"Mix-In-Place" Anonymous Networking Using Secure Function Evaluation

Nilesh Nipane, Italo Dacosta and Patrick Traynor
Converging Infrastructure Security (CISEC) Laboratory
Georgia Tech Information Security Center (GTISC)
Georgia Institute of Technology
{nnipane3, idacosta, traynor}@cc.gatech.edu

ABSTRACT

Anonymous communications systems generally trade off performance for strong cryptographic guarantees of privacy. However, a number of applications with moderate performance requirements (e.g., chat) may require both properties. In this paper, we develop a new architecture that provides provably unlinkable and efficient communications using a single intermediary node. Nodes participating in these Mix-In-Place Networks (MIPNets) exchange messages through a mailbox in an Oblivious Proxy (OP). Clients leverage Secure Function Evaluation (SFE) to send and receive their messages from the OP while blindly but reversibly modifying the appearance of all other messages (i.e., mixing in place) in the mailbox. While an Oblivious Proxy will know that a client participated in exchanges, it can not be certain which, if any, messages that client transmitted or received. We implement and measure our proposed design using a modified version of Fairplay and note reductions in execution times of greater than 98% over the naïve application of garbled circuits. We then develop a chat application on top of the MIPNet architecture and demonstrate its practical use for as many as 100 concurrent users. Our results demonstrate the potential to use SFE-enabled "mixing" in a single proxy as a means of providing provable deniability for applications with near real-time performance requirements.

1. INTRODUCTION

The lack of privacy on the Internet is well documented. Whether by neighbors [55], ISPs [60], advertisers [65] or governments [54], most online communications are easily observable by parties other than the source and intended destination. Unfortunately, the use of encryption is often not enough to guarantee privacy or anonymity. Simply knowing that two endpoints exchange messages may provide sufficient context to reveal sensitive information. Specifically, traffic analysis has repeatedly been demonstrated as an effective means of uncovering not only individual relationships, but also larger structures in corporate [44], social [35] and other [13] organizations.

Private communication over open networks has been studied for more than 30 years. A range of systems have been created during this time across a design space that includes tradeoffs in privacy guarantees, performance and threat model and has resulted in two general anonymous communication architectures – relay-based and superposed. The best known of the former is Tor [17], which is the intellectual descendent of Chaum's mix architecture [10]. Tor delivers iteratively encrypted messages through a series of proxies and achieves high levels of privacy in the presence of a large amount of cross traffic. This approach also provides high performance, making it ideal for applications such as web surfing. Superposed communications architectures provide cryptographically-strong guarantees of privacy without the multi-hop and cross traffic assumptions of systems such as Tor, but do so at great expense to performance. Accordingly, such systems are generally more appropriate for very low bandwidth applications such as ballot casting. An architecture somewhere between these two classes, in which the strong guarantees of superposed systems and the performance of relay-based systems without the need for multiple intermediaries could help facilitate anonymous communications for different classes of applications (e.g., private messages between members of a corporate board, deniable chatting between an executive and a recruiter in a coffee shop, etc). This architecture addresses that void.

In this paper, we develop a protocol and supporting infrastructure capable of providing strong guarantees of sender and receiver anonymity for applications with moderate performance constraints (e.g., chat). *Mix-in-Place Anonymous Networks* (MIPNets) attempt to retain the intuitive security properties of traditional mix networks while reducing the attack surface introduced by requiring the use of multiple intermediary proxies. Using *Secure Function Evaluation* (SFE), clients are able to send and are restricted to receive a subset of the messages stored in an *Oblivious Proxy* (OP). These exchanges also blindly but reversibly modify the appearance (but not the contents) of all other messages stored in the OP, making it appear to an adversary that the contents of all messages are overwritten during each exchange. Accordingly, an adversary is unable to determine which messages are actually modified, yet alone their source and destination. Whereas traditional mix networks rely on a cascade of nodes, *this approach achieves unlinkability by mixing communications in place through a cascade of functions, constantly and indefinitely perturbing the appearance of messages in the system.*

Intuitively, MIPNets work as follows. Clients use SFE to read and write to a central mailbox (the OP) in a round-robin fashion. Because each exchange changes the appearance of all messages in the mailbox, individual interactions can not be determined by the OP or any other party. However, realizing this seemingly simple system presents a number of significant challenges. First, our proposed approach must function like a superposed architecture and prevent the kinds of side-channel attacks that have made identification possible in current mix networks. We achieve this by building our architecture based on a ring topology, thereby excluding the majority of timing, temporal perturbation and frequency-based attacks. We also take a number of steps to make this architecture robust against actively malicious participants, who may attempt to pollute the contents of messages or deny service to other clients within the MIPNet. Finally, we significantly improve the performance of the underlying cryptographic operations and implement a number of optimizations in the hopes of making this architecture usable by classes of applications requiring strong guarantees of anonymity in environments where sufficient cross traffic is not a given and better performance than superposed systems.

We address these challenges through the following contributions:

- **Develop an architecture that provably guarantees the unlinkability of communications between participants:** We present a new architecture for anonymous communication based on Secure Function Evaluation. While a number of provably anonymous communications systems have been proposed previously, *this is the first use of SFE in this domain.* Nodes send and receive messages through an Oblivious Proxy, which is unaware of their contents, destination or veracity. Moreover, the appearance of all messages is blindly altered after each exchange, further obfuscating the attacker's view of communications.

- **Improve and tune performance of SFE primitives:** We build and characterize our architecture using a modified version of Fairplay [42]. We not only implement a more efficient oblivious transfer primitive for Fairplay, but also explore numerous compilation and run-time options to reduce execution time over the naïve use of SFE in our architecture by as much as 98.5%, *allowing these heavyweight cryptographic operations to form the basis of practical, performance conscious, privacy-preserving applications.* The use of these cryptographic primitives as the basis of near real-time applications has not previously been considered possible.

- **Implement and measure core architecture and build an anonymous instant messaging application:** We perform an extensive performance analysis of our proposed architecture. We then construct a sample instant messaging client running on top of our system and demonstrate the ability to process keypresses *as fast as they are entered* for groups of as many as 100 users.

We note that MIPNets are not intended to be a replacement for traditional mix networks such as Tor [17]; rather, they target a different portion of the application space. MIPNets are designed to provide higher assurance of anonymity for classes of applications that require it (e.g., chat in hostile networks with low cross traffic) and are willing to make tradeoffs to achieve these guarantees.

2. RELATED WORK

Anonymity in computer networks has been studied since the inception of such networks. Outside of systems with a trusted and centralized authority [2], two classes of solutions have arisen. Note that these solutions differ significantly from trusted anonymous proxies [8, 56, 78], which operate as a single-hop filtering point and can therefore potentially be coerced into revealing the link between a source and destination, and from anonymous publication and storage systems such as Freenet [12], Free Haven [16] and Publius [73].

Applying cryptographic modifications to traffic through a proxy or *mix* in an untrusted network was first proposed by Chaum [10]. In this scheme, clients select a series of mix nodes, called a *cascade*, through which their messages should pass. Each message is then encrypted with the public key of each mix in the reverse order of the path to its destination. As such messages traverse the cascade, each mix node decrypts the message and exposes the next layer of the encrypted packet and forwarding instructions. While originally suggested for store and forward protocols, the mix architecture was eventually realized for all traffic via Onion Routing [59, 68]. This approach also motivated the creation of a number of other related efforts [51, 30, 14, 25, 6, 24, 15, 47, 20], with Tor [17] being the most widely used and studied of all such systems. Extensions to these schemes include the use of universal reencryption [27, 29], steganography [31] and mix rings for performance [9]. Crowds [61] and Hordes [39] similarly forward packets among a series of nodes.

