DST80 Cryptanalyse

Kaci Amaouche

14 Septembre 2023

Table des matières

[DST80 3](#_Toc147150861)

[Notations : 3](#_Toc147150862)

[DST80 Transpondeurs : 3](#_Toc147150863)

[Structure algorithmique : 4](#_Toc147150864)

[La cryptanalyse de DST80 5](#_Toc147150865)

[Pour Kia et Hyundai : 5](#_Toc147150866)

[Pour Toyota : 7](#_Toc147150867)

[En résumé : 8](#_Toc147150868)

[Pour Kia et Hyundai : 8](#_Toc147150869)

[Pour Toyota : 9](#_Toc147150870)

[Finalement, voici tous les ingrédients nécessaires : 10](#_Toc147150871)

[Liens pour nos ingrédients : 10](#_Toc147150872)

[Plan d’attaque DST80 : 13](#_Toc147150873)

[Pour la récupération de clés : 13](#_Toc147150874)

[Pour Toyota : 13](#_Toc147150875)

[Pour Hyundai/Kia : 15](#_Toc147150876)

[Pour DST40 : 16](#_Toc147150877)

[DST40-Forger-Une-Trame-Valide 17](#_Toc147150878)

[1 DST40 : génération de challenge : 17](#_Toc147150879)

[2.Format de la trame envoyée en mode RF : 17](#_Toc147150880)

[3.Plan d’attaque : 17](#_Toc147150881)

[Algorithme : 18](#_Toc147150882)

# DST80

## Notations :

Nous utiliserons les notations suivantes :

• d´désigne le bit i de la variable , avec étant le bit de poids faible (le plus à droite).

• De même, représente l’octet i de S, avec étant l’octet de poids faible (le plus à droite).

• Nous représentons une rotation de S de i bits vers la gauche par S ≪ i.

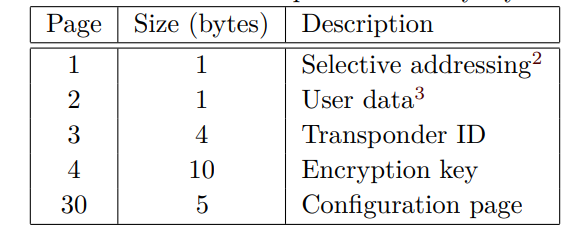
• caractérise une plage allant du bit i au bit j de la variable S.

• De même, un décalage de S de i bits vers la gauche est noté par S ≪ i.

• Enfin, indique une concaténation des octets b et c, avec c étant l’octet de poids faible.

## DST80 Transpondeurs :

Les transpondeurs DST de Texas Instruments sont disponibles dans différents types de boîtiers : le transpondeur classique de type coin (TMS37145), un boitier TSSOP (TMS37126) et un boîtier TSSOP contenant à la fois le TMS37126 et un microcontrôleur MSP430 (TMS37F128). Tous ceux-ci intègrent une clé d’authentification de 80 bits, qui est stockée en interne sous forme de deux fragments de 40 bits, **keyL** et **keyR**. Les transpondeurs Peuvent être configurées soit pour une authentification rapide (le lecteur n’est pas authentifié), soit pour une authentification mutuelle. Les transpondeurs **DST80** sont identifiés de manière unique par un numéro de série de 32 bits stocké dans **l’EEPROM**, composé d’un identifiant unique sur 3 octets et d’un code fabricant sur 1 octet. Selon la fiche technique, les pages 1, 2 et 3 sont toujours lisibles publiquement. De plus, le transpondeur intègre plusieurs pages de m´mémoire utilisateur **EEPROM**, qui peuvent toutes être verrouillées en écriture. Le Tableau suivant donne un aperçu de plusieurs pages significatives de **l’EEPROM** d’un transpondeur DST80.



Agencement de la mémoire du transpondeur DST80

## Structure algorithmique :

Le schéma de m´mémoire du transpondeur DST80 est, tout comme celui du DST40, un réseau de Feistel déséquilibré qui s’exécute sur 200 cycles pour calculer la réponse. Il utilise une clé cryptographique de 80 bits, repartie sur deux registres d’´état de clé indépendants. L’´état interne de 40 bits est initialisé avec le défi de 40 bits généré par le lecteur et mis `a jour avec 2 bits à chaque cycle.

**Définition 1** : Soit la représentation en bits de l’octet b. Alors, la permutation

est d´définie comme suit :

**Définition 2**: Soit la représentation en octets de . Etant donné la permutation définie dans la définition précédente, nous d´définissons l’opération comme suit :

Au d´début de chaque cycle, comme illustré dans l’Algorithme 2, la fonction DST80 Merge(l, r) combine les deux registres d’´état de clé de 40 bits en une clé de cycle de 40 bits, comme le montre l’Algorithme 1. Elle permute les octets des deux clés selon , puis inverse chaque registre en fonction de la valeur de son bit de poids le plus élevé. Enfin, elle renvoie une clé de cycle de 40 bits qui, avec l’´état interne, sert d’entrée à la fonction de Feistel F. Cette fonction de Feistel est identique à son équivalent DST40, produisant une paire de bits qui est renvoyée au registre de d´défi. Contrairement à DST40 (où le registre d’´état de clé est décalé tous les trois cycles), les LFSR contenant les deux clés de 40 bits sont décalés à chaque cycle, avec des bits cibles aux positions 0, 2, 19 et 21

