Enhancing Traffic Analysis Resistance for Tor Hidden Services with Multipath Routing

Lei Yang and Fengjun Li(✉)

The University of Kansas, Lawrence, KS 66045, USA
{lei.yang,fli}@ku.edu

Abstract. Hidden service is a very important feature of Tor, which supports server operators to provide a variety of Internet services without revealing their locations. A large number of users rely on Tor hidden services to protect their anonymity. Around 30,000 servers are running hidden services every day [21]. However, hidden services are particularly vulnerable to traffic analysis attacks especially when the entry guard of a hidden server is compromised by an adversary. In this paper, we propose a multipath routing scheme for Tor hidden servers (*m*TorHS) to defend against traffic analysis attacks. By transferring data through multiple circuits between the hidden server and a special server rendezvous point (SRP), *m*TorHS is able to exploit flow splitting and flow merging to eliminate inter-cell correlations of the original flow. Experiments on the Shadow simulator [11] show that our scheme can effectively mitigate the risk of traffic analysis even when robust watermarking techniques are used.

Keywords: Tor · Hidden services · Anonymity network · Privacy · Multipath routing · Watermarking attack

1 Introduction

To address people's needs for privacy, many low-latency anonymity systems have been proposed to provide anonymity for Internet communications. Among them Tor [5] is the most popular and widely deployed low-latency anonymous communication system today, providing anonymity to millions of users on a daily basis [20]. One major reason that contributes to the success of Tor is its comprehensive anonymous services, which provide three types of anonymity [17], i.e., *sender anonymity*, *receiver anonymity* and *sender-receiver unlinkability*. In particular, Tor allows general users to access Internet sites without disclosing their actual identities to the destination and prevents adversaries from linking two communicating parties (i.e., sender anonymity and unlinkability for general users). Besides, Tor also allows server operators to hide their locations while providing a variety of Internet services via so-called *Tor hidden services*. This is a very appealing feature that makes Tor stand out. Other popular low-latency anonymity systems such as Anonymizer [10] and Java Anon Proxy (JAP) [3]

B. Thuraisingham et al. (Eds.): SecureComm 2015, LNICST 164, pp. 367–384, 2015.
DOI: 10.1007/978-3-319-28865-9_20

do not support such hidden service, since it is out of the scope of their initial designs. Anonymous publishing is of great importance especially for people in countries with strict censorship, therefore, a large number of users with strong anonymity needs deploy their services such as SSH, instant messaging and web servers on the Tor network for its practical support to location-hidden services and low latency. According to the statistics of Tor Project [21], around 30,000 hidden servers are active daily in the Tor network.

However, according to recent studies Tor hidden services are still under the risk of de-anonymization due to specialized traffic analysis attacks [16,4]. It is argued that the current Tor design is vulnerable to traffic analysis attacks if the adversary can monitor a user's traffic entering and leaving the anonymous network at both sender and receiver ends. Since the malicious client always resides at one end of the anonymous path, she can successfully perform the traffic analysis attack if she is able to observe the traffic at the hidden server end. Øverlier et al. proposed the first documented attack against Tor hidden services by exploiting traffic analysis techniques. They experimentally verified that a hidden server can be located within a short period of time if the adversary is able to control one Tor (or preferably two) router(s) [16]. Biryukov et al. also confirmed the practicality of traffic analysis attacks by conducting an opportunistic de-anonymization attack to Tor hidden services [4]. The effectiveness of such attacks is mainly caused by the low latency in anonymized paths, which unwillingly preserves the inter-cell timing correlation between the original flow and the anonymized flow. The adversary can exploit traffic analysis techniques to correlate common patterns between the original flow and the anonymized flow to infer identities and relations of the communicating parties. Therefore, the key to mitigating the threats of traffic analysis attacks is to reduce the timing correlation between cells. Dummy traffic is considered as an effective countermeasure to obscure the timing features of the original flow [18]. However, due to the high cost introduced by dummy traffic, it is not a practical solution for the already heavily loaded Tor network.

In this paper, we propose a multipath routing scheme for Tor hidden services (mTorHS) to defend against traffic analysis attacks. Our scheme routes data cells between the rendezvous point and the hidden server through multiple circuits, which exploits flow splitting and flow merging functionalities of multipath routing to remove identifiable patterns of the original flow. Through experiments on the Shadow simulator [11], we show that mTorHS is resistant to traffic analysis, even when robust watermarking-based techniques are employed. In addition, by integrating multi-flow detection scheme [13] into mTorHS, our scheme is able to combine multiple watermarked flows to detect the presence of watermarks, if they have not been completed destroyed by multipath routing.

Because a large number of abbreviations are used in this paper, we summarize the notions in Table 1. The remainder of this paper is organized as follows. After introducing the background of Tor hidden services and two representative traffic analysis attacks against Tor hidden services in Section 2, we present the threat model in Section 3. Then, we elaborate the detailed design of our multipath Tor

Table 1. Definitions of abbreviations used in this paper.

Abbreviation	Term	Abbreviation	Term
HS	Hidden server	SRP	Server's rendezvous point
Alice	Malicious client	RC	Rendezvous cookie
RP	Rendezvous point	DH	Diffie-Hellman
OR	Onion router	IP	Introduction point

hidden services in Section 4. In Section 5, we experimentally evaluate the effectiveness of *m*TorHS against a very robust watermarking-based attack. Finally, we review the related work in Section 6 and conclude this paper in Section 7.

2 Background

2.1 Tor

The onion-routing-based Tor network is an overlay network contributed by volunteers running Onion Routers (ORs). A client selects three routers by default to establish a circuit to the destination that he wants to access. Then, he packs them into 512-byte cells, encrypts data packets in layers and sends data cells through the circuit. Each router along the circuit peels off one layer of encryption and forwards the cell to the next router until it reaches the last relay (known as "exit"), which further forwards the data to the original destination. Each hop only knows who has sent the data (predecessor) and to whom it is relaying (successor) due to the layered encryption. A router processes the cells that are addressed to itself following the command in the cell, otherwise it simply relays the irrelevant cell to the next hop.

