IJIIT

International Journal of Infinite Innovations in Technology|ISSN:2278-9057
IJIIT|Volume-III|Issue-III|2014-2015 January|Paper-05
Reg. No.:20141205|DOI:V3I3P05

Security Issues for Wormhole Attacks in Wireless Ad Hoc Networks

Mali Basaveshwar Laxman
Scholar of Singhania University, India
Basu_mali@rediffmail.com

Abstract:

Wireless communications offer organizations and users many benefits, such as portability, flexibility, increased productivity, and lower installation costs. Wireless technologies cover a broad range of differing capabilities oriented toward different uses and needs. Wireless local area network (WLAN) devices. Ad hoc networks, such as those enabled by Bluetooth, allow data synchronization with network systems and application sharing between devices. However, risks are inherent in any wireless technology. The loss of confidentiality and integrity and the threat of denial of service (DoS) attacks are risks typically associated with wireless communications. Unauthorized users may gain access to an organization's systems and information, corrupt the organization's data, consume network bandwidth, degrade network performance, launch attacks that prevent authorized users from accessing the network, or use the organization's resources to launch attacks on other networks. As mobile ad hoc network applications are deployed, security emerges as a central requirement.

In this paper, we will discuss only the wormhole attack, a severe attack in ad hoc networks that is particularly challenging to defend against. The wormhole attack is possible even if the attacker has not compromised any hosts and even if all communication provides authenticity and confidentiality. In the wormhole attack, an attacker records packets (or bits) at one location in the network, tunnels them (possibly selectively) to another location, and retransmits them there into the network. The wormhole attack can form a serious threat in wireless networks, especially against many ad hoc network routing protocols and location-based wireless security systems. We consider a general mechanism, called packet leashes, for detecting and thus defending against wormhole attacks, and we present a specific protocol, called TIK, that implements leashes.

1. Introduction to Wormhole attack:

In a wormhole attack, an attacker receives packets at one point in the network, "tunnels" them to another point in the network, and then replays them into the network from that point. For tunneled distances longer than the normal wireless transmission range of a single hop, it is simple for the attacker to make the tunneled packet arrive with better metric than a normal multihop route, for example, through use of a single long-range directional wireless link or through a direct wired link to a colluding attacker. It is also possible for the attacker to forward each bit over the wormhole directly, without waiting for an entire packet to be received before beginning to tunnel the bits of the packet, in order to minimize delay introduced by the wormhole. Due to the

nature of wireless transmission, the attacker can create a wormhole even for packets not addressed to it self, since it can overhear them in wireless transmission and tunnel them to the colluding attacker at the opposite end of the wormhole. If the attacker performs this tunneling honestly and reliably, no harm is done; the attacker actually provides a useful service in connecting the network more efficiently. However, the wormhole puts the attacker in a very powerful position relative to other nodes in the network, and the attacker could exploit this position in a variety of ways. The attack can also still be performed even if the network communication provides confidentiality and authenticity, and even if the attacker has no cryptographic keys. Furthermore, the attacker is invisible at higher layers; unlike a malicious node in a routing protocol, which can often easily be named, the presence of the wormhole and the two colluding attackers at either endpoint of the wormhole are not visible in the route. The wormhole attack is particularly dangerous against many ad hoc network routing protocols in which the nodes that hear a packet transmission directly from some node consider themselves to be in range of (and, thus a neighbor of) that node. For example, when used against an on-demand routing protocol such as dynamic source routing (DSR)[4], or ad hoc on-demand distance vector (AODV)[8], a powerful application of the wormhole attack can be mounted by tunneling each ROUTE REQUEST packet directly to the destination target node of the REQUEST. When the destination node's neighbors hear this REQUEST packet, they will follow normal routing protocol processing to rebroadcast that copy of the REQUEST, and then discard without processing all other received ROUTE REQUEST packets originating from this same route discovery. This attack, thus, prevents any routes other than through the wormhole from being discovered, and if the attacker is near the initiator of the route discovery, this attack can even prevent routes more than two hops long from being discovered. Possible ways for the attacker to then exploit the wormhole include discarding rather than forwarding all data packets, thereby creating a permanent denial-of-service (DoS) attack (no other route to the destination can be discovered as long as the attacker maintains the wormhole for ROUTE REQUEST packets), or selectively discarding or modifying certain data packets.

