

Gone in 360 Seconds – Hijacking with Hitag2

Umgehen der elektronischen Wegfahrsperre moderner Fahrzeuge – basierend auf: [1]

Sebastian Lehmann | 17. Februar 2014

SEMINAR “MODERNE KRYPTOANALYSE”



Was ist eine Wegfahrsperre?

- verhindert Wegfahren auch wenn Fahrzeug geöffnet wurde
- oftmals mit Zentralverriegelung verknüpft (ein System für beides)
- Trend: Verzicht auf mechanischen Schlüssel (Komfort)

Was ist eine Wegfahrsperre?

- verhindert Wegfahren auch wenn Fahrzeug geöffnet wurde
- oftmals mit Zentralverriegelung verknüpft (ein System für beides)
- Trend: Verzicht auf mechanischen Schlüssel (Komfort)

Die elektronische Wegfahrsperre Hitag2

- von NXP Semiconductors (ehem. Philips) entwickelt und vertrieben
- von mehr als 30 Automobilherstellern in über 200 Modellen verbaut (u.a. BMW, VW, Audi A8, Porsche Cayenne)
- benutzt proprietäres Protokoll & Verschlüsselung (48 Bit Schlüssel!)

Was ist eine Wegfahrsperre?

- verhindert Wegfahren auch wenn Fahrzeug geöffnet wurde
- oftmals mit Zentralverriegelung verknüpft (ein System für beides)
- Trend: Verzicht auf mechanischen Schlüssel (Komfort)

Die elektronische Wegfahrsperre Hitag2

- von NXP Semiconductors (ehem. Philips) entwickelt und vertrieben
- von mehr als 30 Automobilherstellern in über 200 Modellen verbaut (u.a. BMW, VW, Audi A8, Porsche Cayenne)
- benutzt **proprietäres** Protokoll, Verschlüsselung (**48 Bit** Schlüssel!)



Funktionsweise von Hitag2 – Übersicht

System besteht aus:

Lesegerät am Zündschloss



Transponder im Schlüssel



(Bilder beispielhaft, einige Variationen vorhanden)

Zum Auf- und Abschließen:

- Schlüssel kommuniziert mit Fahrzeug auf Knopfdruck
- Authentifikation erfolgreich? \Rightarrow Fahrzeug führt gewählte Aktion durch
- Funktechnik: Kurzwelle, Reichweite mehrere Meter (**Mithörgefahr**)

Zum Auf- und Abschließen:

- Schlüssel kommuniziert mit Fahrzeug auf Knopfdruck
- Authentifikation erfolgreich? \Rightarrow Fahrzeug führt gewählte Aktion durch
- Funktechnik: Kurzwelle, Reichweite mehrere Meter (**Mithörgefahr**)

Zur Entsperrung der Wegfahrsperre:

- Sobald Schlüssel eingesteckt: Fahrzeug “sucht” nach Transponder
- Transponder antwortet und Authentifikation erfolgreich? \Rightarrow Freigabe
- Funktechnik: RFID, Reichweite einige Zentimeter

Gemeinsam bekannte Geheimnisse

- Jeder Transponder eigene ID + Schlüssel, registriert im Fahrzeug
- Je Fahrzeug zusätzlich geheimes Passwort

Gemeinsam bekannte Geheimnisse

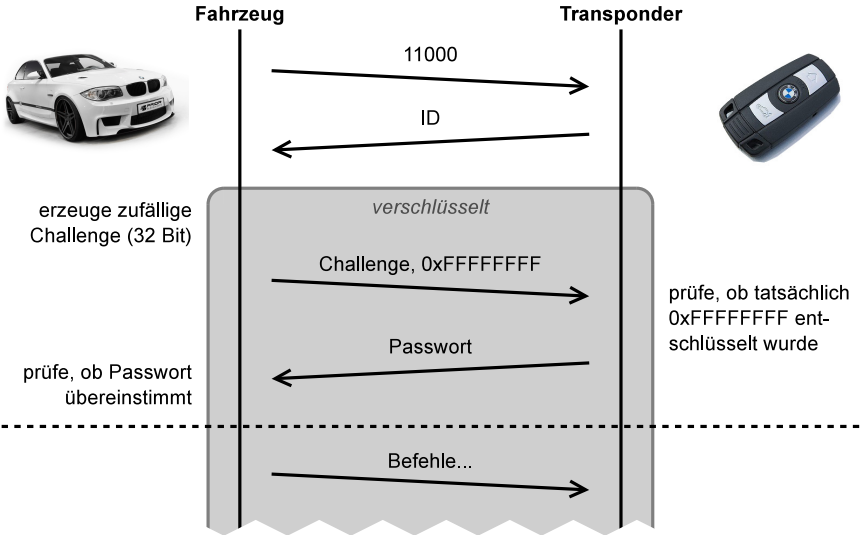
- Jeder Transponder eigene ID + Schlüssel, registriert im Fahrzeug
- Je Fahrzeug zusätzlich geheimes Passwort

