A Novel Unbalanced Tree Structure for Low-Cost Authentication of Streaming Content on Mobile and Sensor Devices

Thivya Kandappu†*, Vijay Sivaraman†, Roksana Boreli*
† School of Electrical Engineering & Telecommunications, UNSW, Sydney, Australia.
* NICTA, Sydney, Australia.
Emails: {*t.kandappu@student.unsw.edu.au, vijay@unsw.edu.au, roksana.boreli@nicta.com.au*}

***Abstract*—We consider stored content being streamed to a resource-poor device (such as a sensor node or a mobile phone), and address the issue of authenticating such content in real-time at the receiver. Per-packet digital signatures incur high computational cost, while per-block signatures impose high delays. A Merkle hash tree combines the benefits of the two by having a single signature per-block (at the root of the tree), while allowing immediate per-packet verification by following a hash-path logarithmic in the number of packets. In this paper we explore how the structure of the Merkle tree can be adapted to improve playback performance for streaming content. We make three specific contributions: First, we develop a new unbalanced authentication tree structure called the α-leaf tree that is a generalisation of the Merkle tree. We derive several key properties of this tree, highlighting the impact of the imbalance parameter α. Second, we present a theoretical model to quantify the benefits of our unbalanced tree structure in reducing start-up delays for streaming applications by optimally readjusting the burden of authentication across packets. Third, we validate via simulation the suitability of our scheme to two representative applications, namely audio streaming to a low-cost sensor device and video streaming to a mobile phone, and demonstrate that start-up delays can be reduced without affecting stall rates. We believe our authentication tree structure is of importance both theoretically, as a generalisation of the Merkle hash tree, as well as practically, for applications requiring real-time verification of streaming content.**

***Index Terms*—Content Authentication, Digital Signature, Merkle Tree.**

I. Introduction

Entertainment content is increasingly being streamed to user devices ranging from computers and televisions to mobile phones and wearable media-players. Sandvine's Spring 2011 report [1] shows that streaming entertainment (e.g. from Net-Flix) now accounts for nearly half of all Internet traffic in the US. Mobile devices are increasingly consuming streaming content too, with YouTube reporting over 200 million views per-day from such devices [2]. Given this staggering growth in consumption of streaming content on a wide range of user devices with varying communication, computing, and storage capabilities, an important question concerns the ability to validate the content in real-time. Authentication is needed to establish the trustworthiness of the source and content of the received messages, protecting against misleading content injected by a masquerading server or tampering by a malicious participant in the peer-to-peer network redistributing the content [3]. The problem is particularly challenging on low-cost mobile devices that are inherently resource-poor, yet more vulnerable to attack over the insecure and potentially lossy wireless channel.

Existing content delivery systems support elementary authentication services of the *sign-all* type [4], [5], [6], wherein packets are individually signed by their source. Since packets can be individually verified immediately upon receipt, this approach is fast in terms of playback at the receiver. However, it incurs high computation overhead, since a stream consisting of n packets requires n signature verifications at the receiver. One way to mitigate this overhead is by amortizing the signature over several packets, such as the *sign-block* approach used in [7] for 3D streaming. However, such a scheme introduces authentication delay, since the receiver has to wait for the entire block before being able to verify the content, and is not loss resilient, since even one packet missing from the block renders the whole block unverifiable at the receiver.

Merkle hash trees [8] combine the benefits of the sign-all and sign-block approach by building a binary hash tree on a block of packets and signing only the root. Only one digital signature verification is required per-block, while individual packets can be verified by hashing along the authentication path from the leaf of the tree to the root containing the digital signature. Since the digital signature (which has typically one to two orders of magnitude greater computational cost than hashing [9]) is performed only once per-block, its cost is amortised over the block. Further, the authentication path for each packet (logarithmic in the number of packets in the block) can be embedded with the packet, allowing for immediate verification that is resilient to packet loss.

In this paper we explore if it is possible to improve upon the Merkle tree for use in authenticating streaming content. Since the Merkle tree is organised as a balanced binary tree, each packet has the same authentication delay. For example, an audio clip of say 1000 packets would be organised as a Merkle tree of height 10. On a low-cost receiver (e.g. a sensor node running TinyOS), each hash takes around 5 ms, and traversing the authentication path would therefore require roughly 50 ms, contributing to increased playback delay. Our proposal is to generalise the Merkle tree to make

it unbalanced, such that authentication paths are short for initial packets, and progressively get longer for later packets. This has two advantages: (a) It allows faster authentication of the initial packets, so playback can start earlier, improving user experience. Later packets can afford to have a longer authentication path (and hence higher authentication delay), since they do not have an immediate deadline for playback. (b) Users do not often watch entire streaming content. Studies of viewer abandonment trends [10] show that a third of viewers abandon playback within 30 seconds, and over half of video views are abandoned within 90 seconds. In such situations, shifting the authentication burden from initial to later packets (by gradually increasing authentication path length) can help reduce the authentication cost incurred prior to abandonment.

In the context of developing a scheme suited to real-time authentication of streaming content, our contributions are threefold:

1) We develop a novel authentication tree structure called the α-leaf tree, which is a generalized (unbalanced) version of the Merkle hash tree. The parameter α determines the degree of imbalance, enabling a range of choice from a balanced tree to a linear chain. We derive several key properties of this structure, such as tree height, path lengths, and path stretch. We also highlight a specific case of the α-leaf tree, called the Fibonacci-leaf tree, that is of special interest later in the paper.
2) We develop an analytical model that quantifies the efficacy of the α-leaf tree structure in real-time authentication of streaming content. We demonstrate that adjusting tree imbalance (via parameter α) can reduce playback delay for required playback performance, though it increases average computational cost for authentication. Our model allows easy evaluation of this trade-off for given system parameters.
3) We validate our model via simulation for two realistic scenarios, corresponding to audio streaming to a sensor device and video streaming to a mobile phone, and show that tree imbalance can be leveraged to substantially reduce start-up delays for given stall rate (or equivalently reduce stall rates for given start-up delay) compared to the balanced Merkle tree, demonstrating its value for practical application.

The rest of the paper is organized as follows. Section II describes background and related work. The structure and properties of the α-leaf tree are explained in Section III, while a model to analyse its performance is developed in Section IV. Our simulation study is detailed in Section V, and the paper is concluded in Section VI.

