STREAM AUTHENTICATION BASED ON GENERLIZED BUTTERFLY GRAPH

Zhishou Zhang[1,2]*, John Apostolopoulos*[3]*, Qibin Sun*[1]*, Susie Wee*[3] *and Wai-Choong Wong*[2]

[1]Institute for Infocomm Research (I^{2}R), Singapore
[2]Department of ECE, National University of Singapore, Singapore
[3]Hewlett-Packard Labs, USA

ABSTRACT

This paper proposes a stream authentication method based on the Generalized Butterfly Graph (*GBG*) framework. Compared with the original Butterfly graph, the proposed GBG graph supports an arbitrary overhead budget and number of packets. Within the *GBG* framework, the problem of constructing an authentication graph is considered as a design problem: Given total number of packets, packet loss rate, and overhead budget, we show how to design the graph (number of rows and columns and edge allocation among nodes) to maximize the expected number of verified packets. In addition, we also propose a new evaluation metric called Loss-Amplification-Factor (LAF), which measures the extent to which the authentication method exacerbates the effective packet loss rate. Experimental results demonstrate significant performance improvements over existing authentication methods like EMSS, Augmented Chain, and the original Butterfly.

Index Terms— Stream authentication, media security

1. INTRODUCTION

Desirable security services for emerging media streaming applications include assurance of data integrity, source authentication, and non-repudiation. Streaming data usually contains a long sequence of packets and each packet has a deadline before which it must be received, verified, and decoded. In addition, streaming data is often afflicted by, and tolerable to, a certain level of loss. Straightforward application of cryptographic digital signature [1] does not provide an effective or efficient solution for streaming data. One extreme case is to generate a single signature for all the packets; while this leads to very low overhead, it is not robust against packet loss because even one lost packet makes all the other packets unverifiable. The other extreme case is to generate and transmit a signature for each packet. This approach is resilient to losses, but has very high overhead (signature size is usually on the order of 100 bytes) and very high complexity (signing and verifying operations have high complexity).

Another common approach of stream authentication amortizes a signature among a group of packets connected as a directed acyclic graph (*DAG*). Each node corresponds to a packet and an edge from node *A* to *B* appends *A*'s hash to *B*. Note that a hash requires much lower overhead (about 20 bytes) and much lower computation than a signature. The graph typically has one packet carrying the signature and each node has at least one path to the signature packet. At the receiver, lost packets are removed from the graph and a packet is verifiable if it has at least one path to the signature packet. The simple hash chain [2] arranges packets into one row, where each packet has an edge to the previous packet and the first packet is signed. It has low overhead as each packet has only one outgoing edge, however it also has low robustness against packet loss as any loss will cut the chain and subsequent packets are not verifiable. To increase the robustness, Efficient Multi-Chained Stream Signature (EMSS) [3], Augmented Chain [4], and Butterfly [5] proposed different ways of adding redundant edges to the graph. In [5], we have experimentally validated that the Butterfly has very good robustness against network loss, due to high fault-tolerance of butterfly graph. Specifically, the butterfly architecture maximizes the independence of the dependencies for a given packet. However, the Butterfly graph has its own limitations: (1) fixed overhead due to its fixed topology, e.g., given an arbitrary overhead budget it is not clear what is the best way to add/ remove edges; (2) total number of packets *N* equals $N_R(\log_2 N_R+1)$, where N_R is the number of rows; (3) the signature packet size grows with *N*, as it contains both the signature and the hashes of all packet in the first column. For large *N* the signature packet may be larger than the network *MTU* and therefore would be fragmented for transmission -- increasing its loss probability and negatively impacting all of the packets in the graph.

In this paper, we try to preserve the Butterfly graph's valuable robustness to losses, while extending it to overcome the above 3 limitations, and refer to this framework as the Generalized Butterfly Graph (*GBG*). The GBG framework supports a wide range of possible authentication graphs, and the problem of finding the best authentication graph for a given situation corresponds to a graph design problem. The input parameters include total number of packets *N*, packet loss rate (*p*), and overhead budget (R_O). In this paper we assume IID losses for simplicity. The output parameters include the number of rows and columns (N_R, N_C), edge placement, and number of transmissions (*M*) of the signature packet. The signature packet may be transmitted multiple times to reduce the probability of loss. R_O accounts for the (multiple) transmissions of the signature packet and overhead data corresponding to edges in the graph. The evaluation metric is verification percentage (or verification probability), defined as the percentage of received packets which are verified. In addition, we also propose a new evaluation metric called the Loss-Amplification-Factor (LAF),

