On PAR for Attack

Chris Arnold carnold@cs.umn.edu
Rob Jansen jansen@cs.umn.edu
Zi Lin lin@cs.umn.edu
James Parker jparker@cs.umn.edu

May 17, 2009

Abstract

In The Onion Router (TOR) system, anonymity is provided by router services run by TOR users who volunteer their computational resources. Scalability concerns stem from the TOR design because volunteers lack an incentive to participate. A payment scheme has been previously introduced which aims at providing economic incentives for volunteers in hopes of increasing both reliability of and participation in TOR. We show that this payment scheme breaks sender - receiver anonymity through a traffic analysis intersection attack and is also vulnerable to traffic injection attacks, enabling TOR exit nodes to unnoticably cause an increase in traffic, and therefore payments, from the client. We simulate our intersection attack on the payment scheme and discuss directions for an improved design.

1 Introduction

Just as the Internet is playing an increasingly larger role in our daily lives, so will privacy play a larger role on the Internet. Traffic analysis allows governments and Internet service providers (ISPs) to monitor Internet usage and link users to their actions online. Traffic analysis is performed by monitoring the path between the user and network services and monitoring the types of traffic the user is sending. Therefore Internet users can be sure that their actions are *not* private. Anonymous systems, on the other hand, protect privacy by preventing linkability of users to their actions on the Internet. There are many situations where anonymity is important, including posting to online support groups and exchanging highly sensitive financial information. Governments and corporations might be interested in protecting their agents and employees by providing the capability of anonymous network traffic [1]. Users in countries where the government restricts access to certain websites might rely on anonymous communication to prevent linkability of their Internet usage. For most people, anonymity is important for maintaining personal security on the Internet.

When openly transmitting on the Internet, a large amount of information can be inferred even if the data being transmitted is hidden. An adversary can infer much information just by monitoring when and where a user sends data packets. Traffic analysis is continuously being performed by ISPs nationwide. For example, both the Chinese and British governments have been known to block wikipedia articles from their citizens [3, 2]. Traffic analysis attacks are still possible, even when using anonymous systems like TOR [11]. The TOR system consists of routers that anonymously transport data for clients. While a large number of attacks on TOR [13, 14, 17, 27] are known to gain various levels of knowledge about the client, increasing the number of users will provide greater diversity and security for TOR.

TOR is currently free to download and use [1]. As such, it relies on volunteers to run routers that provide anonymous transports for clients. If there were not enough routers available for TOR clients to use, the system would become impractical. It can be argued that the ratio of TOR routers to TOR clients will be maintained as new users join the system. However, this argument is weak because there is no guarantee that new users will be as motivated or invested in TOR as current users running routers, especially considering there is no current incentive to run routers. Additionally, the initial users of a system have more experience, investment, and are generally more involved in the system than new users, and therefore would be more likely to contribute to the system. As a result of the uncertainty of these arguments, the scalability of TOR is a great concern. A payment scheme was developed in order to provide incentives for routers contributing to TOR in the hopes of boosting the number of routers in the system. PAR [5] is a hybrid payment scheme that includes the distribution of anonymous payments and identity-bound payments to routers providing service for TOR clients. PAR is described further in section 3.

In the PAR protocol, there is a period of time in which the TOR routers will hold onto their coins before depositing them into the bank. This results in a trade-off between anonymity and double spending detection [20]. Routers can not simply deposit and verify coins in real time because of efficiency and doing so reveals too much timing information to the bank [5]. This timing information can be used to reconstruct the TOR circuits that were used at given time intervals. On the other hand, holding onto the coins for too long will allow more double spending without detecting cheaters. It has been shown that this double spending can be bounded to an acceptable amount for a single user [5]. However, if a group of colluding cheaters got together and double spent the same coin at the same time, the double spending amount increases drastically. This is a fundamental flaw in the PAR protocol, and our goal is to show that this is in fact a real problem and that a better design can be achieved.

The remainder of this paper is outlined as follows. In section 2 we discuss the origin of, and how similar systems provide, anonymity and discuss several attacks against such systems using a variety of traffic analysis methods. We also discuss electronic payments similar to those used in PAR. Section 3 provides an overview of the PAR payment scheme as applied to the TOR system. Section 4 outlines our attacks on PAR. Section 5 describes our TOR and PAR simulation design and provides a discussion of our simulated attack results. In section 6 we discuss an improved payment scheme, using online double spending detection, to work within TOR. Finally, we conclude in section 7.

2 Related Work

Past research associated with this work is focused around the two general areas of anonymity and electronic payments. We provide insight into these areas below.

2.1 Anonymity

The term "anonymity" refers to the unlinkability of identifying information to the elements of an anonymous set. Anonymous networks on the Internet aim to provide anonymity by not disclosing identifying information, such as name, address, or IP address, to either party during electronic communication. The goal of anonymity is to conceal who is communicating with whom. Anonymity on the Internet is challenging because communication, even anonymous forum posts, involves IP addresses which can be used to identify a user.

