



# TEMPA

## Demystifying Tesla's Bluetooth Passive Entry System



TRIFINITE

May Contain Hackers



trifinite.org

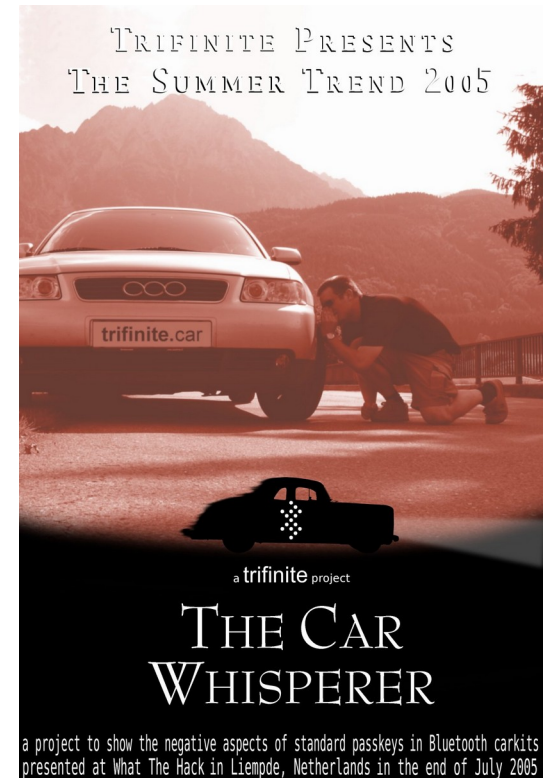
# About **trifinite.org**

- Group founded in August 2004 by
  - Collin Mulliner
  - Martin Herfurt (me)
- Pioneered in Bluetooth (Classic) Security
- Participation in tech. Testing events organized by the Bluetooth SIG – helping vendors with security
- Webpage renewed in 2022 !

# About Me

- Martin Herfurt
- Living in Salzburg/Austria
- Regular participant/speaker at C3 since 1998
- Author of App „Tesla Radar“ ([teslaradar.com](https://teslaradar.com))
- Owner of a black 2019 Tesla Model 3

# Memory Lane – WTH2005



# Be careful using Tesla PAAK/NFC?



# Project TEMPA – Investigating BLE

- Technical Background about Tesla's Passive Entry system
  - Found on all Tesla Models 3/Y
  - Found on Tesla Model S/X 2021+
  - About **2 million+** vehicles to date
- Identifying/Tracking vehicles
- Exchanging messages with vehicles via Bluetooth LE
- Possible impacts on vehicle's security

# Project TEMPA – Investigating BLE

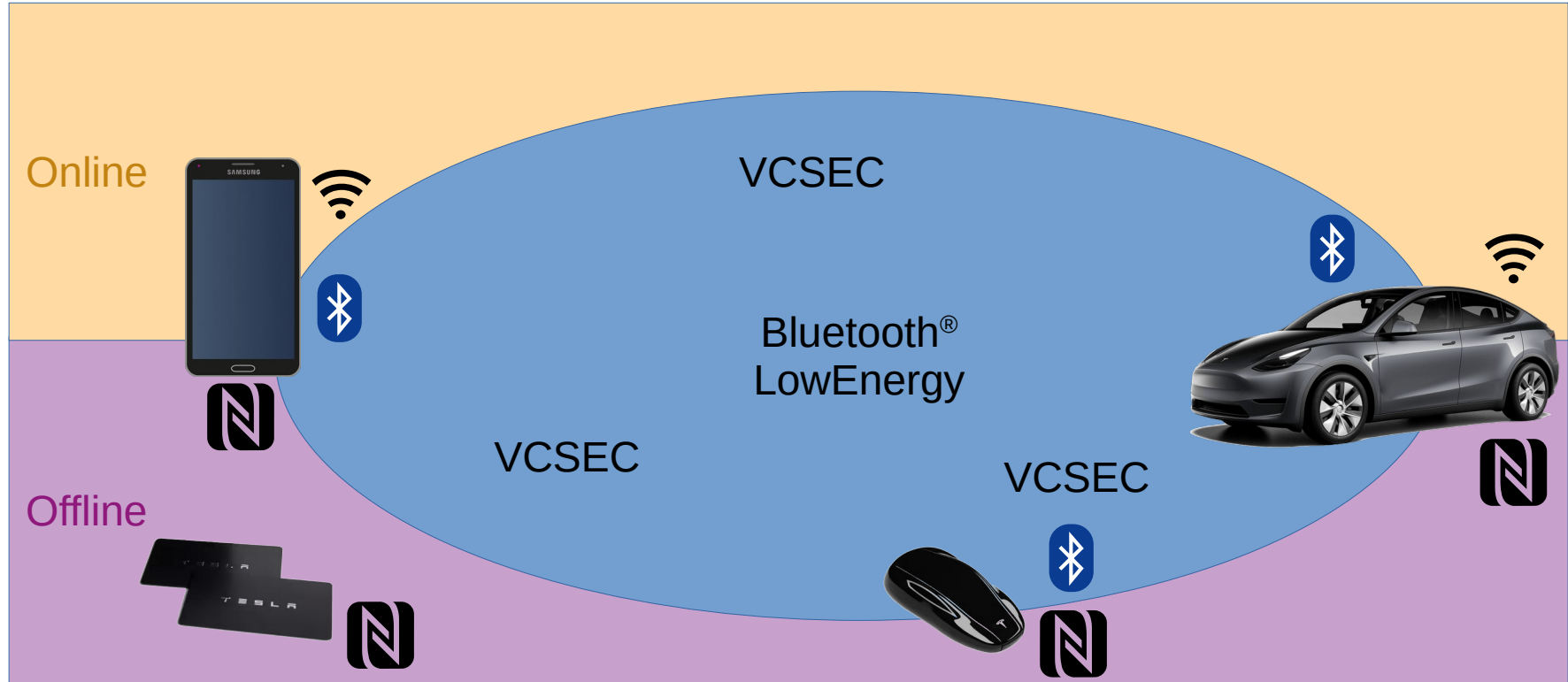
- Some of the things have been (partially) fixed and improved during the time of this research
- Findings reverse-engineered from the official Tesla app for Android and from observed messages
- Research started in 06/2019
- Research intensified in 06/2021 with VCSEC

# PhoneKey

- Tesla's BLE-based Passive Entry System
- Introduced with Model 3 in 2018
- Idea: The owner's phone replaces the car key/fob
- Now also in use in Model Y and 2021 Facelift S/X
- Very likely to be part of future Tesla Models



# Ways to Unlock a Tesla (S/3/X/Y)



# Tesla (S/3/X/Y) Unlock Methods (1)

- NFC-Card
  - Owners get two whitelisted NFC-Cards with car
  - Different form-factors sold on Internet (e.g. KeyRing)
- Usage
  - card is held to driver-side B-pillar to unlock
  - card is held to middle-console to drive/authorize
  - No passive entry!