1-4244-1437-7/07/$20.00 ©2007 IEEE

defined as the ratio of effective loss rate over packet loss rate. The effective loss rate accounts for both lost packets and packets which are received but unverifiable, i.e., $p+(1-p)(1-V)$, where p is packet loss rate and V is the verification percentage. For example, a value of 1.5 means the authentication technique causes the effective loss rate to be 50% higher than when no authentication is used. Ideally, the LAF would equal 1, i.e., all received packets are verified. The closer the LAF for an authentication technique is to 1 the greater its robustness to losses.

The rest of this paper is organized as follows. Section 2 analyzes the Butterfly graph to find a cost-effective edge placement strategy given an overhead budget R_O. Section 3 examines how to relax the Butterfly graph to support arbitrary number of packets and to confine the signature packet within an *MTU*. Based on this analysis, we propose in Section 4 the Generalized Butterfly Graph (GBG) framework and describe how to design an authentication graph for a given situation. Section 5 evaluates the proposed method against existing methods and Section 6 concludes the paper.

2. ANALYSIS OF BUTTERFLY: EDGE PLACEMENT

The butterfly graph has a fixed topology and therefore fixed overhead. This section studies the problem of how to place the edges in the graph given an arbitrary overhead budget R_O. To gain more insight into this problem, we implemented two greedy algorithms to progressively add edges (one at a time) to the authentication graph, starting from the initial state depicted in Fig. 1. Initially, the packets are arranged into the same matrix as in the Butterfly, however each packet is connected to only one packet in the previous column. Then, at every step, the greedy algorithm computes the overall verification percentage for all possible new edges and adds the edge that provides the largest gain. When there are multiple candidate edges with the same gain, then one of the candidate edges is picked randomly. The first greedy algorithm, referred to as *unconstrained greedy*, allows an edge from one packet to any packet in the previous column. The second algorithm, referred to as *constrained greedy*, allows only the edges that appear in original Butterfly.

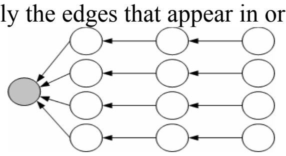

Fig. 1 – Initial state of greedy algorithms (with 12 packets)

Fig. 2 compares the LAF of the two greedy algorithms with 32 packets (N_R=8 and N_C=4), where 24 extra edges are progressively added, starting from the initial state shown in Fig. 1. One observation is when all 24 edges are added, each packet (except packet in the first column) has exactly 2 outgoing edges with both *unconstrained* and *constrained* greedy algorithms. Another observation is a packet's verification probability is maximized if it is directly connected to two packets that are independent of each other for verification (i.e., independence property). The unconstrained greedy has lower LAF when the number of added edge $\alpha \leq 15$, while the constrained greedy has slightly lower LAF when $\alpha > 15$. When α is small, the unconstrained greedy allows more than 2 packets to be connected to a packet with higher verification probability, without violating the independence property. When α is large, it is no longer possible for the unconstrained greedy to find edges that satisfy the independence property, while the constrained greedy algorithm can still find such edges. Therefore, the butterfly-constraint produces *slightly* worse performance in the middle and slightly better performance at the end. We will apply this constraint for edge placement. When the given overhead budget is less than 2 edges per packet, the edges are placed in the middle column first, and then progressively added to the outer columns.

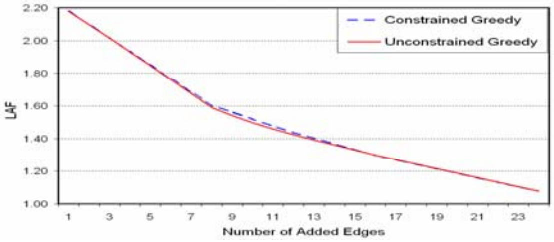

Fig. 2 – LAF of graphs built with unconstrained and constrained greedy

From the above experiment, the butterfly graph provides the best performance of all the graphs tested given an allocation of 2 outgoing edges per node for all nodes except the first column. Next, we examine the best edge placement order beyond 2 edges per packet. Fig. 3 shows the benefit (increment in overall verification percentage) if we place one extra edge into different columns of a butterfly graph with 17 columns. It is most beneficial to place the extra edge in column-9, followed by column-8, 10, 7, 11, 6, 12, 5, 13, 4, 3, 14, 2, 15, 1, and 16. Note that the benefits for columns 4 to 13 are almost the same.

