Keyless Cars Security

CPS and IoT Security

Alessandro Brighente

Master Degree on Cybersecurity

Università degli Studi di Padova

SPRITZ Security & Privacy Research Group

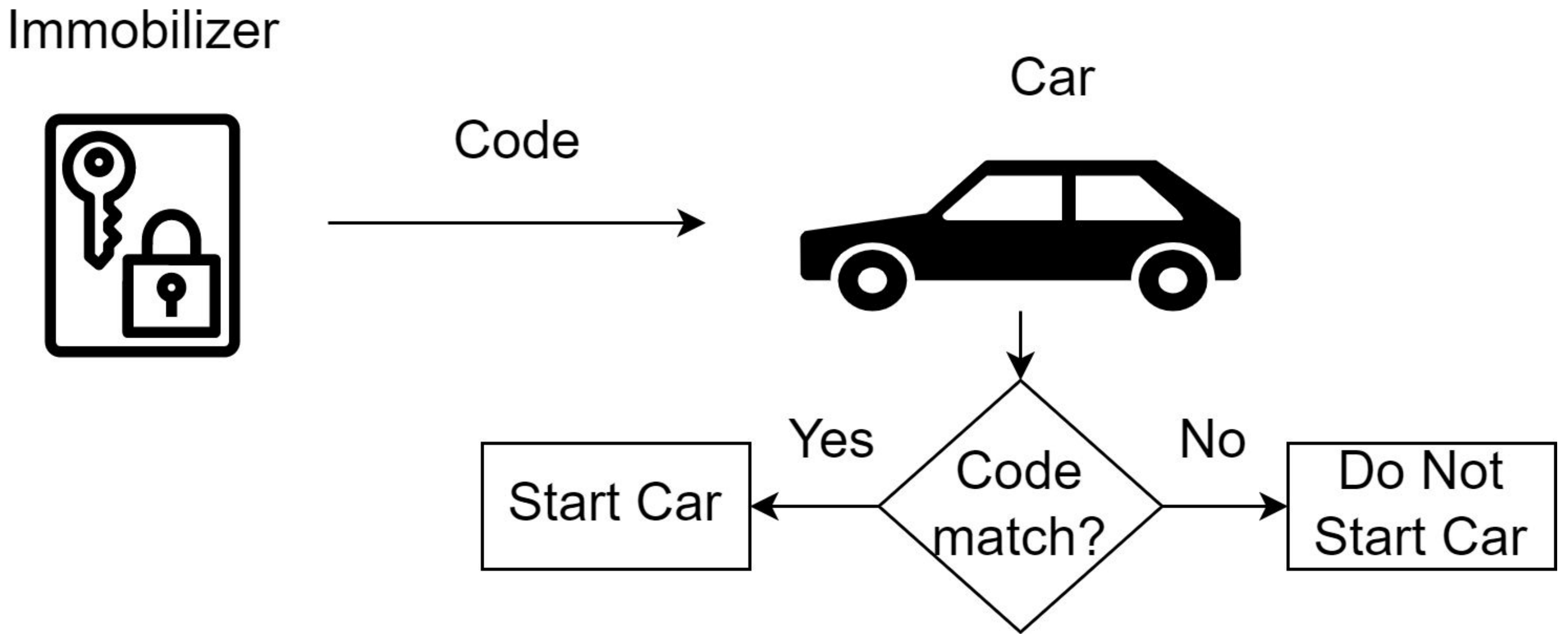
Physical Keys

- At the beginning, there were ignition systems
- Ignition switch is the first step to get a car to start
- Turning the key sends a signal
- This signal starts the ignition system and ignites the fuel vapor
- This system has a critical flaw: you can generate this signal in many ways if you have physical access to the car

From Physical to Cyber Keys

- The first solution involving a cyber-component to enforce security is the *immobilizer*
- The first generation of immobilizers used a small chip embedded into the head of the car key
- Purpose: when the driver inserts the key into the cylinder the chip emits a code/serial number that can be received by the antenna inside the cylinder
- If the code matches the one the car expects, then ignition starts

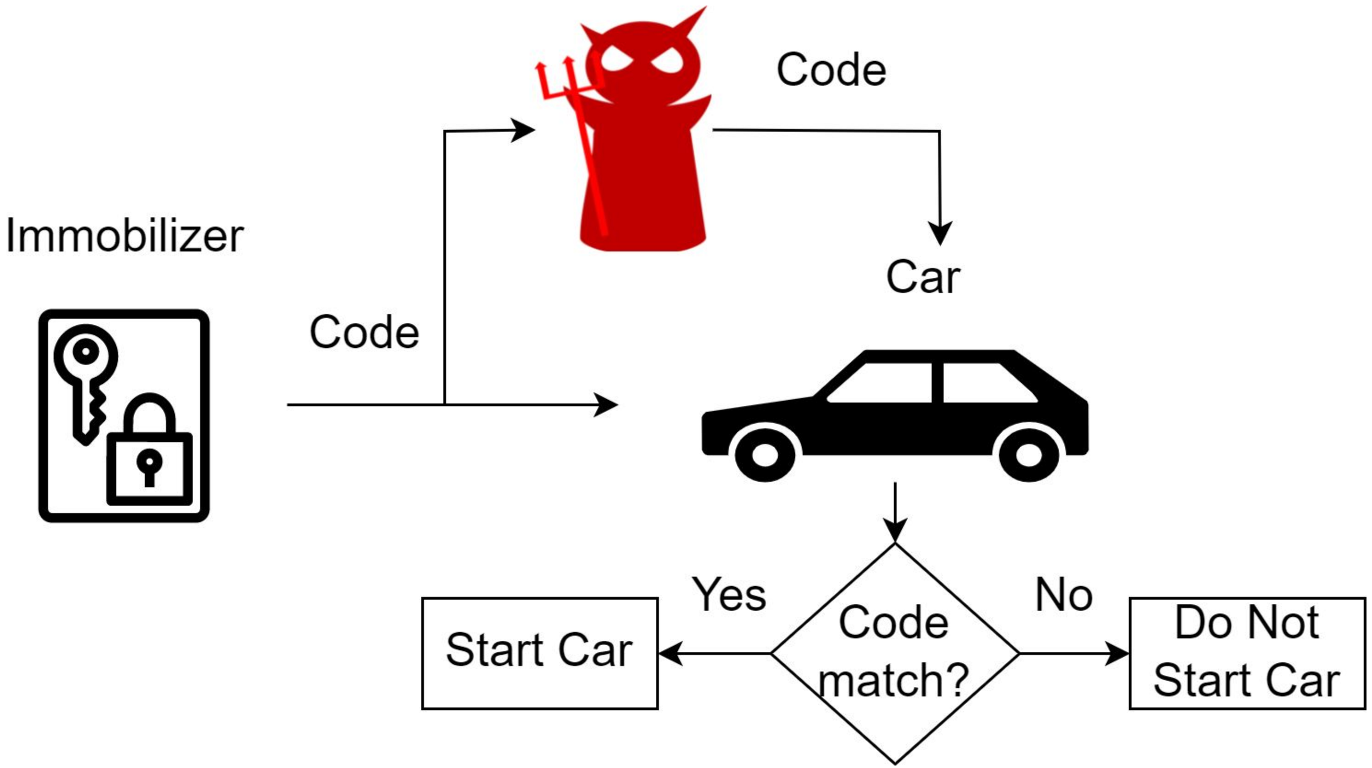
Immobilizer



Flaw of Immobilizers

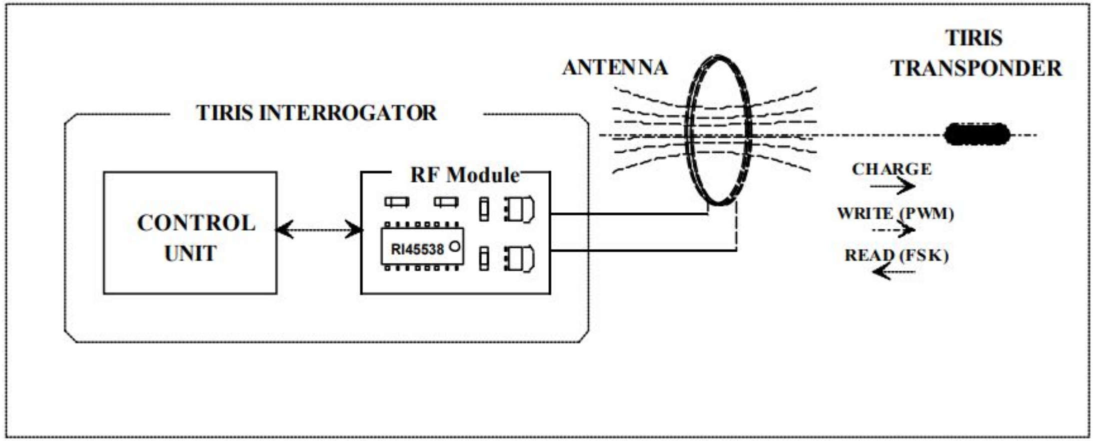
- At a first glance, the solutions is nice since it prevents hotwiring and lockpicking
- However, the immobilizer always transmits the same code
- An attacker with a eavesdropping equipment can easily record the code and later replay it when stealing the car
- As complicated it may seem, it is actually not thanks to devices called *code grabbers*

