PROPOSED

SECURED TRANSMISSION OF FILES

AT

XPRESSI BRANCH ISSUANCE

Proposed By:
Hengbao International Pte Ltd
(189353-K)

Date: 1.10.2020
Version R1A

For more information, please contact:

Hengbao International Pte Ltd
300 Beach Road
Unit 34-06 The Concourse
Singapore 199555

Contact Persons:
The primary contact for all inquiries regarding this proposal shall be directed to:

David Tang
Business Development Director
Mobile: +65 9645 9649
E-mail: davidtang@hengbao.com

Table of Content

1 INTRODUCTION

1.1 Background

Data security breach is probably one of the highest concerns for most financial institutions. This is all the more of greater significance when involving transmission of sensitive client data across two countries where different fiduciary regulations are applied.

To protect against such data breaches, there are many tools and software that assist in such matters, nevertheless there is also a need understand the operations needs of the bank. Therefore it is pertinent to have both domain knowledge of the the subject matter in order to balance between both security needs and functionality needs.

Our proposal, along with our professional consulting services provides the finest personalization solution in the industry. We would like to thank Bred Bank for the opportunity to submit this proposal as a step towards a long and mutually beneficial business relationship.

1.2 Scope of the Proposal

The present requirement is to ensure that there is a secured mode of transmission of data files between servers residing in different locations within the Bank.

In this proposal, we will propose to make use of industry standards such as SSH to make file transfers, and using third-party commercial software to enable the job process. We will describe the proposed SSH FTP software to be used for embossing file transfer between CMS server to Xpressi server, as well as the use of established third-party software to providing such a service.

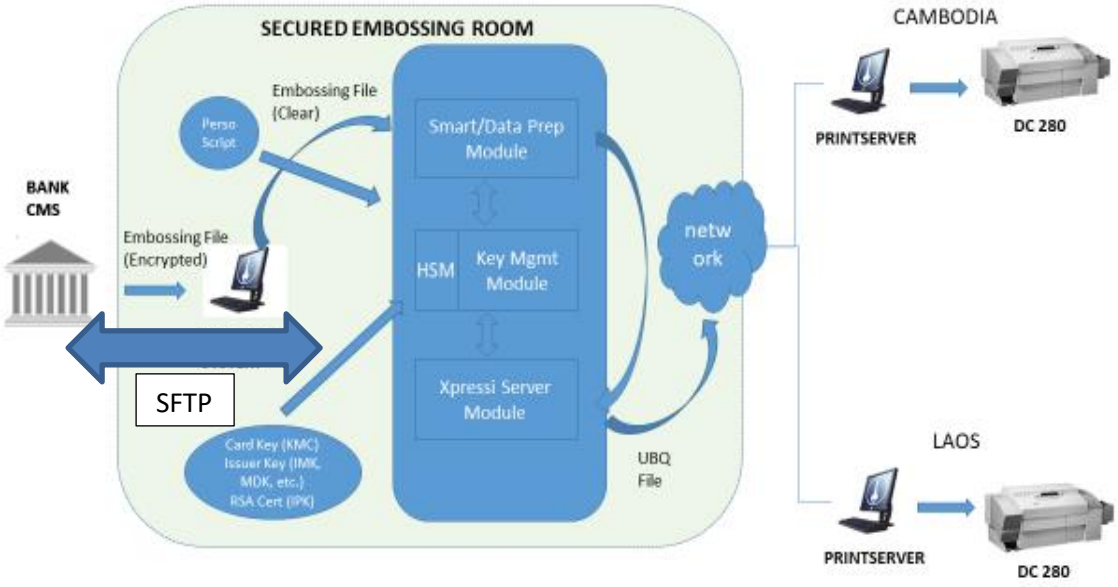
The proposal includes:

a) Basic about SSH
b) About Bitvise SSH FTP Software
c) Proposed Configuration for Bred Bank File Transfer
d) Commercial Price

1.3 File Transfer Requirement

As shown in the Card Personalization System diagram below, the SSH FTP is required to securely transfer the embossing data from Bank CMS Server to Xpressi Card Personalizaion Server.

In this case, we assume that the embossing file is pushed from the CMS server to Xpressi Server.


SECURED EMBOSSING ROOM
Embossing File
(Clear)
Perso
Script
Smart/Data Prep
Module
BANK
CMS
Embossing File
(Encrypted)
HSM
Key Mgmt
Module
netw
ork
SFTP
Xpressi Server
Module
Card Key (KMC)
Issuer Key (IMK,
MDK, etc.)
RSA Cert (IPK)
UBQ
File
CAMBODIA
PRINTSERVER
DC 280
LAOS
PRINTSERVER
DC 280

2 ABOUT SSH

2.1 What is SSH?

The Secure Shell protocol version 2, or SSH2, specifies how a client can connect securely to an SSH server, and then use the resulting secure link to access the server's resources. Among other things, the client can run programs; transfer files; and forward other TCP/IP connections over the secure link.

