doi:10.3969/j.issn.1000-436x.2013.04.010

对一种混合结构洋葱路由方案的密码学分析

李龙海，付少锋，苏锐丹，车向泉
（西安电子科技大学 计算机学院，陕西 西安 710071）

摘 要：对时金桥等提出的混合结构洋葱路由方案进行了分析，发现存在的安全漏洞。第一个漏洞来源于其密码学报文结构的可展性。攻击者能够利用该漏洞改变洋葱消息的路由或在其中嵌入标签以追踪消息路由。另一个漏洞表现在匿名转发服务器容易遭受选择密文攻击。展示了 3 种不同的能够以较低代价破坏发送者和接收者不可关联性的攻击过程。为了避免所提到的攻击，提出了能够利用反向调查捕获恶意节点的修正方案。
关键词：匿名通信；洋葱路由；通用重加密；混合结构
中图分类号：TP393.08 **文献标识码：**A **文章编号：**1000-436X(2013)04-0088-11

Cryptanalysis of a hybrid-structured onion routing scheme

LI Long-hai, FU Shao-feng, SU Rui-dan, CHE Xiang-quan
(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: SHI Jin-qiao *et al*'s hybrid-structured onion routing scheme was analysed and some security flaws were found in their design. The first flaw was derived from the malleability of its cryptographic message format which could be exploited by attackers to redirect an onion message or embed tags into it for tracing its routing path. The second flaw was the vulnerability of relay servers to chosen ciphertext attack. Three different attacks were presented that each broke the sender-receiver unlinkability entirely at a relatively low co To evade these attacks, a modified scheme was also proposed which could capture malicious nodes by using upstream investigation.
Key words: anonymous communication; onion routing; universal re-encryption; hybrid structure

1 引言

随着 Internet 应用范围的不断扩大和信息量的爆炸性增长，对用户隐私性的保护已经成为一个至关重要的问题。在很多应用中，隐私不仅意味着通信内容的秘密性，而且意味着不能泄漏“谁和谁”、“在什么时间”、“通信量大小”等通信上下文信息。匿名通信技术主要研究如何隐藏消息发送者和接收者的身份以及通信双方的对应关系，即如何实现发送者匿名性、接收者匿名性和不可关联性。该技术被广泛应用在匿名电子邮件、匿名 Web 浏览、电子商务、电子选举等对隐私保护要求较高的 Internet 应用系统中。

匿名通信技术的研究可以追溯到 1981 年 Chaum 发表的开创性论文[1]，其中，提出了混合网络(Mix-net)的概念。目前已提出的大多数匿名通信系统都是基于 Mix-net 思想设计的。一个 Mix-net 包含若干个 Mix 服务器。发送者的消息首先被多层加密形成类似于“洋葱”的结构，然后经过多个 Mix 服务器的转发传递给接收者。每个 Mix 服务器对收到的报文进行单层解密并转发给下一个 Mix 服务器。接收者最终获得消息的明文。根据转发路径构成方式的不同，Mix-net 可以分为固定路由和自由路由 2 种。自由路由式 Mix-net 中的发送者随机选取若干个 Mix 服务器构成转发路径，并将这些服务器的网络地址依次封装在洋葱报文的各个层中，

收稿日期：2012-01-13；**修回日期：**2012-05-28
基金项目：国家自然科学基金资助项目(61101142)；中央高校基本科研基金资助项目(K50510030012)
Foundation Items: The National Natural Science Foundation of China (61101142); The Fundamental Research Funds of the Central Universities(K50510030012)

最内层是接收者的地址和消息明文。发送者首先将报文发送给第一个 Mix 服务器。该服务器对收到的报文进行单层解密之后即可获得第二个 Mix 服务器的网络地址，并继续将解密后的报文发送给下一个服务器。依此类推，直至最终的接收者。该转发过程类似于剥洋葱的过程，因此也被称为“洋葱路由”系统。其中，每个转发节点仅知道自己上一跳和下一跳的地址。只要转发路径中有一个以上的节点是诚实的(不泄露自己上一跳和下一跳的地址)，攻击者就无法将所截获消息的发送者和接收者关联到一起。

重放攻击是一种针对洋葱路由系统的十分有效的攻击。攻击者将截获的洋葱报文复制多次发送给目标 Mix 服务器，则在该服务器的某个输出链路上必然出现多个内容重复的报文。攻击者由此可追踪该报文的流动方向，直至接收者。早期的洋葱路由系统一般采用检查重复输入并丢弃的方法对抗重放攻击[2]，其缺点是需要缓存大量的历史数据，并且每个输入都要与缓存数据比对，造成报文处理时间的逐渐增长。另外一些系统采用了定期更换密钥[3]、加入时间戳[4]等方法，但分别存在密钥管理复杂、需要严格时间同步等问题。2004 年 Gomulkiewicz 等提出了一种新型的基于通用重加密(URE, universal re-encryption)算法[5]的洋葱路由方案 URE-Onion[6]。URE-Onion 服务器对收到的报文除了进行部分解密，还要实施随机化重加密。该方法能够保证重复的报文输入同一 Mix 服务器之后输出也是不同的，使得攻击者无法继续追踪下去，因此不再需要缓存历史记录。但不久 Danezis 等提出了一种针对 URE-Onion 的绕路攻击(detour attack)[7]表明该方案存在严重安全漏洞。随后 Klonowski 等对 URE-Onion 进行了改进[8]，利用为每个服务器分配双重密钥的方法阻止攻击者随意修改转发路径，使其无法实施绕路攻击。然而，Borisov 等在文献[9, 10]中又给出了针对 URE-Onion 改进方案的多种攻击，并分别提出了基于上下文敏感加密(context sensitive encryption)和基于标签加密(tag-based encryption)的 2 种改进方法。陆天波等也指出了基本的通用重加密算法对于选择密文攻击的脆弱性，并设计了基于 ElGamal 算法重加密特性的匿名系统 WGRe[11]。

URE-Onion 及其多种改进方案虽然能够抵御重放攻击，但由于完全基于非对称加密算法，效率都很低。如果匿名转发路径长度为 $?$，则每个服务器收到报文之后都要进行 $O(?)$次大素数群上的指数运算，并且长度为 $O(?)$的报文中只有 $1/?$被真正用于加密用户消息，而其他的部分都被用来加密路由信息。针对上述缺点，时金桥等[12]设计了一种混合结构洋葱报文机制 HS-Onion。该设计仅用 URE 算法加密路由信息部分，而消息正文部分采用更高效的对称加密算法加密，因此在传送长消息时具有很大的优势。该方案的另一个创新点是采用了 2 种代数结构不同的公钥加密算法对路由信息部分进行双层加密，以实现对报文的完整性保护进而防止攻击者修改路由并实施绕路攻击。

本文发现时金桥等提出的洋葱路由方案[12]存在的安全漏洞，主要表现在报文结构具有可展性(malleability)，攻击者可以改变洋葱消息的路由或在其中嵌入标签以追踪消息路由。由于没有对恶意行为的调查机制，Mix 服务器很容易被攻击者用作解密预言机(decryption oracle)以获取有利信息。本文展示了基于这些安全漏洞的 3 种不同的攻击方法，这些攻击都能够将消息的发送者和接收者以不可忽略的概率关联到一起。最后本文给出了针对原方案的修正方法，并对修正后系统的安全性和效率进行了讨论。

2　HS-Onion 方案回顾

2.1　通用重加密算法

ElGamal 算法是一种具有同态特性的概率性公钥加密算法。在已知接收者公钥的条件下任何人都可以对 ElGamal 密文进行重加密(re-encryption)，重加密得到的密文与原密文对应相同的明文。在不知道私钥的情况下，攻击者能够将重加密前后的 2 个密文匹配到一起的难度等同于解 Diffie-Hellman Decision 问题。

