ISSN:0975-9646

An Efficient Dynamic TESLA based Authentication Scheme for Secure network Coding

Aparna Jumde[#1], Shyamsundar Gupta[2]

Savitribai Phule Pune University
Maharashtra India

***Abstract*— Network coding based applications are notoriously susceptible to malicious pollution attacks. Packets authentication schemes have been well-recognized as the most effective approach to address this security threat. However, existing packets authentication scheme i. e. the Times keys scheme consumes extra bandwidth for transmission of times keys. Also the Time keys can be captured and replay attack can be launched by the attacker. This motivates to design a new solution to avoid the problems in Time keys scheme. The solution is based on the concept that the source and the destination must agree on the keys along with the network coding process. This scheme of agreeing on the keys is known as adaptive TESLA scheme. Using this proposed scheme the effect of pollution attack and collusion attack is measured in terms of real packets corrupted and dropped. The proposed adaptive TESLA scheme for network coding is also compared with times keys in terms of bandwidth usages.**

***Keywords*— network coding, pollution attack, collusion attack, adaptive TESLA.**

INTRODUCTION

The Multi-hop Wireless Networks (MWNs) are regarded as such a promising solution for extending the radio coverage range of the existing wireless networks. System reliability can be improved through multi-path packet forwarding, which is feasible in MWNs. However, there exist many security and privacy issues in MWNs. Due to the open-air wireless transmission, MWNs suffer from various kinds of attacks, such as eavesdropping, data modification/injection, and node compromising; these attacks may breach the security properties of MWNs, including confidentiality, integrity, and authenticity. In addition, some advanced attacks, such as traffic analysis and flow tracing, can also be launched to compromise the privacy of users, including source anonymity and traffic secrecy.

Network coding refers to a general class of routing mechanisms where, in contrast to traditional “store-and-forward” routing, intermediate nodes modify data packets in transit. Network coding has been shown to offer a number of advantages with respect to traditional routing, the most well-known of which is the possibility of increased throughput in certain network topologies. It has also been suggested as a means of improving robustness against random network failures since, as with erasure codes the destination can recover the original data (with high probability) once it has received sufficiently many correct packets, even if a large fraction of packets are lost. Because of these advantages, network coding has been proposed for applications in wireless and/or ad-hoc networks, where communication is at a premium and centralized control may be unavailable; it has also been suggested as an efficient means for content distribution in peer-to- peer networks and for improving the performance of large-scale data dissemination over the Internet.

A major concern in systems that use network coding is to provide protection against problem is particularly acute because errors introduced into even a single packet can propagate and pollute multiple packets making their way to the destination; this is a consequence of the processing that honest nodes, downstream of any corrupted packets, apply to all incoming packets malicious modification of packets (i.e., “pollution attacks”) by Byzantine nodes in the network.

I. EXISTING SYSTEM

P2P networks with random network coding are known to be susceptible to pollution attack. During such attacks, malicious peers inject polluted packets to their neighbours. Such pollution can rapidly propagate in the network, leading to substantially degraded performance due to the wasted bandwidth of corrupted blocks distribution.

Furthermore, since network coding allows peers to forward packets coded from their received packets, as long as a single input packet is corrupted, all output packets forwarded by a peer will be corrupted. As a result, a single corrupted block will pollute the whole network system and prevent peers from decoding the original blocks, thus degrading the performance of the whole system Therefore, it is crucial to check coded packets whether they are corrupted before using them for encoding in network coding systems, and filter out polluted packets as early as possible.

Time based scheme was proposed to solve this problem. But the time based keys can be captured & replayed in the network.

- Time keys can be captured & replay attack can be launched by the attacker.
- Extra bandwidth is consumed for transmission of time keys & for the large network this will become a critical bottleneck.
- This motivates to design a new solution to avoid the problems in time key scheme.