While providing anonymity against a number of adversaries [69, 21], mix networks have become the target of an increasing number of attacks. These attacks introduced a number of techniques allowing adversaries with limited knowledge and control of a network to link sender and receiver. Through attacks on timing [79, 76, 38, 49, 74, 48, 64, 57, 75, 67] and other vectors [43, 3], such identification may be practically possible.

Chaum also suggested a second class of anonymous communications mechanisms with the Dining Cryptographers Problem [11]. This technique allows a sender to anonymously transmit a single bit as follows: Alice and Bob (who wants to transmit a single bit to Charles) flip a coin in secret. Alice reports the result of the flip to Charles. As the message Charles receives will be the XOR of all of the bits reported by the participants, Bob's report of the coin flip depends on the bit that he wishes to send. For instance, if Alice reports 0 (heads) and Bob wishes to transmit a 1 (tails), Bob claims the flip resulted in a 1. From Charles' perspective, each party is equally likely to have lied about the result, thereby protecting the sender's identity. A number of systems have attempted to implement such *DC-net* protocols through a variety of communications mechanisms [52, 18, 63] and with resistance to collisions and maliciousness [72, 7, 71, 70, 28]. Herbivore [66] addresses many of these issues by broadcasting all messages to all hosts across a star topology and scales through the use of cliques. Such networks are susceptible to denial of service and statistical analysis attacks, wherein nodes in different cliques communicating frequently can be correlated with higher accuracy. The Pynchon Gate [62] and pMixes [45] use Private Information Retrieval (PIR) protocols to achieve receiver anonymity. Senders place messages into mailboxes in a common node where receivers use PIR to retrieve them without a passive adversary being able to determine the content delivered to the destination. However, this approach is limited in a number of ways, including that sender anonymity is not guaranteed and the potential for timing correlation attacks.

3. SECURE FUNCTION EVALUATION

3.1 Background

Before presenting the details of the MIPNet architecture, we provide an overview of the cryptographic primitives used to communicate between clients and the OP.

Secure Function Evaluation [42, 37, 1, 26, 22, 23, 40, 53, 41, 33] allows two or more parties to execute a joint computation traditionally requiring the oversight of a trusted third party without any external intervention. More formally, participants Alice and Bob have input vectors $\vec{a} = a_0 \cdots a_{n-1}$ and $\vec{b} = b_0 \cdots b_{m-1}$ and wish to learn $f(\vec{a}, \vec{b})$ without revealing any information about their inputs that can not be inherently inferred from the output of $f(\vec{a}, \vec{b})$. As an example, SFE creates "circuits" that solve the Millionaires' problem [77], wherein two millionaires want to know who has more money without revealing their actual wealth. Assuming that each millionaire enters their net worth honestly to a function asking the high level question "Do I have greater worth than the other participant?", SFE can return the correct Boolean values to each party without disclosing either input.

While these cryptographic mechanisms have been understood for decades, their uptake and use in real systems has been slow. In particular, the use of such protocols has long required in-depth knowledge of the underlying cryptographic constructions and compelled interested parties to implement their own libraries. The Fairplay compiler [42, 5] greatly simplifies such efforts. Fairplay allows programmers to specify their protocol in a high-level language and then creates a garbled/encrypted Boolean circuit encoding a SFE-version of the desired function. In an exchange between Alice and Bob, Bob uses Fairplay to encode circuits for himself and then transmits a set to Alice. Both parties input their data, exchange their results and then uses them to compute the answer.

The security of these operations is guaranteed through the use of *Oblivious Transfers* (OT) [58, 77, 50]. A 1-out-of-n OT protocol allows Bob to learn one of the n pieces of data possessed by Alice without Alice learning the identity of the specific object. OT protocols also allow for Alice to prevent Bob from learning the contents of the remaining $n-1$ pieces of data. Fairplay implements an optimized 1-out-of-2 OT protocol proposed by Naor and Pinkas [50], which is based on the Diffie-Hellman problem. This protocol is secure in the random oracle model, which is implemented using the SHA-1 hash function.

Two parties engaging in an SFE exchange using Fairplay perform a single OT for each input into the circuit. After these exchanges, Alice is able to evaluate the garbled circuit without further interacting with Bob. Malicious behavior by Alice is defended against by both the security of the OT protocol and the use of SHA-1 to encode the secret used to encrypt the circuit. The same constructions also prevent Bob from malicious behavior, assuming that the circuits encoded by Bob are correct.

The authors of Fairplay recommend the use of a cut-and-choose protocol in the event that Bob's creation of the circuits can not be trusted. In particular, Bob can be required to create m garbled circuits for the function f and send them to Alice. Alice then randomly selects the circuit she wishes to use for her exchange and requests the secrets corresponding to the $m-1$ other circuits. Alice can then independently evaluate the correctness of the ungarbled circuits and can execute the remaining circuit with probability $\frac{1}{m}$ of maliciousness.

3.2 Extensions to Fairplay

Fairplay not only provides an OT primitive on top of which SFE can be built, but also makes it straightforward to implement supplemental OT schemes should improvements become available. We implemented an additional k-out-of-n scheme based on a Two-Lock cryptosystem. Such schemes are often more efficient than related OT mechanisms.

A Two-Lock cryptosystem is defined as follows: Assume that there are two encryption algorithms A and B. Alice encrypts a message m with secret s using scheme A and Bob encrypts the same message with key s' using scheme B. If $A_s(B_{s'}(m)) = B_{s'}(A_s(m))$, then Alice and Bob can be ensured that the contents of the messages they exchange remain confidential without actually sharing a key. Using this scheme, Alice encrypts a message m and sends the quantity $X = A_s(m)$ to Bob. Bob encrypts the received message, resulting in $Y = B_{s'}(X)$, which he then sends to Alice. Alice decrypts Y, creating $C = A_s^{-1}(Y)$ and again sends this value to Bob. Bob then decrypts C such that $m = B_{s'}^{-1}(C)$, thereby recovering the message m initially encrypted by Alice.

The above method can be used to perform k-out-of-n OT as follows. Alice encrypts n messages such that $X_0 = A_s(m_0), \cdots, X_{n-1} = A_s(m_{n-1})$. Bob selects a subset k of the n messages and encrypts them as $Y_0 = B_{s'}(X_{k_0}), \cdots, Y_{k-1} = B_{s'}(X_{k_{k-1}})$. Alice then receives the k quantities of Y, which she can not identify because Bob has encrypted them, and decrypts them resulting in $C_0 = A_s^{-1}(Y_0), \cdots, C_{k-1} = A_s^{-1}(Y_{k-1})$. Finally, Bob decrypts $C_0, \cdots, C_{k-1}$ and reveals the k selected messages.

Two-Lock cryptosystems can be implemented based on either the RSA or Discrete Log problems. While both approaches provide sufficient protection of the encrypted data, systems based on the RSA problem can be made to run more efficiently through the use of small values of e. Because the 0s and 1s are encoded as long strings of bits of length greater than $|N|$ in Fairplay, this optimization is not susceptible to small exponent attacks. From the RSA-based scheme proposed by Huang et al. [32], Alice sends Bob $X_0 = m_0^e, \cdots, X_{n-1} = m_{n-1}^e$. Bob selects k random numbers, one for each message he wishes to ultimately receive, and sends Alice $Y_0 = X_{k_0} \cdot k_0^e, \cdots, Y_{k-1} = X_{k_{k-1}} \cdot k_{k-1}^e$. Alice responds by sending Bob $C_0 = Y_0^d, \cdots, C_{k-1} = Y_{k-1}^d$, from which Bob recovers the messages $m_{k0} = k_0^{-1} \cdot C_0, \cdots, m_{kk-1} = k_{k-1}^{-1} \cdot C_{k-1}$.

