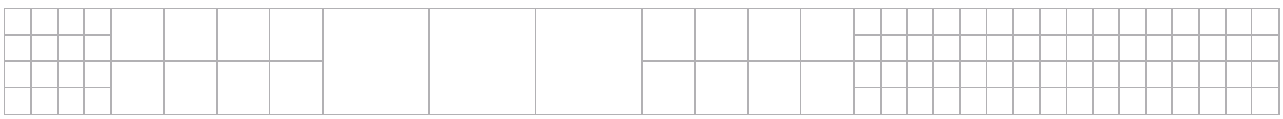
SFTP CONNECTIVITY

SSH Host Keys

Version 1.0

November 2017



Bloomberg

Bloomberg SFTP Connectivity

Bloomberg provides clients an SFTP service over Internet and dedicated leased-line networks to facilitate secure communication of sensitive data files. This document summarizes the security features of the SSH and SFTP protocols and provides guidance on how to most securely use the Bloomberg SFTP service.

SSH Overview

What is SSH?
The SSH (Secure Shell) protocol is a cryptographic network protocol which not only facilitates the establishment of a secure channel to allow network services to communicate and share data securely over an unsecure network, but also defines an extensible process for user authentication as well (typically via a password or key)

At a high level, the following steps take place when establishing an SSH connection.

1. An unsecure TCP connection is established to a destination server.
2. The server advertises the supported SSH protocol version and server host key to the client over the unsecure connection.
3. The client verifies the identity of the server, and continues if the protocol version is supported.
4. The client and server agree on an encryption algorithm, and encrypt the connection using the Diffie-Hellman exchange. At this point, the connection between the client and server is now secure.
5. The client then authenticates with the server using their username and credentials. The server verifies the client is a valid user.
6. At this point, the client has *logged in* successfully and can begin to interact with the server over the agreed protocol – for example, SFTP.

What is SFTP?
The SFTP (Secure File Transfer Protocol) is an extension of the SSH protocol version 2.0 allowing it to provide secure file access, transfer, and management over any reliable data stream.

By leveraging the SSH protocol, SFTP protects itself against password sniffing and man-in-the-middle attacks, ensuring the integrity of the data and authenticating both the client and the server.

While SFTP could potentially be applied on any secure channel other than SSH channels, it typically is only used in the context of an established SSH channel, and as such, requirements for connections SSH typically hold true for SFTP connections as well.

Host Keys and Fingerprints

What is a host key?
A host key actually refers to a pair of asymmetric cryptographic keys. While asymmetric keys are typically used for encryption and decryption, a host key is used by a SSH server to identify itself to a newly incoming SSH connection.

The public portion of the key pair is presented as part of establishing the SSH connection (Step 2 in the SSH Overview).

What is a host key fingerprint?
The exact format of an SSH host key depends on the cipher algorithm used, but the following is an example DSA host key:

```
ssh-dss
AAAAB3NzaC1kc3MAAACBALjShBKIEjqvuctPWVXxLQZiqI1XtP1xT
RuiGNO/hxYsoLFKdzhWVDfLK0FPptCY2UupvzWUcUN78AGpnvxDCO
utjq8BJvhfisKaxKIVlx9rLqRpFmfOttDeknIprymlN505QEtxynP
064j62Xn2g343/32YaZV0QdZuR2APUQsHvzRRmRnZ8B94sVtgRiiu
7hWrAAAAFQCq0wXRSBERaUCpUTQAAAIEAsQCkJH6EW8UO/Uk+JJRy
RhqigHVZyvCRocCxBqkjw8IjkovwNXaBxc3dLqafqm9lcYa5WgTA3
7CIuVyYcdGlRrBYQAmHSfpfda57wshMjMVfjwrzpyaKxb06KVUXPe
sIfRi++NqN2/d0pcqgwode1gK3SkC08AAACBAIrUk0NOTpvEZWZ/D
NI0IKxI8AUpwXzLf4QywlZplgPoiIaTNhxk5KUW8eZObXS8h1TrqO
oee6oB4P97I/5KOw/6Dm8a8n5LW0d2LhwIycZwyLCBXEkXrRZHP9x
XpOx9UZidmJNR6bYHFlrwoZudWrem51n+RnHI4CWjpITF1RWw6
comment@here
```