II. PROBLEM DEFINITION

P2P networks with random network coding are known to be susceptible to pollution attack. During such attacks, malicious peers inject polluted packets to their neighbours. Such pollution can rapidly propagate in the network, leading to substantially degraded performance due to the wasted bandwidth of corrupted blocks distribution. Furthermore, since network coding allows peers to forward packets coded from their received packets, as long as a single input packet is corrupted, all output packets forwarded by a peer will be corrupted. As a result, a single corrupted block will pollute the whole network system and prevent peers from decoding the original blocks, thus degrading the performance of the whole system Therefore, it is crucial to check coded packets whether they are corrupted before using them for encoding in network coding systems, and filter out polluted packets as early as possible.

Time based scheme [1] was proposed to solve this problem. But the time based keys can be captured & replayed in the network.

A. Solution Strategy

The Solution is based on the same concept of time keys but without transmitting the time keys the source & the destination nodes must agree on the keys. This scheme is referred as Adaptive TESLA [7]. Each node with a seed key generates N number of keys using one way hash function. Let seed key be S1. Subsequent keys generated by one way hash chain is S2, S3, S4, S5 ….. SN.

All the nodes are assumed to be time synchronized. Every time once nodes use the key from SN,SN-1,… S2 (ie in the reverse order). The network coding vectors are encrypted & also digital signature is prepared using the Key S for that time interval. After the time interval using S1 & current time stamp , second generation keys are prepared. When any corrupt packets are launched in the network, at the receiver , the digital signature is verified using the Key S for that time interval & dropped if the signature match failed. This way we can avoid the pollution & colluding attack without consuming extra bandwidth.

Also replay attack cannot be launched in our approach because keys at time instant is not same as any, since keys are regenerated at each generation.

B. *Objectives*

The main objectives of the project are as follows:

1. Measure the effect of pollution attack & colluding attack on the proposed solution in terms of number of real packets corrupted & dropped.
2. We will also measure the bandwidth consumed in our approach.
3. The proposed Adaptive TESLA scheme is compared with time key solution in terms of bandwidth usage.
4. Measure the effect of replay attack in the Adaptive TESLA & time key solution in terms of number of real packets corrupted & dropped.

C. *Proposed solution & advantage*

Network coding was first introduced by Ahlswede et al [6]. Subsequently, two key techniques, random coding and linear coding further promote the development of network coding technologies. The random coding makes network coding more practical, while the linear coding is proven to be sufficient and computationally efficient for network coding. Currently, network coding has been widely recognized as a promising information dissemination approach to improving network performance. Primary applications of network coding include the file distribution and multimedia streaming on P2P overlay networks, data transmission in sensor networks , tactical communications in military networks , etc.

Compared with conventional packet forwarding technologies, network coding offers, by allowing and encouraging coding/mixing operations at intermediate forwarders, several significant advantages such as the potential throughput improvement, transmission energy minimization, and delay minimization. We propose keying based on U-Tesla to guard the network against attacks on network coding process. In this, SINK node distributes seed keys for all the nodes. Based on this seed keys node generate keys for a particular instant of time. Both node and sink know the keys at a particular instance of time. Node forms the keys using the U-Tesla mechanism of hashing subsequent keys with time value. Whenever node sends data, it also forms the hash of data using the key at that particular time and then sends this hash value also to the sink in the packet. Sink will verify the hash and if it is valid, it will accept the packet for network coding. If the hash validation fail sink will drop the packet from doing decoding. Since time instance and keys are included for forming hash, the message corruption attack and message replay attack will fail, thereby security is implemented.

III. System Architecture

System architecture is the conceptual design that defines the structure and behavior of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. The System architecture is shown below.

