HECTor: Homomorphic Encryption Enabled Onion Routing

1st Saikrishna Gumudavally
2nd Ye Zhu
Department of Electrical Engineering and Computer Science
Cleveland State University
Cleveland, USA
s.gumudavally@alumni.csuohio.edu
y.zhu61@csuohio.edu

3rd Huirong Fu
Department of Computer Science and Engineering
Oakland University
Rochester, USA
fu@oakland.edu

4th Yong Guan
Department of Electrical and Computer Engineering
Iowa State University
Ames, USA
yguan@iastate.edu

***Abstract*—Despite increasing popularity of anonymous communications, current anonymity networks are still vulnerable to traffic analysis attacks. We propose Homomorphic EnCrypTion enabled Onion Routing (HECTor) to defeat existing traffic analysis attacks. Instead of mixing traffic at the packet level as current anonymity networks, HECTor mixes traffic at the bit level with network coding techniques. In HECTor, homomorphic encryption enables onion-like layered encryption which has been proven successful in existing anonymity networks. We theoretically analyze the performance of HECTor. Our extensive experiments on HECTor show that HECTor can effectively mitigate existing traffic analysis attacks with high throughput.**

I. Introduction

Anonymity networks are becoming increasingly popular. It has been reported that Tor [1], the second generation onion routing, has about 1.2 million users already. Despite popularity of anonymity networks, existing anonymity networks are still vulnerable to traffic analysis attacks. Since packets in existing anonymity networks such as Tor are of the same size, most of effective traffic analysis attacks [2]–[4] are based on packet timing. Our goal is to design an architecture for anonymous communications that can defeat existing traffic analysis attacks.

Our design follows the same philosophy behind the design of current anonymity networks, which is to mix traffic or hide in crowds [5]. Current anonymity networks mix traffic at the packet level with techniques such as batching [6], pooling [6], and rerouting [1], [7]. We propose to mix traffic at the bit level and the mixing at the bit level renders current packet-based traffic analysis attacks largely ineffective. Moreover, because of the mixing, a packet may contain bits from different sources. So the anonymity definition for the new anonymous communication architecture needs to redefined.

In this paper we propose an anonymous communication architecture called Homomorphic EnCrypTion enabled Onion Routing (HECTor) to defeat traffic analysis attacks. HECTor mixes packets at the bit level with network coding, more specifically random linear coding [8] for the mixing in this paper. HECTor also adopts onion-based layered encryption [1], [7] which has been proven successful in anonymity protection: One anonymity network node in a communication path can only process the outer layer of a packet and the inner layers are protected by the layered encryption. However bit-level mixing is not compatible with the layered encryption because bit-level mixing at the node in the path needs to re-mix packet received by the node. Homomorphic encryption is used to make bit-level mixing compatible with layered encryption as homomorphic encryption enables computation over encrypted data.

The rest of the paper is organized as follows: Section II reviews related work. We describe design goals and threat models in Section III. Section IV introduces essential techniques used in HECTor. The design of HECTor is presented the Section V. We theoretically analyze HECTor in Section VI. The performance of HECTor is evaluated in Section VII. We conclude the paper in Section VIII.

II. Related Works

Since Chaum's pioneer idea [9], researchers have applied the idea to different applications such as message-based email and flow-based low-latency communication, and they have developed new defense techniques as more attacks have been proposed. In low-latency anonymous communication networks, users connect to a pool of mixes, which provide anonymous communication, and users select a forwarding path through this core network to the receiver. *Onion routing* [7]and TOR [1] belong to this category.

In the mean time, the researches on traffic analysis attacks, which are designed to compromise communication privacy, are being conducted for various applications and services [10]–[14]. Danezis [11] proposes an attack on the Continuous Mix that can delay packets according to some probability distribution. Since the packet delays are independent, the departure distribution of the packets of a flow can be accurately described (if one ignores queuing) by convoluting the packet-arrival and the delay distribution. This can be used as a basis for measuring similarities among flows and in turn to compromise communication privacy. In [13] Murdoch and Danezis stage an active attack to trace back connections from a server to the victim client by modulating the traffic to the victim at the server and by remotely "sensing" the modulation

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

by probing its interference on cross traffic that is generated by one or more corrupt Tor nodes. In [4], [12], we propose flow correlation attacks to detect the communication path taken by a traffic flow. The adversary detects the communication path with correlation-based approaches.