In recent years, there has been significant research into anonymous networks. Most ideas rely on mixing, introduced by Chaum *et. al.* [10]. The mixing concept involves hiding communication between participants of a system as well as the content of that communication. A more efficient system related to mixing is Crowds [23]. Crowds is an anonymous system that relies on the notion of "blending into a crowd". Requests are forwarded at random so that no member of the crowd can determine with absolute certainty the origin of a request. Thus, sender-receiver unlinkability is maintained. A benefit of distributing requests randomly to members of the crowd is the system will scale to a large number of users. However, encryption is not used between links in the system, so active and passive traffic analysis attacks are possible. Web MIXes [7] extended the idea of mixing to web-based communication, providing unobservability and anonymity of Internet access. One of the important additions in web mixes was the inclusion of constant dummy traffic between mixes and between clients and mixes. This dummy traffic is indistinguishable from real client requests and aids in the prevention of traffic analysis attacks against the system. The problem with this approach is that it results in extremely high overhead as each client using the system is continuously sending mostly unnecessary traffic. MorphMix [24], another web-based mixing system, eliminated the need for central mix servers by requiring that each user in the P2P network be a client and mix and the same time. The strength of the MorphMix system lies in its distributed architecture and its encryption scheme. Morphmix eliminated the use of dummy traffic, claiming it unnecessary due to the size and dynamism of the network. Instead of dummy traffic, they used ideas from Onion Routing [22, 28] including multiple encryptions through an anonymous tunnel to provide strong end-to-end encryption.

TOR [11] is a system similar to MorphMix. TOR, a low latency network that provides anonymous data transfer, was developed in an attempt to improve anonymity and security on the Internet. The TOR system combines the ideas of mixing and anonymous proxy servers. TOR was designed to be a usable low-latency system, therefore it does not include mixing (batching messages) or traffic shaping. TOR consists of *clients* and *routers* and uses the Onion Routing overlay network design where clients encrypt traffic in several onion "layers" and send it through a collection of routers. This collection of routers is known as a *circuit*. Each router peels off its layer of the onion

through decryption and forwards the data to the next router in the circuit. The last router in the circuit will send the traffic to the destination specified by the client. Each router in the circuit will only know its predecessor and successor in the path from sender to receiver thus preventing sender-receiver linkablility. Moreover, since the actual data being sent to the receiver is encrypted several times, only the last router in the circuit will know its content. Traffic analysis, although still possible (see below), is complicated by having each router in the system service multiple circuits at the same time.

There have been several attacks developed against these systems using a variety of traffic analysis methods [13, 6, 17, 14, 18, 21, 27]. A traffic analysis attack against TOR is described in [17] where timing can be used to leak information about the identity of connections. TOR does not claim to provide protection against global adversaries, however, the attack can be performed by relatively weak adversaries using the fact that the traffic volume of a stream of data traveling through a TOR router influences the latency of other data streams traveling through the same router. A mitigation strategy for this attack involves the use of stochastic fair queuing (SFQ) [15]. In SFQ, resources are fairly distributed over the circuits run by a node so that each queue receives a fair portion of throughput even if it has no active circuits (similar to dummy traffic.) This technique was shown to be effective at mitigating the clogging attack on the MorphMix P2P anonymity scheme. Traffic analysis strategies and mitigation techniques are important to consider when developing and analyzing protocols for anonymous systems.

2.2 Electronic Payments

Another important area is that of electronic payments, or e-cash. Some of the more important schemes is in the area of anonymous e-cash [4, 9], as ideas from these schemes can be carried forward into anonymous systems like TOR. It is important that anonymous systems do not use payment schemes that would reveal any information beyond what is already known by participants of the system. Chaum's scheme [9] provides untraceable and transferable coins that reveal cheaters who copy and double spend a coin. This scheme provides complete anonymity except in the case of double spending, which gives users an incentive to stay honest. An efficiency problem exists since in order to provide this double spending verification, the bank must be involved both in issuing the coin and again when the coin is spent (and immediately deposited by the user.) To limit the bank's role and increase efficiency, systems have been developed to allow the withdraw of multiple coins at once [8] and allow coins to be transferred several times before requiring deposit to the bank [19]. As a result of these schemes, a new double spending problem was created. When coins are not immediately verified, a time window exists in which cheaters can double spend coins without getting caught. Another system with the same problem was developed [12] that works like cashier checks where the bank is responsible for accounting and, to keep the system efficent, immediate deposit of the "checks" is not required. Other schemes [26, 16] involve aggregating coins together and depositing them in groups in order to decrease the number of transactions with the bank. Ideas from these schemes will be important in realizing a secure solution to payments in TOR.

3 Overview of PAR Scheme

PAR [5] is a hybrid payment scheme that includes the distribution of anonymous payments and identity-bound payments to routers providing service for TOR clients. The PAR scheme is essentially an onion routing protocol augmented with payment data. In PAR, clients located outside the anonymous network must pay the routers through which they send their encrypted data. This is done by withdrawing the correct number of anonymous coins from a bank and paying the first node in the transfer path. Then, each router in the path is responsible for paying its successor with identity-bound payments, which are based on micropayments [16, 25], and returning receipts to its predecessor. PAR uses these receipts, and digital signatures, to ensure that the payments are made correctly and accurately as intended by the client. The last router will receive payment from its predecessor, return a receipt, and forward data to the receiver as usual. PAR is described in further detail below, using the notation that follows.

Entities C is the client, R_i are TOR routers, S is the server. Both the client and server are generally located outside the TOR network.

Cryptographic material r_i are random numbers generated by the client and used in all subsequent coin generation as the message MSG travels through the circuit. The routers will depend on these r_i to generate coins for the successor. $AC(\cdot)$ and $IDC(\cdot)$ are anonymous coins (A-coins) and identity bound coins (ID-coins), respectively.