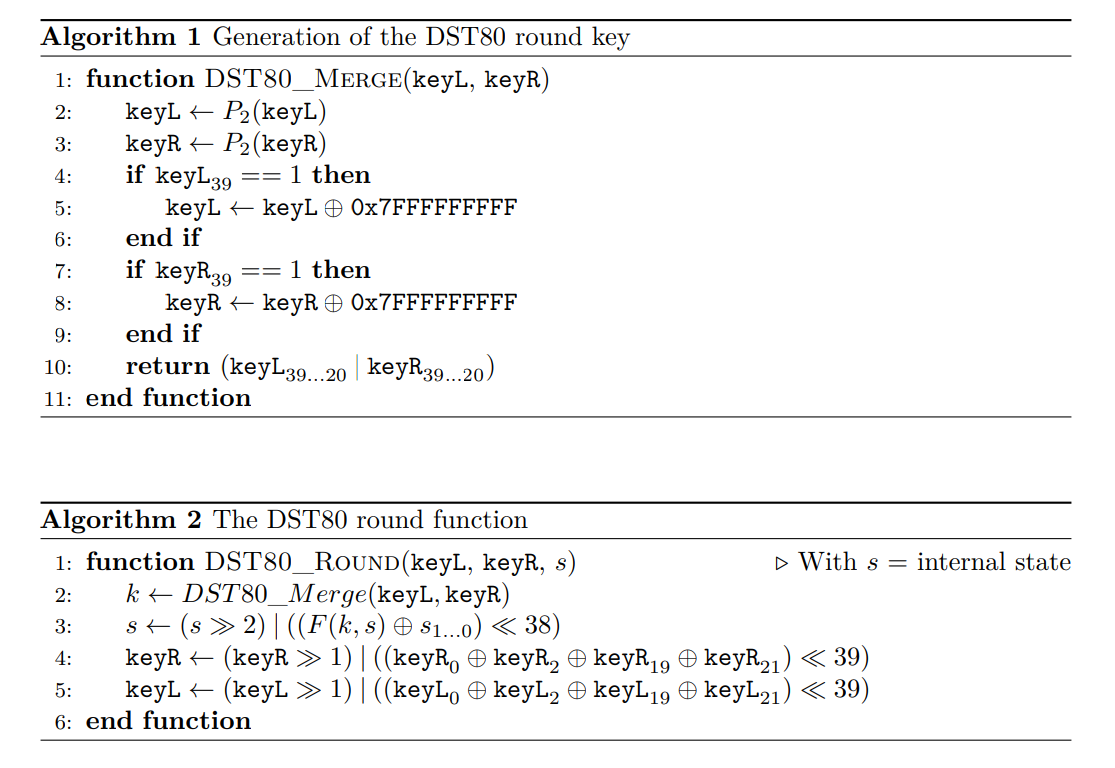


Figure Génération de la clé de cycle DST80 et la fonction de cycle DST80

# La cryptanalyse de DST80

La sécurité du chiffrement DST80 repose sur 80 bits d'entropie dans la clé cryptographique, répartis en deux morceaux de 40 bits : **keyL** et **keyR**. Idéalement, le fabricant de voitures ou le fabricant d'équipement d'origine (OEM) génère une clé secrète aléatoire et unique de 80 bits pour chaque transpondeur apparié. Cependant, en pratique, les auteurs ont découvert que le processus de génération de clé n'est pas totalement aléatoire. Ils ont récupéré les clés secrètes des ECU (Unités de Contrôle Électronique) de systèmes d'immobilisation de trois fabricants différents et ont trouvé des schémas récurrents parmi les clés de chacun d'entre eux. Dans cette section, nous mettons en lumière les schémas de diversification de clés qu'ils ont identifiés en étudiant le micrologiciel de l'immobiliser. Avec la connaissance de ce schéma de génération de clés caché dans le micrologiciel de l'immobiliser, un attaquant peut récupérer les clés cryptographiques d'un transpondeur DST80 avec seulement une paire de challenge-réponse.

## Pour Kia et Hyundai :

Bien que la mémoire flash du microcontrôleur soit généralement assez grande et utilisée pour le code du programme, l'EEPROM contient généralement de petites quantités de données spécifiques à la voiture, voire à l'ECU. Après avoir localisé les numéros de série des transpondeurs appariés dans l'EEPROM, il semblait très probable que les clés DST80 correspondantes y soient également stockées. Étant donné que l'EEPROM interne du microcontrôleur sur l'immobiliser ne fait que 384 octets, il est plausible de récupérer la clé DST80 en effectuant une recherche exhaustive sur les données de l'EEPROM.

Tout d'abord, un défi est généré et la signature correspondante est acquise à partir du transpondeur. En supposant que **keyL** et **keyR** soient adjacents en mémoire, chaque séquence de 10 octets est définie comme clé cryptographique et vérifiée si elle produit la même réponse. Afin d'éliminer tout problème causé par l'endianness, une signature est calculée avec chaque octet de clé à chaque position dans la clé de 10 octets. Notez que pour les EEPROM plus grandes, une analyse d'entropie peut être effectuée pour identifier les zones les plus susceptibles de contenir une clé cryptographique. De plus, si la procédure précédente ne produit pas de clé candidate, elle peut être améliorée en recherchant le binaire par blocs de cinq octets, ou dans le pire des cas, en recherchant par octet. Cependant, cela peut entraîner une explosion d'états et devenir inapplicable pour des tailles d'EEPROM plus grandes.

En utilisant ces techniques, les dix octets de clé ont été récupérés à partir de l'EEPROM interne, comme indiqué dans le Tableau 3. Une première observation est que même si deux transpondeurs physiquement différents sont associés à l'immobiliser, ils partagent la même clé cryptographique. Cela peut être confirmé sans connaître la clé DST80 en envoyant le même défi aux deux transpondeurs, qui génèrent des réponses identiques.

De plus, le Tableau 3 suggère que deux octets à la fois de **keyL** et **keyR** sont identiques dans les quatre clés DST80, laissant six octets inconnus. L'Algorithme 3 clarifie le schéma de génération de clé utilisé par Kia et Hyundai, ne laissant que trois octets d'entropie. Ainsi, à condition d'avoir un bref accès à une télécommande de clé légitime, un attaquant peut immédiatement récupérer une clé DST80 valide à partir d'un transpondeur Kia/Hyundai avec une seule paire de défi-réponse.