The anonymity of physical-layer network coding is investigated in [15]. The author in [15] focuses on message content dependency between an input link and an output link of a message relay.

III. Design Goal and Threat Models

In this paper, we propose HECTor, a novel architecture for anonymous communications. The major design goal is to defeat traffic analysis attacks proven effective to existing anonymity networks. Existing traffic analysis attacks can be largely classified into two categories: timing based traffic analysis attacks and packet size based traffic analysis attacks. In this paper, we focus on mitigating timing-based traffic analysis attacks because packets in existing anonymity networks such as Tor [1] are of the same size and the packet size based traffic analysis attacks are largely ineffective in compromising communication anonymity in these anonymity networks. However timing-based traffic analysis attacks are proven effective in compromising anonymity in existing anonymity networks.

We assume that an adversary would like to compromise anonymity of Alice's communications. The capabilities of the adversary are assumed as follows: (1) For confidentiality, Alice encrypts her packets. So the adversary has no access to packet content. Since packets in the proposed anonymity network are of the same size as in Tor, the information available to the adversary to launch traffic analysis attacks is packet timing. (2) The adversary can collect traffic on any link in the proposed anonymity network for anonymity attacks. (3) We also assume that the adversary can collect traffic sent by Alice. (4) The adversary can correlate Alice's traffic to traffic collected on links in the proposed anonymity network to detect a path used by Alice's traffic.

Same as existing anonymity networks, the proposed anonymity network is not designed to defeat end-to-end traffic analysis attacks because these attacks can compromise communication anonymity by correlating traffic flows entering an anonymity network to traffic flows exiting from the anonymity network. One way to defeat the end-to-end traffic analysis attacks is to run anonymous communication services on the source node and/or the destination node of a traffic flow so that the adversary will not have access to traffic entering or leaving the anonymity network.

IV. Preliminaries

Before presenting the details of HECTor, we would like to briefly introduce the homomorphic encryption and the network coding, the two essential techniques used in HECTor.

A. Homomorphic Encryption

Although the concept of privacy homomorphism [16] was first generated in 1978, fully homomorphic encryption is not possible until the recent breakthrough [17]. In this paper, we use the homomorphic encryption scheme proposed by Dijk *et al.* [18] due to its simplicity and efficiency.

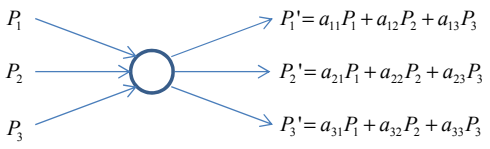


Fig. 1: Random Linear Network Coding

Homomorphic encryption enables computation over encrypted data. The main idea of homomorphic encryption is as follows: Suppose encrypting a plaintext message M_i with the key k results in the ciphertext C_i, i.e., $C_i = E_k(M_i)$. An encryption scheme is homomorphic if, for any function f,

$$\begin{aligned} E_k(f(M_1, M_2, \cdots, M_n)) &= f(E_k(M_1), E_k(M_2), \cdots, E_k(M_n)) \\ &= f(C_1, C_2, \cdots, C_n) \end{aligned} \quad (1)$$

where n denotes the number of plaintext inputs. As shown in Equation 1, homomorphic encryption schemes allow a third party to compute the function f on the encrypted data, i.e., the ciphertext C_i, without knowing the key k. And the output of the function $f(M_1, M_2, \cdots, M_n)$ can be recovered by decrypting the ciphertext $f(C_1, C_2, \cdots, C_n)$, the computation results obtained by the third party. In HECTor, homomorphic encryption enables relay nodes to mix encrypted packets at the bit level without knowing the keys.

B. Network Coding

Network coding was proposed primarily for improving the network throughput [19]. In the traditional networks, a node relays packets by simply forwarding the received packets. In network coding, a node mixes multiple received packets with coding and then forwards the combined packets. A coding vector will be used in mixing to determine how the received packets are combined. The destination nodes recover the original packets from the senders by decoding the combined packets.

