Basic Dropbear build instructions: - First, edit options.h to choose user-defined features to choose, such as which ciphers/hashes you want, which forwarding you want, etc. - Edit debug.h if you want any debug options - Now configure Dropbear's host-specific options (if you are using a cvs copy, "autoconf; autoheader" first) ./configure (optionally with --disable-zlib or --disable-syslog, or --help for other options) - Then compile and optionally install Dropbear: (the Makefile requires GNU make, if you want to make it portable, send me some patches) make make install (installs to /usr/local/sbin, /usr/local/bin by default) You need to generate server keys, this is one-off: ./dropbearkey -t rsa -f dropbear_rsa_host_key ./dropbearkey -t dss -f dropbear_dss_host_key or alternatively convert OpenSSH keys to Dropbear: ./dropbearconvert openssh dropbear /etc/ssh/ssh_host_dsa_key dropbear_dss_host_key And you can now run the server. ./dropbear or './dropbear -h' to get options. If the server is run as non-root, you most likely won't be able to allocate a pty, and you cannot login as any user other than that running the daemon (obviously). Shadow passwords will also be unusable as non-root. The Dropbear distribution includes a standalone version of OpenSSH's scp program. You can compile it with "make scp", you may want to change the path of the ssh binary, specified near the top of the scp.c file. By default the progress meter isn't compiled in to save space, you can enable it with "make scp-progress".

==========

==== Compiling with uClibc: Firstly, make sure you have at least uclibc 0.9.17, as getusershell() in prior versions is broken. Also note that you may get strange issues if your uClibc headers don't match the library you are running with, ie the headers might say that shadow password support exists, but the libraries don't have it. To compile for uClibc the following should work: rm config.cache CC=i386-uclib-gcc ./configure --disable-zlib make clean make make strip ... and that should be it. You can use "make static" to make statically linked binaries, and it is advisable to strip the binaries too. If you're looking to make a small binary, you should remove unneeded ciphers and MD5, by editing options.h It is possible to compile zlib in, by copying zlib.h and zconf.h into a subdirectory (ie zlibincludes), and export CFLAGS="-Izlibincludes -I../zlibincludes" export LDFLAGS=/usr/lib/libz.a before ./configure and make. If you disable zlib, you must explicitly disable compression for the client - OpenSSH is possibly buggy in this regard, it seems you need to disable it globally in ~/.ssh/config, not just in the host entry in that file. You may want to manually disable lastlog recording when using uClibc, configure with --disable-lastlog. One common problem is pty allocation. There are a number of types of pty allocation which can be used -- if they work properly, the end result is the same for each type. Running configure should detect the best type to use automatically, however for some embedded systems, this may be incorrect. Some things to note: If your system expects /dev/pts to be mounted (this is a uClibc option), make sure that it is. Make sure that your libc headers match the library version you are using. If openpty() is being used (HAVE_OPENPTY defined in config.h) and it fails, you can try compiling with --disable-openpty. You will probably then need to create all the /dev/pty?? and /dev/tty?? devices, which can be problematic for devfs. In general, openpty() is the best way to allocate PTYs, so it's best to try and get it working.

==========
==== Public key auth: You can use ~/.ssh/authorized_keys in the same way as with OpenSSH, just put the key entries in that file. They should be of the form: ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAwVa6M6cGVmUcLl2cFzkxEoJd06Ub4bVDsYrWvXhvUV+ZAM9uGuewZBDoAqNKJxoIn0Hyd0Nk/yU99UVv6NWV/5YSHtnf35LKds56j7cuzoQpFIdjNwdxAN0PCET/MG8qyskG/2IE2DPNIaJ3Wy+Ws4IZEgdJgPITYUBWWtCWOGc= someone@hostname You must make sure that ~/.ssh, and the key file, are only writable by the user. NOTE: Dropbear ignores authorized_keys options such as those described in the OpenSSH sshd manpage, and will not allow a login for these keys.