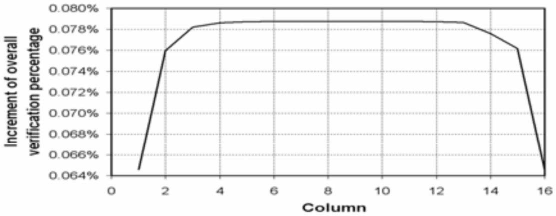

Fig. 3 – Increment in verification percentage when 1 edge is added to different columns of a butterfly with 17 columns.

3. RELAXING THE BUTTERFLY STRUCTURE

The goal of this section is to overcome the second and third limitations of the Butterfly graph. We examine how to relax the Butterfly structure to (a) support an arbitrary number of packets *N,* and (b) to confine the signature packet to within one *MTU*.

In the original Butterfly graph, the signature packet grows with the total number of packets N, as it contains the digital signature and the hashes of all packets in the first column. If the signature packet is too big to fit in one *MTU*, it has to be segmented for transmission, which increases its loss probability and directly impacts the authentication probability for all of the packets. Alternatively, when N is large, we can divide the N packets into smaller groups and form a butterfly graph within each group. However, this simple approach has high overhead and complexity, since there is one signature packet per group and signature packet is costly in terms of computation and transmission. Also note that each signature packet needs to be transmitted multiple times to avoid loss. We have found that we can overcome this problem, without incurring a loss in performance, by limiting the number of rows (N_R) so that the signature packet is confined within one *MTU*, and the extra packets are appended at the end of the graph by adding additional columns. Any consecutive $\log_2 N_R+1$ columns taken from this relaxed graph constitute a Butterfly graph. Fig. 4 shows an example with $N_R=4$ and $N_C=8$. Note that any 3 consecutive columns form a Butterfly graph of 12 packets. Therefore, to support an arbitrary number *N*, we can first construct a butterfly graph, and then append extra packets to the end of the butterfly graph.

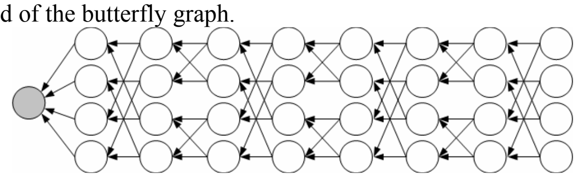

Fig. 4 – Relaxed Butterfly graph with 4 rows and 8 columns

In the Butterfly graph, all packets within a column have the same verification probability, which drops for the first 2 columns and then becomes flat for all subsequent columns. The proposed extension also has this property, as shown in Fig. 5 which compares the verification probability of packets from different columns in a Butterfly (128x8) and a relaxed Butterfly graph (64x16). It shows that the extra columns in the relaxed butterfly have flat verification probability.

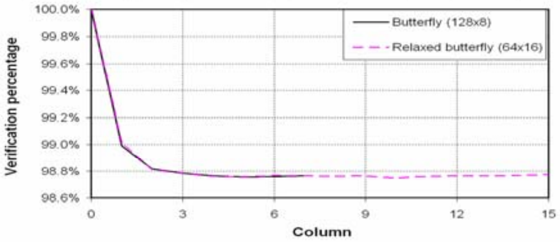

Fig. 5 – Verification percentage of packets in different columns of Butterfly and Relaxed butterfly graph (p=0.1)

4. PROPOSED GENERALIZED BUTTERFLY GRAPH

This section proposes the Generalized Butterfly Graph (*GBG*) framework, and describes how to design an efficient authentication graph, based on the prior analysis.