As SSH Keys are long, it is possible to generate a short representation of the keys by using a hashing algorithm. This short version of the key is known as the fingerprint and typically is formatted for ease of human readability. The following is an example of fingerprint for a 1024 bit key:

```
fa:a7:ca:68:2a:f9:2a:48:bc:99:2d:1a:f9:1e:1e:75
```

What is the purpose of a host key?
Imagine you are tasked with the delivery of an important package to an individual who supposedly is waiting at a specified address. After arriving at that address and ringing the door bell, somebody opens the door and awaits receipt of the package. However - how do

you know that this individual is the person who should be receiving this package? You would naturally ask to see some kind of proof of their identity.

When a network connection is established for the first time between two computers, they face the exact same dilemma. How does the client computer know that the server it is connecting to is indeed the intended destination? With this analogy in mind, the SSH host key is the server's identity.

Trusted third party

To continue the package delivery analogy in the previous section, suppose you asked the individual waiting at the door to present proof of their identity, what kind of documentation would you accept as valid? Most likely it would be something issued by the Government or other trusted institution. That institution is the *trusted third party* in this real-life example.

However, in the SSH Protocol, there is no such trusted third party used to verify the identity of the server, and when establishing a new connection, the client must know in advance what identification information will be accepted – in the case of SSH, this is the server's host key fingerprint.

As there must be an information exchange through some other media (e-mail, telephone) regarding usernames and credentials between a client-party and server-party as part of a setup, it is expected that the host key fingerprint information also be exchanged in advance as well.

Verification and Acceptance

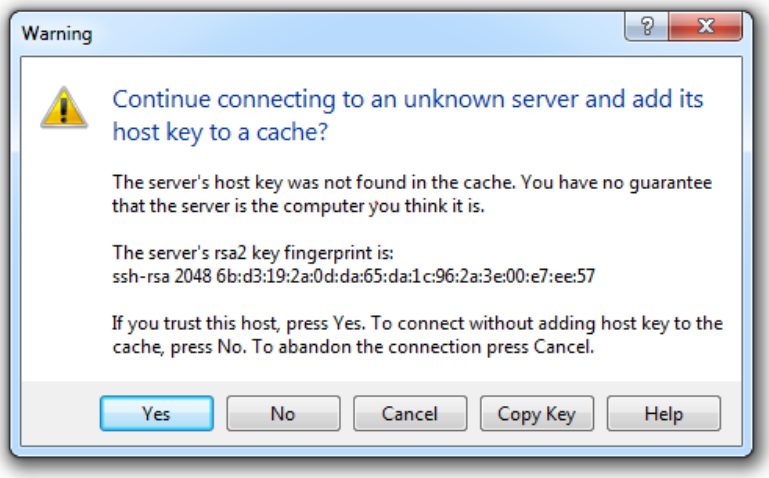
SSH host key verification

As mentioned in the previous section regarding the purpose of a host key, it is important that the SSH client verifies the host key of the server it is connecting to. To facilitate this, the SSH protocol ensure that whenever a client initiates a new connection, the server presents its host key's fingerprint for verification. If this host key has not been previously saved in the known hosts cache, the client software prompts the user to manually accept the new fingerprint before establishing the connection.

This manual prompt for new connections looks something like this in a terminal client:

```
The server's host key was not found in the cache. You
have no guarantee that the server is the computer you
think it is.

The server's rsa2 key fingerprint is:

ssh-rsa 2048
6b:d3:19:2a:0d:da:65:da:1c:96:2a:3e:00:e7:ee:57

If you trust this host, press Yes. To connect without
adding host key to the cache, press No. To abandon
the connection press Cancel.
```

In a Windows SFTP Client (such as SCP), it will look like this:



As the client already knows the fingerprint of the server to expect (as this was previously exchanged during account setup), at this point the client can confirm if the presented fingerprint matches the expected value.

