C-Mix: A Lightweight Anonymous Routing Approach

Vinayak Kandiah, Dijiang Huang, and Harsh Kapoor

Arizona State University

Abstract. Low latency mix networks such as onion routing (Tor), heavily utilize cryptographic operations for transmitting a message to the receiver resulting in substantial computational and communication overhead. To address the performance and security issues of low latency mix networks, we propose a novel anonymous routing scheme called C-Mix. Its design principles are inspired by network coding techniques and the properties of polynomial interpolation. Based on our security analysis and performance evaluations, C-Mix achieves same level of anonymity with comparable computation overhead in comparison to traditional low latency mix networks.

1 Introduction

Communication networks are used in various activities of our everyday life and have become commonplace. A significant portion of the communications taking place over a conventional wired network requires both cryptographic security and user anonymity. Cryptographic security ensures secrecy of the message while user anonymity deals with hiding the identities of the communicating users and the routing path in the network. Several techniques have been proposed to achieve anonymity over communication networks [1,2,3,4]. Extensive research has taken place in the last few decades in a great effort to keep data communications private and anonymous. One main class of solutions that achieves user and path anonymity is based on *Onion Routing*, such as [5,3] and I2P [6], which provide anonymity for low latency networks using cryptographic operations and overlay techniques. Under normal circumstances, mix networks which use several Chaumian nodes provide anonymity, i.e. attackers cannot gain end-to-end routing information from the network. However, they are vulnerable to various active and passive attacks [7,8,9,10,11]. Many implementations of anonymous services for low latency networks and the Internet such as Tor and I2P [5,6] are based on onion routing. For example, onion routing (Tor) performs multiple encryptions and decryptions for both path setup phase and data forwarding phase. The amount of computation performed by the sender of a message (i.e., the sender performs multiple encryption for both path set up and data transmissions) is not commensurate with the computation at the other nodes in the network. Hence, scalability suffers when implemented in networks with low computation capability. This situation is especially true when a user wants to voluntarily use his/her non-high-performance computer (no dedicated hardware for cryptography computation) as a relay node to improve the communication anonymity. Thus, it is highly desirable to improve the computation efficiency of mix networks.

To address above describe issues of Mix networks, we present a novel technique – C-Mix – a coding inspired anonymous routing scheme. This new technique is inspired by network coding [12] where the information is processed and coded by each forwarding node. However, instead of using XOR coding method, we utilize the properties of polynomial interpolations [13] to achieve our security goal. In particular, the proposed anonymous scheme requires the use of the onion structure (public key cryptography) only during the path setup phase. Once a packet forwarding path has been established, no encryption/decryption is required to forward messages. C-Mix reconstructs the message in a distributed hop-by-hop manner. In this way, the intermediate nodes are only required to perform a few multiplications and addition operations over a finite field $\mathbb{Z}_p$ instead of performing the decryption operation. As a result, less computation efforts required at mix nodes in the network, thus improving the efficiency of the system. When compared to AES encryption/decryption, C-Mix significantly reduces the computation involved at the sender nodes and some improvements are achieved at forwarding nodes as well. Specifically, in cases where the size of the message is significantly greater than the size of the finite field $\mathbb{Z}_p$, our scheme greatly reduces the computational overhead at the mix nodes. The primary goal of this new technique is to introduce a novel Mix technology to reduce the computational overhead for Mix nodes. At the same time, it provides same level of anonymity and security as onion routing based mix schemes under strong attack models. Compared to existing low-latency mix networks, C-Mix is built on a stronger attack model, i.e., the adversaries are global passive or active observers. We evaluate the security and anonymity performance of C-Mix under insider colluding attacks, blending attacks, and traffic analysis attacks. The message secrecy can be proven based on the polynomial interpolation properties. We will show that C-Mix has levels of security and anonymity akin to those of the onion routing–based mix networks. Performance testing of the underlying mathematical techniques used in the proposed scheme has been done using real experiments.

The rest of the paper is organized as follows. Section 2 contains the system and models that will be used in C-Mix. In section 3, the formal mathematical foundation of C-Mix is provided. The security, anonymity and performance of the proposed scheme is analyzed and compared with onion routing in section 4. Finally, the scope for future work and concluding remarks are provided in section 5.

2 System and Attack Model

In this section, we first present the fundamentals of polynomial interpolation and the attack models used to evaluate the proposed technique.

2.1 Polynomial Interpolation and Secret Sharing

Polynomial Interpolation based secret sharing scheme [13] has been proposed to share a secret among N_{usr} entities such that any $t \leq N_{usr}$ entities can recover the secret but any $t' < t$ entities know no more about the secret than any non-participating entity. This scheme is based on the interpolation property of polynomials computed over a finite field $\mathbb{Z}_p$ according to which, a polynomial equation $y(x)$ of degree $(t-1)$ can be reconstructed only if at least t points that lie on $y(x)$ are available. A degree $t-1$ polynomial of the form

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_{t-1} x^{t-1} \tag{1}$$

is constructed, where a_0 equals the secret S (i.e., the y-intercept of the curve) and the other coefficients a_1, a_2, $\ldots, a_{k-1}$ are chosen randomly over the field $\mathbb{Z}_p$. Now, N_{usr} points $\{(x_j, y_j) | 1 \leq j \leq N_{usr}\}$ on the curve are obtained and distributed to the participating entities. Any t of these N_{usr} entities can combine their points to reveal the secret. The polynomial $y(x)$ can be reconstructed using the Lagrange Interpolation formula. If the points are represented as $\{(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)\}$, the Lagrange Interpolation formula is given by:

$$y(x) = \sum_{i=1}^{t} b_i y_i \qquad \text{where, } b_i = \prod_{k=1, k \neq i}^{t} \frac{x - x_k}{x_i - x_k}. \tag{2}$$

The traditional secret sharing scheme is operated in a centralized system, i.e., all participants submit their points to a server which computes the secret. In C-Mix, we need to decentralize the secret sharing algorithm, and we require that without the destination's involvement, intermediate nodes cannot collude to derive the secret.

