Digital Investigation 11 (2014) S77–S86

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

BitTorrent Sync: First Impressions and Digital Forensic Implications

Jason Farina[*], Mark Scanlon, M-Tahar Kechadi

UCD School of Computer Science and Informatics, University College Dublin, Dublin 4, Ireland

Keywords:
BitTorrent
Sync
Peer-to-Peer
Synchronisation
Privacy
Digital forensics

A B S T R A C T

With professional and home Internet users becoming increasingly concerned with data protection and privacy, the privacy afforded by popular cloud file synchronisation services, such as Dropbox, OneDrive and Google Drive, is coming under scrutiny in the press. A number of these services have recently been reported as sharing information with governmental security agencies without warrants. BitTorrent Sync is seen as an alternative by many and has gathered over two million users by December 2013 (doubling since the previous month). The service is completely decentralised, offers much of the same synchronisation functionality of cloud powered services and utilises encryption for data transmission (and optionally for remote storage). The importance of understanding BitTorrent Sync and its resulting digital investigative implications for law enforcement and forensic investigators will be paramount to future investigations. This paper outlines the client application, its detected network traffic and identifies artefacts that may be of value as evidence for future digital investigations.

Introduction

With home user bandwidth rising and the growth in professional and non-professional computer power, the volume of data created by each individual computer user is constantly growing. For mobile users, access to this data has long been an issue. With greater connectivity and greater availability of access to the Internet the concepts of “high availability”, “off-site backup” and “resilient storage” have moved away from the domain solely inhabited by large corporations and has started to become increasingly popular with computer users and everyday data consumers. Applications such as Evernote and Dropbox leverage the decreasing cost of hard disk storage seen in Storage as a Service (SaaS) providers, e.g., Amazon S3, to provide data storage on the cloud to home users and businesses alike. The main advantage of services such as Dropbox, Google Drive, Microsoft OneDrive (formerly SkyDrive) and Apple iCloud to the end user is that their data is stored in a virtual extension of their local machine with no direct user interaction required after installation. It is also backed up by a full distributed data-centre architecture that would be completely outside the financial reach of the average consumer. Their data is available anywhere with Internet access and is usually machine agnostic so the same data can be accessed on multiple devices without any need to re-format partitions or wasting space by creating multiple copies of the same file for each device. Some services such as Dropbox, also have offline client applications that allow for synchronisation of data to a local folder for offline access.

Each of the aforementioned services can be categorised as cloud synchronisation services. This means that while the data is synchronised between user machines, a copy of the data is also stored remotely in the cloud. In recent headline news, much of this data is freely available to

* Corresponding author.
E-mail addresses: jason.farina@ucdconnect.ie (J. Farina), mark.scanlon@ucd.ie (M. Scanlon), tahar.kechadi@ucd.ie (M-T. Kechadi).

http://dx.doi.org/10.1016/j.diin.2014.03.010

governmental agencies without the need of a warrant or even just cause. As a result, BitTorrent Sync (also referred to as BTSync, BitSync or BSync), which provides much of the same functionality without the cloud storage aspect is seen by many as a real alternative. The service has numerous desirable attributes for any Internet user (BitTorrent Inc, 2013a):

- Compatibility and Availability – Clients are built for most common desktop and mobile operating systems, e.g., Windows, Mac OS, Linux, BSD, Android and iOS.
- Synchronisation Options – Users can choose whether to sync their content over a local network or over the Internet to remote machines.
- No Limitations or Cost – Most cloud synchronisation services provide a free tier offering a small amount of storage and subsequently charge when the user outgrows the available space. BTSync eliminates these limitations and costs. The only limitation to the volume of storage and speed of the service is down to the limitations of the synchronised users machines.
- Automated Backup – Like most competing products, once the initial install and configuration is complete, the data contained within specified folders is automatically synchronised between machines.
- Decentralised Technology – All data transmission and synchronisation takes place solely in a Peer-to-Peer (P2P) fashion, based on the BitTorrent file sharing protocol.
- Encrypted Data Transmission – While synchronising data between computers on different LANs (the option exists to apply encryption for internal LAN transmission), the data is encrypted using RSA encryption. Under the BTSync API (BitTorrent Inc, 2013b), developers can also enable remote file storage encryption. This could result in users storing their data on untrusted remote locations for the purposes of redundancy and secure remote backup.
- Proprietary Technology – The precise protocol and operation of the technology is not openly documented by the developer resulting in an element of perceived “security through obscurity”. Of course, this requires a significant degree trust on behalf of users that the developers' security has been implemented and tested correctly.

As a result of the above, the BTSync application has become a very popular choice for file replication and synchronisation. The technology had grown to over one million users by November 2013 and doubled to over two million users by December 2013 (BitTorrent Inc, 2013c). The service will be of interest to both law enforcement and digital forensics investigators in future investigations. Like any other file distribution technology, this interest may be centred around recovering evidence of the data itself, of the modification of the data or of where the data is synchronised to. While the legitimate usage of the system, e.g., backup and synchronisation, teamwork, data transfer between systems, etc., may be of interest to an investigation, the technology may also be a desirable one for a number of potential crimes including industrial espionage, copyright infringement, sharing of child exploitation material, malicious software distribution, etc.

Contribution of this work

The contribution of this work includes a forensic analysis of the BTSync client application, its behaviour, artefacts created during installation and use, and remnants left behind after uninstallation. An analysis of the sequence of network traffic and file I/O interactions used as part of the synchronisation process are also provided. This information should prove useful to digital forensic investigators when BTSync is found to be installed on a machine under investigation. Gaining an understanding of how BTSync operates could aid in directing the focus of a digital investigation to additional remote machines where any pertinent data is replicated. Depending on the crime under investigation, these remote machines may be owned and operated by a single suspect or by a group sharing a common goal. While an initial analysis of the network protocol and its operation is included below, comprehensive network analysis is beyond the scope of this paper.

Background

In order to understand how BTSync operates, its important to first understand the technologies its based upon and how a number of similar technologies operate. This section provides some of the required background information.

BitTorrent file sharing protocol

The BitTorrent protocol was designed with the aim of facilitating one-to-many and many-to-many file transfers as efficiently as possible. The protocol is described in BitTorrent Enhancement Proposal (BEP) No. 3 (Cohen, February 2014). The main strength of the protocol is the usage of file parts, each of which can be manipulated and managed separately. While one part of a file downloads, another, already downloaded part can be uploaded to a different peer. In this way, peers can start trading parts even before they have downloaded the entire file themselves. This has the benefit of not only speeding up distribution as each peer can find useful information on a broad range of potential peers but it also helps alleviate the issues of “churn” (Stutzbach and Rejaie, 2006) “Data Leeching” and “free riding” (Karakaya et al., 2009) experienced with older protocols such as Gnutella and eDonkey. Data leeching is where a user downloads an entire file in one go and then removes the share to avoid uploading. Data churn is the natural expansion and retraction of the network horizon as peers leave and join the “swarm” freely resulting in a large variance in the availability of full versions of a file being available from individual sources.

