Secure Communication over Diverse Transports

[Short Paper]

Michael Rogers
michael@briarproject.org

Eleanor Saitta
ella@dymaxion.org

ABSTRACT

This paper describes BTP, a protocol that ensures the confidentiality, integrity, authenticity and forward secrecy of communication over diverse underlying transports, from low-latency, bidirectional transports like TCP to high-latency, unidirectional transports like DVDs sent through the mail.

BTP is designed for use in censorship-resistant delay-tolerant overlays that operate over heterogeneous mixtures of underlying transports. By providing consistent security properties for a very wide range of transports, BTP simplifies the design and implementation of such overlays.

Forward secrecy is achieved by establishing an initial shared secret between each pair of endpoint devices and using a one-way key derivation function to generate a series of temporary shared secrets from the initial shared secret. Once both devices have destroyed a given temporary secret, any keys derived from it cannot be re-derived if the devices are later compromised.

BTP is designed to be compatible with traffic analysis prevention techniques such as traffic morphing: the protocol includes optional padding and uses no timeouts, handshakes or plaintext headers, with the goal of making it difficult to distinguish BTP from other protocols. If unlinkability between communicating devices is required, BTP can use anonymity systems such as Tor and Mixminion as underlying transports.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; E.3 [**Data**]: Data Encryption

Keywords

Delay-tolerant, forward secrecy, traffic analysis

1. INTRODUCTION

Communication across the internet is vulnerable to surveillance and censorship on an unprecedented scale: from a central point, a government or internet service provider can monitor the personal communications, reading habits and movements of an entire population. The potential for authoritarian abuse of such power is clear.

WPES'12, October 15, 2012, Raleigh, North Carolina, USA.

Many privacy-enhancing technologies have been developed to resist internet surveillance and censorship, but when the infrastructure on which such technologies operate is controlled by the adversary they seek to frustrate, there are limits as to what can be achieved. In extreme circumstances, governments may even be willing to shut down communication infrastructure in order to enforce censorship. Such actions have led to increasing interest in communication technologies that can function with or without internet access, including *delay-tolerant overlays*, which use store-and-forward techniques to communicate over a mixture of online and offline links.

In this paper we describe BTP, a transport protocol designed for use in censorship-resistant delay-tolerant overlays. BTP provides a secure unidirectional or bidirectional channel between two endpoint devices, ensuring the confidentiality, integrity, authenticity and forward secrecy of their communication. In separate work we are developing key agreement and messaging protocols that are intended to be used alongside BTP.

1.1 Adversary Model

Since BTP is intended to be used in systems that resist government surveillance and censorship, we must assume the existence of a powerful adversary:

- The adversary can observe, block, delay, replay and modify traffic on all underlying transports.
- The adversary can choose the data written to the BTP layer by higher protocol layers.
- The adversary has a limited ability to compromise endpoint devices. If a device is compromised, the adversary can access any information held in the device's volatile memory or persistent storage.
- The adversary cannot break standard cryptographic primitives such as block ciphers and message authentication codes.

1.2 Design Goals

BTP has the following design goals:

- **Flexibility.** BTP should be able to operate over a wide range of underlying transports, both unidirectional and bidirectional, with bandwidths varying from kilobits to gigabits per second, and with latencies varying from milliseconds to days.
- **Layering.** BTP should treat the underlying transport as a unidirectional or bidirectional byte stream with a simple socket-like interface (open, read/write, close). Likewise, higher protocol layers should be able to treat BTP as a unidirectional or bidirectional byte stream with a similar interface.

