An Application-Level Data Transparent Authentication Scheme without Communication Overhead

Songqing Chen, *Member*, *IEEE*, Shiping Chen, *Senior Member*, *IEEE*, Xinyuan Wang, Zhao Zhang, *Senior Member*, *IEEE*, and Sushil Jajodia

Abstract—With abundant aggregate network bandwidth, continuous data streams are commonly used in scientific and commercial applications. Correspondingly, there is an increasing demand of authenticating these data streams. Existing strategies explore data stream authentication by using message authentication codes (MACs) on a certain number of data packets (a data block) to generate a message digest, then either embedding the digest into the original data, or sending the digest out-of-band to the receiver. Embedding approaches inevitably change the original data, which is not acceptable under some circumstances (e.g., when sensitive information is included in the data). Sending the digest out-of-band incurs additional communication overhead, which consumes more critical resources (e.g., power in wireless devices for receiving information) besides network bandwidth. In this paper, we propose a novel strategy, *DaTA*, which effectively authenticates data streams by selectively adjusting some interpacket delay. This authentication scheme requires no change to the original data and no additional communication overhead. Modeling-based analysis and experiments conducted on an implemented prototype system in an LAN and over the Internet show that our proposed scheme is efficient and practical.

Index Terms—DaTA, authentication, timing correlation, covert channel, data transparent.

1 Introduction

The increasing Internet bandwidth has enabled more and more scientific and commercial applications to use continuous data input via networks in parallel to the traditional local data sets. 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. Grid computing boosts such demands. 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., teleconferencing, remote education, and entertainment) are becoming ubiquitous today in our everyday life [4]. In such applications, more and more data streams carry important or sensitive information. To use oceanic data from NOAA for computing, it is important for the receiver to ensure that the received data are genuine. Stock shareholders are sensitive to the dynamically changing prices received in their PDAs. Thus, it is important for the receiver to be able to authenticate the received data. Applying end-to-end encryption schemes can provide strong authentication of the data and the information source. However, sometimes, this is overkill. For example, when the information (such as the scientific oceanic data, the top five falling stocks) is for the public. It is not necessary to use encryption to hide such information. In addition, encryption normally is expensive in terms of resource consumption, and encryption on a continuous data stream is inefficient.

Other than encryptions, many authentication strategies have been proposed [5], [6], [7], [8], [9], [10], [11]. Although they authenticate different data streams, such as real-time and on-demand data streams, and unicast and multicast data streams, in general, these strategies always apply message authentication codes (MACs), such as HMAC or NMAC, or a one-way hash function, such as SHA or MD5, on a certain number of packets, called a data block, or blocks to generate a message digest. The digest is then embedded into the original data or sent out-of-band to the receiver. In the embedding approach, the idea is to embed the digest into the packet payload by altering some bits in the payload, similar to watermarking digital images to protect the copyright of the image [12] (more in Section 2). This approach does not incur digest communication overhead. However, with the embedded digest, the original data are altered, which is not feasible when the data carry sensitive or even critical information.

The second approach of sending the digest out-of-band or appending an MAC to the original data incurs additional communication overhead. As a result, this approach increases the data bit rate for communicating extra authentication information. More importantly, although the additional

- *S. Chen and X. Wang are with the Department of Computer Science, George Mason University, 4400 University Drive, MSN 4A5, Fairfax, VA 22030. E-mail: {sqchen, xwangc}@cs.gmu.edu.*
- *S. Chen is with Sybase, Inc., 1 Sybase Dr., Dublin, CA 94568. E-mail: shiping.chen@sybase.com.*
- *Z. Zhang is with the Department of Electrical and Computer Engineering, Iowa State University, 2215 Coover Hall, Ames, IA 50011. E-mail: zzhang@iastate.edu.*
- *S. Jojodia is with the Center for Secure Information Systems, George Mason University, MSN 5B5, Fairfax, VA 22030-4444. E-mail: jajodia@gmu.edu.*

Manuscript received 8 Jan. 2008; revised 4 May 2009; accepted 4 Sept. 2009; published online 6 April 2010.
Recommended for acceptance by S. Olariu.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2008-01-0011.
Digital Object Identifier no. 10.1109/TC.2010.80.

network bandwidth for transmitting the digest out-of-band is generally not significant, with the substantially increase of mobile devices and sensor networks, the additional communication consumes critical battery power, which is the most important resource to keep the devices alive. Thus, receiving additional digest data out-of-band should be avoided whenever possible.

In this paper, we propose a novel method, *Da*ta-*T*ransparent *A*uthentication (*DaTA*) without Communication Overhead, to authenticate data streams. Our strategy neither embeds a digest to the original data, nor sends any out-of-band authentication information. Instead, our scheme is based on the timing correlation of data packets between the sender and the receiver. Particularly, the interpacket delays are utilized and some selected packet delays are slightly adjusted (in a range). The interpacket delay increase and decrease represent different bits (0 or 1), and thus, transparently embed the digest. Since we limit the delay adjustment in a small range and the delay adjustment is not cumulative, the application's performance is hardly affected. Furthermore, our authentication strategy is nonfragile, which can continuously authenticate the data stream even if a preceding data block is tampered with, and thus, provides stronger tamper detection capability at the block level.

Modeling-based analysis reveals how the false positives and false negatives of our proposed scheme can be tuned. To evaluate our proposed scheme, we have implemented a prototype system and evaluated the system in an LAN and over the Internet. In the LAN, the experiments are performed under various network jitter patterns including normal and burst, and packet loss on both UDP- and TCP-based streams. The results show that the proposed scheme is robust to packet loss and can succeed when various network jitter patterns exist. In addition, little impact is found on the performance of the application. Over the Internet, the experiments are performed on nodes with 16-hop distance. The results show that our scheme is practical on the Internet.