2.2 Tor Hidden Services

The Tor hidden services proposed in [5] use rendezvous points(RPs) to support hidden TCP-based services, such as web servers and instant messaging servers, without revealing real IP addresses of hidden servers. Figure 1 illustrates basic components of Tor hidden services: (1) To make a service reachable, the hidden server (HS) selects several routers at random as introduction points (IPs) and builds circuits to them. IPs wait for connections on behalf of the hidden server. (2) HS then uploads its service descriptor to the hidden service directory (HSDir). The descriptor containing its public key and a set of introduction points signed by the private key. After this step, HS is ready to accept connections from clients. (3) To connect to the hidden service, we assume a client (Alice) learns about HS's onion address out of band. Then, Alice contacts HSDir and retrieves the service descriptor of HS using this onion address. (4) After getting the set of introduction points and HS's public key from the service descriptor, Alice randomly selects a router as the rendezvous point (RP) by assigning it a rendezvous cookie (RC) which is a one-time secret, and builds a circuit to it (i.e.,

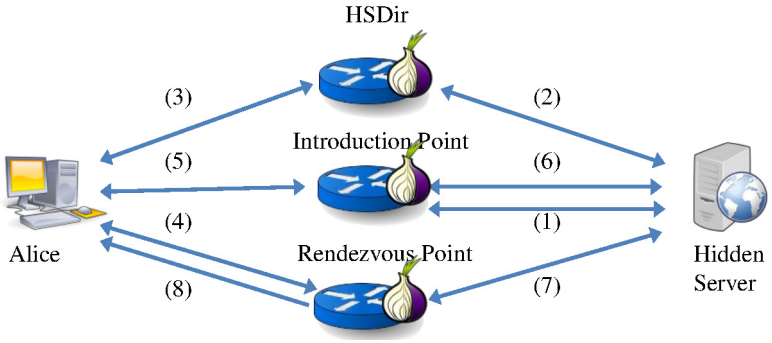


Fig. 1. Tor hidden services architecture

client circuit). (5) After that, Alice sends an introduce message to one of the introduction points and (6) asks the IP to forward it to HS. The message containing the rendezvous cookie, RP address and the first part of a Diffie-Hellman (DH) handshake is encrypted by HS's public key. (7) After decrypting the introduce message, HS establishes a new circuit to Alice's RP (i.e., hidden server circuit), and sends a rendezvous cell with RC and the second part of DH handshake. (8) RP then relays the rendezvous cell to Alice. After verifying RC and generating the end-to-end session key, Alice and HS start communicating with each other through RP.

Because every connection (depicted by solid blue line in Figure 1) is a multihop Tor circuit, no one can learn the actual IP address of either end of the connection. It is worth noting that the complete path between Alice and HS generally consists of six routers in two circuits as shown in Figure 2: among them, three routers including the rendezvous point are selected by Alice, and the other three are chosen by HS. It was intuitively expected that network delay jitter and flow mixing introduced by the six-hop path will make the original flow indistinguishable from other flows, so that the adversary can neither correlate the communication between Alice and HS nor identify the real IP address of HS.

2.3 Traffic Analysis Attack against Hidden Services

It is recognized that Tor hidden service is vulnerable to traffic analysis attacks. In general, traffic analysis attacks can be classified into two categories: *passive* traffic analysis and *active* traffic analysis. Passive traffic analysis correlates the sender's outgoing traffic with the receiver's incoming traffic by comparing the traffic features, such as packet timings and counts. To launch a successful passive traffic analysis, the adversary needs to monitor the traffic for a long time to obtain a reliable traffic pattern. The biggest advantage of passive traffic analysis is its stealth, but it is time-consuming and less accurate compared with the active attacks. To improve the accuracy and reduce the cost, many active traffic analysis techniques have been proposed to generate traffic with a special pattern at one end of the communication path and identify it at the other end.

The security of Tor hidden services was first challenged by Øverlier et al. [16]. They experimentally attacked an early version of hidden services in which the entry guard protection mechanism has not yet been implemented in Tor. In response to a client request, HS will randomly select three routers to build a circuit to RP. Assume a malicious client controls a set of routers in the Tor network. By establishing a large number of connections to HS, she can eventually force HS to choose an entry router (i.e., the fist hop of a circuit) that she controls. Then by requesting files of different sizes at different time from HS, the attacker can generate a special traffic signature and exploit simple traffic analysis techniques (e.g., packet counting combined with timing information) at the malicious entry node and the RP to correlate flows with the same traffic pattern.

Another attack is proposed by Biryukov et al. [4]. They generated traffic with a special pattern and applied packet counting traffic analysis to identify flows with the injected pattern. For example, a malicious RP can send 50 *padding* cells and a *destroy* cell to HS after receiving the rendezvous cell in Step (7) (as shown in Figure 1). If the corresponding malicious entry guard observes 53 cells (including the destroy cell, 50 padding cells and 2 additional *extended* cells in circuit construction) going towards HS and 3 cells (including the rendezvous cell and 2 *extend* cells) leaving HS, the adversary can decide that this malicious guard node is chosen as the entry node by HS.

However, these two attacks also suffer drawbacks. Since Øverlier's attack was conducted in the early stage of Tor with much fewer routers, clients and hidden servers, at that time their generated traffic pattern was unique enough and hence can be preserved after going through the Tor network. Nevertheless, the current Tor with much more traffic will make this simple packet counting based analysis less effective. For Biryukov's attack, because special cells (i.e., *padding* cells) are used to generate a unique traffic signature, it may not be invisible to hidden server. Therefore, more advanced active watermarking-based traffic analysis techniques [23,14,9,8] are proposed, which can make the traffic analysis attacks targeting Tor hidden services more efficient and stealthy. They embed a specific traffic pattern to the victim's flow on the sender side by manipulating the timings of selected cells. The adversary breaks the anonymity guarantee if the watermark is uniquely identified on the receiver side. Compared to passive traffic analysis and other active traffic analysis techniques, watermarking is more robust to flow transformations such as dummy traffic, flow mixing, traffic padding and network jitter, so it is considered a more efficient and severe threat to Tor hidden services.