Immobilizer



Digital Signature Transponder

- Immobilizers of cars such as Ford, Toyota, and Nissan were based on Digital Signature Transponders (DSTs)
- The DST is a tiny RFID chip that, among the others, is enabled with a cypher and a 40-bit secret key

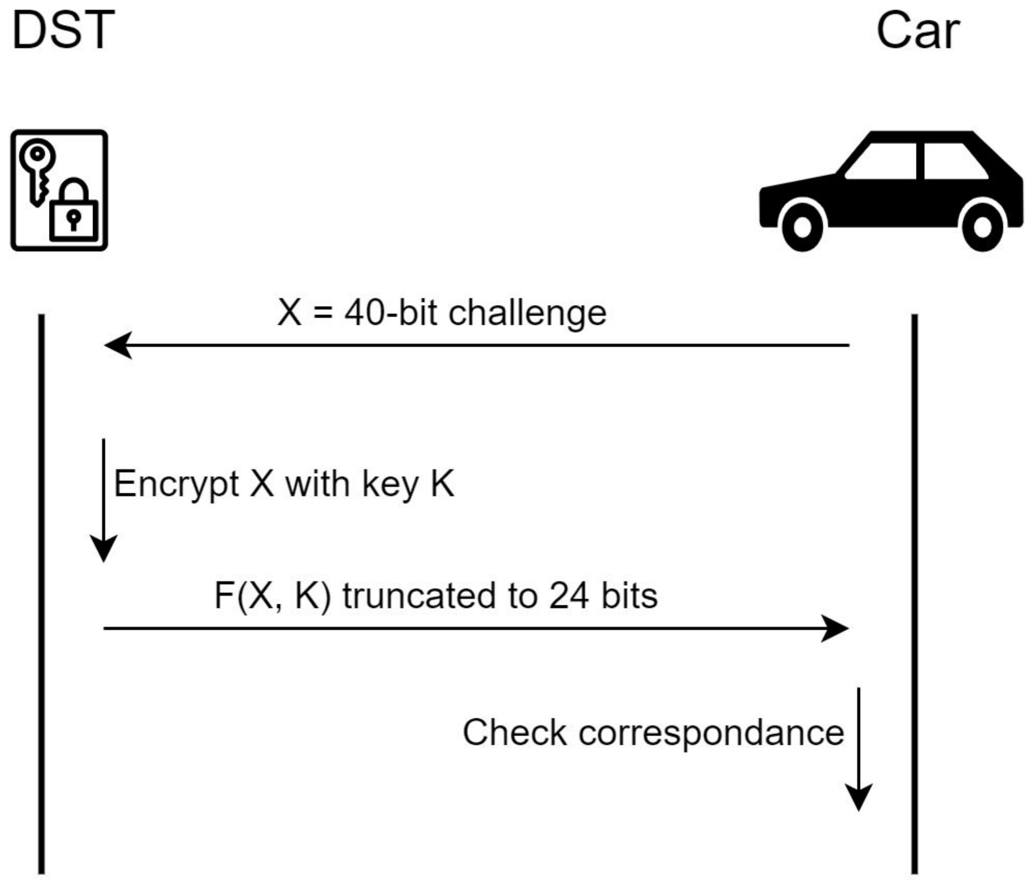


TIRIS DST by Texas Instruments

Digital Signature Transponder

- Immobilizers of cars such as Ford, Toyota, and Nissan were based on Digital Signature Transponders (DSTs)
- The DST is a tiny RFID chip that, among the others, is enabled with a cypher F() and a 40-bit secret key
- The DST and the car both share a copy of the private key K

Digital Signature Transponder

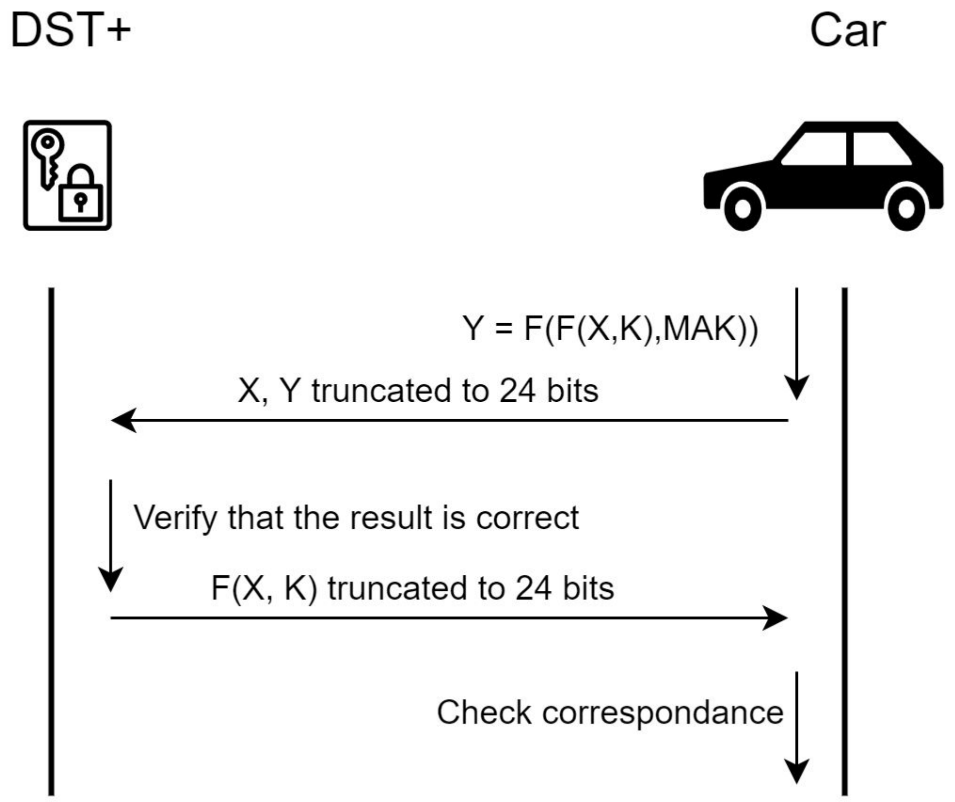


Cloning Attack

- An adversary cannot replay a response as the car should be sending a different challenge for each round
- However, there is a huge problem with the key length, i.e., 40 bits
- An attacker might send a challenge and record a response from a car and try all the 1.1 trillion possible key combinations to infer the private keys
- Huge number, but requires few hours on a FPGA
- The only requirement for an attacker is to get close enough to your key while turning on the car

DST+

- Solve the cloning attack
- Car and DST+ share a key K and a Mutual Authentication Key (MAK)
- If the challenge is not the one expected, the DST+ does not respond


DST+
Car
Y = F(F(X,K),MAK))
X, Y truncated to 24 bits
Verify that the result is correct
F(X, K) truncated to 24 bits
Check correspondance

Further Protection Measures

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

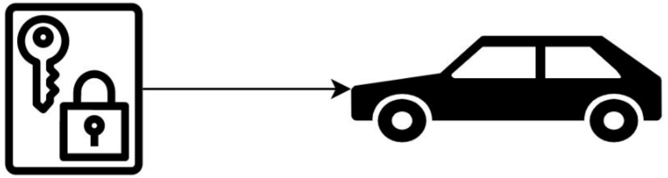
- An evolution of keys is Passive Keyless Entry Systems (PKESs)
- It does not require interaction with the user, thus passive
- The car not only checks for the presence of a legitimate code, but checks also where the key is
- It uses a low-frequency RFID channel to check if the key fob is in remote distance (up to 100 m), outside the car (1-2 m from the door handle), or inside the car
- Only in the last case the engine starts

Different Threat Model

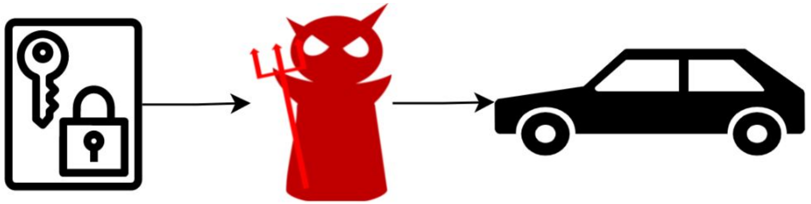
- A different threat model envisions having no direct access to the car’s key
- Still, keys use wireless communications to talk with the car and execute a challenge response algorithm
- How do you steal a car in a minute exploiting this technology and why would this work?