- **Concealability.** BTP should not reveal any plaintext fields that would make it easily distinguishable from other protocols. It should be compatible with techniques such as traffic morphing [1] that are designed to resist traffic analysis and traffic classification.
- **Confidentiality.** The adversary should not be able to learn what data is being transported across a BTP connection.
- **Integrity.** The adversary should not be able to cause either endpoint of a BTP connection to read data from the BTP layer that differs from the data written to the BTP layer by the other endpoint. If the adversary truncates a BTP connection, the receiving endpoint should be able to detect that this has happened.
- **Authenticity.** The adversary should not be able to cause either endpoint of a BTP connection to accept data from any third party as though it came from the other endpoint.
- **Forward secrecy.** The adversary should not be able to learn what data was transported across a BTP connection if, at some later time, the adversary compromises one or both of the endpoint devices.

BTP does not attempt to conceal the identities of the communicating parties or the fact that they are communicating – in other words, it does not provide anonymity, unlinkability or unobservability [2]. However, BTP can use anonymity systems such as Tor [3] and Mixminion [4] as underlying transports if unlinkability between the endpoints is required.

2. OVERVIEW OF THE DESIGN

Achieving forward secrecy without two-way communication is an unusual requirement that is not addressed by existing protocols. BTP addresses this requirement by using a *one-way key derivation function* to generate a series of *temporary shared secrets* from an *initial shared secret*. The initial shared secret is destroyed after deriving the first temporary secret. Each temporary secret is used for a single *rotation period* and destroyed at the end of its *retention period*, which extends beyond the rotation period.

Each endpoint can use the current temporary secret to derive keys for communicating with the other endpoint, even if there is never any communication in the other direction. Once both endpoints have destroyed a given temporary secret, no keys derived from it can be re-derived if the endpoint devices are later compromised.

The second unusual requirement addressed by BTP is concealability. The wire protocol includes optional padding and does not use any timeouts, handshakes or plaintext headers, making it suitable for use with traffic analysis prevention techniques such as traffic morphing. These features are intended to make it difficult for an observer to distinguish BTP from other protocols.

In principle the key derivation scheme and the wire protocol are independent – either could be used alone, but achieving all of the design goals listed above requires both parts of the design.

2.1 Underlying Transports

BTP can operate over any underlying transport that can deliver an opaque stream of bytes. We refer to these streams of bytes as *connections*. A connection may be unidirectional or bidirectional, depending on the nature of the underlying transport. Transports with very high latency are only used for unidirectional connections, even if they are able to carry data in both directions. Sending DVDs through the mail is an example of a unidirectional transport – each DVD carries a unidirectional BTP connection. TCP is an example of a bidirectional transport – each TCP connection carries a bidirectional BTP connection.

Underlying transports do not have access to unencrypted or unauthenticated data; nor are they required to ensure the confidentiality, integrity, authenticity or forward secrecy of the data they carry. BTP is responsible for providing those properties.

Transports must ensure that the bytes sent in each direction across a given connection are received in order, but the connections themselves may be reordered. For example, a transport based on DVDs sent through the mail must ensure that bytes are read from a given DVD in the same order as they were written, but it does not need to ensure that any two DVDs arrive in the same order as they were sent. Transports do not have to ensure that all connections reach their intended recipients.

A transport may impose a maximum connection length, such as the capacity of a storage medium. BTP passes this restriction on to higher protocol layers; it does not fragment connections.

To use a datagram-oriented transport such as UDP, which has no concept of connections, BTP requires an intermediate connection-oriented protocol such as UDT [5].

2.2 Prerequisites

Two parties, Alice and Bob, who wish to communicate using BTP must first establish an *initial shared secret*. BTP is designed to be used with a separate key agreement protocol that securely establishes the initial shared secret. We are developing such a protocol in separate work. BTP does not place any restrictions on the method used to establish the initial shared secret, except that it must not be possible to re-derive the secret from any information retained by the parties after the secret has been destroyed.

If Alice and Bob wish to communicate across more than one transport, they must establish a separate initial shared secret for each transport to ensure they do not reuse keys.