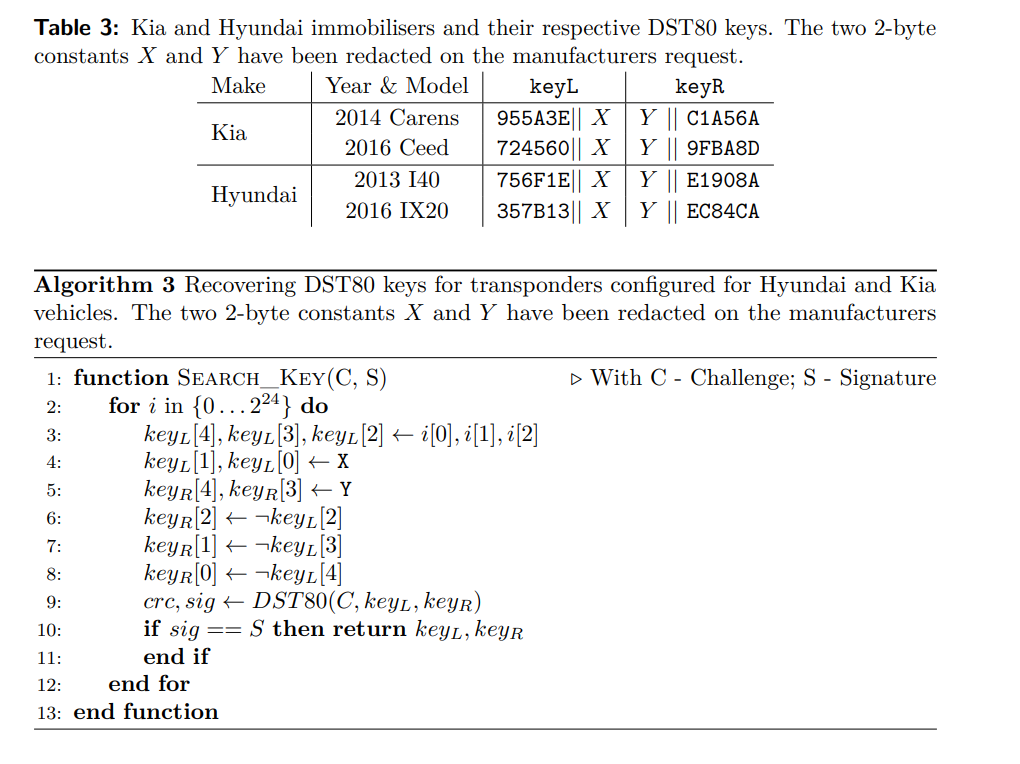


Figure 1 Hyundai/Kia crack

Cette attaque permet de réduire la complexité de à seulement .

## Pour Toyota :

Les auteurs ont effectué une rétro-ingénierie du micrologiciel de l'immobiliser de Toyota afin de récupérer et d'analyser les procédures sensibles liées à la sécurité. À chaque fois qu'un transpondeur est présenté, l'immobiliser dérive la clé DST80 à partir de son numéro de série (stocké à la page 3) et des valeurs d'1 octet stockées aux pages 1 et 2. Étant donné que le byte de poids faible de l'ID du transpondeur DST80 est le code du fabricant, ce byte n'affecte pas la génération de clé.

En plus de ces trois pages de transpondeur toujours lisibles, le schéma de dérivation de clé repose sur trois constantes de sécurité stockées dans la mémoire flash interne de l'immobiliser. D'après leurs expériences, ces constantes de sécurité sont identiques pour tous les immobilisers Toyota DST80 qu’ils ont analysés. Le fait qu'elles soient situées dans la mémoire flash renforce cette hypothèse, car la mémoire flash est moins couramment utilisée pour stocker des secrets uniques.

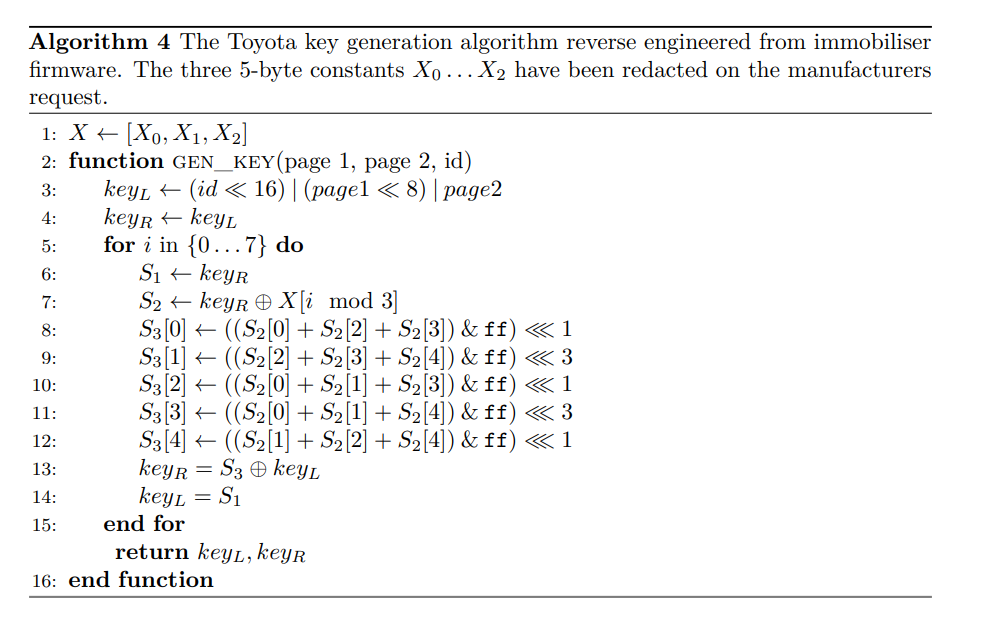


Figure Récupération de clés Toyota

# En résumé :

## Pour Kia et Hyundai :

Les auteurs suggèrent que les clés **keyL** et **keyR** se trouvent dans **l'EEPROM**, et qu'il existe deux constantes **X** et **Y** de 2 octets chacune qui sont fixes pour tous les modèles.

