HiMAC：一种用于消息认证和加密的分层安全协议

张永棠 [1,2]

[1]广东东软学院计算机学院 佛山 中国 528225
[2]南昌工程学院江西省协同感知与先进计算技术研究所 南昌 中国 330003

摘要 为检测并阻止恶意节点伪装成新的可信节点攻击移动自组织网络，该文提出了一种用于消息认证和加密的分层安全协议(HiMAC)。该协议将分层消息认证码用于保护移动 Ad-Hoc 网络中的数据传播。在源和目标之间的由中间节点转发分组时动态地计算可信路由，在每个中间节点对数据包进行签名和加密，防止攻击者篡改数据包或修改其跳数，实现数据可信传输。在 NS2 模拟器中，运用 Crypto++库中的 RSA 算法对 HiMAC 进行测试。结果表明: HiMAC 可以检测和阻止对 MANET 节点和数据包的攻击；与原有的 A-SAODV 安全机制相比, HiMAC 平均跳数减少了 47.1%，平均队列长度减小了 35.5%，节点数据包数量降低 2.5 倍，其性能明显优于 A-SAODV。尽管 HiMAC 的密码操作给路由协议带来了额外的开销，但由于 HiMAC 采用基于信任机制动态建立安全路由，使得节点能够动态地选择路径上的下一个节点，不必始终保持安全路由，使得 HiMAC 中的增减开销可以相互抵消达到平衡。

关键词 网络安全；基于身份的密码学；消息认证；可信计算；移动自组织网络
中图法分类号 TP393.08 DOI 号 10.19363/J.cnki.cn10-1380/tn.2022.05.07

HiMAC: A Hierarchical Security Protocol for Message Authentication and Encryption

ZHANG Yongtang [1,2]

1. School of Computer, Guangdong Neusoft Institute, Foshan 528225, China
2. Institute of Cooperative Sensing and Advanced Computing Technology, Nanchang Technology Institute, Nanchang 330003, China

Abstract In order to detect and prevent malicious nodes from pretending to be new trusted nodes attacking mobile Ad-Hoc network, a Hierarchical Message Authentication Code (HiMAC) for message authentication and encryption is proposed in this paper. The protocol uses layered message authentication code to protect data transmission in mobile Ad-Hoc networks. When the packet is forwarded between the source and the target, the trusted route is calculated dynamically, and the packet is signed and encrypted at each intermediate node to prevent the attacker from tampering with the packet or modifying its hop number, so as to realize the trusted transmission of the data. In NS2 simulator, the RSA algorithm in Crypto library is used to test HiMAC. The results show that HiMAC can detect and prevent attacks on MANET nodes and packets. Compared with the original A-SAODV security mechanism, the average hop number of HiMAC is reduced by 47.1%, the average queue length is reduced by 35.5%, and the number of node packets is reduced by 2.5 times. Although the password operation of HiMAC brings additional overhead to the routing protocol, because HiMAC uses trust-based mechanism to dynamically establish secure routing, nodes can dynamically select the next node on the path without always maintaining a secure route. So that the increase and decrease in HiMAC can offset each other to strike a balance.

Key words network security; identity-based cryptography; message authentication; trusted computing; mobile Ad-Hoc networks

1 引言

移动计算是指智能终端设备在动态无线通信环境下实现资源共享和数据传输，是移动通信和云计算发展而新起的新技术。移动自组织网络(MANET)是分散类型的无线网络，不依赖于预先存在的基础设施，是每个节点通过将数据转发到其他节点来参与路由。MANET 可以使用各种路由方案，如逐跳通

通讯作者：张永棠，博士，教授, Email:gov211@163.com.
本课程得到国家自然科学基金(No.61663029)，广东省高校重点平台与特色创新项目(No. 2020KTSCX771)资助。
收稿日期: 2021-02-04；修改日期: 2021-05-17；定稿日期: 2022-03-15

信或其他经典和现代方法[1]。除了经典路由之外, MANET 还可以使用洪泛来转发数据[2]。当研究网络拥塞时, 洪泛本身就成了一个重要问题。

随着信息共享和数据传播的 MANET 应用程序的快速增长, 优化网络资源和实现数据安全的需求已成为研究界的首要关注点。由于存在不断变化的拓扑结构和节点的高移动性等限制因素, 在MANET 中实现强大的安全性是一项非常具有挑战性的任务。网络拓扑是随着节点加入和移除不断的变化。在这种情况下, 难以在每个节点处维持恒定的路由表, 这成为消耗节点有限资源的非常广泛的过程。此外, MANET 节点在某些情况下可能具有高移动性, 例如自然灾害和疏散场景[3]。考虑到上述挑战, MANET 中的数据传播需要采用统一的安全方法来保持快速和成功的通信效率。

该文提出了一种用于消息认证和加密的分层安全协议(Hierarchical Message Authentication Code, HiMAC), 允许源移动节点安全地将数据包发送到目标移动节点, 同时确保通过可信中间移动节点转发数据包。该协议确保在从数据包发送到目的地时对数据包执行攻击时, 将检测到恶意节点。因此, 该协议创建了源与目标之间的可信路由。与其他协议不同的是, 该协议不会在每个节点维护可信路由, 而是在由中间节点转发分组时动态地计算路由。可信路由确保数据包仅由受信任节点转发。为做到这一点, 每个中间节点在使用目标的公钥加密之前, 将自己的签名和时间戳添加到数据包。当目的地收到数据包时, 则对其进行分层解密, 并通过检查其签名来验证每个中间节点。因此, 目的地可以反向跟踪数据包的路由, 并确保转发数据包的所有节点都是可信任的。

除了验证中间节点之外, HiMAC 还致力于在 MANET 节点之间提供可信赖性。在以前的工作中, 维护节点之间的信任取决于是否存在将证书分发给节点的集中权限。但是, 这种方法在 MANET 中可能无法正常工作, 因为分配的可信第三方可能随时离开网络。此外, 为新到达的节点建立信任可能需要很长时间。基于这些原因, HiMAC 通过所有节点之间的分布式协作来维护信任机制。尽管 MANET 中存在许多关于安全通信的建议, 但它们仍然面临着如何在 MANET 框架中保持效率、信任和安全的问题。

2 文献综述

近年来, 为解决 MANET 安全性问题, 学术界提出了大量的安全机制和协议。

Thanuja 等人[4]介绍了 Ad-Hoc 网络中对路由的不同攻击, 并给出了一种新的安全的请求式 Ad-Hoc 网络路由协议, 解决路由路径的各种类型的拒绝服务攻击。Nagaraju 等人[5]提出了移动自组网安全消息传输(SMT)协议, 他们描述了 SMT 协议更好地匹配, 以支持 Ad-Hoc 网络环境中的实时通信的 QoS。Gurung 等人[6]针对 Ad-Hoc 网络提出了一种安全有效的距离矢量路由协议, 该协议使用单向散列函数而不是加密操作来保护路由消息。

Mohan 等人[7]对移动自组网中各种可能的攻击和对策进行了调查, 提出了一种移动主机中间检测恶意攻击的路由安全算法, 并将其命名为混杂监听路由安全算法(promiscuous listening routing security algorithm, PLRSA)。他们提出的算法本质上是分布式的, 不需要在主机之间进行通信。PLRSA 中的每个节点都可以切换到混杂监听模式, 截获通过移动主机的所有数据包, 以监视附近的其他节点。当一个节点执行恶意行为时, 例如丢弃或篡改数据包, 附近的其他节点将检测到恶意行为[8]。