通用重加密算法(URE)[5]是对 ElGamal 算法的改进。在不知道接收者公钥的情况下，URE 密文也可以被重加密。该算法的详细描述如下。

系统建立　构造 q 阶循环群 G，满足 q 为素数，且在群 G 上离散对数及 Diffie-Hellman Decision 问题难解。随机选取 G 的一个生成元 g。最后将 G 和 g 作为公共参数向所有用户公布。

密钥生成　Alice 任取 $x \in Z_q$ 作为其私钥，计算 $y = g^x$ 作为其公钥。

加密　设明文为 m，为构造针对 Alice 的密文，

Bob 任取 $k_0, k_1 \in Z_q$，然后计算四元组$(a_0, ß_0, a_1, ß_1) = (my^{k_0}, g^{k_0}, y^{k_1}, g^{k_1})$作为输出。

解密 Alice 收到密文$(a_0, ß_0, a_1, ß_1)$之后，计算 $m_0 = a_0 / ß_0^x$ 和 $m_1 = a_1 / ß_1^x$。如果 m_1=1，则输出明文 m=m_0，否则输出⊥。

再加密 设输入密文为$(a_0, ß_0, a_1, ß_1)$，任取 $k_0', k_1' \in Z_q$，计算并输出$(a_0', ß_0', a_1', ß_1') = (a_0 a_1^{k_0'}, ß_0 ß_1^{k_0'}, a_1^{k_1'}, ß_1^{k_1'})$。再加密过程能够保证输入与输出密文对应同一明文。

下文中用符号$\mathrm{URE}_x(m)$表示关于消息 m 的URE密文，解密密钥为 x。由于 URE 是概率性加密算法，所以$\mathrm{URE}_x(m)$实际表示对应 m 的密文集合中的某一个成员。

可以用多个用户的公钥对同一消息 m 进行多层 URE 加密。令 x_i 表示用户 S_i 的私钥，y_i 为对应的公钥，则$\mathrm{URE}_{x_1+x_2+\perp+x_?}(m)$表示密文。

$$(a_0, ß_0, a_1, ß_1) = (m(y_1 y_2 \perp y_?)^{k_0}, g^{k_0}, (y_1 y_2 \perp y_?)^{k_1}, g^{k_1}) = (m g^{k_0 \sum_{i=1}^{?} x_i}, g^{k_0}, g^{k_1 \sum_{i=1}^{?} x_i}, g^{k_1})$$

。

该密文需要用户 $S_1, S_2, \cdots, S_?$ 利用它们的私钥 $x_1, x_2, \cdots, x_?$ 依次进行部分解密才能获得明文 m。例如 S_1 收到密文$\mathrm{URE}_{x_1+x_2+\perp+x_?}(m) = (a_0, ß_0, a_1, ß_1)$后按如下方式进行部分解密。

$$(a_0', ß_0', a_1', ß_1') = \left(\frac{a_0}{ß_0^{x_1}}, ß_0, \frac{a_1}{ß_1^{x_1}}, ß_1 \right)$$

显然，经过 S_1 部分解密后得到的密文可表示为$\mathrm{URE}_{x_2+x_3+\perp+x_?}(m)$。

2.2 URE-Onion 方案

基于 URE 的洋葱路由方案[6,8]主要利用了 URE 的 2 个特性：1)在不知道接收者公钥的情况下也可以对 URE 密文进行重加密；2)可利用多个接收者的公钥对同一消息 m 进行多层加密，每个接收者可以对密文进行部分解密以去除相应的加密层(类似于剥洋葱)。这些方案的基本思想为：设接收者为 $S_{?+1}$，匿名消息发送者首先构造一条到达目的主机的匿名路径 $S_1, S_2, \cdots, S_{?+1}$，其中，$S_1, S_2, \cdots, S_?$ 为中间 Mix 服务器。发送者构造如下格式的洋葱报文：$\{\mathrm{URE}_{x_1}(S_2), \mathrm{URE}_{x_1+x_2}(S_3), \cdots, \mathrm{URE}_{x_1+\perp+x_?}(S_{?+1}), \mathrm{URE}_{x_1+\perp+x_?+x_{?+1}}(m)\}$，该报文由 $?+1$ 个 URE 密文块组成，前 $?$ 块保存了路由信息，最后一块保存了消息 m。发送者将这 $?+1$ 个密文块的顺序随机打乱，然后发送给 S_1。匿名路径上第 $j(1 \le j \le ?)$个服务器 S_j 收到报文后进行如下操作。

1) 部分解密。利用自己的私钥 x_j 将报文中每一个密文块进行部分解密，正常情况下必有一个密文块解密结果为可识别的 Mix 服务器地址 S_{j+1}，即 S_j 的下一跳路由地址。该密文块在下一步被重加密之前被 S_j 替换成随机数。

2) 重加密。S_j 生成新的随机因子并对报文中的每个密文块进行重加密，最后将重加密结果发送给 S_{j+1}。

以为 S_1 例，S_1 对输入报文解密后获得下一跳路由地址 S_2，再进行随机数替换和重加密操作获得输出报文：$\{\mathrm{random}, \mathrm{URE}_{x_2}(S_3), \cdots, \mathrm{URE}_{x_2+\perp+x_?}(S_{?+1}), \mathrm{URE}_{x_2+\perp+x_?+x_{?+1}}(m)\}$。该报文经过 $S_2, S_3, \cdots, S_?$ 的处理和转发最后到达接收者 $S_{?+1}$ 时具有如下格式：$\{\mathrm{random}, \mathrm{random}, \cdots, \mathrm{random}, \mathrm{URE}_{x_{?+1}}(m)\}$。$S_{?+1}$ 利用私钥 $x_{?+1}$ 将各个密文块解密，只有最后一块解密结果为消息 m，其他为⊥。

在上述 URE-Onion 方案中，每个中间 Mix 服务器仅知道自己的上一跳和下一跳的地址，只要转发路径中有一个以上的节点是诚实的，就可以实现发送者和接收者之间的不可关联性。由于 S_j 每次都生成新的随机因子对报文进行 URE 重加密，之后再输出，因此相同的报文输入同一服务器之后的输出也是不同的。这是该方案能抵御重放攻击的关键所在。

2.3 HS-Onion 方案

2.3.1 符号与假设

由于 URE-Onion 方案存在效率和安全性问题，文献[12]提出了基于混合结构报文的改进方案 HS-Onion。其主要创新点是在报文格式设计上引入对称加密机制，提高了报文处理效率，并且在不同密文块之间引入链式加密，使得攻击者无法随意篡改密文块，因此避免了针对原方案的绕路攻击[7]。现将该方案所定义的相关符号及攻击者模型总结如下。

相关符号 HS-Onion 系统由 N 个 Mix 服务器构成，分别用符号 $S_0, S_1, \cdots, S_{N-1}$ 表示。每台服务器既充当用户，又为其他节点提供匿名转发服务。任意服务器 S_i 生成 2 对密钥—(y_i, x_i)和(y_i', x_i')，并发布公钥部分。其中，$y_i = g^{x_i}$ 为 URE 公钥，y_i' 为另一种具有 CCA 安全性的非对称加密算法 E 的公钥。$E_{x'}(m)$ 表示利用该非对称加密算法对消息 m 加密

所生成的密文，解密密钥为 x'。H 表示一个伪随机数生成器，以输入参数 r 为种子生成伪随机数 $H(r)$。$e_k(m)$表示利用对称加密算法对 m 加密所生成的密文，解密密钥为 k。tag 和 tag′ 为 2 个约定好的标记字符串，如果 Mix 服务器对某个密文块的解密结果以 tag 或 tag′ 作为前缀，则表明该密文块解密成功。