Cryptographic functions $H(\cdot)$, $H'(\cdot)$, $H_{AB}(\cdot)$, are cryptographically secure hash functions, where

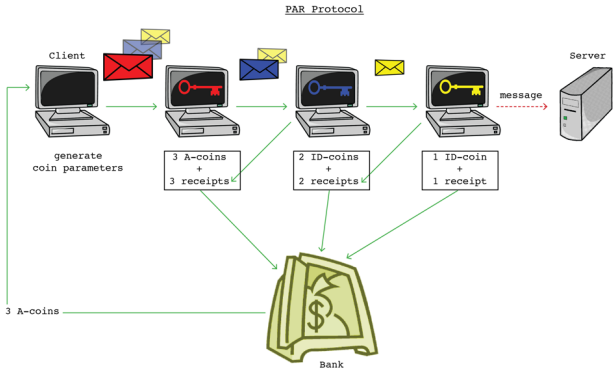

Figure 1: An example of the PAR protocol executed on a three-hop circuit

$H_{AB}(\cdot)$ is a hash function negotiated and shared between A and B. $E_{K_{AB}}(\cdot)$ represents cryptographically secure encryption with a key K shared between A and B while $D_{K_{AB}}(\cdot)$ represents decryption. $sig_A(\cdot)$ is a cryptographic signature by A.

3.1 Coins

Blind signatures (A-coins) In a blinded signature scheme, a signer signs a blinded message provided by a client. The client later recovers a valid signature from the blinded one. As a result, the client obtains a valid (msg, sig) pair while the signer learns nothing about it. The PAR scheme uses the blind signature as anonymous coins (A-coins). An A-coin is in the form of $sig_B(H(r))$.

A valid pair of (msg, sig) is treated as a depositable coin issued by the bank. The bank verifies the validity of the signature but is not able to link the presented signature with any signature it issued before. To prevent double spending by clients, the bank is required to record all the anonymous coins ever presented. Any client that obtains an A-coin is encouraged to immediately contact the bank to verify the coin and uncover users who attempt double spending.

Since A-coins are anonymous, it is difficult to discover who is to blame for double spending. There is no way to prove the real double spender. To migitate the problem, PAR protocol proposes to have every A-coin spender sign the coin. Therefore, once a double spent coin is detected, the coin receiver can show the Bank the sender's signature on the coin to clear himself/herself from blame.

Micropayments (ID-coins) An ID-coin in the form of $IDC_{A\to B}(r) = sig_A\{MC, H(r), B\}$ is employed to implement the payment between onion routers (ORs). An ID-coin is strongly bound to both the identity of the payer node A by the signature and the payee node B by including its ID in the message. The MC part (micro-coin [16]) can be verified without the help from bank.

Both A-coins and ID-coins are in the form of a signature on the hash value of a random number. When coins are paid, neither the hash value nor the random number (pre-image) is revealed to the payee. In the PAR scheme, in order to make the coin depositable, a payee must receive the random number as a *receipt* from some party. The PAR scheme embedded this coin and receipt scheme into TOR messages to provide the incentive of faithfully forwarding messages.

3.2 Protocol

Protocol setup Every TOR participant chooses one public-private key pair (sk^s_U, pk^s_U) for signing purposes and chooses another public-private key pair (sk^e_U, pk^e_U) for encryption purposes. The bank generates a blind signature key pair (sk^b_B, pk^b_B) for signing A-coins. The client establish a hash function H for coins, in addition to another hash function H' for integrity check. The client shares with each node in the circuit a key K_{CR_i} and two adjacent nodes in the path share a secret key $K_{R_iR_{i+1}}$. The client also shares with each router a hash function H_{CR_i} for receipt checking.

Protocol message Suppose client C chooses to build a circuit with t TOR nodes to server S through $R_1, R_2, \cdots, R_t$. For Tor node R_i, the message it receives from previous hop R_{i-1} (where $R_{i-1} = C$ in the case that R_i is the first router in the path) is in the following format:

$$E_{K_{R_{i-1}R_i}}(ID, CoinList, Sig(CoinList), MSG_{C\to R_i})$$

ID indicates the node id of the message destination. Normally $ID = R_i$. $CoinList$ is the payment R_{i-1} sends to R_i. The part of the signature on $CoinList$, $Sig(CoinList)$, is required when A-coins are sent and can be omitted when ID-coins constitute $CoinList$. $MSG_{C\to R_i}$ is further decomposed into the following format:

$$E_{K_{CR_i}}(HList, rList, H_{CR_i}List, R_{i+1}, MSG_{C\to R_{i+1}})$$

$HList$ are hashes of random values and used to generate payments for R_{i+1}. $rList$ are receipts which should be sent back to the previous hop, and $H_{CR_i}List$ are hashes of the receipt for R_i to ensure R_{i+1} returns the correct receipts. $MSG_{C\to R_{i+1}}$ are recursively defined in a similar way. For R_1, $rList$ is $\emptyset$ because C does not need receipts. Similarly, for R_t, $HList$ is $\emptyset$ because S does not need payment.

Protocol 1 Example of PAR communication protocol using a 3 hop circuit

Setup:

1: C generates random numbers r_1, r_2, r_3, r_4, r_5, r_6, and coins $AC(H(r_1))$, $AC(H(r_2))$, $AC(H(r_3))$

2: $MSG_{C\to R_1} = (R_1,\ AC(H(r_1)),\ AC(H(r_2)),\ AC(H(r_3)),\ sig_C\{H'(AC(H(r_1)),\ AC(H(r_2)),\ AC(H(r_3)))\},\ H(r_4),\ H(r_5),\ H_{CR_1}(r_1),\ H_{CR_1}(r_2),\ H_{CR_1}(r_3),\ R_2,\ E_{K_{CR_2}}(MSG_{C\to R_2}))$

3: $MSG_{C\to R_2} = (H(r_6),\ r_1,\ r_2,\ r_3,\ H_{CR_2}(r_4),\ H_{CR_2}(r_5),\ R_3,\ E_{K_{CR_3}}(MSG_{C\to R_3}))$

4: $MSG_{C\to R_3} = (r_4,\ r_5,\ H_{CR_3}(r_6),\ S,\ MSG_{C\to S})$

Execution:

5: $C \to R_1 : E_{K_{CR_1}}(MSG_{C\to R_1})$

6: R_1 : verify $sig_C\{H'(AC(H(r_1)),\ AC(H(r_2)),\ AC(H(r_3)))\}$ and generate coins $IDC_{R_1\to R_2}(H(r_4))$, $IDC_{R_1\to R_2}(H(r_5))$ with $H(r_4)$ and $H(r_5)$.