Singh 等人[9]讨论了在数据链路和网络层保护移动自组网的主要安全问题。他们首先确定了这两层的安全需求, 然后确定了使用多条防御恶意攻击路线创建安全即席网络的设计标准。Yin 等人[10] 讨论了导致拒绝服务(DoS)攻击的灰洞攻击。在灰洞攻击中, 对手会悄悄地丢弃发送给它的部分或全部数据包, 而不是转发它们。Silveira 等人 [11] 提出了一种按需多路径路由协议, 称之为安全多路径路由协议(SecMR), 并分析了其安全性。Tan 等人 [12] 描述了一些 MANET 安全协议的分层安全方法、设计标准和性能分析。

Sarfaraz 等人[13]提出了一种具有主动安全方法的安全路由协议。在本文中, 作者只允许合法节点参与引导过程, 而不是在对手节点参与路由协议时尝试检测对手节点。Yadav 等人[14]提出了两种针对 MANET 的入侵检测技术, 该技术依靠邻居节点的协作工作来检测该邻居中的恶意节点。Indirani 等人[15]提出了一个组密钥协议协议, 用于在没有任何固定基础设施的 MANET 环境中实现端到端安全。Singh 等人[16]比较了 SAODV 和 TAODV, 分别通过加密和基于信任的方式解决路由安全问题。它们还提供了实际资源有限硬件的性能比较。Wang 等人[17]讨论了 DSR 中路由发现过程的安全性。

Maheshbhai 等人[18]为 Ad-Hoc 网络提出了一种安全可靠的证书链恢复协议。在提议的框架中, MANET 用户通过发布和管理公钥证书来担任认证

服务的角色, 可以选择最短和最安全的证书链以减少通信开销并抵抗可能产生错误证书的受损节点。Lim 等人[19]提出了一种基于网格的多路径路由方案, 以使用安全的相邻位置信任验证协议来发现所有可能的安全路径。更好的链路最佳路径由 Dolphin 回声定位算法确定, 以实现高效通信。

Toor 等人[20]提出了一种自适应安全 AODV(A-SAODV)协议, 该协议在 SAODV 的基础上, 针对 MANET 网络开放性特点, 采用了自适应机制、门限机制和信任级别机制对 SAODV 进行了优化。缩短端与端延迟、提高吞吐量、取得安全性与效率的平衡。在实际网络部署中得到了广泛的应用[21-22]。

由于 MANET 可以遇到快速的拓扑变化, 安全性成为在攻击者成功执行攻击之前检测和阻止攻击者的首要问题。现有的安全解决方案能够保护网络免受某种类型的攻击, 但仍然容易受到其他类型的内部和外部攻击。尽管存在许多缓解这些攻击的解决方案, 但它们对于网络中不断变化的拓扑结构来说还不够。因此, 需要一种统一的机制来防止数据包丢失和篡改、数据包重放、模拟和虚假数据攻击。本文将不同节点之间的信任机制与强认证方案相结合, 以便检测和避免网络受到攻击。

上述文献均尝试了各种类型的 Ad-Hoc 网络中的安全威胁的不同解决方案。但并没有框架同时建立了高安全性, 高效率和信任分类的有力证据。这些文献均通过监视新节点在一段时间内的行为来建立这种信任, 其难以建立新到达节点的可信度, 没有检测到恶意行为则将该节点分类为可信赖的。因此, 本文提出的算法可以检测攻击者长时间隐藏其恶意行为, 防止被其他节点声明为可信任之后执行攻击。

3 HiMAC 描述

3.1 信任机制

在 HiMAC 系统中, 认为信任应该基于身份而不仅仅基于行为。由于移动节点实际上是操作它们的人, 因此移动节点的信任应该与其用户的身份相结合。新到达的节点将向其邻居广播其身份。如果其中一个可信邻居确认其可信度, 则该节点被其他邻居保存为信任。如果没有邻居知道新节点, 则它将保持为可疑状态, 直到知道新节点的网络中的某个可信节点发现其存在并声明其可信度为止; 或者直到某个时间过去之后, 新节点的行为被归类为正常并且建立了其可信度。

为维护这种信任机制, 将单个参数添加到邻居列表和路由表中, 这些参数由路由协议在每个节点维护。信任参数值包含恶意、不明确、可能受信任、受信任 4 种值。具体信任机制描述如下:

假设从源 S 到目的地 D 的多条路由示例图如图 1 所示, 想要将数据包从节点 S 发送到另一个节点 D 可选择的“下一跳”节点集为 $\{A_1, B_1, C_1\}$。在此阶段, 从该集合中选择某个节点时会考虑两个因素: 信任值和路由效率。S 首先检查列表中每个节点的信任。如果列表中只有一个节点是“受信任的”, 则将其选为“下一跳”。如果列表中的多个节点是“受信任的”, 则 S 在这些节点中选择位于最有效路由内的节点。即, 到目的地的总路径具有最低成本的节点。例如, 在图 1 中, 假设 B_1 和 C_1 都是“可信的”, 并且路线 $B_1 \to B_2 \to B_3$ 的成本等于 15, 而 $C_1 \to C_2 \to C_3$ 的成本等于 20。则 S 选择 B_1 作为“下一跳”。

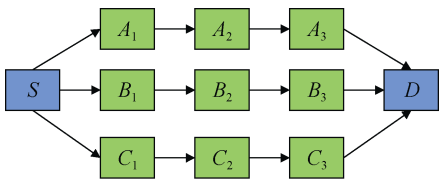


图 1 从源 *S* 到目的地 *D* 的多条路由示例

Figure 1 Example of multiple routes from source *S* to destination *D*

在将包发送到 B_1 之前, S 使用其私钥运用 MAC 算法生成消息签名。然后, S 将消息、生成的签名和时间戳值组合(连接)到一个块中, 并使用接收者 D 的公钥对该块进行加密。在 HiMAC 中, 每个数据包在其有效载荷中都将包含一个额外的元素, 即遍历节点的列表。此列表与包含消息、签名和时间戳的块分开加密。因此, S 将其 ID 号添加到遍历节点列表中。然后 S 用 D 的公钥加密这个列表。接下来, S 将加密块和遍历节点的加密列表组合成一个包, 并将该包发送到“下一跳”。请注意, 将使用一个特殊的字符分隔符将加密块从加密列表中分离出来, 并在加密块中分离消息、签名和时间戳。加密遍历节点列表的原因是为了防止攻击者在不知道目标的情况下恶意更改包中的跃点数。在 SAODV 中, 使用哈希链保护跃点计数。在 HiMAC 中, 我们通过集成遍历节点的列表并按照下面的说明对其进行分层加密来保护跃点计数。因此, 从 S 发送到“下一跳”的最终消息为

$$M_1 = E_{KPU_D}\left\{M_p \left\| MAC_{KPR_S}\left(M_p\right) \right\| T_s\right\} \left\| E_{KPU_D}\left(ID_s\right)\right. \quad (1)$$

其中，M_p 是原始消息，M_1 是密码消息，ID_s 是 S 的 ID 号，T_s 是时间戳值，E_{KPU_D} 表示使用节点 D 的公钥加密，MAC_{KPR_s} 表示使用节点 S 私钥的 MAC 签名，‖是一个特殊字符。

每个中间节点 I 将执行 S 所执行的相同步骤，但 I 将使用 M_1 作为消息，而不是使用 M_p。首先，I 将加密块与加密节点列表分离，将加密块保存为 M_{11}，将加密列表保存为 M_{12}。然后计算“下一跳”节点，使用自己的私钥生成 M_{11} 的签名，生成块 $\{M_{11}\|MAC(M_{11})\|\text{timestamp}\}$，并使用目的地的公钥对生成块进行加密，并将其自身添加到 M_{12} 中，并使用目的地的公钥加密结果。因此，由节点 I 转发到“下一跳”的消息 M_2 为

$$M_2 = E_{KPU_D}\left\{M_{11}\left\|MAC_{KPR_I}(M_{11})\right\|T_s\right\}\left\|E_{KPU_D}(M_{12}\|ID_s)\right. \tag{2}$$