2.2 Attack Model

We now present the attack models, against which the new scheme is evaluated. The attacks chosen in this paper are the ones that are commonly used to evaluate an anonymous routing scheme. In addition, these attacks are representative of the various classes of attacks like passive/active, internal/external. A passive attack is one where the adversary does not modify or alter the functioning of the anonymous scheme. An active attack on the other hand involves some form of intentional modification to the working of the scheme by the attacker like dropping, modifying or delaying the messages. An external attack is a form of attack where the adversary only has access to the communication links and hence cannot affect the intermediate nodes and an internal attack is one where the adversary can observe and/or modify the internal operations performed by the forwarding intermediate nodes in the network. We do not make restrictive assumptions for the attack models enabling us to evaluate our new scheme against a powerful adversary. Proving the anonymity and security of our scheme against a powerful attacker will obviously show the scheme's security and anonymity in a practical attack which may be less powerful. Without losing generality, our discussion focuses on three attacks. The collusion attack is an internal attack, where two or more forwarding nodes share all their knowledge regarding the data transmissions. The blending [7] and traffic analysis attacks are external. However, the blending attack is active while the traffic analysis attack is passive. The C-Mix relies on hop-by-hop in-network data processing, and it does not use cryptographic method to protect the message integrity. We assume that the end-to-end message integrity checking is at the application layer. In section 4, we examine the security and anonymity provided by our scheme when these attacks are deployed.

3 C-Mix Anonymous Routing Scheme

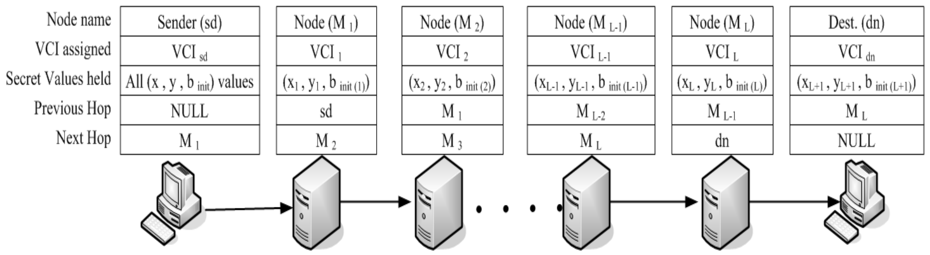
In this section, we start with the principles of our approach and then iteratively refine it by adding new mechanisms to increase the security and anonymity of the scheme. A list of symbols that are frequently used in the description of the C-Mix is provided in Table 1.

C-Mix involves three phases: virtual circuit setup, data transmission, and virtual circuit termination, which are described as follows. However, unlike onion routing, we do not use any form of symmetric encryption or decryption during the data transmission phase. We do not provide much detail about the circuit termination phase as breaking down a virtual circuit is trivial and can be performed by transmitting some equivalent of a KILL message.

Virtual Circuit Setup. In the circuit initiation phase, the sender (or source) identifies the path to be followed to the intended receiver (or destination). Let

Table 1. List of Symbols

Symbol	Description
General	
L	Number of forwarding nodes on the path.
INIT	Integer value chosen by sender for circuit initiation.
$y_{init}(x)$	Polynomial used by sender for circuit initiation.
M_i	Forwarding node i.
sd	Sender.
dn	Destination.
NUM	Number of messages to be transmitted.
I_α	Indicates the current message being transmitted ($1 \leq \alpha \leq NUM$).
$y_\alpha(x)$	Curve chosen by sender for message α.
C-Mix Scheme	
$b_{init(i)}$	Secret b value for node i obtained during circuit initiation.
(x_i, y_i)	Secret point for node i obtained at circuit initiation.
$(x_{\alpha(L+2)}, y_{\alpha(L+2)})$	Point chosen by sender for transmission of message α.
$b_{\alpha(i)}$	b value to be used by node i for transmission of α message.