Alice and Bob must also agree on a *maximum latency* for each transport they wish to use. If the sender of a connection starts writing to the underlying transport at time T_0 and the recipient starts reading from the transport at time T_1, we call $T_1 - T_0$ the *latency* of the connection. For any transport t we can choose some *maximum latency*, L_t, such that the latency of any connection is unlikely to exceed L_t under normal conditions. For example, we might choose one minute as the maximum latency for TCP, or two weeks as the maximum latency for DVDs sent through the mail.

Finally, Alice and Bob must agree on a *frame length*, F_t, for each transport t they wish to use. For all transports, F_t must be at least 64 bytes and at most 2^{15} bytes (32 KiB). The impact of the frame length is discussed in section 6.

2.3 Cryptographic Primitives

BTP uses two cryptographic primitives: a block cipher and an authenticated encryption mode. The authenticated encryption mode must be a stream mode that can use a counter as an initialisation vector, and it must accept *additional authenticated data* that is authenticated but not encrypted. GCM [6] and OCB [7] are examples of suitable modes.

We use b to denote the cipher's block size, k for the key size, and m for the size of the message authentication code. All are measured in bytes. Since the authenticated encryption mode is a stream mode, each ciphertext is m bytes longer than the corresponding plaintext.

3. THE WIRE PROTOCOL

BTP's wire protocol is very simple. A unidirectional connection consists of a pseudo-random *tag* followed by one or more encrypted and authenticated *frames*. The tag can be calculated in advance

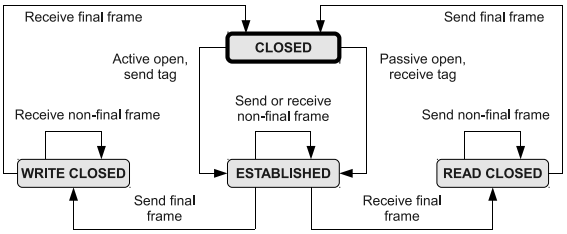

Figure 1: The state machine for bidirectional connections.

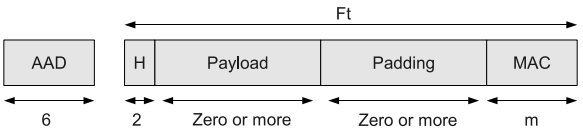

Figure 2: The format of a BTP frame. Lengths are in bytes. AAD is the additional authenticated data (not sent), H is the header and MAC is the message authentication code.

by the recipient, allowing the recipient to recognise the incoming connection without revealing any plaintext fields that would make it easy for an observer to distinguish BTP from other protocols.

The initiator's side of a bidirectional connection likewise consists of a tag followed by one or more frames. The responder's side has no tag, just one or more frames. Figure 1 shows the state machine for bidirectional connections. As with TCP, the sides of a connection may be closed independently.

3.1 Connection Numbers

The connections sent from Alice to Bob during a given rotation period are assigned sequential *connection numbers* starting from zero, as are the connections sent from Bob to Alice. The connection numbers used in each direction are independent from each other and from any other connection numbers used by Alice or Bob.

Neither endpoint can send more than 2^{32} connections to the other during a given rotation period. Each endpoint persistently stores the highest connection number it has sent to the other during the current rotation period, together with a sliding window of the highest connection numbers it has received from the other during each of the current retention periods. The persistent storage of these values is vital, so BTP cannot be used by endpoint devices that lack writable persistent storage (unless the devices never reboot).

3.2 Tags

The pseudo-random tag at the start of each connection is generated by encrypting a predictable *b*-byte plaintext with a *tag key* using the block cipher in ECB mode. Two tag keys are derived from each temporary secret: one for Alice's outgoing connections and the other for Bob's (see section 5.2 for details). The plaintext begins with the connection number as a 32-bit integer, with all other bits set to zero.[1]

It is safe to use ECB mode in this context because all plaintext blocks encrypted with a given tag key are guaranteed to be unique. The tags generated with a given tag key are also unique, and are otherwise indistinguishable from random by anyone who does not know the key, under standard assumptions about block ciphers.