In this paper, we use random liner network coding [20]. Random linear coding allows a node to generate random coding vectors locally and then mix received packets with the random coding vectors. As shown in Figure 1, the node generates a random vector $[a_{11}, a_{12}, a_{13}]$ from a Galois field and then combines the incoming packets P_1, P_2, and P_3 according to the random vector to form the combined packet $P_1' = a_{11}P_1 + a_{12}P_2 + a_{13}P_3$. In the matrix notation, the combined packets can be written as follows:

$$\mathbf{P'} = \mathbf{AP} \quad , \quad (2)$$

where $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$, $\mathbf{P_1} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$, and $\mathbf{P_1'} = \begin{bmatrix} P_1' \\ P_2' \\ P_3' \end{bmatrix}$. The combined packets P_1', P_2', and P_3' will be sent out with their coding vectors in the practical network coding [8]. The coding process continues on the next relay node and the coding vectors sent along with the combined packets are updated to include the locally-generated random

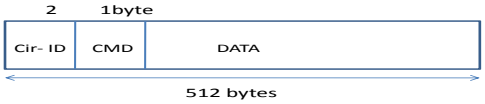


Fig. 2: Control Cell Format

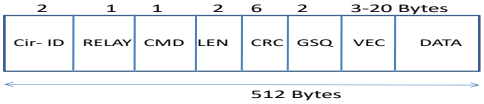


Fig. 3: Relay Cell Format

coding vectors. After receiving the combined packets along with the coding vectors, the destination nodes can recover the original packets by solving a linear system similar as in Equation 2 where $\mathbf{P}'$ contains the received packets and $\mathbf{A}$ contains the overall coding vectors updated by each relay node with locally-generated coding vectors.

V. Design

A. Overview

The main idea of the HECTor is to mix packets at the bit level instead of the packet level. The mixing at the bit level renders the traffic pattern at the packet level largely invisible and thus HECTor is immune to the existing packet-level traffic analysis attacks that are proven detrimental to current anonymity networks.

Network coding enables the mixing at the bit/symbol level. In HECTor, the random linear network coding [19] is used because it allows a node to generate random vectors locally. In comparison with the deterministic network coding which requires each node to use a pre-determined coding vector or a pre-determined encoding function, the random linear network coding eliminates the need of coordination among nodes. So the random linear network coding is more suitable for anonymous communications and it is used in HECTor.

As explained in Section III, we adopt the concept of onion routing [1], more specifically the layered encryption for its proven success in both previous anonymity researches and practice. But the layered encryption is not compatible with the bit-level mixing by default because: (a) The intermediate relay nodes can only decrypt one layer according to the onion routing [1], i.e., peel off the outer layer of the "onion". (b) Mixing layer-encrypted packets requires access to the plaintext, i.e., access to the core of the "onion" messages if traditional ciphers are in use. The incompatibility problem can be solved with the homomorphic encryption which enables a relay node to mix layer-encrypted packets without requiring the access to the core of an "onion", i.e., the plaintext message.

B. Cells

In HECTor, messages are sent in fixed-size cells to prevent traffic analysis attacks based on the packet size [21], [22]. The messages are TLS encrypted. In this paper, we set the cell size as 512 bytes. There are two types of cells: control cells and relay cells.

C. Control Cells

When a node receives a control cell it will act according to the command in the cell. The cell format is as show in the Figure 2. The Circuit identifier (*Cir-ID*) is a local value to identify a circuit between two neighboring nodes. The *CMD* field indicates the command carried in the cell. The value

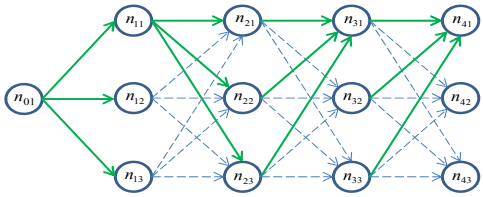


Fig. 4: Cascade Topology (Cell relay on solid links are described in Section V-F.)

of the *CMD* field indicates whether it is a control cell or a relay cell. The commands in a control cell can be *create*, *created*, *combine*, *combined* ,and *destroy*. Most of the control commands are used for cascade construction as described below.

D. Relay Cells

Relay cells are used to transfer data between senders and receivers. As shown in Figure 3, relay cells have additional header fields when compared to control cells. First relay cells are differentiated from control cells by the third byte. The following *CMD* byte after the relay byte is used to indicate relay commands designed for circuit construction. The next byte is used to indicate the length of payload data. *GSQ* denotes the generation sequence number to which the cell belongs. *VEC* denotes the coding vector used to form the relay cell. Both *GSQ* and *VEC* are used to random linear network coding for combining cells in the same generation.

