Contributing

============

We gratefully accept bug reports and contributions from the community. There are some requirements we need to fulfill in order to be able to integrate contributions:

- As with any open source project, contributions will be reviewed by the project team and community and may need some modifications to be accepted.

- The contribution should not break API or ABI, unless there is a real justification for that. If there is an API change, the contribution, if accepted, will be merged only when there will be a major release.

Coding Standards

----------------

- We would ask that contributions conform to [our coding standards](https://tls.mbed.org/kb/development/mbedtls-coding-standards), and that contributions are fully tested before submission, as mentioned in the [Tests](#tests) and [Continuous Integration](#continuous-integration-tests) sections.

- The code should be written in a clean and readable style.

- The code should be written in a portable generic way, that will benefit the whole community, and not only your own needs.

- The code should be secure, and will be reviewed from a security point of view as well.

Making a Contribution

---------------------

1. [Check for open issues](https://github.com/ARMmbed/mbedtls/issues) or [start a discussion](https://lists.trustedfirmware.org/mailman/listinfo/mbed-tls) around a feature idea or a bug.

1. Fork the [Mbed TLS repository on GitHub](https://github.com/ARMmbed/mbedtls) to start making your changes. As a general rule, you should use the ["development" branch](https://github.com/ARMmbed/mbedtls/tree/development) as a basis.

1. Write a test which shows that the bug was fixed or that the feature works as expected.

1. Send a pull request (PR) and work with us until it gets merged and published. Contributions may need some modifications, so a few rounds of review and fixing may be necessary. We will include your name in the ChangeLog :)

1. For quick merging, the contribution should be short, and concentrated on a single feature or topic. The larger the contribution is, the longer it would take to review it and merge it.

1. All new files should include the [Apache-2.0](https://spdx.org/licenses/Apache-2.0.html) standard license header where possible.

1. Ensure that each commit has at least one `Signed-off-by:` line from the committer. If anyone else contributes to the commit, they should also add their own `Signed-off-by:` line. By adding this line, contributor(s) certify that the contribution is made under the terms of the [Developer Certificate of Origin](dco.txt). The contribution licensing is described in the [License section of the README](README.md#License).

API/ABI Compatibility

---------------------

The project aims to minimise the impact on users upgrading to newer versions of the library and it should not be necessary for a user to make any changes to their own code to work with a newer version of the library. Unless the user has made an active decision to use newer features, a newer generation of the library or a change has been necessary due to a security issue or other significant software defect, no modifications to their own code should be necessary. To achieve this, API compatibility is maintained between different versions of Mbed TLS on the main development branch and in LTS (Long Term Support) branches.

To minimise such disruption to users, where a change to the interface is required, all changes to the ABI or API, even on the main development branch where new features are added, need to be justifiable by either being a significant enhancement, new feature or bug fix which is best resolved by an interface change.

Where changes to an existing interface are necessary, functions in the public interface which need to be changed, are marked as 'deprecated'. This is done with the preprocessor symbols `MBEDTLS\_DEPRECATED\_WARNING` and `MBEDTLS\_DEPRECATED\_REMOVED`. Then, a new function with a new name but similar if not identical behaviour to the original function containing the necessary changes should be created alongside the existing deprecated function.

When a build is made with the deprecation preprocessor symbols defined, a compiler warning will be generated to warn a user that the function will be removed at some point in the future, notifying users that they should change from the older deprecated function to the newer function at their own convenience.

Therefore, no changes are permitted to the definition of functions in the public interface which will change the API. Instead the interface can only be changed by its extension. As described above, if a function needs to be changed, a new function needs to be created alongside it, with a new name, and whatever change is necessary, such as a new parameter or the addition of a return value.

Periodically, the library will remove deprecated functions from the library which will be a breaking change in the API, but such changes will be made only in a planned, structured way that gives sufficient notice to users of the library.

Long Term Support Branches

--------------------------

Mbed TLS maintains several LTS (Long Term Support) branches, which are maintained continuously for a given period. The LTS branches are provided to allow users of the library to have a maintained, stable version of the library which contains only security fixes and fixes for other defects, without encountering additional features or API extensions which may introduce issues or change the code size or RAM usage, which can be significant considerations on some platforms. To allow users to take advantage of the LTS branches, these branches maintain backwards compatibility for both the public API and ABI.

When backporting to these branches please observe the following rules:

1. Any change to the library which changes the API or ABI cannot be backported.

1. All bug fixes that correct a defect that is also present in an LTS branch must be backported to that LTS branch. If a bug fix introduces a change to the API such as a new function, the fix should be reworked to avoid the API change. API changes without very strong justification are unlikely to be accepted.

1. If a contribution is a new feature or enhancement, no backporting is required. Exceptions to this may be additional test cases or quality improvements such as changes to build or test scripts.

It would be highly appreciated if contributions are backported to LTS branches in addition to the [development branch](https://github.com/ARMmbed/mbedtls/tree/development) by contributors.

Currently maintained LTS branches are:

1. [mbedtls-2.7](https://github.com/ARMmbed/mbedtls/tree/mbedtls-2.7)

1. [mbedtls-2.16](https://github.com/ARMmbed/mbedtls/tree/mbedtls-2.16)

Tests

-----

As mentioned, tests that show the correctness of the feature or bug fix should be added to the pull request, if no such tests exist.

Mbed TLS includes a comprehensive set of test suites in the `tests/` directory that are dynamically generated to produce the actual test source files (e.g. `test\_suite\_mpi.c`). These files are generated from a `function file` (e.g. `suites/test\_suite\_mpi.function`) and a `data file` (e.g. `suites/test\_suite\_mpi.data`). The function file contains the test functions. The data file contains the test cases, specified as parameters that will be passed to the test function.

[A Knowledge Base article describing how to add additional tests is available on the Mbed TLS website](https://tls.mbed.org/kb/development/test\_suites).

A test script `tests/scripts/basic-build-test.sh` is available to show test coverage of the library. New code contributions should provide a similar level of code coverage to that which already exists for the library.

Sample applications, if needed, should be modified as well.

Continuous Integration Tests

----------------------------

Once a PR has been made, the Continuous Integration (CI) tests are triggered and run. You should follow the result of the CI tests, and fix failures.

It is advised to enable the [githooks scripts](https://github.com/ARMmbed/mbedtls/tree/development/tests/git-scripts) prior to pushing your changes, for catching some of the issues as early as possible.

Documentation

-------------

Mbed TLS is well documented, but if you think documentation is needed, speak out!

1. All interfaces should be documented through Doxygen. New APIs should introduce Doxygen documentation.

1. Complex parts in the code should include comments.

1. If needed, a Readme file is advised.

1. If a [Knowledge Base (KB)](https://tls.mbed.org/kb) article should be added, write this as a comment in the PR description.

1. A [ChangeLog](https://github.com/ARMmbed/mbedtls/blob/development/ChangeLog.d/00README.md) entry should be added for this contribution.