II. Background and related work

We briefly review existing schemes for authenticating data flows received over an untrusted network. The performance metrics we specifically focus on include:

- *Stalls during playback* - Once playback of the streaming content commences, the number of times the playback stalls (due to packets not being received or received packets not yet being authenticated) should be minimised in order to achieve good user experience.
- *Start-up delay* - This is the time, measured from receipt of the first packet, at which playback can commence. This delay should be as small as possible, but not so small that the playback keeps stalling at the receiver due to absence of authenticated content.
- *Tolerance to Packet losses* - The higher the dependency that the authentication scheme imposes among packets, the greater the impact of packet loss on verifiability of received content. It is desirable for the authentication scheme to be robust to packet loss.
- *Authentication overhead* - The communication cost (in units of bytes per packet) and computation overheads (in units of CPU load) that the authentication scheme imposes on the receiver should be kept low.

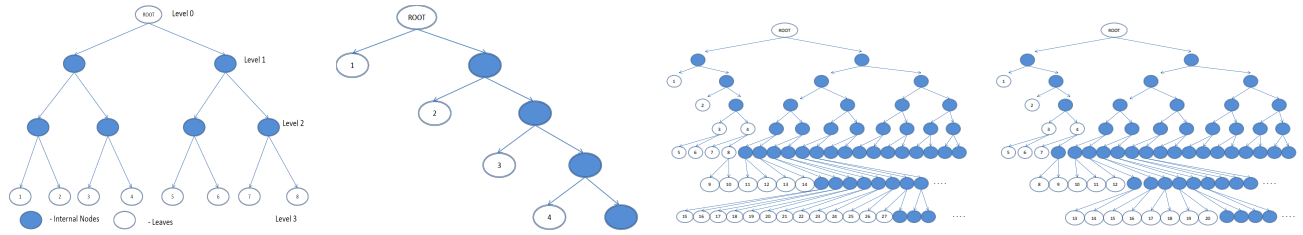
One common way to verify data integrity is to let the sender sign every packet using digital signatures [11]. This scheme is resilient to packet loss as each packet is independently verified. However, digital signature verification is an expensive operation incurring high communication and computation overhead on a per-packet basis at the receiver.

To mitigate this overheads the sender can divide the stored content into blocks, and append a signature to each block, as was done for 3D streaming in [7]. However, the receiver has to obtain the entire block before any packet can be verified hence incurs a delay before playback, and lacks loss resilience, as one lost packet in a block renders it unverifiable.

In hash chaining [12], the sender computes the hash of each piece and concatenates it with the preceding piece and the first piece is then digitally signed, constituting a commitment to the entire content. The receiver only needs to verify the signature for the first piece. This scheme lacks resilience to loss, since even one missing piece at the receiver makes all subsequent pieces unverifiable. One time and fast signature schemes such as [13], [14], [15] can reduce computation and communication overhead, but they are only secure for a short period of time [16].

SAIDA [17] uses erasure codes to amortize a signature over multiple packets. A block of n packets carries the encoded digests and signature of the block. The signature and digests are recoverable, if the receiver gets a minimum of n_e packets ($n_e \leq n$). Redundancy added to the data adds to computation and communication overheads, and incurs delay in the event of packet loss, since the receiver has to wait to receive at least n_e packets to retrieve and authenticate the lost packet. An enhancement to SAIDA, called eSAIDA [18], reduces the overhead at the expense of reducing the resilience to loss.

Perig et al. proposed TESLA [19] and EMSS [20] for secure multicast. TESLA uses MAC, generated using symmetric keys, to authenticate packets. The key used for generating the MACs during an epoch is revealed at a later epoch, and can be authenticated by verifying that it fits in the hash chain of keys. Though this scheme has low computation and communication overheads and is robust to packet loss, it requires loose time



(a) $\alpha = 0$ (b) $\alpha = 0.5$ (c) $\alpha = 0.25$ (d) Fibonacci-leaf tree

Fig. 1. Structure of α-leaf tree for (a) $\alpha = 0$ (b) $\alpha = 0.5$ (c) $\alpha = 0.25$ and (d) Fibonacci-leaf tree

synchronisation, and precludes redistribution of content by third-party servers (as in a peer-to-peer scenario). The schemes proposed in [20], [21] and [22] use a combination of hash functions and digital signatures to authenticate packets. They tolerate packet loss by sending multiple hashes with each packet and achieve relatively high verification rates at the cost of increased overhead and delayed authentication.

Wong and Lam studied content authentication in [23] for lossy multicast flows. They proposed Merkle hash tree [8] which builds a binary tree over a block of data packets. The hash of each data packet constitutes a leaf in the tree, and each internal node of the tree is obtained by concatenating the children and hashing the result. Only the root of the tree is digitally signed. With each data packet, its authentication path, namely the hashes from the leaf to the root, are transmitted. At the receiver, the signature of the root only needs to be verified once per tree. Thereafter, each received packet is verified by validating its hash path from its leaf position in the tree to the root (the number of hash operations is therefore logarithmic in the number of packets in the block). The Merkle tree has low cost and low delay, while being resilient to loss [24]. Karpinski et al. proposed skewed Merkle trees in [25], in order to extend the Merkle tree traversal algorithms without additional time and memory resources. They showed that extra nodes can be appended to a balanced Merkle tree without changing the time and space bounds of the traversal algorithm. In what follows, we generalise the Merkle tree structure and make it more suitable for real-time streaming applications.

III. The α-Leaf Authentication Tree

We now introduce our novel authentication unbalanced tree structure called the α-leaf tree, which is a generalisation of the Merkle hash tree. An α-leaf tree is an unbalanced structure, where the parameter α, $0 \leq \alpha \leq 0.5$, denotes the degree of imbalance. The motivation for the unbalanced tree is to provide shorter authentication paths for initial packets, so that they can be verified faster and playback can start quicker, while shifting the authentication burden towards latter packets, that can take more time to authenticate as they are not immediately needed for playback. As a secondary advantage, an imbalanced tree offers lower overall authentication cost when the user abandons the streaming data mid-stream (which happens very frequently as discussed earlier).

A. Constructing the α-leaf tree

The α-leaf tree is constructed such that the path length (from leaf to the root) gets progressively larger as we scan the leaves left to right. We achieve this as follows: suppose the tree has height h, where levels are numbered from top (level-0) to bottom (level-h). At any level i ($0 < i < h$), fraction α of the nodes (the leftmost ones) are made leaves, while the remaining $(1-\alpha)$ fraction of the nodes are expanded to have two children. As we will see below, this yields a tree whose imbalance can be controlled via the parameter α.