且

$$M_{11} = E_{KPU_D}\left\{M_p\left\|MAC_{KPR_s}(M_p\|T_s)\right.\right\},$$

$$M_{12} = E_{KPU_D}(ID_s).$$

这个过程将在每个中间节点重复，直到数据包到达 D。在这里，D 将采用反向分层解密技术来检索转发数据包的每个中间节点的 ID，使用后者的 MAC 签名检查这些节点的真实性，并重复这个过程，直到它计算出原始消息 M_p。如果转发数据包的所有中间节点都被 D“信任”，并且发现所有这些节点的签名都是正确的，那么该数据包将被认为是完全安全的。如果转发数据包的一个或多个中间节点被 D 分类为“恶意”或“不明确”，D 可能会根据应用程序和用户的决定丢弃数据包。当目的地节点 D 接收到密码消息 M_n 时，其步骤可以详细描述为:

第一步: 将加密块(M_{n1})与加密节点列表(M_{n2})分开，然后 D 对这两部分进行解密，并从 M_{n2} 中检索转发数据包的最后一个节点 I_n 的 ID。

第二步: D 将消息 $M_{(n-1)}$、MAC 签名 MAC_n 和时间戳 T_{s_n} 从 M_{n1} 解密中分离出来。

第三步: D 使用 I_n、$M_{(n-1)}$ 和 MAC_n 作为 MAC 算法的输入，用来检查签名的正确性并确保 MAC_n 的真实性。如果确定了其真实性并且 D“信任”了 I_n，则进入下一步。其中 D 解密 $M_{(n-1)2}$(从 M_{n2} 中删除 I_n 后获得)以获取在 I_n 之前转发数据包的中间节点的 ID，即 $I_{(n-1)}$。

第四步: D 解密 $M_{(n-1)}$ 以获得 $M_{(n-2)}$、$MAC_{(n-1)}$ 和 $T_{S_{(n-1)}}$；之后，D 使用 $I_{(n-1)}$ 的公钥作为 MAC 算法的输入，以检查 $I_{(n-1)}$ 的真实性。

解密的节点列表将包含一个元素，即 S 的 ID。解密的块将包含原始消息 M_p、S 的 MAC 签名和时间戳。在最后阶段，D 将使用其签名对 S 进行身份验证，并在应用程序中使用来自 M_p 的数据。此外，D 将检查加密前中间节点添加的时间戳的列表 $L\{T_s\}$。如果一个或多个时间戳是旧的，或者时间戳不是按降序排列，这表明存在数据包篡改或某种重播攻击的可能性，在这种情况下，D 将丢弃数据包。

值得注意的是，HiMAC 依赖于 MANET 中有效的公钥加密机制的存在。在该方案中，每个移动节点都应该在网络中保存彼此的“可信”节点的公钥。换句话说，一个节点只与其他“受信任”节点共享其公钥。具体实现方法如下:

假设一个节点 N 当前正在将一个新节点 N_n 分类为“不明确”。现在，N 从一个“可信”节点 N_T 接收一个邻居广播数据包，该节点 N_T 包含一个“可信”节点 N_n。N 将 N_n 更改为“潜在受信任”，并询问用户是否应信任 N_n。假设用户承认 N_n 是“可信”的，那么 N 将 N_n 的信任参数更改为“可信”。之后，N 检查 N_T 是否在范围内。如果是，N 向 N_T 发送一个“公钥请求”包，请求它获取 N_n 的公钥。后者使用 N 的公钥对 N_n 的公钥进行加密，并将结果发送到“公钥应答”包中的 N。如果 N_T 不在范围内，N 向其邻居发送一个邻居广播包，请求 N_n 的公钥。当 N 从邻居 N' 接收到包含 N_n 的公钥的“公钥应答”包时，它检查 N' 是否“可信”。如果 N' 可信，N 保存接收到的 N_n 的公钥；否则 N 丢弃 N' 的包。

HiMAC 的主要目标是防止入侵者捕获通过其传输数据的特定路由，并对通过该路由的数据包执行攻击。因此，重点关注数据缓和攻击，重放攻击，洪水攻击和模拟攻击。入侵者将不被允许执行洪水攻击，因为它不会被归类为“受信任”。因此，其数据包不会被中间节点转发。如果入侵者对数据包执行篡改攻击，则目标将发现攻击，因为它无法正确解密收到的消息。即使入侵者遵循 HiMAC 规则并使用其签名对数据包进行签名，目的地将无法验证入侵者，

因为它不知道入侵者的公钥。此外, HiMAC 可用于检测捕获合法消息的攻击者的重放攻击, 并在以后将其作为新消息发送。这可以通过检查中间节点添加的时间戳列表在目的地完成。最后, HiMAC 使目标节点能够检测到模拟攻击。如果将节点 N_a 作为节点 N_o 添加到数据包中, 那么目标节点将检测到攻击, 因为它将无法验证 N_a 的虚假签名, 且不同于预期的 N_o 签名。

3.2 算法描述

HiMAC 协议的操作示意图如图 2 所示。

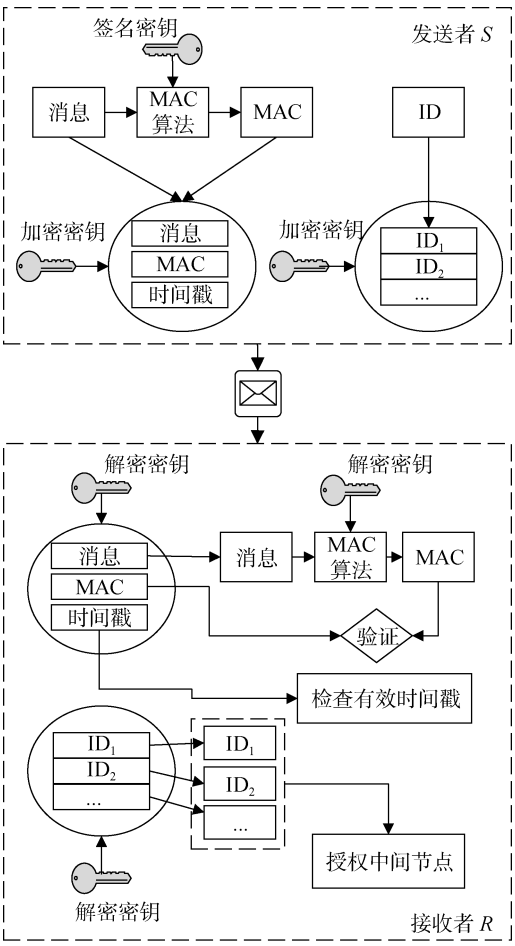


图 2 从发送者 *S* 到接收者 *R* 的 HiMAC 操作

Figure 2 HIMAC operation from sender *S* to receiver *R*

数据消息将从发送者 S 发送到接收者 R。发送者 S 将消息与 MAC 结合在一起, 该消息是借助于 MAC 算法, 并带有时间戳, 消息需要通过中间节点 $I_1, I_2, \cdots$。除了 ID 之外, 每个中间节点还将其 MAC 和时间戳添加到消息中的 ID 列表中。接收节点 R 将解密消息的每一层以生成相应的 MAC, 并在解密 ID 列表之后检查相应节点的身份。在接收器端, 消息被解密并且 MAC 与发送者 MAC 匹配并被检查, 如果它是相同的, 则 R 确认发送者的完整性。否则, 消息完整性未经过认证, 应删除该消息。

发送者 S、中间节点 I 和接收者 R 的 HiMAC 算法分别如算法 1、算法 2 和算法 3 所示。

算法 1. 发送者 S 的算法.