2. Detecting wormhole attack

2.1. Packet leash mechanism for detection of worm hole:

In this section, we introduce the notion of a packet leash as a general mechanism for detecting and, thus defending against wormhole attacks. A leash is any information that is added to a packet designed to restrict the packet's maximum allowed transmission distance. Leashes are designed to protect against wormholes over a single wireless transmission; when packets are sent over multiple hops, each transmission requires the use of a new leash. We distinguish between geographical leashes and temporal leashes. A geographical leash ensures that the recipient of the packet is within a certain distance from the sender. A temporal leash ensures that the packet has an upper bound on its lifetime, which restricts the maximum travel distance, since the packet can travel at most at the speed-of-light. Either type of leash can prevent the wormhole attack, because it allows the receiver of a packet to detect

if the packet traveled further than the leash allows.

2.2. A. Geographical Leashes:

To construct a geographical leash, in general, each node must know its own location, and all nodes must have loosely synchronized clocks. When sending a packet, the sending node includes in the packet its own location P_s and the time at which it sent the packet t_s; when receiving a packet , the receiving node compares these values to its own location P_r , and the time at which it received the packet t_r . If the clocks of the sender and receiver are synchronized to within $\pm\Delta$, and v is an upper bound on the velocity of any node, then the receiver can compute an upper bound on the distance between the sender and itself d_{sr}. Specifically, based on the timestamp t_{sf} in the packet, the local receive time t_r, the maximum relative error in location information δ, and the locations of the receiver P_r and the sender P_s, then d_{sr} can be bounded by A standard digital signature scheme eg. RSA [10] or other authentication technique can be used to enable a receiver to authenticate the location and timestamp in the received packet. In certain circumstances, bounding the distance between the sender and receiver, d_{sr}, cannot prevent wormhole attacks; for example, when obstacles prevent communication between two nodes that would otherwise be in transmission range, a distance- based scheme would still allow wormholes between the sender and receiver. A network that uses location information to create a geographical leash could control even these kinds of wormholes. To accomplish this, each node would have a radio propagation model. A receiver could verify that every possible location of the sender (a radius around p_s) can reach every possible location of the receiver (a radius around p_r).

2.2. B. Temporal Leashes:

To construct a temporal leash, in general, all nodes must have tightly synchronized clocks, such that maximum difference between any two nodes' clocks is Δ. The value of the parameter Δ must be known by all nodes in the network, and for temporal leashes, generally must be on the order of a few microseconds or even hundreds of nanoseconds. This level of time synchronization can be achieved now with off-the-shelf hardware based on LORAN-C [6], WWVB [7], GPS [2, 11], or on-chip atomic clocks currently under development at NIST; although such hardware is not currently a common part of wireless network nodes, it can be deployed in networks today and is expected to become more widely utilized in future systems at reduced expense, size, weight, and power consumption. Although our general requirement for time synchronization is indeed a restriction on the applicability of temporal leashes, for applications that require defense against the wormhole attack, this requirement is justified due to the seriousness of the attack and its potential disruption of the intended functioning of the network. To use temporal leashes, when sending a packet, the sending node includes in the packet the time at which it sent the packet t_s; when receiving a packet, the receiving node compares this value to the time at which it received the packet t_r . The receiver is, thus, able to detect if the packet traveled too far, based on the claimed transmission time and the speed-of-light. Alternatively, a temporal leash can be constructed by instead including in the packet an expiration time, after which the receiver should not

accept the packet; based on the allowed maximum transmission distance and the speed-of-light, the sender sets this expiration time in the packet as an offset from the time at which it sends the packet. As with a geographical leash, a regular digital signature scheme or other authentication technique can be used to allow a receiver to authenticate a timestamp or expiration time in the received packet.