Besides the feature of simultaneous data transparency and zero communication overhead, *DaTA* also differs from existing schemes in the hash chaining usage [13], [14], [8]. Generally, most existing strategies based on hash chaining apply the hash chain across different data blocks to detect block alteration or loss. But a block loss or alteration will result in uncertainty for authenticating all subsequent data blocks. Thus, these strategies are *fragile*. But *DaTA* can continuously authenticate subsequent data blocks even a preceding block is changed or lost.

DaTA is feasible for the application in which the content is highly sensitive and cannot tolerate any changes, and/or the situations in which communication overhead needs to be minimized. Besides authentication, another potential application is to convey covert information as indicated by [15]. For example, in the battlefield, sensors can rely on its born information to authenticate if the received information (such as a command) is from a legal source.

The rest of the paper is organized as follows: Section 2 describes some related work. Section 3 presents our proposed scheme, and Section 4 analyzes the accuracy of the proposed scheme. Section 5 further discusses some design issues. Sections 6 and 7 present the experimental results in local LAN and over the Internet, respectively. Section 8 concludes the paper.

2 Related Work

Watermarking has been widely studied to protect the copyright or the ownership of multimedia objects. Many strategies [16], [17] have been proposed to watermark still multimedia objects, such as an image. Recently, image watermarking has also been extended to authenticate audio and video objects. For the MPEG movie, Hartung and Girod proposed an approach [18] that leverages the DCT transformation procedure to embed the watermark. Lu et al. [7] proposed to combine the content authentication and ownership authentication. More recently, watermarking techniques have also been extended to watermark other data, such as numerical or categorical database [19], [20].

Existing work on stream data and multicast authentication mainly uses MACs and secret-key cryptography. A trivial solution is to generate an MAC per packet and send with the packet together. Various approaches [14], [21] 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 [14] presented a scheme that uses signature amortization over a hash chain. In [21], a Merkle hash tree is used to amortize a signature over multiple packets.

Another line of work [6], [9], [11], [22] was addressing the packet loss problem. In [11], an 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. [23] proposed distillation codes to cope with packet loss and packet pollution.

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.

3 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* (*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. Packet boundary recognition issue is discussed with regard to packet loss, packet fragmentation, and out-of-order delivery then.

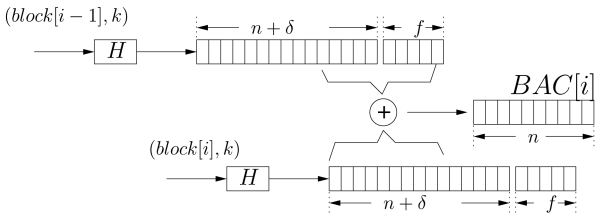

Fig. 1. Generating block authentication code.

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 [24] or SHA [25].
4. X,Y represent 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, where $0 \leq f \leq n$.

3.1 BAC Generation

Fig. 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 string of $n + \delta$ bits, as shown in Fig. 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 to get a binary string of $2n$ bits, which we call the *second-digest*.
3. The first n bits of the second-digest are 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. The entire procedure is as shown in Fig. 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 the flag). 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 that 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 trade-off 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 $0 < \delta < n$.

3.2 Timing-Based BAC Embedding and Extraction

After the BAC is generated, the next step is to embed the BAC. Different from existing strategies where the authentication information is sent out-of-band or embedded into the original data before data transmission, in *DaTA*, the BAC is embedded by adjusting the interpacket delay. In the following context, we present how the BAC bits can be embedded and extracted without touching the content of the packet.

Given any packet flow $P_1, \ldots, P_m$ with time stamps $t_1, \ldots, t_m$, respectively ($t_i < t_j$ for $1 \leq i < j \leq m$), we first independently and randomly choose $2r$ ($r > 0$) distinct packets: $P_{z_1}, \ldots, P_{z_{2r}}$ ($1 \leq z_k \leq m - d$ for $1 \leq k \leq 2r$), and create $2r$ packet pairs: $\langle P_{z_k}, P_{z_k+d} \rangle (d \geq 1,\ k = 1, \ldots, 2r)$, where d is the packet pair distance.

The Interpacket Delay (IPD) between P_{z_k+d} and P_{z_k} is defined as

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

Because all $P_{z_k} (k = 1, \ldots, 2r)$ are selected independently, the $ipd_{z_k,d}$ is independent from each other. Since each P_{z_k} is randomly and independently selected through the same process, $ipd_{z_k,d}$ is identically distributed no matter what interpacket timing distribution the packet flow $P_1, \ldots, P_m$ may have. Therefore, $ipd_{z_k,d}$ $(k = 1, \ldots, 2r)$ is independent and identically distributed (*iid*).

We then randomly divide the $2r$ IPDs into two distinct groups of equal size. Let $ipd_{1,k,d}$ and $ipd_{2,k,d}$ $(k = 1, \ldots, r)$ denote the IPDs in group 1 and group 2, respectively. Apparently, both $ipd_{1,k,d}$ and $ipd_{2,k,d}$ $(k = 1, \ldots, r)$ are *iid*. Therefore, $\mathrm{E}(ipd_{1,k,d}) = \mathrm{E}(ipd_{2,k,d})$ and $\mathrm{Var}(ipd_{1,k,d}) = \mathrm{Var}(ipd_{2,k,d})$.

