修士論文

Fingerprinting Attacks on the Tor Anonymity System and Defense Mechanisms

Tor匿名システムに対する指紋攻撃と防御手法

指導教官　　松浦幹太　准教授

東京大学大学院 情報理工学系研究科 電子情報学専攻

48-086445　　施 屹 **(Yi SHI)**

平成 22 年 8 月 18 日提出

Abstract

As the time advances, privacy is more and more concerned by Internet users. Encryption protected our information, but not all problems could be solved by it. Then we have the anonymity system. Anonymity system was first introduced by David Chaum and it served as the building block for anonymity, as a supplement of encryption. Among all kinds of anonymity systems, Tor is the most famous one and widely used among people and organizations. It is an implementation of the second generation Onion Routing and supports the anonymous transport of TCP streams over the public network. Then, it provides the foundation for a range of applications to communicate over public network without compromising their privacy. The characteristic of low latency makes it very suitable for general purpose tasks like web browsing.

With numerous researches about how to design anonymity system, there are also lots of studies about the attacks towards anonymity systems. Generally, these attacks require powerful assumptions to be implied. To the system designers, these assumptions will reduce their motivation to consider the new defense mechanisms against impractical attacks.

In this paper, first we systematically discuss the background knowledge of anonymity system through the timeline. Then we talk about motivation of attack researches and introduce several attacks by distinguishing them with their threat models. We try to give readers a rough picture of attack research in this field.

Then, we present a novel way to implement a fingerprinting attack against Onion Routing anonymity systems such as Tor. Our attack is a realistic threat in the sense that it can be mounted by nothing but controller of entrance routers; the required resource is very small. However, the conventional fingerprinting attack based on incoming traffic does not work straightforwardly against Tor due to its multiplex and quantized nature of traffic. By contrast, our novel attack can degrade this Tor's anonymity by a metric based on both incoming and outgoing packets. In addition, our method keeps the fingerprinting attack's advantage of being realistic in terms of the required small resource.

Based on the central idea, we also extend our idea in two ways - both the threat model and attack method itself. By these additional researches, we have showed the potential of our idea and hope we could encourage future research on this aspect.

About the evaluation, we try to enhance the reader's understand about the effec-

tiveness of our method by discussing them in a comprehensive manner: experimentally and theoretically. Experiments about extensions are also given in the following sections. In order to enhance further studies and show the significance of our idea, we also discuss general defense ideas and defense mechanism of dummy packets, what we recommend to imply in the future low-latency anonymity systems.

Contents

Chapter 1 Introduction

1.1 Anonymity System

1.1.1 The Motivation of Anonymity System

The Internet brings us convenience, but also hurts our anonymity. With some tools, it is no difficult for any attacker to eavesdrop activities of other users. Individuals and organizations sometimes need anonymity on the Internet. People want to surf webpages, make online shopping, and send email without exposing their identities and activity patterns to others. Encryption solved some parts of this problem, but not everything. It can hide the communication contents such as data payloads, but it can do nothing with the packet headers, which leaks the identity of communication parties. Anonymity system tries to provide the foundation for users to share information over public networks without compromising their privacy.

Here is a simple example: the websites nowadays keep profiling users to provide more suitable services. Large-scale B2C sites like Amazon supplies more suitable candidate items for each user based on their surfing history and transaction records. If we bought some game software, then other games with the same platform or similar genre will be recommended to us on the top page. It makes seller provide better services and gives the buyer convenience, but it also really hurts our privacy. Our transaction records could also be misused by the seller.

Anonymity system could keep websites from profiling individual users. It could also be used for socially sensitive communication: forums or chat rooms for survivors of serious cases, even people with specific illnesses. Journalists may use this kind of system to communicate with whistleblowers and dissidents safely. Corporations use anonymity system as a safe way to conduct competitive analysis.

Moreover, big organizations such as embassies use anonymity systems to exchange information with their home country. Law enforcement could use it for collecting evidence without alerting suspects. Non-governmental organizations usually use anonymity systems to connect to their friends or family while they are abroad, often in the complicated situations, without notifying everybody nearby what they are working with.

1.1.2 Mixnet and the Dining Cryptographers Problem

Mixnet Up to the 80 ' s, 20th Century, shortly after the introduction of public key encryption, David L. Chaum presented the paper - Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms [4]. It was motivated by the object of seeking the solution to a cryptographic problem, "the traffic analysis problem" (the problem of keeping confidential who converses with whom, and when they converse).

This system based on two assumptions: (1) No one can determine anything about the communication participants between a set of sealed (encrypted) items and the corresponding set of unsealed items, or create forgeries without the appropriate random string or private key. That is, in short, indistinguishability and unforgeability. (2) Anyone may learn the origin, destination, and representation of existed messages in the underlying telecommunication system and anyone may inject, remove, or modify messages.

These two assumptions also widely accepeted as the default items in the following researches. With these assumptions, Chaum raised designs with public key encryption to build up a mail system - *Mixnet*. The users will include not only the communication partners but also a series of computers called *mixes* that will process all items of mail before it is delivered. Consider the case which there is one mix only, it uses public key of a node and the communication party. When Alice wants to send a message to Bob through node i, it could be simply described by the following formula:

$$E_i(E_B(M),\ B) \Longrightarrow E_B(M),\ B$$

The $\Longrightarrow$ denotes the transformation of the input by the mix into the output shown on the right-hand side. The mix decrypts the input with its private key in order to output the containing. One might imagine a mechanism that forwards the encrypted message $E_B(M)$ of the output to the receivers who then able to decrypt them with their own private keys.

The purpose of a mix is to hide the correspondences between the items in its input and those in its output. And by using a *cascade*, or series of mixes, they could offer the advantage that any single constituent mix is able to provide the secrecy of the correspondence between the inflow and the outflow of the entire cascade. Incrimination of a particular mix of a cascade that do not correctly process an item is accomplished as with a single mix, but only requires a receiver from the first mix of the cascade, for a mix can use the signed output of its predecessor to show the absence of an item from its own input. An item is prepared for a cascade of n mixes the same as for a single mix. It is then successively sealed for all succeeding mixes:

$$E_n(E_{n-1}(\ldots, E_1(E_B(M),\ B)\ldots)) \Longrightarrow$$

The first mix yields a lexicographically ordered batch of items, with the form:

$$E_{n-1}(E_{n-2}(\ldots, E_1(E_B(M),\ B)\ldots)) \Longrightarrow$$

The items in the final output batch of a cascade are of the form $E_B(M),\ B$, the same as those of a single mix.

The usage of return addresses could also be reached by a similar method: Alice could form an untraceable return address $E_i(A),\ E_A$, where A is its own address and E_A is the public key of Alice. Then Alice can send return address to Bob as part of a message sent by the techniques already described above. (In general, two participants can exchange return addresses through a chain of other participants, where at least one member of each adjacent pair knows the identity of the other member of the pair.) The following indicates how Bob uses this untraceable return address to form a response to Alice, through a new kind of mix:

$$E_i(A),\ E_A(M) \Longrightarrow A, E_A(M)$$

This process could also involve cascade mixes, very similar like we have proposed in the former part.

Mixnet is the very beginning anonymity system, and you could see that it has already solved many problems in anonymity communication. It could be easily developed into both high-latency and low-latency system, but it still lacks some practical solutions. Based on it, many modern anonymity systems are raised in the 21th Century.

The Dining Cryptographers Problem In 1988, another important paper in anonymity communication is also presented by David Chaum, about the famous dining cryptographers problem [3]. It illustrated us a story like this: Three cryptographers are sitting down at the table and made the arrangements for the bill to be paid anonymously. The bill is either paid by one of the cryptographer, or it might have been the third party (It was pretended to be U.S. National Security Agency.). Tree cryptographers respect each other's right to make an anonymous payment, but they wonder if NSA is paying. So they could solve this problem by carrying out the following protocol:

Each cryptographer flips an unbiased coin between him and the cryptographer on his right, so only two of them can see the result. Each cryptographer then states

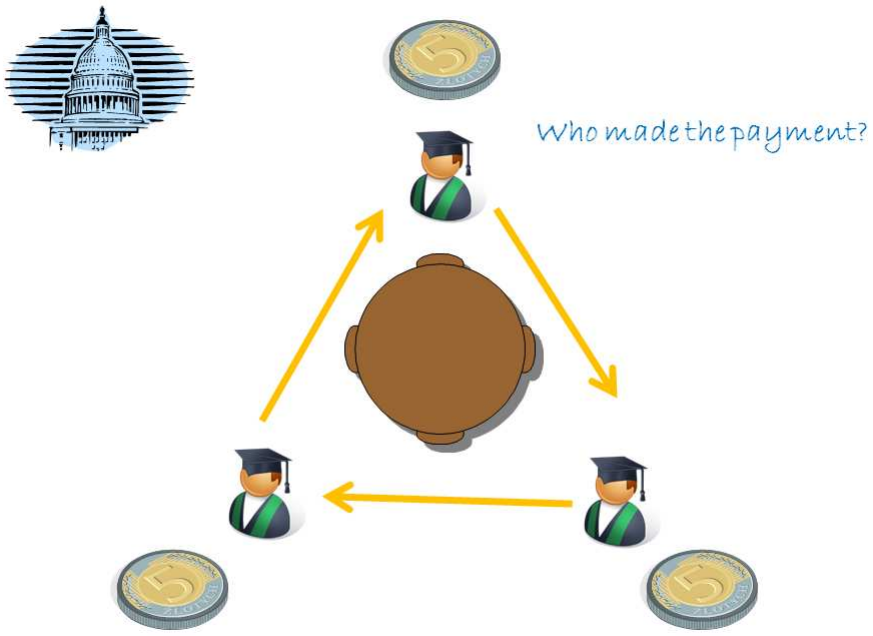


Figure 1.1: Dining cryptographers problem

aloud whether the two coins he can see - the one he flipped and the one flipped by his left-hand neighbor - are on the same or different sides. If one of the cryptographers pays for the bill, he just makes an opposite of what he sees. Then, an odd number of differences indicate that a cryptographer is paying and an even number indicates the NSA is paying.

If the protocol is carried out faithfully, it is unconditionally secure. Consider the dilemma of a cryptographer who is not the payer and wishes to find out which cryptographer is. (There is no anonymity problem about NSA.) There are two cases. (1) The two coins he sees are the same, and one of the other cryptographers said "different," and the other one said "same." If the hidden outcome was the same as the two outcomes he sees, the cryptographer who claimed "different" is the payer; if the outcome was different, the one who said "same" paid for the dinner. For we have assumed the hidden coin is fair, both possibilities are equally likely. (2) The coins he sees are different; if both other cryptographers said "different," then the payer is the closet to the coin that is the same as the hidden coin; if both said "same," then the payer is closest to the coin that differs from the hidden coin. Thus, in each sub case, a nonpaying cryptographer learns nothing about which of the other two is paying.

The cryptographers become intrigued with the ability to make messages public

untraceably. Easily, if they repeated this basic protocol over and over then arbitrary length of message could be distributed anonymously. Mixnet and the dining cryptographers problem formed two general types of modern anonymity system - the low-latency systems and high-latency systems.

1.1.3 Basic Features

In 1986, Pfitzmann and Waidner raised basic concepts for anonymous networks. In [20], they discussed features, performance and fault tolerance of the anonymity system although there were few system at that time. They proposed that three characteristics are important in anonymity system: *Recipient anonymity*, *sender anonymity* and *unlinkability of sender and recipient*.

As they are named, *sender anonymity* means attacker cannot find out the initiator of a message. And receiving message itself can be made completely anonymous if it is delivered by broadcasting. And if the message has an intended recipient, it has the attribute by nobody else could see the addressee, so called *recipient anonymity*. *Unlinkability* is that the relation between sender and recipient of a message hides from everybody but the system and the sender.

Although three features are mentioned by Pfitzmann and Waidner, it is obviously not all the system could achieve these features simultaneously in all conditions. For example, with attacker in the same network, it is impossible to achieve the sender anonymity without keep broadcasting all the times. But no matter what anonymity system it is, the unlinkability is the least requirement to protect user's privacy.

1.1.4 Modern Anonymity Systems

Chaum raised mixnet and the dining cryptographers problem became are two important anonymity systems in the history. But it omitted many practical questions as the node finding, path creating, etc. Modern anonymity system researches focused more on the practical problems and also devoted on how to make the system safer. We will introduce some famous systems here:

Crowds Reiter and Rubin presented crowds in [22], 1998. As it was named, "blending into a crowd" reflected its central idea. The system is consisted by lots of geographical diverse users. Web servers are unable to learn the true source of a request for the probability of a message's initiator is equally to any member of the crowd. And they introduced *degree of anonymity* to describe and prove anonymity properties.

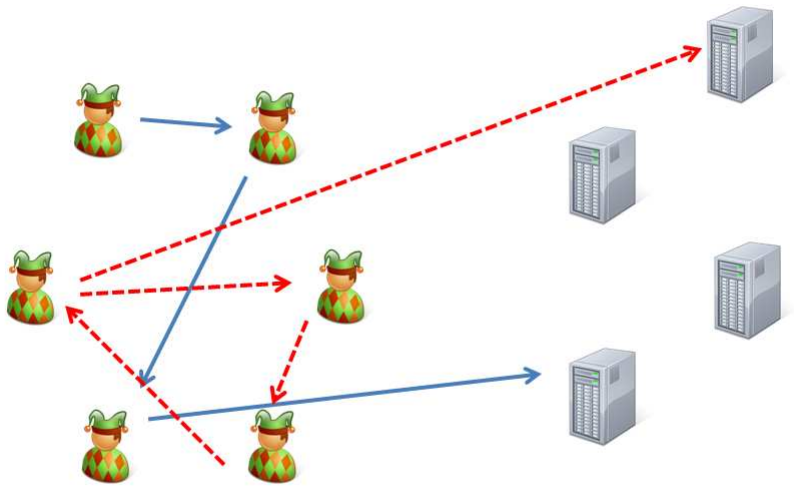


Figure 1.2: Paths in a crowd (Left are jondos and right are web servers)

Basically, the system is consisted by a dynamic group of users and called *crowd*. The users send web requests to various web servers. They defined the users as "sender" and the "receiver" refers to the servers. They considered the anonymity properties against three *distinct* types of attackers: Local eavesdropper, collaborating crowd members, end server.

A user who using crowds first started his proxy which named as *jondo* and then by contacting with the blending server, he could join the crowd. By using jondo, all requests coming from the browser are sent to the jondo. The jondo initiates the establishment of a random path of jondos. It picks up a jondo from the crowd, even could be itself, and forwards the request to it. When a jondo receives a request, it determine whether forward the message to another jondo or not with the probability p. If the result is to forward, then the above process would be executed again and another coin with p is flipped. Otherwise the jondo submits the request to the web server. Subsequent requests to the same server are followed the same path to keep the connection alive, server could replies traverse the same path in reverse. Communication between any two jondos is encrypted using a key known only to the two of them. Figure 1.2 gives the illustration of paths in a crowd.

The *degrees of anonymity* has also given in this paper. Degrees range from *absolute*

privacy, where the attacker cannot perceive the presence of communication, through *beyond suspicion*, *probable innocence*, *possible innocence*, *exposed*, to *provably exposed*, where the attacker can prove the sender, receiver, or their relationship to others.

We do not intend to discuss these degrees in detail here, but it supplied a way to describe different level of anonymity in reasonable way. Also, in this paper, the authors demonstrated us the how to describe the security level of crowds. With their settings, the system is secure towards given attackers in the meaning of anonymity. But in more general way, the system even without end-to-end encryption could make any jondo change or edit the initiator's message easily. Of course that cannot be called "safe".