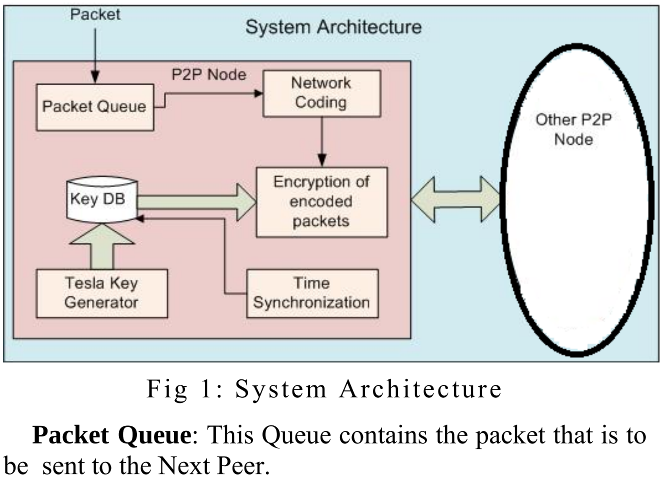


Fig 1: System Architecture

Packet Queue: This Queue contains the packet that is to be sent to the Next Peer.

Network Coding: Encoding of the packets is done in this module.

Time Synchronization: This module is responsible to maintain the time Synchronization between two peers while generating the key.

Tesla Key Generation: The module where actually the key is generated for the encryption.

Key DB: Stores the Key Generated by TESLA key generator.

A. *Working of proposed system:*

This system employ the Paillier cryptosystem as the HEF to apply encryption to GEVs. HEF can not only keep the confidentiality of GEVs, but also enable intermediate nodes to efficiently mix the coded messages. In the Paillier cryptosystem, given a message m and the public key (n, g), the encryption function is described as follows

$$E(m) = gm \cdot rn \ (\mathrm{mod}\ n2\ )$$

where r is a random factor in the Paillier cryptosystem.

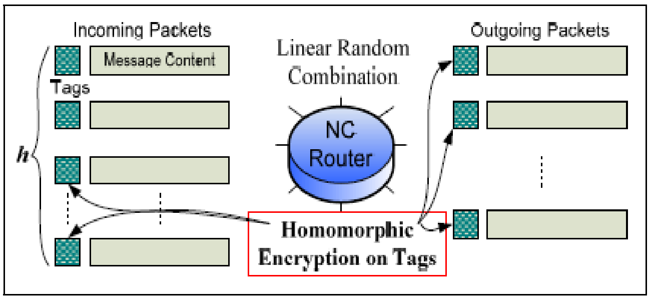


Fig 2: Random network coding with HEF

With HEFs, intermediate nodes are allowed to directly perform linear coding/mixing operations on the coded messages and encrypted tags, as shown in Fig. 5.11. In other words, due to the homomorphism of the HEF, linear network coding can be achieved by operating on the encoded messages and the ciphertext of GEVs, without knowing the decryption keys or performing the decryption operations.

B. *Steps in the Proposed Scheme*

The proposed scheme primarily consists of three phases: source encoding, intermediate random linear recoding, and sink decoding. Without loss of generality, we assume that each sink acquires two keys, the encryption key ek and the decryption key dk, from an offline Trust Authority (TA), and the encryption key ek is published to all other nodes. For supporting multicast, a group of sinks are required to obtain from the TA or negotiate the key pair in advance then, they can publish the encryption key and keep the decryption key private in the group.

Source Encoding: Consider that a source has h messages, say x1,x2 …xh , to be sent out. The source first prefixes h unit vectors to the h messages, respectively, as illustrated in Fig 5.12. After tagging, the source can choose a random LEV and then perform a linear encoding operation on these messages. Thus, one LEV will generate an encoded message with the GEV(which is equal to the LEV temporarily) tagged. To offer confidentiality for the tags, homomorphism encryption operations are employed on these tags.