输入: 接收者 R, 消息 M_S, 私钥 P_r, 时间戳 T_S, MAC 算法 $M_{\text{algo}}(\text{Message,Key})$.

输出: $\text{Encrypted}(M_S, MAC(M), T_S) + \text{Encrypted}(S)$

过程:

1. 应用 $M_{\text{algo}}(M_S, P_r)$ 算法生成签名 $MAC(M)$;

2. 将 $MAC(M)$ 和 T_S 添加到输入消息 M_S 以获取 $(M_S, MAC(M), T_S)$;

3. 使用 R 的公钥加密结果;

4. 使用 R 的公钥加密 ID 并将其添加到消息中.

算法 2. 中间节点 I 的算法.

输入: 接收者 R ,MAC 算法 $M_{\text{algo}}(\text{Message,Key})$, 消息 $\text{Encrypted}(M, MAC(M), T_S) + \text{Encrypted}(L)$, 私钥 P_r, 时间戳 T_{SI}.

输出: $\text{Encrypted}(M_I, MAC(M_I), T_{SI}) + \text{Encrypted}(\text{Encrypted}(S) + I)$.

过程:

1. 将消息分为两部分: $\text{Encrypted}(M_S, MAC(M), T_S)$ 和 $\text{Encrypted}(L)$.

2. 应用 $M_{\text{algo}}(M_I, P_r)$ 算法生成签名 $MAC(M_I)$.

3. 将 $MAC(M)$ 和 T_{SI} 添加到 M_I 以获取 $(M_I, MAC(M_I), T_{SI})$,并使用 R 的公钥对其进行加密;

4. 将 ID 添加到 $\text{Encrypted}(L)$, 并使用 R 的公钥对其进行加密.

算法 3. 接收者 R 的算法.

输入: 发送者 S, S 的私钥 P_u, 中间节点的公钥 $\{P_{UI}\}$, 消息 $\text{Encrypted}(M, MAC(M), T_S) +$

Encrypted(L).

输出: 检查完整性, 消息 M_S.

过程:

do

{

1. 将消息分为两部分 M_1=Encrypted $(M_S, MAC(M), T_S)$ 和 M_2=Encrypted(L);

2. 使用私钥解密 M_2, 并从中提取节点 I 的 ID;

3. 使用私钥解密 M_1, 并从中提取 M、$MAC(M)$ 和 T_S, 并保存 T_S;

4. 应用 $M_{\text{algo}}(M + MAC(C), P_{UI})$ 检查 $MAC(M)$ 的有效性;

5. if $MAC(M)$ 无效

输出完整性错误并退出;

end if

6. if $MAC(M)$ 有效

继续使用 M 和 Encrypted$(L-1)$ 进行下一次迭代;

end if

}

while(Encrypted(L) 非空)

7. 检查所有时间戳 T_S 的值和顺序;

8. 如果一个或多个 T_S 是旧的值或 T_S 顺序不对, 则输出有效性错误并退出;

9. 将最终消息 M_S 传递给应用程序.

4 系统参数分析

本节将分析影响系统参数的各种因素, 并将分析结果与下一节中的模拟结果进行比较。

HIMAC 加密/解密消息所需的时间是系统的一个重要参数。当消息由源节点或中间节点加密并且在目的节点解密时, 两个主要因素将影响加密/解密所需的时间: 1)RSA 密钥的大小; 2)消息的大小。文献[23]证明了随着密钥大小的增加, RSA 加密/解密延迟将增加。此外, 众所周知, 当我们想要使用 RSA 加密大型消息时, 首先使用对称密钥加密算法(例如 AES, 3DES)[24]生成对称密钥, 并使用它来加密消息, 然后我们用 RSA 公钥加密对称密钥。因此, 当使用 RSA 加密/解密大型消息时, 我们使用对称和非对称加密和解密。

将考虑发送者 S 发送的是大小为 S_{d0} 字节的纯文本消息。在源节点和每个中间节点, 将在加密消息中添加附加数据。因此, 应加密的消息大小将在每个中间节点处增加。AES 的加密/解密延迟取决于消息大小 S_d 和处理器的每秒百万指令数 C_p。AES 的加密延迟为:

$$T_{AES-Enc}(S_d, C_p) = \frac{8 \times S_d}{128} \times \frac{T_{AES-Enc}}{C_p} \tag{3}$$

其中 S_d 用 bit 位表示, C_p 用 MIPS 表示。此外, $T_{AES-Enc}$ 是加密一个数据块所需的处理周期数, 等于 128 位密钥的 6168 个处理周期。计算解密一个数据块所需的处理周期数 $T_{AES-Dec}$, 并用它来计算一个数据包的总解密延迟, $T_{AES-Dec}$ 表示为:

$$T_{AES-Dec}(S_d, C_p) = \frac{8 \times S_d}{128} \times \frac{T_{AES-Dec}}{C_p} \tag{4}$$

为分析 HiMAC 的性能, 需要知道两个操作所花费的时间: 1)使用 AES 密钥加密消息的两部分的耗时; 2)使用 RSA 公钥加密 AES 密钥的耗时。 其中, 第二个操作的延迟在所有节点处都是恒定的, 因为它们对 RSA 和 AES 密钥使用相同的大小, 并且此延迟将等于单个 RSA 加密操作的延迟。 对于第一部分, 由于使用 AES 的加密/解密取决于消息大小 S_d, 需要在源节点和每个中间节点 N_i 处导出消息大小。

在源节点, 消息的两部分将使用 AES 密钥分别加密。第一部分的大小将等于 S_{d0} 字节数、签名大小及时间戳大小三者之和。由于 RSA 签名大小始终等于所用密钥的大小, 因此在使用 1024 位密钥时, 我们将获得大小为 128 字节的签名。另外, 考虑到时间戳的大小等于 4 个字节, 第一部分的总大小将等于 $S_{d0}+132$ 个字节。知道 AES 为消息添加了填充以使其成为 AES 块大小的倍数, 则消息大小可以表示为:

$$S_{d1} = \left\lceil \frac{S_{d0} + 132}{Bl} \right\rceil \times Bl \tag{5}$$

其中 Bl 是 AES 中一个加密块的大小, 等于 128 位。对于整个消息的第二部分, 它将取决于中间节点的数量。在本节中, 我们将推导出网络中任意两个节点之间的平均预期跳数 $E[H]$(即中间节点)。假设每个中间节点的 ID 需要 2 个字节, 则第二部分的大小将等于 $2 \times E[H]$。在加密之后, 第二部分的大小对于所