If not, then it is possible that the connection is currently being subjected to a man-in-the-middle attack and the connection should not be established.

SSH known hosts cache

Once a new server host key fingerprint is accepted as part of establishing a connection for the first time, a typical SSH client would save this in a local cache called *known hosts*. As the host key and fingerprint has now been saved, it becomes trusted and all subsequent connections to that server will not require the client to manually accept the fingerprint again.

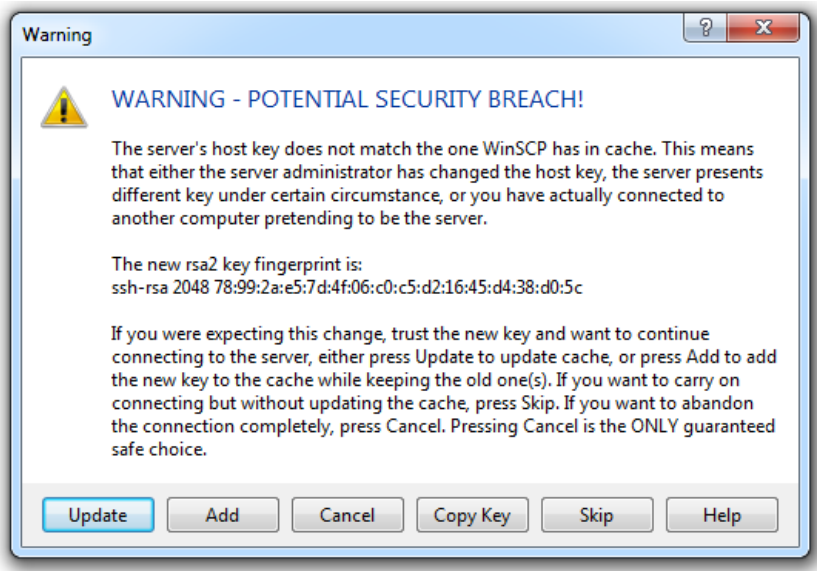
However, the SSH client software still verifies the host key on every subsequent connection established to ensure it is always connecting to the right server. Should the host key fingerprint presented by the server ever differ from the one that is saved in the client's known hosts cache, this indicates that the connection may be being subjected to a man-in-the-middle attack and must be manually accepted.

This manual prompt to accept a change in a host key looks like this in a terminal client:

```
The host key sent by the server is different from the
host key stored in the host key database for myserver
(192.168.0.1), port 22. This may mean that a hostile
party has "hijacked" your connection and you are not
connected to the server you specified.

It is recommended you verify your host key before
accepting.

Server's host key fingerprint (MD5 hash):
14:09:26:bc:13:24:31:5c:f7:6c:39:94:f7:4d:52:14

If you trust this host, enter “y” to add the key to
the host key database and connect. If you do not
trust this host, enter “n” to abandon the connection.

Accept and save? (y/n)
```

Bloomberg SFTP Connectivity – Host Keys

In a Windows SFTP Client (such as SCP), it will look like this:



It is important at this stage that the client-party verifies with the server-party system administrator via another medium (e-mail, telephone) if this change is expected, and if the new fingerprint matches the expected value.

Client or Vendor Hosted SFTP Services

To ensure the security and protection of data provided by Bloomberg, it is the Bloomberg policy that the host key fingerprint of all SFTP connections be verified. Therefore, please be advised that when requesting new data feed from Bloomberg to a client or vendor hosted SFTP service, the fingerprint information of that host SFTP service must be provided.

Please work with the system administrators accordingly to gather this information to facilitate a smooth on-boarding process.

Bloomberg's Hosted SFTP Services

Bloomberg hosts multiple SFTP server environments on both Internet and Leased Line networks. Bloomberg ensures that each of these environments differ by their host keys. The sections below detail the host key fingerprints in various hashing formats for each Bloomberg hosted SFTP environment.