Specifically, the α-leaf tree (where $0 \leq \alpha \leq 0.5$) of height h is constructed as follows: The root (level-0) node has two children. If the number of nodes at level-i is denoted by N_i, then for $0 < i < h$, the leftmost $N_i^L (= \lfloor \alpha N_i \rfloor)$ of these nodes are made as leaves, while each of the remaining $N_i^I (= N_i - \lfloor \alpha N_i \rfloor)$ nodes are made to have two children each. At the bottom level-h, all N_h nodes are leaves.

It can be seen that α determines the degree of imbalance of the tree. Specifically, when $\alpha = 0$, the tree is balanced, since all leaves reside only at the bottom (level-h), as shown in Fig. 1(a), which corresponds to the Merkle tree. On the other extreme, when $\alpha = 0.5$, each level-i, where $0 < i < h$, has exactly one leaf, and the tree degenerates to a linear chain, as shown in Fig. 1(b).

When α takes an intermediate value, say 0.25, the resulting tree is imbalanced, in that successive groups of leaves have increasing path length to the root, as shown in Fig. 1(c). As we will show below, the height of the tree still remains logarithmic.

We also present here a special tree structure, called the **Fibonacci-leaf tree**, that can be counted as a special case of the α-leaf tree. The Fibonacci-leaf tree is constructed as follows: Let F_i denote the i^{th} Fibonacci number where $F_0 = 0$, $F_1 = 1$, and $F_i = F_{i-2} + F_{i-1}$ for $i \geq 2$. Then a Fibonacci-leaf tree of height h is such that at level-i, where $0 < i < h$, the number of leaves is F_{i-1} and the number of non-leaves (each of which will have exactly two children) is F_{i+2}. Needless to say, the top level-0 has no leaves and one non-leaf node, while the bottom level-h has $F_{h-1} + F_{h+2}$ leaves. Fig. 1(d) depicts the structure of a Fibonacci-leaf tree of height $h = 6$. For a Fibonacci-leaf tree of sufficiently large height, the ratio of leaves to nodes at level-i for sufficiently large i approaches 0.19, and hence the Fibonacci-leaf tree can (approximately) be viewed as an α-leaf tree where $\alpha \approx 0.19$.

The implications of the special tree will be discussed in the following sections.

B. Number of Leaves

In this section, we derive expressions for the number of leaves in the α-leaf tree and the Fibonacci-leaf tree. This expression can be used to find the appropriate tree height for the given file size (i.e, number of packets).

Lemma 1. *An α-leaf tree of height h has L_h^α leaves where*

$$L_h^\alpha = \begin{cases} \frac{2^h(1-\alpha)^h - 2\alpha}{1-2\alpha} & \text{if } 0 \leq \alpha < 0.5 \\ h+1 & \text{if } \alpha = 0.5 \end{cases} \quad (1)$$

Proof: The number of leaves N_i^L and number of non-leaves (internal nodes) N_i^I at level-i ($0 < i < h$) of the α-leaf tree is given by, $N_i^L = 2^i\alpha(1-\alpha)^{i-1}$ and $N_i^I = 2^i(1-\alpha)^i$. The total number of leaves in the α-leaf tree of height h is obtained by adding the number of leaves at each level:

$$L_h^\alpha = \sum_{i=1}^{h} N_i^L + N_h^I = \sum_{i=1}^{h} 2^i\alpha(1-\alpha)^{i-1} + 2^h(1-\alpha)^h$$

Summing the above for the two cases: $\alpha = 0.5$ and $\alpha < 0.5$ yields the result. ■

Lemma 2. *A Fibonacci-leaf tree of height h has L_h^F leaves where $L_h^F = F_{h+3} - 1$.*

Proof: A Fibonacci-leaf tree has F_{i-1} leaves at level-i ($0 < i < h$), and $F_{h-1} + F_{h+2}$ leaves at the bottom level-h. Adding these, and using the known equality $\sum_{i=0}^{h-1} F_i = F_{h+1} - 1$ yields:

$$\begin{aligned} L_h^F &= \sum_{i=1}^{h-1} F_{i-1} + F_{h-1} + F_{h+2} = \sum_{i=0}^{h-1} F_i + F_{h+2} \\ &= F_{h+1} - 1 + F_{h+2} = F_{h+3} - 1 \end{aligned}$$

■

C. Path Length

The path length from the packet at the leaf of the tree to the root in the α-leaf tree is representative of the cost of verifying that packet. The balanced Merkle tree has the same path length from any leaf to the root. For the unbalanced α-leaf tree, the path length increases as we consider successive leaves (from left to right), as shown below:

Lemma 3. *In an α-leaf tree of height h the authentication path length l_k of the k^{th} packet is given by:*

$$l_k \approx min\left\{h, \frac{\ln\left[\frac{k(1-2\alpha)}{2\alpha} + 1\right]}{\ln(2-2\alpha)}\right\} \quad (2)$$

Proof: Number of leaves in an α-leaf tree upto level j ($j < h$, where h is the height of the tree) is given by:

$$L_j^\alpha = \sum_{i=1}^{j} 2^i\alpha(1-\alpha)^{i-1} = 2\alpha\left[\frac{(2-2\alpha)^j - 1}{1-2\alpha}\right]$$

For the k^{th} node to be sitting on level l_k,

$$2\alpha\left[\frac{(2-2\alpha)^{l_k-1} - 1}{1-2\alpha}\right] < k \leq 2\alpha\left[\frac{(2-2\alpha)^{l_k} - 1}{1-2\alpha}\right]$$

Simplifying the above equation yields the result. (Eq. (2)) ■

Lemma 4. *In the Fibonacci-leaf tree of height h the authentication path length l_k of the k^{th} packet is given by:*

$$l_k \approx min\left\{h, \left\lfloor log_\varphi\left[(k+1)\sqrt{(5)} + \frac{1}{2}\right]\right\rfloor - 1\right\} \quad (3)$$

Proof: Number of leaves in an Fibonacci-leaf tree upto level j ($j < h$, where h is the height of the tree) is given by:

$$L_j^F = \sum_{i=1}^{j} F_{i-1} = F_{j+1} - 1$$

For the k^{th} node to be sitting on level l_k,

$$F_{l_k} - 1 < k \leq F_{l_k+1} - 1 \quad (4)$$

Simplifying the above equation and using the definition of Fibonacci numbers yields the result. Where φ is the golden ratio. ■