有中间节点将是相同的, 其等于 Bl , 且 $Bl > 2\times E[H]$。

与源节点类似, 每个中间节点的加密过程: 1)使用 128 位 AES 密钥对前一节点加密的消息的两部分进行加密; 2)使用接收者的公共 RSA 密钥加密 AES 密钥。消息的第二部分的大小和加密的 AES 密钥的大小将与先前为源节点计算的大小相同。关于消息的第一部分, 我们将得到如下: 在第一个中间节点, 消息的第一部分的大小是 S_{d1}。第一个中间节点将添加此部分的签名和时间戳, 因此, 总大小将等于 $S_{d1}+132$, 然后它将添加填充以使 AES 块大小的最终大小倍数, 因此中间节点消息大小为

$$S_{d2}=\left\lceil \frac{S_{d1}+132}{Bl} \right\rceil \times Bl \tag{6}$$

可见, 将由节点 N_i 加密的消息第一部分的大小为

$$S_{d(i+1)}=\left\lceil \frac{S_{di}+132}{Bl} \right\rceil \times Bl \tag{7}$$

在得出加密/解密延迟之前, 计算 MANET 中两个节点之间的平均期望跳数。假设一个矩形拓扑, 其面积为 $a\times b$, 节点分布均匀。如果两个节点之间的距离 $k \leqslant r_0$, 则两个节点可以形成直接链路, 其中 r_0 是最大节点传输范围。试图计算任意两个节点之间的期望跳数。根据随机几何[25], k 的概率密度函数为

$$f(k)=\frac{4k}{a^2b^2}\left(\frac{\pi}{2}ab-ak-bk+0.5k^2\right) \tag{8}$$

且 $0 \leqslant k < b < a$。可以得出结论, 如果两个节点之间的距离为 k_0, 则当存在足够数量的节点以形成连接网络时, 它们之间的跳数将趋向于 k_0/r_0。因此, $E(H)$ 是网络中任意两个节点之间的预期最小跳数, 相当于将预期距离 $E(k)$ 除以 r_0。需要注意的是, $E(H)$ 的值代表一个下限, 因为当节点在网络中是稀疏的时, 由于必须通过较长的距离路由才能到达某个节点, 因此跃点数不可避免地会增加。

为计算 HIMAC 加密和解密操作的平均延迟, 使用 Crypto++库中的 RSA 算法进行延迟基准测试。表 1 展示了基准测试结果。

RSA 密钥大小在 512～4096 位之间变化, 每个结果是 100 次操作的平均值。其中 TE_{RSA} 和 TD_{RSA} 分别是单个数据块 RSA 操作的加密和解密延迟, 而 TS_{RSA} 和 TV_{RSA} 分别是单个数据库块 RSA 签名和验证操作的延迟。注意到在表 1 中, 所有延迟都随着密钥大小的增加而增加。

表 1　基准测试结果

Table 1　Benchmark results

操作 (ms)	RSA 密钥大小(位)				
	512	1024	2048	3072	4096
TE_{RSA}	0.0035	0.0186	0.0742	0.2484	0.5776
TD_{RSA}	0.0591	0.3712	2.731	6.629	12.438
TS_{RSA}	0.0587	0.3702	2.716	6.581	12.252
TV_{RSA}	0.0035	0.0185	0.0739	0.247	0.5441

最后, 推导 HIMAC 的平均加密和解密延迟。对于加密, 每个节点 N_i ($i=0,1,\cdots$, 其中 $i=0$ 是源节点, $i>0$ 是中间节点)将加密三个不同的部分。分别为: 第一部分是包含来自前一个节点的消息(或者源节点的纯文本消息)、签名和时间戳的部分, 这部分的延时等于 $S_{d(i+1)}$; 第二部分是 ID 编号列表, 其延时等于 $2\times E(H)$; 第三部分是用 RSA 公钥加密 AES 会话密钥, 需要相当于 TE_{RSA} 的延时。因此, HIMAC 的平均加密延迟可计算为

$$TE_{HiMAC}=TS_{RSA}+\frac{1}{(E[H]+1)}\sum_{i=0}^{E[H]}\left(\left\lceil\frac{8\times S_{d(i+1)}}{128}\right\rceil\times\frac{T_{AES-Enc}}{C_p}\right)+\left(\left\lceil\frac{8\times\left(2\times(E[H]+1)\right)}{128}\right\rceil\times\frac{T_{AES-Enc}}{C_p}\right)+TE_{RSA} \tag{9}$$

对于将在目标节点上执行的解密操作, 其公式可以类似于加密延迟的公式。其主要区别在于: 加密延迟计算为所有中间节点的加密操作的平均值, 而解密延迟计算为目标节点上所有解密操作的总和。因此, 平均预期解密延迟为

$$TD_{HiMAC}=(E[H]+1)\times\left(\begin{array}{l}TD_{RSA}+TV_{RSA}\\+\left(\left\lceil\frac{8\times\left(2\times(E[H]+1)\right)}{128}\right\rceil\times\frac{T_{AES-Enc}}{C_p}\right)\end{array}\right)+\sum_{i=E[H]}^{0}\left(\left\lceil\frac{8\times S_{d(i+1)}}{128}\right\rceil\times\frac{T_{AES-Enc}}{C_p}\right) \tag{10}$$

从式(9)和式(10)可以看出, HiMAC 的加密和解密延迟的取决参数为: 纯文本消息大小 S_{d0}、网络维度 a、传输范围 r_0(来自 $E[H]$)、$T_{AES-Enc}$、$T_{AES-Dec}$、C_p, 以及 $\{TE_{RSA},TD_{RSA},TS_{RSA},TV_{RSA}\}$。

图 3 展示了在 0.1~100 千字节之间变化的 S_{d0} 对加密和解密延迟的影响。为与下一节中的模拟保持一致, 将 a 设置为 1000 m, r_0 设置为 100 m。对于

RSA, 我们使用等于 1024 位的密钥, 对于 AES 会话密钥大小使用等于 128 位的密钥。$T_{AES-Enc}$ 和 $T_{AES-Dec}$ 的值的计算方法与文献[26]中 128 位 AES 密钥的方法类似。另一方面, 计算了在模拟服务器中使用的 $C_p = 250\text{K MIPS}$。

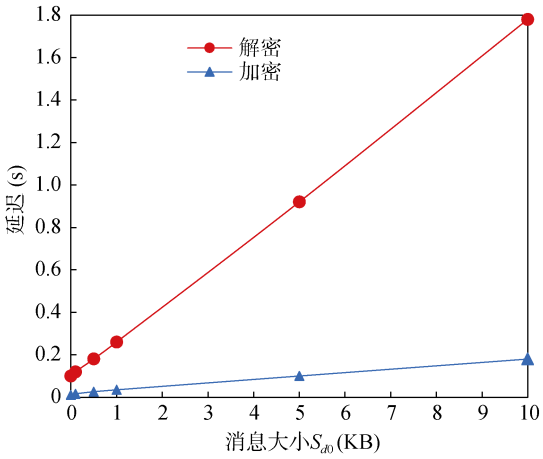


图 3 消息大小 S_{d0} 对加密和解密延迟的影响

Figure 3 Effect of message size on encryption and decryption delay

从图 3 可以看到, HiMAC 的加密和解密延迟随着纯文本消息 S_{d0} 大小的增加而增加。这与预期是一致的, 在这种情况下消息将被 AES 划分为更多的块, 并且每个块将分别加密/解密, 这将增加延迟。还注意到, 由于加密延迟被计算为源节点和中间节点的加密延迟的平均值, 因此 HiMAC 解密延迟比加密延迟大得多。另一方面, 解密延迟是在目标节点计算的, 包括对源节点和中间节点进行的所有加密的解密, 这将远远大于一个节点上的单个加密。此外, 当 S_{d0} 增加到 5 KB 以上时, 解密延迟开始严重增加。因此, 可以推断, 当纯文本消息的大小较大时, 最好将其划分为多个消息, 并分别加密/发送每个消息, 以避免在目标节点处出现较大的解密延迟。在下一节中, 将实验模拟的加密/解密延迟与式(9)和式(10)的分析延迟进行比较。

5 实验评估

5.1 模拟设置

使用网络模拟 NS2 2.35 软件[27]来测试 HiMAC 的性能。将 HiMAC 与文献[20]的 A-SAODV 进行比较, 两者都是在 AODV 中实现的, 并且都使用下面设置参数进行测试。