Tor Onion routing is a distributed overlay network designed to make TCP-based applications anonymized for general purpose Internet activities like web browsing, SSH connection, and instant messaging, etc. (There are other general purpose overlay network on different layer for IP protocol like Tarzan in [9]) As an implementation of the second generation onion routing, Tor is freely available and runnable on most of the operation systems. With the support from United States government and donation from kinds of organizations, Tor grows quickly and has become the most widely used anonymity system in the world. In 2004, the volume of traffic in the whole system is only 16GB per week. Surprisingly, now the number is more than 5TB per week. More than 2000 nodes are running around the world in any minute, also the number of users is more than one million now. The security of anonymity system not only depends on the design, but also correlates with the number of users. Imaging a very safe anonymity system but with only 1 user, then it is nothing difficult for attacker to decide whether the user is communication or not. More users mean the system safer. By this meaning, Tor may be the safest low-latency anonymity system in the world.

Also, there are some arguments about the safety of Tor in practical usage. Due to [10], many passwords, even from embassies are leaked through Tor. But the reason is people do not understand anonymity system enough and misused it. And, this case also provides us how popular the Tor really is.

The contribution of Tor by introducing many solutions such as: perfect forward secrecy, directory servers, congestion control, integrity, configurable policies and so on, a robust and usable anonymity system is provided to users with reasonable tradeoff between anonymity, usability, and efficiency. For the Section 2 and 3, we will talk about the path creation and cell construction in detail as the background knowledge.

Components of the Tor Network The essence of Tor anonymous communication system is an overlay TCP network. As shown in Figure 1.3, there are four different entities:

1. *User.* The user (also called client) uses *onion proxy* (OP) on local to provide the application anonymity transactions on the Tor network.

2. *Server.* The destination which user visited. It acts as a server side application to accept TCP requests from user.

3. *Onion Routers (OR).* Onion routers are the core components to provide anonymity communications in the Tor network. They relay the packets between user and server. Transport Layer Security (TLS) connections are also employed in the Tor network to provide link encryption between two onion routers. The Tor packet size is also restricted into a specific number, which is 512 bytes.

4. *Directory Servers.* They are the nodes information holder in the Tor network. Onion proxy need to query the directory servers before it connect to the Tor network. There are *directory authorities* and *directory caches*. Directory authorities have the authoritative information about the onion routers to make efforts to defend from malicious nodes. Directory caches download information of onion routers from authorities and users download these information from directory caches.

The Design of Tor Cells Onion routers communicate with each other with TLS connections and ephemeral key to achieve perfect forward security. The data modifying and OR impersonating are also prevented.

The data are transmitted through Tor network in fixed-size cells. The default cell size is 512 bytes with a header and a payload. The header includes a circuit identifier (circID) so the circuit related to this cell is specified with it, and a command to tell the Tor network what to do with the payload. Then the cells are either *control* cells or *relay* cells. Control cells are interpreted by the node that receives them and relay cells carry the end-to-end data. The detailed description could be found in [7].

Establishing and Transmitting of TCP Connections To make a connection through Tor network, user chooses several onion routers from the nodes list downloaded from the directory server. Once the path is decided, OP constructs the path incrementally. It negotiates a symmetric key with each OR on the route, one hop at a time.

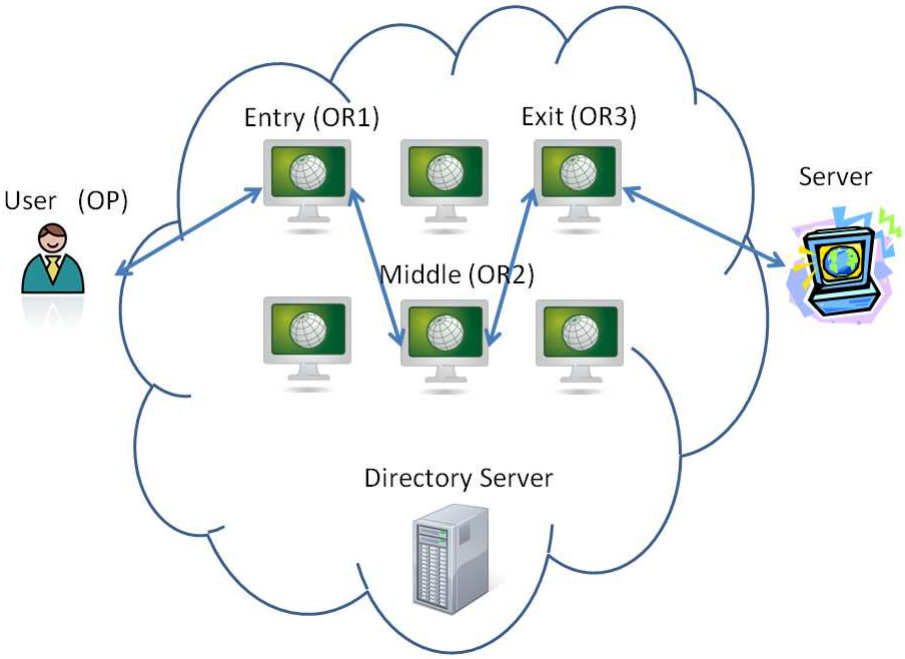


Figure 1.3: Basic Components in Tor Network

First, the OP (user) sends a *create* cell to the OR1. The *create* cell's payload includes the first half of the Diffie-Hellman handshake (g^x), encrypted with the public key of the OR1. OR1 responds with a *created* cell with g^y and a hash of the negotiated key $K = g^{xy}$.[1] Obviously, after this hop has been established, OP could send OR1 relay cells encrypted with negotiated key.

The next step is to extend this path further. OP sends a *relay extend* cell to OR1, and tell OR1 which node the path should be extended to, and an encrypted handshake g^{x_2} for OR2. OR1 copies the encrypted handshake as the payload into a *create* cell then sends it to OR2, as if it was the path initiator. After OR2 returns with the *created* cell, OR1 wraps the payload into a *relay extended* cell and passes it back to OP. So we have the two-hop path now, and OP shared the key $K_2 = g^{x_2 y_2}$ with OR2.

So, it is obviously to see that if we want to extend this path further, just do as the first extending process. If we send the last node the object, then we could extend the path by one hop further. And by now, the default path length for Tor network is 3.

Once the path has been established, user could send relay cells through it. When an OR receives a relay cell, it will be decrypted. Then either it will be delivered to

[1] Noticed this process is slightly different from the ordinary Diffie-Hellman protocol. Since we only need to verify the identification of nodes without caring who the user is, it is safe to be used here.

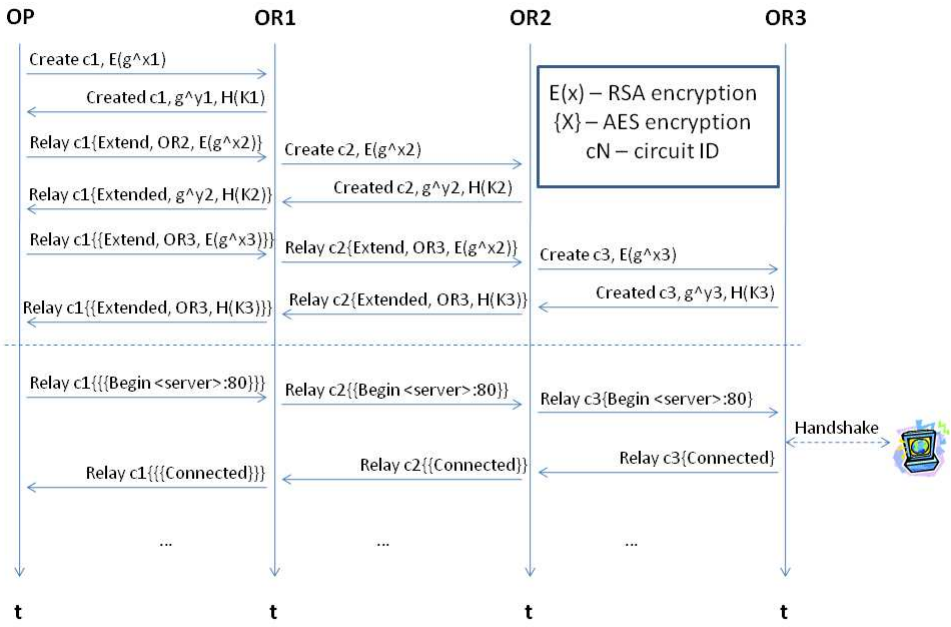


Figure 1.4: The Circuit Building and Beginning of Server Visiting

another OR, or be submitted to the final destination. Since the decryption of payload will only be meaningful at the last node, to distinguish these two situations is not hard for an OR. This *leaky pipe* circuit topology allows OP's packets to exit at different ORs or create circuits with different lengths. Figure 1.4 illustrates the circuit building process and a simple demo of beginning server visiting.

There are still lots of other anonymity systems like PipeNet, Babel, MorphMix, etc [11, 5, 23]. We cannot cover all kinds of anonymity systems here. But thanks to the researchers in this field, their efforts make the Internet life safer and safer to people who really put high importance on their privacy.

1.2 Our Contributions

Our first contribution is raised up the fingerprinting attack towards Tor anonymity system. Since Tor employed several mechanisms to defend it, the ordinary fingerprinting attack do not work well on it. Although names are same, our attack uses a different method to make the attack. Also we have showed the possibility and effectiveness of the attack towards Tor anonymity system.

The second contribution is improvement of the attack with modified threat model, the attack also enhanced by adding the time factor into formula. With the manually tune up, the attack could become even more powerful in some situations.

The effectiveness of our method is discussed in a comprehensive manner: experimentally and theoretically as the third contribution. With the properly evaluation, we could show the threat of this kind of attack.

Finally, we contributed in discussing about the effective defensive mechanism towards the attack and make the suggestion that the new-designed anonymity system should treat fingerprint attack as a practical threat and employ defensive mechanism against it.

1.3 Organization

In the following sections, we will summarize related works in Section 2, the attack methods in Section 3, from the ordinary fingerprinting attack to our original plan and evolution. In Section 4, we will discuss the experiments and evaluations. Countermeasures will be discussed in Section 5, and finally we will give the conclusion in Section 6.

Chapter 2 Related Works

There are several ways to classify the attacks. One of the most widely used is classify attacks as active or passive or something else. But in my opinion, classify attacks by their threat models will be more meaningful for the attackers won't be restricted by whether they should make the attack actively or passively, but restricted by the resources they process. Also we could distinguish attack ' s ability by looking into the different threat models.

2.1 End-to-end Attacker

End-to-end confirmation attack is the main-stream kind of attacks in the anonymity research. It gives us a model: There is an adversary between two anonymity system users, initiator and responder. He could observe all the inflow and outflow of the designated users. And he wants to make sure whether initiator and responder are communicating. More generally speaking, he wants to decide whether initiator and responder are in the same path of anonymity system. Figure 2.1 illustrates the simple form of the timing attack.

The paper in this model, like [1, 14], once raised an arm-race in researching of anonymity system. It really hurts the anonymity of users in an anonymity system. But the nowadays systems, especially low-latency anonymity systems, are explicitly implied that this kind of attacker is not considered in their designation. First, it is so strong assumption for an adversary to achieve. To identify one path, the attacker need to take the control of two points, which maybe so far away between each other. And when we want to identify a user's activities, the point we need to occupy increasing rapidly across different autonomous systems.

The essence of a timing attack is to find a correlation between the timings of packets seen by M_1^I and those seen by an end point M_h^J. The stronger this correlation, the more likely $I = J$ and M_h^J is actually M_h^I. Attacker success also depends on the relative correlations between the timings at which distinct initiators I and J emit packets. That is, if M_1^J and M_1^J happen to see exactly the same timings of packets, then it is not be possible to determine whether the packet stream seen at M_h^J is a match for M_1^I or M_1^J. Hopper et al. discussed how information leaks from timing systematically in [13].

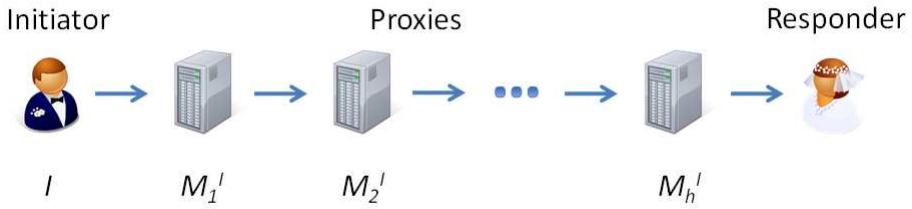


Figure 2.1: A path P^I with an initiator I communicating with a responder. M_1^I and M_h^I, the first and last mixes on the path originating at I, are controlled by attackers.

In the end-to-end attacker model, there is an interesting paper by Pries et al.[21] Recall the purpose of confirmation attack is to confirm that Alice is communicating with Bob. (Also called initiator and responder above) The attack starts from the malicious entry router. The entry router first attempts to identify a target cell from the TCP stream data on a circuit and duplicate that cell. When the cell is duplicated, the cell's source IP and the time of duplication will be logged. This duplicate cell traverses the circuit and consequently arrives at the exit router. The attacker at the malicious exit router should detect an error caused by this duplicate cell and record the time, the original cell's destination IP address and port. In this way, it is confirmed that the target cell is using the entry router and exit router. Since the entry router knows the sender of the cell is Alice and the exit router knows its receiver is Bob, the communication relationship between the sender and receiver is confirmed.

Figure 2.2 illustrates the basic principle of replay attack. You could see it compare to Figure 1.4. It is an interesting attack, for some timing attacks also use techniques like packet dropping to gain some advantage in recognizing the circuit, this kind of attack causes an unusual event and could make the confirmation immediately.

There is a new end-to-end attack in [15], they want to confirm anonymous communication relationship among users accurately and quickly, also make it difficult to detect. So they select the target, embed the signal, record the target, and recognize signals. Through these processes, attacker could prove whether these communication partners are in the same path or not. Also, we could see other works like [17].

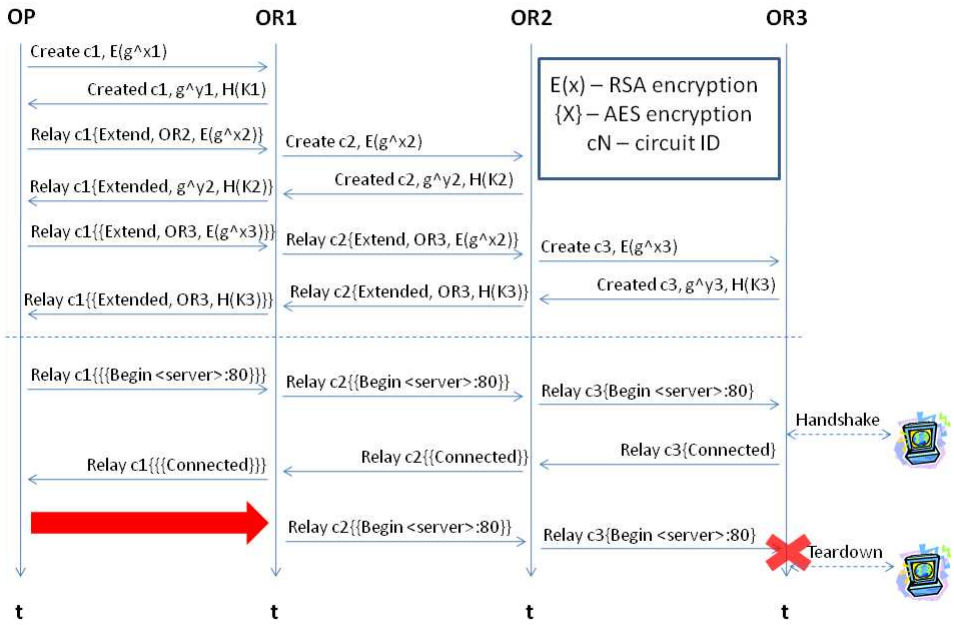


Figure 2.2: Replay Attack on Tor

2.2 Entry Point Attacker

This model has a different from the previous one, that is - the adversary only occupied the entry point. In Figure 2.1, that is, the adversary only stands in M_1^I.