The overall BitTorrent network can be seen as being subdivided into BitTorrent “swarms”. Each swarm consists of a collection of peers involved in the sharing of the same file. The central commonality of a swarm is a unique identifier created from a SHA-1 hash of the file(s) references in the metadata. A peer can be a member of multiple swarms as multiple files are uploaded and downloaded simultaneously. In order to initiate download of content from a particular swarm the user must first download a metadata `.torrent` file (or corresponding magnet URI) from an indexing website. The BitTorrent client application running on the users machine then interprets the metadata and uses it to locate other peers actively participating in that swarm using one or more of the following methods (Scanlon et al., 2010):

1. Tracker Server – Tracker servers are Internet accessible servers that maintain a list of `seeders` (those peers with 100% of a file available and as such are only uploading data) and `leechers` (peers that are beginning the process or are in the middle of the process of downloading information from the swarm) (Cohen, 2003). While the data transfer is in progress, the client application will periodically report to the tracker to update its status and to update its list of active peers.
2. Distributed Hash Table (DHT) – While the original BitTorrent protocol was designed with central repositories of peers stored on servers, clients were developed such as Vuze and μTorrent that also stored a list of active clients on the local machine. This common DHT allows peers to identify peers through requesting information from other BitTorrent clients without the requirement for a central server (these clients serving information from the DHT are likely not involved in the requested swarm). Each peer record in the DHT is associated with the swarms in which it is actively participating. The Mainline DHT, as outlined in BEP No. 5 (Cohen, February 2014), that is used by BitTorrent and BTSync is based on the Kademlia protocol and allows for completely decentralised discovery of peers associated with sharing a particular piece of content (identified by the SHA-1 hash of the content).
3. Peer Exchange (PEX) – Originally, the BitTorrent protocol did not allow for any direct communication between peers beyond the transmission of data, but various extensions of the protocol have resulted in the removal of this restriction. As DHT participation became commonly supported in the major BitTorrent clients, peers started to exchange the local peer caches. Peer Exchange is a BEP outlined a method for when two peers are communicating (sharing the data referenced by a torrent file), a subset of their respective peer lists are shared back and forth as part of the communication. Coupled with DHT, PEX removes a potential vulnerability from the BitTorrent network by allowing for fully distributed bootstrapping, tracking and peer discovery.

Any metadata or network control requests/responses are transmitted using “bencoding”, as explained in BEP No. 3 (Cohen, February 2014). Bencoded data consists of dictionaries and lists consisting of `key:value` pairs. Each key name and corresponding value is prepended by the length (in bytes) followed by a colon. For example the `get_peers` request message can be bencoded as `1:m9:get_peers` (with the ‘m’ representing the key name “message”).

BitTorrent Sync

BTSync is a file replication utility created by BitTorrent Inc. and released as a private alpha in April 2013 (BitTorrent Inc, 2013a). It is not a cloud backup solution, nor necessarily intended as any form of offsite storage. Any data transferred using BTSync resides in whole files on at least one of the synchronised devices. This makes the detection of data much simpler for digital forensic purposes as there is no distributed file system, redundant data block algorithms or need to contact a cloud storage provider to get a list of all traffic to or from a container using discovered credentials. The investigation remains an examination of the local suspect machine. However, because BTSync optionally uses a DHT to transfer data there is also no central authority to manage authentication or log data access attempts. A suspect file found on a system may have been downloaded from one or many sources and may have been uploaded to many recipients.

While the paid cloud synchronisation services offer up to 1 TB of storage (Amazon S3 paid storage plan) the free versions which are much more popular with home users cap at approximately 10 GB. The BTSync storage is limited only by the size of the folder being set as a share (most likely limited by the available disk space). Unless the system under investigation is the creator of the shared folder, it is possible that any files contained therein may have been downloaded without the user’s prior knowledge of the folder’s contents. The BTSync application does not feature a built in content preview utitily. As a result, it blindly and completely synchronises all content within the shared folder without any file selection process available to the user.

Related work

At the time of publication, there are no academic publications focussing on BTSync. However, due to BTSync sharing a number of attributes and functionalities with cloud synchronisation services, e.g., Dropbox, Google Drive, etc., and it is largely based on the BitTorrent protocol, there are a number of relevant related topics of interest. This section outlines a number of related case studies and investigative techniques for these shared technologies. While the specific attributes of a number of popular cloud synchronisation services are outlined below, there is a common generalised architecture employed by these services. There are two main stages to this synchronisation process, as shown in Fig. 1:

- Stage 1 – The local client with the source file (the seeder in P2P terms) and the remote replication target (leecher) both contact the server of authority belonging to the service being used to confirm their credentials.
- Stage 2 – Both seeder and leecher contact the remote storage location, usually cloud based for high availability. The seeder uploads a full copy of each file to be replicated and the leecher downloads a full version of the files it finds in the cloud storage container.

At no point in the process do the clients have to talk directly to one another. An important feature of these

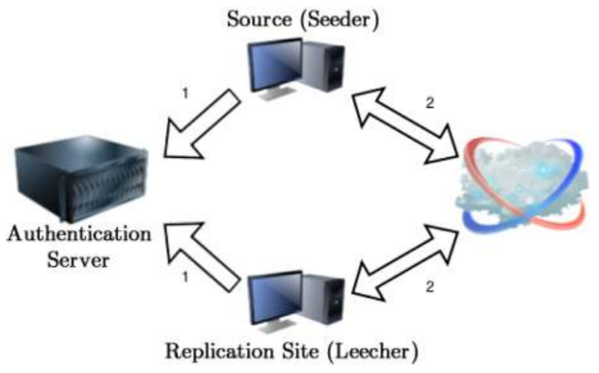


Fig. 1. Operation of cloud file synchronisation services.

services is the fact that there is a full copy of the data being stored on a remote third party server outside the control of either client.

Forensic analysis of cloud synchronisation clients

Forensic investigation of these utilities can be challenging, as presented by Chung et al. in their 2012 paper (Chung et al., 2012). Unless local synchronisation is completely up to date, the full picture of the data may reside across temporary files, volatile storage (such as the system's RAM) and across multiple data-centres of the service provider's cloud storage facilities. Any digital forensic examination of these systems must pay particular attention to the method of access, e.g., usually the Internet browser connecting to the service provider's access page. This temporary access serves to highlight the importance of live forensic techniques when investigating a suspect machine. Cutting power to the suspect machine may not only lose access to any currently opened documents, but would also lose any currently stored passwords or other authentication tokens that are stored in RAM. Chung et al. describe three main forms of online storage in use by consumers:

1. Data Storage for Large Data – Examples would include the services provided by Amazon S3, Dropbox, Google drive, etc.
2. Online Only Office Applications – This includes services whereby an entire productivity suite of tools is accessed in a completely self contained online environment, e.g., Google Docs, Office 365 or Sage Online.
3. Personal Data – Examples would include Evernote, which allows users to save notes to a central store, and Spotify, which allows playlists to be stored in the cloud when users build their online music catalogue.

