Dust: A Blocking-Resistant Internet Transport Protocol

Brandon Wiley

School of Information, University of Texas at Austin
1616 Guadalupe #5.202
Austin, TX 78701-1213
brandon@ischool.utexas.edu

Abstract. Censorship of information on the Internet has been an increasing problem as the methods have become more sophisticated and increasing resources have been allocated to censor more content. A number of approaches to counteract Internet censorship have been implemented, from censorship-resistant publishing systems to anonymizing proxies. A prerequisite for these systems to function against real attackers is that they also offer blocking resistance. Dust is proposed as a blocking-resistant Internet protocol designed to be used alone or in conjunction with existing systems to resist a number of attacks currently in active use to censor Internet communication. Unlike previous work in censorship resistance, it does not seek to provide anonymity in terms of unlinkability of sender and receiver. Instead it provides blocking resistance against the most common packet filtering techniques currently in use to impose Internet censorship.

Keywords: censorship resistance, blocking resistance

1 Introduction

Censorship of information on the Internet has been implemented using increasingly sophisticated techniques. Shallow packet filtering, which can be circumvented by anonymizing proxies, has been replaced by deep packet inspection technology which can filter out specific Internet protocols. This has resulted in censorship-resistant services being entirely blocked or partially blocked through bandwidth throttling. Traditional approaches to censorship resistance are not effective unless they also incorporate blocking resistance so that users can communicate with the censorship circumvention services.

Dust is an Internet protocol designed to resist a number of attacks currently in active use to censor Internet communication. Dust uses a novel technique for establishing a secure, blocking-resistant channel for communication over a filtered channel. Once a channel has been established, Dust packets are indistinguishable from random packets and so cannot be filtered by normal techniques. Unlike other encrypted protocols such as SSL/TLS, there is no plaintext handshake which would allow the protocol to be fingerprinted and therefore blocked or throttled. This solves a

principle weakness of current censorship-resistant systems, which are vulnerable to deep packet inspection filtering attacks.

1.1 Problem

Traditionally, Internet traffic has been filtered using "shallow packet inspection" (SPI). With SPI, only packet headers are examined. Since packet headers must be examined anyway in order to route the packets, this form of filtering has minimal impact on the scalability of the filtering process, allowing for its widespread use. The primary means of determining "bad" packets with SPI is to compare the source and destination IP addresses and ports to IP and port blacklists. The blacklists must be updated as new target IPs and port are discovered. Circumvention technology, such as anonymous proxies, bypass this filtering by providing new IPs and ports not in the blacklist which proxy connections to blacklisted IPs. As the IPs of proxies are discovered, they are added to the blacklist, so a fresh set of proxy IPs must be made available and communicated to users periodically. As port blacklists are used to block certain protocols, such as BitTorrent, regardless of IP, clients use port randomization to find ports which are not on the blacklist.

Recently, "deep packet inspection" (DPI) techniques have been deployed which can successfully block or throttle most censorship circumvention solutions [14]. DPI filters packets by examining the packet payload. DPI can achieve suitable scalability through random sampling of packets. Another technique in use is to initially send packets through, but also send them to a background process for analysis. When a bad packet is found, further packets in that packet stream can be blocked, or the IPs of participants added to the blacklist. The primary tests that DPI filters apply to packets are packet length comparison and static string matching, although timing-based fingerprints are also possible. DPI can not only filter content, but also fingerprint and filter specific protocols, even encrypted protocols such as SSL/TLS. Encrypted protocols are vulnerable to fingerprinting based on packet length, timing, and static string matching of the unencrypted handshake that precedes encrypted communication. For instance, SSL/TLS uses an unencrypted handshake for cipher negotiation and key exchange.

The goal of Dust is to provide a transport protocol which cannot be filtered with DPI. To accomplish this goal it must not be vulnerable to fingerprinting using static string matching, packet length comparison, or timing profiling. Other attacks such as IP address matching and coercion of operators are outside of the scope and are best addressed by use of existing systems such as anonymizing proxies and censorship-resistant publishing systems running on top of a Dust transport layer.

2 Related Work

Censorship resistance is often discussed in connection with other related concepts such as anonymity, unlinkability, and unobservability. These terms are sometimes used interchangeably and sometimes assumed to have specific technical definitions. Pfitzmann proposed a standardized terminology that defines and relates these terms

[13]. Unlinkability is defined as the indistinguishability of two objects within an anonymity set. Anonymity is defined as unlinkability between a given object and a known object of interest. Unobservability is defined as unlinkability of a given object and a randomly chosen object.