Voici le passage correspondant :

"Alors que la mémoire flash du microcontrôleur est généralement assez grande et donc utilisée pour le code du programme, **l'EEPROM** contient généralement de petites quantités de données spécifiques à la voiture ou même à **l'ECU**. Après avoir localisé les numéros de série des transpondeurs appariés dans **l'EEPROM**, nous avons jugé très probable que les clés **DST80** correspondantes y seraient également stockées."

Une recherche exhaustive sur tous les 10 octets possibles suffirait pour récupérer les deux clés. Si cela ne fonctionne pas, il est possible d'analyser différentes **EEPROM** pour identifier les deux constantes **X** et **Y**, puis il existe un algorithme permettant de cracker les deux clés avec une complexité de .

**Il est donc nécessaire de :**

• Soit repérer X et Y en analysant quelques **EEPROMs** et effectuer une recherche par force brute. Pour cela, un **challenge**/**response** est requis (contact avec une vraie voiture, Proxmark3, simulateur, etc.), dont le challenge est l'entrée de l'algorithme **DST80**, et la **response** en est la sortie. Pour confirmer la validité des deux clés trouvées, il faut donc exécuter l'algorithme qui cracke les clés en testant toutes les combinaisons possibles, et tester chacune de ces combinaisons pour voir si l'exécution de l'algorithme **DST80** avec ces combinaisons + le **challenge** produit la bonne **response**.

• Soit effectuer une recherche exhaustive sur tous les octets de **l'EEPROM** en espérant que l'hypothèse des auteurs soit correcte (**Keyl** et **KeyR** se trouvent dans **l’EEPROM**). Dans ce cas également, un **challenge**/**response** est nécessaire.

Pour cela, il suffit de : Soit l’algorithme de chiffrement **DST80**. On va donc tester toutes les clés candidates présentes dans l’EEPROM, soit **keyLcand**, **keyRcand** les deux clés candidates. On fera une boucle qui parcourt tous les 10 octets possibles et on testera si : , si c’est le cas, alors on a retrouvé les deux clés.

## Pour Toyota :

• Première hypothèse :

* Après avoir examiné **l'EEPROM** d'un véhicule Toyota, j'ai observé que la structure des clés **keyL** et **keyR** suit une méthode similaire à celle utilisée par Hyundai et Kia, où plusieurs séquences d'octets se suivent avec leurs compléments.

Cela reste à être vérifié en utilisant la même méthode que celle que nous avons employée avec Hyundai et Kia, c'est-à-dire en exécutant l'algorithme avec une complexité de (Figure 1), en utilisant ces valeurs en tant qu'entrées (Les clés candidates générées par l’algo) avec le **challenge**, puis en testant contre la **response**. Si le triplet produit la même **response**, alors les deux clés sont retrouvées.

•Deuxième hypothèse :

-Les auteurs ont proposé une manière de reconstituer **keyL** et **keyR** avec un algorithme de complexité d'ordre O(1), et pour cela, nous avons besoin de :

• **l'ID** : il se trouve dans la page 3 du transpondeur.

• **page1** et **page2** qui sont : **Selective addressing** et **User data** respectivement.

• les trois constantes de 5 octets **X0, X1, X2** : dans la mémoire flash interne de l'immobiliser.

## Finalement, voici tous les ingrédients nécessaires :

1- **EEPROM** BCM pour Hyundai/Kia utilisant **DST80** : Nécessaire pour tester tous les 10-bytes possibles (Si l’hypothèse des auteurs est correcte, c’est-à-dire : les clés sont présente dans **l’EEPROM**), si leur hypothèse n’est pas correcte, on en aura quand même besoin pour identifier les deux constantes **X** et **Y** afin d’exécuter l’algorithme présenté dans la Figure 1.

2-Firmware Immo (Flash) pour Toyota uitlisant DST80 : Nécessaire pour identifier les trois constantes **X0, X1** et **X2** qui sont identiques pour tous les modèles Toyota **DST80**.

3-Clé de voiture Toyota utilisant DST80 : Nécessaire pour retrouver l’ID, page 1 et page 2 qui servent comme entrées pour l’algo présenté dans la Figure 2.

4-Clés de voitures Hyundai/Kia ou Hyundai : afin de de capturer des trames radios et d’identifier le format de la trame (extraction du keystream) qui sera nécessaire pour la vérification de la validité de nos tests

5- Génération de **challenge/response** : C'est l'élément le plus crucial pour vérifier la validité de nos clés candidates. Étant donné que nous n'avons pas accès à la voiture, il existe des outils tels que **Tango Programmer** ou **Proxmark3** qui permettent de générer des défis/réponses.

