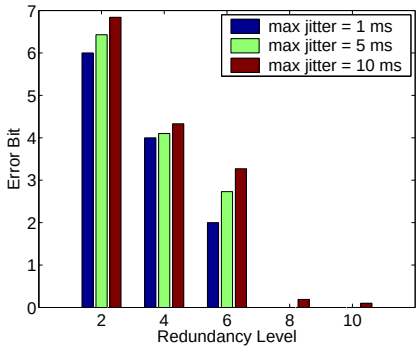

Fig. 6. Redundancy Level: Uniform Jitter Distribution (100%)

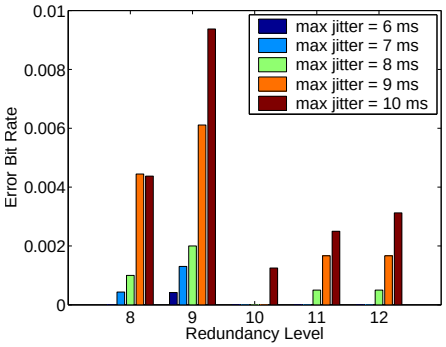

Fig. 7. Redundancy Level vs. BAC Length: Uniform Jitter Distribution (100%)

while the maximum network jitter is 1, 5, and 10 ms, respectively. In general, both figures show that the number of error bits decreases when the redundancy level increases. A more important observation from these two figures is that a redundancy level of 8 produces a nearly 0 error bit in the 24 bits, even when the jitter follows a uniform distribution with the maximum delay in the range of 1-10 ms and 100% packet jitter probability.

As discussed in section V-B, given a fixed data block size, there is always a trade-off between the redundancy level and the BAC length. They should be kept in balance. In the following experiments, we explore this by testing a data block consisting of 1000 data packets. Given a block size of 1000 packets, there are a few pairs of the redundancy level and BAC length, including (6, 24), (7,23), (8,20), (9,18), (10, 16).

Because the BAC length varies, when the network jitter follows a uniform distribution in the range of 1 - 10 ms with the probability of 1% and 100%, experiments are conducted to test the *error bit rate*. The *error bit rate* is defined as the total error bits of the 100 experiments divided by the product of the BAC length and the number of experiments.

When the packet jitter probability is 1%, the error bit rate is always 0. When the packet jitter probability is 100%, Figure 7 shows the error bit rate when the maximum jitter is in the range of 6-10 ms. When the maximum jitter is less than 6 ms, the error bit rate is always 0.

As shown in this figure, although in general the error bit rate approaches 0, a redundancy level of 10 clearly achieves the best result with a BAC length of 16 bits. When the redundancy level further increases, the error bit rate increases, since the collision rate increases due to the shortness of the BAC length.

This set of experiments verifies the existence of the trade-off. Thus, for a fixed block size, the appropriate BAC length and redundancy level should be found to achieve the best authentication performance.

D. Detect/Locate Content Change

As aforementioned, the BAC in our scheme is content based. Ideally, when the content is changed, the extracted BAC should not be consistent with the reference content BAC. However, if due to network fluctuations, the extracted BAC is also changed, and is same as the new content BAC, a false positive happens. The following two experiments are performed to evaluate these situations by either deleting a packet, or changing a bit in a packet randomly.

Firstly, we run an experiment upon packet loss. For this test, we first embed BAC to the workload with the redundancy level of 8 and the adjustment size as 4 ms for the 24 bit BAC. In the embedded workload, we randomly drop a packet. We also randomly perturb the embedded workload with network jitter using the uniform perturbation where the max delay varies from 1 to 10 ms and for each distribution. The experiment runs for 10000 times for each jitter setting to randomly select a packet to drop.