D. Average Path Length

For the unbalanced tree structure, the mean path length is computed as below:

Lemma 5. *The average path length from leaf to root in the α-leaf tree of height h is:*

$$\begin{aligned} A_h^\alpha &= \frac{\alpha}{L_h^\alpha(1-\alpha)}\left\{\frac{2^{h+1}(1-\alpha)^{h+1} - 1}{(1-2\alpha)^2}\right. \\ &\quad \left. + \frac{2^{h+1}h(1-\alpha)^{h+1} - 4(1-\alpha) - 1}{(1-2\alpha)}\right\} \\ &\quad + \frac{2^h h(1-\alpha)^h}{L_h^\alpha} \end{aligned} \quad (5)$$

Proof:

$$\begin{aligned} A_h^\alpha &= \frac{\sum_{i=1}^{h} iN_i^L + hN_h^I}{L_h^\alpha} \\ &= \frac{\sum_{i=1}^{h} i2^i\alpha(1-\alpha)^{i-1} + h2^h(1-\alpha)^h}{L_h^\alpha} \end{aligned}$$

Simplifying the above equation yields the result. ■

Correspondingly, for the Fibonacci-leaf tree we can deduce that:

Lemma 6. *The average path length A_h^F from leaf to root in a Fibonacci-leaf tree of height h is: $A_h^F = \frac{(h-1)F_{h+3} + F_{h+1} + 1}{(F_{h+3} - 1)}$*

Proof: At every level i, ($i \geq 2$), F_{i-1} number of leaves have an authentication path of length i. And at $i = h$, F_{h+2} number of leaves have an authentication path of length h. Summing these with the aid of known equality $\sum_{i=1}^{h} iF_i = hF_{h+2} - F_{h+3} + 2$ yields the result.

■

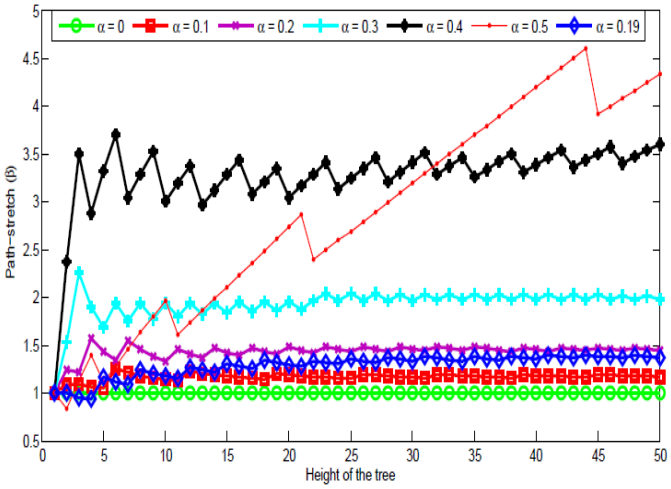


Fig. 2. Path-stretch β with respect to the height of the tree for different α values

E. Path Stretch

We define metric β as the ratio of the average authentication path length in the unbalanced α-leaf tree to the (fixed) path-length in the balanced Merkle hash having the same number of leaves.

$$\beta_\alpha = \frac{A_h^\alpha}{\lceil log_2(L_h^\alpha) \rceil} \tag{6}$$

Metric β thus represents the average "stretch" in the path from leaf to root resulting from the imbalance.

Lemma 7. *The path-stretch β for an α-leaf tree of height h is bounded by $1/log_2(2-2\alpha)$.*

Proof: Eq.(6) in conjunction with Eq.(5) and Eq.(1) can be used to directly calculate the path-stretch β.

$$\lim_{h\to\infty} \beta_\alpha \leq \lim_{h\to\infty} \frac{2\alpha h}{log_2(L_h^\alpha)} + \frac{h(1-2\alpha)}{log_2(L_h^\alpha)} \leq 1/log_2(2-2\alpha)$$

■

For the specific case of the Fibonacci-leaf tree we have:

Lemma 8. *The path-stretch β for the Fibonacci leaf tree is bounded by $1/log_2(\varphi)$, where φ is the golden ratio.*

Proof:

$$\begin{aligned} \beta_F &= \frac{(h-1)F_{h+3} + F_{h+1} + 1}{(F_{h+3}-1)\lceil log_2(F_{h+3}-1) \rceil} \\ &\leq \frac{h-1}{log_2(F_{h+3}-1)} + \frac{F_{h+1}+h}{(F_{h+3}-1)log_2(F_{h+3}-1)} \end{aligned}$$

Using $F_h \approx \frac{\varphi^h - (1-\varphi)^h}{\sqrt{5}}$ where $\varphi = \frac{1+\sqrt{5}}{2}$, we have

$$\lim_{h\to\infty} \beta_F \leq \lim_{h\to\infty} \frac{h}{log_2(F_{h+3}-1)} \leq \frac{1}{log_2(\varphi)} \approx 1.44 \tag{7}$$

■

Fig. 2 illustrates path-stretch β as a function of tree height for different values of α. It is seen that path-stretch increases with α; $\alpha = 0.2$ stretches the average path-length by at most 50%, $\alpha = 0.3$ by around 100%, and as α approaches 0.5, the tree degenerates to a linear chain and path-lengths grow linearly, making the path-stretch unbounded.

IV. Analytical Model

The α-leaf tree permits an unbalanced tree in which the authentication path length for successive packets of the streamed content progressively increases. This has the benefit that initial packets can be verified faster, allowing for shorter delay in playing back the streaming content. However, it comes at the cost of increasing the average path length (since for a given number of leaves an unbalanced α-leaf tree will have larger height than the balanced Merkle tree). In this section we develop a simple analytical model to quantify this trade-off and identify optimal setting of the imbalance parameter α. As stated in section II, we use the number of stalls during playback, start-up delay, tolerance to packet losses, and authentication overhead as measures to evaluate the performance of our α-leaf tree scheme.

A. System Model

Packet Arrival Process: We assume that packets comprising the streaming content arrive to the receiver as a periodic process perturbed by Gaussian noise. In other words, the arrival time of the k-th packet ($k = 1, 2, \ldots$) is $\mathcal{N}(kt, \sigma^2)$, i.e. has a normal distribution with mean kt and variance σ^2, where t is the average inter-arrival time between successive packets, and time is measured from arrival of the 0-th packet. We have validated this model with several captured traces of YouTube traffic (as explained in the next section), which show that the deviation of packet arrival times from the expected periodic are well-approximated as Gaussian.