## Liens pour nos ingrédients :

-1 : EEPROM BCM Hyundai Tucson 2017

<https://cardamp.ru/view_full_one.php?id_damp_list=36620> (Full)

-2 Immo (Flash) Toyota prius 2012

<https://cardamp.ru/view_full_one.php?id_damp_list=32337> (Full)

-3 clé de voiture Toyota:

- <https://cleauto.fr/fr/toyota/3316-toyota-4boutons-4d-g.html>

-[Clé télécommande à bascule HYQ12BBY, 3/4 boutons, 314.4Mhz, puce 4D67 /G, pour voiture Toyota Camry, Avalon, Corolla Matrix, RAV4, Venza, Yari - AliExpress](https://fr.aliexpress.com/item/1005005194880363.html)

-

-4 Clés de voitures Kia/Hyundai :

-[Clé télécommande à 3 boutons pour voiture, 433MHz, puce 4D60 pour KIA Carens Rondo 2012 + P/N: 95430-A4200 - AliExpress](https://fr.aliexpress.com/item/33011996539.html?spm=a2g0o.productlist.main.31.34e11f2eCaHw97&algo_pvid=611ccae8-5d93-4b31-8aff-1e6be41dd363&algo_exp_id=611ccae8-5d93-4b31-8aff-1e6be41dd363-15&pdp_npi=4%40dis%21EUR%2119.43%2117.3%21%21%2120.28%21%21%40211b600e16951364326371205e1198%2167123726549%21sea%21FR%210%21ABS&curPageLogUid=xh6Sp3UuLxkJ)

- <https://cleauto.fr/fr/hyundai/3053-hyundai-3boutons-4d60-dst-80.html>

- <https://cleauto.fr/fr/hyundai-sonata/4290-hyundai-sonata-31-95430-c1210-avec-transpondeur-.html>

- <https://cleauto.fr/fr/accueil/4291-hyundai-elantra-95430-f2110.html>

- <https://cleauto.fr/fr/hyundai/4313-hyundai-3-boutons-95430-h6500h5500.html>

- 5 Outils pour générer des **challenge/response**

Il existe plusieurs outils qui permettent de simuler les sysstème RFID sans avoir accès à la voiture, voici un passage d'un articule scientifuqe dont lequel les auteurs ont utilisé (Proxmark2, Tango Programmer 3, MiraClone) :

"

Through Car Lock Systems we have access to many different kinds of keys and cars with RKE feature, and can

make use of their production facility which includes the metal working equipment to cut keys, as well as the diagnostic

tools to teach new keys to cars. We also have access to special-purpose hardware and software to work with the

RFID transponder chips within the keys without the use of a car, (Proxmark2, Tango Programmer 3, MiraClone 4).

"

1 - Tango programmer :

lien : <https://www.boutique.lacleauto.fr/fr/p/10-tango-programmeur-de-cle-et-transpondeur-3000000000021.html>

"

ango programmer est un appareil de programmation de transpondeurs RFID, qui se concentre sur la résolution des situations de “toutes les clés perdues”.

Il couvre une grande variété de marques, de modèles et de systèmes de véhicules couvrant les trois dernières décennies1.

Il permet de créer, copier, générer et émuler des clés pour différentes marques de voitures, motos et camions2.

Il dispose également d’un logiciel appelé Tango Maker, qui permet de créer des images de transpondeurs personnalisées

"

2 - Proxmark3 :

lien : <https://lab401.com/fr/products/proxmark-3-rdv4>

" Le Proxmark3 est un outil polyvalent pour l’analyse, la recherche et le développement de la sécurité des identifications par radiofréquence (RFID).

Il prend en charge les cartes de proximité à haute fréquence (13,56 MHz) et à basse fréquence (125/134 kHz) et permet aux utilisateurs de lire, d’émuler, de brouiller et de forcer brutalement la majorité des protocoles RFID1.

Il a été conçu à l’origine par Jonathan Westhues et a évolué au fil des ans en un outil standard pour l’analyse RFID2. Il existe plusieurs versions du matériel Proxmark3, optimisées pour différents usages et environnements2.

Le Proxmark3 est utilisé par les chercheurs en sécurité, les développeurs de produits, les forces de l’ordre et les passionnés d’RFID pour tester, espionner, reproduire et cloner des tags RFID

"

3 - MiraColne4 :(Fonctionne que pour Megamos)

"MiraClone 4 est un appareil de copie et de programmation de clés et de transpondeurs pour les véhicules. Il permet de détecter, lire et cloner les transpondeurs à code fixe et certains transpondeurs cryptographiques.

Il permet également de programmer les clés perdues ou endommagées à partir des données Eeprom1. MiraClone 4 est la dernière version du MiraClone, qui a été conçu et développé pour suivre l’évolution constante des technologies de transpondeurs dans les systèmes de véhicules. MiraClone 4 offre une flexibilité complète, car il peut être utilisé comme un outil autonome ou avec un logiciel PC qui facilite son utilisation.

Il dispose également de plus de 460 modules Eeprom au choix, ce qui couvre 1805 modèles de voitures différents1. MiraClone 4 est un outil professionnel et innovant pour la sécurité des clés et des transpondeurs.

"

4-Tmpro2 : (Il ne supporte pas DST80)

"Tmpro2 est un programmeur et un logiciel pour la fabrication, la programmation, la copie et le calcul de clés et de transpondeurs pour les véhicules.

Il peut également lire et programmer les données Eeprom des systèmes d’immobilisation.

Tmpro2 est un outil polyvalent et professionnel pour les serruriers et les spécialistes de la sécurité des clés12.

Il prend en charge plus de 200 modules logiciels différents, qui couvrent une large gamme de marques et de modèles de voitures, de camions et de motos2. Tmpro2 est facile à utiliser, car il peut fonctionner comme un outil autonome ou avec un logiciel PC qui offre une interface conviviale et multilingue12.

Tmpro2 est un outil innovant et mis à jour régulièrement pour suivre l’évolution constante des technologies de transpondeurs dans les systèmes de véhicules2."

5- AVDI:

lien : <https://abrites.fr/products/pkg-avdi>

"AVDI est l’acronyme de ABRITES Vehicle Diagnostic Interface, qui est une interface de diagnostic de dernière génération produite par Abrites ltd.

pour les véhicules, les motos, les camions, les scooters des mers, les VTT et les véhicules utilitaires lourds12. C’est la base sur laquelle repose tout l’écosystème des produits ABRITES, qui offrent un large éventail de fonctionnalités pour le diagnostic, la programmation, la copie et le calcul des clés et des transpondeurs12.

