3-Rolling Code (hopping code):

To improve on this system a number of changes were made to essentially prevent replay attacks (somewhat). This is called rolling code. The way this was done was by making the remotes and cars (or other devices) have a synchronised starting code that was sent and an algorithm that determined the following code to be sent next so that the same code was never repeated. This is very similar to the OTPs used by RSA 2FA tokens.

In a rolling code (code hopping) system, the keyfob transmitter maintains a synchronization counter C . This is usually called “pairing” a remote and relies on some configuration of the device, incremented every time a button is pushed. The car receiver stores the most recent validated synchronization counter it has received N.

When a rolling code receiver gets a message with a keyfob serial number that seems to match one of the authorized keyfobs in its internal memory, it extracts the keyfob's synchronization number C from the message and compares it to the most recent validated synchronization number N (A PRNG seed).

C in the rolling window of acceptance -- C is more than but within "a few" codes K of the last validated transmission ($0 < C-N < K+1$).

When the counter C in the message is in the window of acceptance, then the car accepts that message as an authentic message, the car overwrites N with the latest value C, and the car unlocks the doors or does whatever else the message says to do.

(As long as you hold down the button, the keyfob will repeatedly transmit bit-for-bit the same message over and over several times a second. It isn't until you let up the button and press it down again that it transmits a message with the next sequential counter number).

So it is possible that the sender will get out of range of the k accepteded codes?

Yes. The AVR411 receiver defaults to WINDOW_SIZE = 100. So with a typical AVR411 system, if the car doesn't hear 100 consecutive button-presses, then the car will ignore the keyfob when it finally gets back in range.

With a typical KeeLoq system, if the car doesn't hear 16 consecutive button-presses, then the car will not unlock the doors on the first button-press of the keyfob when it finally gets back in range. However, if the car then hears 2 consecutive button-presses within the resynchronization window ($K < C-N <$ Resynchonization_Window), typically over a thousand button-presses, the car will resynchronize and unlock the doors. With a typical HCS301 system, if the car doesn't hear a few thousand consecutive button-presses, then the car will ignore the keyfob when it finally gets back in range.

note: always check the manual of used remote or receiver for resynchornization.

How does they sync the point of the sequence? ... How are they implemented?

Before the keyfob can be used, the car must "learn" the keyfob. Typically holding down the "learn

button" ("learn switch") for 10 seconds erases the receiver's memory of every keyfob; then letting up on that button makes the receiver go into "learn mode". Typically, pressing both the lock and the unlock button simultaneously on the keyfob makes the keyfob transmit its unique secret key to the receiver. The receiver -- in learn mode -- memorizes the keyfob serial number(s) it hears as authorized keyfobs, and also memorizes the secret key and the current synchronization counter N(PRNG's seed). After everything is quiet for 10 consecutive seconds with no button presses, the receiver times-out of learn mode and goes into normal mode.

note: to enter the learn mode it differ from one remote control to another.

Summary on the operation of rolling code :

a rolling code is much safer from fixed code and learning code also much harder to crack

a rolling code use some sophisticated algorithm to send data which make it robust

a rolling code device should be configured on the receiver by using learning mode which will make the keyfob or remote transmit it's secret code to the receiver which will make it learn the serial number of the remote and it's PRNG seed initial value.

after we exit learn mode we can use remote normaly

a receiver should have a synchronization counter limit k which is usually 255

A syncronization seed PRNG suppose 1

A Syncronization counter C which is incrimented each time by 1

so we have a range from 1 <C-N < 255 and suppose remote increment by 1 each time pressed so if we press the remote to reach a value >255 so the remote will go out of sync with the receiver.

Used chips:

HCS301,etc....

Example on a rolling code RF remote chip:

A general idea on the basic operation of the hcs301 chip algorithm

FIGURE 1-1: CREATION AND STORAGE OF CRYPT KEY DURING PRODUCTION

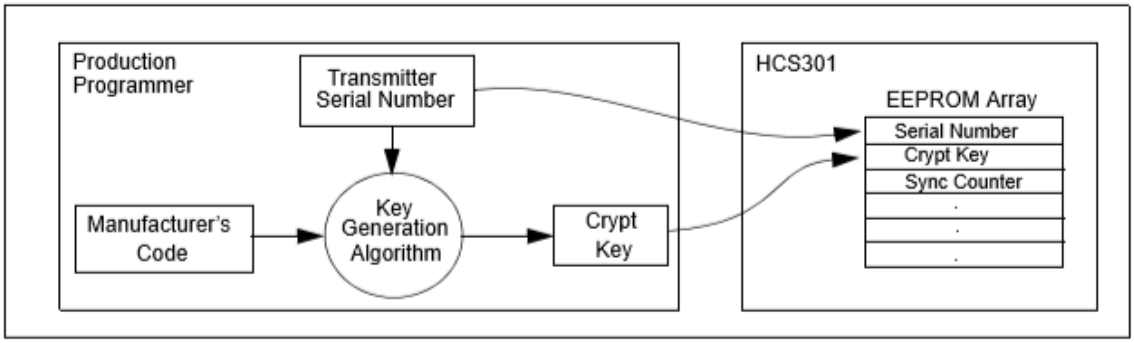

FIGURE 1-2: BUILDING THE TRANSMITTED CODE WORD (ENCODER)

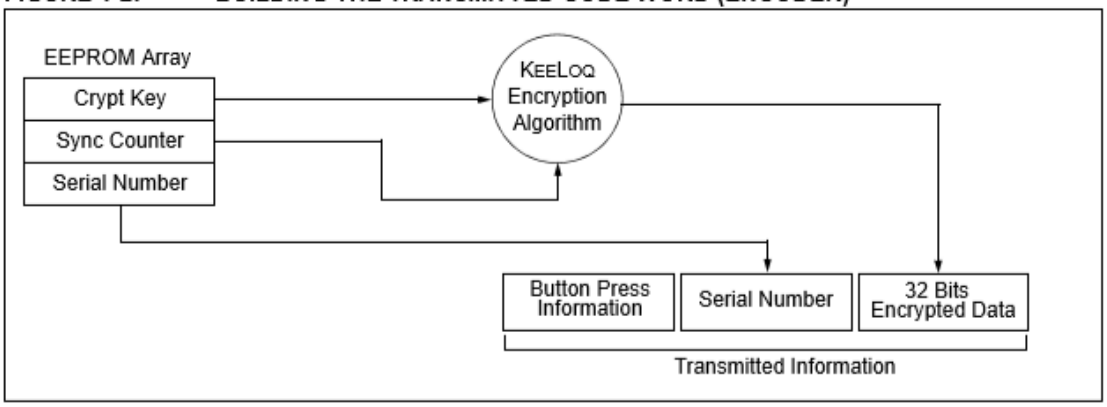

FIGURE 1-3: BASIC OPERATION OF RECEIVER (DECODER)