模拟网络的维度等于 $1000\times1000\ \text{m}^2$。假设无线带宽和传输范围分别为 6 Mbps 和 100 m。移动节点随机分布在地形中, 并遵循随机路线运动模型[27-28]。模拟了 6 种场景, 其中网络中的移动节点数分别设置为 50,100,200,300,400 和 500。对于两种方案, 每个方案重复 10 次, 并且对 10 次重复的读数进行平均以计算结果。网络中节点的移动性遵循 RWP 模型, 其中节点的最小速度 $V_{\min} = 0.01$ m/s、最大速度 $V_{\max} = 2.00$ m/s, 暂停时间设定为 100 s。每个场景持续 4000 s, 其中前 200 s 用于设置网络、安全要求(密钥交换)和路由表。在 200 s 之后, 网络中的每个节点使用测试的安全协议将数据分组发送到另一个随机节点。数据包(有效载荷)的大小设置为 10 KB, 两个数据包之间的时间间隔设置为 100 s。主要的模拟参数设置[29]如表 2 所示。

表 2 模拟参数

Table 2 Simulation parameters

参数项	参数值
模拟时间	4000 s
网络维度	$1000\times1000\ \text{m}^2$
无线带宽	6 Mbps
节点数	50～500
数据包请求速率(每个节点)	1 Packet/100 s
数据包大小	10 KB
节点传输范围	100 m
路由协议	HiMAC 增强型 AODV
节点移动模型	随机路径点(RWP)
节点移动速度	0.01～2.00 m/s
最大队列值	200

HiMAC 的加密和解密操作是使用 Crypto++库[30]中的 RSA 算法来实现的。用于加密、解密、签名和验证的公钥和私钥的长度设置为 1024 位。Crypto ++库已集成到 NS2 中, 其必要的类在 AODV 中使用。攻击模型模拟如下:

在每个场景中, 节点被随机划分为两组。第一组为恶意节点, 占节点总数 N 的 $m\%$, 它们在不应用安全机制的情况下监听数据包并转发它们。第二组受信任节点, 占节点总数 N 的 $(1-m\%)\%$。每当攻击者 N_A 侦听发送到目的地 D 的分组时, 它就将随机数据添加到分组, 递增跳数, 并将分组转发到可信节点之一。当 D 收到数据包时, 它将检测到数据包已被篡改, 因为解密算法将无法解密攻击者添加的数据。在模拟中, m 的值设定为 10%。

将从网络延迟、成功率 R_S、跳数 H_C、队列长度 L_Q、网络流量(Traff)等关键指标[31,32]对 HiMAC 进行实验评估，并与文献[20]A-SAODV 进行对比分析。

5.2 仿真结果

图 4 展示了 HiMAC 和 A-SAODV 的加密，解密和端到端延迟。

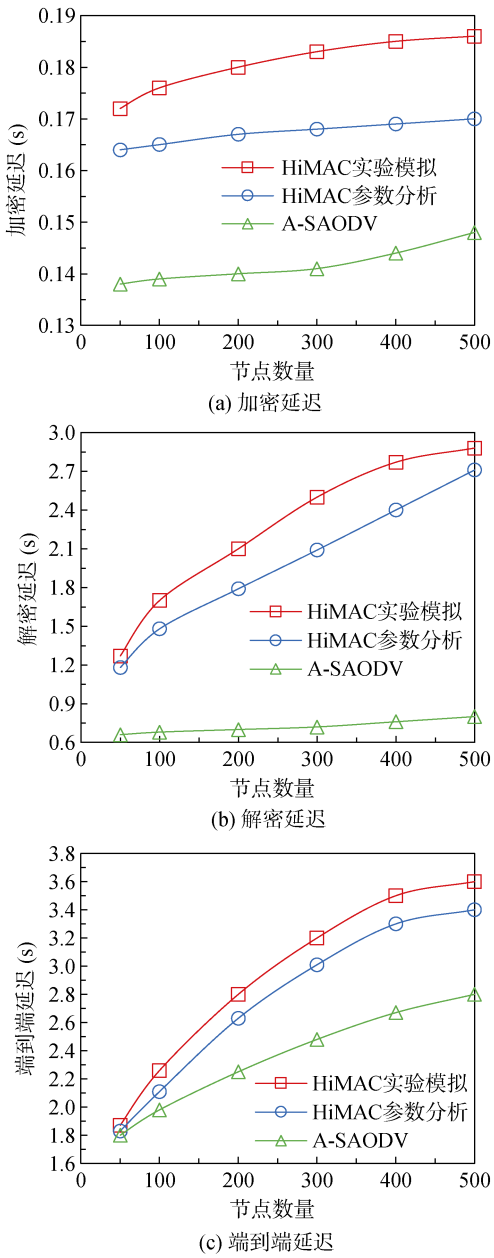


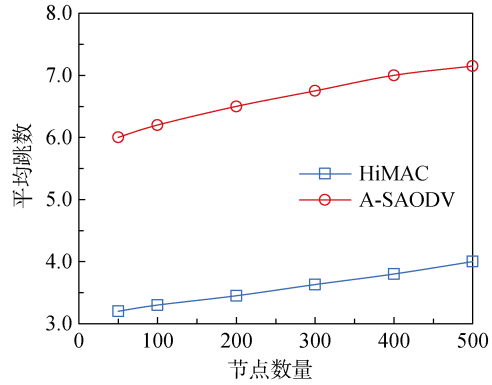
图 4 HiMAC 和 A-SAODV 的延迟对比

Figure 4 Delay comparison of HiMAC and A-SAODV

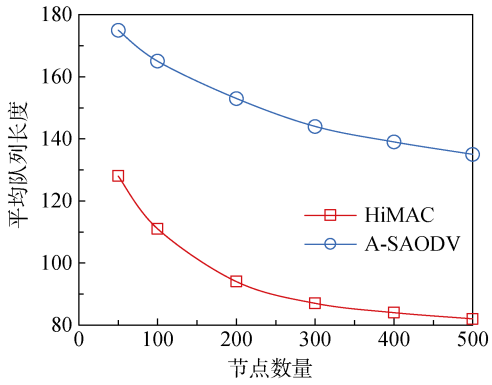
从图 4 中可以看到，HiMAC 的加密和解密延迟远远高于 A-SAODV。然而，HiMAC 的端到端延迟略高于 A-SAODV。首先，HiMAC 的加密延迟高于 A-SAODV，因为 HiMAC 涉及 (ID_S) 和 $\left(M_p \| MAC_{KPR_S}(M_p) \| T_S\right)$ 两个加密部分，而 A-SAODV 的加密延迟不包括第二部分。正如在第 3 节中所述，加密了遍历节点的列表，以防止攻击者在不被目标检测到的情况下篡改跃点计数。HiMAC 的解密延迟也远高于 A-SAODV(如图 4(b)所示)，因为 HiMAC 的解密过程包括解密操作的层次结构，而 A-SAODV 的解密延迟包括单个解密操作。HiMAC 中发生的加密和解密延迟的开销由快速通信延迟补偿，可以从图 4(c)中推断出。HiMAC 的端到端延迟仅略微大于 A-SAODV 的原因，尽管它在加密方面要慢得多，因为它需要更少的通信时间和开销。发生这种情况是因为 A-SAODV 不包括信任机制，其中“下一跳”决策除了路由效率之外还基于信任。此外，A-SAODV 中的自适应“双重签名”过程在建立安全路由之前和转发数据分组之前需要大量时间来获得目的地签名。此外，路由和数据包队列的分离要求数据包在转发之前等待路由建立，如果路由队列不堪重负则需要花费大量时间。通常，在 HiMAC 中通过消息路径上的每个节点都受信任的优点以及目标节点确保转发消息的每个中间节点的完整性来补偿 D_{EtE} 中的轻微开销。最后，我们从图 4 中注意到，从式(9)和式(10)计算的参数分析加密/解密延迟具有与模拟加密/解密延迟类似的值，这反映了分析方程在描述影响延迟的最重要参数时的准确性。