Node name	Sender (sd)	Node (M_1)	Node (M_2)	Node (M_{L-1})	Node (M_L)	Dest. (dn)
VCI assigned	VCI_{sd}	VCI_1	VCI_2	VCI_{L-1}	VCI_L	VCI_{dn}
Secret Values held	All (x, y, b_{init}) values	$(x_1, y_1, b_{init(1)})$	$(x_2, y_2, b_{init(2)})$	$(x_{L-1}, y_{L-1}, b_{init(L-1)})$	$(x_L, y_L, b_{init(L)})$	$(x_{L+1}, y_{L+1}, b_{init(L+1)})$
Previous Hop	NULL	sd	M_1	M_{L-2}	M_{L-1}	M_L
Next Hop	M_1	M_2	M_3	M_L	dn	NULL

Fig. 1. Circuit Setup: Information distributed to the C-Mix nodes

there be L forwarding nodes in the path, which means $L + 2$ nodes (including sender and destination) are involved in the complete route. The sender now prepares an INIT value which is an integer over a finite field $\mathbb{Z}_p$. In rest of the paper, we assume all calculations are done over a finite field $\mathbb{Z}_p$ and we do not explicitly show it. The sender now chooses a polynomial $y_{init}(x)$ over the field $\mathbb{Z}_p$ which has INIT as the y–intercept:

$$y_{init}(x) = \text{INIT} + a_1x + a_2x^2 + a_3x^3 + \ldots + a_Lx^L.$$

The curve $y_{init}(x)$ has a degree L and the sender now determines $L + 1$ points that lie on it as $\{(x_1, y_1), (x_2, y_2) \ldots, (x_L, y_L), (x_{L+1}, y_{L+1})\}$. Now, in order to recompute the INIT value, we can substitute $x = 0$ and $t = L + 1$ in (2) to obtain these new equations:

$$y_{init}(0) = \text{INIT} = \sum_{i=1}^{L+1} b_{init(i)}\ y_i, \qquad \text{where} \quad b_{init(i)} = \prod_{k=1,k\neq i}^{L+1} \frac{x_k}{x_k - x_i}. \tag{3}$$

Now, the sender calculates the values $(b_{init(i)}|1 \le i \le L + 1)$ over the field $\mathbb{Z}_p$. In the circuit initiation phase, the INIT message need not be actually transmitted to the destination. The sender just uses this phase to distribute *triplets* of the form $(x_i, y_i, b_{init(i)})\ |1 \le i \le L)$ to the L forwarding nodes and the triplet $(x_{L+1}, y_{L+1}, b_{init(L+1)})$ to the destination. The x_i, y_i and $b_{init(i)}$ values will be used in the data transmission phase. Thus, the sender uses an onion to securely transmit the triplets to all forwarding nodes and the destination. The sender first prepares a header for the destination which contains the triplet $(x_{L+1}, y_{L+1}, b_{init(L+1)})$ and the public keys of the forwarding nodes in the path. This header is then encrypted using the destination's public key. To the resulting cipher text, the sender attaches a header containing $(x_L, y_L, b_{init(L)})$ along with the next hop and Time To Live (TTL) and encrypts the resulting content with the forwarding node M_L's public key. (Note that M_L is the last forwarding node on the path from sender to destination). In this way, the sender continues adding headers and encrypting iteratively in the reverse order of the path (i.e. last public key encryption will be using M_1's public key). The sender then transmits this onion message into the network. Each intermediate forwarding node i upon decryption of the onion message will perceive the message as:

$$\{nextHop, \{(x_i, y_i, b_{init(i)})\}, TTL, \{payload\}\},$$

where the $\{payload\}$ is an inner onion message. The first forwarding node M_1 on the path receives the onion message from the sender and decrypts it using its private key to read the header. M_1 realizes that this is a request to create a virtual circuit and hence generates a new VCI for this communication interface. The forwarding node then associates the previous hop, next hop and the triplet $(x_1, y_1, b_{init(1)})$ with the VCI and stores this information. This indicates to M_1 that for future data transmissions through the same interface it will have to use $(x_1, y_1, b_{init(1)})$ to operate on the message and forward it to the next hop (indicated by the VCI). In this way, as the onion message is transmitted through the network, a virtual circuit is created from the sender to the destination.

The triplets obtained by each node should not be disclosed as they are considered as secrets and will be used in the data transmission phase. The sender also retains all the triplets that it calculated and transmitted for this path. Note that each pair of neighboring nodes will choose their own VCI for the circuit being created. When the onion message reaches the destination the last layer of the onion message is decrypted and the destination obtains the triplet $(x_{L+1}, y_{L+1}, b_{init(L+1)})$. Figure 1 shows the values that each node on the path from sender to destination will store after the transmission of this onion message. Also in figure 1, though the previous hop of M_1 shows *sd* and the next hop of M_L shows *dn* these are not revealed as the sender and destination. This is because *sd* and *dn* are just identifiers of the nodes and are no different from the identifiers M_1 or M_L. We just use the *sd* and *dn* notations to clearly show the sender and destination to the reader. The destination will also receive the public keys of all the intermediate nodes in this path in the innermost layer of the setup message along with the triplet. Using these public keys, the destination now prepares a reply onion to the sender which will be sent through the reverse path (i.e. $M_L \rightarrow M_{L-1} \rightarrow \ldots M_1 \rightarrow Sender$). The sender on receiving this reply onion will know that the virtual circuit to the destination has been successfully setup and is ready for data transmission.