Cloud file synchronisation services

In various complementary papers on data remnants (Quick et al., 2013a, 2013b, 2013c), Quick et al. offers an additional approach to forensics when dealing with cloud storage investigation. This involves access using the full client application whether or not it has been tampered with by the end user, e.g., perhaps an anti-forensics attempt was made to hide data by uninstalling the application and deleting the synchronised folders. Each of the applications examined stored their authentication credentials on the local system while the client was actively connected to the service, again highlighting the importance of live forensic recovery techniques. It should be noted that while Dropbox and Microsoft OneDrive appear to be very similar utilities, there are distinct differences in the way they are intended to be used. Dropbox (when used with the client application) creates a local folder that synchronises any contents stored in it with an online duplicate of that folder. By default, Dropbox gives 2 GB of storage for free with an option to buy additional storage. OneDrive on the other hand is intended as a predominantly online storage facility with an option to synchronise a copy of the files to a client machine folder. However, this is not the default behaviour and has to be specifically enabled if used as part of the Windows 8.1 operating system. For non-Windows 8 based computers, the user is required to download and install the OneDrive desktop application to enable file synchronisation across devices.

Many Cloud storage utilities provide a method of synchronisation of files which involves some form of periodic checking to determine if changes have been made to any version being viewed locally or to compare offline copies with their online counterparts as soon as communication can be re-established (network connectivity re-enabled or the application or service restarted). For Dropbox, Drago et al. (Drago et al., 2012) identified two sets of servers, the control servers owned and operated by Dropbox themselves and the storage management and cloud storage servers hosted by Amazon's EC2 and S3 services. This identification is also verified by Wang et al. (Wang et al., 2012).

BTSync application & protocol analysis

Table 1 shows the hardware and virtual machines used to perform an analysis on the BTSync application. The tool was installed on all machines outlined using the default installation parameters. A complete list of the files created during the install process is outlined in Table 2.

Default installation includes the creation of a BTSync folder (the location on Windows based machines is `$Volume$\Documents and Settings\ [User]\BTSync`). This folder is automatically populated with three files:

1. `.SyncID` – Stores a 20 byte unique share ID
2. `.SyncIgnore` – A list of files in the folder or subfolder to ignore when synchronising with remote machines.

Table 1
Hardware used in the analysis of the BitTorrent sync application.

Name	Host 1:	Guest 1:
OS	Windows 7 PC (64 bit)	Windows XP SP3
Ram	8 GB ram	512 mb RAM
	Vmware Workstation 8	Bridged network adapter
Name	Host 2:	Guest 2:
OS	Linux Debian laptop	Widows XP SP3
Ram	4 GB ram	512 mb RAM
	VirtualBox 4.2	Bridged network Adapter

Table 2
BitTorrent sync default application files.

File	Purpose
$Volume$\Program Files\BitTorrent Sync\BTSync.exe	Main Executable
$Volume$\Documents and Settings\[User]\Application Data\Microsoft\Crypto\<user SID>	Private Key
$Volume$\Documents and Settings\[User]\Application Data\Bittorrent Sync	Application folder
$Volume$\Documents and Settings\[User]\Application Data\Bittorrent Sync\settings.dat	Configuration Settings
$Volume$\Documents and Settings\[User]\Application Data\Bittorrent Sync\sync.log	Log of Synchronisation Activity
$Volume$\Documents and Settings\[User]\Application Data\Bittorrent Sync\sync.lng	Language File
$Volume$\Documents and Settings\All Users\Desktop\BitTorrent Sync.lnk	Application Shortcut
$Volume$\Documents and Settings\All Users\Start Menu\BitTorrent Sync.lnk	Application Shortcut
$Volume$\Documents and Settings\All Users\Quick Start\BitTorrent Sync.lnk	Application Shortcut

3. `.SyncArchive` (Folder) – An archive to store files that were deleted on a remote synchronised system.

These three files are created whenever any new BTSync share is set up and are used to aid in the control of data exchange between the nodes.

On Linux based machines, the installation directory is wherever the user chooses to unpack the application package. All of the same files are created included the hidden folders. In addition the user interface is a web GUI on `localhost:8888` and the application can generate a configuration file by running the command "`./btsync --dump-sample-config`" from the terminal. If this plain text file is edited it can be used to overwrite the username and password for the web GUI to allow the investigator access without changing any other settings.

BTSync client activity

The options for synchronisation and replication are set for each share on the local machine. As shown in Fig. 2, there are three main distinct settings determining the resources used for peer discovery and the paths available for traffic transmission. BTSync uses similar peer discovery methods to the regular BitTorrent protocol. These methods are outlined below:

1. LAN Discovery – If the option ``Search LAN'' is enabled the client application will start sending peer discovery packets across the LAN utilising the multicast address `IP 239.192.0.0 Port: 3838`. These packets, as displayed in Fig. 3, are sent at a frequency of one every 10 seconds for each share utilising this method.

The local peer discovery packet has a BSYNC header and a message type of "ping" and includes the sending host's IP address, port and the 20 byte ShareID of the share being advertised. Hosts on the LAN receiving the packet will drop it if the ShareID is not of interest to them. Any host that has an interest will respond with a UDP packet to the port advertised. The response does not have a BSYNC header present and the data field only contains the PeerID of the responding peer. This discovery is restricted to Path 'A' in Fig. 2.

2. Tracker – The option "`Use Tracker`" causes the client to search for peers by requesting a peer list from the tracker located at `t.usyncapp.com` which was resolves to three IP addresses:
   - 54.225.100.8
   - 54.225.92.50
   - 54.225.196.38

These three IP addresses are each hosted on Amazon's EC2 cloud service. The client sends a `get_peers` request to the tracker server (as can be seen in Fig. 4). When this request is received, the client's IP addresses gets added to the list of active peers available for that particular ShareID on the tracker. The path to the tracker server taken by the peers is displayed as Path 'B' of Fig. 2. The information keys contained in the `get_peers` message are shown in Table 3. The peer discovery response, as displayed in Fig. 5 consists of a list of bencoded `IP:Port:PeerID:ShareID` entries identifying the known peers with the same secret. Due to the fact that the client only requests this list for a secret it already possesses, the response from the server will always contain at least one active peer, i.e., the requesting client's information.

3. Distributed Hash Table (DHT) – The client can be set to perform peer discovery using a DHT. Using this option,

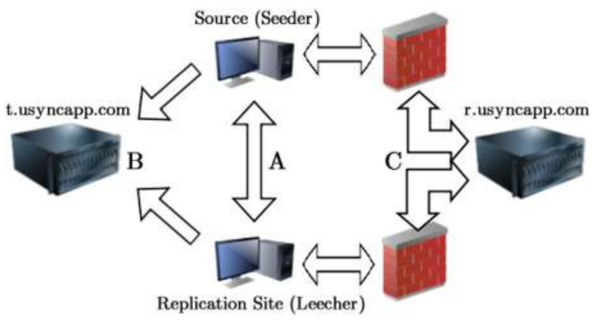


Fig. 2. BTSync synchronisation options.