图 5 展示了 HiMAC 和 A-SAODV 的平均跳数和平均队列长度的对比情况。从图上可以看出，HiMAC 具有比 A-SAODV 更少的跳数和更短的分组队列长度。虽然 A-SAODV 依赖于安全的 RREQ 和 RREP 来建立安全路径，但是这需要很大的路由开销；HiMAC 使用第 3 节中描述的建议信任机制动态建立安全路由。基于“信任”值和普通 AODV 路由数据选择下一跳的过程，减少了建立安全路由所需的开销，并减少了到达目的地的“下一跳”数量。因为 HiMAC 算法的数据包在队列中等待的时间要短很多，这使得它找到最佳路线的概率更高，因此可以更快地到达目的地。

从图 5(a)可以看出，HiMAC 的平均跳数约为 3.5，而 A-SAODV 的平均跳数约为 6.6。此外，从图 5(b)可以看出，HiMAC 中的平均队列长度为 98 个数据包，而 A-SAODV 中的平均队列长度为 152 个数据包，这表明与 A-SAODV 相比，数据包在 HiMAC 中的队列停留时间更短。HiMAC 算法的数据包平均停留在队列中的时间较短，并在到达目的地的途中遍历较少的节点，因此，可以获得更高的成功率。



(a) 平均跳数



(b) 平均队列长度

图 5　HiMAC 和 A-SAODV 的平均跳数和平均队列长度对比

Figure 5　Comparison of average hops and average queue length between HiMAC and A-SAODV

图 6 展示了 HiMAC 和 A-SAODV 的平均成功率。当节点数等于 300 时, HiMAC 的成功率最高可达 96%, 平均值为 91%, 而 A-SAODV 的成功率平均为 79%, 最大值为 82%。当数据包因任何原因被丢弃时, 例如在队列中停留时间过长、队列已满、找不到到目的地的路由时, 它就不会到达目的地。其中, 找不到到目的地的路由是 HiMAC 和 A-SAODV 的共同的问题。因此, 导致成功率之间的差异, 主要是由队列中停留时间过长、队列已满两个原因引起的。A-SAODV 中的数据包队列平均比 HiMAC 中的数据包队列更满, 因此导致更多的数据包丢失。此外, 数据包将保留在 A-SAODV 队列中的时间比 HiMAC 长, 这将导致更多的数据包在队列 TTL 到期时被丢弃。这两个原因解释了 HiMAC 的更高成功率。

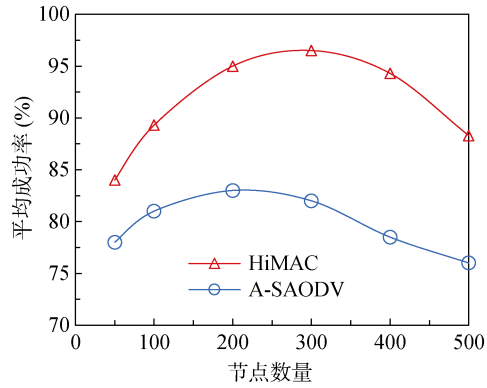


图 6　HiMAC 和 A-SAODV 的平均成功率对比

Figure 6　Comparison of average success rates of HiMAC and A-SAODV

图 7 展示了 HiMAC 和 A-SAODV 在所有节点上发送、转发和接收的数据包的平均流量。

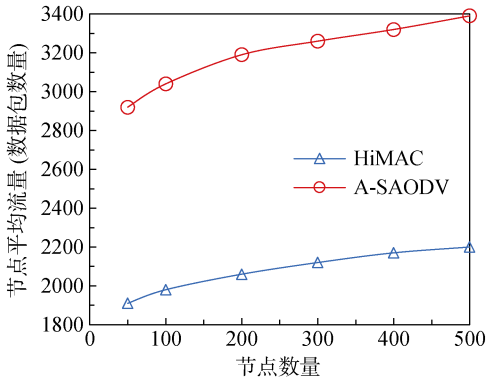


图 7　HiMAC 和 A-SAODV 的平均流量对比

Figure 7　Comparison of average traffic between HiMAC and A-SAODV

从图 7 可以看出, A-SAODV 的发送、转发和接收在网络中的产生的流量比 HiMAC 高很多。导致 A-SAODV 产生更多流量的主要差异是转发数据包的数量。这主要是因为在 A-SAODV 中, 数据包遍历的平均跳数比 HiMAC 高 2.5 倍, 因此转发的分组总数更高。此外, A-SAODV 中的成功率较低, 这导致在未能到达目的地之后重新发送更多数据分组, 这产生了额外数量的转发分组, 无形中消耗了更多流量。

综合上述实验, 提出的 HiMAC 虽然包含大量的加密操作, 但仍能产生非常好的性能。虽然由每个数据包的多个加密和解密操作引起的高处理, 给路由协议带来了额外的开销。但是由于 HiMAC 采用基于信任机制动态建立安全路由, 使得节点能够动态地选择路径上的下一个节点, 而不必始终保持安全路由, 减少了开销。因此, 使得 HiMAC 中的增减开销可以相互抵消, 达到平衡。此外, 通过验证目的地的每个中间节点, HiMAC 消除了路由路径上的中间节点的任何攻击的可能性。只要数据信息量不是很大, HiMAC 就能很好地运行。从分析和模拟结果可以推断出, HiMAC 的加密/解密和端到端延迟将随消息量增大而增加。因此, 需要注意的是, 在数据消息量无限增大时, HiMAC 提供的高安全性优势将无法弥补加密/解密操作增加的开销延迟。

6 结束语

该文提出了一种基于信任机制的 HiMAC 协议, 其中包含分层消息认证码解决方案用于保护移动 Ad-Hoc 网络中的数据传播。该协议依赖于根据用户的知识和专业知识以及节点的行为建立移动节点之间的信任。提出的 HiMAC 可以通过在每个中间节点对数据包进行签名和加密来防止攻击者篡改数据包或修改其跳数。该协议是基于邻居的“信任”动态建立路由, 并且还基于保护路由数据, 以防止恶意节点将错误数据插入到可信节点的路由表中。通过将 HiMAC 的性能与文献[20]中的 A-SAODV 安全机制进行数据分析和仿真测试。结果表明: HiMAC 的性能、安全性明显优于 A-SAODV。

在未来的工作中, 将研究如何在保持安全数据传输的同时减少 HiMAC 产生的加密开销。一个可能的方向是找到最佳密钥大小, 减少加密和解密延迟, 同时保持消息安全; 由于执行了分层加密操作, 因此不需要非常大的密钥, 因为攻击者需要几个解密阶段才能到达原始消息。

参考文献

[1] Chriki A, Touati H, Snoussi H, et al. FANET: Communication, Mobility Models and Security Issues[J]. *Computer Networks*, 2019, 163: 106877.

[2] Kavitha P R, Mukesh R. Detection of Impersonation Attack in MANET Using Polynomial Reduction Algorithm[J]. *International Journal of Network Security*, 2018, 20(2): 381-389.

[3] Jamal T, Butt S A. Malicious Node Analysis in MANETS[J]. *International Journal of Information Technology*, 2019, 11(4): 859-867.

