

# CryptoAuthLib - Atmel CryptoAuthentication Library

## Pre-release notes

9/11/2015 Library renamed from atcang to cryptoauthlib.

9/10/2015

This library is in pre-release form. As such, it is expected to incur changes that are substantial including but not limited to: - function name changes - function additions or deletions - bug fixes - changes to test configurations and test suites - changes, additions, deletions to documentation - hardware abstraction layer support additions or deletions

The implication is: while this is a good reference library, expect it to change before final release. Any code you base upon this library will certainly need to change to work with the first production release.

## Introduction

This code base implements an object-oriented C library which supports Atmel CryptoAuth devices. The family of devices supported currently are:

- [ATECCx08A \(ATECC108A, ATECC508A\)](#)
- [ATSHA204A](#)

Prerequisite skills: - strong C programming and code reading - Atmel Studio familiarity - Knowledge of flashing microcontrollers with new code - Familiarity with Atmel CryptoAuth device functionality

Prerequisite hardware: - [ATSAMR21 Xplained Pro](#) or - [ATSAMD21 Xplained Pro](#)

- [CryptoAuth Xplained Pro Extension](#) or
- [socketed top-board for ATCK101](#) to accept chip packages of your choice

For most development, using socketed top-boards is preferable until your configuration is well tested, then you can commit it to a CryptoAuth Xplained Pro Extension, for example. Keep in mind that once you lock a device, it will not be changeable.

## CryptoAuthLib Architecture

See the 'docs' directory of CryptoAuthLib for supporting documentation including architecture diagrams and more detailed usage docs.

The library is structured to support portability to: - multiple hardware/microcontroller platforms - multiple environments including bare-metal, Windows, and Linux OS - multiple chip communication protocols (I2C, SPI, UART, and SWI)

All platform dependencies are contained within the HAL (hardware abstraction layer).

Currently, the vast majority of testing has been performed on:

- [ATSAMR21 Xplained Pro \(cryptoauth-r21-host firmware\)](#)
- [ATSAMD21 Xplained Pro \(cryptoauth-d21-host firmware\)](#)

These two host containers implement a host test environment and test console to exercise tests. They presume that a CryptoAuth Xplained Pro or other I2C socket for an ATECC508A are connected to the I2C pins of the host Xplained Pro development board.

The unit tests and basic tests exercise the core datasheet commands of the device as well as the more convenient, basic API methods.

If you need an example of how to use a command, the tests are a good place to reference.

## Object Architecture

Even though this is a C library, it follows object-oriented design patterns.

An object is minimally defined to be data and the actions which operate on that data.

Each CryptoAuth device is a composite object, a structure which includes the command table (list of commands) which are valid for the device, and the data used to hold the state of that device.

ATCADevice is the object which represents the Atmel CryptAuth device

ATCACommand is the object which represents the valid methods of the Device.

ATCAInterface is the physical interface object (I2C or SWI instance). Currently, each Device may have a single OATCAInterface.

ATCADevice represents an ATSHA or ATECC family device.

In order to add new protocol support for a platform, you provide a HAL (hardware abstraction layer) C file for the protocol and target. In your project's IDE or Makefile, you select which HAL support you need for the hardware configuration. Generally, there are separate files for each protocol and platform combination - (ie: samd21\_i2c\_asf.c would target SAMD21 MCUs with I2C using the ASF low-level driver support.)

## Directory Structure

```
./lib - The primary library source code
./lib/docs - AppNotes and Doxygen HTML documentation for the library API. Load "index.html" in your browser
./lib/basic - the Basic API way to access the core classes
./lib/atcacert - cert data and cert i/o methods
./lib/hal - hardware abstraction layer code for supporting specific platforms
./lib/crypto - software implementation of crypto algorithms

./test - Unity test code to exercise unit tests for datasheet commands and Basic API methods
./lib/atcacert/test - Unity test code to exercise all CryptoAuthLib certificate features

For production code, test directories should be excluded by not compiling it into
a project, so it is up to the developer to include or not as needed. Test code adds
significant bulk to an application - it's not intended to be included in production code.
```

## Tests

There is a set of unit tests found in the test directory which will at least partially demonstrate the use of the objects. Some tests may depend upon a certain device being configured in a certain way and may not work for all devices or specific configurations of the device.

## Using CryptoAuthLib (Atmel CryptoAuth Library Next Generation)

Using a new library is often easier when you can load an example and see how it works. We've provided examples in the form of "host containers" which are host projects that incorporate CryptoAuthLib and target various processors or communication APIs.

Find a host container that matches your platform. The current list of host containers are:

- [cryptoauth-d21-host](#)
- [cryptoauth-r21-host](#)
- [cryptoauth-win-host](#)

New host containers are under development for various targets

The best way to learn how to use CryptoAuthLib is to study the host test projects that exercise the library and ATECC and ATSHA devices.

New examples will be forthcoming as the software matures. Continue checking the [CryptoAuthentication](#) web page for new updates.

## Using Git to Incorporate CryptoAuthLib as a Submodule

You can include this project in your own project under git.

Using CryptoAuthLib as a git submodule, you can maintain your application separately from CryptoAuthLib.

If your project is already in in git but you haven't yet intergrated CryptoAuthLib, change to the directory where you want to put atca-ng

```
git submodule add -b master <giturl to atcalib>
```

This adds CryptoAuthLib as a subdirectory and separate git repo within your own project. Changes and commits to your project vs CryptoAuthLib will remain separated into each respective repository.

If there is a project you want to checkout that already incorporates CryptoAuthLib as a submodule if you clone the repo that incorporates atca-ng, after cloning, you'll still need to fill out the atca-ng submodule after cloning: `bash git submodule init git submodule update --remote cd cryptoauthlib git checkout master`

Now that CryptoAuthLib is a full-fledged submodule in your git project, in order to easily add it to your project within Atmel Studio, please see this [tip](<http://avrstudio5.wordpress.com/2011/07/12/tip-add-existing-multiple-files-and-folders-to-an-avr-studio-project-quickly/>)

## Incorporating CryptoAuthLib in a project

- 1) In your Makefile or IDE, choose the HAL support you need from the HAL directory and exclude other HAL files from your project.
- 2) For I2C interfaces, define the symbol ATCA\_HAL\_I2C in your compiler's symbol definitions. This will hook up the CryptoAuthLib interface class with your HAL implementation of I2C methods.
- 3) HAL implementations for CDC and HID interfaces to the ATCK101 are also included for use with Windows or Linux versions of the test host.
- 4) Connect a USB to the CDC port of the host platform such as the ATSAMR21 or ATSAMD21 Xplained Pro development boards and type 'help' and hit the return key. This will list a set of available commands to exercise the CryptoAuthLib and chip. The easiest way to see if your board and device are properly connected is to issue a 'sernum' or 'info' command.

```
$ help
Usage:
u204 - run unit tests for SHA204A
u108 - run unit tests for ECC108A
u508 - run unit tests for ECC508A
b508 - run basic tests on ECC508A
lockstat - zone lock status
lockcfg - lock config zone
lockdata - lock data and OTP zones
cd - run unit tests on cert data
cio - run unit tests on cert i/o
info - get the chip revision
sernum - get the chip serial number
crypto - run unit tests for software crypto functions

$ sernum
serial number:
01 23 58 0C D9 2C A5 71 EE
$ info
revision:
00 00 50 00
$
```