The SSH2 protocol is a descendant of the SSH v1.x series of protocols. SSH version 2 is standardized at IETF, and the vast majority SSH implementations now support SSH version 2. SSH version 1 is less secure, and is almost no longer being used.

2.2 How Secure is SSH?

The Secure Shell protocol provides the services of server authentication; encryption; data integrity verification; and client authentication. Server authentication is performed using RSA, DSA, or ECDSA public key algorithms. For encryption and data integrity verification, a number of algorithms are provided which every SSH product can implement in a modular fashion. Client authentication can be performed using a password, a public key, single sign-on Kerberos, and other methods.

The SSH2 protocol specification is publicly available and has been reviewed by several independent implementers. When properly used, the protocol is understood to be secure against all known cryptographic attacks, passive as well as active.

2.3 Client vs Server

In internet protocols, the terms *client* and *server* have specific technical meanings:

- A **client**, when speaking of a program, is a program that initiates connections and requests to other computers.
- A **server** is a program that waits to receive connections and handles requests.

A *client* may run all of the time or some of the time, depending on when a user needs it. A client may more likely run on a desktop computer, but can also run on a server computer if this computer needs to send out request or initiate connections.

A *server* usually runs all the time, in the background. A server may more likely run on a computer in a data center. However, it is also perfectly feasible to use server programs on a desktop computer.

If you are looking for SSH software, you are looking for a **server** if you want to set up a computer to receive connections from other people and their computers. You are looking for a **client** if you wish to connect, using SSH, to someone else's computer.

2.4 SSH compared to SSL/TLS

SSH and TLS/SSL are different protocols used for similar purposes. Both protocols are used to authenticate communicating parties and secure data during transport.

SSL/TLS tend to use X.509 certificates, is based on ASN.1 encodings, and is most commonly used to as a security layer for HTTP, SMTP, and FTP traffic.

The SSH protocol tends to use public keys without a certificate infrastructure, is based on a simpler binary encoding, and tends to be used as a security layer for SFTP and SCP file transfers, terminal shell access, and forwarding of connections for other applications.

SSH can be perceived as a less clunky version of TLS. Due to its deliberate independence from X.509 certificates, SSH lends itself well to connections between entities with an existing trust relationship, where TLS does poorly. TLS lends itself better to connections between strangers.

2.5 SSH Features

SSH is a highly flexible protocol, and many different types of services can use it. The protocol's open architecture allows these services to run at the same time without impeding one another.

An SSH client and server can transfer files using the protocols **SCP** and **SFTP**, which run on top of an established SSH session. While SCP is the old Unix rcp utility transplanted onto a different transport, SFTP is a flexible remote file access protocol that can be used in advanced ways. SFTP is better standardized and widely supported, so often software that provides an SCP-like interface really uses SFTP instead.

Note that SFTP is unrelated to FTP, or to FTP over TLS/SSL. The protocols are independent and very different.

A frequently used service is the **remote console**. This involves allocating a channel within the SSH session, which is then used as transport for a terminal protocol such as vt100 or xterm. The client displays to the user a console window within which the user can execute command line programs on the server.

SSH also provides **exec requests**. An exec request executes a program on the server like a remote console, but without expectation of interactive input. Exec requests are useful for automated remote administration.

Another popular SSH function is **port forwarding**, or TCP/IP connection tunneling. With SSH port forwarding, it is possible to secure a TCP/IP connection established by an independent application that would otherwise be vulnerable to network attacks.

3 ABOUT BITVISE SSH FILE TRANSFER SOFTWARE

3.1 The roles of Bitvise SSH Client and SSH Server

Bitvise SSH Server is used to accept connections from SSH clients. The server is intended to run for a prolonged period of time, and will provide SSH clients that connect with access configured by the server administrator. The SSH server might be configured to provide access to a terminal console, port forwarding, or file transfer to and from the server using SFTP, SCP, or FTPS.

Bitvise SSH Client is used to initiate connections to SSH servers. It is usually used interactively, so it will only run when a user runs it, but it can also be launched unattended to run scripted commands or file transfers, or to maintain an SSH connection for port forwarding. The SSH client is used to access a terminal console on an SSH server, to initiate port forwarding, or to initiate file transfers to and from SSH servers using SFTP.

Both products are connectivity products. They cannot be used standalone. For an SSH server to be useful, you need clients that will connect to it. For an SSH client to be useful, you need an SSH server to connect to.

The two products can be installed on the same machine, but there is no benefit in connecting an SSH client to an SSH server running on the same machine, except for testing.