Given *N* packets, a generalized butterfly graph is constructed as a matrix with N_R rows and N_C columns, where $N_R=2^K$ and $N_C = \lceil N/N_R \rceil$. A packet in *r*-th row and *c*-th column is referred to as $P_{r,c}$, where $0 \le r < N_R$ and $0 \le c < N_C$. The signature packet P_{sig} contains the signature and hashes of all packets in the first column. A packet $P_{r,c}$ is connected to $P_{r,c-1}$ and is possibly connected to other packets in column *c*-1. Note that the *GBG* framework covers many possible graphs. For instance, when $N_R=1$, the GBG graph is the same as Simple Hash Chain; when $N_R=N$, the *GBG* graph is equivalent to an Authentication tree [6] with degree *N*. In addition, if $N=N_R(\log_2 N_R+1)$, the *GBG* graph is exactly the same as a Butterfly graph. In the proposed *GBG* framework, we examine how to design an efficient authentication graph. The input parameters include total number of packets *N*, overhead budget R_O, packet loss rate *p,* and maximum transmission unit (*MTU*), and the output parameters include N_R and N_C, number of transmissions of signature packet *M*, and graph topology (edge placement). The following subsections discuss how to choose appropriate values for these output parameters.

4.1. Number of rows and columns (N_R and N_C)

The value of N_R has two implications: it determines the size of the signature packet which contains signature (*S* bytes) and hashes (*H* bytes per hash) of all packets in the first column; it also affects the robustness against network loss. For instance, if N_R is too big, the signature packet has to be fragmented for transmission, which negatively affects the overall verification probability. On the other hand, if N_R is too small, a burst loss of length N_R may cut the graph into two halves and the second half cannot be verified anymore. Therefore, N_R should be chosen to be the maximum number satisfying two conditions: $N_R=2^K$ and $S+N_R H \le MTU$.

4.2. Number of transmissions of signature packet *M*

If the signature packet is lost, all the other packets are not verifiable, and therefore it has to be transmitted multiple times to increase its probability of being delivered. However, every transmission leads to additional overhead. For instance, if the signature packet is transmitted *M* times, the overhead budget left for the other packets is $R_O-M(S+N_R H)$ bytes. A small value of *M* leads to high loss probability of the signature packet, while a large value of *M* leads to fewer edges in the graph. A 1-D search can be performed to find the optimal value for *M*, e.g., Fig. 7 shows the verification percentage with different values of *M*. (Experiment setting: N=1024, R_O= 40,960 bytes, S=128, H=20, MTU=1500 and p=0.1).

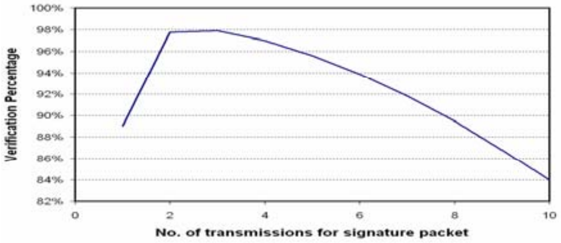

Fig. 6 – Verification percentage for various values of *M*

4.3. Edge placement

Given a N_RxN_C *GBG* graph, the initial graph topology (as in Fig. 1) and *M* transmissions of signature packet occupy $R_{init} = M(S+N_R H)+(N-N_R)H$ bytes. Therefore, the remaining overhead budget is R_O-R_{init}, corresponding to $e = \lfloor (R_O - R_{init})/H \rfloor$ extra edges which can be placed using the algorithm illustrated in

Fig. **7**. The extra edges are placed in round-robin manner, starting from packets in the middle column and progressively going outwards, as motivated by Fig. 3.

Note that the receiver can deduce the graph topology, given the values of N, N_R and R_O. Therefore, these three parameters are signaled in a header in the signature packet. The overhead of these three parameters is negligible, compared with hashes and signatures.

```
EdgePlacement(N_R, N_C, e) {
int L, R, round=0;

while(e > 0) {
  if(N_C%2==0)
    L = R = N_C/2
  else
    L = N_C/2;    R = L + 1;

  for( ; L>0 && e>0; L--, R++) {
    for(int i=0; i<N_R && e>0; i++) {
      if(round == 0)
        create an edge from P_i,L with butterfly constraint;
      else
        create an edge from P_i,L to the first node in column L-1 with least
                incoming edges;
      e --;
    } //end for
    if(L == R)        continue;
    for(int i=0; i<N_R && e>0; i++) {
      if(round == 0)
        create an edge from P_i,R with butterfly constraint;
      else
        create an edge from P_i,R to the first node in column R-1 with least
                incoming edges;
      e --;
    } //end for
  } //end for
  round ++;
 }//end while
} //end EdgePlacement(..)
```