Defining properties such as anonymity and unobservability in terms of unlinkability opens the way for an information theoretical approach. Hevia offers such an approach by defining levels of anonymity in terms of what information is leaked from the system to the attacker [8]. Unlinkability requires the least protection, hiding only the message contents. Unobservability requires that no information is leaked whatsoever. Of particular interest is that an anonymous system of any type can be taken up to the next level of anonymity by adding one of two system design primitives: encryption and cover traffic.

2.1 Censorship-Resistant Publishing and Anonymizing Proxies

One approach to achieving censorship resistance on the Internet is through censorship-resistant publishing services such as Publius [18], Tangler [19], and Mnemosyne [5]. An issue with anonymous publishing systems for practical use is that even a system that provides maximum protection for stored files must still be accessible in order for those files to be retrieved. If communication to the document servers is blocked, then the system is not usable. This requires protection for communications as well as documents. Serjantov [15] proposed a the solution of combining anonymous publishing with anonymous proxies by running the publishing service as a hidden service behind an onion routing network such as Tor [3].

This solution passes on the problem of blocking from the publishing system to the anonymizing proxy. However, anonymizing proxies are also vulnerable to blocking attacks. While a network of proxy nodes can provide protection against destination IP blacklists, they are still vulnerable to various forms of DPI protocol fingerprinting. This problem is dealt with by Kopsell, who proposes a method to extend existing anonymous publishing systems to bypass blocking, a property referred to as "blocking resistance" [9]. In light of the work of Serjantov and Kopsell it is evident that if anonymous proxies are a necessary component of censorship-resistant publishing and blocking resistance is a necessary property of anonymous proxies then blocking resistance is necessary for censorship-resistant publishing.

Kopsel's threat model contains the assumptions that the attacker has control of only part of the Internet (the censored zone), that some small amount of unblockable inbound information can enter the censored zone (perhaps out of band), and that volunteers outside of the censored zone are willing to help although they may have differing amounts of bandwidth to contribute. The attacker is assumed to have vast resources, to control all links outbound from the censored zone to the Internet, and to be an expert in blocking-resistant system design.

Kopsell's solution is divided into two parts: access to the blocking-resistant system, and distributing information about the blocking-resistant system, such as the IPs of proxy nodes. The nodes in Kopsell's system are volunteer-run anonymizing proxies that clients communicate with over a steganographic protocol in order to obtain access to a censorship-resistant publishing system. Clients obtain an invitation

to the network, including the IP addresses of some proxy nodes, through a low-bandwidth, unblockable channel into the censored zone. A number of ideas are proposed for the steganographic data channel such as SSL and SMTP protocols. For the unblockable channel email is used.

Though Kopsell's model for blocking resistance solves the real world issues facing anonymous publication systems and proxies, it relies on the steganographic data and unblockable invitation channels to have certain properties which may not be met in actual implementation. The essential purpose of the steganographic channel is to provide resistance to protocol fingerprinting. Even if the information cannot be recovered from the steganographic encoding, if it is discovered that the channel contains steganographically encoded information then it can be summarily blocked. In other words, the encoding must be undetectable in order to be useful. The constraint on the invitation channel is that it is completely unblockable, as no particular protection is given to information distributed on this channel.

Real world of analysis of attacks has shown that SSL is not a suitable encoding against real attackers as the protocol is easily fingerprinted and summarily blocked or rate limited [14]. Also, Email is an unsuitable channel for invitations because it is not unblockable. Recent attacks have blocked the communication of IP addresses of proxies through email and instant messaging. Given these attacks, what sorts of channels are suitable for invitations and data to be communicated without being vulnerable to blocking?