Authentication Process: It is assumed that the digital signature at the root of the tree is received and verified prior to arrival of the 0-th packet. Upon receipt of a packet, its content is hashed to obtain the corresponding leaf in the tree – this takes time δ^{leaf}. This resulting hash is concatenated with the sibling hash in the tree, and the result hashed to move one level up in the authentication path – this takes time δ^{hash}. This process of concatenating hashes and re-hashing is repeated at each successive level of the tree till the root is reached, whereupon the result is matched with the hash value stored at the root. For the k-th packet that has an authentication path-length of l_k (as derived in Section III-C), the time δ_k for authentication is therefore given by:

$$\delta_k = \delta^{leaf} + l_k \delta^{hash} \tag{8}$$

Note that received packets are authenticated sequentially (i.e. on a single processor) in order of arrival.

Playback Process: For real-time playback of a long media stream, steady-state conditions require that the playback time of (the content in) a packet be larger than the average inter-packet arrival time, otherwise the playback will eventually stall

after every packet waiting for arrival of the next packet. We therefore assume that each packet is played back for time $t + \epsilon$ for some known constant $\epsilon > 0$. Further, playback of the stream starts at some offset Δ from the arrival of the first packet. This offset allows the receiver to protect against jitter in packet arrival times, as well as allows time to authenticate received packets before they need to be played back. Thus the playback of k^{th} packet ($k \geq 0$) will start at time $\Delta + k(t+\epsilon)$, where time is measured from the arrival of the 0-th packet.

B. Probability of Playback Stall

Playback stalls whenever the receiver runs out of authenticated packets. The k-th packet therefore causes a stall if it is ready (namely has arrived and is authenticated) later than its required playback time, i.e.

$$\mathcal{N}(kt, \sigma^2) + \delta_k > \Delta + k(t + \epsilon) \tag{9}$$

If we want to bound the probability p that the k-th packet causes a stall in the playback, we need that:

$$\Pr[\mathcal{N}(kt, \sigma^2) + \delta_k > \Delta + k(t + \epsilon)] \leq p$$

This can be expressed in terms of the standard normal (i.e. zero mean and unit variance) variable $\mathbb{Z}$ as:

$$\Pr[\mathbb{Z} > (\Delta + k\epsilon - \delta_k)/\sigma] \leq p$$

Denoting by $\Phi(z) = P[\mathbb{Z} > z]$ the complementary cumulative distribution of the standard normal, the condition that needs to hold for given stall probability can be expressed as:

$$(\Delta + k\epsilon - \delta_k)/\sigma \geq \Phi^{-1}(p) \tag{10}$$

The above inequality illustrates that for given ϵ and σ, initial packets (i.e. small k) rely on the playback offset Δ to cover for their authentication delay δ_k, whereas later packets (large k) have more time accumulated over the playback of previous packets (the $k\epsilon$ term) to mask their authentication delay. This provides the rationale for the unbalanced tree that gives lower δ_k for earlier packets than later ones, allowing for a smaller playback offset Δ without increasing the probability of playback stalls.

C. Optimising α to Minimize Stall Probability

For given parameters pertaining to packet arrival process, authentication process, and playback process, we can use inequality (10) to adjust the tree imbalance (via parameter α) to minimize playback stalls. To this end, we first explore, for given number of leaves n in the tree (i.e. the tree is built over n packets), the tree imbalance that minimizes stall probability for a chosen packet-k. This can be deduced by differentiating the left side of inequality (10) with respect to α and equating to 0 yielding:

$$\mathrm{d}\left[\frac{\Delta + k\epsilon - \delta_k}{\sigma}\right] / \mathrm{d}\,\alpha = 0 \tag{11}$$

Eq. (11) in conjunction with Eq. (2) and Eq. (8) can be used to directly estimate the value of α that minimizes the stall probability for packet-k.

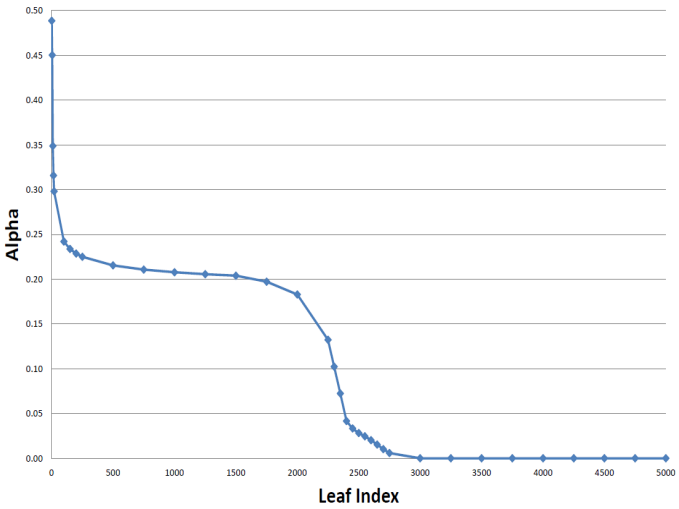


Fig. 3. Optimal α values of each leaf of an α-leaf tree

In Fig. 3 we plot the optimal tree imbalance α as a function of the packet index k. The system parameters correspond to the sensor node scenario of audio streaming as described in the next section, and the tree is built over 5000 packets. The plot confirms that the first few packets favour $\alpha = 0.5$, which corresponds to a completely imbalanced tree (in fact a linear chain) wherein initial packets have very short paths, whereas packets towards the end of the tree favour $\alpha = 0$ corresponding to a balanced tree since that gives them the shortest path to the root. What is however interesting is that the optimal imbalance progressively reduces with each successive packet, and indeed many packets in the middle favour the moderately imbalanced tree of $\alpha = 0.19$; this corresponds roughly to the special Fibonacci-leaf tree structure that we described earlier.