E. Cascade Construction

A cascade topology as in AN.ON/JonDo [23] will be formed after cascade construction. A sample cascade topogy is shown in Figure 4. The number of nodes in the same stage of the cascade topology should be at least no less than the generation size gs, defined as the number of cells in the same generation sent from source nodes so that the exit nodes can have enough linearly independent cells to recover the original messages. The other goal of cascade construction is to establish keys between source nodes and HECTor nodes. The HECTor proxy running on Alice's computer, denoted as Node n_{01} in Figure 4, will share the key k_i with the nodes in the ith stage of the cascade. Due to the space limit, we leave the procedure of cascade construction to [24].

F. Relay Operation

We use the example network in Figure 4 to describe relay operation in HECTor: The HECTor proxy running on Alice's computer, denoted as Node n_{01}, first encrypts Alice's original message into message M_1 with the public key of an exit node (assuming Node n_{41} in this example without loss of generality), and then converts Alice's message M_1 into the layer-encrypted "onion" $O^{01\to 11} = E_{k_1}(E_{k_2}(E_{k_3}(E_{k_4}(M_1))))$ where $O^{x\to y}$ denotes a layer-encrypted packet on the link from

Node x to Node y[1], E denotes the homomorphic encryption, and k_i denotes the key shared between the HECTor proxy running on Alice's computer and the nodes in the ith stage of the cascade topology. Without loss of generality, we assume $O^{0j\to 11} = E_{k_1}(E_{k_2}(E_{k_3}(E_{k_4}(M_j))))$ for $j = 1, 2, 3$ where M_j denotes the message from the jth user, i.e., the jth HECTor proxy[2]. HECTor proxies will also send the same packets to Node n_{12} and Node n_{13}. In other words, $O^{0j\to 11} = O^{0j\to 12} = O^{0j\to 13}$, for $j = 1, 2, 3$. Please note that the links from HECTor proxies are TLS encrypted as well.

Alice's cell $O^{01\to 11}$ enters the HECTor network through Node n_{11}. With the key k_1 shared between HECTor proxies and nodes in the first layer, Node n_{11} is able to decrypt the "Onion" $O^{01\to 11}$ as follows:
$D_{k_1}(O^{01\to 11}) = D_{k_1}(E_{k_1}(E_{k_2}(E_{k_3}(E_{k_4}(M_1))))) = E_{k_2}(E_{k_3}(E_{k_4}(M_1)))$
where D_{k_1} denotes decryption with the key k_1. Similarly $O^{02\to 11}$ and $O^{03\to 11}$ are decrypted into $E_{k_2}(E_{k_3}(E_{k_4}(M_2)))$ and $E_{k_2}(E_{k_3}(E_{k_4}(M_3)))$ respectively. After the decryption, Node n_{11} generates a local random coding vector $\mathbf{A_{11}} = [a_{11}, a_{12}, a_{13}]$ and combines the decrypted packets into one packet with the random coding vector as follows:

$$
\begin{aligned}
& O^{11\to 21} \\
= \; & a_{11}E_{k_2}(E_{k_3}(E_{k_4}(M_1))) + a_{12}E_{k_2}(E_{k_3}(E_{k_4}(M_2))) \\
& + a_{13}E_{k_2}(E_{k_3}(E_{k_4}(M_3))) && (3)\\
= \; & E_{k_2}(a_{11}E_{k_3}(E_{k_4}(M_1)) + a_{12}E_{k_3}(E_{k_4}(M_2)) \\
& + a_{13}E_{k_3}(E_{k_4}(M_3))) && (4)\\
= \; & E_{k_2}(E_{k_3}(a_{11}E_{K_4}(M_1) + a_{12}E_{K_4}(M_2) \\
& + a_{13}E_{K_4}(M_3))) && (5)\\
= \; & E_{k_2}(E_{k_3}(E_{k_4}(a_{11}M_1 + a_{12}M_2 + a_{13}M_3))) && (6)\\
= \; & E_{k_2}(E_{k_3}(E_{k_4}(\mathbf{A_{11}}\mathbf{M}^\intercal))) && (7)\\
= \; & E_{k_2}(E_{k_3}(E_{k_4}(\mathbf{A_g^{11\to 21}}\mathbf{M}^\intercal))) && (8)
\end{aligned}
$$