3 Overview of the Problem and the Threats

The attacks described in Section 2.3 show that the adversary can successfully correlate two communicating parties if she is able to observe the traffic at two ends of a Tor circuit. As shown in Figure 2, the anonymous path between a client and the hidden service server consists of 6 hops. Since a malicious client is always at one end of the path, she only needs to trick HS to choose a compromised

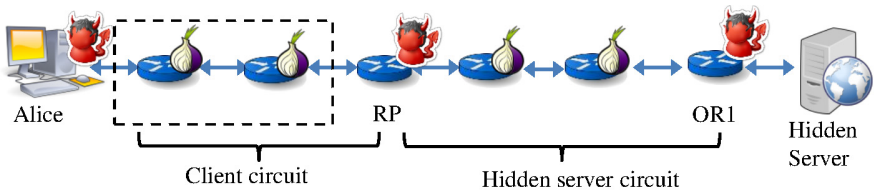


Fig. 2. Threat model in this paper where RP and OR1 are controlled by the adversary

router controlled by herself, i.e., OR1. Her success rate relies on the proportion of compromised routers in the Tor network. Therefore, this attack particularly threatens the hidden services. Moreover, the adversary can further select a node that she controls to be the rendezvous point and build a one-hop circuit to this RP. In this way, she can shorten the path to four and thus reduce the latency between herself and HS to help correlate the traffic pattern. To mitigate this threat, efforts can be made from two perspectives: (1) preventing an adversary from controlling both ends of a circuit to impede the occurrence of traffic analysis (2) reducing the success rate of traffic analysis even when both ends of a circuit are compromised.

The concept of "entry guards" [16] is introduced into the current Tor design to solve this problem following the first direction. Entry guards are a set of routers that are considered reliable by a Tor node to be the first relay of an anonymous path. By default, each user constructs its guard set of three routers, which will expire in 30 to 60 days. After that, the entry guards will be reselected. With entry guards, whenever the hidden server builds a circuit to the rendezvous point in response to a client's request, it will pick an entry guard from the set for its first hop instead of choosing a random router in the network. Since the entry guards are evaluated by several measures and considered reliable, they are less likely to be controlled by the adversary. As a result, the chance that an adversary controls both ends of a circuit is significantly reduced. However, it is unreliable that the security of hidden servers merely relies on the goodness of the entry guard set. Given enough time, a user will eventually select a malicious entry node into his guard set. Johnson et al. showed that for an adversary with moderate bandwidth capacity, it only takes 50 to 60 days to include a malicious router to a user's guard set [12]. As noted by Elahi et al., the design of entry guard is still an unclear research problem [6] and subtle parameter selection is required to achieve expected protection. Therefore, it is critical to develop protection mechanisms which are parallel to the entry-guard-based solution to enhance the resistance of Tor hidden services to traffic analysis attacks in case the guard set is compromised.

In this paper, we make efforts following the second direction to reduce the success rate of traffic analysis when the attacker has successfully controlled both ends of a Tor circuit. Our work can be applied in concert with entry guard protection mechanism. Figure 2 illustrates our threat model. We assume an adversary Alice pretends to be a client of the hidden server and tricks the hidden

server to select a node controlled by her (OR1) to be the entry node in the return anonymous path. Alice then selects another controlled router as the rendezvous point. We assume Alice can exploit any traffic analysis technique to passively observe or actively manipulate the traffic passing through OR1 and RP. The primary goal of the traffic analysis is to identify flows with the same traffic pattern at OR1 and RP to confirm that both malicious nodes are recruited in HS's circuit, from which she can learn the location of the hidden server.

4 Multipath Tor Hidden Services (*m*TorHS)

To prevent the adversary's RP and entry node from correctly identifying the traffic pattern, we present a multipath routing scheme for Tor hidden services. This scheme is based on the key insight that the traffic pattern observed or intentionally generated at the malicious entry guard (e.g., OR1) will be somewhat distorted by flow splitting and flow merging operations in multipath routing and by the multiple routes with different network dynamics. The architecture of *m*TorHS is illustrated in Figure 3. Different from the selection of rendezvous point in the current Tor implementation, the hidden server also selects its own "rendezvous point". To distinguish two rendezvous points, the one selected by the client is denoted as "CRP" and the one selected by hidden server is denoted as "SRP". In respond to a client's request, HS builds an anonymous tunnel consisting of m circuits, where m is a server specific parameter. HS then splits the original flow onto m subflows and attaches each subflow to a circuit in the tunnel. All m circuits will go through the same entry guard OR1 and merge at SRP, which further relays the merged flow towards the client. Next we will present the detailed process of connection initiation and data transmission.

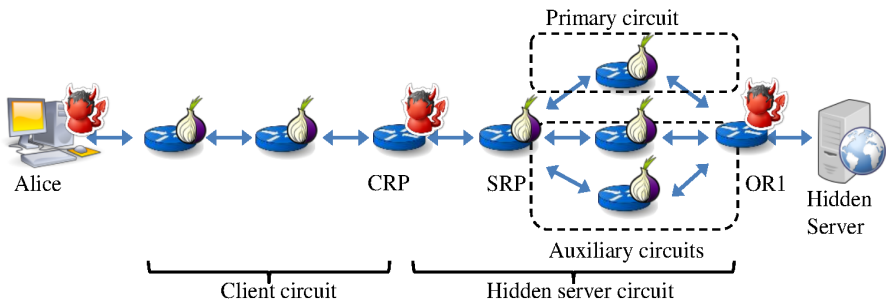


Fig. 3. *m*TorHS architecture

4.1 Connection Initialization at Client Side

For the client, the connection initialization remains the same as the current Tor hidden services (i.e., *step 3-6* in Section 2.2). More specifically, Alice first selects

a client rendezvous point (CRP) and constructs a rendezvous circuit to it by sending an establish_rendezvous request. Then, she builds an introduce circuit to one of HS's introduction points and sends an introduce request, which requests for the hidden service at HS and informs HS with CRP's address (i.e., fingerprint).