In the first glance, it differs a little from the previous one, but actually it is a great difference. In this threat model, many mechanisms against attacker become more useful like defensive dropping, including variable latencies, etc. The most typical attack in this model is fingerprinting attack.[12]

Generally, when user visits a typical webpage, it is consisted by many different files. First, the HTML file is downloaded from the site, then pictures included in the page, background music, flv movie, etc. would also be downloaded after that. If we surf the webpage at www.yahoo.co.jp, about 23 files would be retrieved from the server. Each of them has a specific file size in the most cases. Table 2.1 illustrates an example.

In a typical browser, such as Microsoft Internet Explorer, each file would be downloaded via a separate TCP connection. So that, we could easily detect every TCP flows since they use different ports to transfer the files. Then, attacker can determine the size of each file being returned to the client. All the attacker need to do is just count the total size of the packets on each port.

Table 2.1: The files of top page on www.yahoo.co.jp, 05/29/2009

File Name	Size
index.htm	132KB
84_84_0582.gif	3KB
84_84_0587.gif	2KB
84_84_0953.gif	3KB
84_84_0986.gif	3KB
0529a.jpg	5KB
20090528-00000033-jijp-soci-view-000-small.jpg	6KB
b.gif	1KB
b(1).gif	1KB
b(2).gif	1KB
b(3).gif	1KB
b(4).gif	1KB
b(5).gif	1KB
b(6).gif	1KB
b(7).gif	1KB
b(8).gif	1KB
clr-090413.css	7KB
fp_base_bd_ga_4.1.1.js	92KB
logo.gif	3KB
rain_clods_st.gif	1KB
uranai_090525.gif	2KB
xwetzr_auwmsmeujit0b-a.jpg	20KB
yfa_visual4.js	6KB

This kind of attack is not only can be applied to the plain flows, but also the simple anonymity system just like SafeWeb. With common encryption methods, we do not try to obfuscate the transmitted data for both performance and requirement reasons. If someone monitors the Safeweb user, the number and approximated file size could be determined. For example, the eavesdropper found that the user created 3 connections with the same target, each of the connections received respectively 1324 bytes, 582 bytes, 32787 bytes. Each of these transfer sizes corresponds with a certain file directly. The set of file sizes consists the *fingerprint* of a webpage.

So the attacker could first try to build the fingerprint of the webpages, then monitor the user. When the user is surfing a webpage, connections and related data could be detected by the attacker. Then the attacker just compare the connect data with a set of fingerprints, choose the closest one, then "guess" that the page is what user surfing now. The attack is low-cost and easy to apply, which really hurt the user's anonymity.

2.3 Malicious Nodes

In this model, it assumes the adversary occupies several nodes in the system and then try to find what they could disclose. Figure 2.3 illustrates this model simply.

Almost every anonymous system would make some discussion about this threat model. Some will focus more on it, like [18]. It could be easy turned into the end-to-end attacker model or entry point model. If an adversary controls $m > 1$ of N nodes, he can correlate at most $(\frac{m}{N})^2$ of the traffic. And, with Sybil attack[8], the proportion could be even larger.

Another approach in this threat model is predecessor attack. When using an anonymity system, user will continuously make many connections through different paths. Then in the anonymous system which has lots of malicious nodes, the possibility of connect to a malicious node is greatly increased. So, different malicious nodes may observe same predecessor, although they don't know whether it is a user or just a node in system. They could guess it as a user by statistical inferring. This attack is especially useful in P2P anonymous communication system or all the conditions that the attacker cannot distinguish user and node.

2.4 Outside Points

Most threat models would occupy some points in the system to gain some information to implement the attack. There are also some special cases that the adversary stands

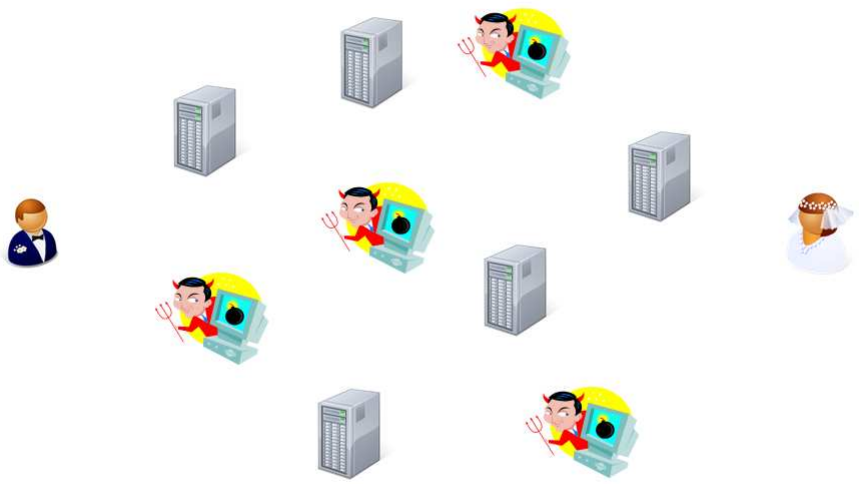


Figure 2.3: Malicious Nodes in Anonymous System

outside the system and try to attack it. It is more generic model, and, of course, more hard to success.

Chakravarty et al. presented a novel mechanism to exposes the identity of anonymous system nodes.[2] They employ a approximately measure method LinkWidth to detect induced traffic fluctuations in anonymity system nodes.

LinkWidth is a tool that allows attacker to estimate available and capacity bandwidth on a path, without additional support or active collaboration from a remote host or any device in the network. To measure end-to-end TCP capacity, the sender emulates the TCP Westwood sender by sending cwin packets. cwin-2 TCP RST packets (called load packets), are sandwiched between two TCP SYN Packets. These TCP SYN packets, sent to closed ports, evoke TCP RST+ACK reply packets. Correct reception of the train of cwin + 1 packet is determined by two TCP RST+ACK packets from the receiver (due to the head and tail measurement packets). Each correct reception of the TCP RST+ACK pair causes cwin to be increased either exponentially (Slow Start phase) or linearly (Congestion Avoidance phase). Since the attacker does not rely on an established TCP connection, the only way to signal a packet loss is by coarse timeout. After sending the train, the sender initializes a timer to wait for the two expected ACKs. The expiration of the timeout causes the readjustment of the

cwin and ssthresh parameters inside a timeout event handler method.

The attacker could use TCP RST packets to avoid generating unnecessary replies, either in the form of TCP RST or ICMP Destination Host/Net Unreachable packets, which could potentially interfere with our forward probe traffic. The time dispersion between two consecutive TCP RST+ACK replies due to the head and tail measurement packets are stored as t_n and t_{n-1}. Thus the capacity/bandwidth is measured as:

$$b_k = \frac{cwin * L}{t_n - t_{n-1}}$$

Here, b_k is the measured " instantaneous " bandwidth (measured throughput), $cwin * L$ is the total data sent (in bits) for the entire train, t_n and t_{n-1} are the times of reception of the two TCP RST+ACK reply packets. The successful reception to a previous train determines how many packets the attackers send in the current train. This method is a direct extension of the packet train method.

Figure 2.4 illustrates how an adversary probes the nodes involved in a circuit. They probe nodes that may possibly be part of anonymity communication paths. An adversary with sufficient bandwidth resources can simultaneously probe all (or a large fraction of) nodes. If some nodes have the similarity bandwidth fluctuation, then the attacker could guess they are in the same path.

This attack requires little by definition, but in practical, it works well only when the attacker uses a well-provisioned probing node is at a network "vantage" point with respect to the victim nodes. Stated simply, this would mean that the bottleneck in the path connecting the adversary to the victim relay should be the latter. This is somewhat like a "pseudo" global passive adversary and limit the usage of this method. Other than that, for all the attacker observed is the fluctuation in bandwidth, so only the actions that is affect bandwidth greatly can be detected. E.g. The paper itself evaluated by whether attacker could aware a 100MB file transfer.

2.5 Black Box Model

Compare to other attacks, it definitely has the strongest assumption. But systems they want to break is also quite strong - *high-latency anonymity systems*. The most famous attack under this model is *long-term intersection attack*. In this attack, a passive attacker observes a really large volume of network traffic and find out some receiver are more likely to receive messages after some specified participants have transmitted messages. Some attacks are presented like [6, 16]. By using coarse-grained timing, the

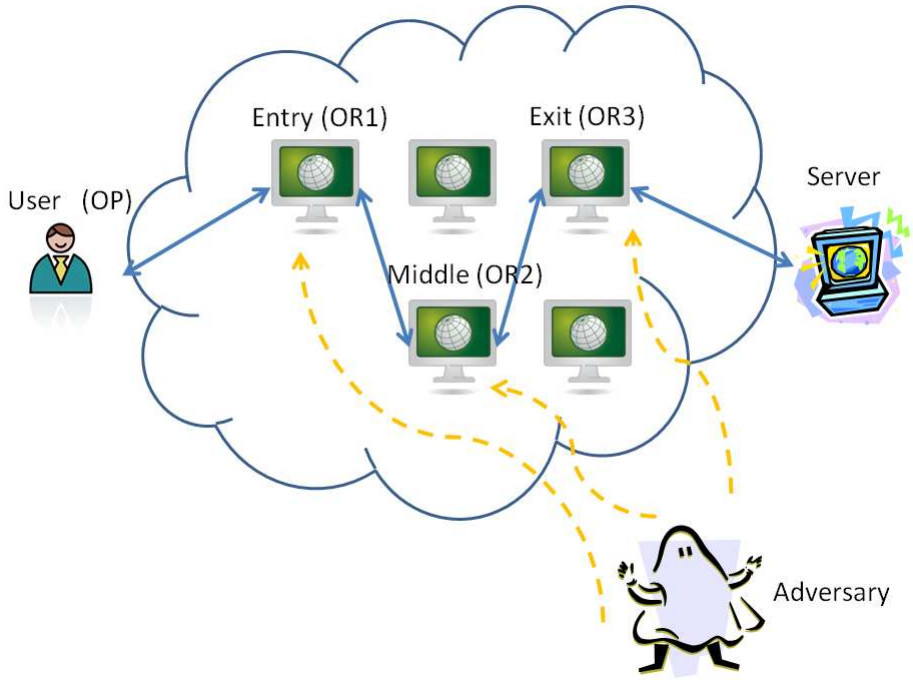


Figure 2.4: Adversary probing available bandwidth of nodes

attacker treats the entire anonymities network as a black box, and correlating traffic that enters and exits the system to determine communication patterns.

We introduce one of the long-term intersection attacks here, the *statistical disclosure attack*. The attack only reveals *likely* recipients with statistical method. In this attack, they model Alice's behavior as an unknown vector $\vec{v}$ whose elements relate to the probability of Alice sending a message to each of the N designated corresponders. The elements of $\vec{v}$ corresponded to the m recipients will be $1/m$; the other $N - m$ elements of $\vec{v}$ will be 0. Other users' "background" traffic are described as a known vector $\vec{u}$ and each of the elements is $1/N$.

The attacker derives an observation $\vec{o_i}$ from output of each round. Elements are reflected the probability of Alice's having sent a message to each particular receiver in that round. In other words, in a round i where Alice has sent a message, each element of $\vec{o_i}$ is $1/b$ if a specific recipient who could receive this message, and 0 if it does not. Then by taking arithmetic mean $\overline{O}$ of a large set of observations, we could get (by the law of large numbers):

$$\overline{O} = \frac{1}{t}\sum_{i=i}^{t} \vec{o_i} \approx \frac{\vec{v} + (b-1)\vec{u}}{b}$$

Table 2.2: Comparison between Threat Models

Threat model	Assumption
Outside point	Very weak
Entry point	Weak
Malicious nodes	Somewhat strong
End-to-end	Strong
Black box	Very strong

So the attacker could estimates Alice's behavior as:

$$\vec{v} \approx b\frac{\sum_{i=1}^{t}\vec{o_i}}{t} - (b-i)\vec{u}$$

With calculation, author has also derived the requirement for the attack that is the attack will only succeed when $m < \frac{N}{b-1}$, and calculates the expected number of rounds for the attack to be succeed (with 95% confidence for security parameter $l = 2$ and 99% confidence for $l = 3$):

$$t > \left[m \cdot l\left(\sqrt{\frac{N-1}{N}(b-1)} + \sqrt{\frac{N-1}{N^2}(b-1) + \frac{m-1}{m}}\right)\right]^2$$

In short, the central idea of this attack is: If the activity of Alice different from other users and this difference exists for a long time, then by applying the statistical method to many observations, attacker could gain some knowledge about the activity pattern of Alice.

Although the mechanism of this attack is easy to understand and the result is not good enough to implement, from the description above, we could know that this model is a more theoretical way for the impractical bandwidth requirement and whole coverage of the anonymity system.

2.6 Comparisons

We have introduced a lot of threat models above, and we give Table 2.2 and 2.3 to make the comparisons between threat models easy to understand.

The first column of both two tables is sorted by the difficulty for attacker to meet the requirements. We could see from the Table 2.2 that entry point has a quite weak assumption so that attacker could implement attacks with the threat model of entry point easily. Outside point model it is the weakest assumption among these

Table 2.3: Comparison between Attacks

Attack name	Threat model and comments	Strength
Fingerpringting attack	Entry point	Somewhat effective
Timing attack	End-to-end	Effective
Replay attack	End-to-end, active	Very effective
Bandwidth probing attack	Outside point, high bandwidth	Effective
Statistical disclosure attack	Black box	Weak (vs High-latency)

models. But we have to aware that without additional support factors; the outside point attacker could merely do nothing since that is the most widely existed potential threats. And we could also see even with the same threat model, the strength of timing attack and replay attack is different due to the attack itself is passive or not. Although statistical disclosure attack is weak, but it is the only attack which could analysis the user's activity pattern in a well-designed high-latency system. If an attacker could achieve the black box model and use it against low-latency system, he could do at least as well as end-to-end model.

In practical, threat model which stronger than entry point is hard to achieve. What's more, high bandwidth is also difficult requirement. So we want to develop an attack which is more realistic to call for the attention on the privacy protection. Also by developing attacks, it could help us to understand the conception of anonymity more clearly. Then make the researchers help the anonymity system become more secure in the future.

Chapter 3 Fingerprinting Attack on Tor

In this section, we will first review the characteristic of Tor and why original fingerprinting attack does not work on it. Then raise our proposal of the fingerprinting attack on Tor, extend the attack from threat model and attack method.

3.1 The Characteristics of Tor

Tor is a low-latency, well developed anonymity system. It uses multi-hop encrypted connections to protect sender and/or receiver anonymity. Tor extends the former onion routing scheme by adding some features like integrity protection, congestion control, and location-hidden service. Tor can be used for both sender and receiver anonymity. Sender anonymity could help a user to use services without disclosing their identities. In Tor's design, it employs two significant characteristics, which prevents the fingerprinting attack to some extent.

First, Tor employs quantized data cells; each data cell is fixed at 512 bytes. So it is obviously difficult for an attacker to detect the accurate size of files transferred by separated connection stream.

Second, Tor uses multiplexing to combine all the TCP streams into one connection. This is not for the safe aspect at first. The original Onion Routing creates a path for each TCP stream. But for the expensive communication cost, Tor decides to use multiplexing to reduce the expensive path-establish cost. And it also provides some resistance to the client against fingerprinting attacks, for the attacker cannot distinguish the connections between each other easily.