3.2 Bitvise SSH Server: Secure File Transfer, Terminal Shell and Tunneling

Bitvise SSH Server provides secure remote access to Windows servers and workstations. Security is our SSH server's key feature: in contrast with Telnet and FTP servers, Bitvise SSH Server encrypts data during transmission. Thus, no one can sniff your password or see what files you are transferring when you access your computer over SSH.

Bitvise SSH Server is ideal for **remote administration** of Windows servers; for **secure file transfer** by organizations using SFTP and SCP; for **advanced users** who wish to access their home machine from work, or their work machine from home; and for a wide spectrum of advanced tasks, such as securing other applications using SSH **TCP/IP tunneling**.

You are looking for an SSH server if you want to set up a computer to **receive connections** from other people and their computers. If you want to initiate connections or file transfers, you are looking for an **SSH client**.

3.3 Bitvise SSH Client: Graphical and Command-line File Transfer, Terminal and Tunneling

Bitvise SSH Client for Windows includes state of the art **terminal emulation**, graphical as well as command-line **SFTP support**, an **FTP-to-SFTP bridge**, powerful **tunneling** features including dynamic port forwarding through integrated proxy, and remote administration for our SSH Server.

You are looking for an SSH client if you wish to **initiate connections** or file transfers to someone else's computer. If you are looking to receive connections, you are looking for an **SSH server**.

3.4 Security in Bitvise

Bitvise SSH Server and Client have an excellent security track record. Since our software was first released in 2001, we have had the following significant issues:

- A denial-of-service vulnerability in WinSSHD 1.1.
- A potential SFTP privilege escalation in WinSSHD versions up to 4.19.
- A denial-of-service vulnerability in Bitvise SSH Server versions 5.50 through 5.58. The likelihood of arbitrary code execution is not known, but is not excluded.
- A denial-of-service vulnerability in Bitvise SSH Server versions 5.xx and 6.xx up to and including version 6.43. The likelihood of arbitrary code execution is not known, but is not excluded.

- A denial-of-service vulnerability on 32-bit Windows in Bitvise SSH Server versions 5.xx and 6.xx up to and including version 6.49, and versions 7.xx up to and including version 7.39. We believe this issue does *not* allow for arbitrary code execution.

All issues were fixed promptly, as soon as they came to our attention.

Vulnerabilities discovered in other SSH implementations have not applied to ours, as our products are developed independently and share no code base with OpenSSH and others. Our SSH protocol implementation is also known as one of the more stringent ones, on several occasions exposing flaws in other implementations that other products did not detect.

When a security vulnerability is discovered in one of our products, it will be fixed promptly and an upgrade version fixing the flaw will be made available for download. When this happens, customers who have purchased licenses will be notified at the technical contact email address associated with their licenses.

3.5 Specifications

Our software implements SSH version 2, SFTP versions 3, 4, and 6, SCP and FTPS according to publicly available standards. In addition, our software supports minor extensions documented here.

3.5.1 SSH

Bitvise SSH Server, SSH Client and our library FlowSshC/Cpp/Net use Bitvise's SSH protocol implementation, **FlowSsh**. Our software implements the following standards:

- RFC 4250: The Secure Shell (SSH) Protocol Assigned Numbers
- RFC 4251: The Secure Shell (SSH) Protocol Architecture
- RFC 4252: The Secure Shell (SSH) Authentication Protocol
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol
- RFC 4254: The Secure Shell (SSH) Connection Protocol
- RFC 4256: Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)
- RFC 4419: Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol
- RFC 4462: Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol
- RFC 4716: The Secure Shell (SSH) Public Key File Format
- RFC 4819: Secure Shell Public Key Subsystem
- RFC 5656: Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer
- RFC 6668: SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol
- RFC 8308: Extension Negotiation in the Secure Shell (SSH) Protocol
- RFC 8332: Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol

In addition, our software implements:

- draft-ietf-curdle-ssh-kex-sha2-10: Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH)
- draft-ietf-curdle-ssh-curves-08: Secure Shell (SSH) Key Exchange Method using Curve25519 and Curve448
  - curve25519-sha256@libssh.org: SSH key exchange using Curve25519
- OpenSSH "PROTOCOL" document: *aes256-gcm@openssh.com* and *aes128-gcm@openssh.com*; Host key update and rotation
- draft-ssh-ext-auth-info-01: Extended authentication information in Secure Shell (SSH)
- draft-ssh-global-requests-ok-00: Sending and Handling of Global Requests in Secure Shell (SSH)
- draft-miller-secsh-compression-delayed-00: delayed compression à la *zlib@openssh.com*. Contains an inherent race condition; the mechanism in RFC 8308 is newer, but preferred.