[4] Thanuja R, Umamakeswari A. Black Hole Detection Using Evolutionary Algorithm for IDS/IPS in MANETs[J]. *Cluster Computing*, 2019, 22(2): 3131-3143.

[5] Regonda N, Santosh D, Patra K, et al. A state-of-the-Art: An Optimal Path Algorithm for Mobile Ad Hoc Network (Manet) and Wireless Ad Hoc Network (Wanet)[J]. *International Journal of Psychosocial Rehabilitation*, 2020, 24(6): 8638-8644.

[6] Gurung S, Chauhan S. A Novel Approach for Mitigating Route Request Flooding Attack in MANET[J]. *Wireless Networks*, 2018, 24(8): 2899-2914.

[7] Chintalapalli R M, Ananthula V R. M-LionWhale: Multi-Objective Optimisation Model for Secure Routing in Mobile Ad-Hoc Network[J]. *IET Communications*, 2018, 12(12): 1406-1415.

[8] Zhang Y T. An Algorithm of Cooperative Filtering for Privacy Protection Based on Substitution Encryption[J]. *Journal of Xinjiang University* (*Natural Science Edition*), 2017, 34(4): 446-451.
(张永棠. 基于代换加密的隐私保护协同过滤推荐算法[J]. *新疆大学学报(自然科学版)*, 2017, 34(4): 446-451.)

[9] Singh M, Mandal J K. Reliability of MANET under the Influence of Black Hole Attack in Adhoc on Demand Distance Vector Routing Protocol[J]. *Journal of Scientific & Industrial Research*, 2017, 76(7):423-426.

[10] Yin L H, Guo Y C, Zhang H B, et al. Threat-Based Declassification and Endorsement for Mobile Computing[J]. *Chinese Journal of Electronics*, 2019, 28(5): 1041-1052.

[11] Batista da Silveira T, Mendes Duque E, Ferzoli Guimarães S J, et al. Proposal of Fibonacci Heap in the Dijkstra Algorithm for Low-Power Ad-Hoc Mobile Transmissions[J]. *IEEE Latin America Transactions*, 2020, 18(03): 623-630.

[12] Tan Y W, Liu J J, Kato N. Blockchain-Based Key Management for Heterogeneous Flying Ad Hoc Network[J]. *IEEE Transactions on Industrial Informatics*, 2021, 17(11): 7629-7638.

[13] Ahmed A S, Kumaran T S, Syed S S A, et al. Cross-Layer Design Approach for Power Control in Mobile Ad Hoc Networks[J]. *Egyptian Informatics Journal*, 2015, 16(1): 1-7.

[14] Yadav A K, Das S K, Tripathi S. EFMMRP: Design of Efficient Fuzzy Based Multi-Constraint Multicast Routing Protocol for Wireless Ad-Hoc Network[J]. *Computer Networks*, 2017, 118: 15-23.

[15] Indirani G, Selvakumar K. A Swarm-Based Efficient Distributed Intrusion Detection System for Mobile Ad Hoc Networks (MANET)[J]. *International Journal of Parallel*, *Emergent and Distributed Systems*, 2014, 29(1): 90-103.

[16] Singh O, Singh D, Singh D R. SAODV: Statistical Ad Hoc On-Demand Distance Vector Routing Protocol for Preventing Mobile Adhoc Network Against Flooding Attack[J]. *Advances in computational sciences and technology*, 2017, 10(8):2457-2470.

[17] Wang H M, Zhang Y, Ng D W K, et al. Secure Routing with Power Optimization for Ad-Hoc Networks[J]. *IEEE Transactions on Communications*, 2018, 66(10): 4666-4679.

[18] Cao Y C, Zhou Y B. Multi-Channel Fusion Leakage Detection[J]. *Journal of Cyber Security*, 2020, 5(6): 40-52.
(曹雨晨, 周永彬. 多源融合信息泄漏检测方法[J]. *信息安全学报*, 2020, 5(6): 40-52.)

[19] Lim L B, Spendlove D J G, Guan L, et al. ADTH: Bounded Nodal Delay for Better Performance in Wireless Ad-Hoc Networks[J]. *Ad Hoc Networks*, 2019, 83: 25-40.

[20] Toor W T, Seo J B, Jin H. Distributed Transmission Control in Multichannel S-ALOHA for Ad-Hoc Networks[J]. *IEEE Communications Letters*, 2017, 21(9): 2093-2096.

[21] Zhao Y M, Xiao S, Gan H P, et al. A Constrained Coding-Aware Routing Scheme in Wireless Ad-Hoc Networks[J]. *Sensors* (*Basel*, *Switzerland*), 2019, 19(10): 2252.

[22] Sufian A, Banerjee A, Dutta P. Energy and Velocity Based Tree Multicast Routing in Mobile Ad-Hoc Networks[J]. *Wireless Personal Communications*, 2019, 107(4): 2191-2209.

[23] Omar M, Boufaghes H, Mammeri L, et al. Secure and Reliable Certificate Chains Recovery Protocol for Mobile Ad Hoc Networks[J]. *Journal of Network and Computer Applications*, 2016, 62: 153-162.

[24] Borkar G M, Mahajan A R. A Secure and Trust Based On-Demand Multipath Routing Scheme for Self-Organized Mobile Ad-Hoc Networks[J]. *Wireless Networks*, 2017, 23(8): 2455-2472.

[25] Vinayagam J, Balaswamy C, Soundararajan K. Certain Investigation on MANET Security with Routing and Blackhole Attacks Detection[J]. *Procedia Computer Science*, 2019, 165: 196-208.

[26] Bar-On A, Dunkelman O, Keller N, et al. Improved Key Recovery Attacks on Reduced-round AES with Practical Data and Memory Complexities[J]. *Journal of Cryptology*, 2020, 33(3): 1003-1043.

[27] Zhang Y T, Zhou F K, Wu S C, Wu Shengcai. Precision agriculture research of wireless sensor network node deployment[J], *Jiangsu Agricultural Sciences*, 2017,45(3):200-205.
(张永棠,周富肯,吴圣才.精确农业无线传感器网络节点部署研究[J].*江苏农业科学*,2017,45(3):200-205.)

[28] Elgazzar K, Oteafy S M A, Ibrahim W M, et al. A Resilient P2P Architecture for Mobile Resource Sharing[J]. *The Computer Journal*, 2014, 58(8): 1689-1700.

[29] Zhang Y T, Fan B. Non-SPF Routing Algorithm Based on Ordered Semi-Group Preference Algebra[J]. *The Journal of China Universities of Posts and Telecommunications*, 2017, 24(6): 14-23.

[30] Crypto++ Library. http://www.cryptopp.com/. 2017.

[31] Ngoc L T, Tu V T. AODVDC: An Improved Protocol Prevents Whirlwind Attacks in Mobile Ad Hoc Network[J]. *International Journal of Network Security*, 2019, 21(2): 333-341.

[32] Zhang Y T, Huang Z Y. An Improved Hierarchical Modulation Scheme for MBMS Systems[J]. *Telecommunication Engineering*, 2017, 57(7): 772-777.
(张永棠, 黄中友. 一种适用 MBMS 业务的改进型分层调制方案[J]. *电讯技术*, 2017, 57(7): 772-777.)

张永棠 于 2018 年在厦门大学信息与通信工程专业获得博士学位。现任广东东软学院教授, 南昌工程学院江西省协同感知与先进计算技术研究所硕士生导师。研究领域为 5G 通信与网络安全。研究兴趣包括: 光通信、可信计算、网络体系结构。
Email:gov211@163.com