# Tesla (S/3/X/Y) Unlock Methods (2)

- PhoneKey
  - Feature of the official iOS/Android app
  - Based on Bluetooth LE (BLE) / NFC
  - Allows „passive entry“ and basic security functions
- Usage
  - Phone is carried by owner
  - Authorization to unlock/drive via BLE / NFC / Online

# Tesla (S/3/X/Y) Unlock Methods (3)

- KeyFob
  - Small Device (sold extra for 160€)
  - Based on Bluetooth LE (BLE)
  - Allows “passive entry“ (in later versions (starting with V. P60))
- Usage
  - Keyfob is carried by owner
  - Authorization to unlock/drive via BLE / NFC
  - Authorization via tap on B-pillar or middle console

# Twitter Poll (1)



**Tesla Radar**  
@TeslaRadar



How do you unlock your Tesla Model 3/Y? ....Please RT

- Key Fob
- Phone Key
- NFC Card

# How does PhoneKey BLE work?

1. Smartphone with app finds vehicle

- Smartphone identifies vehicle
- Smartphone connects to vehicle

2. App on smartphone communicates with car

3. Car (un)locks / starts / opens etc.

# 1. Smartphone with app finds vehicle

- Car advertises GATT services via BLE (Peripheral)
  - Name (standard)
  - To Vehicle (Tesla)
  - From Vehicle (Tesla)
- manufacturer data has iBeacon structure
  - UUID, major ID, minor ID
- There used to be four visible beacons per vehicle!

# BLE Advertisement

- Manufacturer-Data (uses iBeacon format)
  - enables iPhone background vehicle detection
- UUID
  - 74278BDA-B644-4520-8F0C-720EAF059935
- Major/Minor ID (2 bytes each)
  - Random values (collisions possible but unlikely)



## 2. Smartphone identifies vehicle

- BLE device name(s)
  - Structure: **S<8 bytes in hex>C** (D,P,R)
  - Guess: C(enter) D(river side) P(assenger side) R(ear)
- Major/Minor ID (mainly for iPhone)
- <8 bytes in hex>
  - Seemed random at first
  - Unique to vehicle

# Unique to vehicle!

- Always turned on
- Visible to anyone with BLE radio
- Privacy issue!
- Stalking
  - Compare: Privacy discussion concerning Apple AirTag (AirTag even randomizes ID)
  - Similarities to Tesla's PhoneKey

# Correspondence with Tesla (in 2019)

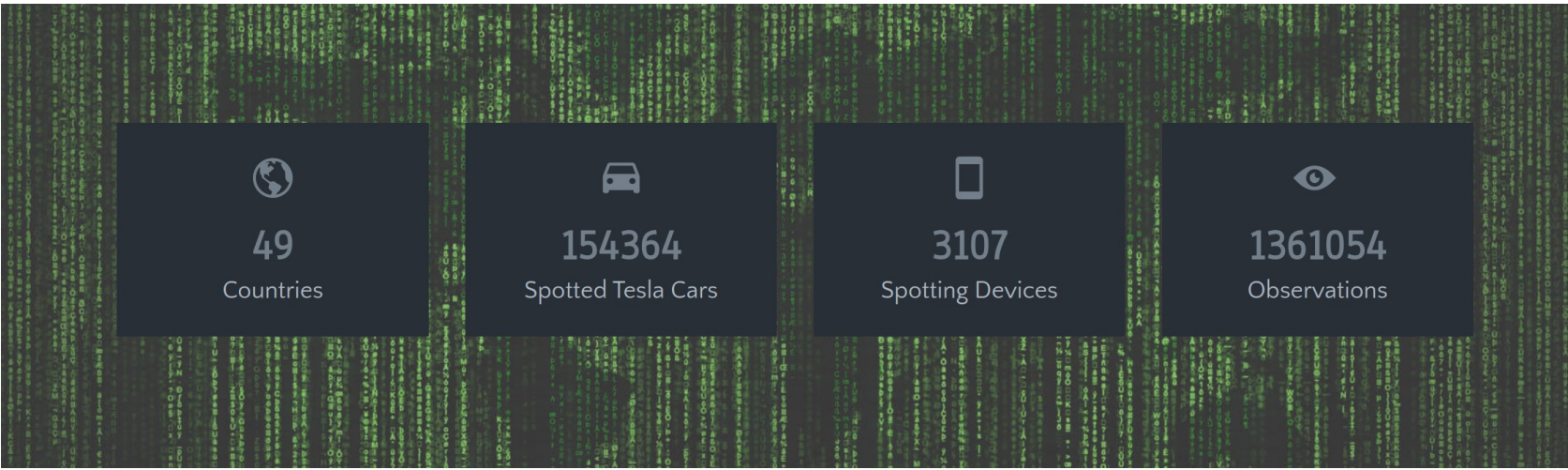
- Complaint concerning unique identifier
  - Does not change over time
  - Owners cannot turn signal off
- Tracking cars/owners becomes possible
- Tesla acknowledged this fact and wrote that this situation and its implications are accepted risks/circumstances

# Tesla Radar




- Project first published in August 2019 (Chaos Camp)
- With no understanding of all the inner workings
- Android App (available in Play Store)
- Crowdsourcing vehicle discovery
- First: Showcasing privacy issues
- Then: Game for the Tesla fan community with rankings etc.
- And: data-collection for research

# Tesla Radar



[www.teslaradar.com](http://www.teslaradar.com)

Tesla Radar		
Rankings are based on activities of the last 30 days		
BY DEVICE	BY COUNTRY	BY REGION
1	Unnamed Device Google, Pixel 5a from California	35314 
2	Rickman Google, Pixel 6 Pro from Florida	14759 
3	Kowa OA-5599 samsung, SM-A217F from Hesse	14210 
4	Kristian M3P samsung, SM-A505FN from Oslo	8884 
5	Google, Pixel 4 from Washington	6836 
6	snow samsung, SM-A125U from Texas	6068 
7	Renegade samsung, SM-N950U from Florida	5511 
8	trifinite.org OnePlus, LE2123 from Salzburg	5178 
9	Unnamed Device samsung, SM-G781U from Colorado	4382 
10	Unnamed Device samsung, SM-N970F from Auvergne-Rhône-Alpes	3853 
11	djoul samsung, SM-S901B from Vaud	3766 
12	skatebambi OnePlus, NE2213 from Skåne	3667 
13	CarinasTeslatray samsung, SM-G780F from Viken	3240 

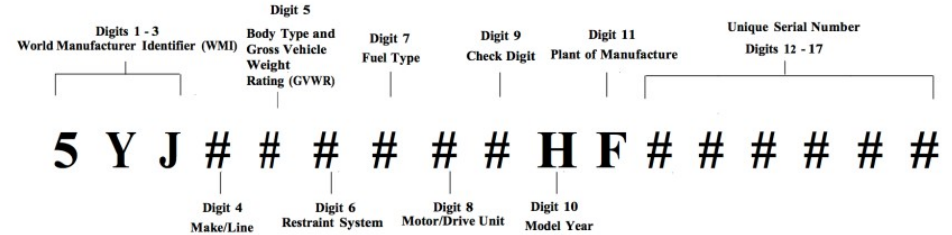


Martin Herfurt @ MCH2022



# VIN Structure (17 Digits)

- Manufacturer ID
- Model Type
- Manufacturing Plant
- Manufacturing Year
- Serial Number
- Check-Digit



- Standard/Dual/Performance
- LHD/RHD?
- Battery Type

# VIN Detection

- 16 character String used as part of the iBeacon name (8 hex-encoded bytes)
- Created from SHA1-hash over Vehicle VIN
  - VIN Identifier
- Reverse ID->VIN via special Rainbow-Table
- Used for identifying vehicles in Owner-List

# VIN Index

- All possible Tesla VINs (with PhoneKey)
  - Research about production numbers in different plants
  - Research on web-pages for used Teslas
- Size: 217140601 objects ~ 20GB
- Hit-Rate: 98.75%
- Used for model-detection in TeslaRadar app



# Wardriving 2.0 (BLE)



# Video: The Tesla Parking Lot Job



<https://youtu.be/eDbSzVTYqBY>

# Correspondence with Tesla (in 2021)

- Bug-Bounty request concerning relay attack
  - Attackers can open car (and maybe steal it or at least some parts / stuff)
- Tesla acknowledged this fact and wrote that this is “a known limitation“ of the Phone Key Feature and that people should use PIN2Drive
- pwn2own: Not interested in Relay-Attacks!