Data Transmission Phase. Let us assume that there are NUM messages to be sent by the sender. The sender then maps these NUM data packets to integers I_1, $I_2, \ldots, I_{NUM}$ over the field $\mathbb{Z}_p$ using a publicly known two–way function $Map()$. This function $Map()$ could be sent to the destination using the onion header during the circuit setup phase. To transmit an integer I_α, where, $1 \leq \alpha \leq NUM$, the sender uses the point $(0, I_\alpha)$ along with the points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L), (x_{L+1}, y_{L+1})\}$ to compute a polynomial $y_\alpha(x)$ of degree $L+1$ using polynomial interpolation. The sender can now identify a new point $\{x_{\alpha(L+2)}, y_{\alpha(L+2)}\}$ that lies on the polynomial $y_\alpha(x)$. The construction of the curve $y_\alpha(x)$ and its relationship with the initial curve $y_{init}(x)$ are shown in figure 2. The nodes on the path already possess $L+1$ points that lie on the

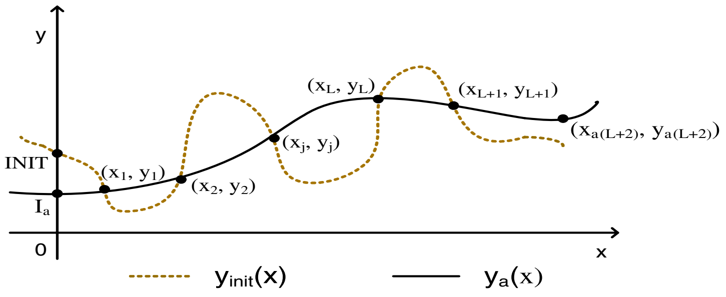


Fig. 2. Relationship between Circuit Initiation and Message Transmission Curves

polynomial $y_\alpha(x)$, i.e., the points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L), (x_{L+1}, y_{L+1})\}$ that were distributed during circuit setup. If we consider $\{x_{\alpha(L+2)}, y_{\alpha(L+2)}\}$ being the $L + 2^{th}$ point, then the sender along with the destination and the L forwarding nodes now have the potential to recalculate I_α using the following equations which are derived from (3):

$$y_\alpha(0) = I_\alpha = \sum_{i=1}^{L+2} b_{\alpha(i)}\ y_i \qquad \text{where, } b_{\alpha(i)} = \prod_{k=1,k\neq i}^{L+2} \frac{x_k}{x_k - x_i}. \tag{4}$$

For nodes $i = 1 \ldots L+1$ (i.e. the forwarding nodes and the destination), $b_{\alpha(i)}$ can be calculated as:

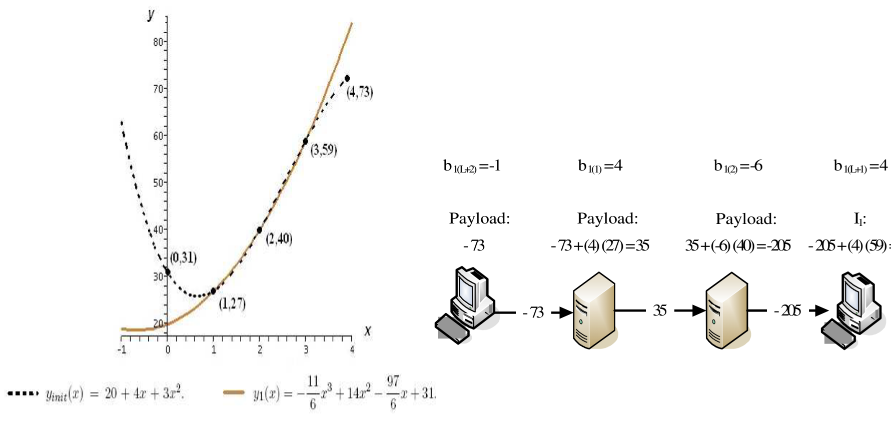
$$b_{\alpha(i)} = b_{init(i)}\ \frac{x_{\alpha(L+2)}}{x_{\alpha(L+2)} - x_i}. \tag{5}$$

From (5), each node i (where, $1 \leq i \leq L+1$) can calculate the $b_{\alpha(i)}$ values from $b_{init(i)}$, x_i and $x_{\alpha(L+2)}$. Each node i knows its $b_{init(i)}$ and x_i, and hence only needs the $x_{\alpha(L+2)}$ value. Also, the nodes are distributed over a network and hence the calculation specified in (4) should be done in a distributed manner while ensuring that the destination is able to determine I_α. The following procedure ensures that these requirements are satisfied.

1. The sender uses (4) to obtain $b_{\alpha(L+2)}$ from which $b_{\alpha(L+2)}.\ y_{\alpha(L+2)}$ can be calculated.
2. The sender transmits this along with $x_{\alpha(L+2)}$ to the forwarding node M_1. (Note that the sender can calculate $b_{\alpha(L+2)}$ as it has stored all the points it distributed to the other nodes).
3. The intermediate node M_1 now computes $b_{\alpha(1)}.\ y_1$ and adds this to the value $b_{\alpha(L+2)}.\ y_{\alpha(L+2)}$ received from the previous hop to generate a new payload. (Note that the node can calculate $b_{\alpha(1)}$ using (5)). This new payload is then transmitted to M_2 along with $x_{\alpha(L+2)}$.
4. M_2 performs operations similar to node M_1 and forwards the payload to the next hop along the path.

This procedure is continued until the destination adds $b_{\alpha(L+1)}.\ y_{L+1}$ to the payload it receives to reveal I_α. Remember that all the above calculations are performed over the finite field $\mathbb{Z}_p$.