Information theory provides a conceptual framework that offers an answer not just to the question of blocking resistance but of its relationship to censorship resistance in general. Censorship-resistance publishing systems provide document unlinkability. Hevia links the definition of unlinkability to information theory through indistinguishability of information transmitted on the channels between the system and the attacker [7] and Boesgaard links document unlinkability to information theoretic perfect secrecy [2]. So censorship resistance is therefore a form of perfect secrecy by means of indistinguishability. Pfitzmann defines unobservability as a form of unlinkability [13] and Perng defines censorship resistance as unobservability [12]. In other words, censorship resistance is unobservability through unlinkability of the object of interest and a random object, which is equivalent in information theory to perfect secrecy. Viewed in this context, a censorship-resistant publishing system would be one in which through observation of the system the attacker cannot obtain sufficient information to distinguish which documents are accessed by users, in other words document unobservability. Anonymous proxies add a similar property, unobservability of the publishing system. The final step, which Kopsell calls blocking resistance, is unobservability of the anonymous proxy, which requires unobservability of the protocol by which clients communicate with the proxies. When these properties are combined, end-to-end unobservability is created from the client to the document.

The ideal communication protocol is therefore one which is unobservable, meaning that a packet or sequence of packets is indistinguishable from a random packet or random sequence of packets. This is not necessarily a steganographic encoding. A steganography encoding is unobservable only so long as the message encoding is not detectable, regardless of if the message can actually be decoded. Additionally, steganographic channels can be blocked if the cover channel is blocked. In the cause of the rate limiting of Tor, SSL was being used as both encryption and steganography.

Rate limiting of the cover occurred because all SSL traffic was summarily rate limited, causing a rate limiting of the embedded message as well and essentially failing to provide blocking resistance.

Steganography is not the only option for unobservable protocols. Encryption is an equally valid means of making messages indistinguishable. Although protocols such as SSL are encrypted, these protocols often have an unencrypted handshake. This unencrypted portion of the communication is what is used to fingerprint and block the protocol. Additionally these protocols may leak other information to the attacker through packet lengths and timing. However, an encrypted protocol without a handshake would be resistant to handshake fingerprinting. With sufficiently secure encryption and a lack of unencrypted handshakes, one encrypted protocol should be indistinguishable from another encrypted protocol.

In the normal use case for SSL, an entirely encrypted connection would not be possible as the communicating peers need to perform a public key exchange in order to determine the session key used to encrypt the conversation. However, unlike a normal SSL connection, Kopsell's model allows for a single out-of-band invitation to be sent prior to the establishment of the data connection.

2.2 Obfuscated Protocols

Several obfuscated protocols have been developed with various goals, including blocking resistance. For instance, BitTorrent clients have implemented three encryption protocols in order to prevent filtering and throttling of the BitTorrent protocol, the strongest of which is Message Stream Encryption (MSE). [11] Analysis of packet sizes and the direction of packet flow have been shown to identify connections obfuscated with MSE with 96% accuracy. [7] MSE also uses a cleartext DH key exchange. However, it does not include static strings in the protocol handshake as the handshake consists solely of the DH parameters, which are unique to each connection.

Other obfuscated protocols which are not designed explicitly for blocking resistance also suffer from cleartext handshakes and often include static strings in the handshake. Obsfuscated TCP (ObsTCP) has gone through several versions, each using a different means to communicate the keys, including TCP options, HTTP headers, and DNS records. [12] The strongest of these is DNS records as TCP options and HTTP headers are easily blocked using static string matching, while DNS records are transmitted on a separate connection from the one carrying the data, requiring correlation between separate connections. However, Sandvine has already demonstrated this ability in the blocking of BitTorrent traffic by monitoring tracker protocol traffic to obtain the ports of BitTorrent protocol connections and then subsequently interfering with the (possibly encrypted) BitTorrent protocol connections. [17][6] A second connection from the same IP can therefore not be used as an out-of-band channel for the purpose of blocking resistance. A newer proposal similar to ObsTCP called tcpcrypt does not blocking resistance as a design goal and subsequently does worse than ObsTCP/DNS as it uses static strings in the handshake protocol. [1]

An attempt has been made to address the cleartext handshake problem in the form of the obfuscated-openssh patch to OpenSSH which encrypts the SSH handshake. [10] An encrypted handshake for an existing encrypted protocol is a good idea as it is the minimal amount change necessary to achieve blocking resistance as long as the protocol already has resistance to packet size and timing attacks. The obfuscated-openssh patch essentially implements its own minimal blocking-resistant protocol, performed before SSH starts and on the same TCP connection. This minimal protocol is similar to Dust in that it is designed to be resistant to static string and packet size matching. Unfortunately, it is not truly blocking resistant because it relies on a false (or perhaps outdated) assumption about the capabilities of filters. The handshake is encrypted with a key that is generated from a seed that is prepended to the beginning of the encrypted part of the handshake. The key is generated by iterated hashes of the seed with the iteration number chosen to be high enough that key generation is slow. The blocking resistance of this technique relies on key generation not being sufficiently scalable to do across all connections simultaneously. However, modern filters are capable of statistically sampling packets and processing them offline to flag packets and then using those results to block IPs which have sent flagged packets. [17] This approach is probabilistic in its ability to block connections, but is highly scalable. Additionally, the introduction of slow key generation may allow for even less expensive timing attacks in which the only information needed to block a connection is the timing between the first and second packets.