3.2 Threat Model

Although many attacks toward low-latency anonymity systems are successful in their assumed environment, Tor and other anonymity systems are considered to be secure in practical use. Many attacks involve a strong adversary, who could perform end-to-end confirmation or even global eavesdrop. And in practical world, it is obviously difficult to achieve this kind of requirement. Even for big organizations to observe all the nodes distributed in the whole world is almost impossible. The advantage of fingerprinting attacks is the low resource requirement. The adversary only needs to occupy the entry

point of the user. Compare to the end-to-end confirmation attacks, they just use the resources which much easier to satisfy make it more possible to implement.

Our fingerprinting attack on Tor uses the same threat model with the fingerprinting attack by Hintz, the attacker is assumed to occupy the entry router of the user and observe all the data flows from the user. He wants to guess what webpage the user is surfing now. The design objective of Tor is attempting to defend against external observers who cannot observe both sides of a user's connections. So we think our threat model is appropriate against low-latency anonymity system.

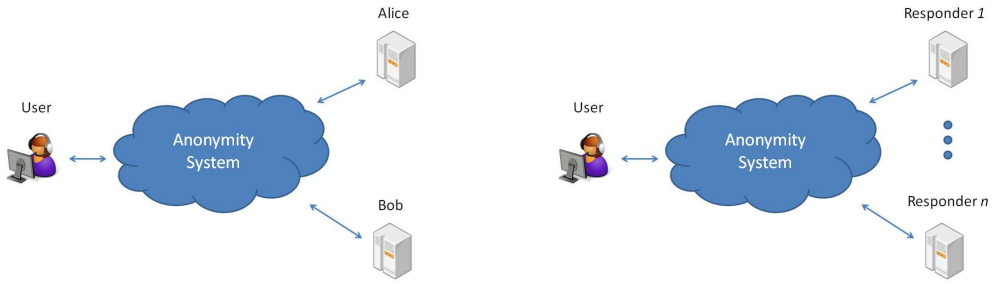
Let us describe the model more formally, assume there is a user and two responders: Alice and Bob. An adversary can watch all the connections related to the user. First, the adversary could use the anonymity system to visit Alice and Bob for many times. Then the user visits either Alice or Bob using the anonymity system under the adversary's observation. Then the adversary would guess which responder the user connected to. We have some a priori probability, which models our suspicion about who is communicating with whom. More precisely, the a priori probability that the user is communicating with Alice is p and the a priori probability that user is communicating with Bob is $1 - p$. If we have no priori information, $p = 1/2$. See Figure 3.1(a).

Then, the model could also easily be extended to n responders, assume now there are n responders, from Responder 1 to Responder n. First, the adversary could use the anonymity system to visit any responder for many times. Then the user visits one responder using the anonymity system under the adversary's observation. Then the adversary would guess which responder the user connected to. We have some a priori probability, which models our suspicion about who is communicating with whom. More precisely, the a priori probability that user is communicating with Responder i is p and the a priori probability that the user is communicating with other responders are $1 - p$. If we have no priori information, $p = 1/n$. See Figure 3.1(b).

3.3 Fingerprinting Attack with Intervals

Attack Method So we come to make our fingerprinting attack towards Tor. The biggest problem is that the only connection makes it hard for the adversary to distinguish each file size and the characteristic of the webpages becomes hard to define.

Generally, if we observe the traffic flow from/to the user, we will see a sequence of packets. If we use the outflow from user to separate the flow, we will see some interesting things. Some intervals may be very short, like 1 or 2 packets between two outflow packets. That means this interval transferred some small files or does



(a) Model with 2 responders (b) Model with n responders

Figure 3.1: Model in Fingerprinting Attack

some protocol transactions, etc. And some intervals may be relatively long, like 5 or 6 packets. This means a bigger file is being transferred. And after the TCP sliding window is fulfilled, the user send the acknowledge packet and continue the transfer process. If the network condition remains stable, this traffic pattern will not change much. So, for the webpages with different files and different loading process, we can distinguish them to some extent.

With a specific packet sequence, we could use the method described above to make a continuous intervals with the different number of packets. We call *all the inflow packets in a sequence, without any outflow packet placed in them*, an **interval**. We then define a vector $\vec{V} = (v_1, v_2, \ldots, v_n)$, where v_i means "the number of intervals with i packets". $n_{\vec{V}}$ means "the total number of intervals in $\vec{V}$". We build a fingerprint vector $\vec{F}$ in advance. Let weight w defined as $n_{\vec{V}}/n_{\vec{F}}$ or $n_{\vec{F}}/n_{\vec{V}}$ which is smaller (equal when $n_{\vec{V}} = n_{\vec{F}}$) than 1. So we use this formula to calculate the similarity score - $S_{Interval}$:

$$S_{Interval} = \frac{\vec{V} \cdot \vec{F}}{\|\vec{V}\|\|\vec{F}\|} \cdot w_{Interval} \tag{3.1}$$

If we have several fingerprints, we could calculate observed $\vec{V}$ with each $\vec{F}_i$ to get several similarity score S_i (Here we omit the label of $Interval$, since it could obviously be used with other methods), then we could sort all the S_i and make the assumption the user is surfing the webpage with the $\vec{F}$ correlated to the largest S_i.

In the ordinary fingerprinting attack, because a webpage is usually consists from about 20 to 30 files, and each file has its own unique file size. It means that the number

of distinguishable webpages is very large. But in our work, the information we used is really limited due to the multiplexing. So if the number of fingerprint we use is too large, we may not have a very high detection rate. When the webpages the user may access are too many, after sorting the similarity S, we do not make the assumption only with the biggest S, but also using a threshold value θ instead. All fingerprints with calculated score larger than θ could be the possible page the user has seen. And we could make this as a set. If we could make sure the user is surfing the same page again and again (but we do not know which page he is watching), then we get other sets. Combine these sets and finally we could get the most possible answer.

The Choice of Fingerprints So far we have discussed our threat model, the score calculation formula and the method to recognize the page. But how can we choose a fingerprint?

Generally, any vector $\vec{V}$ could be a fingerprint but the unique noises are also included in the fingerprint. An adversary may do the sampling work in advance and make a lot of vectors from one page. He wants to use them to achieve a higher detection rate from the data, so which one should he choose?

The fingerprint choosing method is also discussed in ordinary fingerprinting attack paper: the author claims that we should choose the smallest sizes sampled for each file. It is an intuitive idea that if we observed the same thing with the smallest size, then it must be with minimum noises. But in our opinion, for the adversary has almost same network condition as the user. The fingerprint should not only reflect the characteristic of webpage, but also the network condition of user.

We could assume the attacker access a webpage n times and recorded vectors as $\vec{V_1}, \vec{V_2}, \ldots, \vec{V_n}$. We calculate the scores with each other by formula 3.1. Then we could get the scores S_{ij} calculated from $\vec{V_i}$ and $\vec{V_j}$ $(i = 1, 2, \ldots, n-1, j > i)$. So we could choose $\vec{V_i}$ with the maximum S_i' as the fingerprint vector $\vec{F}$, which represents:

$$S_i' = \prod_{j}^{j \neq i} S_{ij} \tag{3.2}$$

3.4 Collusion Threat Model

Although Tor has employed several techniques to defend itself from attackers, it is still hard to completely prevent the information leaking. We have presented an fingerprinting attack towards Tor above, which is based on a practical threat model. Here we will present another threat model, which is stronger than the ordinary one, based

on the *leaky pipe* feature of the Tor anonymity system.

Suppose attacker controls the entry point of the user (That is the minimum requirement of the fingerprinting attack) and m malicious onion routers out of N nodes. It is easy to see that with the probability of m/N, this situation becomes an end-to-end attack. (Notice that the attacker occupies the entry point with the probability of 1, so the probability of end-to-end attack here is different from the ordinary one in the basic model - $\frac{n^2}{M^2}$) But if we do not occupy the exit node but the middle one, with some tricks we could still improve the success rate of our attack.

The purpose of fingerprinting attack is to confirm the webpage which user is visiting. We suppose that we have the both entry point of user and the middle onion router. The first thing we need to do is to confirm that these two positions belong to one circuit. Since here we could use the active attack, like insert time gaps between packets to make some significant events in the entry point for the middle point to observe that. Also we could use some more simple ways, like packet counting attack as well. Similar process is implemented as in [19]. In essence, this process is an end-to-end attack, so high success probability of this step could be expected.

The next step is to create an one-hop circuit. After we have made sure that we occupy the middle onion router, we could build an one-hop circuit from it. Since we know the exit router of this circuit, (Remember that each router knows the previous node and the next node by default so that they could pass the message, but without extra information, they will never know the exact position they are standing at.) the malicious middle onion router could send a *create* cell with a new circuit ID to the exit onion router, and when the OR3 receives this cell, it just builds up a circuit with OR2 as usual and returns a *created* cell. After that we could see that an one-hop circuit has been built up, and from the view of malicious middle router, the one-hop circuit and the ordinary circuit from OR2's view have the same length, same following node, that means roughly same RTT, latency, etc. Figure 3.2 shows the threat model and Figure 3.3 represents the attack process.

Third, after the one-hop circuit is built, we could do the fingerprinting attack. In this time, we do not need to make the fingerprints in advance. Since we could never know which path the user will choose, the fingerprints which made beforehand will contribute nothing to the success rate. So in this situation we will use the observed user's traffic pattern as the fingerprint. Then we will use our one-hop circuit to visit the webpages user could possibly visited and compare to the user's traffic pattern. Then as the normal fingerprinting attack, we will choose the one with the highest similarity score and make the assumption.

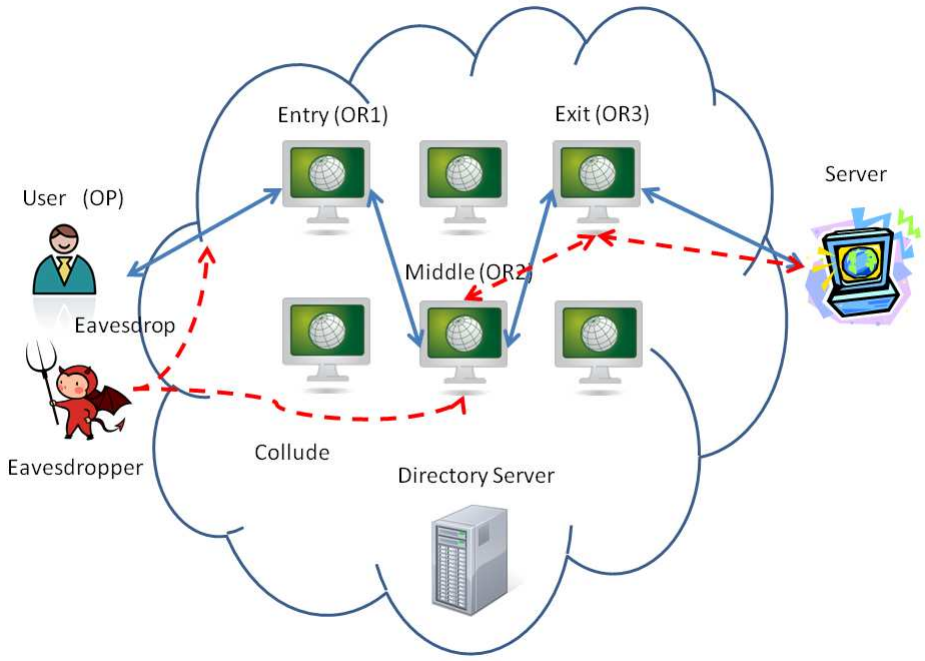


Figure 3.2: The Circuit Building and Beginning of Server Visiting

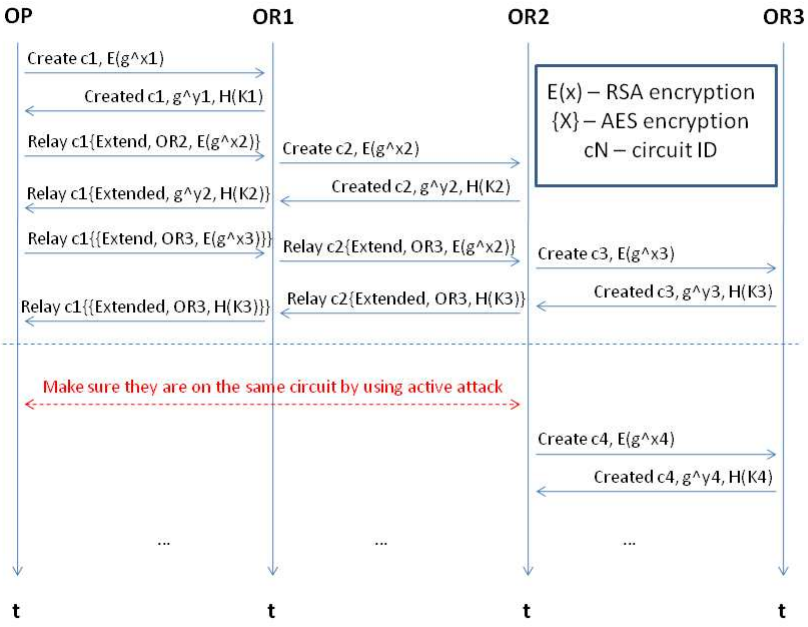


Figure 3.3: The One-Hop Circuit Building

3.5 Fingerprinting Attack with Time Windows

With the definition and application of *interval* into traffic pattern, we could gain some advantage towards anonymity system users, but the result is still not so satisfied to us. The interval vector method omitted the information of relative positions between intervals, so we could get a robust result, which will not change greatly by some abnormal events (e.g. re-transmission, lag, etc.) that may occur quite often in practical network environment. Also, we will not get very good resolution for using so limited information.

So we want to introduce some other factors to get better resolution and success rate for our attack plan. Time is a good candidate for us, it is widely used in all kinds of passive attacks. The problem is: how could we introduce the time into our attack?

First, we tried to make the assumption that all the packets remained the same positions; the time between packets are kept relatively constant (remain same or with same proportion). Then we may want to use a long vector to describe the time between each packet and calculate the similarity by use correlation or other method. Unfortunately, the result is not as good as we expected.

Then we want to use a slightly more rough way to measure that: we tried to make the assumption that the time between intervals (as we claimed above) are kept relatively constant. This is because that the several packets in an interval are transferred in a very short period but the waiting-for-response time is mainly related to the network environment.

Although the result is better than the first one, it is still not a good method. In these two ways, we treated the whole traffic pattern as if it was a "spring". When the network lag is high, the "spring" is stretched and vice versa. But the thing is: practical network is not so stable as we thought, the relative position of intervals also not remained same all the time. We want to find a better way to solve that.

Finally, we have found that by dividing into several windows, calculate the correlation between packets number in each window is a good way to make the resolution better. We also made some assumptions that are:

- Each page is consist by several files with different sizes. (Same as ordinary fingerprinting attack)
- In network transfer, (especially with good network environment), time is largely consumed by the waiting-for-response time than the time which is using for packet transport.

With these two assumptions, even the similar webpages (in the number of files, file

sizes) with different sequences by using this method could be distinguished.

In this method, the basic concept is divide a given traffic pattern by relative time (e.g. 25%, 50%, etc.) That is because the time itself varies greatly due to the different path. Under the given assumption, we could treat the packet transfer time as "very short" and see the waiting-for-response time as the main part of a traffic pattern's time line. Then if the path is slow, the total time is long and vice versa, but the packets in each time window will not change greatly in normal cases. Then by calculate the correlation between two time window series, we could make the guess.