3.5.2 SFTP

SFTP is the native SSH file access protocol. SFTP version 3 can be thought of as an RPC (remote procedure call) mechanism for a generic Unix-based filesystem. Later SFTP versions, 4-6, include additional features and accommodations for non-Unix filesystems.

When the SSH protocol was being standardized, SFTP was being standardized with it. A schism developed when Unix-based developers rejected further development of SFTP beyond version 3, while others insisted on better support for non-Unix filesystems. As a result, no version of SFTP could be standardized by consensus.

Bitvise SSH Server, SSH Client and FlowSshC/Cpp/Net implement the following drafts, which de-facto specify SFTP:

- draft-ietf-secsh-filexfer-02: SFTP version 3
- draft-ietf-secsh-filexfer-04: SFTP version 4
- draft-ietf-secsh-filexfer-13: SFTP version 6
- draft-ietf-secsh-filexfer-extensions-00: SFTP extensions
- OpenSSH "PROTOCOL" document: SFTP v3 extensions and OpenSSH idiosyncrasies

The *check-file-blocks* extension:

When using SFTP version 6, Bitvise SSH Server also advertises the *check-file-blocks* extension. This is because with versions 7.35, we found server implementations that support the *check-file* extensions, but either did not support block-by-block hashing (VShell) or would disconnect if a larger file was hashed block-by-block (*mod_sftp* for ProFTPD). This prevented the functioning of file content synchronization in Bitvise SSH Client and FlowSsh.

We suggest that SFTP servers advertise support for *check-file-blocks* if **all** of the following are true:

- The server **supports block-by-block** file hashing.
- **Any reasonable block size** requested by the client is supported.
- A file can be hashed block-by-block starting from an **arbitrary offset**.

3.5.3 SCP

SCP is an informal protocol originally based on the Unix utility *rcp*. Bitvise SSH Server implements SCP in a manner intended to be compatible with all significant SCP clients we are aware of.

Some SCP clients, notably WinSCP, require an SCP server to provide a Unix-like shell. Bitvise SSH Server implements **BvShell** – a dedicated *Shell access type* setting which provides a hybrid Unix/Windows-style shell. This shell is part of the SSH Server and respects file access limits configured for the user in the SSH Server's virtual filesystem. If enabled for a user, BvShell supports WinSCP in SCP mode, as well as other SCP clients which expect not only SCP, but also a Unix-style shell.

Bitvise SSH Client and FlowSshC/Cpp/Net do not implement SCP - only SFTP over SSH.

3.5.4 FTPS

Historically, the FTP protocol is not based on a secure design, but it is possible to implement it securely. Bitvise SSH Server implements FTP over TLS/SSL in a manner that emphasizes security at the expense of compatibility with less secure clients. For more information, please see Bitvise SSH Server: Compatibility with FTPS Clients.

Bitvise SSH Client and FlowSshC/Cpp/Net do not implement FTPS - only SFTP over SSH.

3.5.5 TOTP

Bitvise SSH Server implements **time-based one-time passwords** as specified in RFC 6238 – *TOTP: Time-Based One-Time Password Algorithm*.

The SSH Server Control Panel can export a TOTP secret key as a 2D code image by encoding the secret key URI as specified in the Google Key Uri Format. The URI is encoded as an image as specified in ISO/IEC 18004 – *QR Code bar code symbology specification*.

3.5.6 SOCKS

Bitvise SSH Client and FlowSshC/Cpp/Net implement the following as part of (A) outgoing SSH connections through a SOCKS or HTTP CONNECT proxy, and (B) dynamic TCP connection forwarding:

- socks4.protocol: SOCKS 4
- socks4a.protocol: SOCKS 4A
- RFC 1928: SOCKS Protocol Version 5
- RFC 2817: *Upgrading to TLS Within HTTP/1.1* – the HTTP CONNECT verb

3.6 bvterm

The purpose of the **bvterm** terminal protocol is to provide a quality of Windows terminal experience that cannot always be achieved with other terminal protocols such as *xterm*.

On Windows, character based applications receive keyboard input and render their output in a Windows Console. bvterm 2.xx is based on the Windows Console API. It may be loosely viewed as a Windows Console API remote procedure call protocol.