Figure 8 and Figure 9 show the result when the jitter in uniform distribution with 1% and 100% probabilities. In these

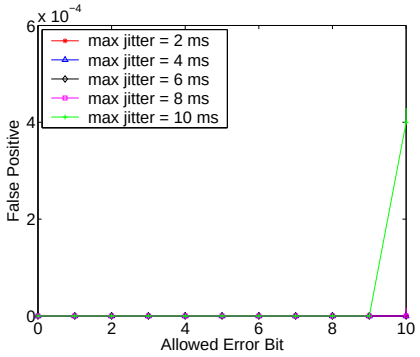

Fig. 8. Locate Packet Loss: Uniform Jitter Distribution (1%)

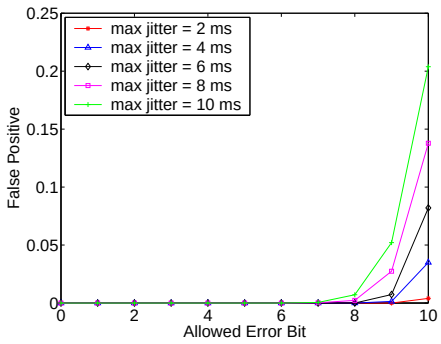

Fig. 9. Locate Packet Loss: Uniform Jitter Distribution (100%)

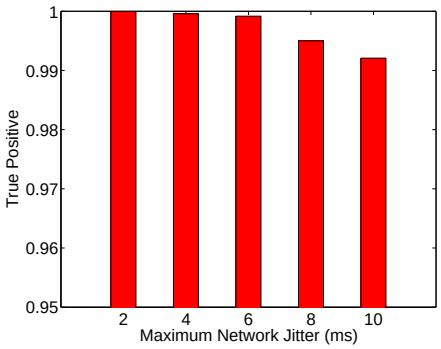

Fig. 10. Authentication Accuracy: Uniform Jitter Distribution (100%)

figures, the x-axis represents the *allowed error bit*. This means how many erroneous BAC bits are allowed to be ignored. An allowed error bit of 0 means that all the 24 BAC bits must be matched. An allowed error bit of 1 means that if only 1 bit is different, the authentication is taken as a success. The y-axis represents *false positive*, which means a failure to detect and locate a dropped packet. It is expected that the larger the allowed error bit number, the larger the false positive. Both figures indicate that unless the allowed error bit is very large (> 8), our scheme can always successfully locate a packet change in a data block. Note an allowed error bit of 8 means among the 24 BAC bits, 1/3 bits are not required to match. This is not commonly acceptable in normal applications.

We further test without deleting a packet, but changing the content of a randomly selected packet and repeated the experiment. The result shows the alteration can be detected with 100% accuracy, with the uniform distribution of 1% or 100%.

E. Authentication Accuracy

The previous experiments have successfully determined that when the BAC length is 24, an adjustment size of 4 ms and a redundancy level of 8 are good for our experimental data streams. With these fixed parameters, the following experiments are conducted to examine the accuracy, denoted as *true positive*, of our authentication scheme when the jitter on packets comes with a 1% or 100% probability.

When the jitter follows a uniform distribution with a probability of 1%, our scheme always achieves 100% true positive. Thus, we only show the true positive for the authentication scheme when the packet jitter probability is 100% in Figure 10. As shown in the figure, when the maximum jitter increases beyond 4 ms, our scheme cannot achieve a 100% true positive. However, even when the maximum jitter is 10 ms, our scheme still achieves a true positive above 99%.

F. Impact on Applications

Having experimented our scheme with different network fluctuation distributions, block sizes, and redundancy levels, now we study the impact of our embedded BAC (delay) on sensitive applications.

First, we watch the movie after it is embedded with the 24 bit BAC, redundancy level of 8, and an adjustment size of 4 ms when artificially introduced network jitter follows the uniform or normal distribution with probabilities of 1% and 100%. We did not observe any jitter even when we set up a 0-second buffer at the client side. To study the effect scientifically, we conducted experiments to calculate the packet-timestamp difference and inter-packet-delay.