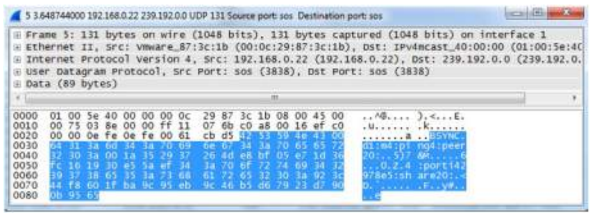
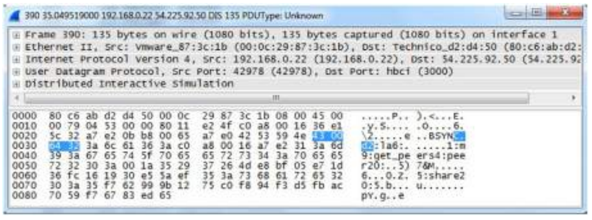

```
5 3.648744000 192.168.0.22 239.192.0.0 UDP 131 Source port: sos  Destination port: sos
⊞ Frame 5: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits) on interface 1
⊞ Ethernet II, Src: Vmware_87:3c:1b (00:0c:29:87:3c:1b), Dst: IPv4mcast_40:00:00 (01:00:5e:40
⊞ Internet Protocol Version 4, Src: 192.168.0.22 (192.168.0.22), Dst: 239.192.0.0 (239.192.0.
⊞ User Datagram Protocol, Src Port: sos (3838), Dst Port: sos (3838)
⊞ Data (89 bytes)

0000  01 00 5e 40 00 00 00 0c  29 87 3c 1b 08 00 45 00   ..^@.... ).<...E.
0010  00 75 03 8e 00 00 ff 11  07 6b c0 a8 00 16 ef c0   .u...... .k......
0020  00 00 0e fe 0e fe 00 61  cb d5 42 53 59 4e 43 00   .......a ..BSYNC.
0030  64 31 3a 6d 34 3a 70 69  6e 67 34 3a 70 65 65 72   d1:m4:pi ng4:peer
0040  32 30 3a 00 1a 35 29 37  26 4d e8 bf 05 e7 1d 36   20:..5)7 &M.....6
0050  fc 16 19 30 e5 5a ef 34  3a 70 6f 72 74 69 34 32   ...0.2.4 :porti42
0060  39 37 38 65 35 3a 73 68  61 72 65 32 30 3a 92 3c   978e5:sh are20:.<
0070  44 f8 60 1f ba 9c 95 eb  9c 46 b5 d6 79 23 d7 90   D.`..... .F..y#..
0080  0b 95 65                                           ..e
```

Fig. 3. BTSync multicast "Seeker" packet.



```
390 35.049519000 192.168.0.22 54.225.92.50 DIS 135 PDUType: Unknown
⊞ Frame 390: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits) on interface 1
⊞ Ethernet II, Src: Vmware_87:3c:1b (00:0c:29:87:3c:1b), Dst: Technico_d2:d4:50 (80:c6:ab:d2:
⊞ Internet Protocol Version 4, Src: 192.168.0.22 (192.168.0.22), Dst: 54.225.92.50 (54.225.92
⊞ User Datagram Protocol, Src Port: 42978 (42978), Dst Port: hbci (3000)
⊞ Distributed Interactive Simulation

0000  80 c6 ab d2 d4 50 00 0c  29 87 3c 1b 08 00 45 00   .....P.. ).<...E.
0010  00 79 04 53 00 00 80 11  e2 4f c0 a8 00 16 36 e1   .y.S.... .O....6.
0020  5c 32 a7 e2 0b b8 00 65  a7 e0 42 53 59 4e 43 00   \2.....e ..BSYNC.
0030  64 32 3a 6c 61 36 3a c0  a8 00 16 a7 e2 31 3a 6d   d2:la6:. .....1:m
0040  39 3a 67 65 74 5f 70 65  65 72 73 34 3a 70 65 65   9:get_pe ers4:pee
0050  72 32 30 3a 00 1a 35 29  37 26 4d e8 bf 05 e7 1d   r20:..5) 7&M.....
0060  36 fc 16 19 30 e5 5a ef  35 3a 73 68 61 72 65 32   6...0.Z. 5:share2
0070  30 3a 35 f7 62 99 9b 12  75 c0 f8 94 f3 d5 fb ac   0:5.b... u.......
0080  70 59 f7 67 83 ed 65                               pY.g..e
```

Fig. 4. BTSync tracker request packet.

any peer will register its details in the form of `SHA-1(Secret):IP:Port` with other peers, even those that do not participate in the swarm. Using this option a user can avoid using any form of tracking server but they may find that peer discovery is slower or less complete.

4. Known Peers – The final, and least detectable, method of peer discovery is the option to ``` ``Use Predefined Hosts'' ```. The user can add a list of IP address:Port combinations to the share preferences. This list of peers will be contacted directly without any lookup with a BSYNC packet containing a `ping` message type.

Data transfer

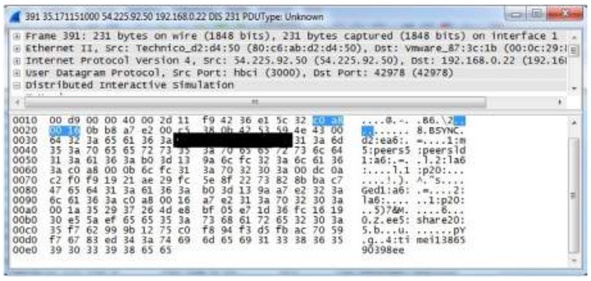
Similar to peer discovery methods, BTSync allows the user to configure a number of options that affect how data is transferred between peers:

1. No options set (Path ‘A’ in Fig. 2). The seeding host will attempt to communicate directly with the replication target (the leecher). This traffic is encrypted by default if it travels outside the local LAN. There is an option in the application preferences to enable or disable encryption of LAN traffic as well if the user prefers.
2. If the communication between the hosts is blocked for some reason, usually if the hosts are on different networks protected by firewalls or in segments or subnets of the same LAN locked down by inbound Access Control Lists, the option ``` ``Use Relay Server when required'' ``` will allow the traffic to bypass these restrictions if possible (this is represented by Path ‘C’ in Fig. 2). The relay server, located at `r.usyncapp.com` resolves to:
   - relay-01.utorrent.com (67.215.229.106)
   - relay-02.utorrent.com (67.215.231.242)

These packets are sent via UDP to port 3000 and contain “ping” messages, as can be seen in Fig. 6. These ping messages contain a 20 byte PeerID and a 32 byte ShareID derived from the secret key. After the initial handshake with the relay server the relay negotiates the data transmission session with the remote peer. This negotiation involves exchange of the 16 byte “nonce” (a one off value used for encryption purposes) and a map of the availability of the file parts, as can be seen in Fig. 7. Once the handshake is complete, the next packet contains the 160 bit public key and the encrypted transfer of data begins. The responsibility for the actual data transfer is retained by the individual clients and only metadata and ping packets are sent unencrypted.

Table 3
Component fields for request packet.