We recommend for the protection of your data that clients always perform their due diligence when connecting to Bloomberg Hosted SFTP services and always verify the fingerprint of the destination server against one of the official values below before establishing the connection.

Americas Internet (General Use)

Format	Size	Fingerprint
RSA1	2048	1c:7a:ff:ce:c0:72:19:8d:77:21:aa:2a:98:0e:28:51
RSA	2048	6b:d3:19:2a:0d:da:65:da:1c:96:2a:3e:00:e7:ee:57
DSA	1024	23:07:0b:b8:4f:6a:e5:10:f2:96:9a:9e:b7:c7:eb:c4
ECDSA	256	N/A

Bloomberg SFTP Connectivity – Host Keys

Americas Internet (Beta)

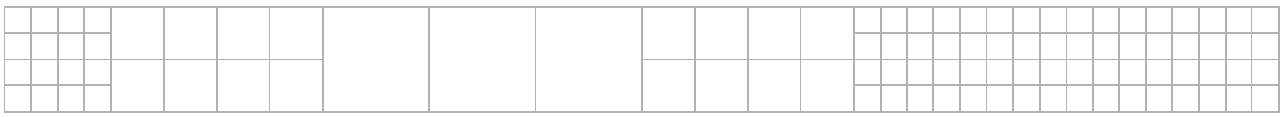
Format	Size	Fingerprint
RSA1	2048	N/A
RSA	2048	ae:42:32:41:26:5e:f2:95:e6:32:c2:64:6b:a6:75:0a
DSA	1024	c7:17:e8:22:2d:d3:d2:60:b4:10:ab:34:32:f7:4d:9c
ECDSA	256	N/A

Americas BVAULT

Format	Size	Fingerprint
RSA1	2048	1f:84:ea:08:1a:c3:92:30:6e:98:a7:5b:04:c2:69:5e
RSA	2048	df:8c:ea:8f:3b:10:ff:ad:f6:30:63:13:83:ea:16:8c
DSA	1024	32:e5:bd:40:54:12:59:c0:59:72:c0:63:f4:45:66:cb
ECDSA	256	N/A

Americas Private/Leased Line (General Use)

Format	Size	Fingerprint
RSA1	2048	27:44:9d:87:88:5a:5c:13:50:f2:2a:58:f8:60:e1:e3
RSA	2048	24:95:29:5c:50:73:49:f5:ba:cb:57:02:8c:78:36:54
DSA	1024	20:81:f1:21:02:54:75:95:de:3d:a3:ab:3a:71:07:57
ECDSA	256	N/A



Bloomberg

Bloomberg SFTP Connectivity – Host Keys

Frankfurt Internet (General Use)

Format	Size	Fingerprint
RSA1	2048	3f:23:17:54:14:b3:77:7b:d7:17:a0:e9:bd:5a:43:c9
RSA	2048	3b:a3:a6:05:fe:c3:28:e7:23:49:83:c8:46:89:cf:e8
DSA	1024	13:3b:15:b5:8f:22:c8:11:27:ee:22:7e:b3:ab:6f:a3
ECDSA	256	ac:90:31:ab:cd:0d:e8:f1:75:38:cb:10:81:24:94:fd

London Internet

Format	Size	Fingerprint
RSA1	2048	N/A
RSA	2048	b2:19:4e:21:17:b6:09:bd:c8:74:87:ad:b3:31:34:a9
DSA	1024	89:fd:7f:90:a6:79:f2:30:cc:1d:75:d9:b3:b3:e2:94
ECDSA	256	N/A

Secondary Private/Leased Line (GMT)

Format	Size	Fingerprint
RSA1	2048	N/A
RSA	2048	5c:c4:ce:7b:24:74:21:57:6e:e0:6e:1d:78:27:66:c9
DSA	1024	5b:63:41:b9:4e:83:38:ca:d1:3c:dc:16:b9:dc:01:63
ECDSA	256	N/A

Bloomberg