3 Design

Dust is a protocol designed to provide protocol unobservability in order to implement Kopsell's concept of blocking-resistance as a necessary prerequisite to achieve censorship resistance. The Dust protocol is designed to protect against an attacker that utilizes Deep Packet Inspection (DPI) to fingerprint protocols for the purpose of blocking or rate limiting connections. In order to establish protocol unobservability, all packets consist entirely of encrypted or random single-use bytes so as to be indistinguishable from each other and random packets.

In order to perform a key exchange without an unencrypted handshake, a novel out-of-band half-handshake technique is used. As in Kopsell's model, a peer must first receive an out-of-band invitation to join the network. This invitation contains the IP address and public key of the receiver. The sender can then complete the handshake by sending a single in-band intro packet followed by any number of data packets encrypted with the session key that was computed in the handshake. The minimal Dust conversation therefore consists of two in-band packets: one intro packet, and one data. The protocol allows for these packets to be be chained together to fit inside a single UDP or TCP packet. The use of a single UDP or TCP packet for communication prevents timing attacks then the payload is sufficiently small.

3.1 Protocol

In order to accept a connection from an unknown host, a Dust server must first complete a key exchange with the client. The Dust server first creates an id and secret pair. The server then sends an out-of-band invite packet to the client, which contains the server's IP, port, public key, the id, and the secret. The invite is encrypted with a password and so is indistinguishable from random bytes. It can then be safely transmitted, along with the password, over an out-of-band channel such as email of instant messaging. It will not be susceptible to the attacks which block email communication containing IP addresses because only the password is transmitted unencrypted. If the invitation channel is under observation by the attacker, and only in the case that the attacker is specifically attempting to filter Dust packets, then the password should be sent by another channel that, while it can still be observed by the attacker, should be uncorrelated with the invitation channel.

In order to complete the handshake, the client uses the IP and port information from the invite packet to send an intro packet to the server. The intro packet is prepended with the random, single-use id from the invite packet. The packet is encrypted with the secret from the invite and contains the public key of the client.

When the server receives a packet from an unknown IP address, it assumes it to be an intro packet and retrieves the id from the beginning of the packet. This is used to look up the associated stored secret. The server uses the secret to decrypt the packet, retrieves the public key of the client, and generates a shared session key. It adds the session key to its list of known hosts, associated with the IP and port from which the intro packet was sent. This completes the second phase of the public key exchange. The client and server can now send and receive encrypted data packets freely. Since Dust packets an be chained inside of TCP or UDP packets, the intro packet may be followed immediately by a data packet, which may constitute the entirety of the conversation.

2 Packet Format

There are three types of Dust packets: invite, intro, and data packets. All three types of packets build upon the basic Dust packet format as shown in Fig. 1.

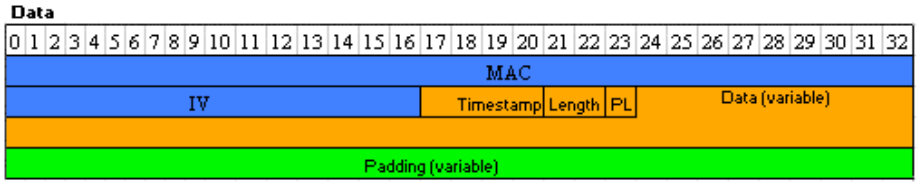

Fig. 1. The general Dust packet format. This is also the format for data packets.