where $\mathbf{M} = [M_1, M_2, M_3]$ and $[]^\intercal$ denotes transposition. The derivation from (3) to (6) is based on the definition of homomorphic encryption in (1). From (7) to (8), we replace the local random coding vector $\mathbf{A_{11}}$ with $\mathbf{A_g^{11\to 21}}$ for consistency with the following derivation, where $\mathbf{A_g^{x\to y}}$ denotes the global coding vector carried in a cell between Node x and Node y. For $O^{11\to 21}$, $\mathbf{A_g^{11\to 21}} = \mathbf{A_{11}}$ because Node n_{11} combines the "onions" from the HECTor proxies for the first time.

Node n_{21} processes the incoming packets in the same way as Node n_{11}: When Node n_{21} receives $O^{11\to 21}$, Node n_{21} decrypts $O^{11\to 21}$ into $E_{k_3}(E_{k_4}(\mathbf{A_g^{11\to 21}}\mathbf{M}^\intercal))$ as Node n_{21} shares the key k_2 with HECTor proxies. Similarly Node n_{21} decrypts $O^{12\to 21}$ and $O^{13\to 21}$ into $E_{k_3}(E_{k_4}(\mathbf{A_g^{12\to 21}}\mathbf{M}^\intercal))$ and $E_{k_3}(E_{k_4}(\mathbf{A_g^{13\to 21}}\mathbf{M}^\intercal))$ respectively, where $\mathbf{A_g^{12\to 21}}$ and $\mathbf{A_g^{13\to 21}}$ denote the global coding vectors used to the form the corresponding cells respectively. Node n_{21} will also generate a local random coding vector denoted as $\mathbf{A_{21}}$, and combine decrypted packets with the random coding vector to form $O^{21\to 31}$ as follows:

$$
\begin{aligned}
O^{21\to 31} &= \mathbf{A_{21}} \begin{pmatrix} E_{k_3}(E_{k_4}(\mathbf{A_g^{11\to 21}}\mathbf{M}^\intercal)) \\ E_{k_3}(E_{k_4}(\mathbf{A_g^{12\to 21}}\mathbf{M}^\intercal)) \\ E_{k_3}(E_{k_4}(\mathbf{A_g^{13\to 21}}\mathbf{M}^\intercal)) \end{pmatrix} \\
&= E_{k_3}(E_{k_4}(\mathbf{A_{21}} \begin{pmatrix} \mathbf{A_g^{11\to 21}} \\ \mathbf{A_g^{12\to 21}} \\ \mathbf{A_g^{13\to 21}} \end{pmatrix} \mathbf{M}^\intercal))) && (9)\\
&= E_{k_3}(E_{k_4}(\mathbf{A_g^{21\to 31}}\mathbf{M}^\intercal) && (10)
\end{aligned}
$$

where $\mathbf{A_g^{21\to 31}}$ denotes the global coding vector used to form $O^{21\to 31}$ from $\mathbf{M}$ and $\mathbf{A_g^{21\to 31}}$ is a global coding vector resulting from local random coding vectors generated by the nodes in the previous stages.

Node n_{31} will process incoming "onions" in the same way as Node n_{21}. To save space, we skip the description of the processing by Node n_{31}.

Node n_{41}, as the last HECTor node in the path, receives $O^{31\to 41} = E_{k_4}(\mathbf{A_g^{31\to 41}}\mathbf{M}^\intercal)$, $O^{32\to 41} = E_{k_4}(\mathbf{A_g^{32\to 41}}\mathbf{M}^\intercal)$, and $O^{33\to 41} = E_{k_4}(\mathbf{A_g^{33\to 41}}\mathbf{M}^\intercal)$. Then Node n_{41} decrypts them into $\mathbf{A_g^{31\to 41}}\mathbf{M}^\intercal$, $\mathbf{A_g^{32\to 41}}\mathbf{M}^\intercal$, and $\mathbf{A_g^{33\to 41}}\mathbf{M}^\intercal$ respectively. Given the global coding vectors $\mathbf{A_g^{31\to 41}}$, $\mathbf{A_g^{32\to 41}}$, and $\mathbf{A_g^{33\to 41}}$ sent with the packets and the decrypted packets $\mathbf{A_g^{31\to 41}}\mathbf{M}^\intercal$, $\mathbf{A_g^{32\to 41}}\mathbf{M}^\intercal$, and $\mathbf{A_g^{33\to 41}}\mathbf{M}^\intercal$, Node n_{41} can recover the messages $\mathbf{M} = [M_1, M_2, M_3]$ if the global coding vectors are linearly independent. In general, to recover the messages $\mathbf{M}$, Node n_{41} needs at least gs linearly independent cells, i.e., cells carrying at least gs linearly independent global coding vectors, where gs is the generation size, i.e., the number of cells sent in one generation from source nodes.