攻击者模型　HS-Onion 方案假设攻击者能够监听所有网络链路并且能够控制部分 Mix 服务器，即采用了全局攻击者模型。

2.3.2　HS-Onion 机制描述

设匿名消息发送者为 S_0，接收者为 $S_{?+1}$。发送者从 Mix 服务器集合中任选 $?$ 个构成匿名转发路径中的节点，不妨设这些节点为 $S_1, S_2, \cdots, S_?$。

洋葱报文结构　为匿名发送消息 m，S_0 构造一个包含 $?+3$ 个密文块的洋葱报文 P。为方便表示，定义函数 $K(s,t)=\sum_{i=s}^{t}x_i+\sum_{i=s}^{t-1}R_i$ $(1\le s<t\le ?+1)$，其中，$R_1, R_2, \cdots, R_?$ 为 S_0 选取的长度合适的随机数。当 $s=t$ 时，定义 $K(s,t)=x_t$。报文 P 的具体格式如下：

$$\begin{aligned}P=\{&\mathrm{URE}_{x_1}(\mathrm{tag}\,\|\,E_{x_1'}(S_2\,\|\,R_1)),\\&\mathrm{URE}_{x_1+x_2+R_1}(\mathrm{tag}\,\|\,E_{x_2'}(S_3\,\|\,R_2)),\cdots,\\&\mathrm{URE}_{K(1,?-1)}(\mathrm{tag}\,\|\,E_{x_{?-1}'}(S_?\,\|\,R_{?-1})),\\&\mathrm{URE}_{K(1,?)}(\mathrm{tag}\,\|\,E_{x_?'}(S_{?+1}\,\|\,R_?)),\\&\mathrm{URE}_{K(1,?+1)}(\mathrm{tag}\,\|\,E_{x_{?+1}'}(k)),\\&\mathrm{URE}_{K(1,?+1)}(1),e_k(m)\}\end{aligned}\tag{1}$$

该报文中前 $?$ 块用于加密路由信息，第 $?+1$ 块包含了对称加密密钥 k，第 $?+3$ 块包含了用密钥 k 加密的消息 m。第 $?+2$ 块的作用见下面“路由协议”的 5)。

路由协议　S_0 将上述报文的前 $?+1$ 块的顺序打乱，然后发送给 S_1。匿名路径上第 $j(1\le j\le ?)$个服务器 S_j 收到报文后进行如下操作。

1) 利用私钥 x_j 对前 $?+2$ 块密文进行部分解密，正常情况下会从其中的一个密文块中得到 $\mathrm{tag}\,\|\,E_{x_j'}(S_{j+1}\,\|\,R_j)$。称该块为属于服务器 S_j 的密文块 B_j。

2) 利用私钥 x_j' 解密 $E_{x_j'}(S_{j+1}\,\|\,R_j)$ 获得下一跳路由地址 S_{j+1} 及随机数 R_j。

3) 用 R_j 作为解密密钥对除 B_j 之外的前 $?+2$ 块密文进行部分解密。

4) 随机生成种子 r_j，计算伪随机串 $H(r_j)$并与第 $?+3$ 块进行异或，因此，最后一个密文块等于 $e_k(m)\oplus H(r_1)\oplus\cdots\oplus H(r_j)$。

5) 利用第 $?+2$ 块密文 $\mathrm{URE}_{K(j+1,?+1)}(1)$ 生成 $\mathrm{URE}_{K(j+1,?+1)}(\mathrm{tag}'\,\|\,r_j)$，并用生成结果替换块 B_j。

6) 将前 $?+2$ 块密文进行重加密。

7) 将前 $?+1$ 个密文块的顺序随机打乱，最后将报文发送给下一跳 Mix 服务器 S_{j+1}。

接收者的处理　$S_{?+1}$ 收到报文之后用私钥 $x_{?+1}$ 对前 $?+1$ 块密文进行解密，正常情况下会从其中的一个块中得到 $\mathrm{tag}\,\|\,E_{x_{?+1}'}(k)$，从其他块中得到 $?$ 个随机种子 $r_1, r_2, \cdots, r_?$。$S_{?+1}$ 用 x_j' 解密 $E_{x_{?+1}'}(k)$ 获得对称密钥 k。利用 k 及 $r_1, r_2, \cdots, r_?$，$S_{?+1}$ 很容易对最后一个密文块 $e_k(m)\oplus H(r_1)\oplus H(r_2)\oplus\cdots\oplus H(r_?)$ 进行解密获得消息 m。

3　对 HS-Onion 的攻击

本节将首先描述 HS-Onion 方案的 2 个主要安全漏洞，然后展示基于这些漏洞的 4 种攻击方法。其中，解密预言机攻击用作其他攻击的基本手段，而另外 3 种攻击的目标是破坏 HS-Onion 的基本安全属性——通信双方的不可关联性，即以不可忽略的概率将所截获报文的发送者和接收者匹配到一起。

3.1　HS-Onion 的安全漏洞

下面所展示的攻击主要利用了 HS-Onion 方案的 2 个安全性漏洞。

1) 报文中包含路由信息的 URE 密文块具有可展性。攻击者在不知道明文 m 和私钥 x 的情况下，可以利用 $\mathrm{URE}_x(m)=(a_0,ß_0,a_1,ß_1)=(mg^{xk_0}, g^{k_0}, g^{xk_1}, g^{k_1})$ 生成关于消息 m' 的有效密文，即令 $\mathrm{URE}_x(m')=(m'a_1^{k_0'}, ß_1^{k_0'}, a_1^{k_1'}, ß_1^{k_1'})$。实际上，2.3.2 节 HS-Onion 路由协议的 5)也利用了该特性。同理，攻击者还可以由 $\mathrm{URE}_x(m)=(a_0,ß_0,a_1,ß_1)$ 生成密文 $\mathrm{URE}_x(Rm)=(Ra_0,ß_0,a_1,ß_1)$，其中，$R$ 为攻击者者选定的值。攻击者可以利用该特性在报文中嵌入标签以追踪报文的路由。

2) HS-Onion 的 Mix 服务器对输入报文进行解密之前没有对其效性做任何验证，并且解密结束后发现非法报文也没有相应的反向追踪机制，因此，诚实服务器很容易被攻击者用作解密预言机以获取有利信息。

3.2 解密预言机

假设恶意服务器 S_a 截获到的报文中含有密文 $\mathrm{URE}_{x_b}(m')$，则 S_a 可以通过如下攻击方法把诚实服务器 S_b 用作解密预言机，将 $\mathrm{URE}_{x_b}(m')$ 解密获得 m'。

Step1 S_a 任取随机数 R'，并用 $y'=g^{x_a R'}$ 作为公钥对 $\mathrm{URE}_{x_b}(m')$ 进行加密，即将其转化为密文 $\mathrm{URE}_{x_a+x_b+R'}(m')$，其中，$x_a$ 为 S_a 的私钥。

Step2 S_a 任取随机数 $R'_1, R'_2, \cdots, R'_?$，并构造包含 $?+3$ 个块的报文 $\{\mathrm{URE}_{x_b}(\mathrm{tag}\,\|\,E_{x'_b}(S_a\,\|\,R')), \mathrm{URE}_{x_a+x_b+R'}(m'), \mathrm{URE}_{x_a+x_b+R'}(R'_1), \mathrm{URE}_{x_a+x_b+R'}(R'_2), \cdots, \mathrm{URE}_{x_a+x_b+R'}(R'_{?-1}), \mathrm{URE}_{x_a+x_b+R'}(1), R'_?\}$，最后，$S_a$ 将该报文发送给 S_b。

Step3 根据 HS-Onion 路由协议，S_b 接收到该报文之后首先用私钥 x_b 对其进行解密，则必然从第一块中得到 $E_{x'_b}(S_a\,\|\,R')$，再用 x'_b 解密后得到下一跳路由地址 S_a 和随机数 R'。之后 S_b 按照 2.3.2 节协议规定步骤对该报文进行处理，并将处理后的报文重新发回给 S_a。当然，为了避免 S_b 的怀疑也可以通过修改报文下一跳地址令 S_b 将报文发往 S_a 的同谋者。