Relaying Signals

- We define as a relay attack a special type of man in the middle attack, where a non legitimate device establishes a communication between two non-proximal legitimate devices



Legit communication



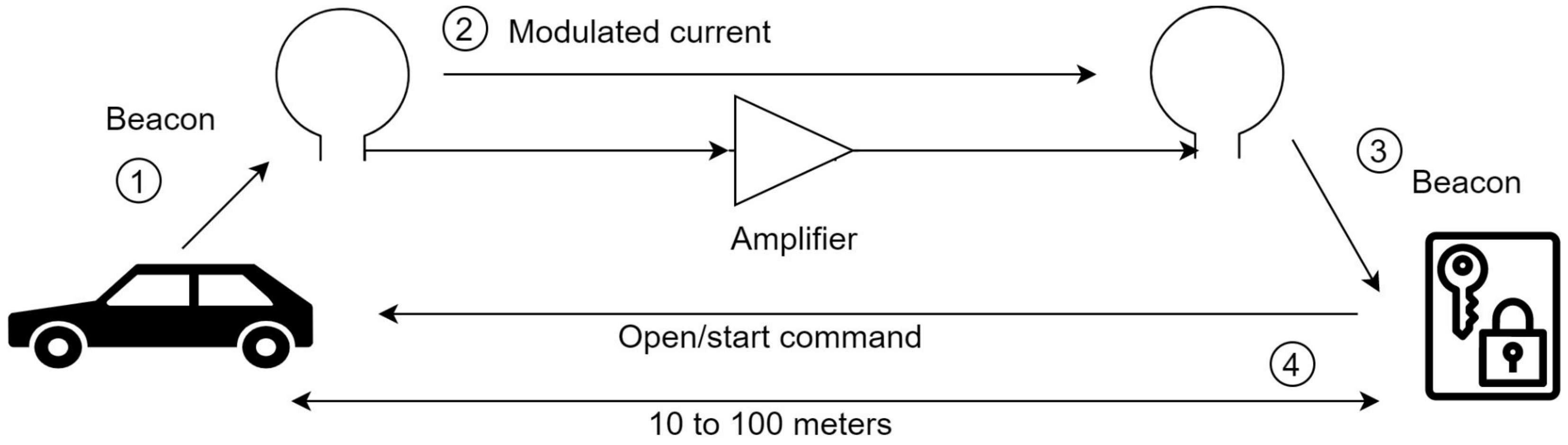
Relayed communication

Advantage of Relay

- When relaying signals, we need to care only about the physical layer
- We do not need to interpret the signal, modify it, infer keys,..
- We just need to demodulate the signal, amplify it if needed, transmit it as digital information using RF, and modulate it near the victim tag
- **Note:** It adds some delay, so we must be sure that the introduced delay is within the range accepted by the application under attack

Relay over Cable

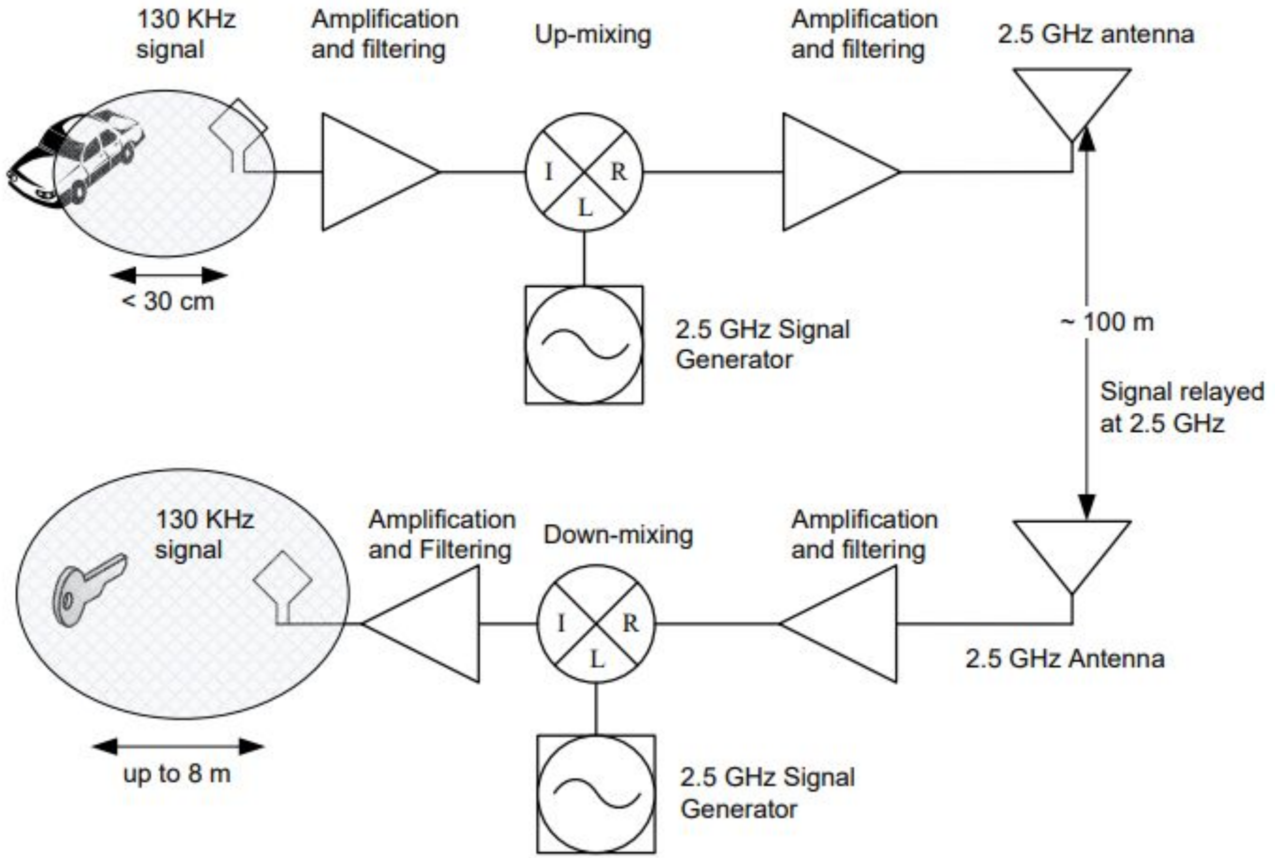
- Two loop antennas connected via a cable that relays the low frequency signal between them



Relay over the Air

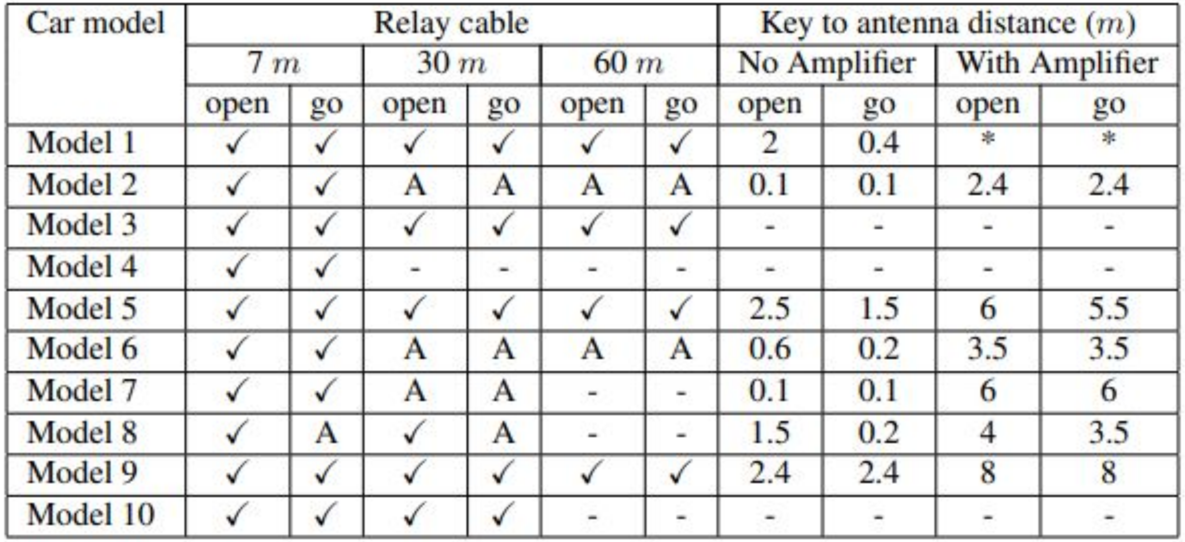
- Cables may bring suspicions.. So let's just remove them
- The relay system is now composed by two parts, an *emitter* and a *receiver*
- The emitter captures the low freq. signal and upconverts it to 2.5 GHz, amplifies it, and transmits it over the air
- The receiver downconverts the signal back to low freq., it amplifies it agains, and sends it to the loop antenna