Let

$$Y_{k,d} = \frac{ipd_{1,k,d} - ipd_{2,k,d}}{2}\ (k = 1, \ldots, r). \tag{2}$$

Then, we have $\mathrm{E}(Y_{k,d}) = (\mathrm{E}(ipd_{1,k,d}) - \mathrm{E}(ipd_{2,k,d}))/2 = 0$. Because $ipd_{1,k,d}$ and $ipd_{2,k,d}$ are *iid*, and $Y_{k,d}$ is also *iid*. We use $\sigma^2_{Y,d}$ to represent the variance.

We represent the average of r $Y_{k,d}$'s as

$$\overline{Y_{r,d}} = \frac{1}{r}\sum_{k=1}^{r} Y_{k,d}. \tag{3}$$

TABLE 1
Authentication Scenarios

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

Here, $\overline{Y_{r,d}}$ represents the average of a group of normalized IPD differences, and we call r the *redundancy level*. According to the property of variance of independent random variables, we have $\mathrm{Var}(\overline{Y_{r,d}} = \sigma^2_{Y,d}/r$. Because $\mathrm{E}(Y_{k,d}) = 0$ $(k = 1, \ldots, r)$, $\mathrm{E}(\overline{Y_{r,d}}) = 0$. Because $Y_{k,d}$ is symmetric, $(k = 1, \ldots, r)$, $\overline{Y_{r,d}}$ is also symmetric. Therefore, the distribution of $\overline{Y_{r,d}}$ is symmetrically centered around 0.

Since the distribution of $\overline{Y_{r,d}}$ is symmetric and centered around 0, the probabilities of $\overline{Y_{r,d}}$ to be positive and negative are equal. If we decrease or increase $\overline{Y_{r,d}}$ by an amount $a > 0$, we can shift its distribution to the left or right by a so that $\overline{Y_{r,d}}$ will be more likely to be negative or positive. This gives us a way to embed and extract a single binary bit probabilistically as follows:

The BAC embedding is done bit by bit. To embed a bit 0, the sender decreases $\overline{Y_{r,d}}$ by a so that $\overline{Y_{r,d}}$ will have >0.5 probability to be less than 0. To embed a bit 1, the sender increases $\overline{Y_{r,d}}$ by a so that $\overline{Y_{r,d}}$ will have >0.5 probability to be greater than 0. By the definition in (3), the decrease or increase of $\overline{Y_{r,d}}$ can be easily achieved by decreasing or increasing each of the r $Y_{k,d}$ by a. By the definition in (2), the decrease of $Y_{k,d}$ by a can be achieved by decreasing each $ipd_{1,k,d}$ by a and increasing each $ipd_{2,k,d}$ by a; the increase of $Y_{k,d}$ by a can be achieved by increasing each $ipd_{1,k,d}$ by a and decreasing each $ipd_{2,k,d}$ by a.

To extract the BAC, the receiver calculates $\overline{Y_{r,d}}$ as it receives the data packets. To extract an embedded bit, the receiver checks whether $\overline{Y_{r,d}}$ is less than or greater than 0. The extraction of embedded binary bit is 1 if the value of $\overline{Y_{r,d}}$ is greater than 0, or 0 if the value of $\overline{Y_{r,d}}$ is less than or equal to 0. It is easy to see that probability of correct extraction is always greater than that of wrong extraction.

3.3 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 3.1). Then, the extracted BAC is compared with the generated 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. Table 1 lists all four possible scenarios and their implications.

Note that in our scheme, every received data block is authenticable independently, which is based on the f-bit 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, where the extracted and generated BACs are completely matched, the current data block is authenticated to be genuine.
- In case 2, the failure of the second-part authentication indicates that the current data block has been changed; although the first part authentication succeeds, we cannot conclude that the preceding block is genuine.
- In case 3, the success of the second-part authentication indicates that the current block is genuine. Therefore, the first-part authentication failure indicates that the preceding data block has been changed/deleted.
- In case 4, the authentication failure on both parts strongly suggests that the current data block has been changed. The status of the preceding block is uncertain.

By 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.

The above analysis shows that *DaTA* has the following features:

- It can detect any tampering, including content alteration and packet/block deletion.
- It is able to locate the change at the data block level so that in some circumstances, the application can make decisions (e.g., retransmission) accordingly.
- It can ensure 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.
- The scheme is able to authenticate the content source in addition to the content itself, which is achieved in the second part of the authentication procedure.

3.4 Block Boundary Recognition

In our scheme, it is important that the receiver can correctly delineate 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 will not be consistent with the sender side. This makes authentication impossible.

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 interpacket delay adjustment is limited by a, where a is the *adjustment size* (see

Section 3.2). Thus, if we find the maximum interpacket 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 tolerate network fluctuations. Thus, we choose 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. Existing research [26] and measurement [27] find that packet loss is less than 0.1 percent on the Internet backbone. However, our strategy is very sensitive to packet loss since it 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 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 is how the extra delay will affect application performance. Since the additional delay introduced by our scheme is always in the granularity of several milliseconds and the scheme only adjusts relevant interpacket timing, the delay added is not cumulative, Thus, the packet rate is barely affected. Neither is 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 [28], 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 percent packets arrive out of order [29]. Thus, we don't consider its effect in this study.