4.2 Multipath Tunnel Construction on Hidden Server Side

After receiving the introduce request, HS decrypts it with its private key and extracts the fingerprint of CRP, rendezvous cookie (RC) and the DH handshake message. Then, HS selects its own rendezvous point (SRP) and constructs a multipath tunnel to SRP, following the same approach described in [24].

SRP Selection. The selection of SRP is very critical. From Figure 3, we see that subflow merging occurs at SRP. Hence, even when multipath routing is adopted between SRP and HS, if the adversary controls SRP and OR1, she can observe traffic patterns before merging from both ends of each subflow and thus perform traffic analysis successfully. The adversary may follow the same strategy as described in [16] to trick HS into selecting a controlled node as SRP by continuously sending a large number of requests. If HS selects a new SRP for each received access request, it may eventually select one of the compromised router. Inspired by the entry guard idea, we propose "rendezvous guard" for SRP selection, which is a set of reliable routers selected by the hidden server. A hidden server initially selects three routers to compose its rendezvous guard set, each of which stays in the set for a random period between 30 and 60 days. Whenever HS builds a rendezvous circuit in response to the client request, it sticks to the same rendezvous guard set and randomly picks one router from it.

Tunnel Initialization. As discussed previously, mTorHS constructs a tunnel with multiple circuits to SRP instead of one anonymous circuit to CRP. In the original Tor hidden service design, the hidden server responds to an introduce request by establishing a four-hop anonymous circuit ending at CRP selected by the client. To ease the presentation, we denote this anonymous circuit as the primary circuit and the other $m-1$ anonymous circuits in the tunnel as the auxiliary circuits. As shown in Figure 3, all circuits merge at the SRP. Therefore, it is the third router for every three-hop anonymous circuit in the tunnel. Besides that, HS follows the default Tor path selection algorithm to select all other routers to form the tunnel. When the circuit is established, HS sends a multipath_m cell[1] along the primary circuit to SRP to request a multipath connection. In response, SRP generates a unique 32-bit tunnel identifier (*TID*) and incorporates it into the replied multipath_ack_m cell to indicate a successful multipath tunnel construction. With *TID*, HS adds each auxiliary circuit to the tunnel by sending a join_m request to SRP along the circuit. SRP acknowledges each successful joining with a joined_m message. Finally, when HS receives $m-1$ acknowledgments, a multipath tunnel is successfully constructed. Note that all

[1] To distinguish from the commands in current Tor, all the newly added commands in mTorHS will end with an m.

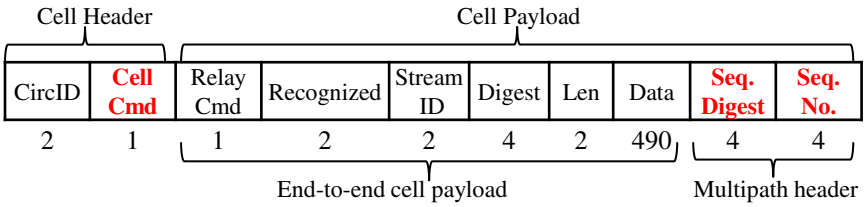


Fig. 4. *m*TorHS cell format

the cells are layered encrypted so only HS and SRP at two ends see *TID* and the newly added tunnel construction command. This prevents the entry and middle nodes of HS's circuit from linking *TID* with HS. In fact, they even do not know if they are involved in any tunnel construction.

Once the tunnel is established, HS follows the same process of the current hidden service protocol to extend path construction to CRP and the client. In particular, HS sends a rendezvous1 cell containing DH handshake message and rendezvous cookie (RC) to CRP along the primary circuit, which verifies RC and joins the client's circuit with the server's primary circuit. Then, CRP sends a rendezvous2 cell containing the DH handshake message to the client to finish the construction. Note that from CRP's view, it sees only the primary circuit connecting itself to SRP, and hence it has no idea about how many circuits are involved in the multipath tunnel between SRP and HS.

Tunnel Management. The hidden server can add new auxiliary circuits or tear down any existing circuit in the tunnel after it is established. In particular, a new auxiliary circuit can join the tunnel by sending a join_m command with the corresponding *TID*. To tear down a circuit, HS immediately stops sending on this circuit and informs SRP to drop it using a drop_m message. Note that the number of cells that have already been sent on this circuit (denoted as n_s) should also be passed to SRP to avoid packet loss. After receiving drop_m cell, SRP extracts n_s and replies with a dropped_m cell after it receives the remaining n_s cells. Finally, HS tears down the circuit when it receives the dropped_m message. Since each tunnel is constructed in response to a client's request, it will be closed after the request is completed. However, this will not result in the closure of all circuits in the tunnel, since the circuits may be reused for other purposes until it gets "dirty" - after its lifetime exceeding 10 minutes and no streams on it, similar as in Tor circuit management.

4.3 Data Transmission between Client and HS

Once the connection is set up, the client and HS can communicate through the anonymous path consisting of the server's and the client's anonymous circuits joined at CRP. Data cells between SRP and HS can be routed through any circuit in the tunnel. To indicate a cell is a multipath cell used in *m*TorHS, we add a new cell command (i.e., MULTIPATH_CELL). HS is responsible for assigning data cells to circuits. Obviously, if HS schedules consecutive cells onto a same subflow,

it is highly likely that the traffic pattern inserted by the malicious guard on this subflow will be preserved in the merged flow and detected by the malicious CRP. To reduce the likelihood of inter-cell correlations, HS randomly assigns data cells to subflows with different capacities. As a result, a data cell from a fast circuit needs to wait at SRP for earlier cells arriving from slow subflows to be merged in an orderly manner. In this way, we utilize the network properties of different circuits to distort or destroy potential traffic patterns inserted by the malicious guard. This greatly reduces the likelihood of inter-cell correlation (we will explain this in Section 4.4).