Figure 11 shows the timestamp differences of the three flows, represented by *trial1*, *trial2*, and *trial3*. In *trial1*, we did not embed BAC to the original data flow and captured the data packets at the client side. To reflect the network jitter in our experimental environment, we repeated the playback and captured the flow as *trial2*. *trial3* is the flow we captured at the receiver side after the original flow is embedded with BAC. Thus, the difference between *trial1* and *trial2* indicates the local network jitter. The result indicates that network jitter in our local experimental environment is trivial, and thus should not affect the correctness of our conducted experiments. The difference between *trial1* and *trial3* represents the experimental network jitter and the additional jitter introduced by our authentication scheme. As shown in the figure, 81% timestamp difference for both scenarios falls into the same range, from -6 to 8 ms. This is trivial to the client side player, particularly when these players normally have a non-zero second buffer.

Figure 12 further shows the IPD values of the flow captured at the client with or without embedded BAC, denoted by *embedded* and *original*. Our introduced BAC does shift the distribution a little bit (about 1 ms) at the beginning. However, The largest portion of the IPD values is still around 15 ms.

VII. Conclusion

Data streams have been commonly used in many Internet applications, such as grid computing and streaming media. More and more such applications demand a reliable and effective authentication mechanism to ensure the data streams transferred over the Internet are genuine. Although plenty of research work has been conducted, existing work shares the characteristics of either sending the authentication information with the original data or out-of-band, which causes additional communication overhead, particularly when the redundancy is added to deal with packet loss, or embedding authentication

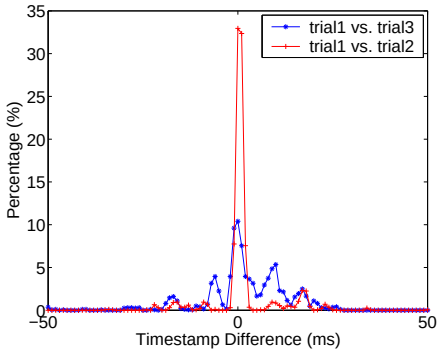

Fig. 11. Timestamp Difference of Authenticated Flow

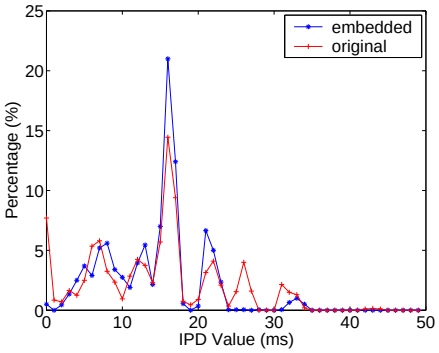

Fig. 12. IPD Breakdowns

into the original data by replacing some non-critical bits, which changes the original data. In this study, we propose a new scheme by adjusting packet timing (delay) to authenticate the data stream. Thus, authentication is done without changing the original packet content and without sending additional authentication information. Extensive experiments are conducted locally and over the Internet based on an implemented prototype system. The results show that our scheme is robust under various network conditions and practical.

VIII. Acknowledgment

We would like to thank William L. Bynum and anonymous reviewers for their helpful comments on this paper. The work is partially supported by NSF grant CNS-0509061/CNS-0524286 and DTO/ARDA/AFRL grant FA8750-05-2-0266.