Step4 上述报文回到服务器 S_a 时具有如下格式：$\{\mathrm{URE}_{x_a}(\mathrm{tag}'\,\|\,r_b), \mathrm{URE}_{x_a}(m'), \mathrm{URE}_{x_a}(R'_1), \mathrm{URE}_{x_a}(R'_2), \cdots, \mathrm{URE}_{x_a}(R'_{?-1}), \mathrm{URE}_{x_a}(1), R'_? \oplus H(r_b)\}$（前 $?+1$ 个密文块的顺序可能是被打乱的）。S_a 利用私钥 x_a 对前 $?+1$ 块密文进行解密可得到 $\{\mathrm{tag}'\,\|\,r_b, m', R'_1, R'_2, \cdots, R'_{?-1}\}$。即便这些解密结果的顺序是随机的，$S_a$ 也能从中识别出 m'，因为 $\mathrm{tag}', R'_1, R'_2, \cdots, R'_{?-1}$ 都是 S_a 已知的值。

3.3 数据指纹追踪攻击

3.3.1 前提条件

实施“数据指纹追踪攻击”要求攻击者能够监听系统的所有链路，并且控制了匿名转发路径中的最后一个节点。HS-Onion 定义的全局攻击者模型完全可以满足这些条件。

另外，对于基于 Internet 的匿名通信系统，要求攻击者能监听系统所有链路是很难满足的，因此，一般采用更实际的局部攻击者模型，即假定攻击者能够控制系统的部分服务器，并且只能够监听恶意服务器自己的输入和输出链路。如果采用局部攻击者模型，“数据指纹追踪攻击”则要求攻击者能够控制匿名转发路径中的第一个和最后一个节点。

3.3.2 攻击过程

该攻击的基本思想是攻击者记录每个发送者输出报文的数据指纹(因为攻击者可以监听网络的所有链路)，然后由匿名转发路径中的最后一个节点利用协议漏洞提取每个转发报文的指纹，如果发现与此前记录的某个指纹相等，那么就能将发送者和接收者匹配到一起。

设发送者为 S_0，接收者为 $S_{?+1}$，匿名转发路径由节点为 $S_1, S_2, \cdots, S_?$ 构成且 $S_?$ 为恶意服务器。S_0 构造关于消息 m 的报文 P，其格式如式(1)所示，然后发送给 S_1。攻击者利用如下的攻击过程将发送者 S_0、接收者 $S_{?+1}$ 和报文 P 关联到一起。

Step1 攻击者计算报文 P 的最后一个密文块 $e_k(m)$的数据指纹，即散列值 $\mathrm{SHA1}(e_k(m))$，并记录二元组$(S_0, \mathrm{SHA1}(e_k(m)))$。

上述记录报文指纹的工作也可由恶意节点 S_1 完成。值得注意的是，S_1 此时无法确定自己是否为第一个转发节点，因为其前驱 S_0 既可能是源节点也可能是中间转发节点。直到攻击完成时，尾节点 $S_?$ 获得路径相关信息才能验证 S_1 是否为首节点。

Step2 攻击者控制的恶意服务器 $S_?$ 每接收到一个报文 P'，首先按照原协议规定对其解密获得下一跳服务器地址 $S_{?+1}$，然后判断自己是否为匿名路径的最后一个转发节点，具体采取的方法是将 $S_{?+1}$ 用作解密预言机将报文中第 $?+2$ 块密文进行解密。如果解密结果为 1，则证明自己为路径尾节点。

下面解释这样做的原因。根据 HS-Onion 路由协议，如果 $S_?$ 是关于报文 P' 的最后一个转发节点，则 P' 经过 $S_?$ 处理之后必然具有如下形式：

$$\{\mathrm{URE}_{x_{?+1}}(\mathrm{tag}'\,\|\,r_1), \mathrm{URE}_{x_{?+1}}(\mathrm{tag}'\,\|\,r_2), \cdots, \mathrm{URE}_{x_{?+1}}(\mathrm{tag}'\,\|\,r_?), \mathrm{URE}_{x_{?+1}}(\mathrm{tag}\,\|\,E_{x'_{?+1}}(k)), \mathrm{URE}_{x_{?+1}}(1), e_k(m)\oplus H(r_1)\oplus\cdots\oplus H(r_?)\} \quad (2)$$

因此，如果第 $?+2$ 块密文 $\mathrm{URE}_{x_{?+1}}(1)$ 解密为 1，则 $S_?$ 必为最后一个转发节点。

Step3 如果发现自己是最后一个节点，则 $S_?$ 利用 $S_{?+1}$ 的解密预言机服务将式(2)所示报文的前 $?+1$ 个密文解密，获得 $\mathrm{tag}'\,\|\,r_1, \mathrm{tag}'\,\|\,r_2, \cdots, \mathrm{tag}'\,\|\,r_?$。由随机种子 $r_1, r_2, \cdots, r_?$ 和密文 $e_k(m)\oplus H(r_1)\oplus\cdots\oplus H(r_?)$，$S_?$ 很容易计算出 $e_k(m)$及指纹 $\mathrm{SHA1}(e_k(m))$。

Step4 $S_?$ 从 Step1 中记录的二元组集合中找到$(S_0, \mathrm{SHA1}(e_k(m)))$就可以将报文 P、P' 以及发送者

S_0、接收者 $S_{?+1}$ 关联到一起。$S_?$ 在 Step3 中得到了 $?$ 个 tag′ 的事实也说明 S_0 与 $S_?$ 之间的路径长度为 $?$，并且 S_1 确实为第一个转发节点。

3.3.3　分析

设构成系统的 N 个 Mix 服务器中被攻击者控制的恶意服务器所占百分比为 d，并且构成转发路径的节点是发送者从全体服务器集合中随机选取的，那么在全局攻击者模型下，上述攻击能够成功匹配发送者和接收者的概率为 d。在局部攻击者模型下，该攻击要求转发路径的首节点和尾节点都是恶意的，因此，攻击成功的概率为 d^2。

3.4　标签追踪攻击

3.4.1　前提条件

实施“标签追踪攻击”要求攻击者能够控制匿名转发路径的第一个节点和最后一个节点。该攻击在全局攻击者模型和局部攻击者模型下都适用。

3.4.2　攻击过程

标签追踪攻击的基本思想是由处于匿名转发路径上游的恶意节点 S_a 利用 URE 算法的可展性在报文中嵌入“标签”。处于下游的另一恶意节点 S_b 如果在收到的报文中发现了相同的标签则可以判断自己与 S_a 处于同一转发路径。如果 S_a、S_b 恰为该路径的首尾节点，则该攻击成功地将发送者和接收者匹配到一起。

设发送者为 S_0，接收者为 $S_{?+1}$，匿名转发路径由 $S_1, S_2, \cdots, S_?$ 构成，且 S_1、$S_?$ 为恶意服务器。S_0 将消息 m 包装成洋葱报文 P，然后发送给 S_1，P 的格式如式(1)所示。基于上述假设，标签追踪攻击的具体攻击过程如下。

Step1　S_1 接收到报文 P 后，首先按原协议规定对其进行解密和再加密，获得下一跳路由地址 S_2 及如下格式的报文 P'：

$$\begin{aligned}&\{\mathrm{URE}_{K(2,?+1)}(\mathrm{tag}'\,\|\,r_1),\\&\mathrm{URE}_{x_2}(\mathrm{tag}\,\|\,E_{x_2'}(S_3\,\|\,R_2)),\\&\mathrm{URE}_{K(2,3)}(\mathrm{tag}\,\|\,E_{x_3'}(S_4\,\|\,R_3)),\ \cdots,\\&\mathrm{URE}_{K(2,?)}(\mathrm{tag}\,\|\,E_{x_?'}(S_{?+1}\,\|\,R_?)),\\&\mathrm{URE}_{K(2,?+1)}(\mathrm{tag}\,\|\,E_{x_{?+1}'}(k)),\\&\mathrm{URE}_{K(2,?+1)}(1), e_k(m)\oplus H(r_1)\}\end{aligned}\tag{3}$$