Key	Explanation
d:	[The Entire Dictionary]
la:	[IP:Port in Network-Byte Order]
m:	[Message Type Header, e.g., get_peers]
peer:	[Local Peer ID]
share:	[Local Share ID]
e:	[End]



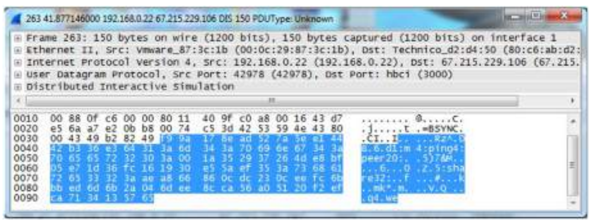
```
391 35.171151000 54.225.92.50 192.168.0.22 DIS 231 PDUType: Unknown
⊞ Frame 391: 231 bytes on wire (1848 bits), 231 bytes captured (1848 bits) on interface 1
⊞ Ethernet II, Src: Technico_d2:d4:50 (80:c6:ab:d2:d4:50), Dst: Vmware_87:3c:1b (00:0c:29:
⊞ Internet Protocol Version 4, Src: 54.225.92.50 (54.225.92.50), Dst: 192.168.0.22 (192.16
⊞ User Datagram Protocol, Src Port: hbci (3000), Dst Port: 42978 (42978)
⊟ Distributed Interactive Simulation

0010  00 d9 00 00 40 00 2d 11  f9 42 36 e1 5c 32 c0 a8   ....@.-. .B6.\2..
0020  00 16 0b b8 a7 e2 00 c5  38 0b 42 53 59 4e 43 00   ........ 8.BSYNC.
0030  64 32 3a 65 61 36 3a [redacted] 31 3a 6d   d2:ea6:. =....1:m
0040  35 3a 70 65 65 72 73 35  3a 70 65 65 72 73 6c 64   5:peers5 :peersld
0050  31 3a 61 36 3a b0 3d 13  9a 6c fc 32 3a 6c 61 36   1:a6:.=. .l.2:la6
0060  3a c0 a8 00 0b 6c fc 31  3a 70 32 30 3a 00 dc 0a   :....l.1 :p20:...
0070  c2 f0 f9 19 21 ae 29 fc  5e 8f 22 73 82 8b ba c7   ....!.). ^."s....
0080  47 65 64 31 3a 61 36 3a  b0 3d 13 9a a7 e2 32 3a   Ged1:a6: .=....2:
0090  6c 61 36 3a c0 a8 00 16  a7 e2 31 3a 70 32 30 3a   la6:.... ..1:p20:
00a0  00 1a 35 29 37 26 4d e8  bf 05 e7 1d 36 fc 16 19   ..5)7&M. ....6...
00b0  30 e5 5a ef 65 65 35 3a  73 68 61 72 65 32 30 3a   0.Z.ee5: share20:
00c0  35 f7 62 99 9b 12 75 c0  f8 94 f3 d5 fb ac 70 59   5.b...u. ......pY
00d0  f7 67 83 ed 34 3a 74 69  6d 65 69 31 33 38 36 35   .g..4:ti mei13865
00e0  39 30 33 39 38 65 65                               90398ee
```

Fig. 5. BTSync tracker response packet.

BTSync keys

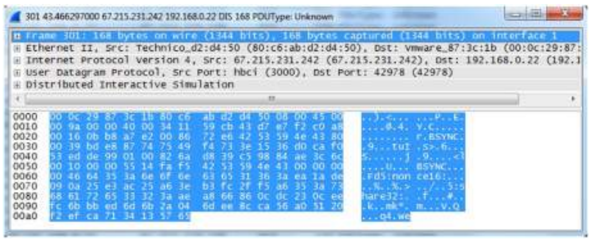
When a share is created by a seeder, a master key is generated. This is the “all access”, or read/write (RW), key that allows the owner of the share to add, remove or modify the contents of the share. The only scenario when this key should be distributed to another peer is when that peer is a trusted collaborator or when that peer is meant as a secondary source of content as opposed to a backup or repository. Read/write Keys can be recognised by the initial character ‘A’ at the beginning of the 33 character “secret” string. All keys are stored in plaintext in the bencoded block describing the corresponding share in the `sync.dat` file. From the master key, three other types of keys can be derived:

1. Read Only – The read-only (RO) key allows the receiving user to read the data being synchronised but not to modify or change the content on the source in any way. If, for some reason, a file in the share is modified or deleted on the local read-only host, its `invalidate` flag in the `<shareID>.db-wal` file is switched from a value of 0 to a value of 1. The result of this is that the file will no longer be synchronised from the source, even if the



```
263 41.877146000 192.168.0.22 67.215.229.106 DIS 150 PDUType: Unknown
⊞ Frame 263: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 1
⊞ Ethernet II, Src: Vmware_87:3c:1b (00:0c:29:87:3c:1b), Dst: Technico_d2:d4:50 (80:c6:ab:d2:
⊞ Internet Protocol Version 4, Src: 192.168.0.22 (192.168.0.22), Dst: 67.215.229.106 (67.215.
⊞ User Datagram Protocol, Src Port: 42978 (42978), Dst Port: hbci (3000)
⊞ Distributed Interactive Simulation

0010  00 88 0f c6 00 00 80 11  40 9f c0 a8 00 16 43 d7   ........ @.....C.
0020  e5 6a a7 e2 0b b8 00 74  c5 3d 42 53 59 4e 43 80   .j.....t .=BSYNC.
0030  00 43 49 b2 82 49 f9 9a  17 8e ad 52 7a 5e e1 44   .CI..I.. ...Rz^.D
0040  42 b3 36 e3 64 31 3a 6d  34 3a 70 69 6e 67 34 3a   B.6.d1:m 4:ping4:
0050  70 65 65 72 32 30 3a 00  1a 35 29 37 26 4d e8 bf   peer20:. .5)7&M..
0060  05 e7 1d 36 fc 16 19 30  e5 5a ef 35 3a 73 68 61   ...6...0 .Z.5:sha
0070  72 65 33 32 3a ae a8 66  86 0c dc 23 0c ee fc 6b   re32:..f ...#...k
0080  bb ed 6d 6b 2a 04 6d ee  8c ca 56 a0 51 20 f2 ef   ..mk*.m. ..V.Q ..
0090  ca 71 34 13 57 65                                  .q4.We
```