Node n_{41} will then try to decrypt each recovered message with its own private key. If the decryption is successful, then the decrypted message will be forwarded to the destination. In this example, Node n_{41} will be able to decrypt message M_1, which is encrypted with Node n_{41}'s public key and then the decrypted message will be forwarded to its destination.

VI. Theoretical Analysis

In this section, we theoretically analyze the decoding probability. As described in the previous section, an exit HECTor node needs to receive at least gs linearly independent cells of one generation to recover messages from HECTor proxies. The decoding probability is defined as the probability that messages can be recovered by an exit HECTor node. In other words, the decoding probability is the probability of receiving at least gs linearly independent cells by exit nodes. Without loss of generality, we assume the coefficients of random coding vectors generated by each HECTor node are randomly chosen from GF(2) field in this paper.

Theorem 1: With l nodes in each stage of the cascade topology, if the generation size is gs and random coding vectors are generated from GF(2) field, the decoding probability at an exit HECTor node can be derived as follows: $p_d = \frac{\binom{l}{gs}}{2^{l-gs}} \prod_{i=0}^{gs-1} (1 - \frac{2^i}{2^{gs}})$.
Due to the space limit, we leave the proof in [24].

[1]In this paper, we use $O_p^{x\to y}$ to denote a layer-encrypted packet on the link from Node x to Node y and p denotes the generation number in network coding. In this subsection, we skip the subscript for simplicity as the packets are from the same generation.

[2]The users of the same HECTor network share the same key with nodes in each layer for the layer encryption after cascade construction.

VII. Performance Evaluation

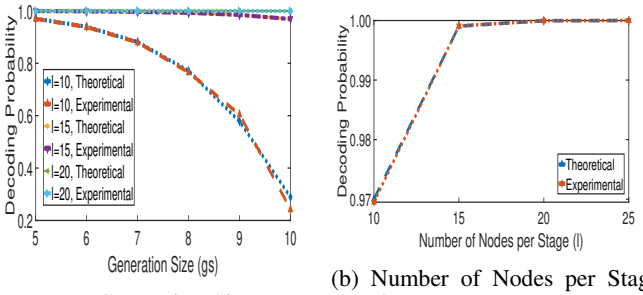
We conducted extensive experiments to validate HECTor in ns-2 [25]. We vary the number of stages in the cascade topology shown in Figure 4 and the number of nodes in each stage for different experiments. All the links in the network are of 10Mbps and the delay on each link is 10 ms unless otherwise specified. All the links between the intermediate HECTor nodes have on/off cross traffic with burst rate 5Mbps, average burst length 500ms, and average idle time 500ms.

A. Performance Metrics

We evaluate anonymity and usability of HECTor networks with the size of anonymity set based on the information theoretical metrics proposed in [26], [27] and throughput respectively. The intermediate performance metric used in performance evaluation is decoding probability, measuring how easy to send a message through HECTor networks.

Size of Anonymity Set: One of the major design goals of HECTor is to protect communication anonymity, i.e., to counter traffic analysis attacks that are proven effective in compromising anonymity networks. Since HECTor cells are of the same size, we focus on HECTor's resistance to timing-based traffic analysis attacks. As most effective timing-based traffic analysis attacks [2]–[4] are correlating traffic flows to compromise anonymous communications, we evaluate HECTor's resistance to traffic analysis attacks by correlating traffic from Alice and traffic received by HECTor exit nodes. The correlation can be based on cross-correlation [2], [3] or mutual information [4].