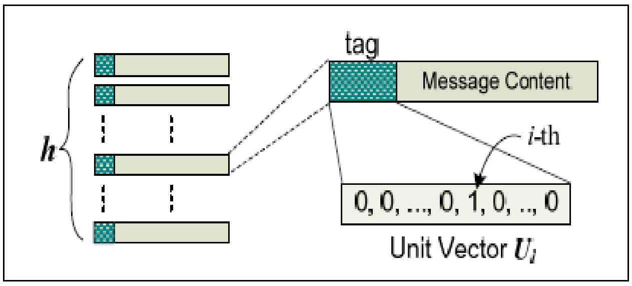


Fig 3: Source encoding

Intermediate Random Linear Recoding: After receiving a number of packets of the same generation, an intermediate node can perform random linear coding on these packets. To generate an outgoing packet, firstly, a random LEV is chosen independently; then, a linear combination of message content of the incoming packets is computed as the message content of the outgoing packet

Sink Decoding: After receiving a packet, the sink first decrypts the packet tag using the corresponding decryption key dk. Once enough packets are received, a sink can decode the packets to get the original messages.

C. *Algorithm of Proposed Scheme*

Assume that there are four source nodes and all are ready to send the data to sink node relayed through two routers

Step 1: Display the network setup with all four source nodes and two router and the destination

Step 2: Generate packets from each node, now assume that each node is sending 100 packets of data

Step 3: The packets from source node are send to the router 1

Step 4: Router 1 will take 25 packets of data from each node. When all 4 blocks are received at router 1 it starts performing random network coding and forms the four generation of messages.

Step 5: With each generation of message a Global encoding vector (GEV) is send which is generated using paillier cryptography.

Step 6: Paillier cryptosystem works as follows

Pick two large prime p and q and let n=p*q. Let lambda (n) =lcm(p-1,q-1). Pick random g ϵ Ẓ*n2 such that L (g lambda mod n2) is invertible modulo n (where L(u) = (u-1) /n. where n and g are public p and q are private. For plaintext x and resulting ciphertext y, select a random r ϵ Ẓ *. Then

ek(x,r) = gm *rn mod n2

dk=(L(y lambda mod n2)/L(g lambda mod n2))* mod n2.

Step 7: The value of lambda is taken as Global encoding vector and send with each generation of message

Step 8 a: Calculate Seed Key which is pseudo-random function(PRF) and it is calculated as follows

Seedkey= Any random number *100 +1000

Step 8 b: Seed Key distribution: Sink distributes seed key for all the nodes as follows

Seedkey = Seedkey * node number *3 + 2 * Seedkey + 5

Step 9: Key generation: Using the seed key, each node will generate the keys for generation of time as

K1 = H(S,Time1);
K2 = H(K1,Time2);
K3 = H(K2,Time3);
K4 =H(K3,Time4);

Both node and sink will generate this keys for some generation and store in them.

Message signing: Whenever node sends data , it forms a cryptographic hash of data with the key at that particular instant of time and send this hash value also in the packet to sink.

Message authentication: When sink receives packet it will again form cryptographic hash of the data and check with original hash in the packet. If both are same, the message is valid, else the message is considered as corrupted and dropped.

Step 10: When the Sink node receives all the generations of composite packets along with the hash keys, sink node starts authenticating each generation of composite packets with the help of seed key

Step 11: Sink node first calculates the hash key of second generation of composite packets using the seed key, authenticate it and then with computed hash key of the second generation it stats recursively calculating the hash key of third generation of composite packets.

Step 12: In this way by recursively calculating the hash value for each generation of composite packets all generations are authenticated

Step 13: Now the Sink node stars decrepting the encoded composite packets

Step 14: It decode all the generation of messages one by one, get the order of blocks in which they were encoded and then stats decoding the packets from each source node

Step 15: All packets are re-assembled and original message is constructed at sink node.

IV. RESULTS

The performance of system is calculated under two parameters for all the three scenarios

- Computational overhead
- Network overhead

As the number of packets increases the computational overhead is goes on decreasing as well as the network overhead also decreases so more number of packets we can send without collusion.