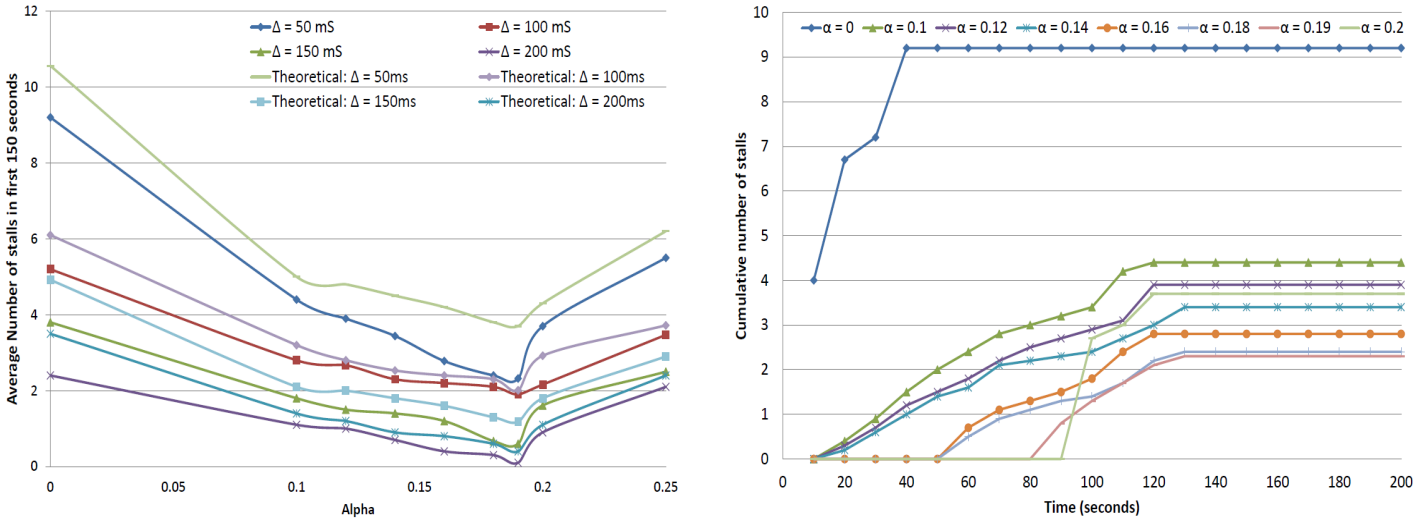
The above study considered each packet (that formed a leaf in the tree) in isolation. It is possible to deduce the optimal tree structure considering all packets. To do so, we can estimate the total number of stalls S in the stream of n packets by summing up the probability of stall over all packets:

$$S = \sum_{k=1}^{n} \Pr\left\{\mathbb{Z} > \frac{(\Delta + k\epsilon - \delta_k)}{\sigma}\right\} \tag{12}$$

Differentiating this with respect to α and equating to 0 yields:

$$\sum_{k=1}^{n} \exp\left\{-\left[\frac{\Delta + k\epsilon - \delta_k}{\sqrt{2}\sigma}\right]^2\right\} \cdot \frac{\mathrm{d}\left[\frac{\Delta+k\epsilon-\delta_k}{\sqrt{2}\sigma}\right]}{\mathrm{d}\,\alpha} = 0 \tag{13}$$

Numerical evaluation of the above equation gave us $\alpha = 0.19$ for the optimal tree imbalance, which again corresponds roughly to the Fibonacci-leaf tree. This strongly indicates that the Fibonacci-leaf tree achieves growth in path length most suited to real-time authentication of streaming content. This is substantiated with simulations of realistic scenarios in the next section.



(a) Number of stalls vs. tree imbalance

(b) Number of stalls vs. time ($\Delta = 50$ ms)

Fig. 4. Audio streaming to sensor device: (a) Average number of stalls versus tree imbalance α, and (b) Cumulative stalls as a function of time

V. Performance Evaluation via Simulation

We now validate the model of the α-leaf tree presented above and evaluate its efficacy in two real-world application via simulation. Our simulation is written in C, and works as follows. The tree structure corresponding to the choice of imbalance parameter α is first chosen. Then packets are generated and delivered to the receiver as per the arrival process. Each packet upon arrival is queued for authentication, and served in FIFO manner. To authenticate packet-k, the receiver hashes the packet content (taking time δ^{leaf} to do so), and then traverses up the authentication path (of length l_k) successively concatenating the hashes and re-hashing (which takes time δ^{hash} at each level). Authenticated packets are moved to the playback queue. Playback commences offset Δ after arrival of the first packet. Packets (if any) in the playback queue are played in FIFO manner for duration $t+\epsilon$ each. If the playback queue goes empty during the process, a stalls occurs. We measure the impact of parameters such as tree imbalance α and playback offset Δ on the number of stalls observed in simulation. The two application scenarios for which we conducted the simulation study are described next.

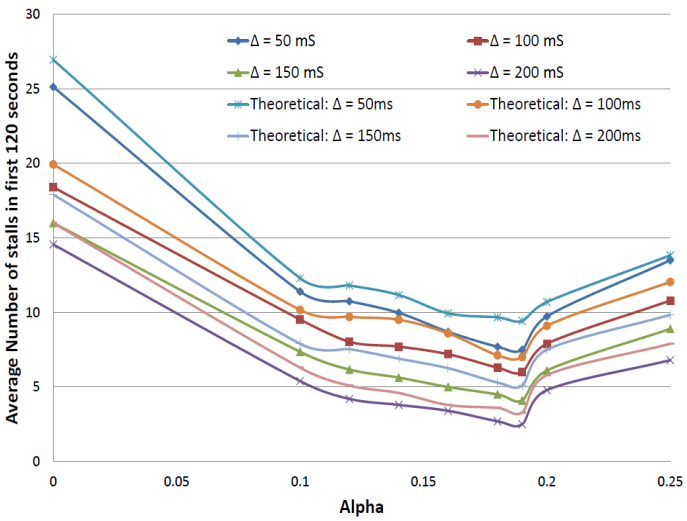
A. Application 1: Audio Streaming to Sensor Device

Imagine a scenario in which visitors to a museum are given low-cost portable devices that can sense their location, and play streaming audio relevant to the artefact in front of which they stand. The received audio stream may need to be authenticated to protect against injection of malicious content. In such a scenario, we evaluate how the α-leaf tree can reduce playback delay and playback stalls to improve the user's streaming audio experience.

