To discuss the methods used to secure an SQLite database, we must first define what it means to ‘secure’ it. One basic principle that is widely accepted in terms of information security is known as the CIA Triad, or “Confidentiality, Integrity, and Availability’. In terms of SQLite, in particular, the primary concerns as it relates to CIA are unauthorized access, and ‘SQLIA’ (SQL Injection Attacks).

Confidentiality in SQLite is commonly achieved through cryptography. Encryption of SQLite files is most commonly achieved using SQLCipher, an SQLite extension that is performs transparent encryption of SQLite databases. In addition to encrypting the saved database, cryptography is a critical line of defense against the most common threat of web applications – SQL Injection. Many times, SQLite databases are subject to accepting untrusted inputs (such as validating a username and password on a web application). By encrypting a database, these attacks can be prevented by altering the untrusted inputs using secure encryption key prior to running the otherwise susceptible query.

Another common practice to achieve confidentiality in SQLite is to implement the ‘principle of least privilege’, which limits privileges of end users to the minimum required to complete their authorized task. This is achieved with a file system of a devices operating system (NTFS or FAT32).

The next element of security, Integrity, is ensured in SQLite databases using constraints, which are used to define conditions that must be met before insert/update/delete operations can be performs on a column. These constraints are implemented when tables are created or altered, when using the CREATE TABLE or ALTER TABLE statements, respectively. Constraints in SQLite fall into the following categories

Domain integrity, which refers to constraints set on table columns in order to restrict the input of values that do not meet the constraints requirements.

Entity Integrity, which are used to ensure that values are unique within the specified column.

Referential Integrity, which are used to validate the relationship between rows and values in different tables.

User-Defined Integrity, which refers to using customized constraints that cannot be achieved using system-defined ones. They are implemented using functions, triggers, and other stored procedures.

The last of the triad, Availability, is hardly a concern when using SQLite. It has been proven to be a highly reliable software library since it’s 1.0 version was released in August of 2000. A likely reason that it has been used in billions of devices around the globe, and will likely continue to be widely used for decades to come.