4 *DaTA* Analysis

In *DaTA*, the authentication information is generated based on the content of the data block (see Section 3.1), which we call *content BAC*. The BAC embedded in (or extracted from) the interpacket timing is called the *reference BAC*. At the receiver side, when the receiver receives the stream flow, the reference BAC extracted from the interpacket 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 the ideal case, $C_1 = C_2 = C$. In the 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 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

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

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 has exactly $0 \leq k \leq n$ bits different from C, then we have

$$X_k = \binom{n}{k} p_1^{n-k} (1 - p_1)^k \times p_2^{n-k} (1 - p_2)^k,$$

and

$$Y_k = \binom{n}{k} p_2^{n-k} (1 - p_2)^k.$$

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

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

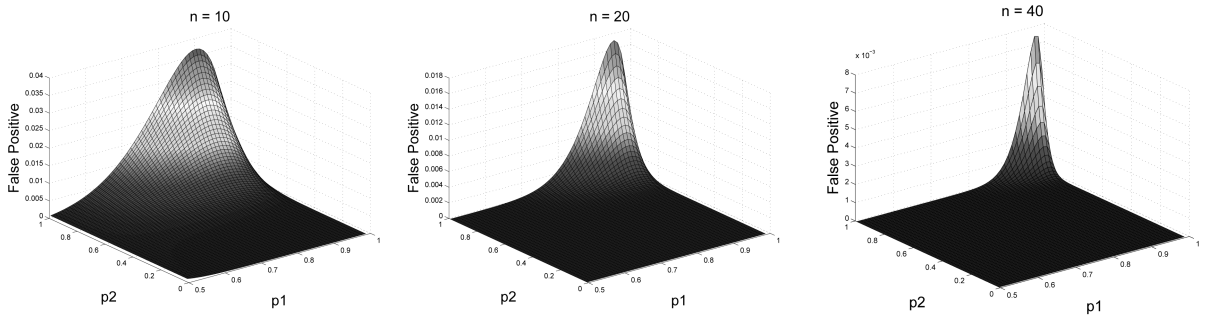

Fig. 2. Distribution of false positive rate for different n.

Based on (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 (6). Fig. 2 shows the distribution of the false positive rate when n is 10, 20, and 40, respectively. 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.

5 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.

5.1 Redundancy Level

In an ideal environment, where there is no network fluctuations, embedding a BAC bit using two packet pairs (four 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 it easier to locate a changed packet. This provides convenience and reduces cost for a remedy: if the application decides to retransmit the tampered block, only 96 packets need to be retransmitted.

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 (3), the larger the redundancy level r, the more centralized to 0 of the $\overline{Y_{r,d}}$ distribution. In another word, the instance of $\overline{Y_{r,d}}$ 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 1,920 packets are needed. If any of these packets is lost or tampered with, 1,920 packets have to be retransmitted. Thus, we should always use an appropriate redundancy level. We further explore this via experiments in Section 6.

5.2 Trade-Off between Redundancy Level and BAC Length

In the above discussion, 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. In other words, 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 6.

5.3 Bidirectionally Embedding a Bit

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 packets of the first and second packet pair, respectively. That means these 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 desirable.

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 nonzero-second buffer (such as encoding buffer) at the sender side, this optimization can be applied.

5.4 *DaTA* Vulnerability

Our scheme is vulnerable to the DoS attacks since if a malicious user is aware of the data-transparent authentication, and keeps disturbing the data flow, at the receiver side, the received data packets will be authenticated to be changed, and will not be accepted. However, this type of attack cannot make the receiver accept faked information.

6 Local Experiments

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 five 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. In the next section, we experiment our system over the Internet.

6.1 Experiment Setup

In local experiments, two different data flows are examined. The first is a 160 second streaming video, which is encoded at 300 Kbps with a rate of 15 frames/second. There is a total of 10,299 packets. The second is the real stock workload captured from [3] on April 13th, 2005. It contains the continuous data flow of 100 different stocks for 1 hour. Due to the uncontrollable data source for this stock workload, we first capture the workload, and then, use TCP Replayer [30] to perform the experiments.

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 [31] 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 are streamed to the destination machine, which runs Windows XP with 1.8 GHz CPU 1.8 and 512 MB memory. Since the results based on the stock workload reflect similar trends as on the video streaming, and video streaming workload is more sensitive to network jitter, in the following context, we omit its corresponding figures for the stock workload for brevity.

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

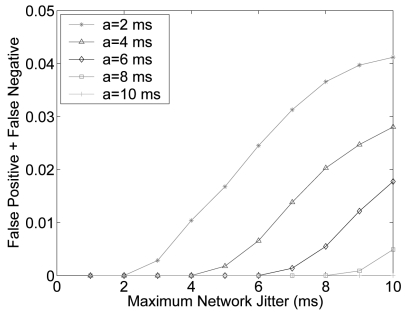

Fig. 3. Boundary detection: uniform jitter (1 percent).

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 1,000 if not explicitly noted. A BAC length of 24 strives to achieve the balance between the false positive and the false negative. We set the packet distance as 1 and the distance between the selected packet pair (for embedding) as 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 interpacket 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 percent) and the extreme case (100 percent). A jitter probability of 1 percent means that among 100 packets, only one packet is randomly selected to be delayed and this emulates the normal network. A 100 percent 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.