3.3 Frames

BTP uses a simple frame format that does not require any plaintext headers. Every frame sent across a given transport, t, is exactly F_t bytes long, except the last frame on each side of each connection, which may be shorter. Each frame consists of a header, zero or more bytes of payload, and zero or more bytes of padding. All padding bytes must be set to zero. The header has the following format:

- Bit 0: Final frame flag. Raised if this is the final frame on this side of this connection, otherwise lowered.
- Bits 1–15: Length of the payload in bytes, as a 15-bit integer.

[1] All integers used by the protocol are big-endian and unsigned.

The final frame flag allows the recipient to distinguish between a connection that has been closed by the sender and a connection that has been truncated by the adversary.

The frames on each side of each connection are assigned sequential *frame numbers* starting from zero. No more than 2^{32} frames can be sent by either side. Each side uses a unique *frame key* derived from the current temporary secret (see section 5.2 for details). Each frame is encrypted and authenticated with the frame key using the block cipher's authenticated encryption mode. The initialisation vector begins with the frame number as a 32-bit integer, with all other bits set to zero. This initialisation vector is guaranteed to be unique for all frames using a given key. The additional authenticated data has the following format:

- Bits 0–31: Frame number as a 32-bit integer.
- Bit 32: Set to zero.
- Bits 33–47: Length of the plaintext (header, payload and padding) in bytes, as a 15-bit integer.

The ciphertext (including the m-byte message authentication code) is sent across the underlying transport, but the additional authenticated data is not. Instead, the sender and recipient calculate the additional authenticated data independently.

The recipient uses sequential frame numbers when calculating the additional authenticated data for received frames. If the adversary deletes or reorders frames, the sender and recipient will use different frame numbers when calculating the additional authetiticated data, so authenticated decryption will fail.

The recipient knows that the plaintext of every frame is exactly m bytes shorter than the ciphertext, and that the ciphertext of every frame except the last is exactly F_t bytes long. The recipient learns the length of the last frame by reading to the end of the underlying transport connection. If the adversary truncates the last frame, the sender and recipient will use different plaintext lengths when calculating the additional authenticated data, so authenticated decryption will fail.

If authenticated decryption fails, or if any frame has an invalid length or payload length or contains non-zero padding, the recipient must close the connection immediately without processing the frame's payload. Higher protocol layers can treat the payloads received from the BTP layer on each side of each connection as a continuous stream of bytes.

4. CONNECTION REORDERING

Since an endpoint does not know when it will receive each incoming connection, or whether connections will be lost or reordered, it must be ready to receive any of several incoming connections at any time. This is achieved by using a *connection reordering window* that spans several consecutive connection numbers. Connections outside the window will not be recognised by the recipient.

Each endpoint keeps a separate window for each of its current retention periods. A window consists of a connection number called the *centre*, which is initialised to zero, and a W_t-bit bitmap. The constant W_t is chosen to reflect the amount of connection loss and reordering expected from transport t. The $\lfloor W_t/2 \rfloor$ connection numbers below the centre and the $\lceil W_t/2 \rceil - 1$ numbers above it are contained in the window, excluding numbers outside the range $[0, 2^{32} - 1]$. The bitmap indicates which connections inside the window have been received.

For each connection inside the window that has not yet been received, the endpoint calculates the tag the sender will use and stores the resulting *expected tag* in a hashtable. By looking up the tag of an incoming connection in the hashtable, the recipient can identify the sender, the transport over which the connection should have arrived, the rotation period and the connection number, allowing the recipient to derive the appropriate frame key for each side of the connection.

When connection n is received, the centre of the window slides to $n+1$ unless the centre was already greater than n. The bitmap and hashtable are updated accordingly. Each window is persistently stored until the end of the respective retention period, at which time the window is destroyed and its tags are removed from the hashtable.