其中，前 $?+1$ 块密文的顺序是随机的。S_1 从 P' 的前 $?+1$ 个密文中（$\mathrm{URE}_{K(2,?+1)}(\mathrm{tag}'\,\|\,r_1)$ 除外）任取一块，不妨设该块为 $\mathrm{URE}_{K(2,j)}(\mathrm{tag}\,\|\,E_{x_j'}(S_{j+1}\,\|\,R_j))$，并在其中嵌入能唯一标识 S_1 的标签 $?_1\in G$，即将其替换为 $\mathrm{URE}_{K(2,j)}(?_1(\mathrm{tag}\,\|\,E_{x_j'}(S_{j+1}\,\|\,R_j)))$。之后，$S_1$ 将加入标签的报文发送给 S_2。

Step2　恶意服务器 $S_?$ 每接收到一个报文 P''，首先用私钥 $x_?$ 进行解密，如果在前 $?+1$ 个密文中发现某个解密结果具有形式 $?_1(\mathrm{tag}\,\|\,E_{x_?'}(S_{?+1}\,\|\,R_?))$，则可以认定自己与上游的 S_1 处于同一转发路径。

在上述攻击中，虽然 S_1、$S_?$ 还需借助其他手段确定自己是否为首尾节点，但处于 S_1 与 $S_?$ 之间的其他 Mix 服务器的路由隐藏作用被完全抵消，这为关联发送者和接收者提供了帮助。

3.4.3　分析

在 Step1 中，S_1 从报文 P' 中任取了一个密文块并猜测该块是关于某个下游恶意节点的路由信息块。如果猜测失败，则该报文到达某个诚实节点时会被解释为非法报文而丢弃，但 HS-Onion 并没有反向追踪机制，S_1 不用担心受到惩罚。因此，为了提高成功率，S_1 可以将 P' 复制 $?$ 份，然后在每一份中任取一个不同的密文块嵌入标签，并将这些复制报文都发送给下一个服务器。处于下游的恶意节点识别出标签后，还可以继续嵌入自己的标签，以方便更下游的恶意节点识别。以此类推，报文 P 转发路径上的多个恶意节点可以利用标签追踪攻击确定它们在路径中的相对位置和上下游关系。下面的定理给出了系统中所有节点协作，并利用这种更复杂的标签追踪攻击能准确关联特定消息发送者和接收者的概率。

定理 1　假设 S_a、S_b 利用标签追踪攻击确定它们分别为同一路径中最上游和最下游的恶意节点，系统中恶意节点所占百分比为 d，转发路径采用固定长度 $?$。如果直接猜测 S_a 的前驱和 S_b 的后继分别为发送者和接收者，则猜测正确的概率为 $d^2/[1-(1-d)^?-?d(1-d)^{?-1}]$。

证明　已知整个转发路径中存在一个恶意节点 S_a，首先嵌入标签，而 S_b 是路径中最后一个追踪到标签的恶意节点。该事件发生的概率等同于长为 $?$ 的路径(不包括发送者和接收者)中包含 2 个及 2 个以上恶意节点的概率，即 $1-(1-d)^?-?d(1-d)^{?-1}$，其中，$(1-d)^?$ 和 $?d(1-d)^{?-1}$ 分别为“不包含任何恶意节点”和“仅包含一个恶意节点”的概率。如果直接猜测 S_a 的前驱节点和 S_b 的后继节点分别为发送者和接收者，则存在一定的误差，因为 S_a 和 S_b 未

必是转发路径的第一个节点和最后一个节点。

转发路径首、尾节点都是恶意节点的概率为 d^2，因此在存在2个恶意节点并利用标签追踪攻击确定它们分别为最上游和最下游的恶意节点的条件下，它们的前驱节点和后继节点分别为发送者和接收者的概率为 $d^2/[1-(1-d)^{?}-?d(1-d)^{?-1}]$。

3.5 猜测接收者攻击

3.5.1 前提条件

猜测接收者攻击仅要求攻击者能够控制匿名转发路径的第一个节点，该攻击同时适用于全局攻击者模型和局部攻击者模型。

3.5.2 攻击过程

该攻击的基本思想是攻击者随机猜测报文接收者的地址，然后利用诚实Mix服务器的无意识转发服务和解密预言机服务验证猜测是否正确。如此反复猜测直到找出真正的接收者。

设发送者为 S_0，接收者为 $S_{?+1}$，转发路径由 $S_1, S_2, \cdots, S_?$ 构成，且 S_1 为恶意服务器。S_0 发送报文 P 到 S_1，P 的格式如式(1)所示。猜测接收者攻击的具体攻击过程如下。

Step1 S_1 接收到报文 P 后首先对其进行解密和再加密，获得下一跳路由地址 S_2 及如式(3)所示的报文 P'。S_1 将 P' 保存以备后面重复实施猜测攻击。

注意此时 S_1 并不能确定自己为第一个转发节点。

Step2 S_1 做2个猜测：1)P 的接收者为 S_t；2)从 P' 的前 $?+1$ 个密文中($\mathrm{URE}_{K(2,?+1)}(\mathrm{tag}' \| r_1)$ 除外)任取一块，不妨设为 $B_j=\mathrm{URE}_{K(2,j)}(\mathrm{tag} \| E_{x'_j}(\text{L}))$，并假定该块是关于 S_t 的密文。

基于以上猜测 S_1 对 P' 做如下修改：1)利用URE算法的可展性，根据 B_j 构造密文 $\mathrm{URE}_{K(2,j)}(\mathrm{tag} \| E_{x'_t}(S_1 \| R'))$，并用该密文替换 B_j；2)将第 $?+2$ 个密文块 $\mathrm{URE}_{K(2,?+1)}(1)$ 替换为 $\mathrm{URE}_{x_1+R'+K(2,?+1)}(1)$；3)将 $\mathrm{URE}_{K(2,?+1)}(\mathrm{tag}' \| r_1)$ 替换为 $\mathrm{URE}_{x_1+R'+K(2,?+1)}(\mathrm{tag}' \| r_1)$。

最后，S_1 将修改过的 P' 发送给 S_2。

Step3 经过一段时间，如果 S_1 从某个服务器 $S_{?+1}$ 接收到报文 P''，并且利用私钥 x_1 将其解密后具有形式：$\{\mathrm{tag}' \| r_1, \mathrm{tag}' \| r_2, \cdots, \mathrm{tag}' \| r_{?+1}, 1, e_k(m) \oplus H(r_1) \oplus \text{L} \oplus H(r_{?+1})\}$，则证明 $S_{?+1}$ 是接收者。将 P'' 解密获得 $?+1$ 个 tag' 的事实也证明 S_1 是第一个转发节点，并且其前驱 S_0 是发送者。至此，HS-Onion方案的不可关联性被完全破坏。

如果 S_1 长时间内没有收到任何符合上述条件的报文，则说明Step2的2个猜测中至少有一个是错的。S_1 重新猜测接收者地址和相应密文块 B_j 并重复上述攻击直至找到接收者。