6.2 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 interpacket 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 interpacket 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. Figs. 3 and 4 show the boundary detection error rates when the network fluctuations follow a uniform distribution for the video stream. In the figures, the x-axis represents the network *jitter* in *millisecond*, while the y-axis denotes the sum of *false positive* and *false negative*. We simply refer to them as the *error rate* in the following context. *False positive* is defined as the error when a nonboundary 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 interpacket delay in the original packet flow is 69 ms, and the average is 15.221 ms,

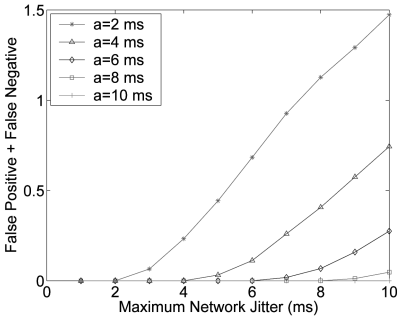

Fig. 4. Boundary detection: uniform jitter (100 percent).

we apply the adjustment size a from 1 to 10 ms and shown on the figures with all even values.

In Figs. 3 and 4, packet jitter occurs with probability 1 and 100 percent, 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 Fig. 4 that 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 [27]. We will further test this in the Internet experiments.

Figs. 9 and 10 show the result when the network jitter follows a normal distribution with the probability of 1 and 100 percent, respectively. The maximum jitter value is in the range of 2-8 ms. A similar trend is reflected in these two figures as the previous ones. Thus, we do not show the results when the network jitter follows the normal distributions in the following experiments for brevity.

Second, we study the impact of packet loss on boundary detection. We artificially delete packets from the data block randomly with the deletion probability of 1/100, 1/1,000, 1/10,000, 1/100,000, and 1/1,000,000 in the entire packet flow, respectively. Experiments are repeated 100 times to check if a false boundary is found. Fig. 5 shows the result. Roughly, adjustment sizes do not make much difference. As reported in [27], the regular packet loss rate on the Internet is less than 0.1 percent. So, an a of 4 ms can detect the boundary with more than 99 percent accuracy.

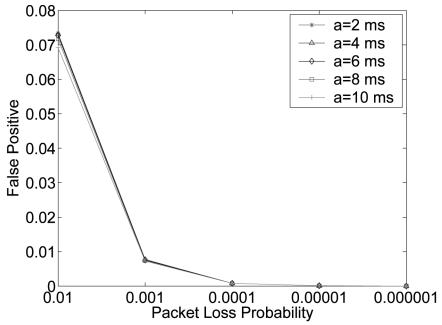

Fig. 5. Boundary detection: packet loss.

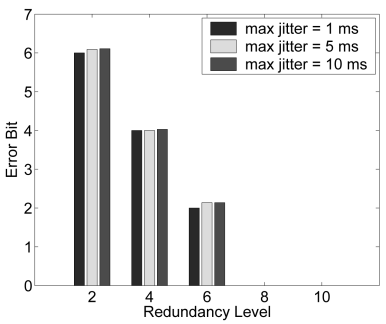

Fig. 6. Redundancy level: uniform jitter (1 percent).

6.3 Redundancy Level/BAC Length

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

We first conduct experiments 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 2,000 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 percent.

We define the *error bit* as the total number of error bits in the experiments in average. Figs. 6 and 7 show the number of error bits when the redundancy level varies 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 percent packet jitter probability.

As discussed in Section 5.2, 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 1,000 data packets. Given a block size of 1,000 packets, there are a few pairs of the redundancy level and BAC length, including (6, 24), (7,23), (8,20), (9,18), and (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 percent, experiments are

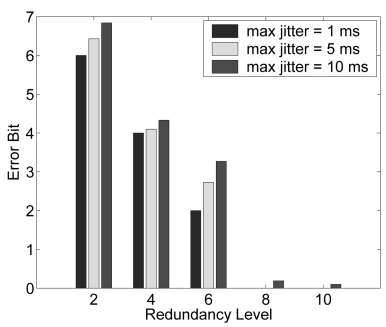

Fig. 7. Redundancy level: uniform jitter (100 percent).

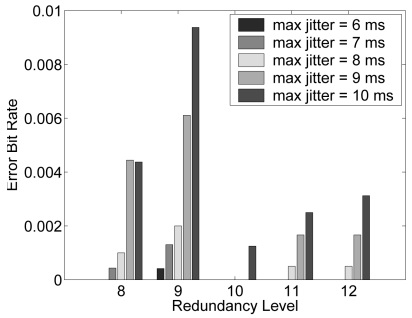

Fig. 8. Redundancy level versus BAC length: uniform jitter (100 percent).

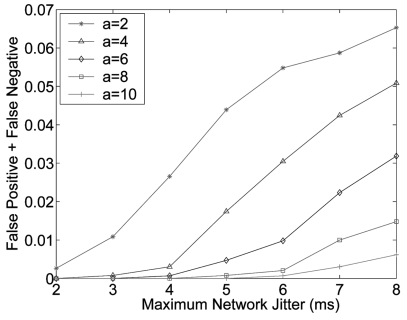

Fig. 9. Boundary detection: normal jitter (1 percent).

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 percent, the error bit rate is always 0. When the packet jitter probability is 100 percent, Fig. 8 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.

The above experiments confirm that there is indeed a trade-off. Thus, for a fixed block size, the appropriate BAC length and redundancy level should be found to achieve the best authentication performance.

6.4 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, in the case of network fluctuations, the extracted BAC may happen to be the same as the new content BAC, which leads to a false positive. The following two experiments are performed to evaluate these situations by either deleting a packet, or changing a bit in a packet randomly.

First, we run an experiment upon packet loss. In 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

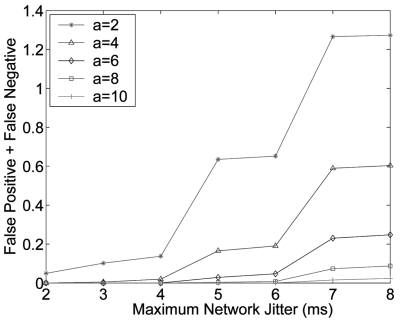

Fig. 10. Boundary detection: normal jitter (100 percent).