Let us discuss it in more detail way: First, decide how many windows should be divided - the total window number n. So the length of each part would be the $(total\ time/n)$. Then we will get a time window divided vector as $(v_1, v_2, ..., v_i, ..., v_n)$. v_i refers to the number of packets in the i-th time window. Here we could treat the inflow and outflow packets seperately, but I believe that the inflow could describe the feature of object better. After that, we could calculate the similarity score with two time window vectors by getting the correlation coefficient of them. That is:

$$\begin{aligned} S_{Time\ Window} &= w_{Time\ Window} * \frac{Cov(\vec{V'}, \vec{F'})}{StdDev_{\vec{V'}} * StdDev_{\vec{F'}}} \\ &= w_{Time\ Window} * Corr(\vec{V'}, \vec{F'}) \end{aligned} \tag{3.3}$$

Two vectors represent as $\vec{V'}$ and $\vec{F'}$, also as $(v'_1, v'_2, ..., v'_n)$ and $(f'_1, f'_2, ..., f'_n)$. $Cov(\vec{V'}, \vec{F'})$ stands for the covariance of two vectors, which is $E[(\vec{V'} - E[\vec{V'}])(\vec{F'} - E[\vec{F'}])]$. $StdDev$ stands for standard deviation, calculated by $\sqrt{E[(\vec{V'} - E[\vec{V'}])^2]}$. And $Corr(\vec{V'}, \vec{F'})$ means correlation coefficient, the same as covariance divided by the multiplier of two standard deviations.

We used $w_{Time\ Window}$ here again and that is slightly different with $w_{Interval}$ used above. It also ranges from 0 to 1, calculated by divide the smaller number of inflow packets of the two vector with the bigger number of inflow packets. Weight is useful to filter out obviously irrelevant samples, and almost without any side-effect. The correlation gives us the information of the trends between variations of packets but not the absolute number of packets. Then weight could help us to introduce absolute number of packets into calculation. Actually, either weight calculated by number of intervals or by number of packets does not differ greatly. So they are somewhat interchangeable.

The time window divided attack results better than the interval method; we shall see that in the following section. But the interval method is much more robust than time window divided method. An abnormal long lag will make this sample completely

worthless in time divided method, but one or two retransmission does not hurt seriously in interval method.

3.6 Combine Two Methods

We have presented two methods before, and both of the two methods have their own suitable cases. It is hard for attacker to analysis each case and determine which method to use, so the combination of two methods are recommended to introduce as many factors as possible.

From intuition, there are equations like this:

$$S_{Combined} = S_{Interval} * S_{Time\ Window} \ / \ w_{Interval} \tag{3.4}$$

$$S_{Combined'} = S_{Interval} + S_{Time\ Window} \tag{3.5}$$

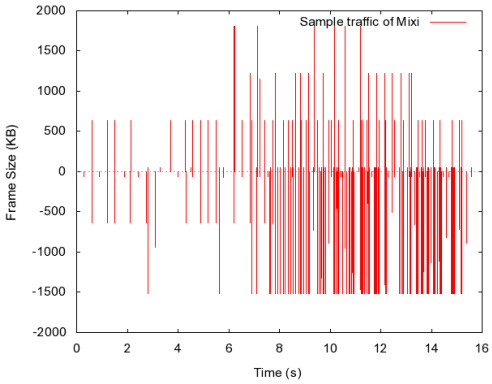
Besides these two basic formulas, we could also adjust each item's weight. Towards different samples, there may be different effective formulas, but we want to discuss in a more general case.

Compare Formula 3.4 and Formula 3.5, I will tend to use the first one for two reasons. First, the Formula 3.4 will give us a result between -1 and 1, which is more formal way and could still introduce other factors in future without change the range of result. Second, in my opinion, I think extreme case should be considered seriously. Compare to the similarity score of 1 and 0, the score of 0.5 and 0.5 maybe the better choice. (Although seem both of them are not the right choice.)

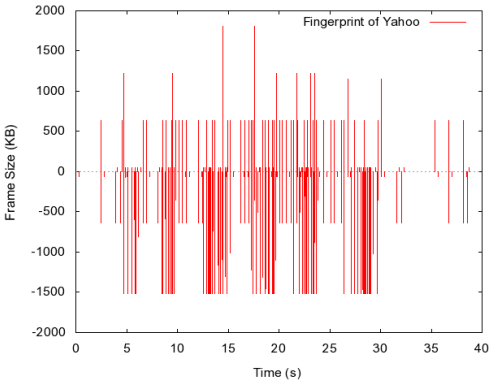
From experiment, the combination gave us better results; we will see them in the following section too.

3.7 Pity Hit

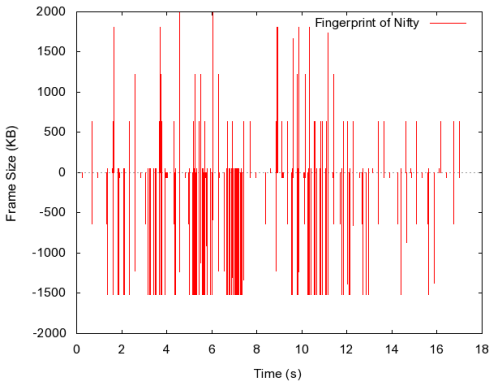
Sometimes, attacker does not need to fully depend on the system to decide which page user is browsing now. What he want is by using attack system, a few susceptible candidates could be reviewed manually (Maybe also with some assistant). Then the system relieves attacker's work load, and still keep a probably high success rate profit from human's experience and knowledge. So in this case, the system do not choose the highest similarity score from all the candidates, instead, top n candidates would be chosen for attacker to decide. If the correct answer falls in the top n candidates in an attack, we now call it a "pity hit". That means the attack still could be succeed due to the help from attacker.



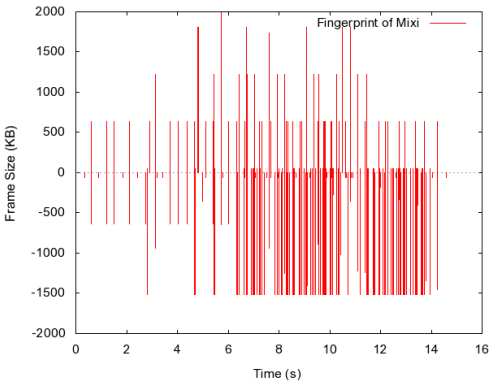
(a) Sample traffic from Mixi



(b) With Yahoo, $S_{Interval} = 0.916$



(c) With Nifty, $S_{Interval} = 0.898$

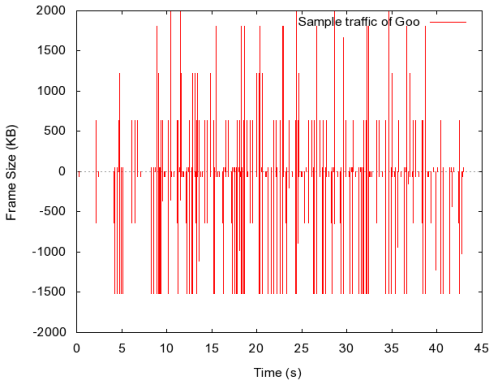


(d) With Mixi, $S_{Interval} = 0.897$

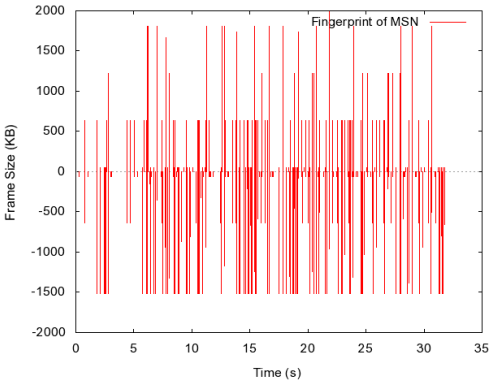
Figure 3.4: Pity hit example 1, sort by $S_{Interval}$

Here we give some observations to illustrate the importance of introducing pity hit. In Figure 3.4, we could see that sort by $S_{Interval}$, the correct answer only listed 3rd place. And by the definition, we could call it a "pity hit". But if we see the result from the picture, the difference in $S_{Interval}$ is small. From graph shape, we could easily decide that fingerprinting from Mixi is closest to our sample traffic. And that is what a real attack system works, by collaborating of human and algorithm.

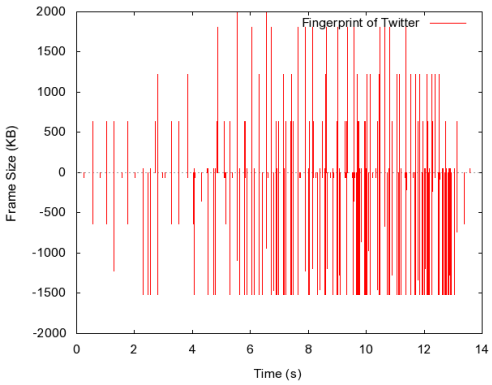
But things are not always so good to us. We shall see another example: Figure 3.5 illustrates another example which sorted by $S_{TimeWindow}$. In this example, it is really hard to say which fits better just by the graph shape. And fingerprint from MSN seems more similar to the sample traffic than the other two. But actually, our eyes could not read all the messages from the traffic shape. If we use $S_{Interval}$ to sort them, the $S_{TimeWindow}$ with Goo itself is as high as 0.985, and the result with MSN



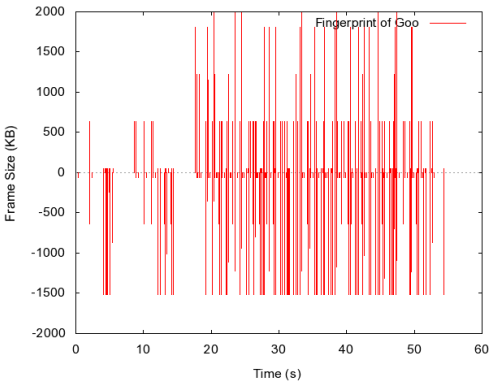
(a) Sample traffic from Goo



(b) With MSN, $S_{TimeWindow} = 0.916$



(c) With Twitter, $S_{TimeWindow} = 0.825$



(d) With Goo, $S_{TimeWindow} = 0.814$

Figure 3.5: Pity hit example 2, sort by $S_{TimeWindow}$

and Twitter is only 0.783, 0.640, respectively.

So from these two examples, we know that pity hit could help the attacker raise the success rate sometimes. But that does not means we could judge all the patterns by our eyes both for high workload and there is still information which could not recognized easily by human. With combination of two methods and human's assist, our attack could do really effective towards existed anonymity systems.

We will also see the result with and without pity hit in the evaluation section.

3.8 Other Applicable Situations

Although this attack is mainly designed towards Tor, it could also be applied in other situations.

First, it is not hard to see every anonymity systems with multiplexing or quantized cells could be attacked by our proposal. And even without multiplexing or quantizing, our proposal also works. You can treat all the connections as if they were one. But it would be ineffective for we discard some useful information by this process.

Second, this kind of attack can not only be applied attacking information regarding webpage surfing, but also other forms of network activities. For example, in instant chatting, there should be differences between one who talks quickly but every sentence is short and another merely talks but using long paragraphs. This kind of differences could be reflected in their traffic flows, although the significance may not be high enough to be detected.

What's more, our scheme does not only apply to the entry point of the path, but also the exit point. Imaging that if you are a curious server administrator who is running a system which accepts both anonymous and non-anonymous visits from anonymity systems, you could record the patterns when users visiting your sites in non-anonymous mode. And someday, for some purpose, a user visits your sites anonymously. Then you could use this scheme to guess which user it is. Just by comparing the historical patterns and the flows you observed.

We just simply described some other possible situations for the application of our proposal. Theoretically, for any kinds of activities with stable traffic patterns, our proposal could be a potential threat.

Chapter 4 Experiments and Evaluation

4.1 Environment and Data Collecting Method

We use Windump to capture the Tor packets (Version 0.2.0.34) on a PC with Intel Core2 Duo 1.86G, 4G RAM, Vista Business. We shall run the windump to observe the port 9001 on the host machine. Then we use Firefox which installed TorButton to surf the webpage. After a webpage is fully loaded, we stop capturing the packets. We use Wireshark to open the PCAP file, filter the obvious noise manually. More precisely, in a short period, all the connections raised from Tor are going through the same path. So most of the packets will obviously have the same destination address (Actually, this address refers to the first node in the path). And some packets with other destination addresses refer to other control packets used in Tor, like establishing new paths. After this process, a data is recorded. We also wrote some programs to analysis the captured data to make the calculation.

4.2 Evaluation of the Interval Attack

Data Analysis First, we shall use Alexa Ranking - Top Sites in Japan[1] to see how our method works in a practical environment. In Figure 4.1, we use n to represent the top n sites' mainpages we used to implement the experiment. We choose the top 20 sites to implement the experiments.

In the experiment, we choose top $n = 5, 10, 15, 20$ sites, and built fingerprint of the site. Then we surfed webpages and recorded the user activity vector, compared with the fingerprint, and guessed which website user is surfing. The success rate represents in the Figure 4.1.

From Figure 4.1, we could see that: the success rate is relatively high when n is small. With n increases, the success rate decreases significantly. There are several reasons for this: First, the information we used is limited, the fingerprint of the webpage is not so unique. So obviously the success rate decreases when n increases. Second, some pages are not suited for fingerprinting, like youtube[2], amazon[3]. The items on

[1] http://www.alexa.com/topsites/countries/JP
[2] http://www.youtube.com/
[3] http://www.amazon.co.jp/

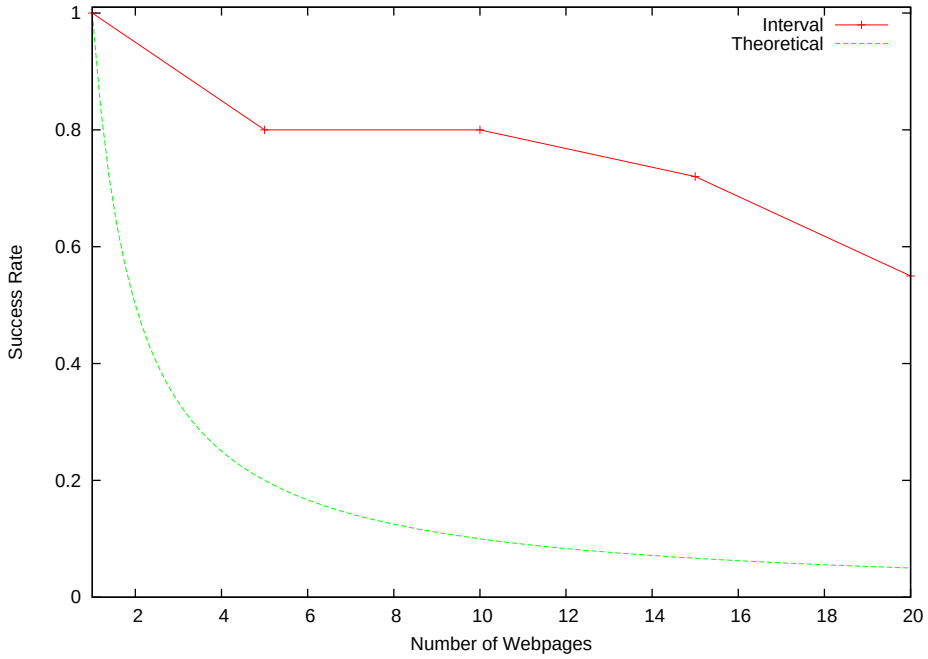


Figure 4.1: Success Rate in Different n

these sites would change from time to time, which hurts the consistency of fingerprinting. Other sites like Yahoo[4] will have ads change frequently, too. But compared with other parts of the page, the ratio of ads is not so large and we could just treat them as noise. Third, the pages we have chosen are all homepages, with the similar design, it increases the difficulty of distinguishing. Fourth, the noise in practical network affects the result a lot, and that's why we need to implement our method instead of just making the simulation. Last, there are some sites hard to see the difference but still be counted as different ones, like Google[5] and Google Japan[6]. This problem also exists in the distinguishing between the original page and phishing page. We will discuss the success rate in more formal way in the following section.