Relay over the Air



Results

Table 4. Experimental results distances summary. Legend: '✓' relay works without amplification, 'A' with amplification, '-' not tested, '*' value will be updated



Car model	Relay cable						Key to antenna distance (m)			
	7 m		30 m		60 m		No Amplifier		With Amplifier	
	open	go	open	go	open	go	open	go	open	go
Model 1	✓	✓	✓	✓	✓	✓	2	0.4	*	*
Model 2	✓	✓	A	A	A	A	0.1	0.1	2.4	2.4
Model 3	✓	✓	✓	✓	✓	✓	-	-	-	-
Model 4	✓	✓	-	-	-	-	-	-	-	-
Model 5	✓	✓	✓	✓	✓	✓	2.5	1.5	6	5.5
Model 6	✓	✓	A	A	A	A	0.6	0.2	3.5	3.5
Model 7	✓	✓	A	A	-	-	0.1	0.1	6	6
Model 8	✓	A	✓	A	-	-	1.5	0.2	4	3.5
Model 9	✓	✓	✓	✓	✓	✓	2.4	2.4	8	8
Model 10	✓	✓	✓	✓	-	-	-	-	-	-

Possible Countermeasure

- Distance bounding denotes a class of protocols where the prover measures an upper bound on its distance to another entity
- We implement RF distance bounding to verify the mutual proximity of the car and the key
- The distance bound is obtained from a rapid exchange of messages
- The verifier sends a challenge to the prover
- Upon reception of the response, the verifier measures the communication round trip time to obtain an estimate of the distance

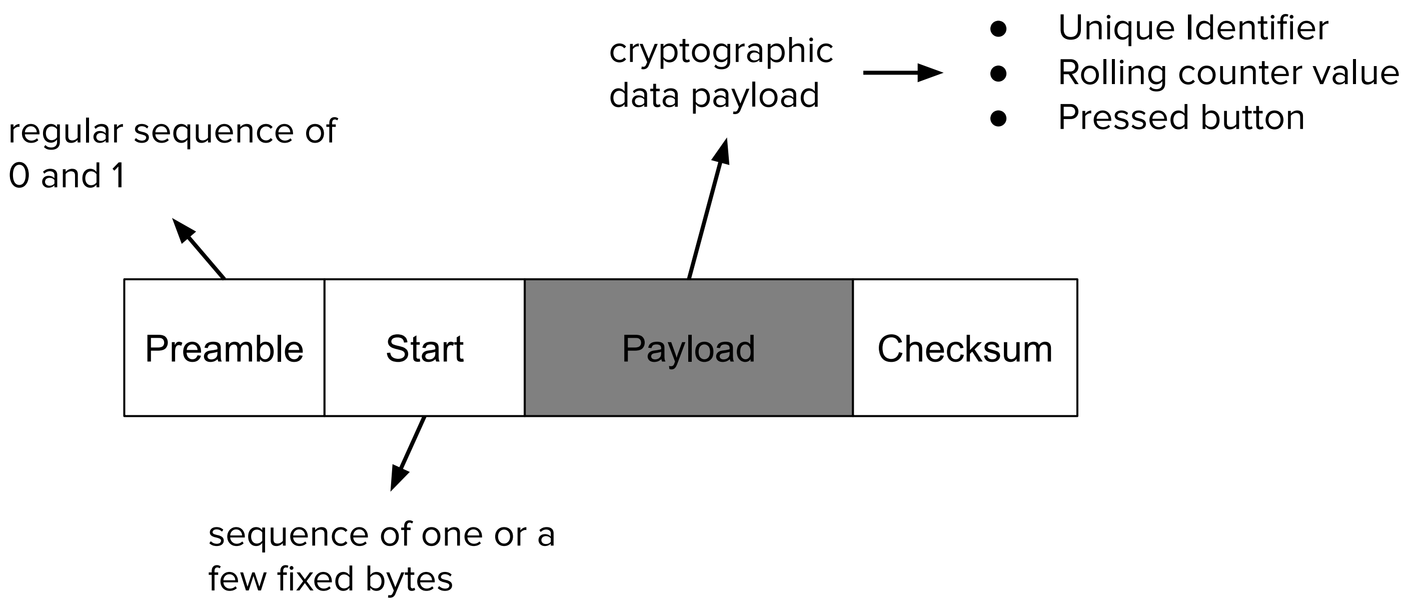
Remote Keyless Entry and Start

- Remote Keyless Entry (RKE) relies on unidirectional data transmission from the remote control (in the car key) to the vehicle
- The key is hence active, and upon pressing a button transmits signals in one of the bands 315 MHz, 433 MHz, or 868 MHz (depending on the country)
- RKE systems enable the user to comfortably lock and unlock the vehicle from a distance, and can be used to switch on and off the anti-theft alarm, when present

Remote Keyless Entry and Start

- The first generation of RKE used no cryptography, solely relied on a fix-code signal
- However, this makes replay attacks super easy
- The next generation of RKE is named after **rolling code systems**
- Rolling codes use cryptography and a counter value that is increased at every button press
- They use a conjunction of the counter and other signals as input to the cypher, and the car checks this information to assess the validity of the signal

Rolling Codes



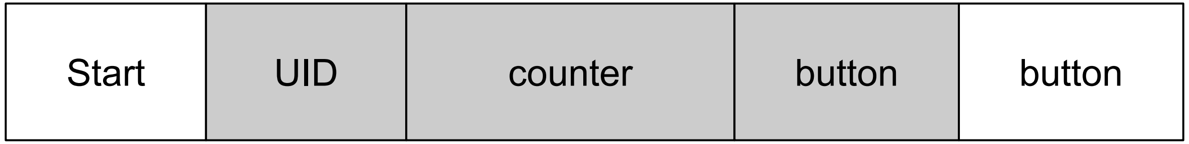
- Unique Identifier
- Rolling counter value
- Pressed button

Authentication in Rolling Codes

- **Implicit authentication:** the entire payload is symmetrically encrypted, and the receiver checks the UID and if the counter is in its validity window
- **Explicit authentication:** the sender computes some sort of message authentication code over the data payload and appends it to the packet

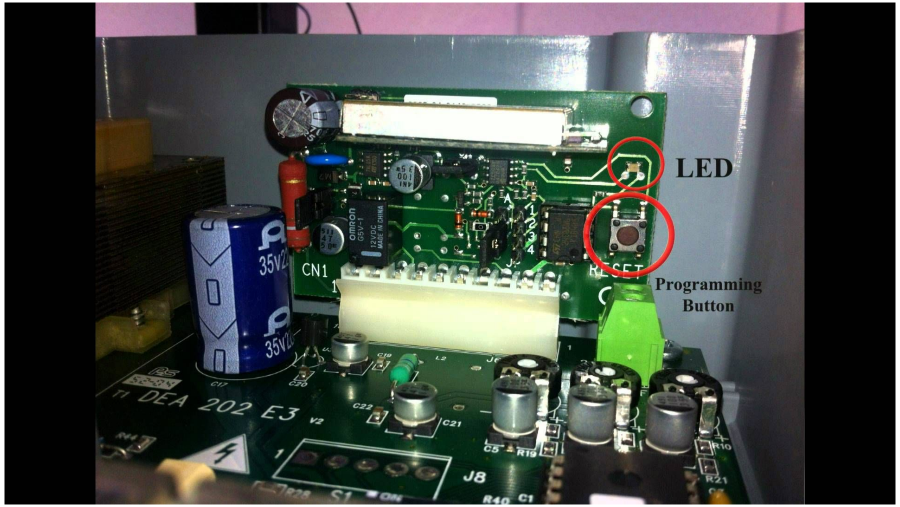
One of the many Implementations

- The grey part is the encrypted part
- The payload is encrypted using a proprietary block cipher
- We assume that such a cypher is the AUT64 (as found in many cars from the VolksWagen group)



One of the many Implementations

- Both the car and remote are synchronized to an initial number (e.g., PRNG seed) ➔ pairing a remote
- It usually relies on some configuration, such as pressing a button or a *master* key



One of the many Implementations

- Once the remote and the car are synchronized, they use an algorithm to choose an initial number x and transform it to the next number in the sequence (x+1)
- The following key press will take the result of the previous keypress as the input
- If a single remote press is intercepted there is not way to determine the next code expected by the receiver (due to encryption)