Expected tags do not need to be stored persistently, since they can be recalculated from the window and the temporary secret.

5. KEY MANAGEMENT

5.1 The Key Derivation Function

BTP's forward secrecy relies on the one-way nature of the key derivation function. If a function produces an output from a secret input and some public inputs, we call it a *one-way function* if it is not possible to derive from the output and the public inputs any information that would allow the secret input to be guessed with a better than random probability of success or calculated with less effort than brute force.

A block cipher fits this definition of a one-way function. When a block cipher is used to encrypt a known plaintext, we can regard the key as the secret input, the plaintext as the public input, and the ciphertext as the output. It is a standard design requirement for block ciphers that no information about the key can be obtained from any number of plaintexts and the corresponding ciphertexts. A block cipher is therefore a natural choice for constructing a one-way key derivation function.

The key derivation function used by BTP is described in section 5.1 of NIST SP 800-108 [8]. The secret input to the function is a k-byte key. The public inputs are an ASCII string called the *label* and a 32-bit integer called the *context*, which are combined with a counter to create $\lceil k/b \rceil$ distinct plaintext blocks. Each plaintext block starts with the null-terminated label followed by the four-byte context. At least one byte must be left free for the counter. Any unused bytes of the block are set to zero and the counter occupies the last byte. The counter starts at zero for the first block and is incremented for each subsequent block. Each block is encrypted with the block cipher in ECB mode. The output of the function is the $\lceil k/b \rceil$ concatenated ciphertext blocks, truncated to k bytes.

The label, context and counter are assumed to be known to the adversary; the k-byte key is the only secret input.

5.2 Key Derivation Inputs

Alice and Bob derive the first temporary secret from the initial shared secret using "ROTATE" as the label and zero as the context. The secret for period $r+1$ is derived from the secret for period r using "ROTATE" as the label and $(r+1) \pmod{2^{32}}$ as the context.

Alice's tag key for rotation period r is derived from the r^{th} temporary secret using "A_TAG" as the label and zero as the context. Bob's tag key uses the label "B_TAG".

The frame keys for Alice's n^{th} outgoing connection during rotation period r are derived from the r^{th} temporary secret using n as the context: the key for Alice's side of the connection uses the label "A_FRAME_A", while the key for Bob's side uses the label "A_FRAME_B". For Bob's outgoing connections, Bob's side uses the label "B_FRAME_B" and Alice's side uses the label "B_FRAME_A".

5.3 Rotation and Retention Periods

Each temporary secret is used for one rotation period and retained for one retention period. The rotation period may be arbitrarily short, but there are two practical constraints on the retention period.

The first constraint is the latency of the underlying transport. As discussed in section 2.2, Alice and Bob must agree on a maximum latency for each transport they wish to use. For example, they might choose one minute as the maximum latency for TCP, or two weeks as the maximum latency for DVDs sent through the mail. The maximum latency for transport t is denoted L_t.

The second constraint is loose clock synchronisation. Requiring the endpoints to have exactly synchronised clocks would restrict the circumstances under which BTP could be used, since many devices lack highly accurate clocks. Therefore we only require that the difference between the endpoints' clocks does not exceed some arbitrarily large constant, C. For devices with clocks that are set manually by their owners, we might reasonably assume that each clock differs from the correct time by up to half an hour, so one hour would be a suitable value for C. For devices with clocks that are set automatically via GPS [9] or NTP [10], ten seconds would be a conservative value for C.

To see how the constraints L_t and C affect the retention period, consider a connection sent across transport t during rotation period r. If the sender's clock is ahead of the recipient's clock, the connection may be received as much as C before the start of r according to the recipient's clock. On the other hand, if the sender's clock is behind the recipient's clock, the connection may be received as much as $C + L_t$ after the end of r according to the recipient's clock. Every temporary secret must therefore be derived C before its rotation period starts and retained for $C + L_t$ after its rotation period ends. Thus a transport with a rotation period of R_t has a retention period of $R_t + 2C + L_t$.