7: $R_1 \to R_2 : E_{K_{R_1R_2}}(R_2,\ IDC_{R_1\to R_2}(H(r_4)),\ IDC_{R_1\to R_2}(H(r_5)),\ sig_{R_1}\{H'(IDC_{R_1\to R_2}(H(r_4)),\ IDC_{R_1\to R_2}(H(r_5)))\},\ MSG_{C\to R_2})$

8: R_2 : verify $sig_{R_1}\{H'(IDC_{R_1\to R_2}(H(r_4)),\ IDC_{R_1\to R_2}(H(r_5)))\}$ and generate coin $IDC_{R_2\to R_3}(H(r_6))$

9: $R_2 \to R_1 : E_{K_{R_1R_2}}(r_1,\ r_2,\ r_3)$

10: $R_2 \to R_3 : E_{K_{R_2R_3}}(R_3,\ IDC_{R_2\to R_3}(H(r_6)),\ sig_{R_2}\{H'(IDC_{R_2\to R_3}(H(r_6)))\},\ MSG_{C\to R_3})$

11: R_3 : verify $sig_{R_2}\{H'(IDC_{R_2\to R_3}(H(r_6)))\}$

12: $R_3 \to R_2 : E_{K_{R_2R_3}}(r_4,\ r_5)$

13: $R_3 \to S : MSG_{C\to S}$

Protocol example Protocol 1 is a detailed example of the PAR protocol in a TOR 3 hop circuit. This example excludes communication with the bank, which happens both as part of the clients' A-coin generations and the routers' coin deposits. The notation is as presented above. More detailed versions of the client message setup and router message forwarding can be found in the Appendix in Protocol 2 and Protocol 3, respectively.

Protocol 1 is now discussed breifly in high level. The client C first sends R_1 a message containing A-coins, C's signature of the payment along with the message to pass along to R_2. R_1 then uses values C provides to generate payment in the form of ID-coins for R_2. Every R_i in R_2 to R_t will send back a receipt to R_{i-1}, along with generating payment in the form of ID-coins for R_{i+1} using values provided by C. The message for R_{i+1} is forwarded to the next router, including imbedded payment generation information, except when the next router is R_t (since R_t is the last router in the path and does not need to pay the server.) Each router also verifies payment information while forwarding the message. If any verifications fail, the protocol is aborted. Note that in this scheme, the TOR node R_t doesn't get the receipt from S. Instead, R_t will get receipts from C before or after the message delivery.

4 Attacks on PAR

In this section, we will describe three attacks on the PAR scheme. The first is a generic attack on payment schemes in TOR that we refer to as the *traffic injection* attack. Since the exit node has access to the (possibly) unencrypted messages going through the circuit, he can modify the content to cause the client to send more packets than intended and thus more coins than intended. Our second attack looks at the assumptions on how clients can collude to double-spend coins and shows that without online double spending checks, colluding clients can double-spend beyond what could be considered reasonable. Our last attack is the *intersection* attack. We show here that using the information from cashed coins and some network monitoring or colluding with routers the bank can break sender-receiver anonymity.

4.1 Traffic injection attack

One known problem with TOR is that the exit node is inherently trusted to forward traffic faithfully. Since the exit node necessarily has access to the traffic flowing through the connection, he is automatically set up for a man-in-the-middle (MITM) attack on the connection. This attack has been shown in the past to sniff e-mail credentials and other sensitive information. Here we consider the implications of injecting new content into unencrypted http traffic. Since without SSL clients browsers naïvely trust the content returned from a http get request, they will automatically download image links and execute javaScript code. Using this fact the exit node in the PAR circuit can inject many remote content links into pages returned to the client, which will cause the clients browser to issue many more http get re-

quests and, in turn, cause far more coins to be sent to the circuit than the client intended.

We note that this attack is highly sensitive to clients' browser behavior. Some of the most common remote content links are image links and javaScript XMLHttpRequest calls. Clients can mitigate the javaScript side of the attack by using any of several "noScript" extensions for their browser, and this is even encouraged since several javaScript calls can be used to undermine client anonymity. However, most of the modern web is useless without images and therfore most client browsers cannot be expected to block <img> tags in the document. Another way to mitigate this attack is using SSL to communicate with the web server. This way, the content returned to the client is guaranteed to be unaltered. This route is not completely secure since most websites don't utilize https and clients therfore can't rely on this mechanism for most of their traffic. But even if SSL is used, the MITM attack can still modify the traffic and present bogus certificates which it has been shown that most users will accept; however a full discussion of MITM attacks and SSL is beyond the scope of this paper.

4.2 Double spending attack

Here we briefly describe how a group of colluding clients can break the double-spending bounds on PAR. In the original scheme, there is a window of time where a client can double-spend coins before the entry node deposits them and detects the double spending. In the case of only one client, the double-spending can be bounded to a reasonable amount. However, we show a group of colluding clients breaks this bound. Consider the following scenario: a group of clients agrees on a set of coins to spend during a time window in between deposits, then the group of clients spends these coins during the time window with the double-spending only detected at the end of the time window when the routers try to deposit the coins. Now the double spending amount is multiplied by the number of colluding clients which can be quite large. A possible fix to this problem is discussed in section 6.