One of the many Implementations

- Devices not only store the next code that needs to be transmitted, but a significant number of them (e.g., 255)
- This is fundamental to have the process working even if in case of erroneous keypresses that increase the counter only on the key side
- As soon as a button keypress that is valid is received, the list is updated to be x iterations from that keypress
- However, if the number of keypresses is too large, there is no way to catch up on sync

One of the many Implementations

- There are few well-known manufacturers providing rolling code based RKE
- Microchip Technology provides Classic, Advances, and Ultimate KeeLoq with publicly available documentation and data sheets
- Companies like NXP, Omron, and TI provide proprietary solutions
- Let’s take a deeper look at KeeLoq (as it is open source), provided that the idea behind rolling codes is always the same

KeeLoq

- Uniqueness in key fob transmission is achieved by incrementing a 16 byte counter in the key fob (and vehicle upon reception)
- A button press is valid if counters at each side are in sync
- If the difference between they key and car counter is small (i.e., < 16), counter synchronization takes place immediately at the first button press without additional steps
- Sync means that the receiver unit in the vehicle invalidates all non-received codes before the one present in the last keyfob signal

KeeLoq

- If the difference is higher than 16 and lower than 2^{15}, the receiver temporarily stores the counter and waits for a subsequent transmission (same button has to be pressed once more)
- If the subsequent transmission has counter previous +1, the receiver resynchronizes on the last transmission received
- If any of the above fails, the key fob signal received by the vehicle is discarded

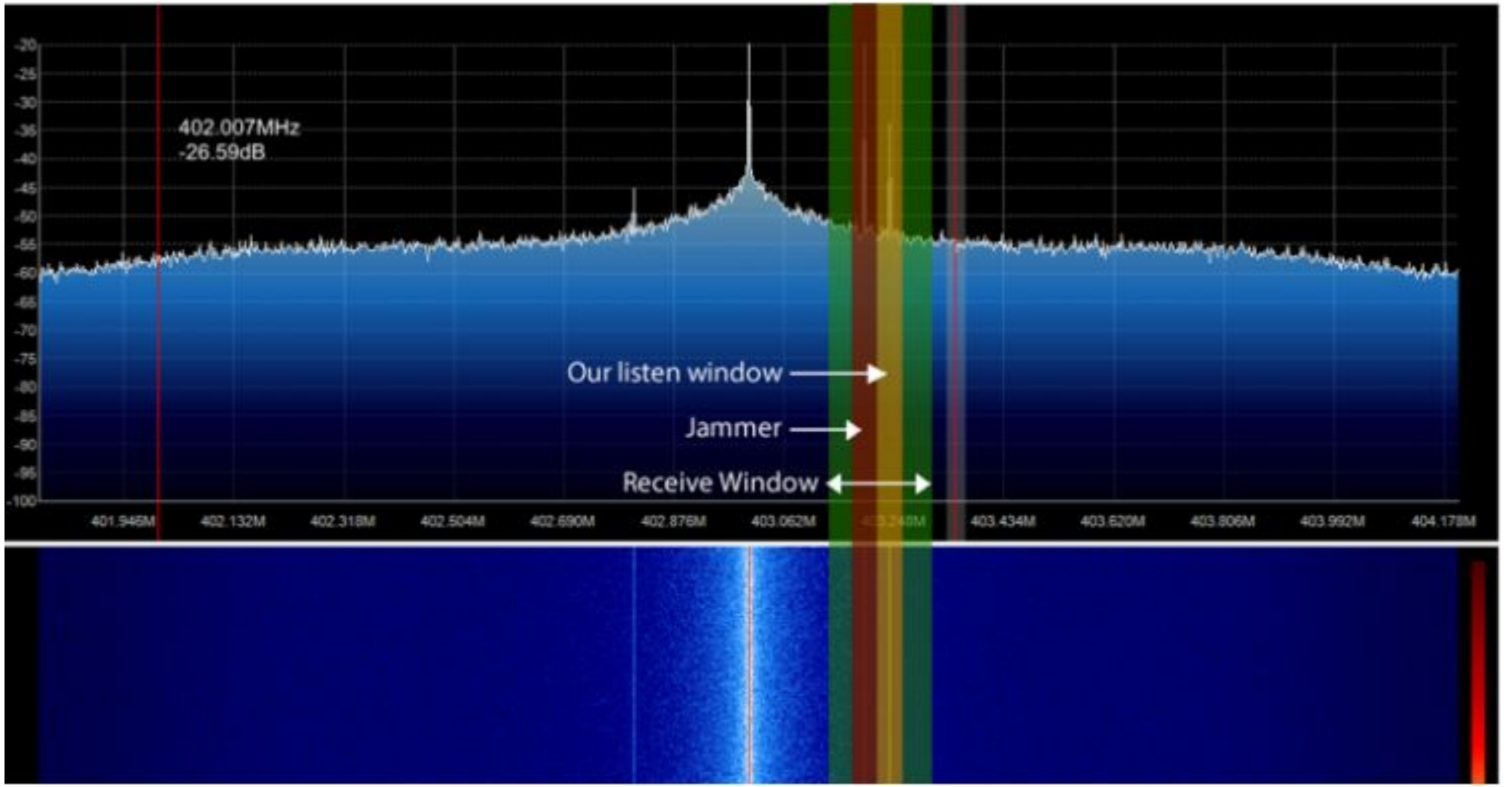
Missing Link Attack

- As an attacker, we would like the receiver not to be able to receive any of the radio transmissions from the remote
- To achieve this, we can jam the communication between the key and the car
- This is a technique used by thieves to have drivers not to close their cars in the parking lot
- Notice that we can use a transmitter on a similar frequency, as receivers have large windows for detecting signals (to account for temperature changes and flat batteries)

RollJam Attack

- Remote jamming works, but the person walking away from the car may notice that it did not lock (no noise)
- We want to develop a technique that is indistinguishable from the standard interaction between car and owner
- We integrate jamming and replay attack in the *RollJam* attack
- We want to jam the signal, capture it, and somehow reply it as a legitimate communication between key and car

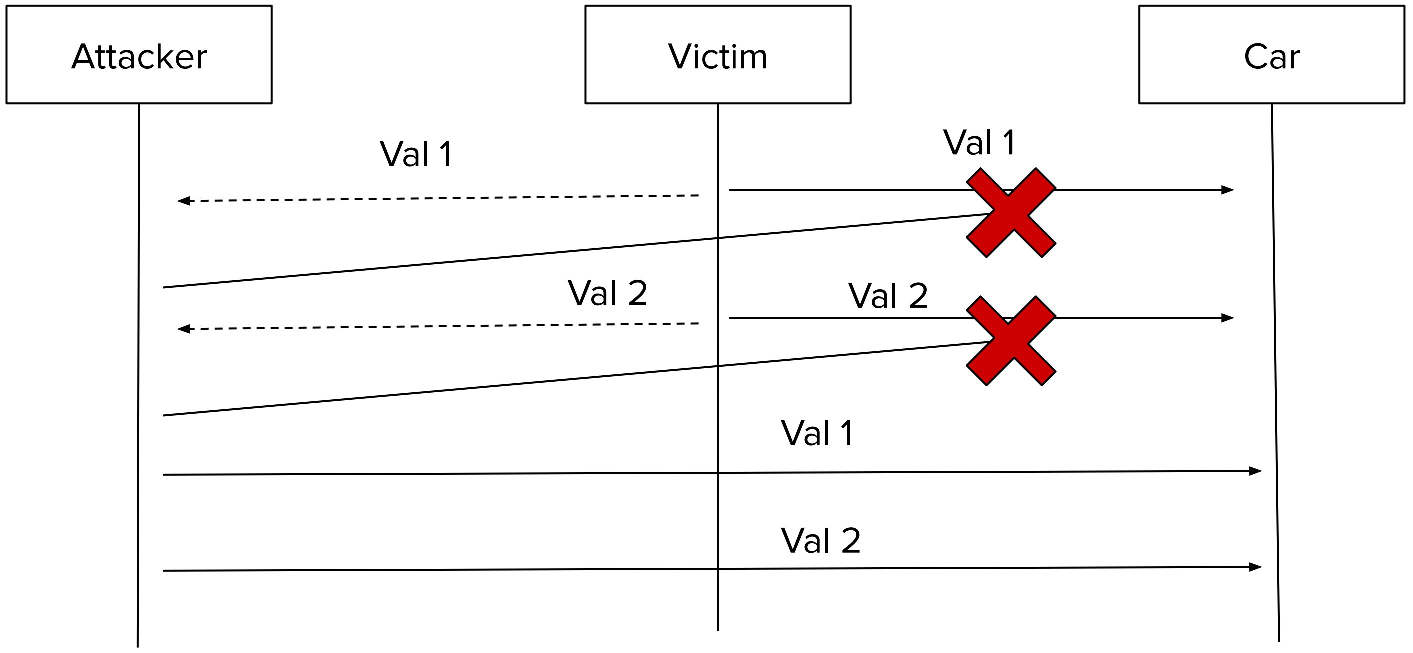
RollJam Attack



RollJam Attack

- While the signal is jammed, we can use the same window to capture the first legitimate message that the key sent to the car
- This is possible because the jamming is not in the exact same frequency of the signal
- When someone cannot get a remote to work, the instinctive reaction is to press the remote again
- As soons as this happens we can get a second (successive) instance of a legitimate transmission
- We can replay the two signals in the captured order