AVDI est un outil polyvalent et professionnel, qui prend en charge presque toutes les marques de véhicules sur le marché12.

"

# Plan d’attaque DST80 :

## Pour la récupération de clés :

### Pour Toyota :

Avant de présenter les solutions, nous exposons les descriptions de cinq constantes cruciales :

**Page1** : Selective adressing, c’est une constante d’un byte utilisée dans l’algo de génération de clé.

**Page2** : User data, c’est également une constante d’un byte utilisée pour générer la clé

**Page3** : l’**ID** du transpondeur, utilisée également pour générer la clé

**Page30** : Configure page, Elle nous offre la possibilité de commuter entre les deux modes de chiffrement DST40 et DST80.

**X0, X1, X2 :** Trois constantes de 5 bytes identiques à tous les modèles Toyota et sont utilisées pour générer la clé.

#### Première solution :

-Une fois qu’on a acheté la clé, on le met dans le lecteur et on extrait la **page1**, **page2** et **page3.**

-Une fois qu’on a acheté le Firmware qui contient la mémoire flash interne de l’immobiliser, on extrait les trois constantes **X0, X1, X2**.

-On génère un **challenge/response** avec **Tango Programmer** ou **CN900mini,** puis on lance l’algo représenté dans la **figure 2,** on récupère la clé en quelques millisecondes, puis on chiffre le même **challenge** qu’on a utilisé dans **Tango Programmer** ou **CN900min** avec la clé qu’on récupérée et on vérifie qu’on obtient le même chiffré (**response**).

Lien : <https://cardamp.ru/view_full_one.php?id_damp_list=32337>

#### Deuxième solution :

Dans le cas où l'acquisition du Firmware s'avérerait difficile (en raison de problèmes d'achat, d'un Firmware invalide ou de son absence) :

- On achète plusieurs transpondeurs Toyota utilisant le **DST80**, il se peut que l’on tombe sur un transpondeur avec la **page30** non verrouillée, dans ce cas, Un transpondeur configuré pour fonctionner avec **DST80** et dont la **page30** est laissée déverrouillée peut être rétrogradé pour utiliser **DST40** à la place. Dans ce scénario, le transpondeur utilise la clé **keyL**, l'une des deux clés de 40 bits utilisées dans **DST80**, pour calculer une réponse **DST40**. Cela signifie qu'un attaquant peut rétrograder le transpondeur et récupérer la moitié de la clé de 80 bits en utilisant seulement deux paires de challenge-réponse, réduisant ainsi considérablement la complexité computationnelle de l'attaque de à chiffrements. Après être revenu en mode 80 bits, le transpondeur réutilisera la clé de 80 bits d'origine.

Dans cette situation, nous aurons réussi à récupérer la clé de chiffrement. Étant donné que cet algorithme est gourmand en termes de temps, et étant donné que nous disposons d'un algorithme qui génère la clé de manière linéaire en fonction de **page1**, **page2**, **page3**, et des trois constantes **X0, X1** et **X3**, nous n'aurons besoin d'effectuer cette méthode de recherche exhaustive qu'une seule fois pour récupérer la clé. Une fois que nous aurons obtenu la clé, nous pourrons ensuite reconstituer les trois constantes **X0, X1 et X2**. En fin de compte, nous n'aurons donc plus besoin du firmware.

Lien :

-- <https://fr.aliexpress.com/item/1005003283329840.html>

* [https://fr.aliexpress.com/item/1005004292670056.html?spm=a2g0o.productlist.main.107.7d 33347da4zXxj&algo\_pvid=51d17f64-8e1f-4e0a-9674-6a929e13f1e4&algo\_exp\_id=51d17f64- 8e1f-4e0a-9674-6a929e13f1e4 - 53&pdp\_npi=4%40dis%21JPY%2111472%215736.0%21%21%2175.62%21%21%4021038ede1 6958087225708472e29b6%2112000028656362261%21sea%21JP%210%21AB&curPageLogUi d=zFUpkpz5R6kE](https://fr.aliexpress.com/item/1005004292670056.html?spm=a2g0o.productlist.main.107.7d%0933347da4zXxj&algo_pvid=51d17f64-8e1f-4e0a-9674-6a929e13f1e4&algo_exp_id=51d17f64-%098e1f-4e0a-9674-6a929e13f1e4%09-%0953&pdp_npi=4%40dis%21JPY%2111472%215736.0%21%21%2175.62%21%21%4021038ede1%096958087225708472e29b6%2112000028656362261%21sea%21JP%210%21AB&curPageLogUi%09d=zFUpkpz5R6kE)

* <https://cleauto.fr/fr/transpondeur-de-programmation/2988-transpondeur-4d-80bit.html>
* <https://fr.aliexpress.com/item/1005004291722935.html?spm=a2g0o.productlist.main.103.41ab2b8e3NraBG&algo_pvid=cc30498d-7047-411f-97fc-564a896be012&algo_exp_id=cc30498d-7047-411f-97fc-564a896be012-51&pdp_npi=4%40dis%21JPY%219254%214627.0%21%21%2161.00%21%21%40211b613116958222697065776ed573%2112000028651839268%21sea%21JP%210%21AB&curPageLogUid=EImT1VmuMhCn>
* <https://fr.aliexpress.com/item/1005003764068068.html?spm=a2g0o.productlist.main.111.41ab2b8e3NraBG&algo_pvid=cc30498d-7047-411f-97fc-564a896be012&algo_exp_id=cc30498d-7047-411f-97fc-564a896be012-55&pdp_npi=4%40dis%21JPY%213110%212767.0%21%21%2120.50%21%21%40211b613116958222697065776ed573%2112000027084202398%21sea%21JP%210%21AB&curPageLogUid=nuXuArf1pBuX>

#### Troisième solution :

-Après avoir examiné **l'EEPROM** d'un véhicule Toyota (Prius 2010-2015), j'ai observé que la structure des clés **keyL** et **keyR** suit une méthode similaire à celle utilisée par Hyundai et Kia, où plusieurs séquences d'octets se suivent avec leurs compléments.