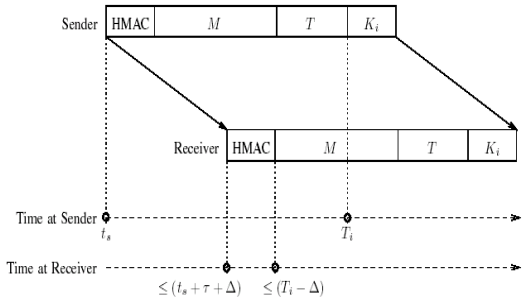
2.3 TIK (TESLA with instant) Protocol description:

Our TIK protocol implements temporal leashes and provides efficient instant authentication for broadcast communication in wireless networks. TIK stands for TESLA with instant key disclosure, and is an extension of the TESLA broadcast authentication protocol [9]. The intuition behind TIK is that the packet transmission time can be significantly longer than the time synchronization error. In these cases, the receiver can verify the TESLA security condition (that the corresponding key has not yet been disclosed) as it receives the packet (explained below); this fact allows the sender to disclose the key in the same packet, thus motivating the protocol name "TESLA with instant key disclosure."

TIK implements a temporal leash and, thus, enables the receiver to detect a wormhole attack. TIK is based on efficient symmetric cryptographic primitives (a message authentication code is a symmetric cryptographic primitive). TIK requires accurate time synchronization between all communicating parties, and requires each communicating node to know just one public value for each sender node, thus enabling scalable key distribution.

We now describe the different stages of the TIK protocol in detail: sender setup, receiver bootstrapping, and sending and verifying authenticated packets.

Fig 4.2 Timing of a packet in transmission using TIK



1) Sender Setup: The sender uses a pseudorandom function (PRF [3]) and a secret master key x to derive a series of keys K_0, $K_{1.....,}$ K_w, where $K_i = Fx(i)$. The main advantage of this method of key generation is that the sender can efficiently access the keys in any order. Assuming the PRF is secure, it is computationally intractable for an attacker to find the master secret key x, even if all keys K_0, $K_{1.....,}$ $K_{w-1,}$ are known. Without the secret master key x, it is computationally intractable for an attacker to derive a key K_i that the sender has not yet disclosed. To construct the PRF function F, we can use a pseudorandom permutation, i.e., a block cipher [5], or a message authentication code, such as HMAC [1].

The sender selects a key expiration interval and, thus, determines a schedule with which each of its keys will expire. Specifically, key K_0 expires at time T_0, key K1 expires at time $T_1 = T_0 + I\ldots$, & and key Ki expires at time $T_i = T_{i-1} + I = T_0 + i. I.$ The root of the resulting hash tree is, or simply. The value commits to all keys and is used to authenticate any leaf key efficiently. In a hash tree with $\log_2$ (w) levels, verification requires only $\log_2$ w hash function computations (in the worst case, not considering buffering), and the

authentication information consists of $\log_2 w$ values.

2) Receiver Bootstrapping: We assume that all nodes have synchronized clocks with a maximum clock synchronization error of Δ. We further assume that each receiver knows every sender's hash tree root m, and the associated parameters T_0 and I . This information is sufficient for the receiver to authenticate any packets from the sender.

3) Sending and Verifying Authenticated Packets: To achieve secure broadcast authentication, it must not be possible for a receiver to forge authentication information for a packet. When the sender sends a packet P, it estimates an upper bound t_r on the arrival time of the HMAC at the receiver. Based on this arrival time, the sender picks a key K_i that will not have expired when the receiver receives the packet's HMAC ($T_i > T_r + \Delta$). The sender attaches the HMAC to the packet, computed using key K_i, and later discloses the key K_i itself, along with the corresponding tree authentication values, after the key has expired.

Because of the time synchronization, the receiver can verify after receiving the packet that the key K_i used to compute the authentication has not yet been disclosed, since the receiver knows the expiration time for each key and the sender only discloses the key after it expires; thus, no attacker can know K_i and, therefore, if the packet authentication verifies correctly once the receiver later receives the authentic key K_i, the packet must have originated from the claimed sender. Even another receiver could not have forged a new message with a correct message authentication code, since only the sender knew the key K_i at the time t_r that the receiver received the packet. After the key K_i expires at time T_i, the sender then discloses key K_i (and the corresponding tree authentication values); once the receiver gets the authentic key K_i, it can authenticate all packets that carry a message authentication code computed with K_i. This use of delayed key disclosure and time synchronization for secure broadcast authentication was also used by the TESLA protocol [9].