Theoretical Discussion In this part, we will discuss the effectiveness of this attack in theory. We will discuss two topics: The factors that related to success rate and make an estimate of how many webpages (or webpage groups) could be distinguished without too high error rate. Although we use All topics we discuss here could also be applied to other methods, so we will use S, w without pointing out it is used with

[4] http://www.yahoo.co.jp/
[5] http://www.google.com/
[6] http://www.google.co.jp/

interval or others.

First, Let us discuss with the success rate. We will use the method in [14] to show the attack success probability formally: We use $V \sim F$, to indicate that the attacker's test says that vector $\vec{V}$ and fingerprint $\vec{F}$ are from the same site. And we use $V = F$ to indicate that the event that vector $\vec{V}$ and fingerprint $\vec{F}$ are from the same site. We have the *false positive rate*, $Pr_{fp} = Pr(V \sim F | V \neq F)$, and *false negative rate*, $Pr_{fn} = Pr(V \nsim F | V = F)$, are both known. We can therefore obtain:

$$\begin{aligned} Pr(V \sim F) &= Pr(V \sim F | V = F)Pr(V = F) + Pr(V \sim F | V \neq F)Pr(V \neq F) \\ &= (1 - Pr_{fn})Pr(V = F) + Pr_{fp}(1 - Pr(V = F)) \\ &= (1 - Pr_{fn} - Pr_{fp})Pr(V = F) + Pr_{fp} \end{aligned}$$

Which leads us to obtain:

$$\begin{aligned} Pr(V = F | V \sim F) &= \frac{Pr(V = F \wedge V \sim F)}{Pr(V \sim F)} \\ &= \frac{Pr(V \sim F | V = F)Pr(V = F)}{Pr(V \sim F)} \\ &= \frac{(1 - Pr_{fn})Pr(V = F)}{(1 - Pr_{fn} - Pr_{fp})Pr(V = F) + Pr_{fp}} \end{aligned} \tag{4.1}$$

Suppose $Pr(V = F) = 1/n$, e.g., we are observe n sites and the adversary has no additional information about which site the user is likely surfing. Then, the success probability depends on Pr_{fp} and Pr_{fn}.

In the simplest case, we first assume the false positive rate and false negative rate are constant. Then, with $Pr_{fn} = Pr_{fp} = 0.1$ and $n = 10$, which means the user could surf 10 webpages and we've made all the fingerprints of them, we could get $Pr(V = F | V \sim F) = (0.9 \cdot 0.1)/(0.8 \cdot 0.1 + 0.1) = 50\%$. And if we improve Pr_{fn} and Pr_{fp} to 0.01, then with 10 webpages, the success probability is about 91.7%. As n rises to 100 webpages, this probability also falls to only 50%. With $n = 1000$, it is less than 10%.

But as we see in the evaluation above, the Pr_{fn} and Pr_{fp} rises with n. So, we will describe the false positive rate and the false negative rate as a function of n. We also use the assumption $Pr(V = F) = 1/n$ discussed above. Then the Equation 4.1 would be:

$$
\begin{aligned}
Pr_{Success} &= \frac{(1 - F_{fn}(n))/n}{((1 - F_{fn}(n) - F_{fp}(n))/n) + F_{fp}(n)} \\
&= \frac{1 - F_{fn}(n)}{1 - F_{fn}(n) - F_{fp}(n) + F_{fp}(n) \cdot n}
\end{aligned} \tag{4.2}
$$

Actually, it is almost impossible to make a reasonable function to reflect the relationship between n and the error rate, for it is affected greatly by the sites we have chosen. But we could assume Pr_{fn} and Pr_{fp} have a linear relationship with the increase of n, then from the Equation 4.2, we could see the numerator falls with n, and the denominator increases even faster, which will leads the success probability decreasing even faster.

The equations we listed above tell us if we want to increase the success rate, there are several points: First, to improve the accuracy, that is, decrease the false positive and false negative rate. Second, make the webpages we need to guess as few as possible, what means make the n lower. What's more, we assume the adversary knows nothing in advance. So the $Pr(V = F)$ equals $1/n$. But if in some situation, $Pr(V = F)$ is greater than $1/n$, which means the adversary gets some additional information from other ways, the success rate itself will also be raised.

Then, we shall come to how many webpages we could distinguish without high error rate, if not choose the webpages randomly but we could choose by ourselves.

Notice that the similarity S consists of two components, the relative interval ratio and the vector's dot product. First, we take a look at the relative interval ratio. We have implemented an experiment to get that the mainpage of Yahoo Japan have an average interval of 159.2105, with the standard deviation of 14.8495. Figure 4.2 shows the distribution of intervals of Yahoo Japan.

From our observation, the intervals of webpages often fall in the range from 50 to 600. We can choose the page freely here, webpages with more than 1000 intervals are not so rare in practical. But here we just want to make an theoretical estimate; we will choose the range of interval up to 600.

As our experiment about Yahoo Japan, the standard deviation is approximately 10% of the intervals, that means, with about $\pm$ 20% gap between two sites, there is about 95% chance the vector could be recognized correctly. Roughly speaking, there are $log_{1.4}(600/50) + 1 \approx 8.38$ slots for us to choose webpages with high detection rate.

Then we come to the dot product of vectors. In our implementation, the vector is limited to 5-dimension. Because intervals with more than 5 packets are so rare, intervals with more than 5 packets would be treated as one with just 5 packets.

Theoretically speaking, if we use 20% gap as we do in the discussion about interval,

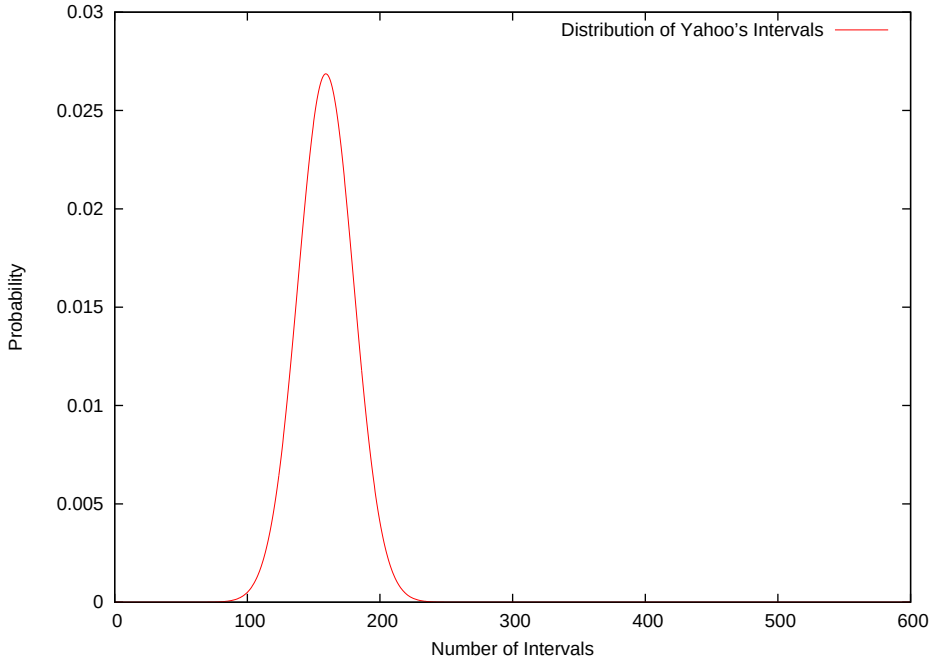


Figure 4.2: Distribution of Yahoo's Intervals

then there are a lot of available slots for us to choose, considering we have 5-dimension to do the permutation. But actually, in typical situation, the intervals with 1 or 2 packets dominated in the total dimensions, for there are a lot of transactions to be done(Also, in some extreme condition, such as file transferring, we could expect to observe a lot of long intervals). By our observation, in the situation with similar intervals, there are about 3 or 4 significantly different results. Combine this with the result about interval, we have approximately 20 to 40 available slots for choosing webpages to be recognized.

We have mentioned in the publication before that it is hard to improve this result, unless we could find some way to significantly reduce the noise. And, in the following sections, we shall see the improved results with time window method. So it is expected to be improved further by following research.

4.3 Evaluation of Collusion Threat Model on Tor

It is a little difficult to measure the effectiveness directly, but we could also evaluate this model with indirect methods. Since the fingerprinting attack is a kind of passive attack and we use this attack under the assumption of Dolev-Yao model, the encryption is considered perfectly. So the information we could used is really limited. For example,

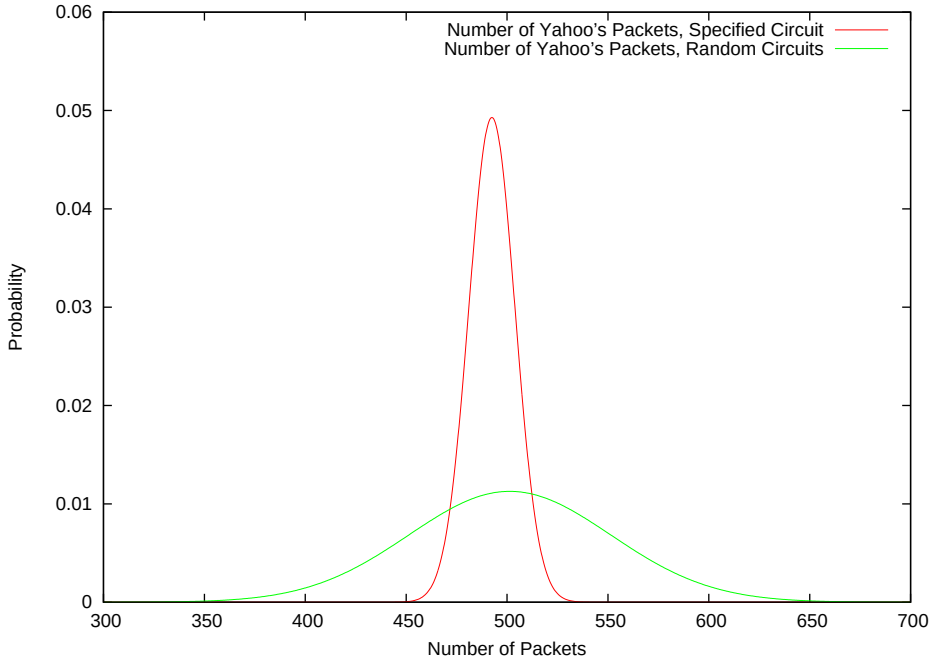


Figure 4.3: The Packets Number Distribution of Yahoo Top Page

time, number of packets, size, etc. If we could show the better distribution of them under this model, then it should be achieved better result by any possible fingerprinting attacks.

Let us make the fingerprinting calculation formula abstract to $S = Function(\vec{V}, \vec{F})$. It describes the process that user uses the fingerprint to determine whether a traffic pattern is according to a specified webpage or others.

Since in the Tor anonymity system, there are both end-to-end encryption and peer-to-peer encryption existed. All the information observers could get are the number of packets, the time of packet, etc. If we could show that the distributions of these features vary less in this model, it could reflect that fingerprinting attack will works more effectively here.

First we will see the distribution about the total number of packets. With a specified path, the packet dropping probability is relatively stable so the result will be closer to a specified number. On the contrary, with randomly selected path, the packet dropping probability varies greatly, it will cause the number of packets hard to expect. Figure 4.3 shows us the result intuitively and Table 4.1 shows us the comparison about standard deviation of packet numbers.

Then we come to see the distribution about the loading time of the webpage. Like the number of packets, when the circuit is decided, both the latency time and the

Table 4.1: Comparison about Standard Deviation of Packet Numbers between Random Circuits and Specified Circuit

Environment	Standard Deviation (on average)
Random Circuits	35.39179
Specified Circuits	8.84590

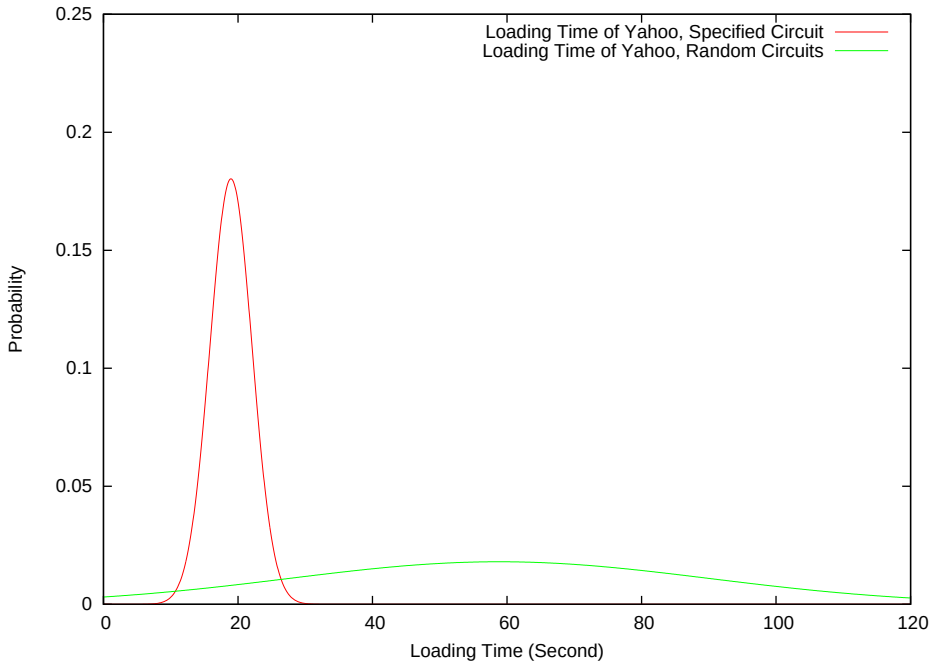


Figure 4.4: The Loading Time Distribution of Yahoo Top Page

dropping probability are also decided. When we put the results of different circuits together, huge differences caused by both geographical and network environment will impact greatly on the distribution of times. Figure 4.4 shows us the result intuitively and Table 4.2 shows us the comparison about standard deviation of transfer times. Notice that we have manually filtered some replay packets stuck somewhere by accident, which will greatly enhance the variety in the total transfer time.

Last, we will use the method we have presented above to show the improvement in the real attack scheme. We will choose the interval method to make the experiment.

If we have several fingerprints, we could calculate observed $\vec{V}$ with each $\vec{F_i}$ to get several similarity S_i, then we could sort all the S_i and make the assumption the user is surfing the webpage with the $\vec{F}$ correlated to the largest S_i.

It is obviously that with more stable network environment, the S will be higher

Table 4.2: Comparison about Standard Deviation of Transfer Time between Random Circuits and Specified Circuit

Environment	Standard Deviation (on average)
Random Circuits	22.13481
Specified Circuits	2.03525

Table 4.3: Comparison about $S_{Interval}$ between Random Circuits and Specified Circuit

Environment	Average Score	Standard Deviation
Random Circuits	0.85090	0.10501
Specified Circuits	0.97977	0.02113

compare to one calculated in the randomly chosen circuits. We have chosen the previous experiment data which collected from one circuit and the data without any distinguish about the collecting circuits. Table 4.3 shows the result and we can see that when the data are collected from one circuit, the similarity score is significant higher than in the randomly chosen circuits. This result reflects the advantage of collusion threat model.

4.4 Evaluation of Time Window Attack and Combination

In this section, we shall see the experiment result when we using time window attack, and also the combination of these two attack methods. As what we have done in 4.2, we use Alexa Ranking and choose top 20 sites to implement the experiments.