RollJam Attack



Example

RollJam Problems

- Many modern cars use different frequencies for lock and unlock
- If the attacker only monitors the frequency of the unlock button, then they will only be able to unlock the car
- It requires the legitimate owner to send a legitimate lock, before being able to perform the unlock,given that lock and unlock use the same rolling code
- Solutions: permanently jam the lock frequency such that the user will need to manually lock the car

RollBack Attack

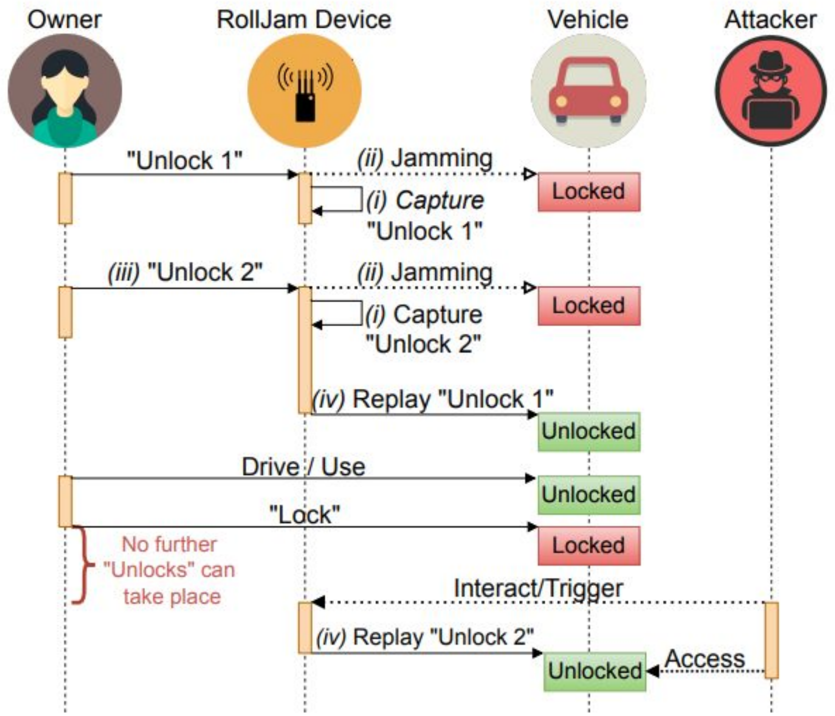
- Again, RollJam requires the jammer to be always there
- Although rolling codes have been designed to prevent replay attacks, they are still chances that replay might work
- We said that there may be some synchronization issues between key and car for which resynchronization is needed
- Let’s exploit this mechanism to evade the security provided by rolling codes

RollBack Attack

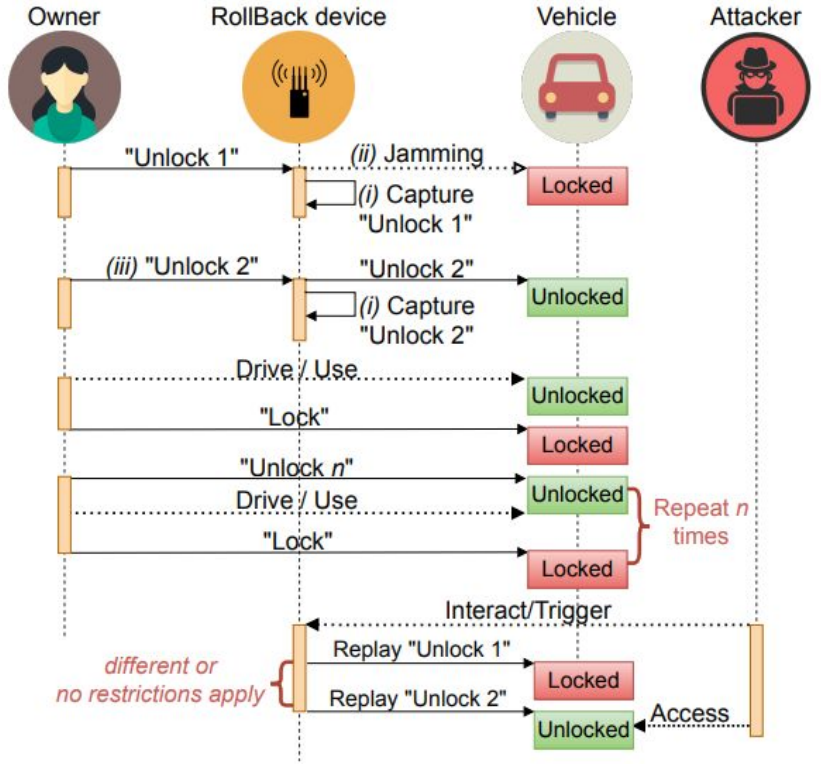
SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

RollJam



RollBack



RollBack Attack

- Again, RollJam requires the jammer to be always there
- Although rolling codes have been designed to prevent replay attacks, they are still chances that replay might work
- We said that there may be some synchronization issues between key and car for which resynchronization is needed
- Let’s exploit this mechanism to evade the security provided by rolling codes

RollBack Reconnaissance

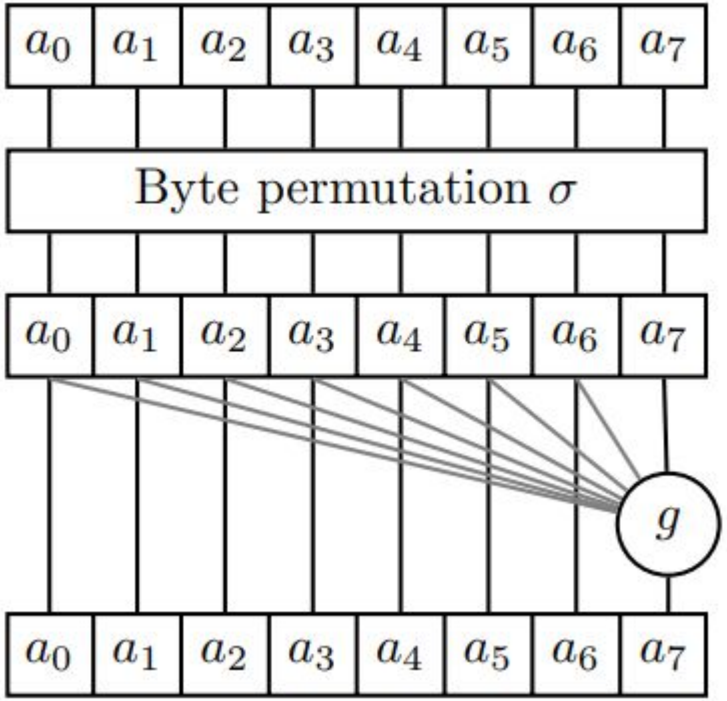
- The attacker places the RollBack-device near the target car
- When the victims comes back to their car, they will try to unlock it via the keyfob sending $Send_V(\mathcal{S}^i_{unlock})$
- RollBack-device:
  - Capture the signal
  - Jam the frequency band to prevent reception
- The victim assumes lousy reception and resses the same button again, sending $Send_V(\mathcal{S}^{i+1}_{unlock})$
- The RollBack-device captures the signal, but lets the car receive it ➜ the victim unlocks and drives away (no doubts)

RollBack Exploitation

- By design, the two codes captured by the attacker are no longer valid
- The victim can lock and unlock their car at will
- However, we have also seen that resync requires two consecutive packets
- This means that the attacker can replay the two previously captured packets at any time and get access to the car

One of the many Implementations

- AUT64 is an iterated cipher that operates on 8-byte blocks
- In each round, the state is first permuted
- Then, byte 7 is updated using the round function g(a_0, ..., key_i)
- Key_i = 32-bit round key
- Total of 12 rounds