# Twitter Poll (2)



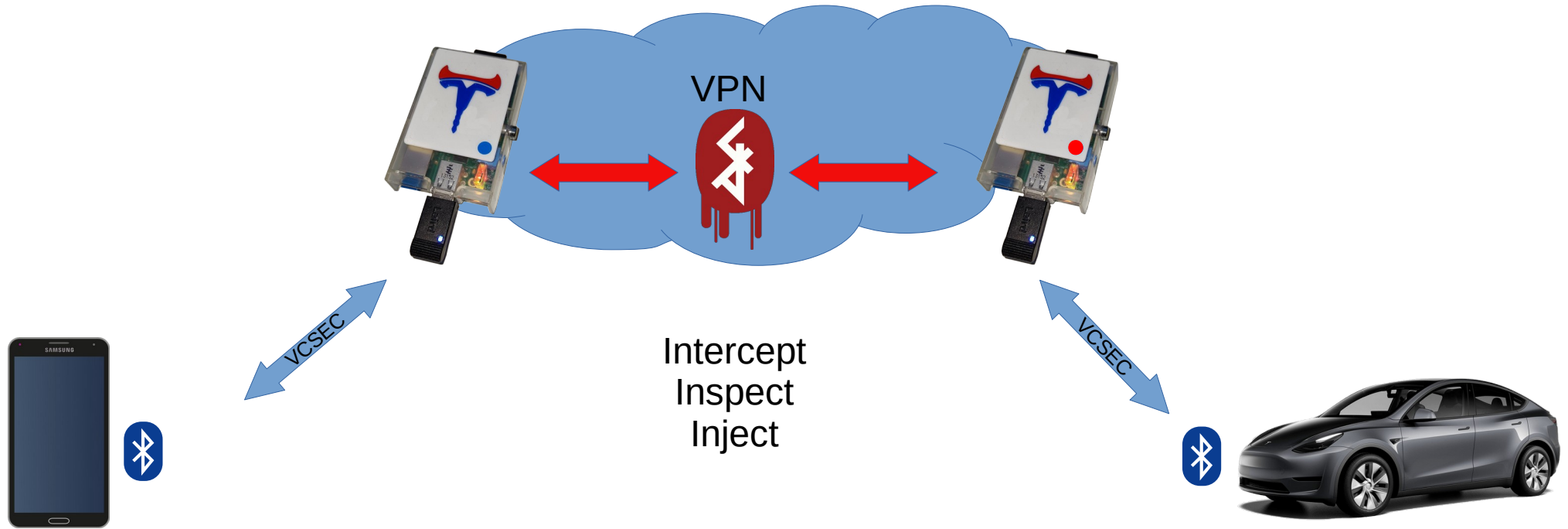
**Tesla Radar**  
@TeslaRadar



As a Tesla Owner: Which of these features are active in your car(s)? [#Tesla](#) [#Poll](#)  
Please RT for reach!

- None
- Sentry Mode
- Pin to Drive
- Both

# MitM Relay Attack from Video



# BLE-Endpoints (Characteristics)

- Service
  - 00000211-B2D1-43F0-9B88-960CEBF8B91E
- Characteristic: **To Vehicle** (write)
  - 00000212-B2D1-43F0-9B88-960CEBF8B91E
- Characteristic: **From Vehicle** (read/subscribe)
  - 00000213-B2D1-43F0-9B88-960CEBF8B91E

# VCSEC Protocol

- Based on Google Protocol Buffers (protobuf), later Square/Wire
  - Perfect match for limited bandwidth in BLE
- Defines interaction between Security Devices and the Vehicle
- Deducted Use-cases
  - PhoneKey
  - KeyFob
  - TP (Tire Pressure Subsystem)
  - Backend-Communication (?)
  - Maybe even more use-cases



# VCSEC History (1)

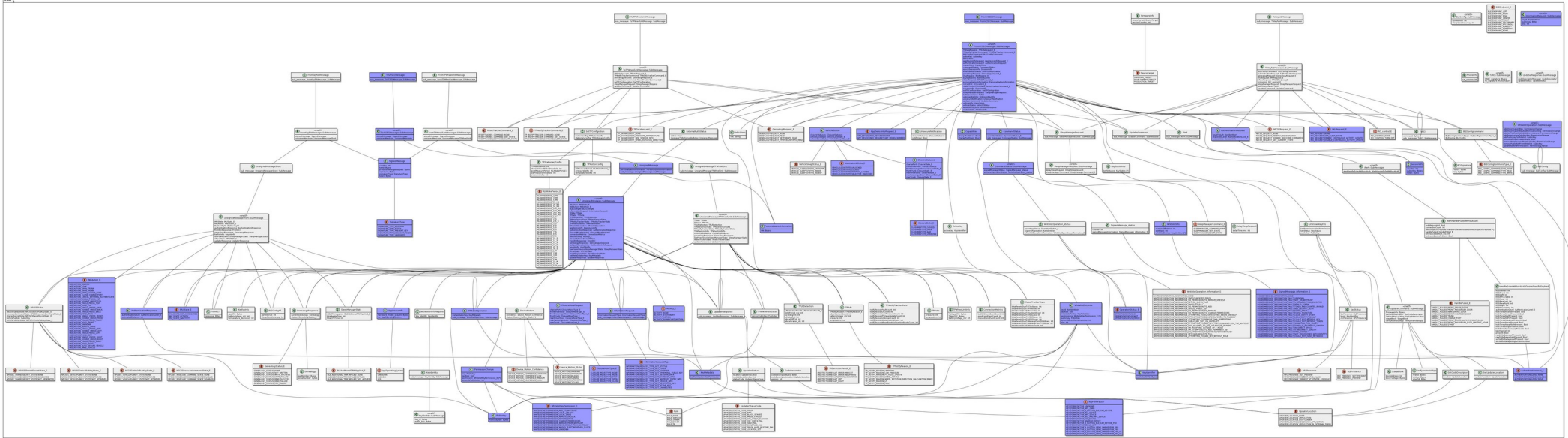
- Introduced in App V3.3.5-344 (April 2018)
- App Version 3 makes use of Google/Protobuf
  - Extractable with pbtk tool
- Current App Version 4 uses Square/Wire
  - Very similar output – but no extraction with pbtk
  - Custom script to extract proto-file from POJOs from decompiled Android app (experimental)
  - Further obfuscation of VCSEC starting with app 4.9.0