Fig. 6. BTSync relay request packet.



```
301 43.466297000 67.215.231.242 192.168.0.22 DIS 168 PDUType: Unknown
⊞ Frame 301: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface 1
⊞ Ethernet II, Src: Technico_d2:d4:50 (80:c6:ab:d2:d4:50), Dst: Vmware_87:3c:1b (00:0c:29:87:
⊞ Internet Protocol Version 4, Src: 67.215.231.242 (67.215.231.242), Dst: 192.168.0.22 (192.1
⊞ User Datagram Protocol, Src Port: hbci (3000), Dst Port: 42978 (42978)
⊞ Distributed Interactive Simulation

0000  00 0c 29 87 3c 1b 80 c6  ab d2 d4 50 08 00 45 00   ..).<... ...P..E.
0010  00 9a 00 00 40 00 34 11  59 cb 43 d7 e7 f2 c0 a8   ....@.4. Y.C.....
0020  00 16 0b b8 a7 e2 00 86  72 e6 42 53 59 4e 43 80   ........ r.BSYNC.
0030  00 39 bd e8 87 74 75 49  f4 73 3e 15 36 d0 ca f0   .9...tuI .s>.6...
0040  53 ed de 99 01 00 82 6a  d8 39 c5 98 84 ae 3c 6c   S......j .9....<l
0050  00 10 00 00 55 14 fa f5  42 53 59 4e 43 00 00 00   ....U... BSYNC...
0060  00 46 64 35 3a 6e 6f 6e  63 65 31 36 3a ea 1a de   .Fd5:non ce16:...
0070  09 0a 25 e3 ac 25 a6 3e  b3 fc 2f f5 a6 35 3a 73   ..%..%.> ../..5:s
0080  68 61 72 65 33 32 3a ae  a8 66 86 0c dc 23 0c ee   hare32:. .f...#..
0090  fc 6b bb ed 6d 6b 2a 04  6d ee 8c ca 56 a0 51 20   .k..mk*. m...V.Q 
00a0  f2 ef ca 71 34 13 57 65                            ...q4.We
```

Fig. 7. BTSync relay nonce exchange packet.

version on the source is updated or the local copy is deleted. RO keys are recognisable by the starting character 'B' prepended to the 32 character secret string. It should be noted that this was originally the character 'R' but it was changed with post alpha releases.
2. 24 Hour – The 24h key can be either a RO or RW key that has a time limit of 24h before it expires and cannot be used. The 24h time limit refers to the time during which the remote peer must use the key to gain access to the share. Once used successfully the peer will have continued access until the share is deleted or the source changes the authentication key. 24h keys start with the character 'C'. These key types are useful for security as they are only vulnerable to a third party interception for a limited period of time. The key stored in `sync.dat` is not the 24h key, it is the corresponding, non-expiring RW or RO equivalent.
3. Encrypted – There is an encrypted key that can be generated that creates an encrypted repository on the remote peer. The files synchronised are stored in their encrypted state remotely and cannot be read by the operator of the remote machine unless they are given the decryption key as well. This type of key is only possible to produce if the developer API has been installed and further analysis is outside the scope of this paper. Investigators should be aware that an encryption key is recognisable by the character 'D' at the start of the 33 character sequence.

In addition to the key strings, BTSync gives users the option of creating a RW or RO QR code that they can scan into a mobile application if preferred. Seeders must be very careful about what keys they distribute and to whom they are distributed. A RW key sent to the wrong person could subvert the assurance of file integrity and negate many of the benefits of BTSync over a shared folder hosted at a neutral location.

Sample keys taken from the same BTSync Share:
RW: `ACHY3VFJZ3RJ3DE2CHPUGE6W7EZSRA3OR`
RO: `BY6G6B7KIBGELLXE2RL65C34CAGPV7LUJ`
24h RW: `CBJIK32CLMWF2P7JLFYRGC3JRTEZ6JLPU`
24h RO: `CCYGZN6R67O67QB7HGLL4F5BAVA3AJ5LC`

Sources of interest to forensic investigation

To determine what can be found without resorting to specialist forensic utilities the BTSync application was installed and three folders were synchronised. The default settings were chosen at installation which include:

- BTSync runs at startup.
- BTSync service icon in the system tray (right click to hide).
- BTSync shortcut placed on the desktop of the All Users profile.
- BTSync added to the "All Users" profile quick launch.

In order to gather sample network data, three separate synchronisations were set up and monitored:

1. To `$Volume$\Documents and Settings\[User]\Desktop\sharedfolder` from a separate Linux laptop on the same LAN.
2. From `$Volume$\Documents and Settings\[User]\Desktop\sf2` on localhost to a separate Linux laptop on the same LAN.
3. Performed using a secret key posted on Reddit (Reddit, February 2014). The folder advertised itself as containing Gameboy ROMs with the read-only shared key of "RUAM2ED5ISKYR7LVELNVX56LLHQ47GBOZ". The application does not provide an indication as to what size the remote folder is or what files it contains before commencing the download.

As each folder was shared and assigned a secret key (either generated locally or copied from another source) a file was created in the folder: `$Volume$\Documents and Settings\[User]\Application Data\BitTorrent Sync\` with the ShareID of the folder created. This is the same share ID found in the file `.SyncID` created in the share folder itself.

The name of the db files created when the shared folder was added to BTSync consisted of the contents of the `.SyncID` file (35F762999B1275C0F894F3D5FBAC7059-F76783ED). This is the 20 byte share code that gets advertised to `t.usyncapp.com` when BTSync sends out its `get_peer` message, as can be seen in Fig. 4.

As each synchronisation was run, the BTSync logfile located at `$Volume$\Documents and Settings\[User]\Application Data\Bittorrent Sync\sync.log` is updated to record events. A sample of what this log filed contains is outlined in Table 4. The behaviour seen in the sync.log file corresponds with that observed in the captured network activity and the system activity recorded.

Table 5 presents the system activity logged during the synchronisation process. This was performed in a monitored session whereby a text file named "sample3.txt" was created on the source host (seeder) and then the read/write secret was shared to the prepared receiving folder on the repository (leecher). The synchronization process is shown from the point where apply was clicked on the repository. In the table `AppData` is shorthand for `~User\Application Data\Bittorrent Sync` and `Share` represents the path to the folder that has been allocated to receive the data. In this particular instance it is `C:\Documents and Settings\User\Desktop\sharedfolder`.

The shared folder is populated with the application control files and the 20 byte shareID is written to the

Table 4
Sample contents of BitSync log file.

```
[2013-12-01 12:41:33] Loading config file version 1.1.82
[2013-12-01 12:41:33] Loaded folder \\?\~User\BTSync
[2013-12-01 12:41:33] Loaded folder \\?\~User\Desktop\sharefolder
[2013-12-01 12:41:33] Loaded folder \\?\~User\Desktop\sf2
[2013-12-01 12:43:44] Got ping (broadcast: 1) from peer
   192.168.0.11:27900
   (00DC0AC2F0F91921AE29FC5E8F2273828BBAC747) for share
   35F762999B1275C0F894F3D5FBAC7059F76783ED
[2013-12-01 12:43:44] Found peer for folder
   \\?\~User\Desktop\sharefolder
   00DC0AC2F0F91921AE29FC5E8F2273828BBAC747
   192.168.0.11:27900 direct:1
[2013-12-01 12:43:45] Sending broadcast ping for share
   55045F90CA4C1A42DDB78DCD132F3ACC33E946EC
[2013-12-01 12:43:45] Requesting peers from server
[2013-12-01 12:43:45] Sending broadcast ping for share
   35F762999B1275C0F894F3D5FBAC7059F76783ED
```

`.SyncID` file.The database files are created in the User application data folder. The filenames used for these database files are the same as the ShareID stored in the `.SyncID` file. `.SyncIgnore` is created in the share folder and 822bytes are written to it. The data written are the explanation of the file's purpose and usage as well as a short list of files usually generated by an Operating System.