4.3 Intersection attack

We now present our main attack against the PAR system. First we note that the identity-bound ID-coins used between routers and subsequently deposited at the bank gives away information about which routers a given router was forwarding traffic to during a certain time period. Using this information the bank can perform an intersection style attack[17] to deduce a particular path through the PAR network. To execute this we note that clients circuits are refreshed every 10 minutes (meaning they now use a different three routers for their circuit) and the refresh time depends on when the circuit was opened. Thus, during a 10 minute time interval of a circuit we are interested in we do the following: for every deposit interval in the time period we look at which routers the first router was paying, and the routers they were paying to give us the possible circuits. In each deposit interval, some circuits going through the first hop will refresh and use a different circuit and some other circuit will be constructed that use the first hop as shown in figure 2. The key is that we look for the circuit that is constant for the 10 minute interval of the circuit we are interested in. Unless there is a circuit that was constructed through the first node at about the same time as the target circuit (highly unlikely), this attack will yield the single circuit of interest.

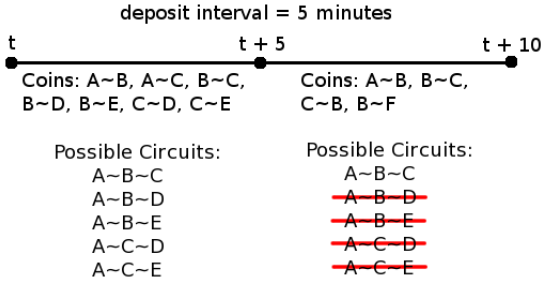

Figure 2: An example of the intersection attack with deposits being made every 5 minutes. The potential circuit list is reduced to a single circuit based on coin timng and identity information.

The above attack only discloses a circuit through the PAR network. To extend this to break sender-receiver anonymity, the bank has two options. First, the bank can collude with some of the routers or the bank can perform some passive traffic analysis of the network. By colluding with routers at the beginning and end of the circuit, the bank can verify that they were handling the same circuit and the routers together know the sender and the receiver. However, to perform this attack targeted at a certain user, the bank would need to collude with a large portion of the entry/exit routers in the network. Another option is the bank can use passive traffic analysis to link the sender and receiver. To link the sender with the circuit, the bank can enumerate the clients using PAR (this is because the clients must purchase their coins through the bank), and then using standard meth-

ods (e.g. ICMP pings) determine which clients are online during which time periods and use another intersection attack to determine which client was using the circuit of interest. To complete the attack, the bank can watch the traffic at the exit node to determine which server the client was communicating with during the circuits duration. We verify these claims empirically using a discrete-event based simulation of TOR and the PAR system which we detail in the next section.

5 Experimental Results

In this section we present our experimental results. We start by giving an overview of the discrete based simulation used to generate the data on which we ran the intersection attack. Next we detail the intersection attack, along with our assumptions and methodologies. Finally we present the results of running the attack.

5.1 TOR and PAR simulator

Our simulation system is a discrete-event based simulation of TOR and PAR. Since our intersection attack revolves around the timing of circuits and traffic, we ignore the cryptographic details in our simulation and focus on simulating client behavior and the timing of the deposits. Globally, we simulate a situation with 200 clients, 50 routers, and 50 servers. We simulate the run of this system for 30 minutes which includes a 10 min start up phase. Below, we detail how individual parties in the simulation behave.

Client Behavior At the start of the simulation a client selects a random starting time between 0 and 10 minutes at which time he begins his communications. At this time he randomly selects three routers to serve as the circuit, and a server with which to communicate. He then generates traffic according to the Pareto distribution as defined by:

$$\Pr[X > x] = \left(\frac{x}{x_m}\right)^{-k}$$

with parameters $x_m = \frac{1}{3}$ and $k = \frac{3}{2}$. He talks with this server for a time drawn uniformly random on 0.5 to 5 minutes when he selects another server at random to talk to. Finally, after 10 minutes of using the same circuit he refreshes the circuit and chooses three different routers at random to use.

Router Behavior A router in the simulation simply forwards packets and coins as specified by the created circuits. The router randomly assigns a delay uniformly on 5-10ms representing the computation time of the router. The router then stores the collected coin for later deposit. The deposit interval is left as an experimental parameter.

Server Behavior The servers simply receives packets from connection and responds with 1-3 packets after a delay of 5-20ms.

Network Behavior The network forwards packets with a uniform delay of 50 to 100ms.

Bank Behavior The bank simply accepts coin deposits from the routers and logs the type of coin and the payer/payee as appropriate.

5.2 Attack

We use the data generated from the above simulation as input for our implementation of the intersection attack. For the attack we assume either some collusion between PAR routers and the bank or some network monitoring by the bank by which the bank can observe when circuits are started with which entry node. This mirrors the assumptions as outlined in section 4 where we note the in order to break sender receiver anonymity, the bank must collude with routers or perform traffic analysis. Our original implementation of the attack proceeds as follows: for each circuit start up observed do the following; slice the 10 minute circuit duration into time periods corresponding to the length of the deposit interval, for the first time period use the ID-coin data to determine which routers the entry node was paying during the time period and the routers those routers were paying to build a list of potential circuits. For the remaining time periods, if a coin representative of a potential circuit is not observed in the time period discard that circuit as no longer being a potential circuit. The output of the attack is the list of circuits left after the discard phase.