# VCSEC History (2)

- Introduced in App V3.3.5-344 (April 2018)
- Four major iterations so far
  - VCSEC.proto v1 (2018-04-12 - V3.3.5-344)
    - 22 Messages and 9 Enums
  - VCSEC.proto v2 (2019-11-28 - V3.10.2-388)
    - 53 Messages and 27 Enums
  - VCSEC.proto v3 (2020-06-21 - V3.10.6-407)
    - 62 Messages and 32 Enums
  - VCSEC.proto v4 (2022-05-13 – V4.8.1-1032)
    - 77 Messages and 45 Enums

# VCSEC – App Version 4.8.1 (05/22)



77 Messages

45 Enumerations

Colored entities are referenced in decompiled BLE plugin code

# (De)Serializing messages via shell

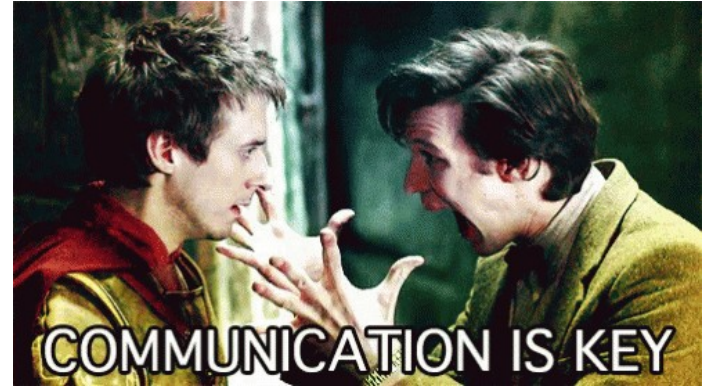
- Serialized messages are often represented as hex-encoded strings (e.g. 00040a021001)
- Size prefix (2 octets) – not compatible with protoc
- Shell scripts in Tesla VCSEC Archive (github)

```
$> cat message.txt | protoc --encode=VCSEC.ToVCSECMessage  
-I . VCSEC.proto | xxd -p -l 100
```

```
$> cat message.hex | xxd -r -p | protoc --  
decode=VCSEC.FromVCSECMessage VCSEC.proto
```

# Whitelisted Keys (InformationReq)

```
whitelistInfo {
  numberOfEntries: 9
  whitelistEntries {
    publicKeySHA1: "$\206\202d"
  }
  whitelistEntries {
    publicKeySHA1: "S` \031\375"
  }
  ...
  whitelistEntries {
    publicKeySHA1: "\221=\210\205"
  }
  whitelistEntries {
    publicKeySHA1: ";\223\300\027"
  }
  slotMask: 511
}
```



# Service Key (Most likely NFC)

```
whitelistEntryInfo {
  keyId {
    publicKeySHA1: "$\206\202d"
  }
  publicKey {
    PublicKeyRaw: "\004\333\243\225\271\237\217:\\"022*yCX\000\3741_\
357b\261w\216\315\\367\313j\037\201wH\006q\204\350\264V\025\0054Sc\
305L\356\234\216\343\nZ\033\005>/L\032\214\373W7Q\322\255\244"
  }
  keyRole: ROLE_SERVICE
}

sessionInfo {
  token: "\256\006Mj\270\237\277Y\310\223\023w\235\221<I\270\375,5"
  publicKey: "\004M)d\2136\372\201J\rh\253\354\220cZ\307_\
276\320\3568\212G\016\202f\223\025m\267\360\241!}\232\372vH\304_\
3532\244\023\016@1hbA\315\276g(+22q\235\3663R.\367"
}
```

# Example (NFC)

```
whitelistEntryInfo {
  keyId {
    publicKeySHA1: "S` \031\375"
  }
  publicKey {
    PublicKeyRaw: "\004\323\332\321U-\320;=\215\014\331\025)C\303c*/\\\
024\016\007\207\347dd\r\216Q5\342v\362\360\2
67\336{\224\354R\376\332\203\243Z\377_\3267D\3577\215V\343P\315A\306\3603}\3027"
  }
  metadataForKey {
    keyFormFactor: KEY_FORM_FACTOR_NFC_CARD
  }
  slot: 1
  keyRole: ROLE_OWNER
}

sessionInfo {
  token: "^v\355*\345\374#\242Y\374\277N\277\347\202\303\355\265\t\177"
  publicKey: "\004M)d\2136\372\201J\rh\253\354\220cZ\307_\276\320\3568\212G\016\202f\
223\025m\267\360\241!}\232\372
vH\304_\3532\244\023\016@1hbA\315\276g(+22q\235\3663R.\367"
}
```

# Example (PhoneKey)

```
whitelistEntryInfo {
  keyId {
    publicKeySHA1: "U\2346\373"
  }
  publicKey {
    PublicKeyRaw: "\004>\347\2741[\240\372\030\334h\017\034Z\251\304o\272\202$\320\010N3\374\005\362\032\316#\}\323\270\241\262\ '\337\375\243\200\316d\245\007\337\266F\017\036\335\201pM\017\254S\022\274\200\320W\210\307\3230"
  }
  metadataForKey {
    keyFormFactor: KEY_FORM_FACTOR_ANDROID_DEVICE
  }
  slot: 4
  keyRole: ROLE_OWNER
}

sessionInfo {
  token: "h\234*\257\022\234o\375\223+\367}\330\030a\021r)/\301"
  counter: 44
  publicKey: "\004M)d\2136\372\201J\rh\253\354\220cZ\307_\276\320\3568\212G\016\202f\223\025m\267\360\241!\}\232\372vH\304_\3532\244\023\016@1hbA\315\276g(+22q\235\3663R.\367"
}
```

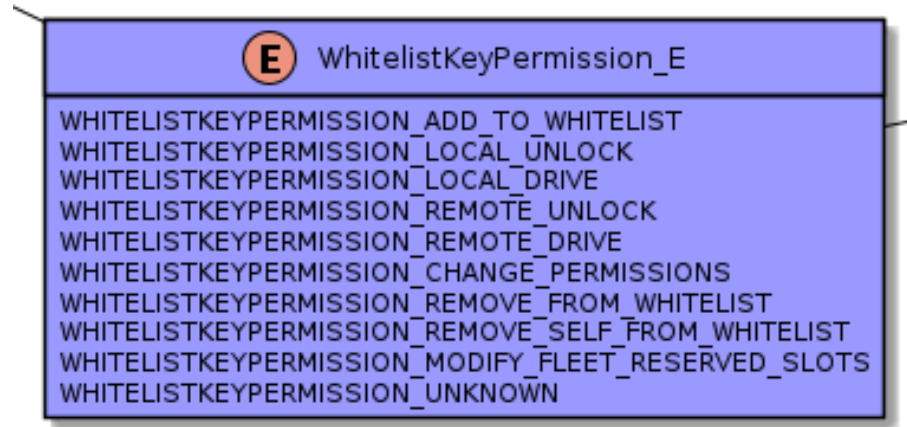
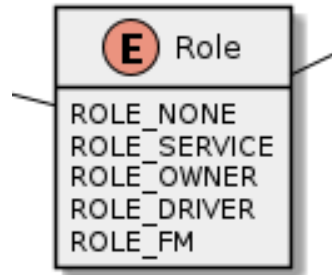
# Example (KeyFob)

```
whitelistEntryInfo {
  keyId {
    publicKeySHA1: "\007\273\036o"
  }
  publicKey {
    PublicKeyRaw: "\004\005\375\367GJ\235\32235\253\255\207\007HL\"|\177S\225=]\
016\211\237\377Rs)v\370\274\307@#\016]$\276\342\314\024\261\373\2067\342\316\337TA\
262\017\330\004\353\353J\337\307\265{\007v\002"
  }
  metadataForKey {
    keyFormFactor: KEY_FORM_FACTOR_3_BUTTON_BLE_CAR_KEYFOB_P60
  }
  slot: 5
  keyRole: ROLE_OWNER
}

sessionInfo {
  token: "\322\304J\250\277>\036i(\0229\022{\255$\323v\027\\\245"
  counter: 2479
  publicKey: "\004M)d\2136\372\201J\rh\253\354\220cZ\307_\276\320\3568\212G\016\202f\
223\025m\267\360\241!}\232\372vH\304_\3532\244\023\016@1hbA\315\276g(+22q\235\3663R.\
367"
}
```