In a Dust packet, the MAC is computed using the ciphertext, IV, and a key which differs depending on the type of packet. Using a MAC allows for the contents of the

packet to be verified and corruption or tampering to be detected. The IV, or initialization vector, is a single-use random value used to encrypt the ciphertext and compute the MAC. This ensures that the ciphertext and MAC values will be different even when sending the same data. Since the IV is random and the MAC is computed using the IV, both values are effectively random to an observer. The rest of the packet, excluding the padding, are encrypted into the ciphertext. The ciphertext includes a timestamp to protect against replay attacks, lengths for the data and padding, and the data itself. A separate padding length (PL) value is needed because several Dust packets may be contained inside a single UDP or TCP packet. Finally, a random number of random bytes of padding are added to randomize the packet length.

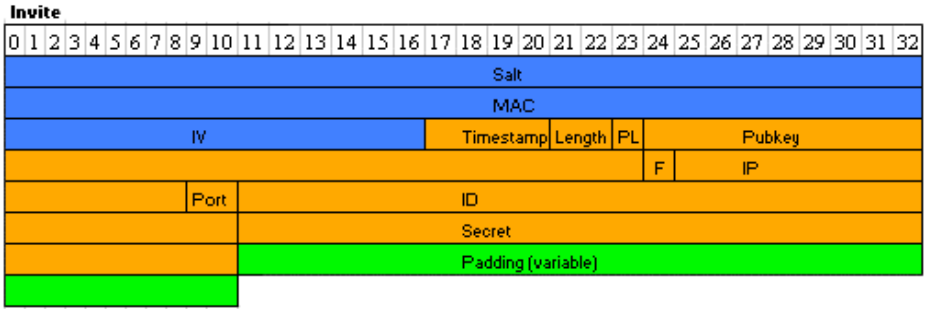

Fig. 2. The format of an invite packet.

An invite packet has the format show in Fig. 2. An invite packet, being a Dust packet, contains all of the same fields as a data packet, such as MAC, IV, and padding. The key used in an invite packet to encrypt the ciphertext and compute the MAC is a PBKDF function using a password and a random salt value. The salt value is prepended to the packet. The use of both salt and a PBKDF makes it difficult to decrypt the packet by brute force. This protects the contents of the invite packet against decryption unless the password is known.

The invite packet includes the information necessary for the client to connect to the server and complete the handshake. It contains the server's public key, the IP and port where the peer can be contacted, a flags byte which specifies if the peer accepts UDP or TCP connections and whether the IP is an IPv4 or IPv6 address, and an id and secret pair to be used in the completion of the handshake.

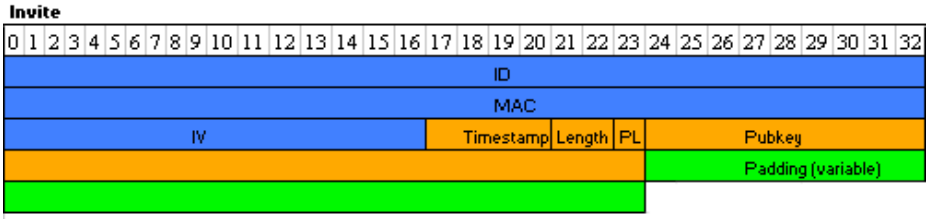

Fig. 3. The format of an intro packet.

An intro packet has the format shown in Fig. 3. The id used in the intro packet is the same as the one used in the invite packet. This is effectively a single-use random value as when it was contained in the invite packet it was encrypted and it is only seen in plaintext in the intro packet. The id is used by the server to link the intro packet to the stored single-use random secret. This secret is used to encrypt the ciphertext and to compute the MAC for the intro packet. Since each id is a single-use value, only one intro packet can be sent for each invite packet received by the client. The rest of the fields in an intro packet are the same as a general Dust packet. The content of an intro packet is the public key of the client.

Once the server has obtained the client's public key from the intro packet, the key exchange is complete and a shared session key is computed by both sides for use in encrypting the data packets. The data packets are simply general data packets are shown in Fig. 1 with no extra fields. In a data packet, the content is the data to be sent and the key used to encrypt the ciphertext and to compute the MAC is the shared session key derived from the exchanged public keys and locally stored private keys.

3. Discussion

The Dust protocol provides protocol unobservability by providing protection against the major methods of protocol fingerprinting through DPI. By encrypting or randomizing all bytes in all packets, static string matching is defeated. By randomizing packet length, length matching is defeated. By allow for a full conversation to be transmitted in a single UDP or TCP packet, timing attacks are defeated in the case of sufficiently small messages. Additionally, protection is provided against a number of specific attacks on the protocol. Packet corruption is defeated by use of a MAC. Replay attacks are defeated within a certain time window by use of a timestamp. Brute force decryption of invite packets are defeated by use of salt and a PBKDF. Additionally, any fields that are not encrypted are always randomized and single-use so that the attacker cannot gain additional information about the protocol even through long-term protocol observation.