Data Cell Format. Since the capacity of each circuit in the tunnel varies, different delays will be introduced to these subflows. Therefore, the data cells in different subflows may arrive at SRP out of order. To solve this issue, we modify the format of Tor data cell to incorporate a 4-byte *sequence digest* and a 4-byte *sequence number* for multipath data packets, as shown in Figure 4. Originally the 512-byte cell consists of a 3-byte cell header including a circuit identifier and a cell command for cell type, and 509-byte cell payload with a payload header and the payload data. We use 8 bytes of the cell payload as *multipath header* for cell reordering, where 4 bytes are used as sequence digest for integrity check and 4 bytes are used for sequence number. The multipath header is only used by SRP and HS to reorder data cells, and the remaining 501-byte end-to-end cell payload is used to carry the real payload data between client and HS.

Data Cell Encryption. In Tor anonymous routing, data cells are encrypted in layers with the shared session keys of the intermediate relays in the order of their relative positions in the anonymous path. Since the end-to-end cell payload and the multipath header are designed for the client and SRP respectively, HS needs to encrypt the two parts separately. The end-to-end cell payload should be encrypted in *five* layers with the inner-most layer encrypted by the end-to-end session key and the outer-most layer encrypted by the key of OR1, while the multipath header is only encrypted in *three* layers with the keys of SRP, the middle router OR2 and the entry router OR1, respectively. When an intermediate router receives a multipath cell, it applies its secret session key onto both end-to-end cell payload and the multipath header to unwrap one encryption layer. Consequently, at SRP the multipath header will be completely unwrapped and recognized by SRP for further processing, while the end-to-end cell payload is still encrypted and remains secure.

Data Cell Reordering. To merge multiple circuits in a tunnel, SRP orders the received cells from all circuits according to their sequence number and temporarily stores the out-of-order cells in a buffer. When SRP receives a multipath data cell from a subflow, it first decrypts the multipath header and generates a digest for the last four bytes of the multipath header using the symmetric key shared with HS. If it is the same as the received sequence digest, SRP verifies the sequence number is not tampered. If the sequence number of this cell is what SRP expects, it will be immediately forwarded to CRP, otherwise it will be stored and ordered according to the sequence number. The multipath header

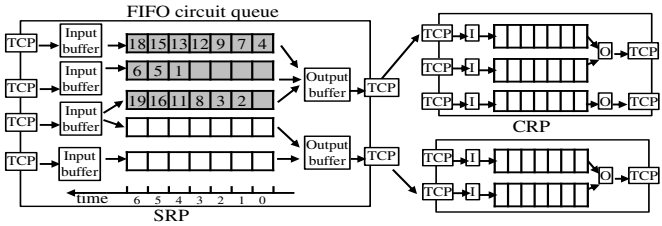


Fig. 5. Tor router queuing architecture [2]

field is only used for data cell reordering. Hence, after SRP reorders the cells and merges them into one output stream, this field becomes useless. To avoid unnecessary information leak, SRP will replace it with random bits. Similarly, when the client sends data to HS, the client reserves these eight bytes for SRP by padding them with random numbers.

4.4 Discussions

In Tor network, two Tor routers are connected over a TCP connection, which is multiplexed by several circuits. Due to the multiplexing of a TCP connection, flow mixing actually occurs at every router. However, we argue that the cell distribution is well preserved after flow mixing so that a maliciously inserted traffic pattern can still be observed by the attacker. First, let us explain the data cell processing at a Tor router. When a cell arrives from a TCP connection, it triggers the *connection read event* of libevent to first put it into the application-layer input buffer and then send to the corresponding circuit queue according to its circuit identifier. As shown in Figure 5, five different circuits arrive SRP from four TCP connections. Then, a *connection write event* will select a circuit based on pre-determined scheduling algorithms such as priority-scheduling [19] to pull cells from the circuit queue and send them to the output buffer. As a low-latency system, a router will send out cells in the circuit queue as fast as possible until the output buffer is full. Therefore, it is not surprising that cells from a same subflow will be outputted in a batch with inter-cell features well preserved.

In this paper, we propose a multipath routing approach that introduces an interdependent subflow mixing to SRP data cell processing. For example, in Figure 5 suppose the circuits in gray belong to the same tunnel. Each of them is associated to a subflow, which transfers a portion of data cells. Since the malicious guard has no clue about the flow membership of the subflows passing through it, it has to treat each subflow independently when inserting detectable traffic patterns. Oppositely, SRP will treat subflows of a same flow in a way that considers flow interdependency. In particular, when subflows are merged at SRP, cells from one subflow may be inserted into two cells that are adjacent in another subflow. This interpolation causes difficulty in pattern detection on the merged flow. Passive traffic analysis such as packet counting will fail. Moreover, due to differences in router bandwidth and other network dynamics, the

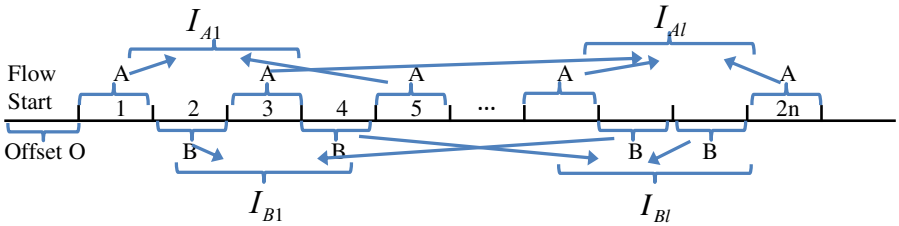
capacity of circuits vary [22,24]. Some cells with larger sequence numbers from a fast circuit (e.g., cell 4 on the first circuit in Figure 5) may arrive earlier than those with smaller sequence numbers but assigned to a slow circuit (e.g., cell 1 on the second circuit). The cells must be reordered at SRP. Consequently, the waiting time introduced by such reordering will distort or destroy the inter-cell timing correlations of a manipulated subflow and makes active traffic analysis less effective.