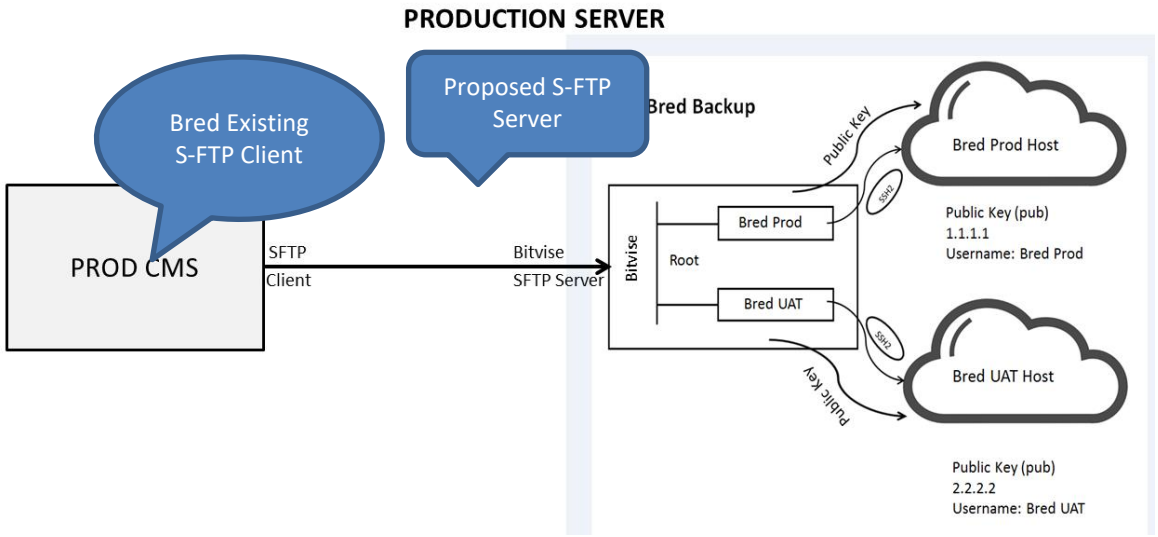
The current bvterm version is 2.0. This version is supported by:

- Bitvise SSH Server versions 5.xx and newer.
- Bitvise SSH Client versions 4.25 and newer.

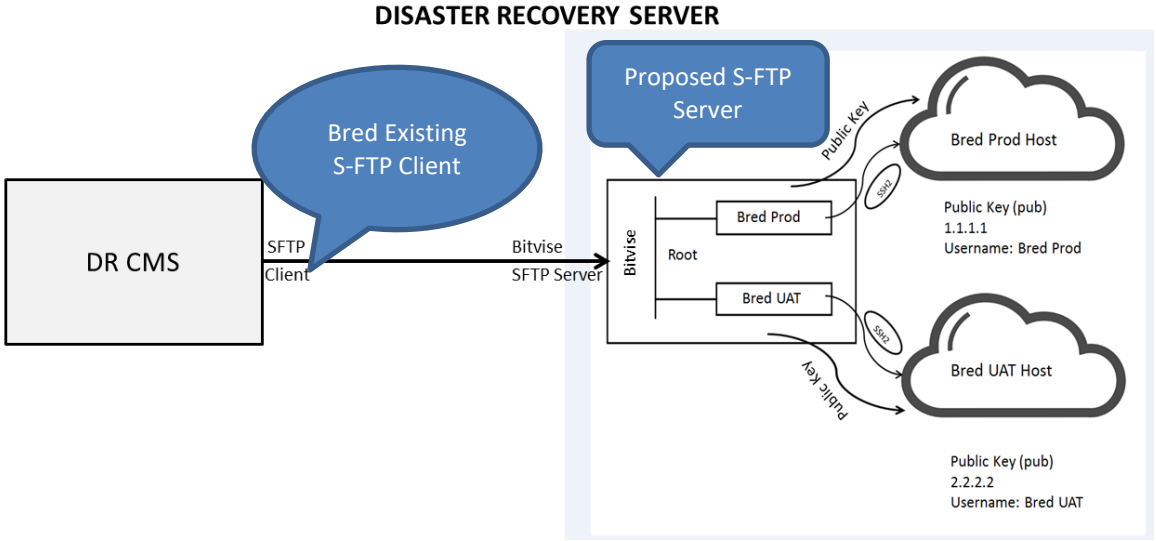
4 PROPOSED SSH FTP FOR BRED BANK

4.1 File Transfer Diagram

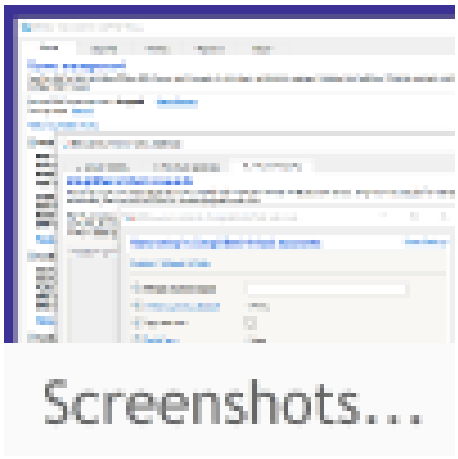
4.1.1 S-FTP for Production System



4.1.2 S-FTP for Disaster Recovery System



4.2 Bitvise SSH Server: Secure file transfer and terminal shell access for Windows



Bitvise SSH server supports **all desktop and server versions of Windows**, 32-bit and 64-bit, from Windows XP SP3 and Windows Server 2003, up to the most recent – Windows 10 and Windows Server 2019.