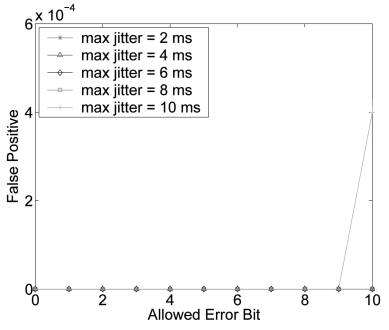

Fig. 11. Locate packet loss: uniform jitter (1 percent).

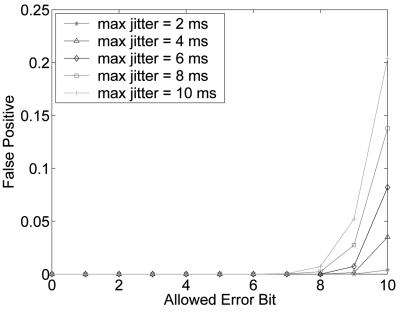

Fig. 12. Locate packet loss: uniform jitter (100 percent).

network jitter using the uniform perturbation where the max delay varies from 1 to 10 ms and for each distribution. The experiment runs for 10,000 times for each jitter setting to randomly select a packet to drop.

Figs. 11 and 12 show the result when the jitter is in uniform distribution with 1 and 100 percent probabilities. In these 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 that among the 24 BAC bits, one-third 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

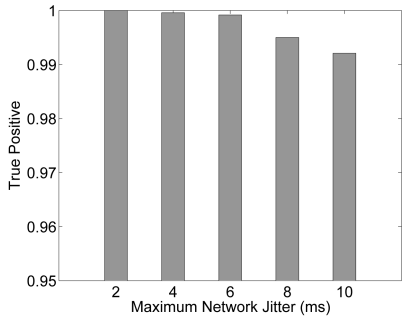

Fig. 13. Authentication accuracy: uniform jitter (100 percent).

experiment. The result shows that the alteration can be detected with 100 percent accuracy, with the uniform distribution of either 1 or 100 percent.

6.5 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 percent probability.

When the jitter follows a uniform distribution with a probability of 1 percent, our scheme always achieves 100 percent true positive. Thus, we only show the true positive for the authentication scheme when the packet jitter probability is 100 percent in Fig. 13. As shown in the figure, when the maximum jitter increases beyond 4 ms, our scheme cannot achieve a 100 percent true positive. However, even when the maximum jitter is 10 ms, our scheme still achieves a true positive above 99 percent.

6.6 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 percent. 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 time stamp difference and interpacket delay.

Fig. 14 shows the time stamp 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

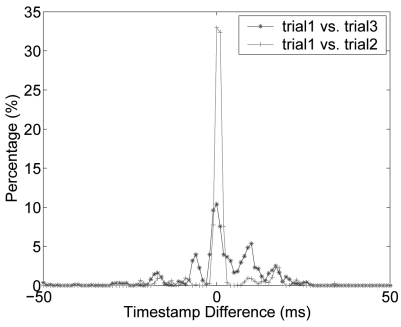

Fig. 14. Time stamp difference of authenticated flow.

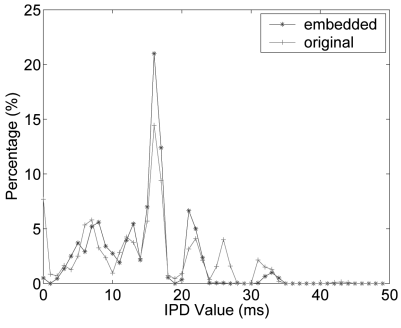

Fig. 15. IPD breakdowns.

between *trial1* and *trial3* represents the experimental network jitter and the additional jitter introduced by our authentication scheme. As shown in the figure, 81 percent time stamp 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 nonzero second buffer.

Fig. 15 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.

7 Internet Experiments

To study the performance of our proposed scheme in practice, we further conduct experiments over the Internet.

The movie played in the previous section is now played over the Internet. The experimental platform includes two machines. The first runs the media server, and is located in Fairfax, Virginia. The second runs the media player, and is located in Ames, Iowa. *traceroute* indicates that there are 16 hops between two machines.

First, we study the impact of network fluctuations to the boundary detection. Since local experiments suggest an adjustment size a of 4, we test such a setup in 10 runs. Fig. 16 shows the result. As indicated by the figure, both the false positive and false negative are larger than their counterparts in Fig. 3. But their absolute percentages are still very small. When we further increase the adjustment size a, we cannot detect any false positive and false negative. This suggests that the adjustment size of our proposed scheme should be adjusted according to the present network situations when it is used and a 4 ms is a good start point.

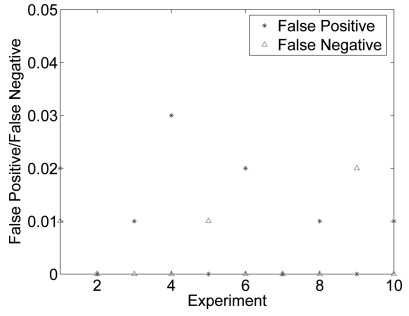

Fig. 16. Internet experiment: boundary detection ($a = 4$).