Fig.4. 2 shows the sending and receiving of a TIK packet. The figure shows the sender's and receiver's timelines, which may differ by a value of up to the maximum time synchronization error Δ. The time t_s here are the time at which the sender S begins transmission of the packet, and time T_i is the disclosure time for key K_i. The packet contains four parts: a message authentication code (shown as HMAC in Fig. 4.2), a message payload (shown as M), the tree authentication values necessary to authenticate K_i (shown as T), and the key used to generate the message authentication code (shown as K_i). The TIK packet is transmitted by S as S –> R: $\{HMAC_{Ki}$ (M), M,T,$K_i\}$ Where the destination R may be unicast or broadcast. After the receiver R receives the HMAC value, it verifies that the sender did not yet start sending the corresponding key K_i, based on the time T_i and the synchronized clocks. If the sender did not yet start sending K_i, the receiver verifies that the key K_i at the end of the packet is authentic (using the hash tree root m and the hash tree values T), and then uses K_i to verify the HMAC value in the packet. If all these verifications are successful, the receiver accepts the packet as authentic. The TIK protocol already provides protection against the wormhole attack, since an attacker who retransmits the packet will

most likely delay it long enough that the receiver will reject the packet because the corresponding key has already expired and the sender may have disclosed it. However, we can also add an explicit expiration timestamp to each packet for the temporal leash, and use TIK as the authentication protocol. For example, each packet could include a 64-bit timestamp with nanosecond resolution, allowing over 580 years of use starting from the epoch. Since the entire packet is authenticated, the timestamp is authenticated.

A policy could be set allowing the reception of packets for which the perceived transmission delay, i.e., the arrival time minus the sending timestamp, is less than some threshold. That threshold could be chosen anywhere between T-Δ and T+Δ , where the more conservative approach of T-Δ never allows tunnels but rejects some valid packets, and the more liberal approach of T+Δ never rejects valid packets, but may allow tunneling of up to 2cΔ past the actual normal transmission range. With a GPS-disciplined clock, time synchronization to within Δ=183 ns with probability $1-10^{-10}$ is possible. If a transmitter has a 250 m range, the threshold T-Δ accepts all packets sent less than 140 m and some packets sent between 140 and 250 m; the threshold T+Δ accepts all packets sent less than 250 m but allows tunneling of packets up to 110 m beyond that distance.

6. Conclusion:

In this paper, we have introduced the wormhole attack, a powerful attack that can have serious consequences on many proposed ad hoc network routing protocols; the wormhole attack may also be exploited in other types of networks and applications, such as wireless access control systems based on physical proximity. To detect and defend against the wormhole attack, we introduced packet leashes, which may be either geographic or temporal leashes, to restrict the maximum transmission distance of a packet. Finally, to implement temporal leashes, we presented the design and performance analysis of a novel, efficient protocol, called TIK, which also provides instant authentication of received packets.

7. References

1] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *Advances in Cryptology – CRYPTO '96*, edited by Neal Koblitz, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, Berlin Germany, 1996.

[2] Tom Clark. Tom Clark's Totally Accurate Clock FTP Site. Greenbelt, Maryland. Available at ftp://aleph.gsfc.nasa.gov/GPS/totally.accurate.clock/.

[3] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[4] David B. Johnson, David A. Maltz, and Josh Broch. The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In *Ad Hoc Networking*, edited by Charles E. Perkins, chapter 5, pages 139–172. Addison-Wesley, 2001.

[5] Shafi Goldwasser and Mihir Bellare. Lecture Notes on Cryptography. Summer Course "Cryptography and Computer Security" at MIT, 1996–1999, August 1999.

[6] David L. Mills. A Computer-Controlled LORAN-C Receiver for Precision Timekeeping. Technical Report 92-3-1, Department of Electrical

and Computer Engineering, University of Delaware, Newark, DE, March 1992.
[7] David L. Mills. A Precision Radio Clock for WWV Transmissions. Technical Report 97-8-1, Department of Electrical and Computer Engineering, University of Delaware, Newark, DE, August 1997.
[8] Charles E. Perkins and Elizabeth M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, February 1999.
[9] Adrian Perrig, Ran Canetti, Doug Tygar, and Dawn Song. Efficient Authentication and Signature of Multicast Streams over Lossy Channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 56–73, May 2000.
[10] Ron L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.