Bitvise SSH Server includes the following:

- **SFTP server**: Secure file transfer using SFTP - compatible with a wide variety of clients
- **SCP server**: Secure file transfer using SCP - compatible with command line and graphical clients
- **FTPS server**: Secure file transfer using FTP over TLS/SSL - compatible with secure FTPS clients
- **SSH server**: Secure remote access via console - vt100, xterm and bvterm are supported
- Secure remote access via **GUI** - Remote Desktop or WinVNC required
- Secure, effortless **Git integration**
- Secure TCP/IP connection **tunneling** (port forwarding)

4.3 Professional SSH server

We continue to invest considerable effort to create the best SSH software we can. These are some of the features that make Bitvise SSH Server special:

- **Ease of use**: Bitvise SSH Server is designed for Windows, so that it is easy to install and configure. In a regular Windows environment, it will work immediately upon installation **with no configuring**. (We do however recommend tightening down settings to restrict access only to those accounts and features that you use.)
- **Encryption and security**: Provides state-of-the-art encryption and security measures suitable as part of a standards-compliant solution meeting the requirements of **PCI**, **HIPAA**, or FIPS 140-2 validation.
- **FTPS support**: Can handle file transfer connections using FTP over TLS (SSL) in addition to SFTP and SCP over SSH. See compatible clients.
- **Unlimited connections**: Bitvise SSH Server imposes no limits on the number of users who can connect, and gets no more expensive for a larger number of connections. The number of simultaneous connections is limited only by system resources!
- **Two-factor authentication**: Connections using SSH, SFTP and SCP clients can require an additional time-based one-time password. Compatible with RFC 6238 authenticator apps, including Microsoft Authenticator, Google Authenticator, LastPass, Authy, WinAuth, or FreeOTP.
- **Windows groups**: Bitvise SSH Server natively supports configurability through Windows groups. No need to define account settings for each Windows account individually. The SSH server knows what groups a user is in and, if configured, will use appropriate Windows group settings. **Virtual filesystem mount points** can be inherited from multiple groups.
- **Quotas and statistics**: The SSH Server can be configured with per-user and per-group quotas and bandwidth limits, and keeps a record of daily, monthly, and annual usage statistics.
- **Speed**: SFTP transfer speed mostly depends on the client, but Bitvise SSH Server allows clients to obtain some of the fastest transfer speeds available. With Bitvise SSH Client, SFTP file transfer

speeds in the tens or hundreds of MB/s can be obtained. **SFTP v6** optimizations, including *copy-file* and *check-file* for remote file hashing and checksums, are supported.

- **Virtual filesystem:** Users connecting with file transfer clients can be restricted to a single directory, or several directories in a complex layout. Users connecting with terminal shell clients can also be restricted in the same way if their *Shell access type* is set to **BvShell**.
- **Large files**: The SSH Server supports files of any size that are supported by the filesystem you configure to store files and the client software you are using to connect. Windows filesystems have these maximum file sizes.
- **Git integration**: Set an account's shell access type to *Git access only*, and configure the path to your Git binaries and repositories. The account can now securely access Git, without being given unnecessary access to the system.
- **Obfuscated SSH** with an optional keyword. When supported and enabled in both the client and server, obfuscation makes it more difficult for an observer to detect that the protocol being used is SSH. (Protocol; OpenSSH patches)
- **Single sign-on**: Bitvise SSH Server supports GSSAPI-enabled Kerberos 5 key exchange, as well as NTLM and Kerberos 5 user authentication. This means that, using Bitvise SSH Client or another compatible GSSAPI-enabled client, **any user in the same Windows domain**, or a trusted one, can log into the SSH server without having to verify the server's host key fingerprint, and without even having to supply a password! Using Windows group-based settings, the user's account doesn't even have to be configured in the SSH server.
- **Virtual accounts**: want to set up an SFTP server with many users, but don't want to create and manage 1000 Windows accounts? No problem. Bitvise SSH Server supports virtual accounts, created in SSH server settings, backed by the identity of one or more Windows accounts. SSH server settings for these accounts are also configurable on a virtual group basis.
- **Bandwidth limits**: Separate upload and download speed limits can be configured for each user and group.
- **Excellent terminal support**: Bitvise SSH Server provides the best terminal support available on the Windows platform. Our terminal subsystem employs sophisticated techniques to render output accurately like no other Windows SSH server. When used with Bitvise SSH Client, our bvterm protocol supports the full spectrum of a Windows console's features: colors, Unicode characters, and large scrollable buffers.
- **BvShell:** Users whose filesystem access should be restricted to specific directories can have their *Shell access type* configured to *BvShell*. Similar to *chroot*, this provides access to a limited terminal shell which can allow for more powerful access than a file transfer client, but still restricts the user to root directories configured for them.
- **Telnet forwarding**: The SSH Server can be configured to forward terminal sessions to a legacy Telnet server, providing SSH security to existing Telnet applications.
- **Flexibility**: most SSH server features can be configured individually on a per-account basis from the user-friendly Bitvise SSH Server Control Panel. Using Bitvise SSH Client, the SSH server's Control Panel can be accessed and configured through the same user-friendly interface from **any remote location**.
- **Server-side forwarding**: with Bitvise SSH Server and Client, a server and multiple clients can be set up so that all port forwarding rules are configured centrally at the server, without requiring any client-side setting updates. The SSH clients only need to be configured once, and port forwarding rules can easily be changed when necessary.
- **Scriptable settings**: Using the supplied BssCfg utility, or using PowerShell, all settings can be configured from a text file, **from a script**, or interactively from the command-line.
- **Multi-instance support**: Bitvise SSH Server supports multiple simultaneous, independent installations on the same computer for customers needing completely separate instances for different groups of users. Multiple SSH server versions can run concurrently, as separate instances on the same server.
- **Master/slave configuration**: In environments with multiple SSH server installations, one can be configured to run as master, and others can be configured to run as slaves. Slave installations can be configured to synchronize their settings, host keys, and/or password cache with the master. This feature can be used both for **cluster support**, and to reproduce aspects of SSH server settings on a large number of similar installations.
- **Delegated administration**: Users of the SSH Server who do not have full administrative rights can be granted limited access to SSH Server settings, where they can add or edit virtual accounts using the