Dust is designed to protect against current attacks, which are based on matching fingerprints of protocols against blacklists of known protocols. An obvious counteract against the Dust protocol is to switch from blacklist filtering to whitelist filtering. This is not addressed for two reasons. First, blacklists are the method currently in widespread use, whereas whitelists are not. Defeating blacklists is a significant step towards bypassing existing censorship attempts. Second, an approach which can bypass a whitelist has disadvantages over an approach designed to bypass blacklists. Steganography must be employed to encode traffic inside of whitelist-compatible traffic. As has been discussed, attempting this encoding allows for the possibility of introducing additional information that could be used for fingerprinting, such as filtering of the cover. The Dust approach is more simple and efficient to implement than a steganographic approach and so is preferable when only blacklist filtering is considered relevant.

4. Limitations

Dust does not attempt to protect against attacks that are already addressed by anonymizing proxies and censorship-resistance publishing systems. Specifically, no attempt is made to obscure sender or receiver IP addresses or ports or to protect server operators from coercion. These attacks would ideally be addressed by a system such as proposed by Kopsell consisting of an anonymizing proxy network allowing access to a censorship-resistant publishing system and using the Dust protocol as a blocking-resistant transport protocol.

In order for timestamps to be effective, Dust requires the client and server clocks to be reasonably synchronized, such as with NTP, as packets with out-of-date timestamps will be discarded. This is a possible area of future work for the protocol as clock synchronization may not always be available. Packet sequence numbers, logical clocks, and application-level clock synchronization are possible options to be considered for future revisions, although each comes with its own advantages and disadvantages.

Dust does not provide retransmission or reordering of dropped or reordered packets and provides no mechanism for acknowledgement of received packets. This is left to higher level protocols built on top of Dust. The reason for this is that Dust focuses on a minimal design that provides maximum blocking resistance. An ideal message for use with the current Dust protocol would fit inside a single UDP packet as this does not reveal any timing information that can be used for fingerprinting. Additional layers must be careful to not leak timing information to the attacker. This is considered to be a separate but related problem in unobservable protocol design.

No explicit mechanism for NAT hole punching is provided in the protocol. For IPv6 use, hole punching should not be necessary. For IPv4 use, Dust is compatible with and has been tested with Teredo, which provides end-to-end IPv6 connectivity on top of IPv4, including NAT hole punching even if both peers are behind NAT. In the case that Teredo has been blocked, TCP can be used instead of UDP as long as only the client is behind NAT. As implementing hole punching will complicate the protocol and open the way to timing attacks, the use case of a IPv4 server behind NAT without Teredo is left unsupported and would have to be implemented by individual applications when relevant.

5. Future Work

There are a number of enhancements to the Dust protocol that could protect against additional attacks. An obvious addition is a reliable transmission protocol on top of the basic Dust protocol which included packet acknowledgements. This would require a randomized packet scheduler in order to avoid leaking timing information. Once implemented, it could protect against packet loss attacks such as dropping the first packet between any two IPs, which in the case of Dust is the crucial introduction packet. Once a reliable protocol is available, a secondary key exchange could occur along with periodic key rotation, allowing for forward secrecy of conversations.

An additional area of research is how to add steganographic encoding to Dust packets. This would protect against whitelist attacks, but would require careful design to avoid leaking additional information to the attacker that could be used for fingerprinting. The problem of the blocking of the cover traffic would also need to be addressed.

In addition to the extension of the Dust protocol to protect against further attacks, there is also work to be done in the evaluation of the Dust protocol in real world scenarios. This is the most immediate next phase of research. Actual Dust traffic will be evaluated against real world censorship in current use on the Internet and its performance compared to other protocols used in circumvention technologies. The distinguishable characteristics of each protocol will be compared to determine their degree of protocol unobservability in both theoretical and practical terms.

6. Conclusion

Dust fills an important gap in the field of censorship resistance and privacy-enhancing technologies. By focusing exclusively on blocking resistance it solves real world attacks on existing censorship-resistant publishing and anonymous proxy systems. An ideal system combining the Dust protocol for communication, an anonymous proxy system for routing, and a censorship-resistant publishing system running as a hidden service would provide end-to-end unobservability and maximum protection against attackers.