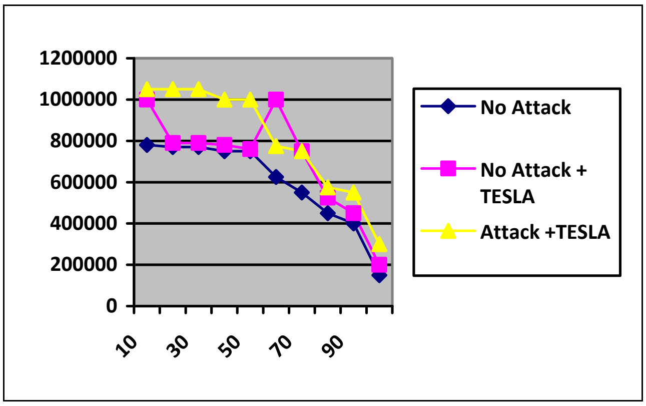
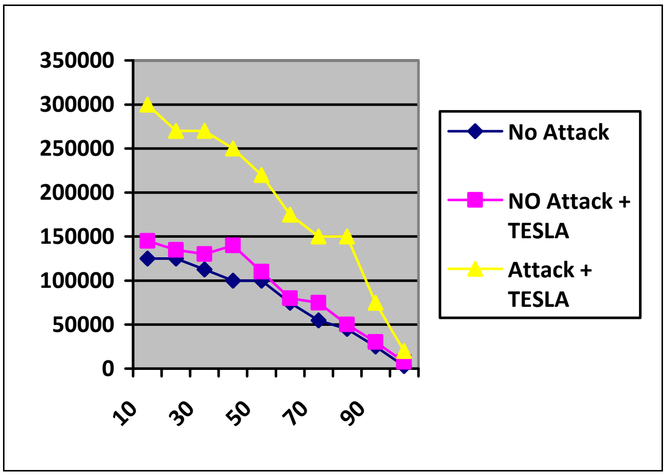


Fig 4: network overhead

As number of packets increases the rate that is MB per second goes on decreases in all three scenarios.



V. CONCLUSIONS

Although the time keys scheme, which is a in-network security scheme based on time and space properties of network coding, is not a efficient scheme . As attacker can capture the times keys and may introduce an replay attack into the network. The proposed scheme use network coding along with the encryption of times keys. It saves the bandwidth of the network as well as it reduces total corrupted packet. Thus the network coding increases the throughput of network and the TESLA technique that generates the seed keys both at source and sink makes the system more efficient.

REFERENCES

[1] Ming He · Zhenghu Gong · Lin Chen · HongWang · Fan Dai · Zhihong Liu “Securing network coding against pollution attacks in P2P converged ubiquitous networks” Springer Science , Published online 26 June 2013.

[2] Frédérique Oggier and Hanane Fathi “An Authentication Code Against Pollution Attack” in Network Coding, Proc. IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 19, NO. 6, DECEMBER 2011.

[3] Jing Dong, Reza Curtmola, Cristina Nita-Rotaru, “Practical Defenses Against Pollution Attacks in Intra-Flow Network Coding for Wireless Mesh Networks” WiSec’09, March 16–18, 2009, Zurich, Switzerland.

[4] F. Oggier and H. Fathi, “Multi-receiver authentication codes for network coding,” in Proc. 46th Annu. Allerton Conf. Commun., Control, Comput., 2008, pp. 1225–1231.

[5] P. Chou, Y. Wu, and K. Jain, “Practical network coding,” in Proc. Allerton Conf. Commun., Control, Comput., 2003, pp. 40–49.

[6] R. Ahlswede, N. Cai, S. Li, and R. Yeung, “Network Information Flow,” IEEE Trans. Information vol. 46, no. 4, pp. 1204-1216, July 2000.

[7] A.Perrig Ran Canetti, J. D. Tygar, D, “The TESLA Broadcast Authentication Protocol” in CrypoBytes, 5:2, Summer/ Fall 2002, pp 2-13.