An Example of Using C-Mix Basic Scheme. A simple example is provided to illustrate the operations of the C-Mix basic scheme. To simplify the presentation, the following example is demonstrated over the characteristic field zero. Assume the following 3-hop routing path for this example: $sd \rightarrow M1 \rightarrow M2 \rightarrow dn$, where sd is the sender and dn is the destination. Hence, for this case $L = 2$. Let the INIT value chosen by the sender be 20 and let the polynomial of degree 2 be: $y_{init}(x) = 20 + 4x + 3x^2$. The sender then locates the following 3 points on the curve: $\{(1, 27), (2, 40), (3, 59)\}$. The sender also calculates the $b_{init(i)}$ values as: $b_{init(1)} = 3$, $b_{init(2)} = -3$, $b_{init(dn)} = 1$. The sender distributes the triplets $\{(1, 27, 3), (2, 40, -3), (3, 59, 1)\}$ to the forwarding nodes and the destination. This completes the circuit setup phase. Now, let us assume that the sender needs to transmit $I_1 = 31$ to the destination. The sender first constructs a polynomial of degree 3 (as, $L + 1 = 3$) using the following points: $\{(0, 31), (1, 27), (2, 40), (3, 59)\}$. The resulting polynomial of degree $L + 1 = 3$ is $y_1(x) = -\frac{11}{6}x^3 + 14x^2 - \frac{97}{6}x + 31$.



(a) (b)

Fig. 3. (a) Shared points on the initialization and I_1 polynomials. (b) Calculations performed at each node during transmission of I_1.

Note that the two curves $y_1(x)$ and $y_{init}(x)$ share the points held by the intermediate nodes and the destination as shown in figure 3(a). The calculations performed at each node and the payload forwarded during transmission of I_1 is shown in figure 3(b). Next the sender finds the $L + 2^{th}$ point on this curve as $(4, 73)$. The $b_{1(L+2)}$ value is calculated by the sender as -1. The payload for initial transmission is prepared as $b_{1(L+2)}\ y_{1(L+2)} = -73$ which is transmitted along with $x_{1(L+2)} = 4$ to M_1. The forwarding node M_1 calculates $b_{1(1)}$ as: $b_{1(1)} = b_{init(1)} \frac{x_{1(L+2)}}{x_{1(L+2)} - x_1} = 3\ \frac{4}{4-1} = 4$. Thus, the new payload is calculated as $-73 + (4)(27) = 35$ and it is forwarded to M_2. Similar to the previous step, M_2 can calculate $b_{1(2)} = -6$ and recompute the new payload as

$35 + (-6)(40) = -205$. The destination gets this new payload and calculates $b_{1(L+1)} = 4$ and $I_1 = -205 + (4)(59) = 31$. It can also be observed from this simple example that the intermediate nodes cannot collude to derive I_1.

Using Globally Determined $x_{\alpha(L+2)}$ Values. Previously presented C-Mix scheme is susceptible to colluding nodes or passive traffic analysis attack, which allow the adversary to obtain information about the path. The anonymous routing scheme based on polynomial interpolation needs to ensure that a forwarding node calculates different $b_{\alpha(i)}\ y_i$ values for each transmission along the virtual circuit. Also, colluding nodes on a virtual circuit should not be able to obtain path information by comparing $x_{\alpha(L+2)}$ values that are propagated by the sender.

To achieve the above desired property, we propose to use a globally known function $\Psi(\alpha)$ which determines an $x_{\alpha(L+2)}$ for every data transmission I_α where $(1 \leq \alpha \leq NUM)$. This function should be made accessible to all the users (senders and destinations) and forwarding nodes across the network. The forwarding nodes will use $\Psi(\alpha)$ for all virtual circuits that they are handling. Thus, all nodes on the path keep a track of the number of messages transmitted through each virtual circuit. After the sender has calculated the new polynomial $y_\alpha(x)$ for the message I_α, it uses $\Psi(\alpha)$ as its x–coordinate $x_{\alpha(L+2)}$ (where α is basically one plus the number of messages it has previously sent through the circuit). The sender obtains the corresponding $y_{\alpha(L+2)}$ from the polynomial. It then calculates the $b_{\alpha(L+2)}\ y_{\alpha(L+2)}$ value and transmits this as the payload to the first forwarding node M_1. As it is easy for M_1 to keep track of the number of messages it has forwarded so far for this particular VCI, it can determine $x_{\alpha(L+2)} = \Psi(\alpha)$ and update its $b_{\alpha(1)}$ as $b_{init(1)}\ \frac{\Psi(\alpha)}{\Psi(\alpha)-x_1}$. The node M_1 can then determine $b_{init(1)}\ y_1$, update the payload and forward it to the next node on the path. In general, a node M_i (or the destination) can update its $b_{\alpha(i)}$ as:

$$b_{\alpha(i)} = b_{init(i)}\ \frac{\Psi(\alpha)}{\Psi(\alpha) - x_i}. \tag{6}$$

In this manner, all nodes in the network including the sender and the destination can calculate the appropriate value for each transmission α. This ensures that each node calculates different $b_{\alpha(i)}\ y_i$ for every transmission. This will prevent a passive attacker from correlating incoming and outgoing messages at a node because the difference in value between the incoming and outgoing messages is different for every transmission. This approach of using globally determined values also mitigates the collusion attack. Note that the forwarding node uses the same global function $\Psi(\alpha)$ for all the virtual circuits that it supports and the number of transmissions (α) for every virtual circuit begins at 1 and increments by 1 for every transmission. As a result, two or more colluding nodes cannot compare their $\Psi(\alpha)$ values to determine path information for a particular sender–destination pair because for a particular α the $\Psi(\alpha)$ value is the same for every virtual circuit in the system.