There is no necessary relationship between the rotation period R_t and the practical constraints L_t and C; even if L_t and C are large, R_t can be arbitrarily small. But the smaller the value of R_t, the more temporary secrets each endpoint will have to retain at any time. Key management is simplified if we choose $R_t = 2C + L_t$, so that only two temporary secrets need to be retained at any time.

Using the examples of L_t and C given above, the retention period would be 2 minutes 40 seconds for devices with automatic clocks communicating by TCP; 4 hours 2 minutes for devices with manual clocks communicating by TCP; 4 weeks 40 seconds for devices with automatic clocks communicating by mail; or 4 weeks 4 hours for devices with manual clocks communicating by mail.

6. DISCUSSION

We have tried to keep BTP's design as simple as possible. It is constructed from well-known primitives: block ciphers, authenticated encryption, one-way functions, key rotation and loosely syn-

chronised clocks. We use these primitives to address two unusual requirements: forward secrecy without two-way communication, and concealability.

The wire protocol is suitable for use with traffic morphing: frames can be arbitrarily segmented and delayed by the morphing process, while padding can be added if the morphing process requires more traffic than higher protocol layers are providing. Padding can also be used more straightforwardly to disguise the amount of data sent over a connection.

6.1 Applicability

Because it is designed for use in delay-tolerant overlays, BTP is optimised for latency-insensitive data transfer rather than latency-sensitive interactive applications. Fixed-length frames would be inefficient for interactive traffic, as a frame must be sent each time the sender needs to flush data to the underlying transport.

Each endpoint device must allocate storage for every other device with which it communicates. In the context in which we intend to deploy BTP it is reasonable to assume that each device will communicate with a limited number of other devices, but BTP may not be suitable for use in contexts where this assumption does not hold.

BTP does not provide proof of delivery for the data it transports. If proof of delivery is required, it must be provided by higher protocol layers.

6.2 Engineering Considerations

For each underlying transport they wish to use, Alice and Bob must each persistently store one temporary secret, one outgoing connection number and two connection reordering windows (assuming $R_t = 2C + L_t$). To avoid excessively frequent writes to persistent storage, we suggest a minimum rotation period of one minute (with a corresponding retention period of two minutes) regardless of the latency of the transport and the accurary of the endpoints' clocks.

When a device reboots after a period of inactivity it may need to 'fast-forward' through several rotation periods to reach the current period. Each temporary secret takes $\lceil k/b \rceil$ block cipher operations to derive, which should not be an excessive burden even for a low-powered device that has been inactive for thousands of periods. The tag key for each period can also be derived cheaply if there are any stored connections that need to be recognised.

The frame length, F_t, presents a tradeoff between buffering latency and framing overhead: shorter frame lengths will reduce buffering latency, while larger frame lengths will amortise the framing overhead across larger payloads. Different values of F_t are likely to be appropriate for different transports. For example, TCP might use a frame length of 512 bytes, whereas DVDs sent through the mail might use a frame length of 2^{15} bytes (32 KiB).

The size of the connection reordering window, W_t, presents another tradeoff, this time between memory overhead and reliable recognition of incoming connections: the larger the window, the more expected tags must be held in memory, but the more likely it is that a connection arriving out of order will be recognised.

The retention period presents a third tradeoff, between reliable connection recognition and forward secrecy. If a transport has highly variable latency it may be difficult to choose an appropriate maximum latency for the transport, and hence an appropriate retention period. If the retention period is too short, recipients may frequently fail to recognise incoming connections due to the corresponding temporary secrets having been destroyed. On the other hand, if the retention period is too long, connections with latency below the maximum will be vulnerable for longer than necessary to decryption if a device is compromised.

6.3 Security Considerations

Unlike most cryptographic protocols, BTP does not require any source of randomness apart from the initial shared secret. It follows, however, that the strength of every key depends on the strength of the initial shared secret – BTP cannot recover from a low-entropy initial shared secret.