Si l’hypothèse des auteurs est correcte (les deux clés **keyL** et **keyR** se trouvent dans **l’EEPROM**), une solution est donc d’acheter une clé Toyota Prius 2010-2013, générer un **challenge/response** avec **CN900mini,** puis tester tous les 10-bytes possibles (Avec l’algorithme que j’ai implémenté : Toyota\_key\_recvoer\_eeprom.ipynb), et si deux clés candidates renvoient la même **response** avec le même **challenge,** on a donc retrouvé la bonne clé et ensuite on pourra reconstituer et retrouver nos 3 constantes **X0, X1** et **X2**.

Lien : <https://fr.aliexpress.com/item/1005004424156228.html?spm=a2g0o.productlist.seoads.4.2f8630ad54Nrmb&p4p_pvid=2023092706502756028690248100000252736_2&s=p>

### Pour Hyundai/Kia :

#### Première solution :

Si l'obtention et l'achat de l'EEPROM sont réalisables :

-1 : Si l’hypothèse des auteurs est correcte (Les deux clés **keyL** et **keyR** se trouvent dans **l’EEPROM**) :

-On récupère ces deux clés en testant tous les 10-bytes possibles présents dans **l’EEPROM** (j’ai implémenté l’algo qui gère ça)**,** ensuite on génère un **challenge**/**response** avec **Tango Programmer** et on vérifie si ces deux clés, avec le même **challenge**, produisent la bonne **response**.

-2 : Si l’hypothèse des auteurs n’est pas correcte :

-On va identifier les deux constantes **X** et **Y,** on génère un **challenge/response** avec **Tango Programmer,** puis on exécute l’algo présenté dans **figure1**

Lien : <https://cardamp.ru/view_full_one.php?id_damp_list=36620>

#### Deuxième solution :

Si l’obtention et l’achat de l’EEPROM ne sont pas réalisables :

- On commande et on achète une clé de voiture Kia ou Hyundai figurant déjà dans l'article de cryptanalyse de DST80 (ce qui nous donne 6 octets de la clé), il devient possible de procéder à une recherche exhaustive des 4 octets restants pour récupérer les deux constantes **X** et **Y**. Une fois en possession de ces deux constantes, il devient réalisable d'exécuter l'algorithme de complexité pour retrouver la clé en utilisant seulement une paire de **challenge/réponse**.

Lien :

[https://fr.aliexpress.com/item/33001636133.html?spm=a2g0o.productlist.main.33.34e11f2e CaHw97&algo\_pvid=611ccae8-5d93-4b31-8aff-1e6be41dd363&algo\_exp\_id=611ccae8-5d93- 4b31-8aff-1e6be41dd363- 16&pdp\_npi=4%40dis%21EUR%2122.81%2120.07%21%21%2123.81%21%21%40211b600e1 6951364326371205e1198%2167012753320%21sea%21FR%210%21ABS&curPageLogUid=md 0U4ojsFJKe](https://fr.aliexpress.com/item/33001636133.html?spm=a2g0o.productlist.main.33.34e11f2e%09CaHw97&algo_pvid=611ccae8-5d93-4b31-8aff-1e6be41dd363&algo_exp_id=611ccae8-5d93-%094b31-8aff-1e6be41dd363-%0916&pdp_npi=4%40dis%21EUR%2122.81%2120.07%21%21%2123.81%21%21%40211b600e1%096951364326371205e1198%2167012753320%21sea%21FR%210%21ABS&curPageLogUid=md%090U4ojsFJKe)

#### Troisième solution :

Si la deuxième solution ne fonctionne pas (Clé erronée ou introuvable) :

-On achète des transpondeurs Hyundai utilisant DST80 et on applique la deuxième solution appliquée à Toyota (Récupérer la clé en récupérant d’abord 40 bits ensuite les autres 40 bits), et une fois la clé récupérée, on va identifier une bonne fois pour toutes les deux constantes **X** et **Y**.

Lien :

<https://fr.aliexpress.com/item/32921349032.html>

<https://fr.aliexpress.com/item/1005001406785367.html?spm=a2g0o.productlist.main.35.a2c37ac8QVLMT4&algo_pvid=f7fc8899-bcec-4654-8722-065a762cdd7e&algo_exp_id=f7fc8899-bcec-4654-8722-065a762cdd7e-17&pdp_npi=4%40dis%21JPY%21457%21384.0%21%21%213.01%21%21%40211b88ec16958232276364697ef3e8%2112000015962529341%21sea%21JP%210%21AB&curPageLogUid=qqSLBNf8IMlj>

(4D70, 4D63)

## Pour DST40 :

On aura besoin d’une clé ou transpondeur type DST40 pour générer un challenge/response et pouvoir brute forcer :

<https://fr.aliexpress.com/item/1005004055597099.html> (CN11, CN2, CN5, CN64)

<https://fr.aliexpress.com/i/32923715933.html> (tout type)

<https://fr.aliexpress.com/i/1005001456616857.html>

# DST40-Forger-Une-Trame-Valide

## 1 DST40 : génération de challenge :

1. Le système commence par récupérer un compteur de 40 bits `a partir de la mémoire EEPROM du transpondeur DST40.

2. Ce compteur de 40 bits est utilisé comme entrée ou ”challenge” pour la fonction de compression DST40.

3. La fonction de compression DST40 utilise ce challenge ainsi qu’une clé secrète de 40 bits (la même que celle utilisée dans le système PKES) pour produire une réponse de 3 octets.