For correlation, we first generate the cell count vector $X = [x_1, x_2, \cdots, x_v]$ from Alice's traffic, where x_j is the number of cells in the jth sample interval and v is the length of the vector. We set the length of the sample interval to be 10 ms as in [4] and similar results are obtained with different length of sample intervals. In the same way, we generate the cell counter vector according to the traffic received by an exit node $Y = [y_1, y_2, \cdots, y_v]$. The cross correlation between Alice's traffic and traffic received by a HECTor exit node can be calculated as follows: $\rho = \frac{\sum_{i=1}^{v}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{v}(x_i-\overline{x})^2 \sum_{i=1}^{v}(y_i-\overline{y})^2}}$ where $\overline{x}$ and $\overline{y}$ denote the sample means of X and Y respectively. The mutual information between Alice's traffic and traffic received by the HECTor exit node can be calculated as follows: $I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\log(\frac{p(x,y)}{p(x)p(y)})$ where $p(x)$ and $p(y)$ are marginal probability distribution functions of X and Y respectively and $p(x,y)$ is the joint probability distribution function of X and Y. If the correlation, calculated as cross-correlation or mutual information, between Alice's traffic and the traffic received by the actual exit HECTor node is higher than any other possible exit HECTor nodes, then the attack is successful. Without loss of generality, we assume the first actual HECTor exit node is the actual exit node for Alice's traffic and we denote the probability of a successful attack as p_1. The probability p_1 is calculated as the ratio between the number of successful attacks and the total number of attacks.



(a) Generation Size

(b) Number of Nodes per Stage (gs=5)

Fig. 5: Decoding Probability

The probability of an unsuccessful attack that detects the ith HECTor exit node as the actual exit node is denoted as p_i where $i \neq 1$. Obviously $\sum_{i=1}^{l} p_i = 1$ where l denotes the number of nodes in each stage. The size of anonymity set, denoted as A, is defined based on the information theoretical metrics proposed in [26], [27] as follows: $A = -\sum_{i=1}^{l} p_i \log_2(p_i)$.

Decoding Probability: Since the coding vectors are randomly generated, it is possible that the cells received by a HECTor exit node are linear dependent and then the original message may not be recovered. The decoding probability measures the possibility of recovering original messages at an exit HECTor node. We compare the experiment results on decoding probability with the theoretical results derived in the previous section.

B. Decoding Probability

In this set of experiments, we focus on decoding probability with different HECTor configurations.

Figure 5 shows decoding probability with the different generation sizes and different numbers of nodes in each stage of the cascade topology. In this set of experiments, there are four stages in the cascade topology.

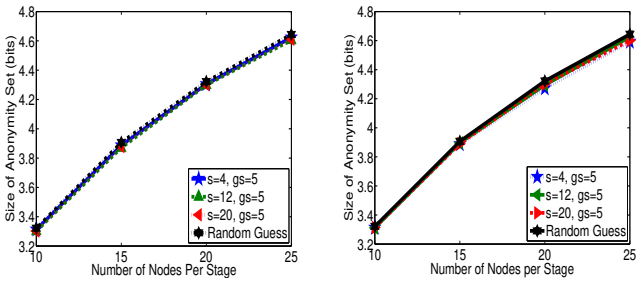
In Figure 5a, we can observe that when the generation size increases, the decoding probability decreases. The observation is consistent with intuition: When the generation size increases, more linearly independent cells are needed for successful decoding and in turn the decoding probability decreases. We can also observe that the decoding probability is close to 100%.

In the next set of experiments, we fix the generation size to five. In Figure 5b, we can observe that the decoding probability increases when l, the number of the nodes per stage, increases. It is also consistent with intuition: When l increases, an exit HECTor node will receive more cells. So the HECTor node will more likely have enough linearly independent cells for successful decoding.

In both Figure 5a and 5b, we can observe that the curves from experiment results are very close to the curves from our theoretical results.