5 Experiment Evaluation

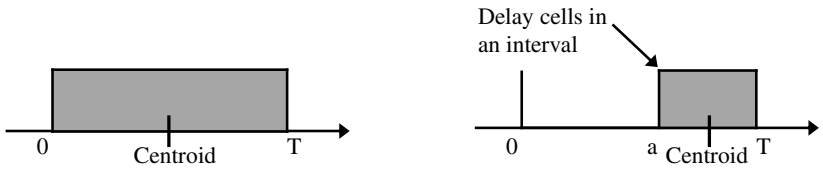
In this section we test the performance of *m*TorHS against a well-known active traffic analysis scheme, i.e., interval centroid-based watermarking (ICBW) [23], and evaluate the enhanced anonymity in our multipath hidden services. In particular, we conducted experiments with the Shadow simulator [11], which is an accurate, discrete event simulator running real Tor protocol over a simulated Internet topology. We implemented the multipath Tor router (please read [24] for details) and plugged it into the Shadow simulator to support multipath hidden services in a private Tor network. We also implemented an adversary node following the threat model described in Section 3, which first inserted watermarks to flows at the malicious entry guard OR1 using ICBW protocol, and then examined packets at the malicious client rendezvous point for expected traffic signatures.

5.1 Implementing ICBW Watermarking Scheme

To assess the resistance of the proposed scheme against traffic analysis, we implemented a state-of-the-art traffic watermarking scheme, the interval centroid-based watermarking (ICBW) protocol to attack low-latency anonymity systems [23]. The ICBW was verified on a leading commercial anonymizing service platform www.anonymizer.com as an effective attack. Here we briefly explain its working mechanism. As illustrated in Figure 6, ICBW embeds a watermark into a sufficiently long flow by intentionally changing the centroid of several randomly selected intervals. This scheme divides the duration of flow starting from an offset O into $2n$ intervals of equal length T. The centroid is then calculated by averaging each packet's relative arrival time to the start of its interval. The intervals are randomly grouped into two subsets ($\{I_{A_1}, ..., I_{A_l}\}$ and $\{I_{B_1}, ..., I_{B_l}\}$), each with l elements. Each element in set A and B contains r intervals for redundancy such that $n = rl$. The random grouping is illustrated in Figure 6a. To encode a watermarking bit 1 (or 0), two elements (I_{A_i} and I_{B_j}) of the set A and B are selected, respectively. The packets in all intervals of I_{A_i} (or I_{B_i}) will be delayed by a maximum value of a. Figure 6b illustrates the delaying, which actually changes the distribution of relative arrival time from $U(0, T)$ to $U(a, T)$ where $U(,)$ stands for uniform distribution. After encoding, the difference between the average centroids of I_A and I_B will be $\frac{a}{2}$ for watermark bit 1 and $-\frac{a}{2}$ for watermark bit 0. To decode, the decoder starts from the same offset O and checks the



a. Random grouping of intervals of a packet flow for encoding



b. Distribution of cell timing before and after delaying

Fig. 6. Random grouping intervals for a packet flow in ICBW [13,23]

existence of the watermark. Because each watermark bit is encoded by averaging the delays of packets that are randomly selected from many intervals, ICBW is very robust to network delay jitter and flow mixing along a circuit. We refer the readers to [23] for details.

For ICBW, we follow the suggested parameter setting: 32-bit watermarks are randomly generated and the redundancy r is set to 20. The interval length T and the maximum delay a are set to 500ms and 350ms, respectively. We use an offset $O = 10s$ to delay cells in the selected intervals according to the watermark bits at the malicious guard and meanwhile log the arrival time of each cell using the same offset at CRP. From the logged arrival time we compute the difference between the average centroid of I_A and I_B to derive a watermark bit 1 if it is closer to $\frac{a}{2}$ or 0 if it is closer to $-\frac{a}{2}$. Hamming distance, which is the number of mismatched bits, is computed between the derived watermark and the original watermark to evaluate how successful the watermarking attack is. Since the network delay is unknown, a set of different offsets are tested. The one that matches most to the inserted watermarks is chosen as the correct decoding offset to decode the watermark.

5.2 Implementing *m*TorHS

We implemented *m*TorHS on Tor v0.2.5.6-alpha. The construction of the client's circuit remains the same as in the current Tor hidden service design, but we change the implementation for the server circuit construction. As explained previously, the server's circuit consists of an entry guard, a middle relay, SRP and CRP. Since we assume the malicious client controls the guard node and CRP, we fix the selection of the two nodes in the implementation. The middle relay is randomly selected from the Tor router set and SRP is randomly selected from the "rendezvous guard set". For simplicity, in our current implementation we form

the rendezvous guard set with routers flagged as entry guard. This is because the concept of "rendezvous guard set" is derived from the idea of entry guard set to denote a set of routers trusted by the hidden server. In the future, we will develop selection criteria to assist the selection of reliable rendezvous guards.

Finally, we build a small private Tor network in the Shadow simulator with 50 Tor routers, 1 hidden server, 20 general HTTP servers, 1 malicious client and 100 general web clients to run our experiments. Among the 50 routers, two are configured as malicious CRP and OR1. This is a general case for evaluation. Obviously, the fewer the general clients in the network, the more likely the adversary identifies the hidden server. So, we choose an extreme setting for comparison, where the adversary is the only client in the network. As pointed out by Wang et al. in [23], the longer the flow, the more robust the watermark. To ensure a sufficiently long flow for successful watermarking, we let the client to request a 100MiB file at the hidden server under both settings.