# Roles and Permissions



FM = Fleet Manager (?)

# Service Key Permissions

WHITELISTKEYPERMISSION\_ADD\_TO\_WHITELIST  
WHITELISTKEYPERMISSION\_LOCAL\_UNLOCK  
WHITELISTKEYPERMISSION\_LOCAL\_DRIVE  
WHITELISTKEYPERMISSION\_REMOTE\_UNLOCK  
WHITELISTKEYPERMISSION\_REMOTE\_DRIVE  
WHITELISTKEYPERMISSION\_CHANGE\_PERMISSIONS  
WHITELISTKEYPERMISSION\_REMOVE\_FROM\_WHITELIST  
WHITELISTKEYPERMISSION\_REMOVE\_SELF\_FROM\_WHITELIST  
WHITELISTKEYPERMISSION\_MODIFY\_FLEET\_RESERVED\_SLOTS

# FromVCSEC



- All VCSEC messages that originate from Vehicle
- Most frequent messages:
  - vehicleStatus
  - authenticationRequest
  - commandStatus
- Observation: No cryptographically protected messages from vehicle!

# FromVCSEC – Examples (1)

001c1a1a12160a14d658de76f3a930b63410c6b6382a554781979d041802

```
--- FromVCSECMessage ---
authenticationRequest {
  sessionInfo {
    token: "\326X\336v\363\2510\2664\020\306\2668*UG\201\227\235\004"
  }
  requestedLevel: AUTHENTICATION_LEVEL_DRIVE
}
-----
```

# FromVCSEC – Examples (2)

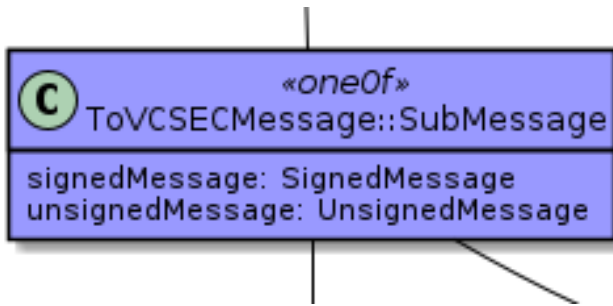
00072205120308de15

```
--- FromVCSECMessage ---  
commandStatus {  
  signedMessageStatus {  
    counter: 2782  
  }  
}  
-----
```

00040a021001

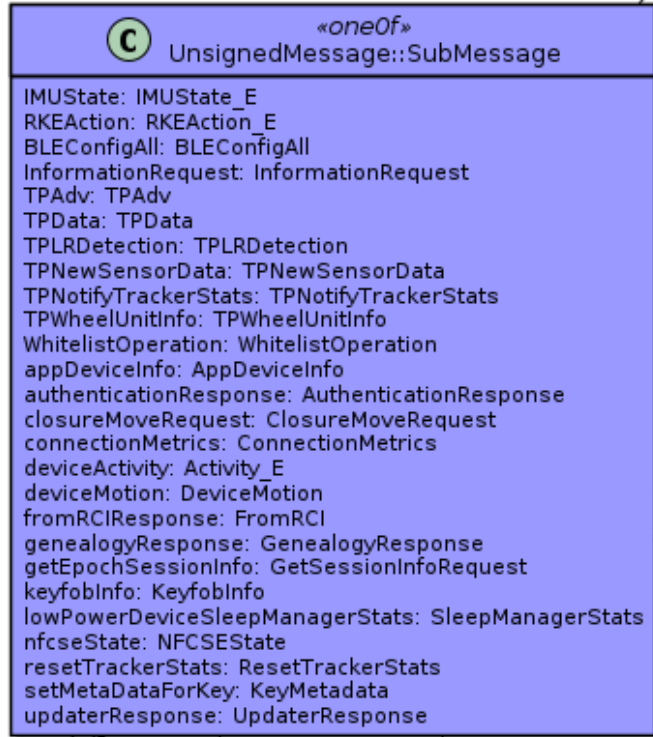
```
--- FromVCSECMessage ---  
vehicleStatus {  
  vehicleLockState: VEHICLELOCKSTATE_LOCKED  
}
```

# ToVCSEC



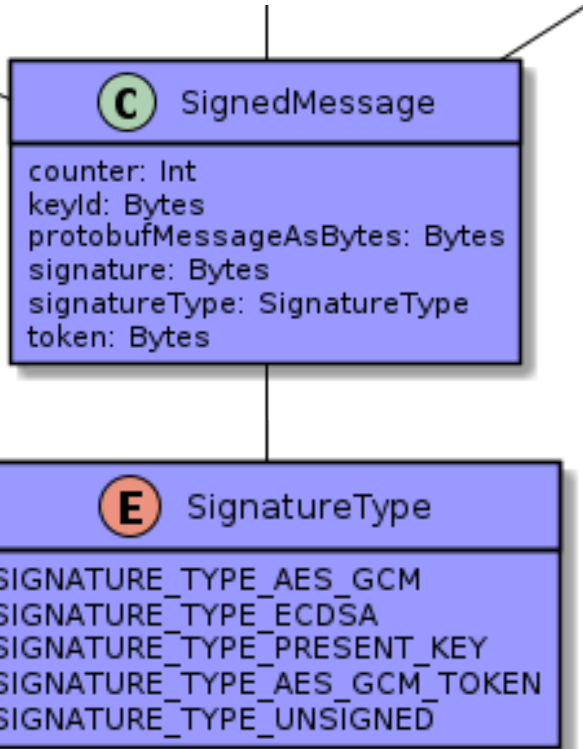
- All VCSEC messages that are sent to vehicle
- Depending on use-case:
  - unsignedMessage
    - Not cryptographically protected
  - signedMessage
    - Crypto: AES-GCM (AEAD)

# unsignedMessage



- Used for messages **without** direct security context
- Used as encapsulating message for signedMessage cryptograms

# signedMessage



- Used for messages **with** direct security context
- Used as encapsulating message for signedMessage cryptograms
- IMUState: used for mitigating relay attack(!?)