实际上，上述攻击经过修改后可以用于确定构成匿名路径的任意节点。

3.5.3 分析

首先分析3.5.2节攻击的原理。在Step2中如果 S_1 猜测正确，即 $S_t=S_{?+1}$，$B_j=\mathrm{URE}_{K(2,?+1)}(\mathrm{tag} \| E_{x'_{?+1}}(k))$，那么Step2中经过修改的 P' 必然具有如下形式：

$$\{\mathrm{URE}_{x_1+R'+K(2,?+1)}(\mathrm{tag}' \| r_1),$$
$$\mathrm{URE}_{x_2}(\mathrm{tag} \| E_{x'_2}(S_3 \| R_2)), \cdots,$$
$$\mathrm{URE}_{K(2,?)}(\mathrm{tag} \| E_{x'_?}(S_{?+1} \| R_?)),$$
$$\mathrm{URE}_{K(2,?+1)}(\mathrm{tag} \| E_{x'_{?+1}}(S_1 \| R')),$$
$$\mathrm{URE}_{x_1+R'+K(2,?+1)}(1), e_k(m) \oplus H(r_1)\}。$$

这相当于将匿名路径延长了一个节点，S_1 作为终点，$S_{?+1}$ 被当作最后一个转发节点。由于报文中不含路径长度信息，因此这些修改不会被其他转发节点发现。

修改后的报文 P' 经过 $S_2, S_3, \cdots, S_{?+1}$ 的逐级转发重新回到服务器 S_1 时必然具有如下形式：$\{\mathrm{URE}_{x_1}(\mathrm{tag}' \| r_1), \cdots, \mathrm{URE}_{x_1}(\mathrm{tag}' \| r_{?+1}), \mathrm{URE}_{x_1}(1), e_k(m) \oplus H(r_1) \oplus \text{L} \oplus H(r_{?+1})\}$。该报文在Step3被解密后得到 $\{\mathrm{tag}' \| r_1, \mathrm{tag}' \| r_2, \cdots, \mathrm{tag}' \| r_{?+1}, 1, e_k(m) \oplus H(r_1) \oplus \text{L} \oplus H(r_{?+1})\}$。至此，$S_1$ 验证了猜测结果，确定了发送者和接收者，攻击结束。

如果 S_1 猜测错误，则 P' 在转发过程中会被某个诚实节点视为非法报文而丢弃，但由于没有反向追踪机制，S_1 不用担心受到惩罚。

下面分析“猜测接收者攻击”的代价。假设系统包含 N 个Mix服务器，转发路径采用固定长度 $?$，则 S_1 最多需要进行 $?(N-2)$ 次猜测验证过程。在 N 不算太大时，该攻击是非常有效的。为了缩短攻击时间，S_1 还可以将多个猜测验证过程并行进行，但需要在 P' 中针对不同的猜测加入不同的标记，具体过程就不再详细叙述了。

4 对HS-Onion的修正

4.1 两点修正

为了避免所提的3种攻击，在此提出对HS-Onion方案的两点修正措施。在原协议框架下，很难完全

消除 URE 算法的可展性，因为该特性是报文能够被随机化重加密和中间转发节点能够将自己的对称密钥 r_i 添加到报文的关键。因此，这些修正的主要目的是提高攻击者随意篡改报文的代价，尽量避免关键节点为攻击者提供解密预言机服务器。

修正 1　限制 Mix 服务器只提供匿名转发服务，本身不作为通信用户。类似于被广泛使用的洋葱路由系统 Tor[13]，修正后 HS-Onion 由匿名路由核心网络和外围用户群构成。N 个 Mix 服务器构成核心网络。发送者和接收者都属于外围用户群，它们利用核心网能够隐藏路由的转发服务来实现匿名通信。在全局攻击者模型下，匿名集合由同一时刻接入核心网的所有用户构成。

修正 2　增加对恶意节点的反向追踪机制。诚实 Mix 服务器在发现某个报文非法之后可以启动反向追踪机制，沿匿名路径回溯查找制造非法报文的源头。在调查过程中，从发现非法报文的节点 S_j 开始，沿路径逆向依次要求其前驱节点 S_{j-1}, S_{j-2},…提交证据证明自己正确处理了被追踪报文，直至找到恶意节点。每个节点的证明过程不能破坏其他报文的匿名性。目前已知的几种 URE 洋葱路由方案[6,8~10]都采用了基于零知识证明技术的反向追踪机制，使得 Mix 服务器可以在不公开密钥 x_i 的条件下证明自己合法地处理了报文。由于 HS-Onion 采用了 2 种代数结构不同的公钥算法，所以直接采用该方法较为困难。在此笔者设计了公开部分解密密钥和零知识证明相结合的证明方法。

证明　修正协议要求 Mix 服务器为每个输出报文增加数字签名之后再转发给下一个服务器，这样可以防止恶意节点对自己的攻击行为进行抵赖。设服务器 S_j 从前驱节点 S_{j-1} 接收到的报文 $P_{j-1}=\{B_{0,j-1},B_{1,j-1},\cdots,B_{?+3,j-1}\}$，经过解密和再加密处理后转发给下一级的报文为 $P_j=\{B_{0,j},B_{1,j},\cdots,B_{?+3,j}\}$。用符号 ZK{$x$:$A(x)$}表示关于秘密 x 的零知识证明，$A(x)$ 是一个关于 x 的断言，该证明能够在不暴露 x 的条件下证明断言 $A(x)$取真。下文的证明中 x 在 $A(x)$中都以离散指数形式存在，构造该类证明主要用到基于离散对数的零知识证明技术，具体构造方法可以参考文献[14]。

S_j 证明自己正确处理了报文 P_{j-1} 的具体过程如下。

1) S_j 公布如下内容：私钥 x'_j，下一跳地址 S_{j+1}，R_j，r_j，P_{j-1} 中属于 S_j 的密文块 $B_{u,j-1}$，P_j 中属于 S_j 的密文块 $B_{v,j}$ 以及由 P_{j-1} 生成 P_j 时所使用的所有 URE 再加密随机参数。

2) 为证明 x'_j、S_{j+1} 和 R_j 的合法性，S_j 公布 $\mathrm{ZK}\{x|B_{u,j-1}=\mathrm{URE}_x(\mathrm{tag}\|E_{x'_j}(S_{j+1}\|R_j))\wedge y_j=g^x\}$。

3) 针对集合$\{B_{0,j-1},B_{1,j-1},\cdots,B_{?+2,j-1}\}/\{B_{u,j-1}\}$中的每一个 URE 密文 $B_{p,j-1}=(a_0,ß_0,a_1,ß_1)$，$S_j$ 公布密文 $B_{q,j}\in\{B_{0,j},B_{1,j},\cdots,B_{?+2,j}\}$、密文 $B'_{p,j-1}=(a'_0,ß_0,a'_1,ß_1)$ 和 $\mathrm{ZK}\{x\mid a_0=a'_0ß_0^{x+R_j}\wedge a_1=a'_1ß_1^{x+R_j}\wedge y_j=g^x\}$，并证明 $B'_{p,j-1}$ 利用 1)公布的相关随机参数经过 URE 再加密后可以生成 $B_{q,j}$。3)证明了 P_j 的前 $?+2$ 个密文($B_{v,j}$ 除外)都可以由 P_{j-1} 中的密文经过部分解密和再加密生成。

4) 为证明 $B_{v,j}$ 和 r_j 的合法性，S_j 公布密文 $B'_{?+2,j}=(a_0,ß_0,a_1,ß_1)$ 并证明 $B'_{?+2,j}$ 经过 URE 再加密可以生成 $B_{?+2,j}$，且密文 $(a_0(\mathrm{tag}\|r_j),ß_0,a_1,ß_1)$ 经过再加密可以生成 $B_{v,j}$。