5.3 Results

We perform the ICBW attack on the original Tor and the proposed mTorHS, where m is set to 2, 4, 6 and 8. To rule out random noise, we repeat the watermarking attack for ten times for each setting. The results shown below are the average results of ten experiments. Table 2 shows the comparison in terms of Hamming distance between Tor and mTorHS under different settings. A larger Hamming distance indicates that the anonymity system can better transform the original flow and prevent the traffic analysis. No matter in general cases or extreme cases, mTorHS can better obscure the embedded watermark in the victim's flow. The Hamming distance achieved on mTorHS is always larger than the maximum Hamming distance threshold (i.e., 8) [23]. When the Hamming distance exceeds the threshold, the adversary has less confidence to correlate the watermarked flow to the suspected flow. We note that with m increasing, the Hamming distance does not increase obviously. One reason might be that when we decode the watermark, we tried a set of different offsets and picked the minimum value.

Table 2. Comparison of Hamming distance between Tor and mTorHS with different m where each flow is encoded using different watermarks.

	Tor	mTorHS			
		m=2	4	6	8
General case	6	9	9	10	12
Extreme case	3	8	9	9	11

From the adversary's perspective, if she wants to circumvent the multipath routing scheme, she should use the same watermark for different flows. Since two circuits between OR1 and HS multiplex the common TCP connection, the

cells that HS sends out through two different subflows arrive at OR1 within the same coding interval of 500ms. If the adversary uses different watermarks to encode them, it is possible that one subflow is delayed while the other one not, which causes the distribution of the delayed subflow is squeezed to $U(a, T)$ while the other one is still $U(0, T)$. When SRP receives these cells from two subflows, SRP will merge them so that the distribution of the merged flow will be a uniform distribution $U(x, T)$ where $0 < x < a$ depending on which subflow the majority of the cells belong to. Therefore, when the malicious CRP receives the merged subflow, she cannot recover the correct centroid of this interval. To avoid this, the adversary ought to use the same watermark to encode all flows going through OR1 so that they will have a same distribution. When they are merged at SRP, the merged flow still preserves the distribution. Table 3 shows the results when the adversary embeds the same watermark to all flows. In order to verify this assumption without being influenced by general traffic, we perform this experiment in the extreme cases. As shown in Table 3, when the same number of subflows are used, the Hamming distance of the merged flow in cases where a same watermark is embedded is always smaller than the one when different watermarks are used.

Table 3. Comparison of Hamming distance between Tor and mTorHS with different m where all flows are encoded using the same watermark.

	Tor	mTorHS			
		m=2	4	6	8
Extreme case	3	4	7	6	7

However, once the adversary encodes multiple flows with the same watermark, her watermarking is vulnerable to the multi-flow attacks [13] (from the defending perspective, we call it multi-flow detection (MFD) in this paper.) The idea of MFD is that the MFD detector will aggregate all flows into a single flow after it collects a number of watermarked flows. This aggregation scheme in MFD is different from our subflow mixing, which overlaps the relative arrival time of each flow to the same start. If several abnormally long periods of silence (i.e., no packets for hundreds of milliseconds) are observed in the aggregated flow, the detector considers the presence of watermarking attack, extracts the watermarking keys and removes the watermarks from the observed flows. Since SRP merges multiple subflows, which is naturally compatible to MFD, we deploy the MFD detector at SRP. Figure 7 shows the aggregated arrival time of six flows with and without the presence of watermark. Compared to the aggregated unwatermarked flows, the silence of the victim's aggregated flow is more obvious and periodic. Once SRP recognizes the existence of a watermark with higher confidence, it can remove it by randomly delaying some cells on the suspicious flow.

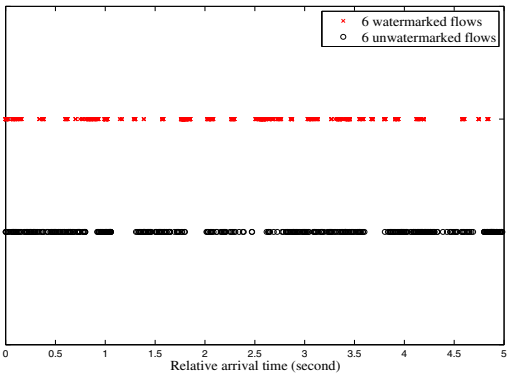


Fig. 7. Comparison of time pattern of the aggregated flow between watermarked flows and unwatermarked flows.

6 Related Work

Attacks to Hidden Services. In addition to the two traffic analysis attacks against Tor hidden services discussed in Section 2.3, clock skew based attacks are also proposed to break the anonymity of hidden servers. [15,25] found that the load changes on the victim's computer will result in temperature changes, which further cause the victim's clock to deviate from the real clock time and results in *clock skew*. Therefore, the adversary can periodically build many connections to the victim to generate a specific clock skew pattern. Meanwhile, she can measure the clock skew of a set of candidates and tries to detect a matched pattern.

Multipath Routing for Performance. The solution proposed in this work is based on multipath routing in the current Tor network. Other multipath routing schemes [1,24] have been proposed to improve the performance for general clients on Tor. Alsabah et al. [1] exploited multipath routing solutions to improve the performance for bridge and video streaming users, while Yang et al. [24] proposed a scheme to better utilize low-capacity routers to support bandwidth-intensive applications. We adopted the design of [24] in this work, but any multipath routing based approach can be applied in the proposed scheme.

Defense for Hidden Services. Entry guard proposed in [16] is an effective solution to protect hidden servers. Elahi et al. implement a framework to study Tor's entry guard design and empirically explores how the parameters affect the anonymity [6]. Besides, Hopper proposed a protection mechanism for Tor hidden services from another perspective – he explored the challenges in protecting Tor hidden services against botnet abuse [7].