# Cryptographic Keys

- VCSEC uses asymmetric encryption based on ECC Keypairs
  - Based on prime256v1 curve

```
$> openssl ecparam -name prime256v1 -genkey -noout -out created_key.pem
```
- Shared secret is derived used via Diffie-Hellman key exchange
  - 128 bit
- Used for authentication/encryption

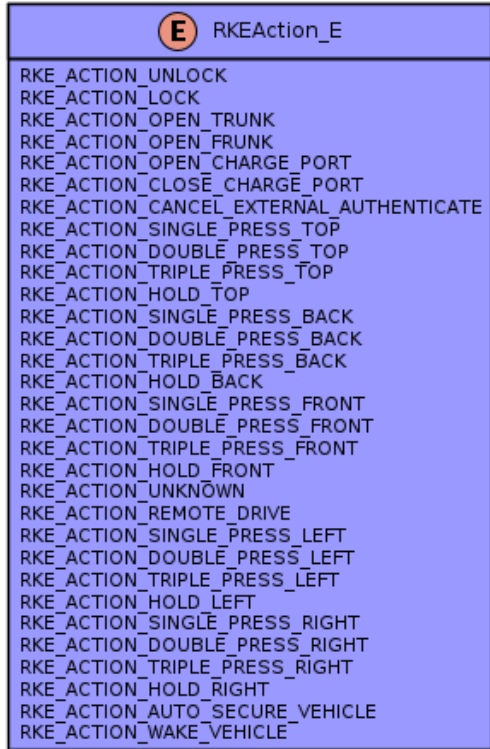
# Signed Messages

- Galois Counter Mode with Associated Data  
AES-GCM AEAD
  - Intends to prevent replay attacks (counter)
    - Rolling Code
  - Additional Data (session token data in requests)
    - Works as challenge for the correct response
    - Also intends to prevent replay attacks

# GCM Structure (Tesla)

- SharedSecret 16 octets
- Invocation-Counter only 4 octets (not 8) (counter)
- Signature/Tag (GMAC) 16 octets
- Additional Data (optional) 20 octets (session token)
  - **SIGNATURE\_TYPE\_AES\_GCM**
  - **SIGNATURE\_TYPE\_AES\_GCM\_TOKEN**
  - **SIGNATURE\_TYPE\_PRESENT\_KEY**

# RKAction\_E



- Used for control commands in app/fob
- Is encapsuled in unsignedMessage before encryption

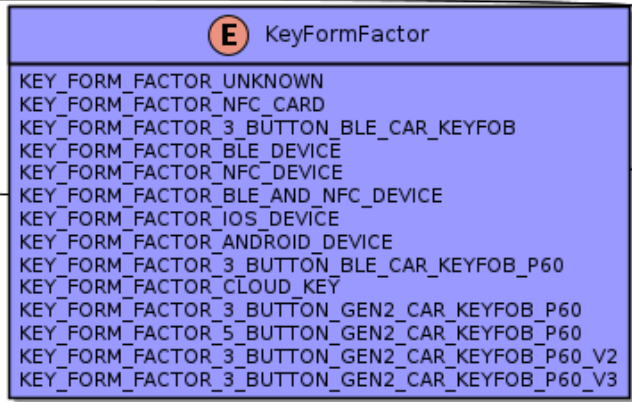
# SignedMessage\_information\_E

- What could possibly go wrong with encryption?
- Some attacks play with these

**E** SignedMessage\_information\_E

```
SIGNEDMESSAGE_INFORMATION_NONE
SIGNEDMESSAGE_INFORMATION_FAULT_UNKNOWN
SIGNEDMESSAGE_INFORMATION_FAULT_NOT_ON_WHITELIST
SIGNEDMESSAGE_INFORMATION_FAULT_IV_SMALLER_THAN_EXPECTED
SIGNEDMESSAGE_INFORMATION_FAULT_INVALID_TOKEN
SIGNEDMESSAGE_INFORMATION_FAULT_TOKEN_AND_COUNTER_INVALID
SIGNEDMESSAGE_INFORMATION_FAULT_AES_DECRYPT_AUTH
SIGNEDMESSAGE_INFORMATION_FAULT_ECDSA_INPUT
SIGNEDMESSAGE_INFORMATION_FAULT_ECDSA_SIGNATURE
SIGNEDMESSAGE_INFORMATION_FAULT_LOCAL_ENTITY_START
SIGNEDMESSAGE_INFORMATION_FAULT_LOCAL_ENTITY_RESULT
SIGNEDMESSAGE_INFORMATION_FAULT_COULD_NOT_RETRIEVE_KEY
SIGNEDMESSAGE_INFORMATION_FAULT_COULD_NOT_RETRIEVE_TOKEN
SIGNEDMESSAGE_INFORMATION_FAULT_SIGNATURE_TOO_SHORT
SIGNEDMESSAGE_INFORMATION_FAULT_TOKEN_IS_INCORRECT_LENGTH
SIGNEDMESSAGE_INFORMATION_FAULT_INCORRECT_EPOCH
SIGNEDMESSAGE_INFORMATION_FAULT_IV_INCORRECT_LENGTH
SIGNEDMESSAGE_INFORMATION_FAULT_TIME_EXPIRED
SIGNEDMESSAGE_INFORMATION_FAULT_NOT_PROVISIONED_WITH_IDENTITY
SIGNEDMESSAGE_INFORMATION_FAULT_COULD_NOT_HASH_METADATA
```

# RKAction\_E



- Used for control commands in app/fob
- Is encapsuled in unsignedMessage before encryption

# SignedMessage Example

0018ea02150a1308f4051801320c0804180120772877306d386f

--- ToVCSECMMessage ---

```
signedMessage {  
  protobufMessageAsBytes: "h\001\251\242"  
  signatureType: SIGNATURE_TYPE_AES_GCM_TOKEN  
  signature: "}\246\023E\306\257/\274\037\026\032\375#\355\222"  
  keyId: "\'\365\030\021"  
  counter: 2781  
}
```

-----

# Key Enumeration (unrestricted)

- Formfactors (what kind of devices?)
- Active Keys (how many users/keys)
- Counters (which key is used over time?)
- Service Key ID (maybe service region?)
  - Two alternating keys identified (Europe?)



# Whitelisting Keys

- Process requires key with OWNER\_ROLE & NFC
- Max. 19 keys can be enrolled per vehicle
  - More keys / slots / channels possible?
  - `WHITELISTKEYPERMISSION_MODIFY_FLEET_RESERVED_SLOTS`
  - Fleet mgmt is a business feature introduced in 02/22
- Whitelisted keys are referenced with keyID
  - KeyID = first 4 bytes of SHA1(public key)

# Process: Whitelisting a Key (1)

- Log in to Tesla Account
- Get assigned Vehicle VIN(s) from Owner-API
- Get VIN Identifier
  - SHA1 over VIN and get first 8 bytes
- Find Vehicle
- Begin Whitelisting Process

# Process: Whitelisting a Key (2)

- Send:

`INFORMATION_REQUEST_TYPE_GET_EPHEMERAL_PUBLIC_KEY`

- Receive: Vehicle's public key

- Send:

`INFORMATION_REQUEST_TYPE_GET_WHITELIST_INFO`

- Receive: Number of currently whitelisted keys

# Process: Whitelisting a Key (3)

- Generate WhitelistOperation message
  - Use your previously generated keypair (ECC prime256v1)
- Send WhitelistOperation (wrapped in SignedMessage) with SignatureType  
`SIGNATURE_TYPE_PRESENT_KEY`
- Tap NFC-Key for Authorization (Fascia or B-Pillar)
- Receive: WhitelistOperationStatus