BTP's forward secrecy relies on the ability to destroy old keys. This seems unavoidable given our design goals: unidirectional transports do not allow ephemeral keys to be agreed in-band, so the information required to decrypt each connection must be known to the recipient in advance. Unfortunately, securely deleting data from persistent storage – especially solid-state storage – is difficult with current operating systems and hardware [11]. The best defence may be to encrypt the stored data with a key derived from a passphrase, though that shifts the problem to protecting the passphrase.

If a device is compromised while it is inactive, it may contain secrets that have outlasted their retention periods but have not been destroyed. Again, passphrase-protected encrypted storage would seem to be the best defence.

The adversary can attack the connection reordering window by blocking traffic on an underlying transport until the sender's connection number exceeds the recipient's window; the recipient will then be unable to recognise the sender's connections even after the attack has ceased. BTP limits the impact of such attacks by using a new window for each rotation period, allowing the endpoints to recover within one retention period following the end of the attack.

7. RELATED WORK

Protocols for delay-tolerant networking have received considerable research attention in recent years, but that research has not generally aimed at resisting censorship. The IETF's delay-tolerant networking architecture "has a basic security model, optionally enabled, aimed at protecting infrastructure from unauthorized use" [12]. Kate *et al.* [13] describe an architecture for anonymous and secure communication in delay-tolerant networks; unlike BTP, it relies on a trusted central authority. Ferreira *et al.* [14] describe a transport-layer abstraction for peer-to-peer networks. Their abstraction is limited to bidirectional transports, but unlike BTP it is capable of operating over datagram-oriented transports such as UDP. Forward secrecy and concealability are not among their design goals.

Pseudo-random tags have previously been used to conceal the network traffic of anti-censorship systems [15, 16], and to conceal network services [17, 18].

One-way key derivation functions have been used in numerous protocols, including Lamport signatures [19] and the Guy Fawkes protocol [20]. Bellare and Yee [21] analyse the use of one-way functions to provide forward secrecy for symmetric encryption.

Brown *et al.* [22] describe how short-term encryption keys can be associated with long-term signature keys to provide forward secrecy for PGP [23]. To discover each other's short-term keys the communicating parties need access to an out-of-band key distribution mechanism such as a key server. In-band key distribution would require regular communication in both directions, a restriction we wish to avoid.

TLS [24] includes cipher suites that provide forward secrecy, but it cannot be used over unidirectional transports. OTR [25] is more flexible: after an initial key exchange, two parties can communicate intermittently in one or both directions, but forward secrecy is only achieved if there is regular communication in both directions.

Øverlier and Syverson [26] distinguish between *immediate for-*

ward secrecy and *eventual forward secrecy*. BTP provides eventual forward secrecy, as each connection is vulnerable to decryption until the end of the corresponding retention period.

Anderson [27] distinguishes between *forward security*, meaning that "the compromise of a key now does not necessarily expose future traffic", and *backward security*, meaning that "the compromise of a key now does not necessarily expose old traffic". The property we refer to as forward secrecy (looking forward from the moment the key is used) is what Anderson would call backward security (looking backward from the moment the key is compromised). The difference is one of perspective.

8. FUTURE WORK

We have tried to make it difficult for a passive observer to distinguish BTP from other protocols, but it may be possible to detect whether a device is accepting BTP connections through active probing. For example, every connection starts with a b-byte tag, so if a device always accepts $b-1$ bytes of random data but closes the connection after b bytes, it may be accepting BTP connections. Preventing such attacks is an important task for future work.

Other important issues that must be addressed in future work include attacks on the endpoints' clocks and revocation of compromised secrets. We have not formally proven the security of the wire protocol or the key derivation mechanism.