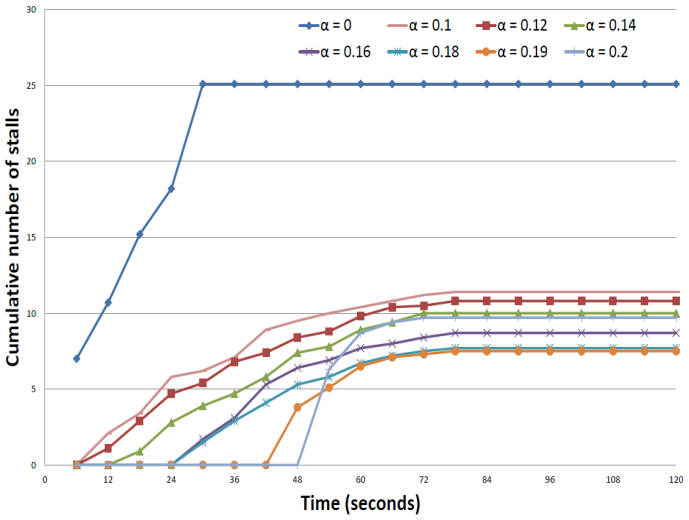
The parameters of this application setting are as follows: packet size is exponentially distributed with mean 28 bytes, comprising 20 byte payload of G.729 coded voice and 8 bytes of headers (compressed layer-2 / IP / UDP / RTP headers), as per [26]. The G.729 codec is used with 8 Kbps default rate, yielding an overall channel data rate of 11.2 Kbps data rate. The mean packet inter-arrival time is therefore $t = 20$ ms, and the deviation of the arrival time from its periodic expected value is Gaussian with zero mean and standard deviation of $\sigma = 8$ ms. Playback of each audio packet takes $t+\epsilon = 30$ ms, consistent with the capabilities of the sensor mote platform. We used SHA-1 algorithm for hashing, which produces a 20 byte result, and takes $\delta^{leaf} = 6.5$ ms to hash a 28 byte packet and $\delta^{hash} = 4.3$ ms to re-hash a hash on the sensor mote platform [27]. We evaluate via simulation the playback performance for audio files of different lengths (150, 300, and 450 seconds of playback, corresponding to 5000, 10,000, and 15,000 packet respectively) for various playback offset Δ and tree imbalance parameter α.

Fig. 4(a) shows, for several choices of playback offset Δ, how the number of stalls observed in simulation varies with increasing tree imbalance α. Each point in the curve is obtained from 50 simulation runs for that parameter setting, and the curves depicted here correspond to tree size of 5000 leaves (the curves for other tree sizes were similar in character and are omitted for brevity). It should be noted that without any authentication, there was on average less than one stall even with the lowest playback delay considered. Alongside the simulation results, we also show curves that plot the number of stalls computed from Eq. (12) of our analytical model. We make several observations from this plot:

- For given Δ, the average number of stalls in the playback observed in simulation correspond well with the predictions from analysis. Though there is a small offset between the analytical and simulation curves (due to the analysis being more conservative as it is an upper bound), the shapes match very well, providing validity to our model. This means that for any given system parameters, the analytical model can be employed to



(a) Number of stalls vs. tree imbalance



(b) Number of stalls vs. time ($\Delta = 50$ ms)

Fig. 5. Video streaming to mobile phone: (a) Average number of stalls versus tree imbalance α, and (b) Cumulative stalls as a function of time

rapidly determine optimum tree structure best suited to the application.

- As the playback offset Δ is increased from 50 to $100, 150, 200$ ms, the curve for stall count shifts down, confirming our intuition that stalls can be reduced by increasing the playback offset.
- For given Δ, playback stalls decrease with increasing tree imbalance α, till they reach a a minimum and then start increasing again. More interestingly, the optimum imbalance that minimises stalls corresponds to $\alpha = 0.19$, which corresponds to the special Fibonacci-leaf tree structure. This confirms our observation from analysis in the previous section.
- Lastly, our results indicate that playback stalls can be reduced either by increasing playback offset Δ, or by adjusting tree imbalance. For example, if the receiver wanted to ensure no more than 10 stalls in the audio playback in this application, it would need to set the playback delay well over $\Delta = 200$ ms if it used the balanced Merkle tree. However, if it used our Fibonacci-leaf tree, the playback delay could be set much lower at $\Delta = 50$ ms while still achieving less than 10 playback stalls. Use of the imbalanced tree structure for authentication can therefore allow a four-fold reduction in playback offset for this application without impacting stall performance.

To better understand where the stalls are occurring, we plot in Fig. 4(b) the running count of the number of stalls over time (as the playback progresses) for various degrees of tree imbalance α (the playback offset is fixed at $\Delta = 50$ ms). This plot confirms our intuition that when the tree is balanced ($\alpha = 0$), all the stalls happen early in the playback, since the path lengths are independent of packet position in the tree. However, when α is increased (to say 0.19), i.e. the tree becomes skewed to the right, there are no stalls in the first 90 seconds of playback. This clearly shows the two advantages of using the unbalanced authentication tree structure – not only does the number of stalls reduce, but also stalls get pushed to later in the stream, by which time there is a good chance the user might have abandoned the stream anyway.

B. Application 2: Video Streaming to Mobile Phone

Users are increasingly watching streaming video on their mobile phones. The content may even be sourced from a peer-to-peer network having malicious participants who tamper with the content, and this necessitates authentication. We evaluate how the α-leaf tree can be tuned to improve playback experience (lesser stalls and lower playback delay) for such users, bearing in mind that many viewers abandon watching the content mid-stream.

The parameters of our video streaming application are derived from our captures of packet streams from YouTube, and from reported measurements [28]. The mean packet size was set at TCP's default of 1500 bytes. Our captured traces verified that the packet arrival process is normally distributed around a periodic mean of $t = 5$ ms (on a wireless link of 2.0 Mbps) with standard deviation of $\sigma = 2$ ms. The mobile phone plays the streaming video (MPEG-4 or FLV encoded) at a 1 Mbps rate [28], corresponding to a playback time of $t + \epsilon = 12$ ms per packet. We use SHA-1 for hashing, which it estimated to take $\delta^{leaf} = 35$ ms to hash a packet and $\delta^{hash} = 0.5$ ms to re-hash a hash, as per data available for mobile phone platforms in [29]. We evaluate below via simulation the playback performance for video clips of 60, 120, and 180 seconds (comprising 5000, $10,000$ and $15,000$ packets) for various setting of playback offset Δ and tree imbalance α.