However, it is likely that different senders ($sd_1, sd_2, \ldots$) have different number of messages to transmit (NUM_{sd_1}, $NUM_{sd_2}, \ldots$). Consider a case where a sender has a lot of messages to transfer than the average in the network. After a period of time, only that sender will have very high α values. In this scenario, when forwarding nodes collude, they will notice the presence of a virtual circuit with a large α value. Although two or more non-contiguous forwarding nodes cannot determine with complete certainty that the virtual circuits with the high α are on the same path, the probability of these colluding forwarding nodes being on the same path is high. In order to offer better protection against collusion in such scenarios it is necessary to ensure that each virtual circuit can only transmit messages until a certain α_{max} and a corresponding $\Psi(\alpha_{max})$ are reached. After the sender has sent α_{max} messages through the path, the circuit must be broken down and a new virtual circuit (which may take a different path) must be created. Thus the sender needs to continually do this until all its NUM_{sd} messages are transmitted. The sender uses this new virtual circuit to transmit the remaining messages ($I_{\alpha_{max}+1} \ldots I_{NUM_{sd_X}}$) messages assuming ($NUM_{sd_X} < 2\alpha_{max}$). However, when the sender transmits the message $I_{\alpha_{max}+1}$, it is the first message transmitted through the new virtual circuit. As a result all nodes in the path consider $\alpha = 1$ and use $\Psi(1)$ to calculate $b_{\alpha(i)}$. Thus, the sender also uses $x_{\alpha(L+2)} = \Psi(1)$ to calculate $b_{\alpha(L+2)}$ and $y_{\alpha(L+2)}$. In effect, the virtual circuit uses $\alpha = 1 \ldots (NUM_{sd_X} - \alpha_{max})$ for the transmission of messages ($I_{\alpha_{max}+1} \ldots I_{NUM_{sd_X}}$). In this case, we assumed that the sender sd_X only needed 2 virtual circuits to transmit all messages to the intended destination. However, in general the number of virtual circuits that need to be constructed by a sender is given by the following equation:

$$\text{Number of VCs for } sd_X = RndUp(\frac{NUM_{sd_X}}{\alpha_{max}}),$$

where $RndUp()$ rounds the result to the ceiling function.

We must note that each forwarding node may support multiple virtual circuits. The forwarding node also keeps track of the α value for each of these virtual circuits. When the α_{max} value is reached for a particular VC, only that circuit is torn down while the functioning of all other VCs supported at the forwarding node is unaffected. Using this scheme of Globally determined $x_{\alpha(L+2)}$ values bounded by α_{max}, C-Mix scheme is resilient to collusion attacks by comparing α values.

4 Security, Anonymity, and Performance Analysis

In this section, we first discuss how the various attacks can be launched on the C-Mix scheme and what the attacker can learn from these attacks. The impact of the revealed information on the security and anonymity of the proposed scheme is also evaluated. In the second part of this section, we present the performance analysis of the C-Mix scheme which is based on experimental results.

4.1 Anonymity and Security Analysis of the C-Mix Scheme against Attacks

The attack model is described in section 2.2. The effect of the various attacks and the extent of information revealed from them are discussed here.

Resilience to Collusion Attack. This is a type of attack where several forwarding nodes in C-Mix collude and try to obtain the path information and the message being transmitted. In a scenario where the colluding nodes are contiguous, the path information will be revealed. This is obvious because a forwarding node can compare the messages it sent with the messages received by neighboring node. Now we consider a case where there is at least one uncompromised node FN_h in the network. This is sufficient to preserve the end–to–end path anonymity. The colluding nodes can track the path up to the node FN_h. However, the colluding nodes do not know the results of the calculations performed by FN_h. Hence, they cannot track the path by just observing the output messages of FN_h.

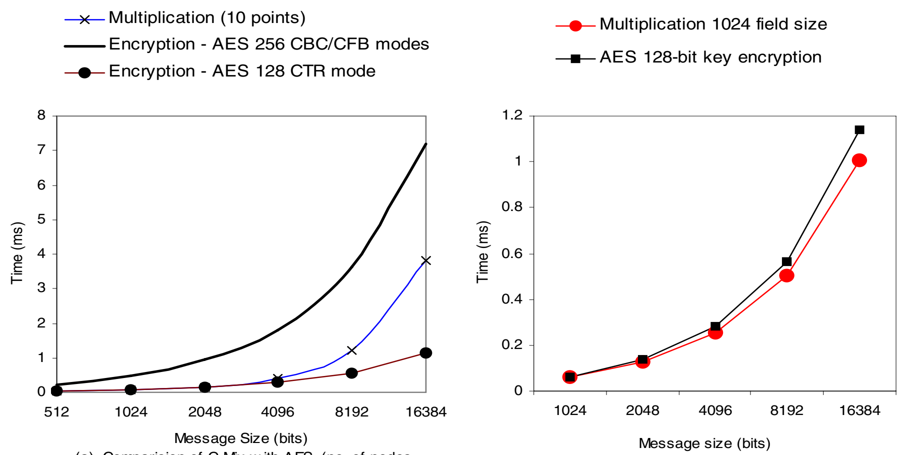
Thus, when a collusion attack is launched, the level of anonymity and security provided by the C-Mix scheme is similar to those provided by the onion routing–based schemes. Even in onion routing–based schemes, a single uncompromised node is sufficient to preserve end–to–end path anonymity and non-contiguous colluding nodes can compare the number of messages transmitted through a virtual circuit.