Next the synchronization process starts with the creation of `sync.dat.new` which will be renamed to `sync.dat` and eventually `sync.dat.old` as subsequent synchronisations take place. The `<ShareID>.db-wal` file is created to act as a holding area for data to be written to the SQLite database file of the same name. Next the data is received and written to a synchronisation delta file in preparation for merging into a fully synchronized text file. File data waiting merger can be identified by the extension `!sync` and `!sync(X)`.

The registry keys outlined in Table 6 were found after installation.

Next a file was deleted from the remote host and 10 min were given to ensure the local host had synchronised

Table 5
Example file I/O during the Client's synchronisation procedure.

Action	File	I/o	Path
Create	`.SyncID`	20B	Share
Create	`<ShareID>.db`		AppData
Create	`<ShareID>.db-journal`		AppData
Write	`<ShareID>.db-journal`	512B	AppData
Write	`<ShareID>.db`	3KB	AppData
Delete	`<ShareID>.db-journal`		AppData
Create	`.SyncIgnore`	822B	Share
Create	`sync.dat.new`	822B	AppData
Rename	`sync.dat` to `sync.dat.old`	450B	AppData
Rename	`sync.dat.new` to `sync.dat`	822B	AppData
Create	`<ShareID>.db-wal`		AppData
Create	`sample3.txt.!sinc`	33B	Share
Rename	`sample3.txt.!sinc` to `sample3.txt.!sinc.`	33B	Share
Write	`sample3.txt.sync.sync1`	33B	Share
Rename	`sample3.txt.sync.sync1` to:`sample3.txt`		Share

Table 6
Created BTSync registry keys during installation.

```
HKCR \Applications BTSync.exe \shell \open \command
HKCU \Software \Classes \Applications \BTSync.exe \shell \open
   \command
HKCU \Software Microsoft \Windows \CurrentVersion \Run
HKCU \Software \Microsoft Windows \ShellNoRoam \MUICache
HKLM \SOFTWARE \Microsoft \ESENT \Process \BTSync \DEBUG
   <−−if debug log enabled
HKLM SOFTWARE \Microsoft \Windows \CurrentVersion \Uninstall
   BitTorrent Sync
HKLM \SYSTEM \ControlSet001 Services \SharedAccess \Parameters
   \FirewallPolicy \StandardProfile \AuthorizedApplications \List
value: (C: \Program Files \BitTorrent Sync
   \BTSync.exe:*:Enabled:BitTorrent Sync)
HKU S-1-5-21…-1003 \Software Classes \Applications \BTSync.exe
HKU \S-1-5-21…-1003 \Software \Classes \Applications \BTSync.exe
   shell \open \command
HKU \S-1-5-21…-1003 \Software \Microsoft Windows \Current
   Version \Run
HKU \S-1-5-21…-1003 \Software \Microsoft \Windows \ShellNo
   Roam \MUICache
HKU \S-1-5-21…-1003_Classes \Applications \BTSync.exe \shell \
   open \command
C: \Program Files \BitTorrent Sync \BTSync.exe
C: \Documents and Settings \All Users \Desktop \BTSync.lnk
```

completely. While the file had been removed completely from the original host, on the local host it was instead moved from the main folder to a hidden subfolder (`.SyncArchive`) and not moved to the recycle bin. It is unknown at this time if there is any trigger or flag set that would result in this hidden file being deleted completely off the system. If not, then a remote host could theoretically constantly add and remove files to a synchronisation folder in order to deliberately occupy space on the local host with hidden files and so perform a form of low-tech denial of service attack by filling local storage.

BTSync does not come with any uninstaller of its own and must be removed from the Control panel. After uninstall the system was rebooted to ensure that the service had stopped running and any post uninstall clean-up had been performed, file locks cleared etc. A number of associated registry keys were still present, as outlined in Table 7.

In addition to this, all shared file folders used in synchronisations were still present as well as the default BTSync share created at install. The application folder was also still present in the `$Volume$\Documents and Settings\[User]\Application Data` folder but the sync.log file had been emptied.

As well as registry keys BTSync creates several other files that may be of interest to the forensic investigator. These files are located in the directory `$Volume$\Documents and Settings\[User]\Application Data\Bittorrent Sync\`. The contents of each file is outlined below:

- `<40 character share ID number>[.db, .db-shm, .db-wal]` – These files contribute to a SQLite3 database. The database describes the contents of the share directory corresponding to the share ID. It contains filenames, transfer piece registers and hash values for each individual file and its constituent pieces. While the .db file stores information on the schema of the database

Table 7
Registry keys Remaining after uninstallation.

HKCR \Applications BTSync.exe \shell \open \command
HKCU \Software \Classes \Applications \BTSync.exe \shell \open \command
HKCU \Software Microsoft \Windows \CurrentVersion \Run (“C: \Program Files \BitTorrent Sync \BTSync.exe”/MINIMIZED)
HKCU \Software Microsoft \Windows \ShellNoRoam \MUICache
HKLM \SOFTWARE \Microsoft ESENT \Process \BTSync \DEBUG (BTSync Rot 13 encoded=OGflap)
HKCU \Software \Microsoft Windows \CurrentVersion \Explorer \UserAssist \75048700-EF1F-11D0-9888-006097DEACF9 \Count Key=HRZR_EHACNGU:P: \Qbphzragf naq Frggvatf \BFv \Qrfxgbc \OGFlap.rkr
HKU \S-1-5-21…-1003 \Software \Microsoft Windows \Current Version \Explorer \UserAssist \75048700-EF1F-11D0-9888-006097DEACF9 \Count Key=HRZR_EHACNGU:P: \Qbphzragf naq Frggvatf \BFv \Qrfxgbc \OGFlap.rkr

the db-wal file contains bencoded entries for each file within the share in the format:

```
<Filename>:invalidated1:main.
hash:<20 byte hash>:mtime:
<timestamp of modification time>:npieces1:
owner20:<20 byte PeerID of the Seeder>:
path < path to file within share>
perm:420:size<bytes>:state1:timestamp:type1.
pvtime0:sig:<32 byte signature><filename>
```

- `settings.dat` – This is a bencoded file with a fileguard key (this key is a salted hash value ensuring that the file has not been edited by another tool besides the BTSync application itself). This file contains a log of settings for the application including the settings used to generate the Cryptographic keys and the registered URLs for peer searches.
- `sync.dat` – This is a bencoded file with a fileguard key. This file lists what files have been synchronised across the network. The directory paths and the shared secret used can be recovered from this file. This file is perhaps of most interest to the investigator due ot the large amount of timestamped and option recording it contains. Each share has an entry that is laid out in the following style:

```
path:<full path to share folder>:
secret:<33 character Key>:
pub_key:<32 byte ShareID used in Relay
messages>:
stopped_by_user[0|1]:
use_dht[0|1]:use_lan_broadcast[0|1]:
use_relay[0|1]:use_tracker[0|1]:
use_known_hosts[0|1]:
known_hosts:<contents of known hosts
option>:
peers:<list of peerIDs involved in sync>:
last_sync_completed<timestamp>:
invites<list of swarm invites received>:
folder_type0:
delete_to_trash[0|1]:
mutex_file_initialized[0|1]:
directTotal<IO direct to/from peer>:
relayTotal<IO total between peer and relay>
```

- `settings.dat.old` – This is the previous settings file. BTSync rotates through two settings generations deleting the old file when it is time to update with new data.