C. Resistance to Traffic Analysis Attacks

Figure 6 shows the effect of the number of nodes per stage on resistance to traffic analysis attacks. Figure 6a and Figure 6b shows the effect when traffic correlation is through mutual



(a) Mutual Information (b) Cross Correlation

Fig. 6: Effect of the Number of Nodes per Stage on Resistance to Traffic Analysis Attacks

information and cross correlation respectively. In both Figure 6a and Figure 6b, the random guess curves are generated by assuming the HECTor exit nodes to have the same probability to be the actual exit node of Alice's traffic. In other words, $p_i = \frac{1}{l}$ where p_i denotes the probability that the ith exit node is determined as the exit node for Alice's traffic and l denotes the number of nodes per layer. Then the size of anonymity set can be derived as follows: $A = -\sum_{i=1}^{l} p_i \log_2(p_i) = -\sum_{i=1}^{l} \frac{1}{l} \log_2(\frac{1}{l}) = \log_2(l)$.

From both figures, we can observe that the curves of experiments results are very close to the corresponding curves of random guess. It indicates that HECTor is highly effective in anonymity protection. We also observe that the size of anonymity set increases with the number of nodes per stage. It is consistent to our intuition as the number of the candidate exit nodes increases with the size of the anonymity set.

VIII. Conclusion

In this paper we propose Homomorphic EnCrypTion enabled Onion Routing (HECTor) for anonymous communication. HECTor mixes traffic at the bit level. Homomorphic encryption enables layered encryption in HECTor. We theoretically analyze the performance of HECTor. Our extensive experiments on HECTor show that HECTor can effectively mitigate existing traffic analysis attacks with high throughput.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proc. of the 13th USENIX Security Symposium*, San Diego, CA, August 2004, pp. 303–320.

[2] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 183–195.

[3] L. Molgedey and H. G. Schuster, "Separation of a mixture of independent signals using time delayed correlations," *Physical Review Letters*, vol. 72, no. 23, pp. 3634–3637, June 1994.

[4] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "Correlation-based traffic analysis attacks on anonymity networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 954–967, Jul. 2010.

[5] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Trans. Inf. Syst. Secur.*, vol. 1, no. 1, pp. 66–92, 1998.

[6] C. Díaz and A. Serjantov, "Generalising mixes," in *Privacy Enhancing Technologies*, R. Dingledine, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 18–31.

[7] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing for anonymous and private internet connections," *Communications of the ACM (USA)*, vol. 42, no. 2, pp. 39–41, 1999. [Online]. Available: citeseer.ist.psu.edu/goldschlag99onion.html

[8] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, no. 1, 2003, pp. 40–49.

[9] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[10] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, Springer-Verlag, LNCS 2009. Springer-Verlag, LNCS 2009, July 2000, p. 96–114.

[11] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2004)*, ser. LNCS, vol. 3424, Toronto, Canada, May 2004, pp. 35–50.

[12] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, ser. LNCS, vol. 3424, May 2004.

[13] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[14] Y. Zhu and R. Bettati, "Unmixing mix traffic," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2005)*, Springer Berlin / Heidelberg. Springer Berlin / Heidelberg, May 2005, p. 110–127. [Online]. Available: http://www.springerlink.com/content/l5110366246k5003/

[15] O. Trushina, "On the anonymity of physical-layer network coding against wiretapping," in *2016 XV International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY)*, Sept 2016, pp. 158–161.

[16] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation, Academia Press*, pp. 169–179, 1978.

[17] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009, crypto.stanford.edu/craig.

[18] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 24–43.

[19] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *Information Theory, IEEE Transactions on*, vol. 46, no. 4, pp. 1204 –1216, jul 2000.

[20] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Information Theory, 2003. Proceedings. IEEE International Symposium on*, june-4 july 2003, p. 442.

[21] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proc. of the 2002 IEEE Symposium on Security and Privacy*. Oakland, CA, USA: IEEE Computer Society, 2002, pp. 19–30.

[22] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," in *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 35–49.

[23] O. Berthold, H. Federrath, and S. Köpsell, *Web MIXes: A System for Anonymous and Unobservable Internet Access*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 115–129.

[24] S. Gumudavally, "A coding enabled anonymity network," Master's thesis, Cleveland State University.

[25] S. McCanne and S. Floyd, "'The network simulator - ns-2'," Available: http://www.isi.edu/nsnam/ns/.

[26] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2002)*, R. Dingledine and P. Syverson, Eds. San Francisco, CA: Springer-Verlag, LNCS 2482, April 2002, pp. 41–53.

[27] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2002)*, San Francisco, CA, April 2002, pp. 54–68.