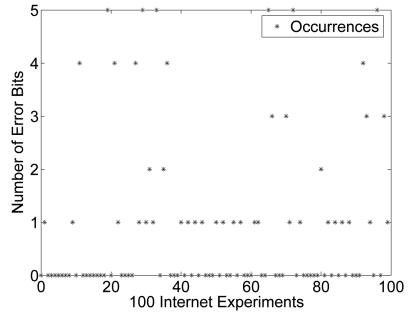

Fig. 17. 100 Internet experiments.

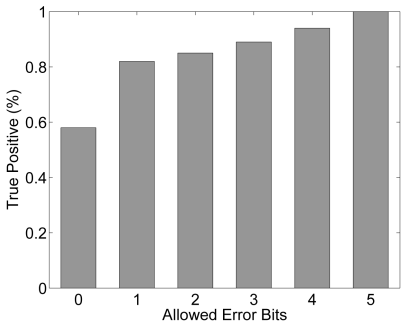

Fig. 18. True positive in 100 Internet experiments.

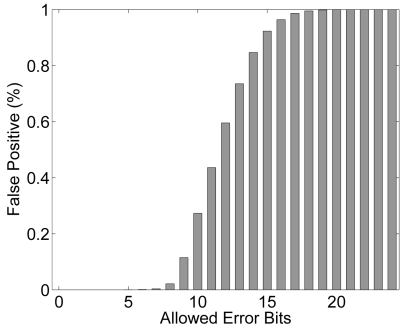

Fig. 19. Cumulative false positive in 9,900 decodings.

Second, we study the authentication accuracy. Since the previous experiments indicate that for a BAC length of 24, an adjustment size of 4 ms and a redundancy level of 8 are good for our experimental data streams, we use these parameter setups to perform tests. We repetitively performed the experiments for 100 times.

Fig. 17 shows the number of error bit occurrences when we decode the received data stream. The result indicates that at most there are five error bits, and most of the experiments show an exact match.

Fig. 18 shows the true positive when the number of allowed error bits varies. As shown in the figure, exact match accounts for 58 percent, and the true positive is larger than 80 percent if we allow only one error bit. When the allowed error bit number is 5 or larger, the system achieves a 100 percent true positive rate.

Since with a larger number of allowed error bits, the false positive also increases with the true positive, we examine the false positive of experimented streams as follows: For each of these 100 data streams collected from movie playbacks, we also randomly generated another 99 packet flows to decode the received data stream. If a match is found, it is a false positive. On decoding these 9,900 flows, Fig. 19 shows the cumulative false positive and an allowed error bit of 5 or less produces no false positive. Figs. 18 and 19 suggest that an allowed error bit number of 4 or 5 is sufficient in the Internet for our proposed authentication strategy.

8 Conclusion

Data streams have been used in many Internet applications, such as grid computing and streaming media. More and more applications like these demand a reliable and effective authentication mechanism to ensure the genuineness of data streams transferred over the Internet. Although plenty of research work has been conducted, existing work shares the characteristics of either slightly changing the original data, or sending the authentication information out-of-band, neither of which is desirable when the data carry sensitive information or when the data are transmitted to mobile devices. 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 and show that our scheme is robust and practical.

Acknowledgments

The authors would like to thank anonymous reviewers for their helpful comments on this paper. The work is partially supported by the US National Science Foundation (NSF) grant CNS-0746649 and US AFOSR grant FA9550-09-1-0071. An earlier version [32] of this paper has been published in the proceedings of SecureComm 2006.

References

[1] "National Hurricane Center," http://www.nhc.noaa.gov/, 2010.
[2] "National Oceanic and Atmospheric Adminstration," http://www.nesdis.noaa.gov/, 2010.
[3] CiberStock Quote & Chart/Share Price, http://www.advfn.com/, 2010.
[4] M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and Analysis of a Streaming Media Workload," *Proc. Third USENIX Symp. Internet Technologies and Systems*, Mar. 2001.
[5] M. Chen, Y. He, and R. Lagendijk, "A Fragile Watermark Error Detection Scheme for Wireless Video Communications," *IEEE Trans. Multimedia*, vol. 7, no. 2, pp. 201-211, Apr. 2005.
[6] P. Golle and N. Modadugu, "Authenticating Streamed Data in the Presence of Random Packet Loss," *Proc. SPIE Security and Watermarking of Multimedia Contents*, Jan. 2001.

