

**BitWise Encryption Overview**  
**by Kevin Hock, BitWise Client Programmer**  
**Document Revision 1.2, October 6th, 2005**

**Overview**

Data transmitted over the Internet passes through many servers and/or routers and there are many opportunities for third parties to intercept that transmission. Preventing interception is impossible; instead, the data must be made unreadable (encrypted) during transmission, with a way for the intended recipient to be able to transform the received transmission back to its readable form (unencrypted). BitWise automatically encrypts all data exchanged (including file transfers) between BitWise users to ensure confidentiality. The purpose of this document is to explain how encryption works in simple, easy to understand terms.

**Brief Introduction to Encryption**

When a message is *encrypted*, that means that it is transformed into a form that is not readable; the encrypted form often looks like random characters or gibberish. When a message is *decrypted*, it is returned to its original readable form. Encryption techniques fall into two categories: symmetric key and key pairs. BitWise uses both types of encryption to provide easy, automatic encryption and decryption.

When using a symmetric key, the same key is used to both encrypt and decrypt a message. This is analogous to one person locking something in a room, and then sending copies of the key to anyone that they want to have access to the room. Just like a lock in the real world can be picked, a symmetric encryption key can also be "picked," but it takes a lot of computational power (just like picking a lock takes a lot of patience and skill). Likewise, just as in the real world, sending a key to another person is sometimes difficult (more on this later).

Key pairs use two *different* keys: one key is called the public key, and the other key is called the private key. Key pairs are based on prime numbers, and because of the mathematics on which key pairs are based, only the private key can decrypt a message encrypted with the public key, and only the public key can decrypt a message encrypted with the private key. The public key is made available to everyone, but the private key is kept secret.

Imagine a door with two locks, one keyed to the public key, and the other keyed to the private key. Both locks are unlocked, and a pile of keys that fit the public key lock are available on a table nearby. When anyone wants to leave something in the room, they take a public key and lock the door. Upon locking the door with the public key, the private key lock is also locked automatically. Anyone using the public key cannot enter the room and read the message because the private key lock is still locked. Only the person with the private key can open the door and look at what is in the room.

The intended recipient of the message then comes to pick up the message, with possession of the private key (the *only* person with the private key). The private key can be used to unlock both locks, and access to the room (and its contents) is granted. The important point here is that only someone with the private key (which should be closely guarded) can open the door and have access to the contents of the room. It is possible to "pick the lock," but again, doing so requires a large amount of time, skill and determination, just like in real life.

## How BitWise uses Encryption

BitWise uses both symmetric keys and key pairs to provide seamless, easy and automatic encryption of all BitWise messages and file transfers. For a variety of technical and theoretical reasons, it is much faster and more efficient to encrypt long messages with a symmetric key, and so the heart of the encryption used in BitWise is based on symmetric keys. As mentioned above, sending that symmetric key without it being intercepted or copied is not always easy. BitWise uses a key pair to safely transmit the symmetric key.

Each time a user logs in to BitWise (it is worth noting that the BitWise server and client have a key pair used to encrypt passwords for transmission to the server when logging in), the BitWise client generates a random RSA key pair. RSA is an encryption standard which was selected for its prevalence of use and proven security. By generating a new public and private key pair for each login, anyone attempting to crack the keys will be hopelessly behind, because by the time they have cracked one key pair, the user will most certainly have a new one (or multiple new ones). It's like trying to hit a moving target.

Let's say that user A (the initiator) initiates a connection to User B (the listener). The initiator sends its public key to the listener. The listener then generates a random Blowfish symmetric key (Blowfish is also a common and time-proven encryption method), and encrypts the randomly generated Blowfish key using the initiator's public key. This message can only be decrypted by the private key, which is known only to the initiator.

The listener sends the encrypted Blowfish key back to the initiator, who decrypts the message, giving the initiator a copy of the symmetric key generated by the listener. Only the initiator can decrypt the message because only the initiator has the private key. Anyone intercepting the encrypted Blowfish key would be unable to learn the symmetric key selected because they do not possess the private key needed to decrypt the message.