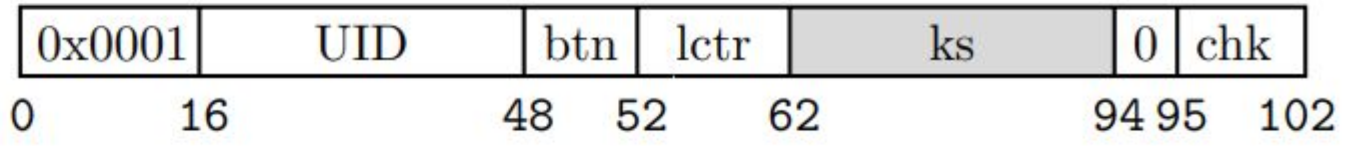

a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
Byte permutation σ
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
g
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7

Master Problem

- Doing the math, the effective key size of AUT64 is 91.55 bit
- Finding the key via exhaustive search is not practical
- However, the problem is that this RKE system uses a global master key which is independent from the vehicle or remote control
- This means that the same key is stored in millions of ECUs and RKE remotes, without any key diversification
- The sole means by which the vehicle determines if a rolling code is valid is hence by whitelisting certain UIDs and checking if the counter is within the validity window

RKE Based on Hitag2 Cipher

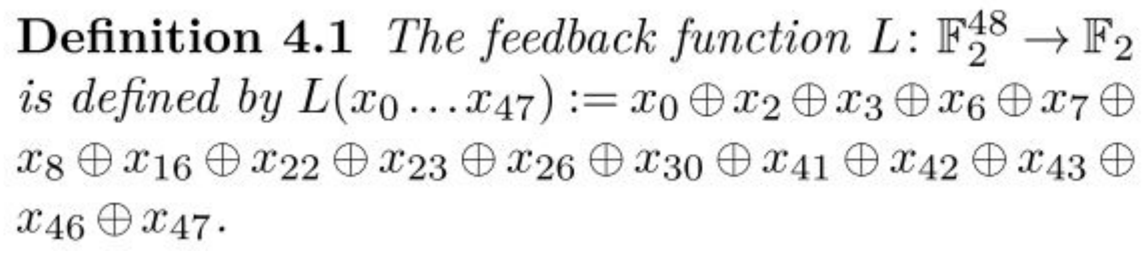
- Hitag2 consists of a 48-bit LFSR and a non-linear filter function f
- In the rolling code scheme, when a button is pressed it transmits the following message



- The initial state of the stream cipher consists of the 32-bit UID concatenated with the first 16 bits of the key k
- ctr is incremented and then initialization vector (iv) = ctr || btn is XORed with the last 32 bits of the key and shifted into the LFSR

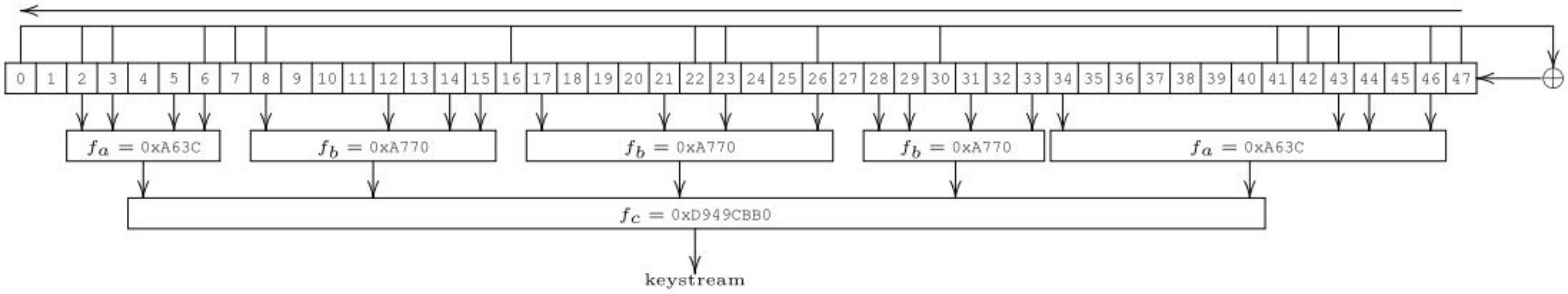
RKE Based on Hitag2 Cipher

- The cipher consists of a 48-bit Linear Feedback Shift Register and a non-linear filter function f
- Each clock cycle, twenty bits of the LFSR are input to f to generate a bit of the keystream
- The LFSR is then shifted one bit to the left, using the feedback polynomial to generate a new bit on the right



Definition 4.1 *The feedback function* $L\colon \mathbb{F}_2^{48} \to \mathbb{F}_2$ *is defined by* $L(x_0 \ldots x_{47}) := x_0 \oplus x_2 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{16} \oplus x_{22} \oplus x_{23} \oplus x_{26} \oplus x_{30} \oplus x_{41} \oplus x_{42} \oplus x_{43} \oplus x_{46} \oplus x_{47}$.

RKE Based on Hitag2 Cipher

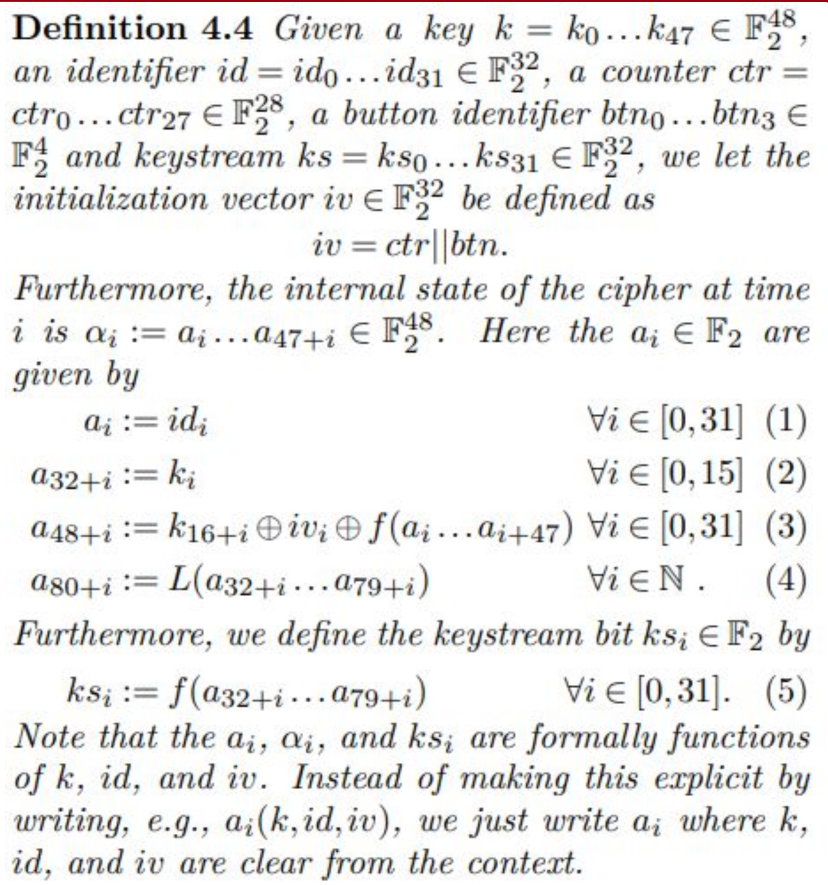


Rolling Code on Hitag2

- During the authentication protocol, the internal state of the stream cipher is initialized
- The internal state consists of the 32-bits UID concatenated with the first 16 bits of the key k
- The counter ctr is incremented and iv = ctr || btn is XORed with the last 32 bits of the key and shifted into the LFSR
- For this point, the next 32 bits of the keystream (output by the cipher ks) are sent as a proof of knowledge of the secret k

RKE Based on Hitag2 Cipher

- The next 32 bits of keystream, which are output by the cipher ks, are sent as proof of knowledge of the secret key k



Definition 4.4 *Given a key* $k = k_0 \dots k_{47} \in \mathbb{F}_2^{48}$, *an identifier* $id = id_0 \dots id_{31} \in \mathbb{F}_2^{32}$, *a counter* $ctr = ctr_0 \dots ctr_{27} \in \mathbb{F}_2^{28}$, *a button identifier* $btn_0 \dots btn_3 \in \mathbb{F}_2^4$ *and keystream* $ks = ks_0 \dots ks_{31} \in \mathbb{F}_2^{32}$, *we let the initialization vector* $iv \in \mathbb{F}_2^{32}$ *be defined as*

$$iv = ctr || btn.$$

Furthermore, the internal state of the cipher at time i *is* $\alpha_i := a_i \dots a_{47+i} \in \mathbb{F}_2^{48}$. *Here the* $a_i \in \mathbb{F}_2$ *are given by*

$$a_i := id_i \qquad \forall i \in [0,31] \quad (1)$$

$$a_{32+i} := k_i \qquad \forall i \in [0,15] \quad (2)$$

$$a_{48+i} := k_{16+i} \oplus iv_i \oplus f(a_i \dots a_{i+47}) \quad \forall i \in [0,31] \quad (3)$$

$$a_{80+i} := L(a_{32+i} \dots a_{79+i}) \qquad \forall i \in \mathbb{N}\,. \quad (4)$$