Blending Attack. A successful blending attack discloses the portion of the path which is enclosed by the compromised nodes. However, it is necessary to check if any other information is revealed by the attack. To increase the amount of information unknown to attacker at each node, we allow each node to have more than one triplet. We call the number of triplets held by a node for a virtual circuit as a "multiplicity". In the first case, assume that the attacker has also been able to obtain the secret multiplicities of each node. Consider a blending attack at a node on the transmission path. The adversary can observe the incoming and outgoing message at the compromised node and calculate their difference as a constant K over the field $\mathbb{Z}_p$. In this case, if the attacker can carry out adequate blending attacks he/she can construct sufficient equations which the forwarding node would have used to obtain K. If the number of equations is equal to or more than the number of unknowns, the system of equations can be solved over the field $\mathbb{Z}_p$. The solution will reveal the triplets held by the node i. However, this information in turn can only reveal the next hop (which is already known to the attacker) and does not compromise other forwarding nodes. In a more practical scenario where the adversary does not know the multiplicity, it is more difficult for the attacker to obtain equations and solve them. The attacker will have to guess the lowest and highest possible multiplicities and obtain systems of equations for each possible multiplicity value in the range. Also, the blending attack does not compromise the secrecy of the message I_α as the adversary cannot obtain the receivers triplets. When compared to onion routing–based schemes, the security of the C-Mix scheme is the same – the message cannot be revealed. However, in terms of anonymity C-Mix may reveal the triplets of forwarding nodes in addition to the next hop. But, this is not very useful as the compromised triplets cannot be used to track the path at other nodes.

Traffic Analysis Attacks. Finally, the impact of traffic analysis attacks on the proposed scheme is discussed. The simplest form of traffic analysis involves identifying transmission paths by examining the network for changes in the size of messages and similarities in the appearance of the messages. Both these techniques will be unsuccessful against our scheme because we use globally determined $x_{\alpha(L+2)}$ values and multiple $(x_i, y_i, b_{init(i)})$ triplets for each node. These two techniques make the same message I_α appear differently over different links. However, C-Mix is still vulnerable to long term traffic analysis attacks like the hitting set attack [14] as these attacks are based on the volume of traffic sent and received by the end users. The secrecy of the message I_α is not compromised by passive traffic analysis attacks. Even though the payloads are not encrypted, the attacker cannot deduce the final value of the message by observing the intermediate values. This is due to the fact that the values held by the destination can be obtained only if the exact polynomial used is known and there is no way for the passive attacker to determine the polynomial. Also, it is not possible for the adversary to deploy dictionary attacks on the observed data values to deduce the message (I_α). A dictionary attack is one where the adversary obtains various cipher–text messages which were encrypted using the same symmetric key. Then using frequencies of occurrence for the letters in the English Alphabet, the attacker decodes the cipher text to obtain the plain–text. In the proposed scheme, even though the same function $Map()$ is used for every transmission to convert text to integer messages, the value to be added by the destination to recover the integer I_α changes for every transmission. As a result, the frequency of occurrences of the numbers in the messages transmitted from the last forwarding node to the destination is distorted. As for anonymity of C-mix scheme the simplest form of traffic analysis involves identifying transmission paths by examining the network for changes in the size of messages and similarities in the appearance of the messages. Since all the calculations are performed in the finite field $\mathbb{Z}_p$, such attack will be unsuccessful.We note that the C-Mix scheme is potentially still susceptible to well-deployed traffic analysis attacks like the timing attack [15]. This is potentially because the existing low-latency mix networks are relatively static and only involve small number of mix nodes, for example, Tor only allows 3-hop path length route. Due to the computational and communication efficiency of C-Mix, we can allow a longer path and involve more mix relay servers. With more nodes involved the end-to-end latency may be deviated more and thus the consequence of timing attack can be also mitigated.

4.2 Computational Performance Assessment of C-Mix Based on Experimental Results

In this section, we present the performance assessment based on our experimental results. The time taken to perform the encryptions in AES and the field



(a) Comparision of C-Mix with AES (no. of nodes = 10, Message size = Field size)

(b) Comparison of C-Mix with AES-128 CTR mode (field size = 1024, no. of nodes = 10)

Fig. 4. Performance Evaluations

multiplications were measured using the C/C++ code [16] and Maple, respectively. The hardware configuration of the machine used for testing is: AMD 64 bit, 2 Ghz processor with 1G RAM. We assume an anonymous path that has 10 nodes. The encryption time was measured for AES 128-bit key and AES 256-bit key encryption in CTR, CBC and CFB modes for various message sizes (512 bits to 16,384 bits). The time taken to perform multiplications with various values of finite field and message sizes were recorded. The value of the finite field $\mathbb{Z}_p$ is determined by the bit representation size of this value i.e. a 512 bit field means that the value of the field is between $2^{511} + 1$ and 2^{512}. In the first case, we assumed the sizes (not values) of the message and the field to be the same. With this assumption, we compare the time taken to perform AES encryption with time taken to perform field multiplications on various message sizes in Figure 4(a). For message sizes up to 4096 bits, it can be seen that C-Mix scheme has computation cost between AES 128-bit key encryption and AES 256-bit key encryption.