[7] C. Lu, H.M. Liao, and L. Chen, "Multipurpose Audio Watermarking," *Proc. Int'l Conf. Pattern Recognition (ICPR '00)*, Sept. 2000.
[8] S. Miner and J. Staddon, "Graph-Based Authentication of Digital Streams," *Proc. IEEE Symp. Security and Privacy*, 2001.
[9] A. Pannetrat and R. Molva, "Real Time Multicast Authentication," *Proc. Network and Distributed System Security Symp. (NDSS '03)*, Feb. 2003.
[10] J. Parlk, E. Chong, and H. Siegel, "Efficient Multicast Packet Authentication Using Signature Amortization," *Proc. 2000 IEEE Symp. Security and Privacy*, 2000.
[11] A. Perrig, J.D. Tygar, D. Song, and R. Canetti, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," *Proc. IEEE Symp. Security and Privacy*, 2000.
[12] L. Qiao and K. Nahrstedt, "Watermarking Method for mpeg Encoded Video: Towards Resolving Rightful Ownership," *Proc. IEEE Int'l Conf. Multimedia Computing and Systems (ICMCS '98)*, June 1998.
[13] S. Ben-David, J. Gehrke, and D. Kifer, "Detecting Change in Data Streams," *Proc. 30th Very Large Data Bases (VLDB) Conf.*, Aug. 2004.
[14] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto '97)*, 1997.
[15] S. Cabuk, C.E. Brodley, and C. Shields, "Ip Covert Timing Channels: Design and Detection," *Proc. ACM Conf. Computer and Comm. Security (CCS)*, Oct. 2004.
[16] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman, "Electronic Marking and Identification Techniques to Discourage Document Copying," *Proc. IEEE INFOCOM*, June 1994.
[17] J. Fridrich and M. Du, "Images with Self-Correcting Capabilities," *Proc. IEEE Int'l Conf. Image Processing*, 1999.
[18] F.H. Hartung and B. Girod, "Watermarking of mpeg-2 Encoded Video without Decoding and Reencoding," *Proc. SPIE/ACM Conf. Multimedia Computing and Networking*, Feb. 1997.
[19] Z. Liu, X. Li, and Z. Dong, "Multimedia Authentication with Sensor-Based Watermarking," *Proc. ACM Multimedia Workshop Security*, Sept. 2002.
[20] L.C. Yung and C.S. Fu, "A Robust Image Authentication Method Distinguishing jpeg Compression from Malicious Manipulation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 2, pp. 153-168, Feb. 2001.
[21] C.L. Wong and S.S. Lam, "Digital Signatures for Flows and Multicasts," *Proc. Int'l Conf. Network Protocols (ICNP)*, 1998.
[22] P. Rohatgi, "A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication," *Proc. Sixth ACM Conf. Computer and Comm. Security (CCS)*, Nov. 1999.
[23] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. Tygar, "Distillation Codes and Applications to dos Resistant Multicast Authentication," *Proc. Network and Distributed System Security Symp. (NDSS '04)*, Feb. 2004.
[24] "Rfc 1321—The md5 Message-Digest Algorithm," http://www.faqs.org/rfcs/rfc1321.html, 2010.
[25] National Institute of Standards and NIST FIPS PUB 180 Technology, "Secure Hash Standard," US Department of Commerce, May 1993.
[26] L. Ciavattone, A. Morton, and G. Ramachandran, "Standardized Active Measurements on a Tier 1 ip Backbone," *IEEE Comm. Magazine*, no. 41, no. 6, pp. 90-97, June 2003.
[27] "Global ip Network Home," http://ipnetwork.bgtmo.ip.att.net/pws/, 2010.
[28] C. Shannon, D. Moore, and K. Claffy, "Characteristics of Fragmented ip Traffic on Internet Links," *Proc. ACM Internet Measurement Workshop*, Nov. 2001.
[29] Y. Zhang, V. Paxson, and S. Shenker, "The Stationarity of Internet Path Properties: Routing, Loss, and Throughput," ACIRI, technical report, 2000.
[30] "Tcpreplay: Pcap Editing and Replay Tools for *nix," http://tcpreplay.sourceforge.net/, 2010.
[31] "Apple Darwin Streaming Server," http://developer.apple.com/darwin/projects/, 2010.
[32] S. Chen, S.P. Chen, X. Wang, and S. Jajodia, "Data-Data Transparent Authentication without Communication Overhead," *Proc. Second Int'l Conf. Security and Privacy in Comm. Networks (SecureComm '06)*, Aug. 2006.

Songqing Chen (M'03) received the PhD degree in computer science from the College of William and Mary, Williamsburg, Virginia. He is an assistant professor of computer science at George Mason University, Fairfax, Virginia. His research interests include the Internet content delivery systems, Internet measurement and modeling, operating systems and system security, and distributed systems and high-performance computing. He is a member of the IEEE. He is a recipient of the US National Science Foundation (NSF) CAREER Award and the AFOSR YIP Award.

Shiping Chen received the PhD degree in information technology from George Mason University, Fairfax. He is an active researcher and practitioner in information security field. His expertise and interest include data security and privacy, network attack tracing and prevention, and database security. He is currently working for Sybase, Inc., Dublin, California, and focusing on developing the next-generation secure database management systems. He is a senior member of the IEEE.

Xinyuan Wang received the BS degree in computer science from Peking University, the MS degree in computer science from Chinese Academy of Space Technology, and the PhD degree in computer science from North Carolina State University in 2004 after years professional experience in networking industry. He is currently an assistant professor in the Department of Computer Science at George Mason University. His main research interests are around computer network and system security—including malware analysis and defense, attack attribution, anonymity and privacy, VoIP security, and digital forensics. He received the 2009 National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award.

Zhao Zhang received the BS and MS degrees in computer science from Huazhong University of Science of Technology, China, in 1991 and 1994, respectively, and the PhD degree in computer science from the College of William and Mary in 2002. He is an associate professor of computer engineering at Iowa State University. His research interests include computer architecture, parallel and distributed systems, and architectural support for system security. He is a member of the IEEE and a senior member of the ACM.

Sushil Jajodia received the PhD degree from the University of Oregon, Eugene. He is a university professor, BDM international professor of information technology, and the director of the Center for Secure Information Systems at the George Mason University, Fairfax, Virginia. The scope of his research interests encompasses information systems security, distributed databases, and temporal databases. He has authored six books, edited 34 books and conference proceedings, and published more than 350 technical papers in the refereed journals and conference proceedings. He received the 1996 Kristian Beckman Award from IFIP TC 11 for his contributions to the discipline of Information Security, 2000 Outstanding Research Faculty Award from Mason's Volgenau School of Information Technology and Engineering, and 2008 ACM SIGSAC Outstanding Contributions Award for his research and teaching contributions to the information security field and his service to ACM SIGSAC and the computing community.