Speicher im Transponder

- 8 Register mit je 32 Bit
- enthält seine ID, seinen Schlüssel, gemeinsames Passwort

Nr.	Speicherinhalt (je 32 Bit)	
0	ID	
1	Schlüssel [0...31]	
2	Schlüssel [32...47]	(unbenutzt)
3	Passwort	
4	(herstellerspezifisch)	
5	(herstellerspezifisch)	
6	(herstellerspezifisch)	
7	(herstellerspezifisch)	

Funktionsweise von Hitag2 – Authentifikation



- Befehle sind 5 Bit lang

Bitfolge	Befehl	Erklärung	Antwort
11xxx	read	Speicherbereich xxx lesen	Speicherinhalt
01xxx	<u>read</u>	Speicherbereich xxx lesen	Speicherinhalt invertiert
10xxx	write	Speicherbereich xxx schreiben	–
00***	halt	Transponder “schlafen legen”	–

- Befehle sind 5 Bit lang

Bitfolge	Befehl	Erklärung	Antwort
11xxx	read	Speicherbereich xxx lesen	Speicherinhalt
01xxx	read	Speicherbereich xxx lesen	Speicherinhalt invertiert
10xxx	write	Speicherbereich xxx schreiben	–
00***	halt	Transponder “schlafen legen”	–

+ mindestens 1 Wiederholung bitinvertiert, z.B.

- 1111100000 ⇒ Speicherbereich 7 lesen
- 111110000011111 ⇒ Speicherbereich 7 lesen

... keine obere Grenze

- Befehle sind 5 Bit lang

Bitfolge	Befehl	Erklärung	Antwort
11xxx	read	Speicherbereich xxx lesen	Speicherinhalt
01xxx	read	Speicherbereich xxx lesen	Speicherinhalt invertiert
10xxx	write	Speicherbereich xxx schreiben	–
00***	halt	Transponder “schlafen legen”	–

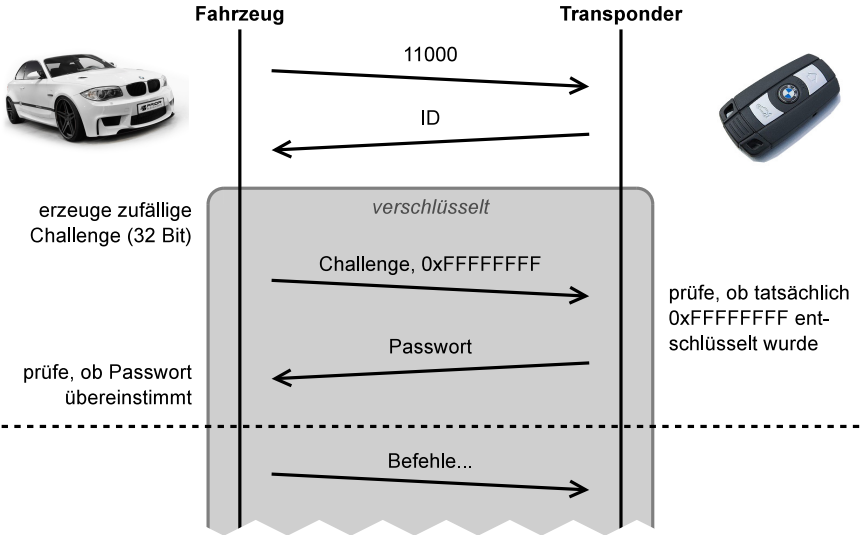
+ mindestens 1 Wiederholung bitinvertiert, z.B.