We are currently working on open source implementations of BTP and related key agreement and messaging protocols.[2]

9. ACKNOWLEDGEMENTS

The authors would like to thank the Small Media Foundation and the Open Internet Tools Project for supporting this work; Meredith L. Patterson, Emerson Tan and Ben Kurtz for helpful discussions; and Zooko Wilcox-O'Hearn, the anonymous reviewers and our shepherd Aniket Kate for their comments and advice.

10. REFERENCES

[1] C. Wright, S. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *16th Annual Network and Distributed Security Symposium*, 2009.

[2] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *Int. Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[3] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium*, 2004.

[4] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *IEEE Symposium on Security and Privacy*, 2003.

[5] Y. Gu. UDT: UDP-based data transfer protocol, 2010. `http://tools.ietf.org/html/draft-gg-udt-03`.

[6] M. Dworkin. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. Special Publication 800-38D, NIST, 2007.

[7] P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security*, 6(3), 2003.

[8] L. Chen. Recommendations for key derivation using pseudorandom functions (revised). Special Publication 800-108, NIST, 2009.

[2] `http://briarproject.org/`

[9] US Department of Defense. *Global Positioning System Standard Positioning Service Performance Standard*. Washington, DC: Department of Defense, September 2008.

[10] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network time protocol version 4: Protocol and algorithms specification. RFC 5905, IETF, 2010.

[11] M. Wei, L.M. Grupp, F.E. Spada, and S. Swanson. Reliably erasing data from flash-based solid state drives. In *9th USENIX Conf. on File and Storage Technologies*, 2011.

[12] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant networking architecture. RFC 4838, IETF, 2007.

[13] A. Kate, G.M. Zaverucha, and U. Hengartner. Anonymity and security in delay tolerant networks. In *3rd Int. Conf. on Security and Privacy in Communication Networks*, 2007.

[14] R.A. Ferreira, C. Grothoff, and P. Ruth. A transport layer abstraction for peer-to-peer networks. In *3rd Int. Symposium on Cluster Computing and the Grid*, 2003.

[15] A. Houmansadr, G.T.K. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *18th ACM Conf. on Computer and Communications Security*, 2011.

[16] E. Wustrow, S. Wolchok, I. Goldberg, and J.A. Halderman. Telex: Anticensorship in the network infrastructure. In *20th USENIX Security Symposium*, 2011.

[17] P. Barham, S. Hand, R. Isaacs, P. Jardetzky, R. Mortier, and T. Roscoe. Techniques for lightweight concealment and authentication in IP networks. Technical Report IRB-TR-02-009, Intel Research Berkeley, 2002.

[18] E.Y. Vasserman, N. Hopper, J. Laxson, and J. Tyra. SilentKnock: Practical, provably undetectable authentication. In *12th European Symposium on Research in Computer Security*, 2007.

[19] L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Palo Alto, CA, USA, 1979.

[20] R.J. Anderson, F. Bergadano, B. Crispo, J.H. Lee, C. Manifavas, and R.M. Needham. A new family of authentication protocols. *Operating Systems Review*, 32(4):9–20, 1998.

[21] M. Bellare and B. Yee. Forward-security in private-key cryptography. In *Cryptographers' Track, RSA Security Conf. (CT-RSA)*, 2003.

[22] I. Brown, A. Back, and B. Laurie. Forward secrecy extensions for OpenPGP, 2005. `http://www.links.org/dnssec/draft-brown-pgp-pfs-04.txt`.

[23] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP message format. RFC 4880, IETF, 2007.

[24] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol, version 1.2. RFC 5246, IETF, 2008.

[25] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *Workshop on Privacy in the Electronic Society*, 2004.

[26] L. Øverlier and P. Syverson. Improving efficiency and simplicity of Tor circuit establishment and hidden services. In *7th Workshop on Privacy Enhancing Technologies*, 2007.

[27] R. Anderson. Two remarks on public key cryptology. Technical Report 549, University of Cambridge Computer Laboratory, 2002.