The 1-out-of-2 OT scheme implemented with Fairplay requires a total of three modular exponentiations by the sender and two modular exponentiations by the receiver. Moreover, this approach requires two random number generations and a total of four message transfers. The RSA-based scheme also requires three modular exponentiations by the sender, but these operations are faster given our use of small values of e. The receiver side need only perform a single modular multiplication and one modular division. This approach also requires only a single random number generation and also requires the exchange of four messages. Accordingly, we expect to see a slight but measurable difference in the per-OT efficiency of our approach. We present our experimental results in Section 5.

4. SYSTEM ARCHITECTURE

In this section, we present the details of the MIPNet architecture. We begin by discussing the intuition behind our design and formalize the desired system guarantees. We then present a simple version of our architecture that protects against a globally passive adversary and assumes that all participants in the network are benign. Our second model provides additional protections against active attackers. We then discuss methods for graceful scaling and characterize the probability of collisions in the second system and prove that the previously defined guarantees hold in MIPNets.

Note that we focus on the design of the MIPNet architecture and, for the time being, assume that all clients participate in the network for long periods of time. Due to space limitations, we discuss membership management issues in greater detail in our technical report.

4.1 Design Intuition

Mix networks are an attractive model for anonymous communications for a number of reasons. The use of multiple layers of encryption offers an intuitive mechanism for obscuring the relationship between source and destination. However, it is often what happens *between proxies* in such systems that allows an adversary to link two endpoints. For instance, by injecting additional traffic into a proxy suspected to be in the path of a specific flow, an adversary can observe increased latency between source and des-

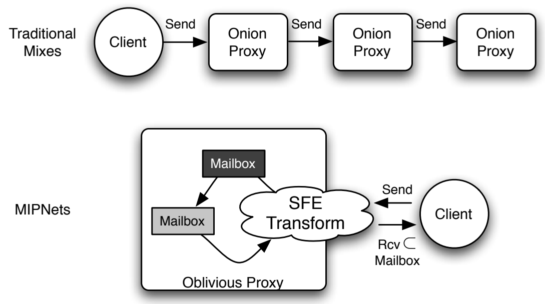


Figure 1: A conceptual comparison of traditional mix networks and MIPNets. Note that at every "Send", an adversary has an opportunity to perturb traffic. Mixing in place allows for such attack vectors to be largely removed.

tination [19]. Moreover, flows can be watermarked by perturbing packet interarrival times, allowing the adversary to identify the participants of a flow with a high degree of accuracy without injecting additional traffic. While the use of multiple intermediaries reduces the probability of a single proxy being able to determine the sender and receiver, it also increases the attack surface of the entire system.

The MIPNet architecture attempts to reconcile these issues by *eliminating the need to use multiple proxies*. The use of only a single proxy would appear flawed as no number of layers of encryption would prevent a proxy in possession of all of the necessary keys from linking source and destination. However, this issue can be overcome through the use of SFE between clients and the proxy. Instead of giving the OP messages that it can decrypt, interpret and forward, we simply require it to hold messages in a mailbox. Clients use SFE exchanges not only to send and retrieve the message stored within a subset of the slots in the mailbox, but also to decrypt and reencrypt the entire mailbox. Because of the properties of SFE, this final operation can take place without clients revealing the key to the OP or the OP revealing the entirety of the mailbox to each client. It is this cascade of *functions* that provides for anonymous communications in MIPNets. This "middle-path" allows us to enjoy much of the simplicity of relay-based anonymous communication while not having to rely on multiple proxies. Figure 1 highlights this difference.

The work on anonymous buses [4] and flash mixes [34] are the most similar related efforts to ours; however, our approach is novel in a number of ways. Unlike the former, the use of SFE makes it so that no client is ever able to see the entire contents of the mailbox. Second, while buses are secure in the honest-but-curious model, our solution is secure in the presence of a malicious adversary. Unlike both schemes, our approach also prevents the mixing node from knowing the relationship between messages before and after mixing. Finally, the performance evaluation and parameter characterization of our system is in much greater depth than previous works.

4.2 Desired System Properties

More formally, our system strives to provide the following guarantees: Let $\mathcal{C}$ be the universe of all possible clients and $\mathcal{S}_{\mathcal{A}} \in \mathcal{C}$ be the set of clients currently connected to the OP.

Definition 1. Sender Anonymity: Let client $c \in \mathcal{S}_{\mathcal{A}}$ be the sender of message m_i, denoted $c = sender(m_i)$. A system provides sender anonymity iff a passive adversary can determine that $c \in \mathcal{S}_{\mathcal{A}}$ and $sender(m_i) \in \mathcal{S}_{\mathcal{A}}$, but *not* that $c = sender(m_i)$ with probability greater than $\frac{1}{|\mathcal{S}_{\mathcal{A}}|} + \epsilon$, where ϵ is a negligible value.

Definition 2. Receiver Anonymity: Let client $c \in \mathcal{S}_{\mathcal{A}}$ be the receiver of message m_j, denoted $c = receiver(m_j)$. A system provides receiver anonymity iff a passive adversary can determine that $c \in \mathcal{S}_{\mathcal{A}}$ and $receiver(m_j) \in \mathcal{S}_{\mathcal{A}}$, but *not* that $c = receiver(m_j)$ with probability greater than $\frac{1}{|\mathcal{S}_{\mathcal{A}}|} + \epsilon$.

Definition 3. Unlinkability: A system provides unlinkability if it provides both sender and receiver anonymity as no two messages can be attributed to any client.

4.3 Basic Architecture

The simple version of the MIPNet architecture, presented for ease of understanding, works as follows: an Oblivious Proxy OP engages a client C_i in an SFE exchange. As its input, OP provides a mailbox M, which contains all communications currently being exchanged between members of the MIPNet. The mailbox itself contains a number of slots, each consisting of a two bit vector, for each of the n participants. The first bit of the slot, $M_{i,0}$ (the "read bit"), signifies whether or not the vector represents real communication. The second bit of the slot, $M_{i,1}$ (the "data bit"), represents a bit of data being sent to a client. Note that if $M_{i,0} = 0$, the value of $M_{i,1}$ is of no consequence to the receiver. This allows participating clients to inject cover traffic when they are not attempting to communicate with another participating node. C_i inputs its identifier i, at least one vector R_i and the intended destination j of R_i.

Instead of simply using OTs to prevent OP from learning the slot read by a client, we use SFE to gain anonymity of reads and to make the message and slot written by the client indistinguishable from the other messages in the mailbox. We achieve this additional guarantee as follows. All nodes participating in the MIPNet share a keystream k, and enter the current and "next" value of k as an input to the exchange. The keystream is used to blindly decrypt and reencrypt (via XOR within the SFE exchange itself, which is natively supported by Fairplay) all of the slots in the mailbox. Because of the properties of SFE, we can perform these operations without exposing the contents of the remaining $n-1$ slots *not* read by C_i.