7 Conclusion and Future Work

Tor hidden service is a very important tool to provide receiver anonymity to server operators, but it is vulnerable to traffic analysis attacks especially when the entry guard protection is broken. In this paper, we propose a multipath routing based scheme that exploits flow mixing and flow merging to distort or destroy inserted traffic patterns in a victim's flow. We believe this is an effective complement to the existing protection mechanism. Besides, since the multipath architecture is naturally compatible to detection mechanisms based on multiflows, it can be further integrated with multiflow detection protocols to detect the presence of watermarks. We experimentally verify the effectiveness of our scheme in defending one of the most robust watermarking schemes on the Shadow simulator.

The performance issues of Tor have been recognized as a big obstacle impeding Tor's further expansion, so it is important to evaluate the cost introduced by our proposed multipath routing architecture. Based on the findings of other multipath routing work on general Tor services [1,24], we believe that the multipath routing schemes usually improve the performance when a larger aggregated auxiliary circuits bandwidth contributes to the tunnel. However, the proposed multipath hidden services introduce more complexity to onion routers (e.g., separate encryption for end-to-end data and multipath header). In our future work, in addition to the evaluation on the Shadow simulator, we will also deploy multiple onion routers in the live Tor network to explore its impact on the performance of Tor hidden services. Besides, we will also test different scheduling schemes when HS splits traffic to multiple subflows, e.g., round-robin and proportional scheduling.

Acknowledgments. This work was partially supported by the National Science Foundation under Award EPS0903806, KU General Research Fund under Award GRF2301075, and KU Research Investment Council Strategic Initiative Grant under Award INS0073037.

References

1. AlSabah, M., Bauer, K., Elahi, T., Goldberg, I.: The path less travelled: overcoming Tor's bottlenecks with traffic splitting. In: De Cristofaro, E., Wright, M. (eds.) PETS 2013. LNCS, vol. 7981, pp. 143–163. Springer, Heidelberg (2013)
2. AlSabah, M., Bauer, K., Goldberg, I., Grunwald, D., McCoy, D., Savage, S., Voelker, G.M.: DefenestraTor: throwing out windows in Tor. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 134–154. Springer, Heidelberg (2011)
3. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: a system for anonymous and unobservable internet access. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 115–129. Springer, Heidelberg (2001)
4. Biryukov, A., Pustogarov, I., Weinmann, R.: Trawling for Tor hidden services: detection, measurement, deanonymization. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 80–94. IEEE (2013)
5. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proc. of the 13th USENIX Security Symposium (2004)

6. Elahi, T., Bauer, K., AlSabah, M., Dingledine, R., Goldberg, I.: Changing of the guards: a framework for understanding and improving entry guard selection in Tor. In: Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society, pp. 43–54. ACM (2012)
7. Hopper, N.: Challenges in protecting tor hidden services from botnet abuse. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 316–325. Springer, Heidelberg (2014)
8. Houmansadr, A., Borisov, N.: Swirl: a scalable watermark to detect correlated network flows. In: Proceedings of the Network and Distributed Security Symposium - NDSS 2011. Internet Society, February 2011
9. Houmansadr, A., Kiyavash, N., Borisov, N.: Rainbow: a robust and invisible non-blind watermark for network flows. In: Proceedings of the Network and Distributed Security Symposium - NDSS 2009. Internet Society, February 2009
10. Anonymizer Inc. Anonymizer. https://www.anonymizer.com/
11. Jansen, R., Hopper, N.: Shadow: running Tor in a box for accurate and efficient experimentation. In: Proceedings of the Network and Distributed System Security Symposium - NDSS 2012, February 2012
12. Johnson, A., Wacek, C., Jansen, R., Sherr, M., Syverson, P.: Users get routed: traffic correlation on Tor by realistic adversaries. In: Proceedings of the 20th ACM Conference on Computer and Communications Security (2013)
13. Kiyavash, N., Houmansadr, A., Borisov, N.: Multi-flow attacks against network flow watermarking schemes. In: USENIX Security Symposium (2008)
14. Ling, Z., Luo, J., Yu, W., Fu, X., Xuan, D., Jia, W.: A new cell counter based attack against Tor. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 578–589. ACM (2009)
15. Murdoch, S.J.: Hot or not: revealing hidden services by their clock skew. In: Proc. of the 13th ACM Conf. on Computer and Communications Security (2006)
16. Øverlier, L., Syverson, P.: Locating hidden servers. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy, May 2006
17. Pfitzmann, A., Waidner, M.: Networks without user observability. Computers & Security **6**(2), 158–166 (1987)
18. Shmatikov, V., Wang, M.-H.: Timing analysis in low-latency mix networks: attacks and defenses. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 18–33. Springer, Heidelberg (2006)
19. Tang, C., Goldberg, I.: An improved algorithm for Tor circuit scheduling. In: Proc. of the 2010 ACM Conf. on Computer and Communications Security (2010)
20. TorProject. Estimated Number of Clients in the Tor Network. https://metrics.torproject.org/clients-data.html
21. TorProject. Unique .onion Address. https://metrics.torproject.org/hidserv-dir-onions-seen.html
22. Wang, T., Bauer, K., Forero, C., Goldberg, I.: Congestion-aware path selection for Tor. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 98–113. Springer, Heidelberg (2012)
23. Wang, X., Chen, S., Jajodia, S.: Network flow watermarking attack on low-latency anonymous communication systems. In: IEEE Symposium on Security and Privacy, SP 2007, pp. 116–130. IEEE (2007)
24. Yang, L., Li, F.: mTor: a multipath Tor routing beyond bandwidth throttling. In: 2015 IEEE Conference on Communications and Network Security (CNS). IEEE (2015)
25. Zander, S., Murdoch, S.J.: An improved clock-skew measurement technique for revealing hidden services. In: USENIX Security Symposium (2008)