4. La réponse de 3 octets de la fonction DST40 est ensuite concaténée avec d’autres informations, y compris un préambule, un octet d’action (lock/unlock), et les deux octets les moins significatifs du compteur de 40 bits.

5. Le paquet RF final est construit en utilisant cette concaténation, et il est transmis `a la voiture

. 6. Après la transmission, le compteur de 40 bits est incrémenté et `a nouveau stocké dans la mémoire EEPROM du transpondeur. Ce compteur joue un rôle crucial dans la prévention des attaques de rejeu.

## 2.Format de la trame envoyée en mode RF :

|  |  |  |
| --- | --- | --- |
| Préambule (8 bits) | Response (24 bits) | Counter (16 bits) |

## 3.Plan d’attaque :

Dans ce scénario, nous disposons de la clé de chiffrement et d’une (ou deux) trame(s) RF. Si nous ne disposons pas de la clé, nous pouvons envoyer des challenges aux transpondeurs et recevoir des réponses (à condition que l’ID soit correct), ce qui nous permettrait de retrouver la clé.

Etant donné que la réponse fait 24 bits (R1) et que la taille de la clé est de 40 bits (K), une seule trame ne suffirait pas (environ 2^16 clés peuvent produire la même réponse). Supposons que nous ayons capture une première trame, nous connaissons donc 24 bits du chiffré (R1) et les 16 bits les moins significatifs du clair (challenge), car le challenge est en fait ce même compteur (C) qui se trouve dans ces bits. Cette information réduit l’espace des 2^16 clés, mais il n’est pas certain qu’elle soit suffisante pour réduire l’espace des clés `a une seule. Il est donc préférable de capturer deux trames RF

Voici un algorithme qui permet de retrouver le challenge utilisé lors d’une transmission :

Si les deux transmissions capturées ne sont pas consécutives, une solution consiste à ajouter une boucle lors de la deuxième étape de l'algorithme, limitée à 16 itérations (c'est-à-dire que si la deuxième transmission est capturée après 15 autres appuis).

### Algorithme :

Entrée : (R1, R2, K)

Sortie : C

1-list\_counter=[]

2-Pour tout C dans 0...2^40:

Si dst40(C, K) == R1 :

Ajouter C à list\_counter

3-Pour tout C dans list\_counter:

Pour tout i dans 0...15 :

Si dst (C + i, K) == R2 :

Retourner C + i

Après avoir obtenu le compteur (C), la création d'une trame valide implique l'envoi d'une séquence initiale (préambule), dont la découverte peut nécessiter une attaque de force brute dans le pire des cas. Cette séquence est suivie d'une resonse, qui est le résultat du chiffrement du compteur incrémenté (C + i) additionné aux 16 bits les moins significatifs du compteur (C+i). À chaque envoi de trame ultérieur, nous augmentons la valeur du compteur (C+i) et la conservons pour les transmissions ultérieures.

## 4-Capture de trames RF Toyota DST40 et identification de données chiffrées

### Génération et capture de trames :

J'ai enregistré un total de 60 trames RF Toyota (TOY-3P-1) DST40 en appuyant successivement 20 fois sur chaque bouton. J'ai créé trois fichiers distincts en utilisant l'outil URH, chaque fichier contenant 20 trames capturées consécutivement pour chaque bouton. Les trois fichiers se trouvent dans ce même repo.

### Analyse de trame :

#### Raisonnement par élimination : on identifie et élimine les mauvais bits

On reçoit des trames de taille environ 510 bits.

Le codage utilisé est Manchester.

Nous cherchons spécifiquement une séquence de 24 bits présentant une entropie élevée, indiquant une grande variabilité par rapport aux autres séquences. Cette séquence est supposée être la réponse (response), suivie d'une autre séquence de 16 bits, le compteur (counter), qui devrait varier beaucoup moins et donc une faible variation.

- Tout d'abord, il est évident que les bits de 1 à 9 sont constants et toujours égaux à 111111111. De temps en temps, des erreurs peuvent survenir (où un zéro est détecté à un certain endroit), et je suppose que cela est dû à des problèmes de transmission et de capture, car la majorité des occurrences montrent clairement le 111111111 prédominant sur les exceptions.

-Ensuite, on voit aussi que les bits de 10 à 35, ils sont quasi constants et ne varient pas, il y a principalement deux valeurs qui se répètent aléatoirement à chaque appuie, donc le chiffré n’est pas censé être là.

-On voit également que les bits de 128 à 136 sont fixes (000000000), parfois aussi y a des erreurs qui sont forcément dû à la capture des trames.

-Finalement, grâce à cyberchef, j’ai pu repérer et voir qu’à partir du bit 137 jusqu’à la fin, ce sont des répétitions successives des bits 9 à 126.

On peut donc conclure que le chiffré (response) se trouve entre le bit 36 et 127, ce qui fait 92 bits.

#### Identifier la séquence chiffrée de 24 bits (response) :

Pour identifier la séquence chiffrée de 24 bits, j'ai mis en place un script que vous pouvez trouver dans le même référentiel (Entropie-Shannon.py). Ce script contient deux fonctions principales : la première calcule l'entropie d'une séquence donnée, tandis que la deuxième fonction prend une séquence donnée en paramètre (pour nous, ce serait les bits de 36 à 127 pour chaque ligne). Cette deuxième fonction calcule ensuite l'entropie de toutes les combinaisons possibles, c'est-à-dire les 68 combinaisons possibles (car 92 - 24 = 68), puis elle renvoie la séquence avec la plus grande entropie, supposée être le texte chiffré.

En théorie, l'algorithme devrait renvoyer le même positionnement de la séquence dans chaque ligne, mais dans la pratique, il y a parfois des positions différentes. Dans ce cas, la logique veut que vous choisissiez la position qui apparaît le plus fréquemment.