Recovering destroyed evidence

A number of the above artefacts prove that BTSync was installed on a client machine. It is possible that some or all of the incriminating files themselves may prove unrecoverable from the local hard disk due to anti-forensic measures. Should the secret be recovered for a given share, it is possible that a synchronisation of the suspect secret will enable the forensic investigator to recover this lost information from any other nodes still active in the share. Regular file system forensic analysis identifying synchronisation artefacts left behind from a particular share combined with this subsequent data synchronisation can prove that the suspect machine was involved in the sharing of incriminating material. Like any other digital investigation, the evidence gathered from the synchronisation process should be collected into a suitable digital evidence bag. Due to the value of BTSync metadata in the recovery of files stored remotely, a suitable P2P oriented evidence bag should be selected, such as that proposed by Scanlon and Kechadi (Scanlon and Kechadi, 2014). The after-the-fact recovery of data from remote machines is beyond the scope of this paper.

Conclusion

This paper gave a first look at a new use for a popular and widespread file synchronisation protocol. BTSync is not intended to replace BitTorrent as a file dissemination utility. However, it is still being used for this purpose. This is facilitated though websites publicly providing shared secrets, e.g., Reddit (Reddit, February 2014), as a form of dead-drop. The developers of the application describe it as an end-to-end encrypted method of transferring files without the use of a third party staging area. The reason for this is to try and ensure that the content and personal details remain hidden from unauthorised access. Initial analysis of the installation procedure identified files most likely to be of use to a forensic examiner confronted with a suspect live system or image running BTSync. However while the presence of a SyncID hidden folder can explain how a file was transferred to a system there is currently no way known outside of the application itself to determine the file’s origin or any further synchronisation points. From an investigative perspective, the decentralised nature of BTSync will always leave an avenue of gathering information identifying nodes sharing particular content (while providing many desirable redundancy and resilience against attack).

For the digital investigator working on a case involving BTSync, the description of the registry keys and files outlined can aid in identifying the content that may have been present on the local machine. Seeing as though BTSync merely requires any user wishing to join a particular synchronised folder to have the key, an investigator could also join the shared folder to download the data having recovered the corresponding files through hard drive analysis. Subsequent monitoring of the network communications using common tools, e.g., WireShark, tcpdump or libpcap, can aid in the identification of other nodes syncing the same content. In a number of investigative scenarios, this may focus the investigation in a beneficial direction resulting in the discovery of additional pertinent evidence or additional suspects.

Future work

From this initial analysis of the BTSync system, much further work needs to be done. The following list amounts to the list of areas for future investigation:

- Network Analysis – Identification of BTSync traffic and subsequent analysis to determine differentiation from standard BitTorrent traffic.
- Investigation Utility – A standalone application to extract relevant information from a suspect live or imaged machine running BTSync.
- Automated Share Detection – Identifying BTSync shares advertised by BTSync clients and detecting network activity to or from known locations.
- Crawling – To systematically follow connections to or from a share and identify new connections as they are discovered.
- Enumeration – Identifying individual shares and all active swarm members by the participating IP addresses and peer identifiers.
- Geolocation – Geolocating identified IP addresses may prove pertinent to recovering additional evidence regarding the suspect or may aid in the identification of others involved.
- API Analysis – Testing the provisioned API to determine what features can be leveraged to assist in forensic investigations.
- Recovery of Deleted Shares – In the scenario where a suspect has securely deleted any incriminating evidence from the local machine, the identification of trace information on the machine may result in the evidence being recoverable from other remote hosts. Due to BitTorrent's reliance on regular hashing for file distribution, the resultant hashes of remotely gathered files may be resolvable to the suspect's machine.

References

BitTorrent Inc. BitTorrent Sync User Manual [Online; accessed February 2014], http://www.bittorrent.com/help/manual/; 2013a.

BitTorrent Inc. BitTorrent Sync Developer API [Online; accessed February 2014], http://www.bittorrent.com/sync/developers/api; 2013b.

BitTorrent Inc. BitTorrent Sync Article [Online; accessed February 2014], http://blog.bittorrent.com/2013/12/05/bittorrent-sync-hits-2-million-user-mark/; 2013c.

Chung H, Park J, Lee S, Kang C. Digital forensic investigation of cloud storage services. Digit Investig 2012;9(2):81–95.

Cohen B. Incentives build robustness in bittorrent. In Proceedings of the Workshop on Economics of Peer-to-Peer systems, Vol. 6; 2003. pp. 68–72.

Cohen B. The BitTorrent Protocol Specification and Enhancement Proposals [Online; accessed February 2014], http://www.bittorrent.org/beps/bep_0000.html; 2014.

Drago I, Mellia M, Munafo MM, Sperotto A, Sadre R, Pras A. Inside dropbox: understanding personal cloud storage services. In: Proceedings of the 2012 ACM Conference on Internet Measurement Conference. IMC '12. New York, NY, USA: ACM; 2012, ISBN 978-1-4503-1705-4. pp. 481–94.

Karakaya M, Korpeoglu I, Ulusoy O. Free riding in peer-to-peer networks. Internet Comput IEEE 2009;13(2):92–8. http://dx.doi.org/10.1109/MIC.2009.33.

Quick D, Choo KKR. Forensic collection of cloud storage data: does the act of collection result in changes to the data or its metadata? Digital Investigation 2013a;10(3):266–77.

Quick D, Choo KKR. Google drive: forensic analysis of data remnants. Journal of Network and Computer Applications 2013b;40:179–93.

Quick D, Choo KKR. Digital Droplets: microsoft SkyDrive forensic data remnants. Future Generation Computer Systems 2013c;29(6):1378–94.

Reddit. BTSecrets [Online; accessed February 2014], http://www.reddit.com/r/btsecrets; 2013.

Scanlon M, Kechadi MT. Digital evidence bag selection for P2P network investigation. In: Future information technology. Springer; 2014. pp. 307–14.

Scanlon M, Hannaway A, Kechadi MT. A week in the life of the Most Popular BitTorrent Swarms. In: 5th Annual Symposium on Information Assurance (ASIA'10); 2010.

Stutzbach D, Rejaie R. Understanding churn in peer-to-peer networks. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement. IMC 06. New York, NY, USA: ACM; 2006, ISBN 1-59593-561-4. pp. 189–202. http://dx.doi.org/10.1145/1177080.1177105.

Wang H, Shea R, Wang F, Liu J. On the impact of virtualization on dropbox-like cloud file storage/synchronization services. In: Proceedings of the 2012 IEEE 20th international workshop on quality of service, vol. 11. IEEE Press; 2012. pp. 1–9.