remote administration interface in Bitvise SSH Client. Limited administration tasks can be delegated without requiring full administrative access.

4.4 Windows version compatibility

Bitvise SSH Server supports the following Windows versions:

- Windows Server 2019
- Windows Server 2016
- Windows 10
- Windows Server 2012 R2
- Windows Server 2012
- Windows 8.1
- Windows Server 2008 R2
- Windows Server 2008
- Windows Vista SP1 or SP2
- Windows Server 2003 R2
- Windows Server 2003
- Windows XP SP3

A recent Bitvise SSH Server version should be used on all platforms. The SSH Server is network-facing, security-sensitive software. Using a recent version is the only way to receive updates. Therefore, we do not recommend indefinite use of older versions.

4.5 Encryption and security features

4.5.1 SSH, SFTP and SCP:

- Key exchange algorithms:
  - Curve25519
  - ECDH over elliptic curves secp256k1, nistp256, nistp384, nistp521 using SHA-512, SHA-384, or SHA-256
  - Diffie Hellman with group exchange using SHA-256 or SHA-1
  - Diffie Hellman with fixed 4096, 3072, 2048, or 1024-bit group parameters using SHA-512, SHA-256, or SHA-1
  - GSSAPI key exchange using Diffie Hellman and Kerberos authentication
- Signature algorithms:
  - Ed25519
  - ECDSA over elliptic curves secp256k1, nistp256, nistp384, nistp521 using SHA-512, SHA-384, or SHA-256
  - RSA using 4096, 3072, 2048, 1024-bit key sizes with SHA-512, SHA-256, or SHA-1
  - DSA using SHA-1 (legacy)
- Encryption algorithms:
  - AES with 256, 128-bit keys in GCM mode
  - AES with 256, 192, 128-bit keys in CTR mode
  - AES with 256, 192, 128-bit keys in CBC mode (legacy)
  - 3DES in CTR or CBC mode (legacy)
- Data integrity protection:
  - AES with 256, 128-bit keys in GCM mode
  - HMAC using SHA-256, SHA-1
- Server authentication:
  - Client verifies server identity using server host key fingerprint or public key
  - Automatic synchronization of new host keys to client supported
- Client authentication:
  - Password authentication with Windows accounts - local or Active Directory
  - Password authentication with virtual accounts - configurable password policy
  - Public key authentication

- Kerberos single sign-on using GSSAPI
- Two-factor authentication with a time-based one-time password

4.5.2 FTP over TLS (SSL):

- TLS security:
  - Available TLS versions and cipher suites depend on the installed version of Windows
  - TLS versions 1.0, 1.1 and 1.2 can be enabled individually in *Advanced settings*
  - ECDHE, RSA and DHE cipher suite families can be enabled individually
- Authentication:
  - Can use self-signed or CA-signed server certificate
  - Password authentication with Windows accounts - local or Active Directory
  - Password authentication with virtual accounts - configurable password policy
- Requires secure clients:
  - Only secure FTPS is supported - plaintext FTP connections are not accepted
  - FTPS clients must support explicit TLS using the *AUTH TLS* command
  - FTPS clients must support passive mode and use the TLS resume feature for data connections