This approach turned out to be inadequate however. Upon running the attack we discovered that depending on the deposit interval as much as 50% of the circuits had no possible PAR circuits discovered. The problem was that the above implementation relied on the assumption that a client would be sending a packet at least every time period. Upon examination of the raw data this turned out to not be the case (due to the random nature of sending packets as per the Pareto traffic distribution.) To adapt to this, we modified our implementation to account for this problem. Our solution was relatively simple; in the case that the original attack returned no possible circuits we look for the circuit that "missed" the least amount of time periods.

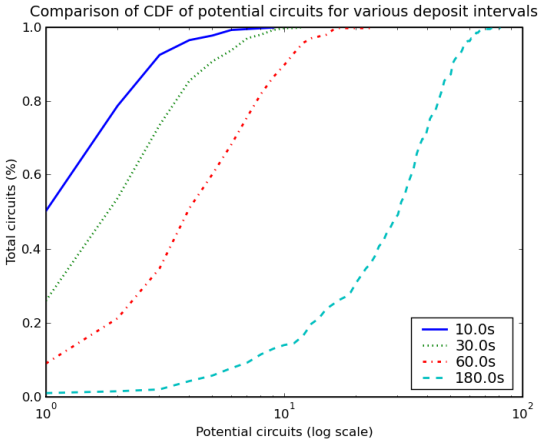

Figure 3: CDF of potential circuits discoverd by the intersection attack for deposit intervals of 10, 30, 60, and 180 seconds. The attack is less effective using higher deposit intervals.

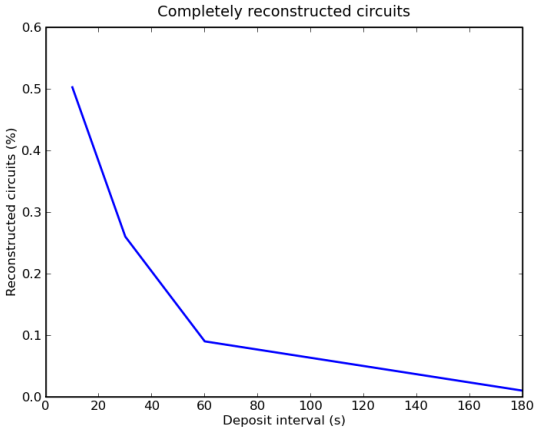

Figure 4: Percentage of reconstructed circuits per deposit interval. We consider circuits reconstructed when the list of potential paths forming the curcuit was reduced to 1.

5.3 Results and discussion

We now present the results of running the improved experiment in figures 3 and 4. In figure 3 we present a CDF showing the percentage of circuits with no more than x potential circuits. The different lines represent data from deposit intervals of 10 seconds, 30 seconds, 60 seconds (= 1 minute), and 180 seconds (= 3 minutes.) As noted before the data shows that as the deposit interval increases the effectiveness of our attack decreases. One of the main points of the data is that in order to significantly reduce the effectiveness of the attack, the deposit interval needs to be greater than 3 minutes. This means that the double spending bound is pushed far beyond what could be considered reasonable. In figure 4 we present the same data represented differently. This graph shows the percent of circuits reduced to one potential circuit as a function of the deposit interval. Again we note that in order to reduce the number of circuits de-anonymized to a negligible percentage the deposit interval needs to be increased to at least 3 minutes.

Our attack was successful at confirming the double spending problem as the main fundamental weakness in the design of PAR. The results clearly show the trade-off between the amount of double spending allowed and anonymity provided by a TOR circuit. This fundamental problem prevents the PAR payment scheme from being completely secure and therefore overwhelmingly supported. Ideally in an anonymous routing scheme, a user would not be required to deposit a coin within any specified time period. This would prevent the bank from using timing attacks since a given coin could be deposited at an indefinite and random time in the future. This would allow the level of anonymity provided by a TOR circuit to be maintained. However, double spending can not be simply ignored.

Although double spending is a significant problem, we feel that all is not lost. We have ideas for implementing a new scheme based on the PAR design that nullifies the double spending problem by eliminating the need for immediate verification by the bank. Our ideas for this new design are presented below.

6 Extensions

As seen by our attacks, the bank can get non-trivial information by watching the transactions of the signed microcoins. We propose a distributed verification scheme which provides online double spending detection while leaking virtually no additional information. The idea for this scheme comes from Osipkov et al. [20] with small modifications to fit the TOR network.

6.1 Witness overview

If we have online verification with a single verifier, that verifier can do a timing attack to recreate the paths. The solution is to employ online verification with multiple verifiers, called witnesses. In our scheme, these witnesses will simply be the other routers in the TOR network. Using this scheme, we

must ensure that a coin is bound to a specific witness to prevent double spending. Each witness is assigned a range of numbers and if the hash value of that coin falls in their range, that witness' signature is required for the coin to be deposited. The designations of witnesses to hash ranges is called the witness list. If a witness signs the same coin twice to allow double spending, that witness will to have reimburse the extra amount due with its own money. This requires all witnesses to provide credit with the coin distributer before the signing of coins is allowed. If a witness verifies too many double spent coins, it should be ejected from the system. A witness will not profit from acting maliciously since the amount it allows to get double spent will come out of the credit it provided.

Since TOR routers are signing each other's payments, routers with little downtime and few dropped packets are more beneficial to the system. The benefit to having a witness list is that it allows the network to exclude routers that are either frequently offline, busy or uncooperative by not signing. Routers that hinder the witness scheme can have their range of hash values reduced until they become more stable.