- 1111100000 ⇒ Speicherbereich 7 lesen
- 111110000011111 ⇒ Speicherbereich 7 lesen

... keine obere Grenze

- **Wichtig:** Lesebefehle kennzeichnen sich durch eine 32 Bit Antwort
- Zugriffsrechte auf Bereiche 0 bis 3 (ID, Schlüssel, PW) konfigurierbar

Funktionsweise von Hitag2 – Authentifikation

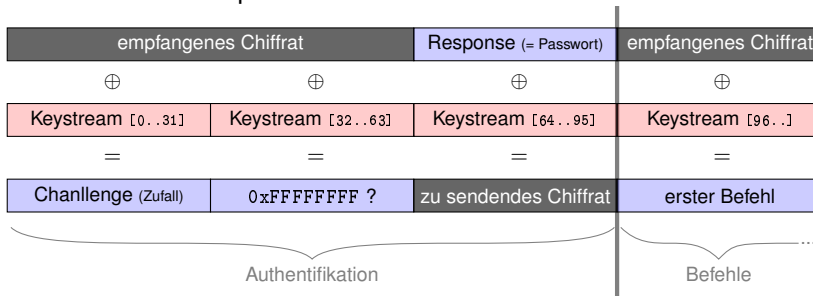


Ver- und Entschlüsselung

- beide Seiten konstruieren identischen Keystream
- bitweises XOR zur Ver- und Entschlüsselung
(vgl. *One Time Pad* und *OFB*)

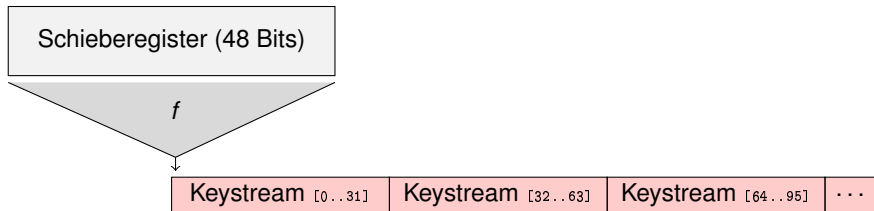
Ver- und Entschlüsselung

- beide Seiten konstruieren identischen Keystream
- bitweises XOR zur Ver- und Entschlüsselung
(vgl. *One Time Pad* und *OFB*)
- aus Sicht des Transponders:



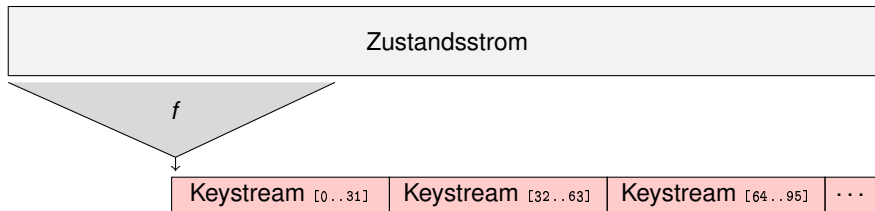
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)



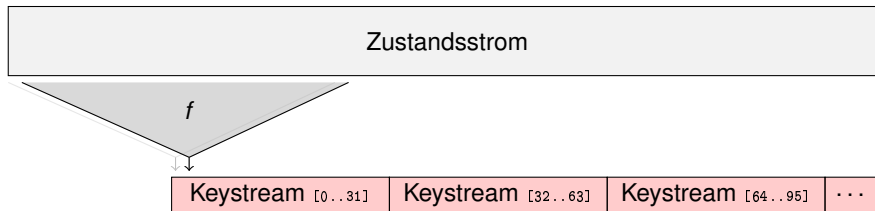
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)



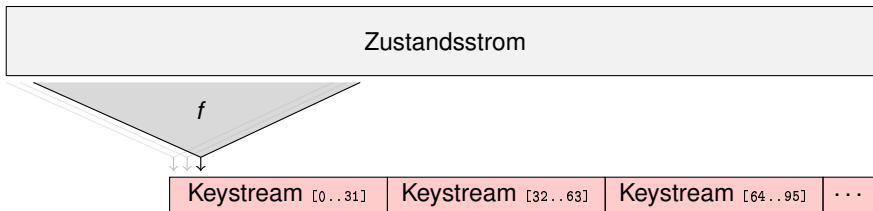
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)