Additionally, the design of the Dust protocol furthers the state of theory in the field by proposing an information theoretic bridge between censorship-resistant publishing, anonymous proxies, and blocking-resistant protocols based on the property of unobservability. A relatively unexplored area of the field is opened by proposing the centrality of blocking resistance instead of unlinkability in censorship resistance and the adoption of Kopsell's attack model in which an attacker which does not have the power of global eavesdropping.

Those wishing to use the Dust protocol for academic or practical purposes can find the source code for its implementation at http://github.com/blanu/Dust.

References

1. Bittau, A., Hamburg, M., Handley, M., Mazieres, D., and Boneh, D. The case for ubiquitous transport-level encryption. 19th USENIX Security Symposium., (2008).
2. Boesgaard, C.: Unlinkability and Redundancy of Content in Anonymous Publication Systems. http://www.diku.dk/hjemmesider/ansatte/pink/haven/unlink.pdf (2004)
3. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium. (2004)
4. Dingledine, R. Tor and circumvention: Lessons learned. The 26th Chaos Communication Congress, (2009).
5. Hand, S., Roscoe, T.: Mnemosyne: Peer-to-Peer Steganographic Storage. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. pp. 130-140. Springer-Verlag, Berlin (2002)

6. Harrison, D. BEP 008: Tracker Peer Obfuscation. Retrieved from: http://www.bittorrent.org/beps/bep_0008.html.
7. Hjelmvik, E and John, W. Breaking and Improving Protocol Obfuscation. Department of Computer Science and Engineering, Chalmers University of Technology, Technical Report No. 2010-05, ISSN 1652- 926X. (2010)
8. Hevia, A., Micciancio, D.: An Indistinguishability-Based Characterization of Anonymous Channels. In: Borisov, N., Goldberg, I. (eds.) PET 2008. pp. 24-43. Springer-Verlag, Berlin (2008)
9. Kopsell, S., Hilling, U.: How to Achieve Blocking Resistance for Existing Systems Enabling Anonymous Web Surfing. In: Proceedings of the Workshop on Privacy in the Electronic Society. pp. 103-115. ACM Press, New York (2004)
10. Leidl, B. Obfuscated-OpenSSH README. Retrieved from: https://github.com/brl/obfuscated-openssh/blob/master/README.obfuscation. (2010)
11. Message Stream Encryption. http://wiki.vuze.com/w/Message_Stream_Encryption (2006)
12. Obfuscated TCP. Wikipedia. Retrieved from: http://en.wikipedia.org/wiki/Obfuscated_TCP. (2010)
12. Perng, G., Reiter, M.K., Wang, Chenxi: Censorship Resistance Revisited. In: Barni, M. (ed.) IH 2005. pp. 62-76. Springer-Verlag, Berlin (2005)
13. Pfitzmann, A., Kohntopp, M.: Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology. In: Federrath, H. (ed.) Anonymity 2001. pp. 1-9. Springer-Verlag, Berlin (2001)
14. Sennhauser, M.: The State of Iranian Communication. http://diode.mbrez.com/docs/SoIN.pdf (2009)
15. Serjantov, A.: Anonymizing Censorship Resistant Systems. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. pp. 111-120. Springer-Verlag, Berlin (2002)
16. Topolsky, R. Comments of Robert M. Topolsky In the Matter of Petition of Free Press et al. for Declaratory Ruling that Degrading an Internet Application Violates the FCC's Internet Policy Statement and Does Not Meet an Exception for "Reasonable Network Management". Federal Communications Commission WC Docket No. 07-52, 08-7. (2008)
17. Using NetFlow Filtering or Sampling to Select the Network Traffic to Track. Retrieved from: http://www.cisco.com/en/US/docs/ios/netflow/configuration/guide/nflow_filt_samp_traff.html#wp1064305. (2006)
18. Waldman, M., Rubin, A.D., Cranor, L.F.: Publius: A Robust, Tamper-evident, Censorship-resistant Web Publishing System. In: 9th USENIX Security Symposium.
19. Waldman, M., Mazieres, D.: Tangler: a Censorship-resistant Publishing System Based on Document Entanglements. In: Computer and Communications Security. pp. 126-135. ACM Press, New York (2001)