References

[1] "National hurricane center," http://www.nhc.noaa.gov/.
[2] "National oceanic and atmospheric adminstration," http://www.nesdis.noaa.gov/.
[3] CiberStock Quote & Chart/Share Price, ," http://www.advfn.com/.
[4] M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and analysis of a streaming media workload," in *Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems*, March 2001.
[5] M. Chen, Y. He, and R. Lagendijk, "A fragile watermark error detection scheme for wireless video communications," in *IEEE Transactions on Multimedia*, 2003.
[6] P. Golle and N. Modadugu, "Authenticating streamed data in the presence of random packet loss," in *Proceedings of SPIE Security and Watermarking of Multimedia Contents*, San Jose, CA, January 2001.
[7] C. Lu, H. M. Liao, and L. Chen, "Multipurpose audio watermarking," in *Proceedings of International Conference on Pattern Recognition (ICPR'00)*, Barcelona, Spain, September 2000.
[8] S. Miner and J. Staddon, "Graph-based authentication of digital streams," in *Proceedings of IEEE Symposium on Security and Privacy*, 2001.
[9] A. Pannetrat and R. Molva, "Real time multicast authentication," in *Proceedings of Network and Distributed System Security Symposium (NDSS'03)*, San Diego, CA, February 2003.
[10] J. Parlk, E. Chong, and H. Siegel, "Effi cient multicast packet authentication using signature amortization," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, 2000.
[11] A. Perrig, J. D. Tygar, D. Song, and R. Canetti, "Effi cient autentication and signing of multicast streams over lossy channels," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2000.
[12] S. Ben-David, J. Gehrke, and D. Kifer, "Detecting change in data streams," in *Proceedings of the 30th VLDB Conference*, Toronto, Canada, August 2004.
[13] R. Gennaro and P. Rohatgi, "How to sign digital streams," in *Proceedings of Crypto97*, 1997.
[14] C. L. Wong and S. S. Lam, "Digital signatures for fbws and multicasts," in *Proceedings of ICNP*, 1998.
[15] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, November 1999.
[16] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. Tygar, "Distillation codes and applications to dos resistant multicast authentication," in *Proceedings of Network and Distributed System Security Symposium (NDSS'04)*, February 2004.
[17] Z. Liu, X. Li, and Z. Dong, "Multimedia authentication with sensor-based watermarking," in *Proceedings of ACM Multimedia Workshop on Security*, Magdeburg, Germany, September 2002.
[18] L. C. Yung and C. S. Fu, "A robust image authentication method distinguishing jpeg compression from malicious manipulation," in *IEEE Transactions on Circuits and Systems for Video Technology*, February 2001, vol. 11 (2).
[19] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman, "Electronic marking and identifi cation techniques to discourage document copying," in *Proceedings of IEEE INFOCOM*, Toronto, Canada, June 1994.
[20] J. Fridrich and M. Du, "Images with self-correcting capabilities," in *Proceedings of IEEE International Conference on Image Processing*, 1999.
[21] F. H. Hartung and B. Girod, "Watermarking of mpeg-2 encoded video without decoding and reencoding," in *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking*, San Jose, CA, February 1997.
[22] S. Cabuk, C. E. Brodley, and C. Shields, "Ip covert timing channels: Design and detection," in *Proceedings of CCS*, Washington, DC, October 2004.
[23] J. Ioannidis and M. Blaze, "The architecture and implementation of network-layer security under unix," in *Proceedings of USENIX Security Symposium*, Santa Clara, CA, October 1993.
[24] S. Kent and R. Atkinston, "Rfc 2401: Security architecture for the internet protocol," IETF, September 1998.
[25] H. Yin and H. Wang, "Building an application-aware ipsec policy system," in *Proceedings of USENIX Security Symposium*, Baltimore, MD, August 2005.
[26] "Rfc 1321 - the md5 message-digest algorithm," http://www.faqs.org/rfcs/rfc1321.html.
[27] National Instutite of Standards and NIST FIPS PUB 180 Technology, "Secure hash stardard," U.S. Department of Commerce, May 1993.
[28] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the internet," in *Proceedings of the 12th ACM Conference on Computer Communications Security*, 2005.
[29] L. Ciavattone, A. Morton, and G. Ramachandran, "Standardized active measurements on a tier 1 ip backbone," in *IEEE Communications Magazine*, June 2003.
[30] "Global ip network home," http://ipnetwork.bgtmo.ip.att.net/pws/.
[31] C. Shannon, D. Moore, and k claffy, "Characteristics of fragmented ip traffi c on internet links," in *Proceedings of ACM Internet Measurement Workshop*, San Francisco, CA, November 2001.
[32] Y. Zhang, V. Paxson, and S. Shenker, "The stationarity of internet path properties: Routing, loss, and throughput," Tech. Rep., 2000.
[33] S. Chen, S.P. Chen, X. Wang, and S. Jajodia, "Data – data-transparent authentication without communication overhead," Tech. Rep., Dept. of Computer Science, George Mason University, 2006.
[34] "Apple darwin streaming server," http://developer.apple.com/darwin/projects/.