# Process: Whitelisting a Key (4)

- Send:

`INFORMATION_REQUEST_TYPE_GET_WHITELIST_INFO`

- Receive: Number of currently whitelisted keys
- Verify that your new key is in the list
  - KeyID: First 4 Bytes from SHA1 over your public key
- Start using your key

# Authorization Timer (130 seconds)

- Introduced in August 2021
  - <https://bit.ly/3anslsl> (driveteslacanada.ca)
- For more convenience with NFC-KeyCard
- No extra NFC swipe is required during this time
  - Allows starting car
  - **Allows whitelisting a key**

# Gone in under 130 seconds

*PROJECT TEMPA PRESENTS:*  
**GONE IN UNDER**  
**130**  
**SECONDS**

**NIKOLA'S RAGE**



**K.I.T.T. EVATHEKÄFER EVIL TESLAKEE**

trifinite.org

AT CONFERENCES FROM 06/2022



<https://youtu.be/yfG4JS71eUY>

# Owning a key allows

- Unlocking/Locking the vehicle
- Drive the vehicle
  - PIN2Drive as recommended mitigation
  - PIN2Drive: 4-digit PIN has to be entered in order to drive



# NOT a numbers game - Bypass2Drive

## NOT a Numbers Game - Bypass2Drive



<https://youtu.be/vWM98f3-vvc>

# Tesla's Broken Trust Model



# HOWTO: Emulating a Tesla Vehicle

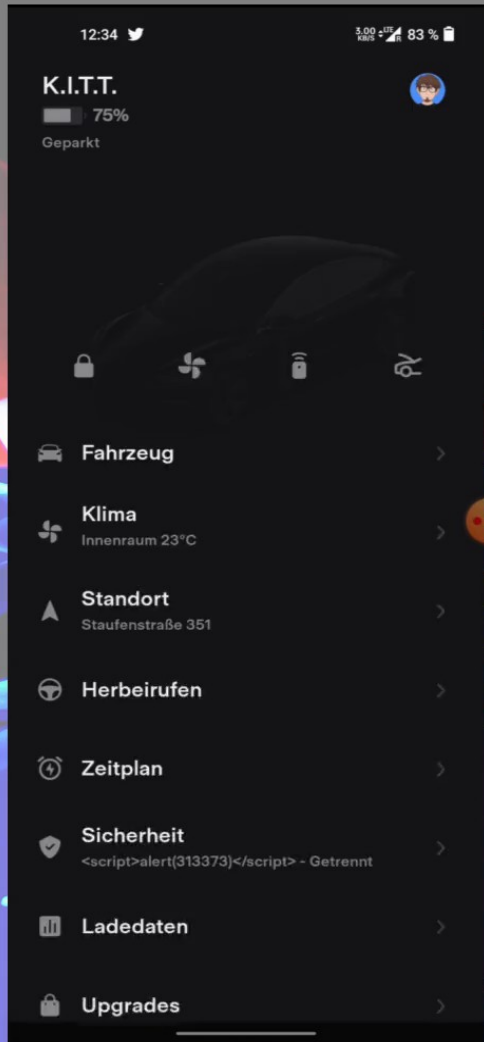
- Expose iBeacon structure
- Use of EIR Advertising (not with bluez dbus-api)
  - EIR: 0201061aff4c00021574278bdab64445208f0c720eaf05993500001337c5  
green: length, blue: iBeacon UUID, orange: iBeacon major, purple: iBeacon minor
  - ScanResponse: 030222111309533066373838356332616631613665663943  
green: length, blue: Vehicle Name (VIN identifier)
- Implement Service with *FromVehicle* and *ToVehicle* Characteristics
- Tool on github: temporary

# Key Drop Attack

- PhoneKey App sends signed message
- Attacker answers for vehicle:
  - `SIGNEDMESSAGE_INFORMATION_FAULT_NOT_ON_WHITELIST`
- PhoneKey app invalidates whitelisted Key
- User is locked out (and has to use NFC)

# Key Enrollment vs. Key Restoration

- Keydrop attack: Phone holds on to key
- Restoration: Vehicle „remembers key“
  - Restoration process also possible via B-Pillar
- Full Whitelisting process only for keys that are not known to vehicle

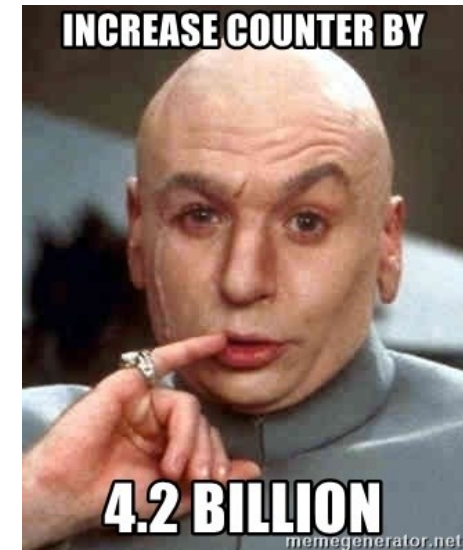


```
root@mate temporary> ./temporary.py kitt.json
```

# Crypto Counter Confusion Attack



- PhoneKey App sends signed message
- Attacker answers for vehicle:  
`SIGNEDMESSAGE_INFORMATION_FAULT_IV_SMALLER_THAN_EXPECTED`
- PhoneKey app will ask vehicle for correct counter value
- Attacker answers for vehicle with maximum value of 32-Bit integer (unsigned)
- Owner is in trouble (app re-install required)



# Crypto Counter Confusion Attack

- iOS app allows max value of 4294967295 (uint32)
- Android app allows only 2147483647 (int32)
- When set to the highest value, counter cannot be increased anymore → disfunctional key → app has to be re-installed
- Also recovery after KeyDrop leads to a strange situation
- **Observation:** Several keys with same name require middle console tap when starting to drive
- Re-opens vector for key enrollment attack



# VCSEC SessionInfo

- Unsigned request

--- FromVCSECMessage ---

```
sessionInfo {  
  token: "\377\377\226\300\270\017z$ v\312{\337\225$\341\275x\332y"  
  counter: 3051  
  publicKey: "\004M)d\2136\372\201J\rh\253\354\220cZ\307_\276\320\3568\212G\016\202f\223\025m\267\360\241!}\232\372vH\304_\3532\244\023\016@1hbA\315\276g(+22q\235\3663R.\367"  
}
```

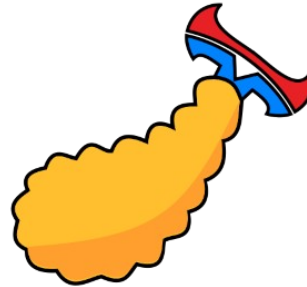
# Authorization Replay Attack



- Vehicle sends token for GCM-AEAD (mainly used for passive entry functions)
- Token can be requested from car (SessionInfo)
- Use token in order to collect Authorization Responses via temporary™ tool
- Dispense Responses via tempara™ tool when asked by vehicle