As output, C_i receives the decrypted string of bits stored in M_i and OP receives a new set of pseudorandom bits M'. While C_i has blindly changed the *appearance* of all of the slots within the mailbox M, it has only changed the *content* of a single slot (as guaranteed by the circuit). Because all slots appear to have been overwritten from the perspective of OP, C_i is equally as likely to have written to any one of the slots. Accordingly, a passive adversary able to see both M and M' can not guess which message was written by C_i with probability greater than $\frac{1}{n} + \epsilon$.

OP then engages C_{i+1} and continues to service all remaining clients in order based on the ring topology of the MIPNet. This service model removes the ability to infer communications between two clients based on the frequency and temporal relationship of reads and writes.

It is important to note that the assumption of a shared key is not necessarily unrealistic. Even with such an assumption, the inability to read the entire contents of M limits the amount of data a single malicious client can leak per round. However, we remove this assumption in the next subsection to make MIPNets more robust.

4.4 Improved Architecture

The basic MIPNet architecture works well given a number of assumptions. Most importantly, it requires that all clients are well behaved and that they do not attempt to deny service by always transmitting messages to all or a subset of the other clients in each round. Additionally, it assumes that all clients will protect the keystream used to encrypt all communications between members. Such expectations are *not* necessarily realistic given that the anonymity of

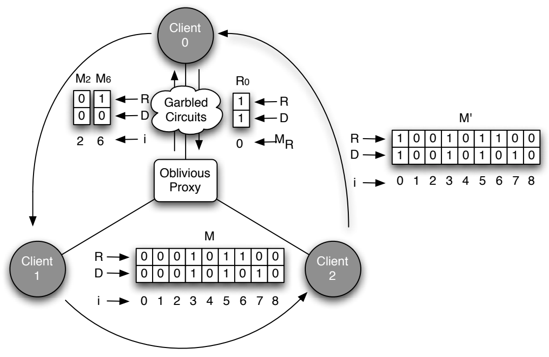


Figure 2: Improved architecture for MIPNets: C_0 and OP exchange messages (encryption removed for example). M' is shown to the side to show the output as seen by OP - note the only change from M occurs in slot 0. When the contents of the mailbox are reencrypted via the SFE exchange, it is not possible for OP to determine which slot has been updated by C_0.

communications in currently deployed systems can be breached using a number of more active techniques [79, 76, 38, 49, 74, 48, 64, 57, 75, 67, 43, 3]. Accordingly, we must make such a system more robust against the real adversaries it is likely to face. We specifically seek to make collusion to link or leak communications and denial of service attacks more difficult for adversaries.

4.4.1 Improved Mechanisms

A compromised client and malicious OP could collaborate to determine the pairs of communicating legitimate clients in the simple MIPNet architecture. The malicious client C_M could give OP access to the keystream k shared between all participating nodes. The OP could then simply decrypt M after each exchange and determine the destination, content and intent (i.e., whether or not the read-bit was set) of the message sent by that client. We address this concern through the use of shared keys between adjacent client nodes. Instead of encrypting M using a k shared between all participants, each client C_i will encrypt the contents of M using the pairwise keystream $k_{C_i,C_{i+1}}$ it shares with the next recipient. This approach makes the leakage of communications far more difficult with an arbitrary compromised client as only C_{i+1} can successfully extract messages from M. We discuss additional mechanisms to reduce the threat of multiple colluding clients in our extended technical report.

A number of techniques to address the denial of service issue have been proposed in superposed anonymous communications systems. Through channel reservation, k-anonymous protocols and a variety of additional techniques [72, 7, 71, 18, 70, 28], such attacks can be reduced. We instead propose to mitigate such attacks through the use of additional storage and a modification to the previous protocol. Specifically, we expand the size of the mailbox M to contain a total of $O(mn)$ bits for a system supporting n users, where m is a coefficient decided upon by the system. Instead of writing to a constant destination, clients select the destination slot based on a secret shared between them. For instance, a client C_i communicates with another client C_j in round S by placing a message in slot $M_R = MAC(S, k_{C_i,C_j})\ mod\ (m \times n)$. During the next exchange, these two clients communicate via $M'_R = MAC(S{+}1, k_{C_i,C_j})\ mod\ (m{\times}n)$. We discuss how m is selected and characterize the probability of collisions in the next subsection.

Like the basic protocol, OP selects M as its input to the exchange. C_i inputs the vector R_i it wishes to send and selects the slot to which the message should be written. Because multiple clients can now legitimately communicate with C_i concurrently, C_i calculates the $n-1$ mailboxes from which it should read using the hashing method described above. C_i also inserts the current and "next" values from the keystream $k_{C_i,C_{i+1}}$. As output, C_i receives $n-1$ slots and OP receives M'. Note that even if a participating adversary were to attempt to deny service by overwriting mailboxes intended for other legitimate nodes, it would be unable to determine the intended destination of the communication because of the randomization of slots within M.

Figure 2 provides an overview of this architecture without the use of encryption to ease understanding. C_0 sends the bit 1 to C_1, signified by the both the Read and Data bits of R_0 being set to this value. C_0 also specifies M_0 as the destination as this was the slot calculated using the previously mentioned technique. OP inserts the entire mailbox M as its input. Through the use of our SFE circuit, C_0 receives M_2 and M_6, which contain a read bit of 0 from C_1 and a read bit 1 and a data bit 0 from C_2, respectively. OP receives M', which differs from M only in one slot (M_0). Note that when encryption is being used, all of the contents of M' will appear to have been changed from the perspective of OP.

We note that the above protocol description does not provide any guarantees of collision detection and retransmission within the architecture itself. Accordingly, we require that such functionality is specified by the application and demonstrate the use of such mechanisms in our extended technical report.

4.4.2 Communication Characteristics

Our "spread-spectrum" inspired mailbox selection approach prevents malicious clients or a compromised OP from regularly overwriting the messages destined for a single node. However, legitimate collisions remain possible. We characterize the probability of such events by further understanding the relationship between the number of clients n, the mailbox multiplication coefficient m and the number of messages s a client sends per exchange with the OP. Because more than one client can send to a single participant in this system without necessarily colliding with other messages, we assume that each client reads $n-1$ messages per exchange with the OP. Additionally, while clients can send s messages per exchange, only one can be sent to a specific participant.

We focus on the probability of collisions during the steady-state operation of the protocol. When a client C_i wants to send messages to s other clients, at most $s(n-1)$ slots will be occupied by unread messages. Each client will therefore have at least $mn - s(n-1)$ open positions to fill with its new messages. Before sending any messages, C_i reads all of the messages sent to it. We can be certain that all s messages sent by C_{i+1} during the previous round have been read. Similarly, we can say that at least $s-1$ messages by C_{i+2} have also been read, and so on. Accordingly, of the $s(n-1)$ slots that have been previously filled, at least $s + (s-1) + (s-$

2) $+ \cdots + 1$ messages have already been read. These slots can be treated as empty, making the total number of available positions:

$$A \geq mn - s(n-1) + \frac{s(s+1)}{2}$$

The number of messages that C_i can submit per round must be less than or equal to the number of available slots. Accordingly:

$$\begin{aligned} s &\leq mn - s(n-1) + \frac{s(s+1)}{2} \\ ns - \frac{s(s+1)}{2} &\leq mn \\ s(1 - \frac{s+1}{2n}) &\leq m \end{aligned}$$

Given that clients can submit between 1 and $n-1$ messages per exchange, a system based on this protocol must set a lower bound for m to between $1 - \frac{1}{n}$ and $\frac{n-1}{2}$ to provide the minimum conditions for all s messages to be sent without collision.