Furthermore, we define the keystream bit $ks_i \in \mathbb{F}_2$ *by*

$$ks_i := f(a_{32+i} \dots a_{79+i}) \qquad \forall i \in [0,31]. \quad (5)$$

Note that the a_i, α_i, *and* ks_i *are formally functions of* k, id, *and* iv. *Instead of making this explicit by writing, e.g.,* $a_i(k, id, iv)$, *we just write* a_i *where* k, id, *and* iv *are clear from the context.*

Correlation Attack on Hitag2

- The purpose of the attacker is to retrieve the key
- It requires a minimum of four rolling codes, but it would be faster and more precise by having more traces
- Rolling codes may have an arbitrary counter value, i.e., non consecutive
- However, this is good as it increases correlation
- We denote as <UID, iv^j, ks^j>, $j = 0,\ldots,n-1$, $n>3$, n authentication traces

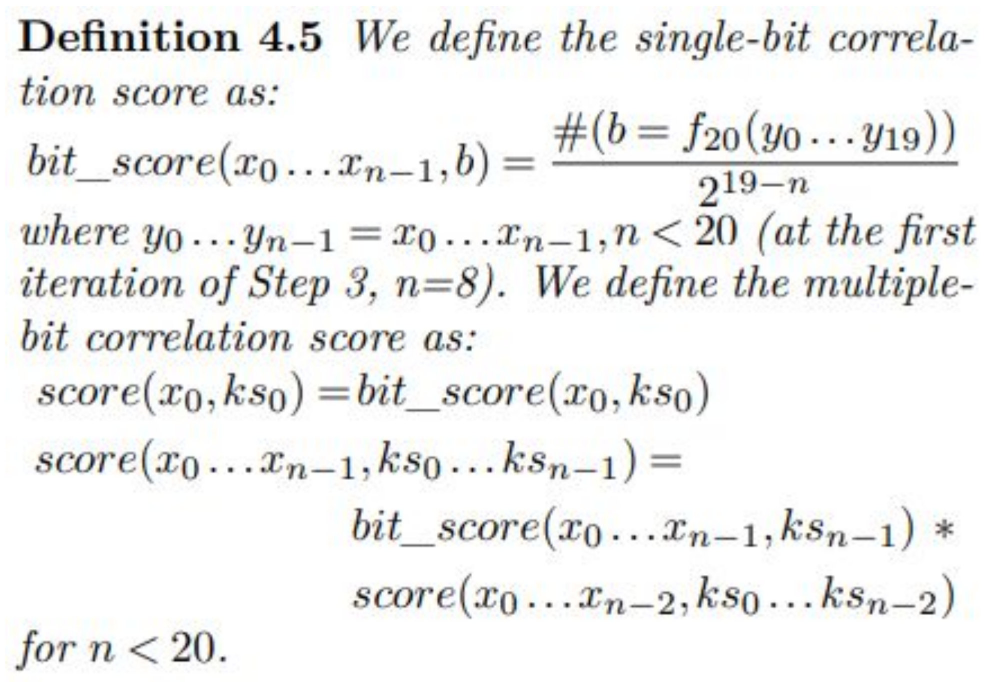
Correlation Attack on Hitag2

- The adversary first <u>guesses</u> a 16-bit window corresponding to LFSR stream bits $a_{32}, \ldots. a_{47} = k_0, \ldots, k_{15}$
- Together with UID, this gives the adversary $a_0, \ldots, a_{47}$, which is constant over traces
- The adversary can hence compute $b_0 = f(a_0, \ldots, a_{47})$
- The adversary shifts this 16-bit window to the left of the LFSR, until bits $a_{32}, \ldots, a_{47}$ are on the very left of the LFSR, i.e., the point where the cipher starts outputting ks

Correlation Attack on Hitag2

- The adversary computes a correlation score for this guess
- The window determines 8 input bits $x_0, \ldots, x_7$ to the filter function f_{20}, while the remaining 12 inputs remain unknown
- The correlation is taken as the ratio of those 2^{12} input values $x_8, \ldots, x_{19}$ that produce the correct keystream bit ks_0
- Shifting the window further to the left, the adversary can perform tests on multiple keystream bits ($ks_0, \ldots, ks_{15}$)

Correlation Attack on Hitag2



Definition 4.5 *We define the single-bit correlation score as:*

$$bit\_score(x_0 \dots x_{n-1}, b) = \frac{\#(b = f_{20}(y_0 \dots y_{19}))}{2^{19-n}}$$

where $y_0 \dots y_{n-1} = x_0 \dots x_{n-1}, n < 20$ *(at the first iteration of Step 3, n=8). We define the multiple-bit correlation score as:*

$$score(x_0, ks_0) = bit\_score(x_0, ks_0)$$

$$score(x_0 \dots x_{n-1}, ks_0 \dots ks_{n-1}) = bit\_score(x_0 \dots x_{n-1}, ks_{n-1}) * score(x_0 \dots x_{n-2}, ks_0 \dots ks_{n-2})$$

for $n < 20$.

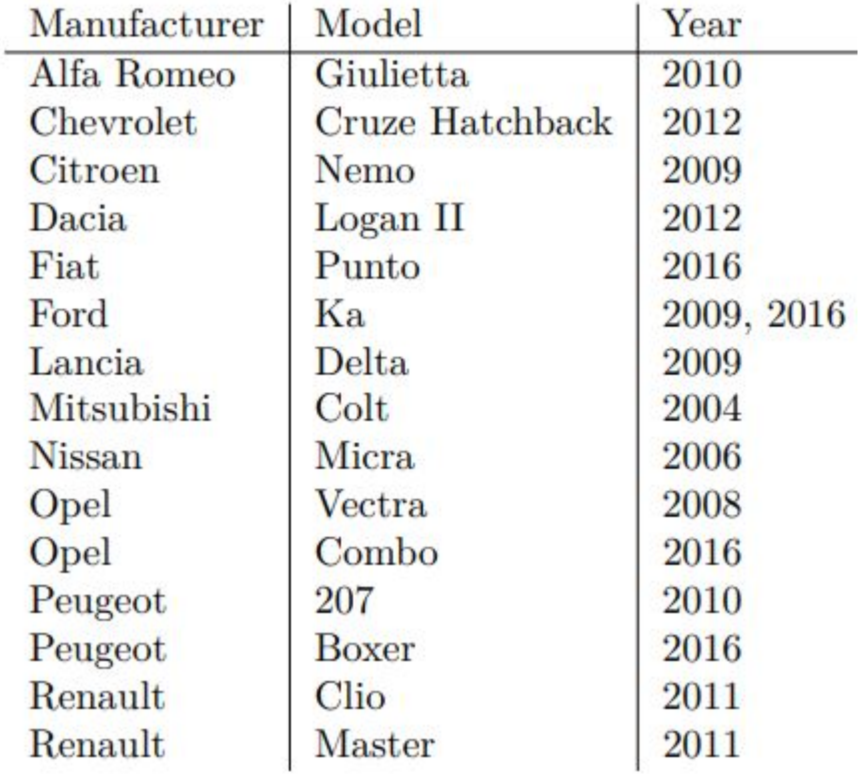
- The adversary assigns this guess the average score over all traces
- so far this scoring computation is independent of the value iv as it happens before iv gets to have any influence on it

Correlation Attack on Hitag2

- The adversary sorts guesses based on their score and stores them in a table, discarding guesses with lowest score if needed
- Experiments show that 400, 000 guesses are usually sufficient
- For each guess in the table, the adversary goes back to Step (1) and proceeds as before, except that she will now extend the window size by one guessing the next LFSR stream bit ($a_{48}, \dots, a_{51}$)
- The power of this attack comes from using the window on the right of the LFSR to compute the necessary keystream bits to correct the internal state

Results

- On average, the attack recovers the cryptographic key in approximately 1 minute of computation
- It requires between 4 and 8 rolling codes



Manufacturer	Model	Year
Alfa Romeo	Giulietta	2010
Chevrolet	Cruze Hatchback	2012
Citroen	Nemo	2009
Dacia	Logan II	2012
Fiat	Punto	2016
Ford	Ka	2009, 2016
Lancia	Delta	2009
Mitsubishi	Colt	2004
Nissan	Micra	2006
Opel	Vectra	2008
Opel	Combo	2016
Peugeot	207	2010
Peugeot	Boxer	2016
Renault	Clio	2011
Renault	Master	2011