In the experiment, we choose top $n = 5, 10, 15, 20$ sites, and build fingerprint of the site. Then we surfed webpages and recorded the user activity vector, compared with the fingerprint, and guessed which website user is surfing by time window and combination methods. The success rate represents in the Figure 4.5.

From the Figure 4.5, we could see that if we choose the webpage whose fingerprints have the highest similarity score, time window shows better results than the interval method. And combination of two methods performed best in this situation. There are some points we shall notice here: First, time window do not always outperform the interval method, we could see that from the graph. Actually, both of them have their own suitable cases as we have discussed earlier. Second, the success rate does not

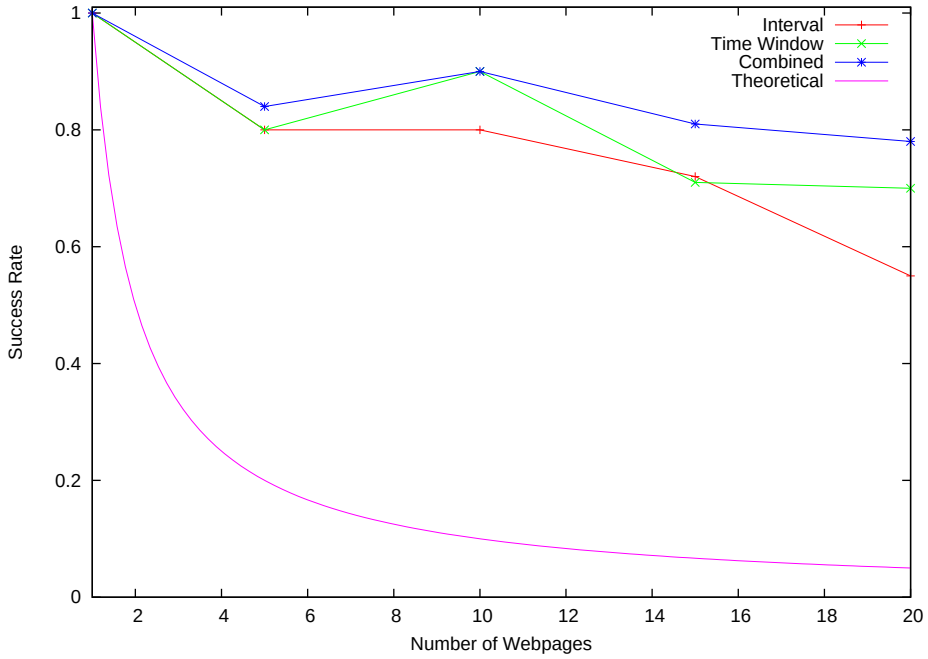


Figure 4.5: Success rate with Interval, Time Window and Combined methods

always drop as the number of webpages increases. Consider in the 5 webpages case, there are two pages are very close to each other. And when the number of webpages increases to 10, maybe the new pages are all easy to be distinguished. Then the success rate would be increased in the total.

4.5 Evaluation of Pity Hit

We have seen in Section 3.7, pity hit is useful when we want to implement an attack system with both convenience and high success rate. In this section, let us see the results when employing the pity hit into the experiment above.

We could see that by loosing the restriction - treat the situation that if the similarity score of correct answer falls in the highest 3 candidates, we see that is a successful attack, the success rate of all 3 methods are increased. But this time, time window becomes the weakest attack, then the combination. The interval method is most efficient method this time.

The reason of causing this problem is the result of interval method is far more robust than the time window method. Typically, there are two types of irregular events in the network transfer which may affect the analysis of traffic patterns. One is retransmission caused by packet losses, integrity checking error and other reasons.

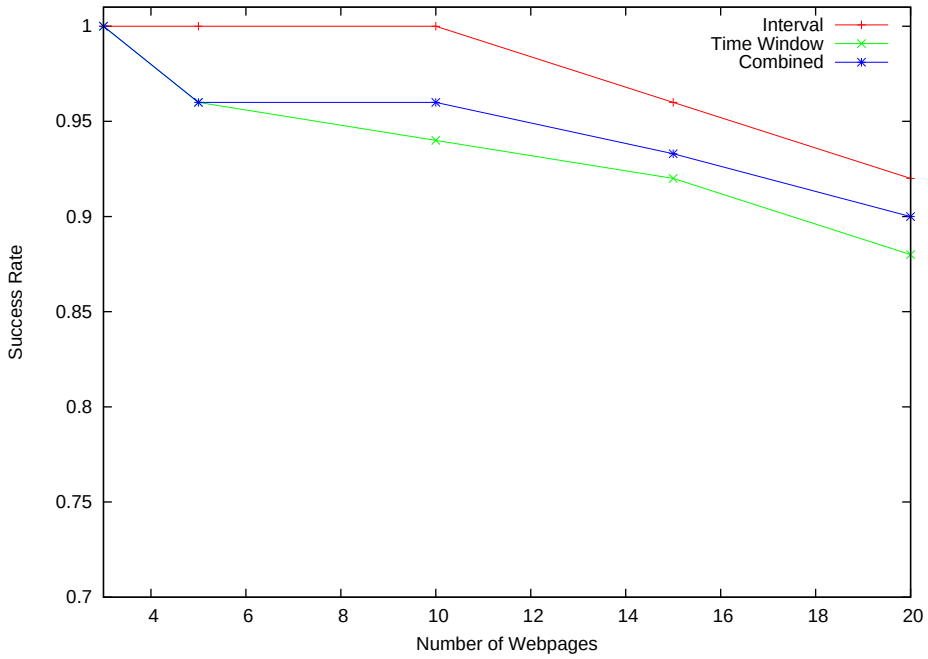


Figure 4.6: Success rate with pity hit

Then the TCP protocol would automatically require retransmission to ensure the integrity of the system. The consequence of that do not affect interval vector greatly, a retransmission will typically increase v_1 simply. And generally, v_1 is more than 50, or even bigger. So the similarity score will not be affected greatly unless the network environment is extremely bad.

Another event is time lag, that is also common irregular event in the network. Normally, a round-trip time of a packet is between several milliseconds to several hundred milliseconds. Sometimes, due to the fault in the network, packets would arrive after several seconds or even lost in the network. That will not affect interval method, but could complete destroy the score calculated by time window method. Image that we have a traffic pattern which last 20 seconds, with 4 splits, each time length will be 5 seconds. And if there is a lag inserted in it which lasts for 10 seconds, each time length will be 7.5 seconds and change the packet numbers in each time window greatly. Especially for we have to calculate correlation coefficient of two time window vectors.

According to these two reasons, we could know the reason why interval method outperforms time window method when we employing pity hit. But that not means time window method is valueless. Actually, time window method provides far better resolution in recognizing different webpages. Consider general cases, we still recommend

the usage of combined methods. It could provide a trade-off between two methods and sometimes have a greater success rate.

Chapter 5 Countermeasures

In this section, we will first discuss some general countermeasures to our attack and some countermeasures which are believed to be effective toward fingerprinting attack. Then, we will discuss the dummy packet method in detail.

5.1 General Discussion about Countermeasures

Change the Fixed Cell Size It is believed a longer Tor cell size will make it harder to attack, e.g. Increase the Tor's cell size from 512 bytes to 1024 bytes. But unfortunately, in our attack scheme, it will have little impact. Tor's fixed cell size gives the system some advantage in traffic analysis theoretically. But the protocol it uses is still built on TCP. So no matter what the cell size is, it could still be wrapped by TCP packet and be divided into 1500 bytes a packet in Ethernet. If there exists a scheme to analysis Tor's cell from TCP packets, this defense method could have some results, but not in our proposal.

Make Odd Requests Odd requests refer to some surfing actions which is unusual. For example, a user always surfing several pages meanwhile, restricting the scripts or pictures downloading, etc. If there is a page with vector $\vec{V_1}$ and another with vector $\vec{V_2}$, then when we view these two pages at same time, the adversary could get a $\vec{V_3}$ equals $\vec{V_1} + \vec{V_2}$, and $\vec{V_3}$ has no difference with the vector $\vec{V'}_3$ which has the same elements as $\vec{V_3}$, although it may be observed from one single page. Other odd requests like the restriction on downloading some specific files. Like the combination of two pages, it is also difficult for an adversary to match the characteristic from the fingerprint vector. Although this kind of defensive method seems to be so effective against fingerprinting attack, it depends on the user's action. But we cannot make the system's security depends how users use this system. It is dangerous to assume the users have the knowledge in security and will work in a secure way. Moreover, it is not hard to develop some kind of explorer plug-in to achieve this objective. Like TorButton, if we activate this plug-in, it will randomly disable some kinds of files in the current webpage, maybe forbid running script or download pictures. It will help us in the anonymity, but we do not think users would really accept some plug-ins like this.

Run Own Entry Node Entry nodes are also called "guard nodes". And people believed that they could guard your traffic from malicious nodes. First, it is not so useful to run a node by oneself when the adversary occupies the entry router, especially the time when they are allocated in the same Ethernet. Second, to run an own entry node and achieve the requirement of anonymity is very costly. That means, to make an adversary unable to distinguish the flows from a user. The own node may accept many connections from other users, which may hurt the usability of company's network and unacceptable. But running a node with only permitted user also makes this node meaningless. How to make the balance could be a question to network administrators.

Dummy Packets Defensive Dropping is a defensive method against timing attacks introduced by Levine et al. [14]. It employs the mechanism of dummy packets. The communication initiator constructs some of the dummy packets. These dummy packets are transferred on the path as normal packets. But to each packet, there is a probability P_{drop} to be dropped in each node rather than passing it on to the next node. If the number of dummy packets is randomly placed with a sufficiently large frequency, the correlation between every visiting will be greatly reduced. As we see in this theoretical discussion part, the increasing in the false positive rate and false negative rate will greatly reflected in the situation where we need to recognize object from a lot of webpages.

In a more general form, we could call the defensive dropping as a kind of dummy packets. Actually, the proposed way in the defensive dropping is not efficient enough against our attack. Since the fingerprinting is made of traffic pattern between user and webpage, and the feature of webpage is critical for attacker to make the guess in future. The webpage itself is hard to generate the dummy packets (for it could not distinguish whether the communication partner is using anonymity system or not), so the defensive dropping is almost useless here.

What's more, if there is a malicious node in the path, the node could easily drop all the dummy packets and reduce the effect of that defense mechanism. There are lots of adjustable parameters in this defense mechanism; we will discuss them in the following section.

Although it is an effective way to defend against not only end-to-end attacks but also fingerprinting attack, we must notice that it is a really expensive defense mechanism, especially in low-latency anonymity system. If the number of the dummy packets is relatively small, then these dummy packets are no more than normal background traffics. But with many dummy packets, it is unacceptable for consuming so many resources. What's more, use more dummy packets in sensitive connection is also not

a good idea for it gives the adversary a clear sign to notice the sensitive data transfer. So how to determine the sufficient number of packets will leave to be an open question for further research.

5.2 Dummy Packets in Tor Anonymity System

The object of introducing dummy packets into anonymity system is to "distort" the normal traffic pattern and make it indistinguishable. As we said before, there are a lot of parameters could discuss. In this section, we shall discuss them one by one.

The Type of Dummy Packets Roughly, we could distinguish the dummy packets into peer-to-peer dummy packets and end-to-end dummy packets. End-to-end dummy packets are generated by initiator of a message. End-to-end packets could be either encrypted or plain to the nodes in the path. Packets generated in defensive dropping method could be called end-to-end dummy packets without encryption, for every packet have a probability P_{drop} to be dropped in each node. So every nodes should be able to aware the packet itself is dummy or not. Or it could be encrypted as the data, after several decryption, it could be disposed at the exit. And it could also generated by any node when send back the message by encrypting dummy packets as the data packet then it could only be distinguished by the initiator.

Peer-to-peer dummy packets are generated by the nodes in the system. They are encrypted by the symmetric key between nodes so they are only invisible to the outsiders. All received dummy packets are disposed immediately, and new dummy packets are generated in the following circle. (Of course, keep the dummy packets in some probability and send to the next node is acceptable. But with different algorithm, we could achieve the same goal.)

Both peer-to-peer and end-to-end dummy packets have their own advantage and disadvantage. To peer-to-peer dummy packets, nodes in the path do not need extra computation but directly dispose them and then generate the new dummy packets. But if there is attacker in the node, he could just dispose the dummy packets from the previous node and omit the dummy packets when communication with the next node. Things could be even worse that the malicious node could make some tricky dummy packets, we see that as a potential threat.

To the end-to-end dummy packets, it should be generated by every nodes and wrapped as the data for no node really knows the position of itself. And generated dummy packets are encrypted again and again when they are sent back to the initiator. The message would become longer and longer for dummy packets are added by each

node. Although the default path length in Tor is only 3, actually we could increase that to 5, 10, 20 or even longer. Finally, the system will become unusable.

The Generation Rule Dummy packets could be generated both by time and by packet. They are distinguished by the rule of how to decide inserting a dummy packet. By time means at any time point, there is some possibility to generate a dummy packet and transfer it. And by packet means after any packet, there is some probability to generate a dummy packet and transfer it.

Although we have just demonstrated these two ways, we also want to mention that generate dummy packet after every packets with some probability is not safe. First, find some safe way to generate packets itself after a given event itself is not so secure. When the attacker knows the rule, seems they could be easily eliminated since they are not so "natural" traffic. We could not make sure that generate them by time could be the perfectly safe. But at least randomize in time is a better way to make some intentional fireworks after a shot.

Someone may argue that if the system is not in use, the dummy packets generated by time could be a waste. But first, well developed dummy packets could make attackers even hard to distinguish whether system is now in use or not. And if he could not even know if the system is in use, he could do nothing in the further attack. What is more, save bandwidth is not so meaningful when system is not in use. On contrary, when the system is busy, generate dummy packet after every normal packets with some probability will give system more burden than the other method. So we think do not consider the normal traffic but just generate dummy packets with some probability p all over the time.

What we want to point out is: p could be either constant value or some formulas, but there is no evidence to tell us that when the attacker knows the rule, some method is safer than other ones. So maybe the simplest way is the best. What we want is to find some practical and reasonable defense mechanism which could efficiently decrease the success rate of attackers, not to make the system perfect secure.

Experiment and Parameters It is hard to consider all the factors in the dummy packet employing here, so we want to do some experiments and just illustrate the efficiency of this idea. We will use data captured from Tor to make the experiments. In the experiment, we will randomly select one traffic pattern. If there is no dummy packet in it, either interval or time window method could have the answer of 1. Then we want to add some dummy packets into this traffic and calculate again with $S_{Interval}$ and $S_{Time\ Window}$. Of course, the lower S, we have the better protection.

In the experiments, we have discovered two parameters directly lead to the efficiency of the defense mechanism. The first one is *number of dummy packets generated every second*, or we could call it *density*. It is very trivial that as this number increases, the protection effect will also increase. But with the traffic emerging into the dummy packets, the marginal utility will also become weaker and weaker. And no doubt higher density will increase the cost of the anonymity system, and then the usability will also be hurt.

Another factor is the *coverage ratio*, here we define it as for the whole traffic pattern, how many parts in it could be inserted with dummy packets. Increase it will lead to higher cost and vice versa. But what makes this factor really interesting is that the higher coverage ratio will not always lead to the better protection.

In our attack framework, we have discussed mainly two calculation methods of similarity score, one is interval and another is time window. These two different attack methods have different sensitivity towards different coverage ratios. To the interval method, if the coverage ratio is low, that means many dummy packets are focused in a short period of time. The result is the length of a few intervals will be increased. But since we have limited the maximum element in an interval vector, the affected number of intervals is small, that will not change the result dramatically. For example, the change with coverage ratio which is 0.1 may only cause v_1 decreased by 2 and v_5 increased by 2. And most of the intervals still remains the same. On the contrary, if the coverage ratio is high, then more intervals' length are changed so the interval vector will be transformed greatly, so the $S_{Interval}$.