For this new payment scheme, we will not use the signed ID-coins due to the information that they leak. Instead we will use purely anonymous coins with a slight modification as described by Osipkov et al. [20]. This adds two expiration dates to every coin: a soft expiration where the coin is no longer able to be deposited but can be redeemed for another coin, and a hard expiration where the coin is worthless. The soft expiration is implemented to allow clients to get a new coin for witnesses that are possibly offline or too busy to verify the coin. Since the witnesses and the coin distributer need to remember the coins for double spending purposes, the expiration dates ensures that they do not get overwhelmed trying to remember if a coin has been spent.

6.2 Broker

In this section we analyze the usefulness of the broker described by Osipkov et al. [20] in regards to the TOR network. In their scheme, the broker distributes the coins and marks each coin with the witness list used. Additionally, the broker injects witness list version information into a coin when it is created. This allows a coin to carry its own designation to a witness list and thus doesn't require any witness list synchronization between every witness.

However, since we only want witnesses to be inside the TOR network to prevent information leakage, coins must be specifically bought from the broker for use inside the TOR network to ensure the proper witness list is used. The broker also becomes a single point of failure for the anonymity of the TOR network. If the witness list is skewed to make a subset of routers sign the majority of the coins, collusion between that subset of routers, possibly owned by the same person, results in the timing attack vulnerability. To prevent this, we must ensure that the majority of routers in the TOR network are each responsible for a moderate range of the witness list. A broker must also be very careful on how it validates complaints about uncooperative witnesses. We realize that malicious users or even malicious TOR routers might claim a particular witness is being uncooperative in order to skew the witness list. How these complaints are handled is outside the scope of this paper, but should be given consideration.

If we do not use a broker to insert the witness list into every coin, this allows the client to use a coin from any trusted bank. This requires the management of a witness list that is fairly synchronized between every TOR router. Depending on how often the witness list is updated, this could potentially be a difficult problem. If the witness list is managed to reduce uncooperative users, wherever this list is managed would once again become a single point of failure for the anonymity of the network. At the cost of an increased coin verification failure, the TOR network could simply split witness signature responsibility equally between all routers. Clients would not lose any money, because any coin that has an uncooperative witness can just be returned to the bank after the soft expiration date for another coin. Instead, having uncooperative routers on the witness list simply increases the average latency.

Double spending can still be prevented without using a centralized witness list at the cost of slightly more network traffic. When there is a witness list update due to a router possibly joining or leaving TOR, every router should ask both the current witness and previous witness if they had seen the coin already. While this doubles the amount of witness signature traffic, this double checking only needs to last until the maximum soft expiration time from a bank has passed. Once this time has passed, we can be assure no coins were double spent and only the current witness should be consulted. These updates could be planned to happen when the network typically experiences lower than normal usage.

A problem with incorporating coins that are spendable outside the TOR network is that a client might possibly only spend the coin once inside the TOR network, but also once outside. The TOR routers forwarding this traffic wouldn't find out until the coin was deposited, which is not done in an online man-

ner. To prevent this, the anonymous coins could be modified to have an identification number attached to them when created to signify how this coin is to be spent, instead of a witness list. Coins can only be redeemed if the depositor is valid for that identification number. This identification number could be in the blinded portion of the coin so the bank doesn't know how this coin is going to be spent to maintain client anonymity. If a client wishes to spend the coin in a different manner, the coin can be redeemed after the soft expiration date, but before the hard expiration.

7 Conclusions

In order to increase the number of routers in the TOR network, we think the implementation of a payment scheme is practical. We show a current payment scheme is vulnerable to a variety of attacks, including a simulated traffic analysis intersection attack. From the results of this attack, it is evident that the main weakness in the current scheme is the trade-off between allowed double spending and reduced anonmity. We propose to fix these flaws, and eliminate this trade-off, by implementing a distributed online verification solution. ID based coins are not used in our scheme due to the amount of information leaked when deposited to the bank. Instead we plan to use a modified version of the anonymous coin for all transactions.

There are still some outstanding problems with our anonymous routing payment scheme. The biggest problems are managing the witness list and accounting for data on the return path (from server to client). How can a client be charged for data that is flowing back from the server? One way would be to have the exit node notify the client through the circuit that data is ready to be delivered. The client could then send its payment to the circuit and receive the data. However, this method means that the exit node is responsible for buffering data while the client constructs and sends the necessary coin parameters to the circuit. While the exit node is expected to be servicing multiple circuits, this buffering problem is intensified. There is not a clear way to handle this situation, and more investigation is required to determine if an elegant solution exists.

References

[1] The Tor Project. `https://www.torproject.org/`.

[2] Wikinews. `http://en.wikinews.org/wiki/British_ISPs_restrict_access_to_Wikipedia_amid_child_pornography_allegations`.

[3] Wikipedia. `http://en.wikipedia.org/wiki/Blocking_of_Wikipedia_in_mainland_China`.

[4] R. Anderson. *Security Engineering*. Wiley New York, 2001.

[5] Elli Androulaki, Mariana Raykova, Shreyas Srivatsan, Angelos Stavrou, and Steven M. Bellovin. Par: Payment for anonymous routing. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 219–236, Leuven, Belgium, July 2008. Springer.

[6] A. Back, U. Moller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. *Lecture Notes in Computer Science*, 2137:245–257, 2001.

[7] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[8] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 302–321, 2005.

[9] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings of Crypto*, volume 88, pages 319–227, 1988.

[10] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

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

[12] Barry Hayes. Anonymous one-time signatures and flexible untraceable electronic cash. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 294–305, 1990.

[13] Andrew Hintz. Fingerprinting websites using traffic analysis. In R. Dingledine and P. Syverson, editors, *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*, pages 171–178, San Francisco, USA, April 2002. Springer.

[14] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. How much anonymity does network latency leak? *ACM Transactions on Information and System Security*, forthcoming 2009.