At this point, both computers have the same symmetric key, so that a message encrypted by one computer can be decrypted by the other. Because the symmetric key was exchanged using the key pair, only the two computers involved know the symmetric key, and only those two computers can decrypt messages sent between the two computers. To anyone reading the data between the computers, only meaningless gibberish would appear. Further, by having a new symmetric key for each connection (a connection would generally last the duration of one conversation or one file transfer), anyone trying to "pick the lock" is faced again with a moving target; the key is constantly changing with each new connection.

## Using your own RSA Keys (BitWise Plus and Professional)

There are two down sides to using random RSA keys (as described above). First, using random keys does not provide identity management. One of the benefits of using key pair cryptography is that you can generate a public and private key pair that represent *you*. Only you can decrypt messages encrypted with your public key, since only you have the matching private key. As long as you keep your private key safe, your public key definitively represents you.

Second, using random keys does not protect against a "man in the middle" attack. Basically, a third party, in between the individuals trying to communicate, manipulates the network to receive the public key sent by the initiator to the recipient. The man in the middle then sends a different public key to send to the listener. The listener uses the public key sent by the man in the middle to encrypt a message, which the man in the middle can decrypt with his

private key. The man in the middle then re-encrypts the message using the original public key sent by the initiator and sends on the re-encrypted message. Neither the initiator or the recipient knows that the man in the middle has intercepted the encrypted message. Having a defined, unchanging public key, as described in the previous paragraph, thwarts these attacks.

BitWise Plus and Professional users can generate or load their own RSA keys using the Encryption preferences panel. This defined public key will be used in place of the randomly generated RSA key, allowing others to track and compare your public key over time. If your public key is not received, or is different than the public key stored from previous conversations, your contacts will be alerted. This way, your contacts can take steps to verify that it really is you on the other end, and not an imposter.

## **Encryption Strength**

The "strength" of the encryption used is dependent on both the encryption method itself and the length of the key(s). Key length works well with the lock analogy: a lock requiring a key a foot long would be much harder to pick than a lock requiring only a one inch key. BitWise Personal uses 512-bit RSA keys and 128-bit Blowfish keys. BitWise Plus uses 1024-bit RSA keys and 256-bit Blowfish keys. Professional users have the option of using RSA encryption up to 2048-bit and Blowfish encryption up to 448-bit (256-bit outside of the USA and Canada).

Even if you perceive 128-bit keys to be "weak" (whether or not a key is "weak" often depends on your personal preference and level of security desired), days and days would still be needed to break any messages sent using BitWise. In almost all the cases, having encryption is like having locks on your house: burglars go looking for easier targets, and someone looking for a good time reading IMs would go elsewhere.

One last factor that is important in determining the "strength" of an encryption key is the quality of the random numbers used to generate. Good, truly random numbers are needed for good encryption, yet random number generation is one of the most difficult processes for even modern day computers. BitWise uses a random number generator whose algorithm is based on the PGP randPool implementation, which has been shown to generate sufficiently random numbers for strong encryption.

## **Verifying Encrypted Data**

Since BitWise is not open source software, we are sometimes asked how to verify that the data transmitted via BitWise is, in fact, encrypted. Such verification is easy to do by running a *packet sniffer* on yourself (a variety of tools are available for Windows, Linux or Mac OS X, some of them for free). A packet sniffer intercepts network traffic as it passes by and through your ethernet card, and while packet sniffers can be used maliciously, it is perfectly acceptable to packet sniff yourself. If you log in to BitWise and packet sniff your own network data on port 4137, you can watch the data being sent by BitWise. Type some text to a friend and send it; you will see what looks like "garbage" being sent across the network. This is the same garbage that someone else would see if they were maliciously sniffing your network data. If you were using a non-encrypted messenger, you would see your conversation in the packet sniffer in plain, readable text, easy for any snoop to read.

## **Cryptography Implementation**

For the more technically inclined, the cryptographic library used in BitWise is the Crypto++ library, a free, open-source cryptographic library that has been recently been validated by NIST and CSE for FIPS 140-2 level 1 conformance (which essentially means that it has been verified to be properly implemented and secure for use on a PC). Any discussions of Blowfish or RSA keys, or random number generation, directly refer to the respective specific implementations in the Crypto++ library. The Crypto++ library is considered in high regard by developers worldwide, and has our full confidence. BitWise is using Crypto++ 5.2.1, the newest release at the time of this writing.