However, the computation time for C-Mix can be further optimized by selecting a finite field size which is smaller than the message size. In this case, the C-Mix will process the message in fixed–size blocks (determined by field size). For example, a field size of 1024 bits can process a 2048 bit message by operating on the first 1024 bits and then on the second block. Note that choosing a field size smaller than message size does not compromise the security of the message. Each block of size k bits will have the same security offered by using a k bit field over a k bit message. The next comparison considers message whose sizes are multiples of 1024 and shows the time taken to perform AES encryption and C-Mix operations using a fixed 1024 bit size finite field. It can be seen from Figure 4(b) that the time taken for preparing messages of all sizes is least when C-Mix with a fixed size finite field. In onion–routing techniques, the intermediate node just needs to perform one decryption and forward the message. In C-Mix, the forwarding node needs to update its $b_{\alpha(i,j)}$ values for each point it holds. This involves one multiplication for each point possessed by the node. The time observed for decryption of various size messages using 128 bit keys is very close to the time taken to perform an encryption.

4.3 Communication Performance Analysis of C-Mix

After the path establishment phase, C-Mix sender does not need to create an onion structure for data transmission (see Figure 3). Instead, the sender just needs to specify the outgoing VCI number and attach the initial value I_0. Once the second C-Mix node received the message, it checks its routing table and swap a new outgoing VCI number and computer $I_1 = I_0 + b_1 y_1$ over the finite field $\mathbb{Z}_p$. Then the new computed value I_1 is the new payload. Due to the field operation, the payload size will never increase. Using C-Mix, the communication overhead is similar to the low latency mix network solutions such as Tor [17].

4.4 Storage Performance Analysis of C-Mix

In Tor, each intermediate node stores a shared key with the source. While in C-Mix, each intermediate node needs to store one or multiple coordinates distributed by the source. Thus, the storage complexity of C-Mix is at the same level of Tor.

5 Conclusion

Most of the existing anonymous routing schemes like Tor are based on the onion routing technique. Onion routing requires multiple encryptions to be performed by a sender for the transmission of each message. The forwarding nodes in the transmission path decrypt once and forward to the next hop. Due to the heavy use of encryption, these schemes have a considerable computational overhead (especially at the sender). As a result, scalability suffers in large networks.

In this paper, we have proposed a novel anonymous routing technique called C-Mix inspired by network coding and the properties of polynomial interpolation. C-Mix reduces the computation time required by the forwarding nodes for all message sizes. In general, our proposed C-Mix scheme is built on strong attack models and resilient to several global passive and active attacks, which are not addressed by existing low-latency mix networks.

However, there is still scope for improving C-Mix and for applying these techniques in other computing environments. Particularly, our research challenges are **(a)** Improving the computational efficiency further by reducing the number of multiplications performed by the sender. **(b)** Enabling the sender to dynamically change the path to the destination during the message transmission phase using predistributed secrets. **(c)** Anonymity and security performance under traffic analysis attacks is required for longer path length in large C-Mix networks. **(d)** Apply the principles of C-Mix in wireless network environments.

Acknowledgement

The authors would like to thank anonymous reviewers for their insightful comments to improve the quality of this paper.

References

1. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
2. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology 1(1), 65–75 (1988)
3. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding Routing Information. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 137–150. Springer, Heidelberg (1996)
4. Gulcu, C., Tsudik, G.: Mixing E-mail with Babel. In: Proceedings of the Symposium on Network and Distributed System Security, pp. 2–16 (1996)
5. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium (August 2004)
6. I2P: Anonymizing Network, `http://www.i2p.net/`
7. Serjantov, A., Dingledine, R., Syverson, P.: From a Trickle to a Flood: Active Attacks on Several Mix Types. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 36–52. Springer, Heidelberg (2003)
8. Back, A., Moller, U., Stiglic, A.: Traffic analysis attacks and trade-offs in anonymity providing systems. In: Moskowitz, I.S. (ed.) IH 2001. LNCS, vol. 2137. Springer, Heidelberg (2001)
9. Danezis, G.: Statistical disclosure attacks: Traffic confirmation in open environments. In: Proceedings of Security and Privacy in the Age of Uncertainty (SEC 2003), pp. 421–426 (2003)
10. Levine, B., Reiter, M., Wang, C., Wright, M.: Timing Attacks in Low-Latency Mix Systems. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 251–265. Springer, Heidelberg (2004)
11. Zhu, Y., Fu, X., Graham, B., Bettati, R., Zhao, W.: On flow correlation attacks and countermeasures in mix networks. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 207–225. Springer, Heidelberg (2005)
12. Ahlswede, R., Cai, N., Li, S., Yeung, R.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
13. Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)
14. Kesdogan, D., Pimenidis, L.: The Hitting Set Attack on Anonymity Protocols. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 326–339. Springer, Heidelberg (2004)
15. Murdoch, S., Danezis, G.: Low-cost Traffic Analysis of Tor. In: IEEE Symposium on Security and Privacy, pp. 183–195. IEEE CS, Los Alamitos (2005)
16. Gladman, B.: AES-CTR C Implementation, `http://fp.gladman.plus.com/cryptography_technology/fileencrypt/index.htm`
17. Dingledine, R., Mathewson, N.: Tor Protocol Specification, `http://www.torproject.org/svn/trunk/doc/spec/tor-spec.txt`