Let us see how the coverage ratio works in the time window method. When the coverage ratio is high, that means, almost in every time window, there would be approximately the same (at least the estimation would be same) number of dummy packets. And due to the calculation of correlation, if a series of numbers changed in the almost same amount of value, it then has really small effect on the correlation coefficient. But when the coverage ratio is low, the thing comes completely different. We will see that in this case, dummy packets flow into one time window and if the number of packets in that time window is fewest in the beginning, it may become the most one in the end. The correlation coefficient would change dramatically, it even may turn into minus. And just as a result of average, low coverage ratio is still quite good in the time window method.

We will treat the cost as the multiplication of these two factors. That is:

$$Average\ Cost\ =\ Number\ of\ Dummy\ Packets\ per\ Second\ *\ Coverage\ Ratio \quad (5.1)$$

Average		Coverage Ratio									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Number of Dummy Packets per Second	5	0.998	0.994	0.987	0.979	0.969	0.954	0.939	0.923	0.907	0.889
	10	0.996	0.987	0.971	0.949	0.923	0.890	0.854	0.817	0.781	0.747
	15	0.994	0.979	0.953	0.917	0.874	0.823	0.770	0.718	0.671	0.635
	20	0.992	0.972	0.937	0.890	0.834	0.769	0.709	0.647	0.597	0.557
	25	0.991	0.967	0.923	0.867	0.802	0.728	0.663	0.598	0.544	0.499
	30	0.990	0.964	0.914	0.851	0.779	0.699	0.629	0.558	0.503	0.463
	35	0.989	0.961	0.905	0.837	0.760	0.676	0.603	0.531	0.473	0.435
	40	0.989	0.958	0.899	0.826	0.743	0.658	0.582	0.510	0.451	0.413
	45	0.988	0.956	0.893	0.815	0.730	0.641	0.565	0.493	0.431	0.393
	50	0.987	0.954	0.886	0.807	0.718	0.627	0.550	0.477	0.415	0.379

Figure 5.1: Average $S_{Interval}$ without $w_{Interval}$ when employing the dummy packets

For example, if we have a density of 15 dummy packets per second and a coverage ratio of 0.6. That means the average cost would be 9 dummy packets per second. And assume all the dummy packets are 1.5 KB, and then the additional cost for one Tor connection is around 13.5 KB/s.

From the 5.1, we could see that keep the average cost constant, there is a tradeoff between density of dummy packets and coverage ratio. Interval method works well under the low density and high coverage ratio, but time window method works nice when the density is high and coverage ratio is low.

What we have omitted is weight. For $weight_{Interval}$, low coverage ratio will increase the weight a little and vice versa. For $weight_{Time\ Window}$, since it is calculated by the number of total packets, coverage ratio has no effect on it. The density will always increase change the weight. But for both the situation, weight would change significantly and the multiplication could effectively low the similarity score. We suppose that the when attacker knows the existence of dummy packet, he will just omit the weight and make the calculation.

Here we use a traffic pattern captured by Tor of the Yahoo's main page, and using two different similarity calculation methods without weight. The two parameters are adjusted to show us the effect of dummy packets under this circumstance. All the slots are calculated with 30 times of sampling and take the average value.

From these tables, we could see as the results are just run in tendency which is exactly what we have discussed above. The color in Table 5.1 and 5.3 show us the safe levels of combination with color tone. We could also refers to the Equation 5.1 and see these combinations: density 50, coverage 0.1; density 25, coverage 0.2; density 10,

StdDev		Coverage Ratio									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Number of Dummy Packets per Second	5	0.0016	0.0029	0.0051	0.0069	0.0096	0.0124	0.0145	0.0169	0.0194	0.0214
	10	0.0017	0.0043	0.008	0.0105	0.0144	0.0178	0.0201	0.0239	0.0274	0.0273
	15	0.0019	0.0051	0.0097	0.0136	0.0191	0.0243	0.0249	0.0268	0.0284	0.0262
	20	0.0023	0.0057	0.0116	0.0166	0.0205	0.0235	0.0244	0.0257	0.0277	0.0268
	25	0.0023	0.0063	0.0124	0.0171	0.0211	0.0254	0.0223	0.0239	0.0244	0.0241
	30	0.0025	0.0064	0.0117	0.017	0.0209	0.024	0.0228	0.023	0.0243	0.0239
	35	0.0026	0.0067	0.0125	0.0163	0.0189	0.0229	0.0205	0.0217	0.0233	0.0231
	40	0.0028	0.0064	0.0123	0.0167	0.0203	0.0214	0.0217	0.021	0.021	0.0212
	45	0.003	0.0063	0.0124	0.0172	0.0215	0.0211	0.0214	0.0211	0.0214	0.0213
	50	0.003	0.006	0.0117	0.0177	0.0202	0.0197	0.0206	0.0196	0.0201	0.0196

Figure 5.2: Standard deviation of $S_{Interval}$ without $w_{Interval}$ when employing the dummy packets

Average		Coverage Ratio									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Number of Dummy Packets per Second	5	0.937	0.932	0.929	0.917	0.916	0.917	0.924	0.928	0.960	0.984
	10	0.908	0.851	0.841	0.824	0.841	0.857	0.879	0.904	0.921	0.970
	15	0.850	0.779	0.758	0.734	0.765	0.798	0.830	0.876	0.910	0.965
	20	0.798	0.727	0.680	0.672	0.700	0.747	0.788	0.842	0.897	0.943
	25	0.764	0.678	0.621	0.642	0.651	0.706	0.753	0.815	0.886	0.929
	30	0.736	0.632	0.598	0.603	0.614	0.667	0.719	0.783	0.864	0.932
	35	0.707	0.600	0.570	0.568	0.590	0.635	0.694	0.757	0.842	0.913
	40	0.678	0.591	0.536	0.564	0.568	0.610	0.672	0.739	0.833	0.900
	45	0.652	0.578	0.526	0.557	0.552	0.588	0.652	0.720	0.816	0.878
	50	0.626	0.559	0.525	0.550	0.548	0.576	0.635	0.705	0.806	0.891

Figure 5.3: Average $S_{Time\ Window}$ without $w_{Time\ Window}$ when employing the dummy packets

StdDev		Coverage Ratio									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Number of Dummy Packets per Second	5	0.0013	0.0196	0.024	0.0277	0.0304	0.0281	0.0342	0.0532	0.0656	0.0317
	10	0.0039	0.0315	0.0335	0.0323	0.0355	0.034	0.039	0.0515	0.0745	0.0613
	15	0.0075	0.0329	0.0314	0.0295	0.0408	0.0423	0.0465	0.0508	0.0623	0.054
	20	0.0107	0.0325	0.0328	0.0359	0.0425	0.0431	0.0453	0.0531	0.0834	0.0859
	25	0.0136	0.0388	0.0395	0.0431	0.0482	0.0477	0.05	0.0569	0.0714	0.1096
	30	0.016	0.0339	0.0408	0.0424	0.0461	0.0449	0.0492	0.0608	0.0742	0.1129
	35	0.0181	0.0313	0.0407	0.0412	0.0411	0.0439	0.0475	0.0606	0.0804	0.1344
	40	0.0191	0.0287	0.0335	0.0395	0.0453	0.046	0.0499	0.065	0.0868	0.1465
	45	0.0211	0.0275	0.0347	0.04	0.047	0.0441	0.0497	0.0645	0.0896	0.1634
	50	0.0249	0.0363	0.0402	0.0411	0.0435	0.0459	0.0517	0.068	0.0921	0.1455

Figure 5.4: Standard deviation of $S_{Time\ Window}$ without $w_{Time\ Window}$ when employing the dummy packets

Table 5.1: Average and standard deviation of top matching similarity score with different methods

	$S_{Interval}$	$S_{TimeWindow}$	$S_{Combine}$
Average	0.931155	0.859238	0.829721
StdDev	0.046082	0.139079	0.143015

coverage 0.5; density 5, coverage 1. The result in interval method and time window method give us completely different tendency. The results from interval method are 0.987, 0.967, 0.923 and 0.889. Meanwhile, the results from time window method are 0.626, 0.678, 0.841 and 0.984. The two tables about the standard deviations tell us the result is basically stable, especially with the interval method.

Since two different methods give us different recommend combinations, we have to find some other standard to set up a threshold. Also we could observe that just by simply increasing the number of dummy packets per second, the result becomes better and better. But that also make the system eventually unusable. To solve all of these, we have to find some good trade-off.

We have also made the statistic analysis about the attack evaluation in order to get the average similarity score of the highest matching case. So we get the Table 5.1:

From the table above, we could see that for $S_{Interval}$, make the score less than 0.85 is safe enough. But for $S_{TimeWindow}$, due to the great variety, we recommend 0.7 as the safe threshold. More dummy packets could sometimes increase the similarity score, since the whole traffic is now emerged with dummy packets and in this sense they are

similar, too. We still want to point out that these results do not consider the weight, which could worsen results.

With the table and discussion above, we think coverage around 50%, approximately 20 dummy packets per second could be recommended. And by the Equation 5.1, we could estimate the cost is approximately 15KB/s on average for a Tor connection.

Chapter 6 Conclusion

In this paper, we have presented a novel fingerprint attack against the most famous anonymity system - Tor. Our scheme works by analyzing users' traffic flows in the anonymity system. We use outflow packets to divide a flow into several intervals, turn the traffic flow into vector, and give a formula to calculate the similarity of two vectors in this scheme. We also give several extensions towards our attack plan. It can be easily implemented by network administrators, governments, or ISPs. The experimental results showed our scheme to be very effective. The user's anonymity is really degraded by this simple and practical attack. Then, we have given both the extensions in the threat model and in the attack method itself. As we have discussed, this effectiveness still has a potential of being improved even more, but we have showed the different potentials of this attack.

Meanwhile, we have given a theoretical reasonable estimation of the effectiveness, showed the simple model of fingerprinting attacks on anonymity systems. Also, the following experiments have showed the improvement of extensions. We have discussed them in both theoretical and practical ways to help readers have the conception of effectiveness of our plan.

Finally, we discussed several countermeasures, especially focus on the dummy packets. Also, we have done some experiments on the dummy packets mechanism. The result showed the need for the use of dummy traffic in the low-latency anonymity systems. Since there is no low-latency system employed dummy packets now, it is critical to keep in mind that anonymity system is not as safe as people think. We strongly recommend that when design new anonymity system, employment of the dummy packets should be considered as an important defensive mechanism.

Acknowledgements

本研究にあたり、日頃から常にご指導を頂きました東京大学生産技術研究所 松浦幹太准教授に心から感謝致します。松浦先生には、研究の内容、進め方や考え方のみならず、研究に対する姿勢や着眼点など基礎的なところから教えて頂き、また学会参加など多数の各種活動の機会を与えて頂いたことで、修士生活を非常に有意義に過ごすことができました。

松浦研の定期ミーティングでの発表時に、様々な的確な助言をくださったり、研究内容に関して議論していただいたNHK 放送技術研究所の小川一人さん、警察庁情報技術解析課の岡田智明さん、産業技術総合研究所の田沼均さん、情報通信研究機構の野島良さん、始めとする今まで修士2年間の間にお世話になった松浦研ミーティングの参加者の皆様に感謝致します。

そして、私たちの研究活動が円滑に進むように日頃から尽力してくださっている教授室秘書の仲野さん、橋詰さん、小倉さんにも改めて深く感謝致します。

また、松浦研の技術職員である細井琢朗さん、特別研究員である北川隆さん、松浦研メンバーである楊鵬さん、Jacob Schuldt さん、松田隆宏さん、北田亘さん、渡邉悠さん、中井泰雅さん、千葉大輝君、Bongkot Jenjarrussakul 君、市川顕君、崔永錫君にも、日頃から研究室内での議論や、松浦研定期ミーティングにおいて、活発に議論をしたり、適切な助言をいただきました。改めて感謝致します。

そのほか、留学生活に世話になりました皆様、松浦研究室においてお世話になりました全ての方にお礼を申し上げます。皆様のおかげで松浦研究室での2年間の生活は素晴らしいものとなりました。

Bibliography

[1] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, *Proceedings of Information Hiding Workshop*, pages 245–257. Springer-Verlag, LNCS 2137, April 2001.

[2] Sambuddho Chakravarty, Angelos Stavrou, and Angelos D. Keromytis. Identifying proxy nodes in a tor anonymization circuit. In *Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, pages 633–639, Washington, DC, USA, 2008. IEEE Computer Society.

[3] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[4] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[5] Wei Dai. Pipenet 1.0. Post to Cypherpunks mailing list, January 1998.

[6] George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In *Security and Privacy in the Age of Uncertainty, International Conference on Information Security (SEC2003), May 26-28, 2003, Athens, Greece*, pages 421–426. Kluwer, 2003.

[7] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[8] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[9] Michael J. Freedman and Robert Morris. Tarzan: a peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and Communications Security*, pages 193–206, New York, NY, USA, 2002. ACM.

[10] Goodin. Tor at heart of embassy passwords leak, Sepemter, 10 2007.

[11] Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, February 1996.

[12] Andrew Hintz. Fingerprinting websites using traffic analysis. *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers 2003*, 2482/2003:229–233, 2003.

[13] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. How much anonymity does network latency leak? In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 82–91, New York, NY, USA, 2007. ACM.

[14] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems. In Ari Juels, editor, *Proceedings of Financial Cryptography*, pages 251–265. Springer-Verlag, LNCS 3110, February 2004.

[15] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Dong Xuan, and Weijia Jia. A new cell counter based attack against tor. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 578–589, New York, NY, USA, 2009. ACM.

[16] Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of Privacy Enhancing Technologies workshop*, volume 3424 of *LNCS*, pages 17–34, May 2004.

[17] Prateek Mittal and Nikita Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In Paul Syverson, Somesh Jha, and Xiaolan Zhang, editors, *Proceedings of the 15th ACM Conference on Computer and Communications Security*, pages 267–278, Alexandria, Virginia, USA, October 2008. ACM Press.

[18] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[19] Lasse Overlier and Paul Syverson. Locating hidden servers. *Security and Privacy, IEEE Symposium on*, 0:100–114, 2006.

[20] Andreas Pfitzmann and Michael Waidner. Networks without user observability – design options. In *Proceedings of EUROCRYPT 1985*. Springer-Verlag, LNCS 219, 1985.

[21] R. Pries, Wei Yu, Xinwen Fu, and Wei Zhao. A new replay attack against anonymous communication networks. *Proceedings of the IEEE International Conference on Communications, 2008. ICC '08.*, pages 1578–1582, May 2008.

[22] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[23] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society*, Washington, DC, USA, November 2002.

Publications

査読つき国際会議論文

<1> Yi Shi, and Kanta Matsuura. Fingerprinting Attack on the Tor Anonymity System. In *Eleventh International Conference on Information and Communications Security (ICICS2009)*, volume 5927 of *LNCS*, pages 425–438. Springer, Beijing, China, December 2009.

国内発表

<2> 施　屹, 松浦　幹太. 匿名通信システム Tor に対する指紋攻撃. In **コンピュータセキュリティシンポジュウム** *2009 (CSS2009)*, IPSJ, pages 877–882. 富山, 日本, October 2009.

<3> Yi Shi, and Kanta Matsuura. A Collusion Threat Model for Fingerprinting Attack on the Tor. In *2010* **年暗号と情報セキュリティシンポジウム** *(SCIS '10)*. IEICE, 高松, 日本, Jan. 2010.

<4> 施　屹, 松浦　幹太. 匿名通信システム Tor に対する時間拡張指紋攻撃と対策. In **コンピュータセキュリティシンポジュウム** *2010 (CSS2010)*, IPSJ. 岡山, 日本, October 2010. 発表予定.