We calculate the probability of having no collisions during a round. We assume that there have been no prior collisions in the previous $n-1$ exchanges as this gives us a worst case analysis (i.e., fewer available slots). We let x be equal to the number of slots that are filled when C_i begins its exchange. Specifically:

$$\begin{aligned} x &= s(n-1) - \frac{s(s+1)}{2} \\ &= s((n-1) - \frac{s+1}{2})) \end{aligned}$$

The probability that no collisions occur during a round is therefore:

$$\begin{aligned} P_i &\geq \frac{mn-x}{mn} \times \frac{mn-(x+1)}{mn} \times \cdots \times \frac{mn-(x+s-1)}{mn} \\ &\geq (1-\frac{x}{mn}) \times (1-\frac{x+1}{mn}) \times \cdots \times (1-\frac{x+s-1}{mn}) \\ &\geq \prod_{i=1}^{s}(1-\frac{x+i-1}{mn}) \end{aligned}$$

This derivation provides an important insight. Ensuring a low probability of collision requires the values $\frac{x}{ms}$ and $\frac{x+k-1}{mn}$ to be as close to 0 as possible. Such conditions are only possible if the value of s is reasonably small (i.e., closer to 1) when compared to n or if the value of m is very large as compared to s.

The probability that an individual client experiences a collision during the course of a round is:

$$P \leq (1 - \frac{s}{nm})^{n-1}$$

Note that the probability of collision for a single node is dependent on the distance (i.e., the number of exchanges between other clients and OP) between the source and destination of a message. We study the performance tradeoffs associated with varying these parameters in the next section.

4.5 Scaling

The "improved" architecture works well for relatively small groups of nodes. However, given the high "end-to-end" delay associated with an increasing number of users and a strict round-robin scheduling algorithm, provisions must be made to aid graceful scaling.

Our architecture can be scaled to support large numbers of users by partitioning nodes into groups - similar approaches have been applied to address security scaling issues in other networks [46]. Each group is treated as if it is its own MIPNet - a mailbox is "passed" between this subset of participants, each of whom read and write messages through the previously described mechanisms. In addition, nodes are able to read from the mailboxes of all of the other groups associated with the MIPNet. Figure 3 provides an overview of this approach. Note that groups are only able to write to their own mailbox; however, they can communicate across groups by reading from all mailboxes. Additionally, in order for a node in one group to be able to read an entry in another group, all nodes in the same position in each group must have access to the same keystream.

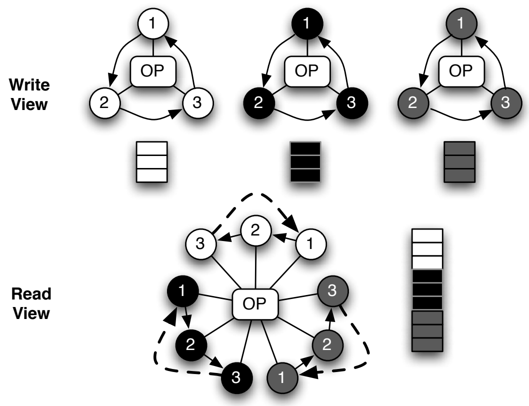


Figure 3: Supporting large numbers of users by separating nodes into groups. Each group of nodes write to their group mailbox (write view). However, nodes in each group are able to read from the other groups' mailboxes (read view). This approach allows operation within the groups to occur in parallel, thereby reducing the end-to-end latency of messages from ng to n, where n is the number of users in each MIPNet and g is the number of MIPNets.

Assuming that we form g MIPNets, each of which contains n nodes for a total of ng users, this approach reduces the time between rounds from ng to n. Messages can accordingly be delivered to their intended destination in much less time.

4.6 Security Guarantees

We now prove that MIPNets provide unlinkability.

LEMMA 1. *MIPNets provide sender anonymity.*

PROOF. The MIPNet architecture is secure against malicious adversaries/attacks based on five assumptions. Like Fairplay, we model SHA-1 as a random oracle and require that a client does not terminate the protocol before sending the output it generates for the OP back to it. Also as in Fairplay, to obtain security against malicious (rather than honest-but curious) adversaries, we rely on a cut-and-choose protocol for distributing garbled circuits to clients. Unlike Fairplay, we implement our OT protocol using a two-lock cryptosystem by Huang et al. [32] based on RSA. Accordingly, each OT operation is secure based on the hardness of the RSA problem. Finally, clients use AES in counter mode to generate a shared keystream, which is pseudorandom assuming that AES is a secure block cipher.

As described in Section 4.4, client C_i inputs the vector R_i, j, i, k, k' into the SFE exchange with OP and receives the slot $i \oplus k$. OP inputs the mailbox M and receives M', which is $M \oplus k \oplus k'$. Because M' and M both appear to be indistinguishable from random bits, it is not possible for OP to guess the slot j written by C_i with probability greater than $\frac{1}{mn} + \epsilon$. As $\frac{1}{mn} \leq \frac{1}{n}$, MIPNets satisfy Definition 1. □

LEMMA 2. *MIPNets provide receiver anonymity.*

PROOF. Receiver anonymity is the dual of sender anonymity as shown in Lemma 1. Accordingly, we rely upon the same cryptographic assumptions. As described in Section 4.4, client C_i receives slot i, but OP is unable to determine the identity of i due to the use of OT with probability greater than $\frac{1}{mn} + \epsilon$. As $\frac{1}{mn} \leq \frac{1}{n}$, MIPNets satisfy Definition 2. □

THEOREM 1. *MIPNets provide unlinkability.*

PROOF. By Lemmas 1 and 2 and Definition 3, MIPNets provide unlinkability. □

5. PERFORMANCE ANALYSIS

In this section, we explore a range of optimizations and tradeoffs in the hopes of providing the most efficient MIPNet architecture.

5.1 Experimental Setup

We analyzed the performance of both the new OT primitive we have implemented and the "improved" MIPNet architecture. Our experiments were run on two servers, each with 8 2-GHz Quad-Core AMD Opteron processors, 16 GB of memory, 1 Gbit Ethernet card and running 2.6.24 Linux Kernel (Ubuntu 8.04.2). The client and OP code were implemented in SFDL and compiled into Java objects using Fairplay.

5.2 Performance Evaluation

5.2.1 OT Performance

We first compare the performance of the two OT primitives. As a simple comparison, we evaluate the performance of each mechanism in the Millionaire's Problem over a range of input sizes. The results are shown in Figure 4 for 100 iterations of the protocol, with 95% confidence intervals included.

As an interesting aside, our first set of results for these experiments exhibited a saw-toothed behavior for a small number of input bits. After much debugging, we realized that the observed irregularities in our timing data were a result of Nagle's Algorithm acting upon TCP. In particular, transfers were being delayed until either buffers were entirely filled or a timeout occurred. We were able to eliminate such behavior by setting the `TCP_NODELAY` socket option, and use this setting throughout the remainder of our experiments.

The results of this experiment clearly show that for the same circuit, the RSA-OT primitive is significantly more efficient than the NP-OT primitive provided with Fairplay. For all of the tested input values, the RSA-OT scheme completed its execution in less than 50% of the time as was required by the NP-OT scheme. Moreover, the use of additional JIT optimization further improved the performance of the RSA-OT scheme by approximately 20%. While the performance difference between the current and new mechanisms on a per-OT scale is small, the use of a relatively large number of OT exchanges makes the aggregate improvement significant. Accordingly, we use the RSA-OT primitive with JIT optimizations as the basis for the remainder of our experiments.