4.5.3 Additional security features:

- Denial of service protection through throttling of incoming connections
- Login attempt delay for concurrent logins for same user or from same IP address
- Automatic temporary IP address blocking with IP whitelist
- Username blacklist
- Configurable client IP address, product version string restrictions
- Account-specific IP address restrictions
- IP-based access rules configurable by country

4.6 FIPS 140-2 validation

When FIPS is enabled in Windows, our software uses Windows built-in cryptography, validated by NIST to FIPS 140-2 under certificates #2937, #2606, #2357, and #1892. On Windows XP and 2003, our software uses the Crypto++ 5.3.0 FIPS DLL, originally validated by NIST under certificate #819 (historical). When FIPS mode is not enabled, additional non-FIPS algorithms are supported.

4.7 Cryptographic implementations and availability

Current Bitvise software versions (8.36 and higher) use the following cryptographic implementations for different algorithms, on different versions of Windows:

Algorithm	Windows XP, Server 2003	Windows Vista to 8.1, Server 2008 to 2012 R2	Windows 10, Server 2016 to 2019
Signature			
RSA	Crypto++ 5.3	Windows CNG	Windows CNG

Ed25519	n/a	DJB	DJB
ECDSA (NIST curves)	Crypto++ 5.3	Windows CNG	Windows CNG
ECDSA/secp256k1	Crypto++ 5.3	OpenSSL	Windows CNG
1024-bit DSA	Crypto++ 5.3	Windows CNG	Windows CNG
Non-standard DSA	Crypto++ 5.3	Crypto++ 5.6	Crypto++ 5.6
Key exchange			
Classic DH	Crypto++ 5.3	Windows CNG	Windows CNG
Curve25519	n/a	DJB	DJB
ECDH (NIST curves)	Crypto++ 5.3	Windows CNG	Windows CNG
ECDH/secp256k1	Crypto++ 5.3	OpenSSL	Windows CNG
Encryption			
AES	Crypto++ 5.3	Windows CNG	Windows CNG
3DES	Crypto++ 5.3	Windows CNG	Windows CNG

Integrity			
GCM	n/a	Windows CNG	Windows CNG
HMAC-SHA2	Crypto++ 5.3	Windows CNG	Windows CNG
HMAC-SHA1	Crypto++ 5.3	Windows CNG	Windows CNG

5 PRICE

5.1 Professional Services For Applying SSH System (Includes 3rd Party SSH Service License)

Item	Description	Qty	Unit Price (USD)	Total Price (USD)
1	**Installation of Bitvise Server Software for the followings:** • **Main Xpressi Server** • **Back-up Xpressi Server**	2	3,500.00	7,000.00
2	**Configuration of Bitvise SSH Software for the followings:** • **BRED Production Folder** • **BRED UAT Folder**	2		
3	**NOTE:** **Standard Use Bitvise Server License Per Installation:** • Bitvise SSH Server is licensed per machine; a fee needs to be paid for each machine on which the SSH server runs. • The right to use the SSH server on an individual machine is portable; if you remove Bitvise SSH Server from a machine on which you had the right to use it, you can install it and use it on a different machine. • A purchased right to use Bitvise SSH Server is permanent and, unless extended, applies to all Bitvise SSH Server versions published within 12 months of your purchase.			

5.1 Online Support and Upgrades

Item	Description	Qty	Unit Price (USD)	Total Price (USD)
1	**Online Support and upgrades:** • Upgrades and technical support for purchased Bitvise SSH Server installations are free for up to 12 months after your purchase. • Subsequent support and upgrades are available annually at 20% of the current price for the installations you have purchased.	2	750.00/year	750.00/year

5.2 On-Site Support and Other Maintenances

Item	Description	Qty	Unit Price (USD)	Total Price (USD)
1	On-site Support for Bitvise SSH Software	1	1200.00/trip	1200.00/trip

6 TERMS & CONDITIONS

Price	• All prices quoted INCLUDE freight • All prices quoted EXCLUDE any sales or government tax. • All prices quoted EXCLUDE travelling and accommodation cost
Standard Reimbursement Charges for Professional Services and Maintenance Support	• Hotel Accommodation • Flight tickets between sites i.e. Kuala Lumpur & Cambodia, Thailand, Laos • Local transportation and airport transfer • Out of pocket expenses
Out of Pocket Expenses (Per Diem)	• Per Diem: US50.00 per day • Accommodation, Flights, Airport Transfer: Based on actual claim
Payment	• 50% upon issuance of PO • Remaining to paid prior to delivery
Validity	• This proposal is valid up within 3 months from the date of issue.

-End of proposal-