5) 针对 $B_{?+3,j}$ 的合法性，S_j 需要证明 $B_{?+3,j}=B_{?+3,j-1}\oplus H(r_j)$ 成立。

在证明过程中，S_j 只公开了私钥 x'_j 而没有公开 x_j，因此并不会对 S_j 过去转发报文的匿名性造成影响。

4.2　分析

4.2.1　安全性

下面具体分析修正措施对本文所提 3 种攻击的防范作用。

1) 关于数据指纹追踪攻击。经过修正 1 之后，接收者 $S_{?+1}$ 不再提供匿名转发服务，因此在数据指纹追踪攻击中恶意节点 $S_?$ 无法利用 $S_{?+1}$ 的解密预言机服务获得 $e_k(m)$及其数据指纹。但仅有修正 1 是不够的，攻击者可以采用由转发路径首尾节点 S_1 和 $S_?$ 合谋实施的指纹追踪方法：作为第一个转发节点的恶意服务器 S_1 从报文 P 的前 $?+1$ 个密文中任取一个并猜测它是属于 $S_?$ 的密文 $\mathrm{URE}_{K(1,?)}(\mathrm{tag}\|\mathrm{L})$。$S_1$ 由 $\mathrm{URE}_{K(1,?)}(\mathrm{tag}\|\mathrm{L})$ 构造 $\mathrm{URE}_{K(1,?)}(1)$，并将第 $?+2$ 块密文 $\mathrm{URE}_{K(1,?+1)}(1)$ 替换成 $\mathrm{URE}_{K(1,?)}(1)$。如果 S_1 猜测正确，则在该报文到达恶意节点 $S_?$ 时，$S_?$ 仍然可以从中获得 $e_k(m)$，从而将 S_1 和 $S_?$ 关联到一起，S_1 的前驱和 $S_?$ 的后继作为发送者和接收者也被关联到一起。和改进之前比，指纹追踪攻击必须依赖于随机猜测。如果猜测错误，则攻击者将被反向调查机制捕获。

设系统中恶意节点所占百分比为 d，转发路径

采用固定长度 $?$，下面的定理描述了在局部攻击者模型下修正方案抵抗指纹追踪攻击的能力。

定理 2 修正方案中攻击者利用数据指纹追踪攻击能成功关联发送者和接收者的概率为 $d^2/?$，而攻击者被捕获的概率为 $1-d$。

证明 基于随机猜测的指纹追踪攻击能够成功的条件是：① 路径首尾节点 S_1 和 $S_?$ 都是恶意节点；② S_1 能够猜中对应 $S_?$ 的密文块 $\mathrm{URE}_{K(1,?)}(\mathrm{tag}\,\|\,\mathrm{L})$。这 2 个条件成立的概率分别为 d^2 和 $1/?$，并且它们为相互独立的随机事件，因此，攻击能够成功的概率为 $d^2/?$。

在攻击中，S_1 从前 $?+1$ 个密文块中随机选择了 $B_j=\mathrm{URE}_{K(1,j)}(\mathrm{tag}\,\|\,E_{x_j'}(\mathrm{L}))$ 并将第 $?+2$ 个密文块 $\mathrm{URE}_{K(1,?+1)}(1)$ 替换成 $\mathrm{URE}_{K(1,j)}(1)$。在该报文到达节点 S_j 时，S_j 将发现第 $?+2$ 个密文的解密结果为 1，因此很容易发现该报文为非法报文。如果 S_j 为诚实节点，将会触发反向调查过程，恶意节点 S_1 必然被捕获。如果 S_j 为恶意节点，只会将该非法报文简单丢弃。因此，S_1 是否被捕获决定于它随机选择的密文 B_j 所对应节点 S_j 是否为恶意节点。因为构成转发路径的节点是发送者从全体服务器集合中随机选取的，所以 S_j 为恶意节点的概率为 d，而 S_1 被捕获的概率为 $1-d$。

2) 关于标签追踪攻击。在修正方案中，标签追踪攻击也必须由转发路径的首尾节点合谋实施，即由第一个转发节点嵌入标签，由最后一个转发节点从流经报文中提取标签。另外，标签追踪攻击也依赖于随机猜测。如果猜测错误，则攻击者将被反向调查机制捕获。定理 3 描述了在局部攻击者模型下修正方案抵抗标签追踪攻击的能力。

定理 3 修正方案中攻击者利用标签追踪攻击能成功关联发送者和接收者的概率为 $d^2/?$，而攻击者被捕获的概率为 $1-d$。

证明 标签追踪攻击成功的条件是：1) 路径首尾节点 S_1 和 $S_?$ 都是恶意节点；2) S_1 嵌入标签的密文块 $\mathrm{URE}_{K(1,j)}(\mathrm{tag}\,\|\,\mathrm{L})$ 恰好对应节点 $S_?$。这 2 个条件能够成立的概率分别为 d^2 和 $1/?$，因此，攻击能够成功的概率为 $d^2/?$。如果与嵌入标签的密文块 $\mathrm{URE}_{K(1,j)}(\mathrm{tag}\,\|\,\mathrm{L})$ 相对应的节点 S_j 为诚实节点，则 S_1 必然被反向调查机制捕获。如果 S_j 为恶意节点，则只会将该非法报文简单丢弃。因此，S_1 是否被捕获决定于它随机选择的密文 B_j 所对应节点 S_j 是否为恶意节点，即 S_1 被捕获的概率为 $1-d$。

3) 关于猜测接收者攻击。修正 1 使得攻击者无法再利用接收者 $S_{?+1}$ 的匿名转发服务验证自己对接收者的猜测，但攻击者可以利用类似的攻击思想猜测和验证构成转发路径的各个中间节点。这样做同样存在随机猜测过程，即需要攻击者随机选择密文块 B_j，并将其篡改为 $\mathrm{URE}_{K(2,j)}(\mathrm{tag}\,\|\,E_{x_t'}(S_1\,\|\,R'))$。如果 B_j 对应的节点 S_j 是诚实的，攻击者制造的非法报文将会被发现，因此，攻击者被捕获的概率同样为 $1-d$。

由以上分析可以看出，对原方案的两点修正虽然无法完全消除 Mix 服务器的解密预言机漏洞，但在很大程度上提高了攻击者篡改密文和利用解密预言机的代价。在典型的 Internet 分布式匿名系统中，一般认为恶意节点所占比例 d 的上限为 0.2[15]。在该条件下，根据定理 2 和定理 3 的结论，恶意节点在实施 3 种攻击时被捕获的概率不低于 $1-d$，即 80%。为了进一步提高反向追踪对攻击的抑制作用，还可以采用制定严厉惩罚措施等非技术性手段。

4.2.2 效率

在正常情况下，每收到一个报文，修正方案中的 Mix 服务器除了需要进行原协议规定的 $2?+3$ 次 URE 解密、$?+2$ 次 URE 再加密操作以及一次公钥算法 E 的解密操作之外，只需额外进行一次数字签名操作，因此服务器运算量增加很小。

在发现有非法报文的异常情况下，该报文转发路径上的部分节点时被迫参与反向追踪过程。每个相关节点为构造自己的合法性证明，需要进行 $3(?+2)$ 次群 G 上的指数运算，而验证该证明则需要进行 $10?+24$ 次指数运算和一次公钥算法 E 的解密操作。在异常情况下，最大的额外开销在于每个参与调查的服务器都要更换密钥，并向系统其他节点广播新密钥。因此，为了防止频繁攻击行为对性能的影响，可以采用加重惩罚力度的方法来降低服务器作弊的概率。

另外，为了能够在反向追踪中证明自己在上一级输出报文基础上做了正确的处理，每个服务器还需对包含数字签名的输入报文进行缓存。如果要求诚实服务器发现非法报文后必须立即启动反向追踪机制，那么 Mix 服务器只需缓存较短时间段内收到的报文，因此修正方案中缓存数据量较小，并且不需要将输入与缓存数据进行比对以检查是否有重复报文。与传统的基于缓存法对抗重放攻击的