[15] J. McLachlan and N. Hopper. Don't Clog the Queue! Circuit Clogging and Mitigation in P2P Anonymity Schemes. *Lecture Notes in Computer Science*, 5143:31–46, 2008.

[16] S. Micali and R.L. Rivest. Micropayments revisited. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 149–163, 2002.

[17] S.J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *IEEE Symposium on Security and Privacy. IEEE CS, May*, 2005.

[18] S.J. Murdoch and P. Zielinski. Sampled traffic analysis by internet-exchange-level adversaries. *Lecture Notes in Computer Science*, 4776:167, 2007.

[19] Tatsuaki Okamoto and Kazuo Ohta. Disposable zero-knowledge authentications and their applications to untraceable electronic cash. In *CRYPTO '89: Proceedings on Advances in cryptology*, pages 481–496, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[20] I. Osipkov, E.Y. Vasserman, N. Hopper, and Y. Kim. Combating double-spending using cooperative p2p systems. In *Proceedings of the 27th International Conference on Distributed Computing Systems*. IEEE Computer Society Washington, DC, USA, 2007.

[21] J.F. Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. *Lecture Notes in Computer Science*, pages 10–29, 2001.

[22] MG Reed, PF Syverson, and DM Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected areas in Communications*, 16(4):482–494, 1998.

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

[24] 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 (WPES 2002)*, Washington, DC, USA, November 2002.

[25] R.L. Rivest. Peppercoin micropayments. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 2–8, 2004.

[26] R.L. Rivest and Adi Shamir. Payword and micromint: Two simple micropayment schemes. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 69–87, 1997.

[27] Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In Dieter Gollmann and Einar Snekkenes, editors, *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*, pages 116–131, Gjvik, Norway, December 2003. Springer.

[28] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 83–100, Berkeley, CA, July 2000.

Appendix

Included below are detailed versions of the actions of the client and each router in the PAR scheme. These protocols have been included for completeness, but are not necessary in understanding the main contributions of the paper. Protocol 2 is a detailed protocol of message setup that is performed by the client in the PAR scheme. Protocol 3 is a detailed protocol of steps the routers take in forwarding messages in PAR.

Protocol 2 PAR: client message setup

Client C wants to send Server S a message: $MSG_{C \to S}$
C sets up a circuit to S via R_1, R_2, ..., R_t.
for $i = 1$ to t **do**
  C picks $(t - i + 1)$ random numbers for R_i: $\{r_{(i,j)}\}$ $(1 \leq j \leq t - i + 1)$.
end for
/* C now recursively generates the onion message $MSG_{C \to R_1}$ */
for $i = 1$ to t **do**
  if $i == 1$ **then**
    $HList := H(r_{(i+1,\cdot)})$.
    $rList := \emptyset$
    $NextHop := R_{i+1}$
  else
    if $1 < i < t$ **then**
      $HList := H(r_{(i+1,\cdot)})$.
      $rList := r_{(i-1,\cdot)}$
      $NextHop := R_{i+1}$
    else
      $HList := \emptyset$.
      $rList := r_{(i-1,\cdot)}$
      $NextHop := S$
    end if
  end if
  $H_{CR_i}List := H_{CR_i}(r_{(i,\cdot)})$
  $MSG_{C \to R_i} := E_{K_{CR_i}}(HList, rList, H_{CR_i}List, NextHop, MSG_{C \to NextHop})$
end for
$AC(r_{(1,\cdot)}) := Generate\_AC(r_{(1,\cdot)})$
$ACList := AC(r_{(1,\cdot)})$
$C \to R_1 : E_{K_{CR_1}}(R_1, ACList, sig_C\{H'(ACList))\}, MSG_{C \to R_1}))$.

Protocol 3 PAR: router message forwarding

Setup: Router R_i parses incoming message $MSG_{Sender \to R_i}$:

1: $(ID, CoinList, Sig(CoinList), MSG_(C \to R_i)) = D_{K_{Sender R_i}}(MSG_{Sender \to R_i})$;

2: $(HList, rList, H_{CR_i}List, NextHop, MSG_{C \to NextHop}) = D_{K_{CR_i}}(MSG_{C \to R_i})$;

Execution:

3: **if** $ID \neq R_i$ **then**

4: abort execution;

5: **end if**

6: **if** $Sig(CoinList) \neq \emptyset$ **then**

7: check $Sig(CoinList)$ is valid; Otherwise abort execution;

8: **end if**

9: **if** $rList \neq \emptyset$ **then**

10: $R_i \to R_{i-1}$: $E_{K_{R_{i-1}R_i}}(rList)$

11: **end if**

12: **if** $HList \neq \emptyset$ **then** /* payments needed for next hop */

13: $IDC_{R_i \to R_{i+1}}(r_{(i+1,\cdot)}) := Generate\_IDC(H(r_{(i+1,\cdot)}), R_i, R_{i+1})$

14: $IDCList := IDC_{R_i \to R_{i+1}}(r_{(i+1,\cdot)})$

15: $R_i \to R_{i+1}$: $E_{K_{R_i R_{i+1}}}(R_{i+1}, IDCList, MSG_{C \to R_{i+1}})$.

16: $R_i \leftarrow R_{i+1}$: $E_{K_{R_i R_{i+1}}}(r_{(i,\cdot)})$.

17: R_i checks $r_{(i,\cdot)}$ versus $H_{CR_i}(r_{(i,\cdot)})$, making sure that receipts are authentic.

18: **else** /* No payment needed. Next hop should be the destination */

19: $R_i \to S$: $MSG_{C \to S}$

20: **end if**