Fig. 7 – Algorithm to allocate ***e*** extra edges in (N_RxN_C) *GBG* graph

5. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed *GBG* authentication graph, we implemented 4 authentication graph approaches and measured their performance in the following setting: N=1024, p=0.01~0.2, MTU=1500, S=128 and H=20. The first method is the proposed *GBG* method with N_R=64 and N_C=16, the extra edges are placed using the algorithm in Fig. **7**, and the optimal value of M is selected by a 1-D search. The second method is EMSS authentication [3] where each packet is connected to a packet that is randomly selected from the preceding 64 packets and the signature packet contains the hash of the first 10 packets. The third method is Augmented Chain [4] $C_{8,8}$, where each packet is connected to a previous packet and another packet that is 8 packets away in the high-level chain, and 7 packets are iteratively inserted between two consecutive packets in high-level chain. The fourth method is the Butterfly graph with N_R=128 and N_C=8, and the signature packet contains hashs of all packets in the first column.

Fig. 8 compares the LAF of the four approaches at various overhead levels for p=0.1. It shows that the proposed GBG significantly outperforms *EMSS* when overhead is below 40 bytes per packet (i.e., 2 hashes/packet). However, the gap decreases when overhead is greater then 40 bytes per packet.

Fig. 9 compares the LAF of the four approaches at various loss rates ranging from 0.01 to 0.2, where the overhead is fixed at 40 bytes per packet (i.e., 2 hashes/packet). It shows that the proposed GBG method outperforms the other three methods at all loss rates.

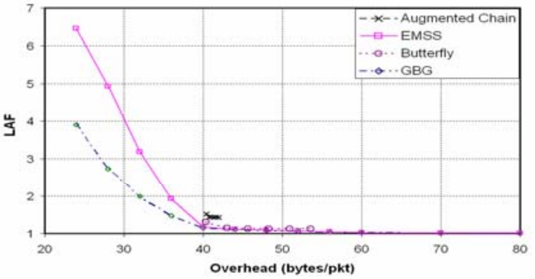

Fig. 8 – Comparison of LAF at various overhead (p=0.1)

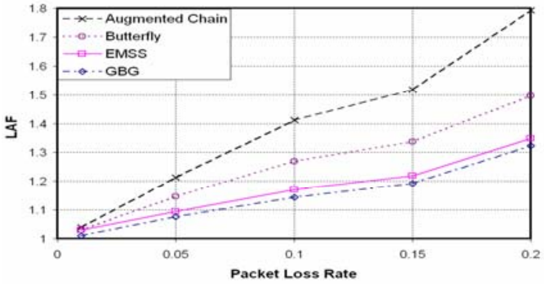

Fig. 9 – Comparison of LAF at various loss rates (overhead = 40 bytes per packet)

6. SUMMARY

This paper examined how to improve the Butterfly graph to provide improved stream authentication. We first examined how to support an arbitrary overhead budget by performing edge allocation to maximize the percentage of verified packets. We also proposed enhancements to support an arbitrary number of packets and to make the signature packet fit within one *MTU*. Based on this analysis, we proposed the Generalized Butterfly Graph (*GBG*) framework and described how to design a graph to maximize verification probability given a total number of packets, overhead budget, and packet loss rate. Experimental results demonstrate the performance improvement of the proposed method over existing methods.

7. REFERENCES

[1] B. Schneier, Applied Cryptography, Wiley, 1996. pp. 429-502

[2] R. Gennaro and P. Rohatgi, "How to sign digital streams,", in Advances in Cryptology – CRYPTO'97, pp. 180-197

[3] A. Perrig, R. Canetti, J. Tygar and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in Proc. of IEEE Symposium on Security and Privacy, 2000, pp. 56-73

[4] P. Golle and N. Modadugu, "Authentication streamed data in the presence of random packet loss," ISOC Network and Distributed System Security Symposium, 2001, p.13-22

[5] Z. Zhang, Q. Sun and W.C. Wong, "A proposal of butterfly-graph based stream authentication over lossy networks," in Proc. IEEE International Conference on Multimedia & Expo, July 2005

[6] C.K. Wong and S. Lam, "Digital Signature for Flows and Multicasts," The University of Texas at Austin, Department of Computer Sciences, Technical Report TR-98-15, July 1998