洋葱路由系统相比，基于 HS-Onion 仍然具有很大优势。

5　结束语

基于通用重加密算法的洋葱路由系统能够有效抵御重放攻击，但存在严重的效率问题。为此，时金桥等提出了一种综合利用通用重加密和对称加密算法的混合结构洋葱路由方案 HS-Onion，降低了传输长消息时的计算复杂度。本文指出了 HS-Onion 方案的 2 处安全漏洞，并展示了 3 种能够完全破坏发送者和接收者不可关联性的攻击方法。这些攻击采用了与匿名通信系统中一些典型攻击类似的思想，如关系攻击[16]、路径猜测攻击[9]、分组计数器攻击[17]等，因此，对分析其他匿名通信系统的安全性具有一定的借鉴作用。近年来出现了很多针对匿名通信系统的关联攻击[15,18,19]，它们主要面向基于虚电路的低延时匿名通信系统(如 Tor[13])，并且大都采用了基于时域或频域的通信流量统计分析。与这些攻击方法相比，本文的攻击直接利用了 HS-Onion 方案的密码学漏洞，不需要连续缓存多个报文进行综合分析，因此要准确和高效得多。本文另一贡献是给出了针对 HS-Onion 方案的修正方法。这些修正以较小的代价在很大程度上降低了攻击者篡改密文和利用解密预言机获得有利信息的概率。

HS-Onion 及此前的所有基于通用重加密的洋葱路由方案都采用了启发式的安全性分析方法，而其中很多方案在提出后不久即被攻破，该事实表明在严格定义的模型下给出形式化安全性证明是十分必要的。这也是大多数匿名通信系统在进行安全性分析时所遇到的问题。因此，下一步的工作应该包括为混合结构洋葱路由系统建立能够正确反映系统安全属性及敌手攻击能力的安全模型，并在该模型下对方案的安全性进行证明。

参考文献：

[1] CHAUM D. Untraceable electronic mail, return addresses, and digital pseudonyms[J]. Communications of the ACM, 1981, 24(2):84-88.

[2] MOLLER U, COTTRELL L, PALFRADER P. Mixmaster protocol-version 2[EB/OL]. http://www.eskimo.com/rowdenw/crypt/Mix/draft-moeller-mixmaster2-protocol-00.txt, 2003.

[3] DANEZIS G, DINGLEDINE R, MATHEWSON N. Mixminion: design of a type III anonymous remailer protocol[A]. Proceedings of the 2003 IEEE Symposium on Security and Privacy[C]. Oakland, California, USA, 2003.2-15.

[4] KESDOGAN D, EGNER J, BUSCHKES R. Stop-and-go MIXes: providing probabilistic anonymity in an open system[A]. Proceedings of Information Hiding Workshop (IH 1998)[C]. Portland, Oregon, USA, 1998.83-89.

[5] GOLLE P, JAKOBSSON M, JUELS A. Universal re-encryption for mixnets[A]. Proceedings of the 2004 RSA Conference, Cryptographer's Track[C]. San Francisco, USA, 2004.163-178.

[6] GOMULKIEWICZ M, KLONOWSKI M. Onions based on universal reencryption anonymous communication immune against repetitive attack[A]. International Workshop on Information Secur ty Applications(WISA04)[C]. Jeju Island, Korea, 2004. 400-410.

[7] DANEZIS G. Breaking four mix-related schemes based on universal re-encryption[A]. Proceedings of Information Security Conference 2006 (ISC 2006)[C]. Samos, Greece, 2006.46-59.

[8] KLONOWISKI M, KUTYLOWSKI M, LAUKS A. Repelling detour attack against onions with re-encryption[A]. Proceedings of International Conference on Applied Cryptography and Network Security 2008 (ACNS 2008)[C]. New York, USA, 2008. 296-308.

[9] BORISOV N, KLONOWISKI M, MIROSLAW K. Attacking and repairing the improved ModOnions protocol[A]. Proceedings of the 12th International Conference on Information Security and Cryptology[C]. Seoul, Korea, 2009. 258-273.

[10] BORISOV N, KLONOWISKI M, MIROSLAW K. Attacking and repairing the improved ModOnions protocol-tagging approach[J]. KSII Transactions on Internet and Information Systems, 2010, 4(3): 380-399.

[11] 陆天波, 秦宝山, 李洋. 重加密匿名通道 WGRe[J]. 通信学报, 2009, 30(4):66-73.
LU T B, QIN B S, LI Y. WGRe: a re-encryption anonymous tunnel[J]. Journal on Communications, 2009, 30(4):66-73.

[12] 时金桥, 方滨兴, 郭莉. 抵御 MIX 重放攻击的混合结构消息报文机制[J]. 通信学报, 2009, 30(3):21-26.
SHI J Q, FANG B X, GUO L. Hybrid-structured onion scheme against replay attack of MIX[J]. Journal on Communications, 2009, 30(3): 21-26.

[13] DINGLEDINE R, MATHEWSON N, SYVERSON P. Tor: the second-generation onion router[A]. Proceedings of the 13th USENIX Security Symposium[C]. San Antonio, USA, 2004. 303-320.

[14] CAMENISCH J, STADKER M. Efficient group signature schemes for large groups[A]. Proceedings of Crypto'97[C]. Santa Barbara, California, USA, 1997. 410-424.

[15] WANG Q Y, MITTAL P, BORISOV N. In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems[A]. Proceedings of the ACM CCS 2010[C]. Chicago,

Illinois, USA, 2010. 308-318.

[16] PFITZMANN B. Breaking efficient anonymous channel[A]. Proceedings of Eurocrypt'94[C]. Perugia, Italy, 1994. 332-340.

[17] LING Z, LUO J Z, YU W. A novel cell counter based attack against tor[A]. Proceedings of the ACM CCS 2009[C]. Chicago, Illinois, USA, 2009. 578-589.

[18] ZHU Y, FU X W, RICCARDO B, *et al*. Analysis of flow-correlation attacks in anonymity network[J]. International Journal of Security and Networks, 2007, 2(1):137-153.

[19] STEFAN S, SEBASTIAN C. Using linkability information to attack mix-based anonymity services[A]. Proceedings of 9th International Symposium on Privacy Enhancing Technologies[C]. Seattle, WA, USA, 2009. 94-107.

作者简介：

李龙海（1976-），男，河北冀州人，博士，西安电子科技大学副教授、硕士生导师，主要研究方向为匿名通信、隐私保护技术和计算机网络安全。

付少锋（1975-），男，陕西户县人，西安电子科技大学副教授，主要研究方向为计算机网络安全和嵌入式系统。

苏锐丹（1978-），男，河南灵宝人，博士，西安电子科技大学副教授，主要研究方向为电子政务和网络安全协议设计。

车向泉（1972-），男，内蒙古赤峰人，西安电子科技大学高级工程师，主要研究方向为计算机网络安全和嵌入式系统。

（上接第 87 页）

[12] GOLDREICH O, GOLDWASSER S, MICALI S. How to construct random functions[J]. Journal of the Association for Computing Machinery, 1986, 33(4):792-807.

[13] **邓淼磊, 马剑峰, 周利华**. RFID **匿名认证协议的设计**[J].**通信学报，**2009, 30(7):20-26.
DENG M L, MA J F, ZHOU L H. Design of anonymous authentication protocol for RFID[J].Journal on Communications, 2009,30(7):20-26.

作者简介：

肖锋（1985-），男，湖北武汉人，北京邮电大学博士生，主要研究方向为物联网信息安全与隐私保护。

周亚建（1971-），男，陕西镇安人，北京邮电大学副教授、硕士生导师，主要研究方向为移动通信、无线传感网络、信息安全等。

周景贤（1982-），男，河南信阳人，北京邮电大学博士生，主要研究方向为无线传感网络及物联网安全。

钮心忻（1963-），女，浙江湖州人，博士，北京邮电大学教授、博士生导师，主要研究方向为信息安全、信息隐藏与数字水印技术、数字内容安全、软件无线电等。