5.2.2 Circuit Size

The size of the garbled circuits created by Fairplay is directly proportional to their performance. However, equivalent but more efficient encodings of these functions are possible. *Ordered Binary Decision Diagrams* (OBDDs) provide a graph-based representation of SFE circuits and have been demonstrated to significantly reduce the bandwidth used by such exchanges. However, in some rare cases, OBDDs can experience an exponential blowup in size and become far less efficient than the circuits produced by Fairplay.

In order to improve the performance of our protocol, we compare the sizes of the functions generated by both Fairplay and the OBDD

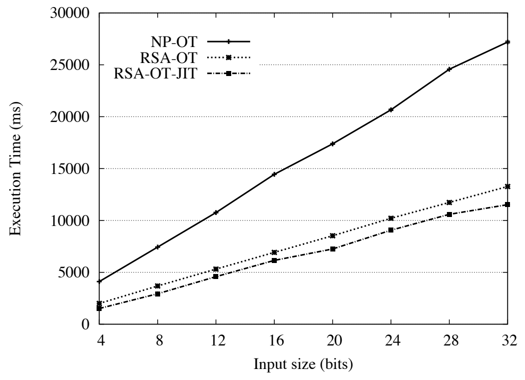


Figure 4: Performance evaluation for the "Millionaire's Problem" comparing the oblivious transfer scheme included with Fairplay (NP-OT) and the RSA-based Two-Lock Cryptosystem (RSA-OT). RSA-OT is more than 50% more efficient than the NP-OT mechanism.

Data	Fairplay			OBDD		
Bits	n=3	n=5	n=7	n=3	n=5	n=7
1	396	3264	6272	292	2328	4590
2	483	3793	7253	380	2856	5562
3	570	4322	8234	468	3384	6534
4	657	4851	9215	556	3912	7506
5	744	5380	10196	644	4440	8478
6	831	5909	11177	732	4968	9450
7	918	6438	12158	820	5496	10422

Table 1: Circuit size for Fairplay and OBDD compiler circuits.

compiler created by Kruger et al. [37]. The results of these tests are provided in Table 1. For a fixed number of users in the MIPNet (n = 3, n = 5 and n = 7) and a fixed $m = n - 1$, we varied the number of data bits carried in each input vector (R) entered by the client per exchange. The functions produced by the OBDD compiler were smaller in all cases, with improvements ranging from 11% to 27% for n = 3, 15% to 29% for n = 5 and 14% to 27% for n = 7.

Accordingly, we use the OBDD compiler to generate the functions used in the remainder of the paper.

5.3 Instant Messaging Application

With an understanding of the performance profile of our system, we now explore the potential for applications to run on top of a MIPNet. We implement a simple Instant Messaging (IM) client, which we call MIPChat. As an example, we envision MIPChat being used in a low cross-traffic scenario to provide a deniable communications medium between a reporter and his or her source as they sit on opposite sides of a public space such as a coffee shop or train station. MIPChat resembles an IRC chat client, with received communications multiplexed into a single window. Outgoing messages are run in a second screen, and prepended with the identifier of the intended destination (e.g., "`@alice`" is mapped to Client 2).

Figure 5 provides an overview of a MIPChat client. Clients without messages to send generate random messages via OpenSSL's `RAND_bytes()` call, which generates cryptographically strong pseudo-random byte streams, and pass these messages to a buffer serviced by the underlying MIPNet client. These messages are sent to an arbitrary client with the read bit in the slot set to zero so as

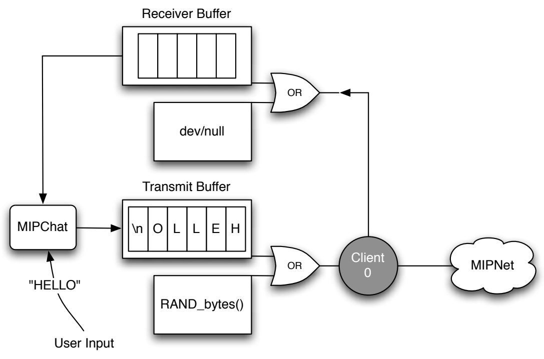
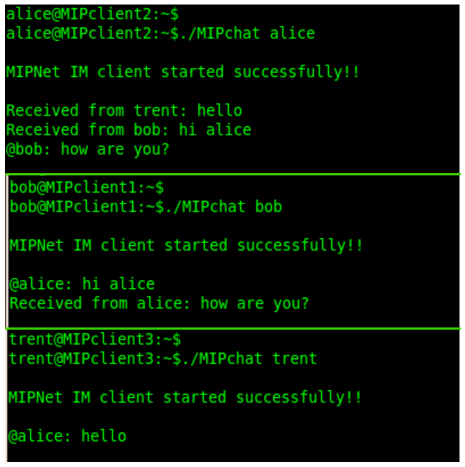


Figure 5: The MIPChat application and its relationship to the MIPNet architecture.



```
alice@MIPclient2:~$
alice@MIPclient2:~$./MIPchat alice

MIPNet IM client started successfully!!

Received from trent: hello
Received from bob: hi alice
@bob: how are you?
```

```
bob@MIPclient1:~$
bob@MIPclient1:~$./MIPchat bob

MIPNet IM client started successfully!!

@alice: hi alice
Received from alice: how are you?
```

```
trent@MIPclient3:~$
trent@MIPclient3:~$./MIPchat trent

MIPNet IM client started successfully!!

@alice: hello
```

Figure 6: A screenshot of the MIPChat client. Clients inject pseudorandom garbage until they have real messages to send. Messages are buffered and displayed by the client when they are received in totality.

to indicate the lack of content in the message. The receiving client simply discards the messages and awaits the next exchange. When the user presses return, the real message is parsed for destination and sent to the same buffer, where it is sent to the requested host. On the receiving side, the client collects incoming legitimate messages in a buffer and, upon receiving an EOL character, pushes the contents of the buffer to the client where they are displayed for the user. Figure 6 provides a screen shot of this application, in which "Bob" and "Trent" have both sent messages to "Alice", and Alice has responded to "Bob". Recall that the source of the messages is clear to the receiver in the absence of collisions as only a specific sequence of slots in the mailbox could be correctly filled by the sender based on the shared key hashing mechanism discussed in Section 4.4.

Two small changes to the underlying MIPNet can allow for the efficient support of a much larger user population. First, we break users into groups as was proposed in Section 4.5 to reduce the "end-to-end" latency caused by adding users. Second, we take advantage of how users are likely to actually use the MIPChat client. Specifically, users are unlikely to attempt to talk to every participant in the network concurrently. While a user may exchange messages with a small number of participants at the same time, few users will ever talk to every member on their contact list in parallel. From an informal analysis of chat logs in our own lab, we saw that no user

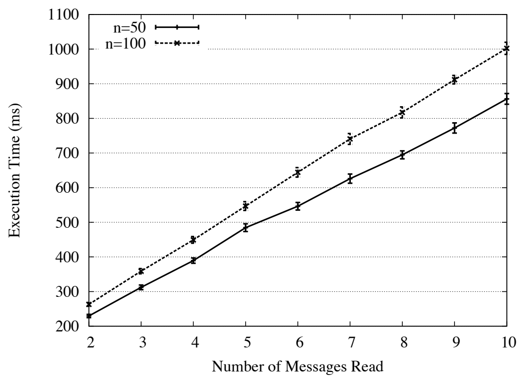


Figure 7: The performance profile of MIPChat running on top of our architecture. Note that we can support large user populations (e.g., $ng = 100$) with this approach and still process keystrokes as or nearly as fast as users enter them.

ever spoke to more than three individuals on their contact list at the same time. The execution time of an SFE exchange with an OP can therefore be dramatically reduced by limiting the number of slots actually read by a client in a single interaction. Users can check the slots in which they expect a message during each round and also intermittently monitor other slots in the mailbox for attempts to initiate communications. We leave the specific synchronization scheme between clients (e.g., explicit "start" messages, synchronized sleep periods, etc) to each implementation of this system as these mechanisms have been thoroughly studied.