Fig. 5(a) depicts, for chosen tree size of $10,000$ packets (about 120 seconds of playback), how the number of stalls (averaged over 50 simulation runs) in the first 120 seconds varies with the tree imbalance parameter α. It should be noted that in the absence of any authentication, there was on average

less than one stall during the 120 second playback period, even for the lowest playback delay $\Delta = 50$ ms considered. Authentication imposes a penalty, increasing the number of stalls to above 15 for a balanced Merkle tree even when the playback offset Δ is increased to 200 ms - such a high number of stalls can be annoying to the user. We again note the efficacy of an unbalanced tree. The Fibonacci-leaf tree ($\alpha = 0.19$) reduces the number of stalls from 25 to 8 for playback offset $\Delta = 50$ ms, and from 15 to 3 for higher playback delay $\Delta = 200$ ms. Stated another way, the Fibonacci-leaf tree yields less than 10 stalls with a tight playback delay of $\Delta = 50$ ms, while the Merkle tree has higher stalls (nearly 15) even with a much looser playback delay of $\Delta = 200$ ms. This clearly demonstrates that using unbalanced trees can reduce playback delays by a factor of four or more without impacting stall rates even in this important application of video streaming on a mobile phone.

Fig. 5(b) shows where the stalls occur in the video playback. Again, it is evident that stalls occur at the beginning when a balanced tree structure is used, while increasing imbalance reduces the number of stalls as well as pushes them to later in the stream, giving us the added benefit that the user might have abandoned the stream by then anyway.

VI. Conclusion

Content streaming to portable devices such as low-cost sensor nodes and mobile phones is growing rapidly, and real-time authentication of such content received over the wireless medium, potentially from untrusted peers comprising the distribution network, is of growing importance. The Merkle hash tree is a well-accepted construct that can be used to achieve low-cost authentication by amortising the cost of a digital signature over a large block of data, while still permitting loss resilient instantaneous authentication of packets by repeated hashing from the leaf to root of the tree. In this paper, we have generalized the Merkle tree structure to allow for imbalance, i.e. varying path-length from leaf to root. Our novel construct, called the α-leaf tree, permits gradually increasing path lengths for successive leaves. We derived several key properties of this unbalanced tree structure, and related them to the imbalance parameter α. We developed an analytical model to illustrate how the tree imbalance can be tuned to minimise playback stalls for given system parameters and application needs. We then validated our model via simulation of two realistic applications, one for audio streaming to a low-cost sensor device, and the other for video streaming to a mobile phone, and showed how our unbalanced tree structure readjusts the burden of authentication across the packets allowing streaming applications to have lower start-up delay for given playback stalls. We believe our proposal is of both theoretical and practical relevance, and may have broader application beyond the scenarios considered in this paper.

References

[1] Sandvine-Intelligent Broadband Networks. (Spring 2011) Global Internet Phenomena Report. http://www.wired.com/epicenter/2011/05/netflix-traffic/.

[2] Microsoft Tag. The Growth of Mobile Marketing and Tagging. http://tag.microsoft.com.

[3] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena, "Pollution in P2P Live Video Streaming," *International Journal of Computer Networks and Communications*, vol. 1, pp. 99–110, 2009.

[4] S. Ababneh, A. Khokar, and R. Ansari, "A Multimedia Content Authentication and Recovery Protocol in peer-to-peer Networks." in *Proc. of IEEE International Conf. on Electro/Information Technology*, 2008.

[5] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, I. S. S. Shenker, , and H. Yu, "Open DHT: A Public DHT service and its uses," in *Proc. SIGCOMM*, 2005.

[6] T. Wolfi, "Public Key Infrastructure Based on peer-to-peer Network," in *Proc. of 38th Hawaii Int. Conf. System Sciences*, 2005.

[7] M. C. Chan, S. Y. Hu, and J. R. Jiang, "Secure peer-to-peer 3D Streaming," *Multimedia Tools and Applications*, vol. 45, pp. 369–384, 2009.

[8] R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function," in *Proc. CRYPTO*, 1987.

[9] P. Dutta, J. Hui, D. Chu, and D. Culler, "Securing the Deluge Network Programming System," in *IPSN*, 2006.

[10] Visible-Measures-Corporation Research Brief. (2010) Understanding Viewer Abandonment Trends in Short-Form Online Video content. [Online]. Available: http://corp.visiblemeasures.com/contact-us/abandonment-research/

[11] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for obtaining digital signature and public key cryptosystems," in *Communications of the ACM, pages 120-126*, 1978.

[12] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," in *Advances in Cryptology - CRYPTO*, 1997.

[13] L. Lamport, "Constucting digital signatures from a one-way function," SRI-CSL-98, SRI International Computer Science Laboratory, Tech. Rep., 1979.

[14] A. Perig, "The BiBa one-time signature and broadcast authentication protocol," in *Conference on Computer and Communications Security*, 2001.

[15] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *7th Australasian Conference ACSIP*, 2002.

[16] A. Habib, D. Xu, M. Atallah, B. Bhargava, and J. Chuang, "Verifying Data Integrity in Peer-to-Peer Streaming," in *Multimedia Computing and Networking*, 2005.

[17] J. Park, E. Chong, and H. Siegel, "Efficient Multicast Stream Authentication Using Erasure Codes," in *ACM Trans. Inf. Syst. Secur.*, 2003.

[18] Y. Park and Y. Cho, "The eSAIDA Stream Authentication Scheme," in *Proceedings of the International Conference on Computational science and Its Applications*, 2004.

[19] A. Perrig, R. Cannetti, J. D. Tygar, and D. Song, "The TESLA Broadcast Authentication Protocol," in *CryptoBytes*, 2002.

[20] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," in *IEEE Symposium on Security and Privacy*, 2000.

[21] P. Golle and N. Modadugu, "Authenticating Streamed Data in the Presence of Random Packet Loss," in *Proceedings of the Network and distributed Systems Securuty Symposium*, 2001.

[22] Z. Zhang, Q. Sun, and W. Long, "A Proposal of Butterfly-graph Based Stream Authentication over Lossy Networks," in *ICME*, 2005.

[23] C. Wong and S. Lam, "Digital signatures for flows and multicasts," in *IEEE/ACM Transactions on Networking*, 1999.

[24] R. Tamassia and N. Triandopoulos, "Efficient Content Authentication in peer-to-peer Networks," in *Proc. of the International Conference on Applied Cryptography and Network Security*, 2007.

[25] K. Marek and N. Yakov, "A Note on Traversing Skew Merkle Trees," in *ECCC*, 2004.

[26] Cisco: Voice over ip - per call bandwidth consumption. [Online]. Available: http://www.cisco.com

[27] J. Hui, "Deluge: TinyOS Network Programming - The real way to program your motes," University of California, Berkeley, Tech. Rep., 2005.

[28] *http://en.wikipedia.org/wiki/Youtube*.

[29] Crypto ++ 5.6.0 benchmarks. [Online]. Available: http://www.cryptopp.com/benchmarks.html