# Tools and Resources on github



<https://github.com/trifinite>

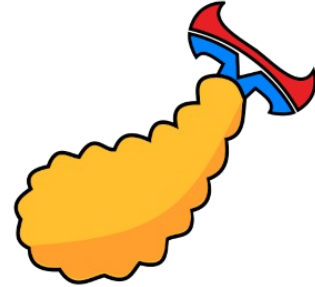
# Temporary – impersonate a Tesla

- temporary.py (on github)
  - BLE Tesla Vehicle emulator
  - Based on bleno (pybleno)
- Version 0.1.2
  - Key drop attack
  - Crypto counter attack
  - de/encoding of VCSEC messages



# Tool: tempara

- tempara.py (on github)
  - Tesla VCSEC client based on Bleak library
  - For **your** Tesla, only!
- Version 0.1.1
  - template for key enumeration
  - de/encoding of VCSEC messages



# Resource: VCSEC Archive

- All VCSEC.proto files to date (on github)
- Provided for educational purposes
- Derived from decompiled Android app
- Shell scripts to get started (protoc rquired)
  - decode.sh script
  - encode.sh script



# Tool: VINTag

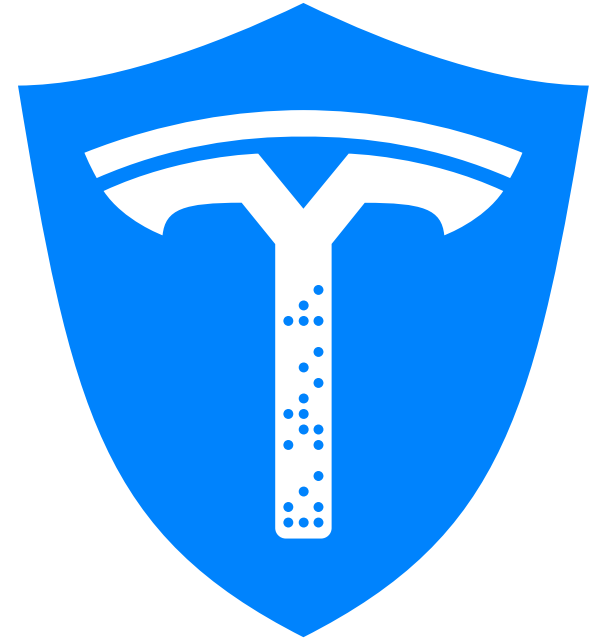
- VINTag.py (on github)
  - API Client for VIN decoding
  - Requires free RapidAPI account / API key
- API Endpoints:
  - <https://rapidapi.com/trifinite/api/tesla-vin-identifier>
  - s3xy: resolves Model Type
  - location: manufacturing location
  - year: manufacturing year
  - vin: complete VIN detection (not free)



# TeslaKee: Doesn't talk to strangers!

- Does talk to your car
- Replacement for Tesla's PhoneKey
- Protection against:
  - Relay Attacks
  - Theft
  - Soon (Q3/2022) available for Android... and maybe later for iOS

[www.teslakee.com](http://www.teslakee.com) - Please leave your contact to stay in the loop!





# Conclusion (1)

- Relay-Attacks are possible
  - PIN2Drive feature should be used / promoted better
  - Tesla PhoneKey really talks to anyone
- NFC-KeyCard
  - Authorization Timer permissions have to be restricted
- App
  - Online- and Offline-Realms have to be united

# Conclusion (2)

- VCSEC does **not** stand for “**V**ehicle **C**ontrol **S**ecurity“
  - It stands for **V**ehicle **C**ontrol **S**econdary
- Convenience/UX trumps™ Security
  - PhoneKey cannot easily be deactivated, etc.
  - Authorization Timer Issues

# What about the KeyFob?

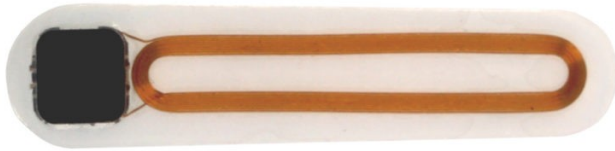
- Research in Progress
- Vehicle initiates connection to KeyFob
- GATT-Structure similar to Vehicle when connection via PhoneKey
- Only connectable when in motion (10s Timeout)
- Shorter Messages compared to PhoneKey comm

# Credits

- Slawomir Jasek, SecuRing (gattacker.io)
- Sandeep Mistry, noble/bleno
- Skylot, jadx
- Lex Nastin (similar work)  
<https://teslabtapi.lexnastin.com/>
- Josh Welder / Samed Ozdemir

# Already too late?

## TeslaFlex “Key Card” Implant



Sold out (!) at <https://dangerousthings.com/product/teslaflex/>

# Thanks for your attention!

Questions?

[trifinite.org/martin](https://trifinite.org/martin)

Slides:

[trifinite.org/tempa](https://trifinite.org/tempa)

Twitter:

[@mherfurt](https://twitter.com/mherfurt) [@trifinite\\_org](https://twitter.com/trifinite_org)

Patreon:

[patreon.com/mherfurt](https://patreon.com/mherfurt)



<https://thehackermind.com>

A little more background in  
Episode 48

# HandlePulledWithoutAuthSpecificPayload

```
HandlePulledWithoutAuthDeviceSpecificPayload
RSSICenter: Int
RSSIFront: Int
RSSILeft: Int
RSSINFCradle: Int
RSSIRear: Int
RSSIRearLeft: Int
RSSIRearRight: Int
RSSIRight: Int
RSSISecondary: Int
authenticationLevel: AuthenticationLevel_E
highThreshCenterPresent: Bool
highThreshFrontPresent: Bool
highThreshLeftPresent: Bool
highThreshNFCPresent: Bool
highThreshRearLeftPresent: Bool
highThreshRearPresent: Bool
highThreshRearRightPresent: Bool
highThreshRightPresent: Bool
highThreshSecondaryPresent: Bool
keyChannel: Int
present: Bool
rawDeltaBayesLeftPresent: Bool
rawDeltaBayesRightPresent: Bool
sortedDeltaBayesLeftPresent: Bool
sortedDeltaBayesRightPresent: Bool
```

- Alert-Message
- Introduced in app Version 4.3.0
- First vehicle firmware 2022.12.3

# FromVCSEC – Alert with Payload

0023ea02200a1e08d806180128013215080618012075287b305f3867680170017801880101

```
--- FromVCSECMessages ---  
alert {  
  alertHandlePulledWithoutAuth {  
    timeSinceAlertSet_ms: 856  
    connectionCount: 1  
    authRequested: true  
    deviceSpecificPayload {  
      keyChannel: 6  
      present: true  
      RSSILeft: -59  
      RSSIRight: -62  
      RSSIRear: -48  
      RSSICenter: -52  
      highThreshLeftPresent: true  
      highThreshRightPresent: true  
      highThreshCenterPresent: true  
      highThreshRearPresent: true  
    }  
  }  
}
```



# From VCSEC – Alert with less details

0018ea02150a1308f4051801320c0804180120772877306d386f

```
alert {  
  alertHandlePulledWithoutAuth {  
    timeSinceAlertSet_ms: 756  
    connectionCount: 1  
    deviceSpecificPayload {  
      keyChannel: 4  
      present: true  
      RSSILeft: -60  
      RSSIRight: -60  
      RSSIRear: -55  
      RSSICenter: -56  
    }  
  }  
}
```