Figure 7 shows the performance of the MIPChat client supporting $ng = 50, 100$ clients, $m = 2$, $s = 1$, 5 nodes assigned to each group and a varying number of the total slots read. As before, this experiment shows the mean of 100 iterations of the experiment for each point and provides 95% confidence intervals two orders of magnitude smaller than the mean. This approach provides significant savings over the standard MIPNet architecture. Specifically, if a client only reads five messages during each round (e.g., three conversations and polls for two additional conversation requests), each round takes an average of 484.43 and 546.32 ms for $ng = 50, 100$, respectively. Because users are not constantly typing, the slightly slower processing to input speed here is unlikely to significantly impact perception of throughput for this application. Note the longest delay to deliver a message (i.e., from client i to client $i-1$) has also been greatly reduced in this configuration to approximately three seconds. Given that the average user types between 19 and 33 words per minute [36], or one character every 333.3 to 526.3ms, *our infrastructure is capable of processing characters faster than the majority of users enter them for all but one of the above cases.*

These values indicate that the MIPNet architecture is capable of supporting applications with some tolerance to delay (e.g., chat). We will show the practicality of additional applications including email in our future work.

5.4 Performance Comparison

We conduct one final set of experiments to measure the total performance improvement over the naïve use of the underlying cryptographic primitives. Specifically, we recreate the MIPChat client without any of the optimizations discussed in this work (e.g., RSA-OT, groups, etc) and simply compile our circuits using an unmodified version of the Fairplay compiler and execute them in the "basic" architecture.

Unfortunately, this most basic comparison was not possible because the Fairplay compiler exceeded the 1GB memory maximum set by the Java virtual machine. Accordingly, we compiled our circuits using only the OBDD compiler but without any additional optimizations. For $n = 100$, we recorded 50 iterations of the protocol and observed an average time of 35.626 seconds, with a 95% confidence interval of ± 0.161 seconds. The performance profile of our optimized MIPChat client provides a reduction in execution time of 98.5%. This dramatic improvement over the naïve application of SFE primitives demonstrates that our optimizations and careful parameterization can allow these constructions to form the basis of a system with near real-time performance requirements.

6. CONCLUSION

We present Mix-In-Place Networks (MIPNets), an architecture built on Secure Function Evaluation that replaces multiple intermediary nodes with a cascade of functions in a single proxy. Through the use of SFE, this proxy remains oblivious to the source, destination and content of all messages exchanged within the system. We demonstrate that our proposed architecture provides provable deniability and then, through extensive performance analysis using a number of optimizations that dramatically reduce the execution time (greater than 98%), demonstrate the practicality of this approach for applications with near real-time requirements, including instant messaging for as many as 100 clients. Finally, we discuss a number of enhancements to potentially further improve the performance, scalability and robustness of our architecture. In so doing, we have not only demonstrated the practicality of using a single node to mix traffic with provable properties, but also shown that SFE can address a void in the anonymous communications space for applications with moderate performance constraints and no guarantees of adequate cross traffic.

Acknowledgments

We would like to thank Louis Kruger for providing us with the source code for the OBDD compiler, and Chris Peikert, William Enck, Chaitrali Amrutkar, Brendan Dolan-Gavit and our anonymous reviewers for their helpful comments. This work was supported in part by the US National Science Foundation (CCF-0916031). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

[1] G. Aggarwal, N. Mishra, and B. Pinkas. Secure Computation of the k^{th}-Ranked Element. In *Proceedings of Eurocrypt*, 2004.

[2] P. Baran. On Distributed Communications: IX. Security, Secrecy, and Tamper-Free Considerations. Technical Report RM-3765-PR, The RAND Corporation, 1964.

[3] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-Resource Routing Attacks Against Tor. In *Proceedings of the ACM Workshop on Privacy in Electronic Society (WPES)*, 2007.

[4] A. Beimel and S. Dolev. Buses for Anonymous Message Delivery. *Journal of Cryptology*, 16(1), 2001.

[5] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP – A System for Secure Multi-Party Computation. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2008.

[6] O. Berthold, H. Federrath, and S. Kopsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. In *Proceedings of Designing Privacy Enhancing Technologies*, 2000.

[7] J. Bos and B. den Boer. Detection of Disrupters in the DC Protocol. In *Proceedings of Eurocrypt*, 1989.

[8] J. Boyan. The Anonymizer: Protecting User Privacy on the Web. *Computer-Mediated Communication Magazine*, 4(9), 1997.

[9] M. Burnside and A. Keromytis. Low Latency Anonymity with Mix Rings. In *Proceedings of the 9th International Information Security Conference (ISC)*, 2006.

[10] D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2), 1981.

[11] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1), 1988.

[12] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, January/February 2002.

[13] C. Cortes, D. Pregibon, and C. Volinsky. Communities of Interest. In *Proceedings of the International Symposium of Intelligent Data Analysis (IDA)*, 2001.

[14] W. Dai. PipeNet 1.1. `http://www.weidai.com`, 1998.

[15] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the IEEE Symposium on Security and Privacy (OAKLAND)*, 2003.

[16] R. Dingledine, M. J. Freedman, and D. Molnar. The Free Haven Project: Distributed Anonymous Storage Service. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[17] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the USENIX Security Symposium (USENIX)*, 2004.

[18] S. Dolev and R. Ostrovsky. XOR-Trees for Efficient Anonymous Multicast Reception. In *Proceedings of CRYPTO*, 1997.

[19] N. S. Evans, R. Dingledine, and C. Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *Proceedings of the USENIX Security Symposium (USENIX)*, 2009.

[20] H. Federrath. JAP – Anonymity and Privacy. `http://anon.inf.tu-dresden.de/index_en.html`, 2008.

[21] J. Feigenbaum, A. Johnson, and P. Syverson. A Model of Onion Routing with Provable Anonymity. In *Proceedings of Financial Cryptography*, 2007.

[22] J. Feigenbaum, B. Pinkas, R. Ryger, and F. Saint-Jean. Secure Computation of Surveys. In *Proceedings of the EU Workshop on Secure Multiparty Protocols (SMP)*, 2004.

[23] M. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *Proceedings of Eurocrypt*, 2004.

[24] M. J. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2002.

[25] I. Goldberg and A. Shostack. Freedom Systems 2.0 Architecture. `http://osiris.978.org/%7Ebrianr/crypto-research/anon/www.freedom.net/products/whitepapers/Freedom_System_2_Architecture.pdf`, 1999.

[26] O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 1987.

[27] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In *Proceedings of the RSA Conference, CryptographerâĂŹs Track*, 2004.

[28] P. Golle and A. Juels. Dining Cryptographers Revisited. In *Proceedings of Eurocrypt*, 2004.

[29] M. Green and G. Ateniese. Identity-Based Proxy Re-encryption. In *Applied Cryptography and Network Security (ACNS)*, 2007.

[30] C. Gulcu and G. Tsudik. Mixing Email with Babel. In *Proceedings of the ISOC Symposium on Network and Distributed Systems Security (NDSS)*, 1996.

[31] T. Heydt-Benjamin, A. Serjantov, and B. Defend. Nonesuch: A Mix Network with Sender Unobservability. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, 2006.

[32] H.-F. Huang and C.-C. Chang. A New Design for Efficient t-out-n Oblivious Transfer Scheme. In *Proceedings of the International Conference on Advanced Information Networking and Applications (AINA)*, 2005.

[33] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster Secure Two-Party Computation Using Garbled Circuits. In *Proceedings of the USENIX Security Symposium (SECURITY)*, 2011.

[34] M. Jakobsson. Flash Mixing. In *Proceedings of the ACM Principles of Distributed Computing (PODC)*, 1999.

[35] L. Johansen, M. Rowell, K. Butler, and P. McDaniel. Email Communities of Interest. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2007.

[36] C. M. Karat, C. Halverson, D. Horn, and J. Karat. Patterns of Entry and Correction in Large Vocabulary Continuous Speech Recognition Systems. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 1999.

[37] L. Kruger, S. Jha, E.-J. Goh, and D. Boneh. Secure Function Evaluation with Ordered Binary Decision Diagrams. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2006.

[38] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright. Timing Attacks in Low-Latency Mix-based Systems. In *Proceedings of Financial Cryptography*,

2004.

[39] B. N. Levine and C. Shields. Hordes: A Multicast Based Protocol for Anonymity. *Journal of Computer Security*, 10, 2002.

[40] Y. Lindell and B. Pinkas. Privacy preserving data mining. 15(3), 2002.

[41] P. D. MacKenzie, A. Oprea, and M. Reiter. Automatic Generation of Two-Party Computations. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2003.

[42] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay - A Secure Two-Party Computation System. In *Proceedings of the USENIX Security Symposium (SECURITY)*, 2004.

[43] N. Mathewson and R. Dingledine. Practical Traffic Analysis: Extending and Resisting Statistical Disclosure. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2005.

[44] P. McDaniel, S. Sen, O. Spatscheck, J. Van der Merwe, B. Aiello, and C. Kalmanek. Enterprise Security: A Community of Interest Based Approach. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2006.

[45] C. A. Melchor and Y. Deswarte. From DC-nets to pMIXes: Multiple Variants for Anonymous Communications. In *Proceedings of the IEEE Symposium on Network Computing and Applications (NCA)*, 2006.

[46] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *Proceedings of ACM SIGCOMM*, 1997.

[47] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mix-master Protocol Version 2. Technical Report Internet-Draft, IETF Network Working Group, 2004.

[48] S. J. Murdoch. Hot or Not: Revealing Hidden Services by their Clock Skew. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2006.

[49] S. J. Murdoch and G. Danezis. Low-cost Traffic Analysis of Tor. In *Proceedings of the IEEE Symposium on Security and Privacy (OAKLAND)*, 2005.

[50] M. Naor and B. Pinkas. Efficient Oblivious Transfer Protocols. In *Proceedings of the ACM Symposium on Discrete Algorithms (SODA)*, 2001.

[51] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable Communication with Very Small Bandwidth Overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, 1991.

[52] A. Pfitzmann and M. Waidner. Networks Without User Observability. *Computers & Security*, 6, 1987.

[53] B. Pinkas, M. Naor, and R. Sumner. Privacy Preserving Auctions and Mechanism Design. In *Proceedings of the ACM Conference on Electronic Commerce*, 1999.

[54] K. Poulsen. FBI retires its Carnivore. `http://www.securityfocus.com/news/10307`, 2005.

[55] J. Powers. The attack of the eavesdropping neighbors. `http://www.michigandaily.com/content/attack-eavesdropping-neighbors`, 2002.

[56] Proxify.com. Proxify anonymous proxy - surf the Web privately and securely. `http://proxify.com/`, 2008.

[57] Y. J. Pyun, Y. H. Park, X. Wang, D. Reeves, and P. Ning. Tracing Traffic Through Intermediate Hosts that Repacketize Flows. In *Proceedings of IEEE INFOCOM*, 2007.

[58] M. O. Rabin. How to Exchange Secrets with Oblivious Transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

[59] M. Reed, P. Syverson, and D. Goldschlag. Proxies for Anonymous Routing. In *Proceedings of Annual Computer Security Applications Conference (ACSAC)*, 2006.

[60] C. Reis, S. D. Gribble, T. Kohno, and N. Weaver. UW CSE and ICSI Web Integrity Checker. `http://vancouver.cs.washington.edu/`, 2007.

[61] M. Reiter and A. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information System Security (TISSEC)*, 1(1), 1998.

[62] L. Sassaman, B. Cohen, and N. Mathewson. The Pynchon Gate: A Secure Method of Pseudonymous Mail Retrieval. In *Workshop on Privacy in the Electronic Society (WPES)*, 2005.

[63] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P^5: A Protocol for Scalable Anonymous Communication. In *Proceedings of the IEEE Symposium on Security and Privacy (OAKLAND)*, 2002.

[64] V. Shmatikov and M.-H. Wang. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2006.

[65] R. Singel. Google-DoubleClick Privacy Fight Hangs Over Fed's E-Advertising Forum. `http://blog.wired.com/27bstroke6/2007/10/google-doublecl.html`, 2007.

[66] E. G. Sirer, S. Goel, M. Robson, and D. Engin. Eluding Carnivores: File Sharing with Strong Anonymity. In *Proceedings of the European SIGOPS Workshop*, 2004.

[67] M. Srivatsa, L. Liu, and A. Iyengar. Preserving Caller Anonymity in Voice-over-IP Networks. In *Proceedings of the IEEE Symposium on Security and Privacy (OAKLAND)*, 2008.

[68] P. Syverson, D. Goldschlag, and M. Reed. Anonymous Connections and Onion Routing. In *Proceedings of the IEEE Symposium on Security and Privacy (OAKLAND)*, 1997.

[69] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In *Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[70] L. von Ahn, A. Bortz, and N. J. Hopper. k-Anonymous Message Transmission. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2003.

[71] M. Waidner. Unconditional Sender and Recipient Untraceability in Spite of Active Attacks. In *Proceedings of Eurocrypt*, 1990.

[72] M. Waidner and B. Pfitzmann. The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Service ability. In *Proceedings of Eurocrypt*, 1989.

[73] M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the USENIX Security Symposium (SECURITY)*, 2000.

[74] X. Wang, S. Chen, and S. Jajodia. Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2005.

[75] X. Wang, S. Chen, and S. Jajodia. Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems. In *Proceedings of the IEEE Symposium on Security and Privacy (OAKLAND)*, 2007.

[76] X. Wang, D. Reeves, and S. Wu. Inter-packet Delay Based Correlation for Tracing Encrypted Connections Through Stepping Stones. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2002.

[77] A. C. Yao. How to Generate and Exchange Secrets. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 1986.

[78] YouHide.com. Anonymous Proxy Server. `http://www.youhide.com/`, 2008.

[79] Y. Zhang and V. Paxson. Detecting Stepping Stones. In *Proceedings of the USENIX Security Symposium (SECURITY)*, 2000.