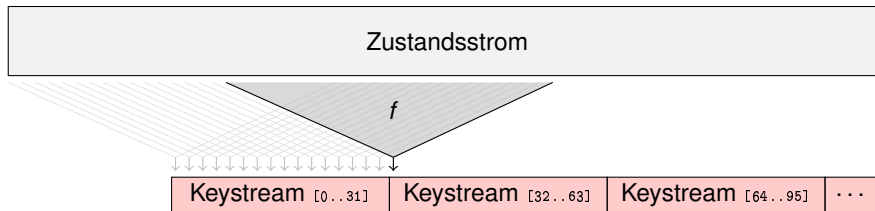
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)



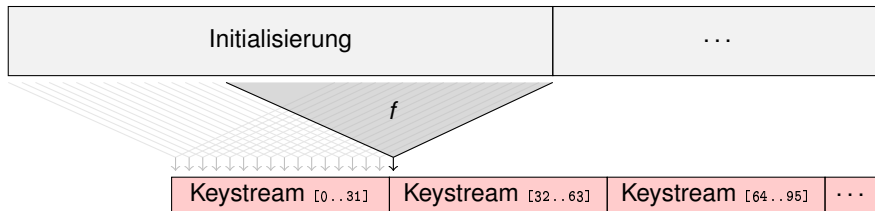
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)



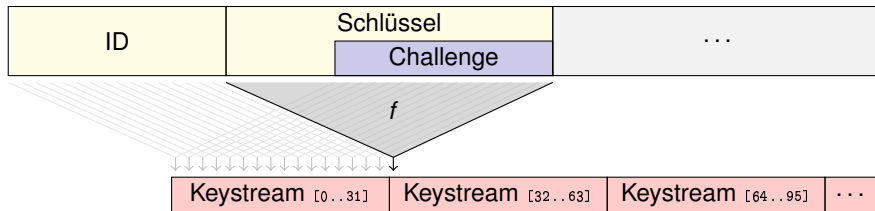
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)



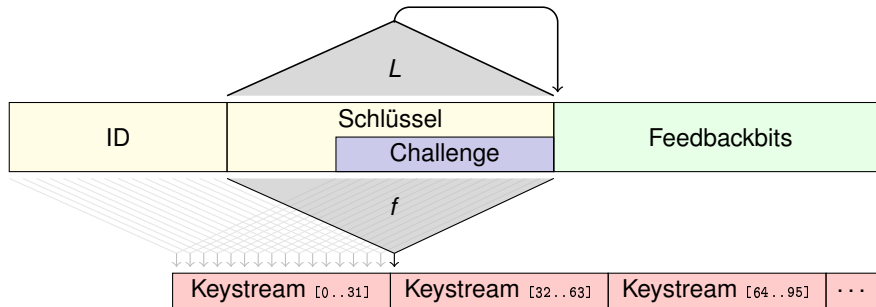
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)
- für die ersten 32 Bit Keystream: Challenge fließt in Registerinhalt ein (gleichzeitig mit Ver-/Entschlüsseln dieser)



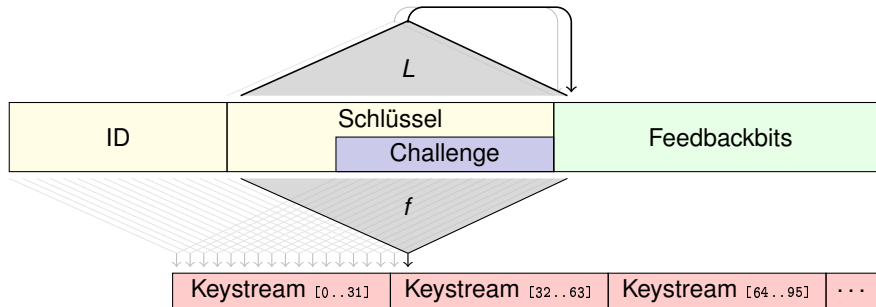
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)
- für die ersten 32 Bit Keystream: Challenge fließt in Registerinhalt ein (gleichzeitig mit Ver-/Entschlüsseln dieser)
- anschließend: konstruiere neue Bits mit Feedbackfunktion L



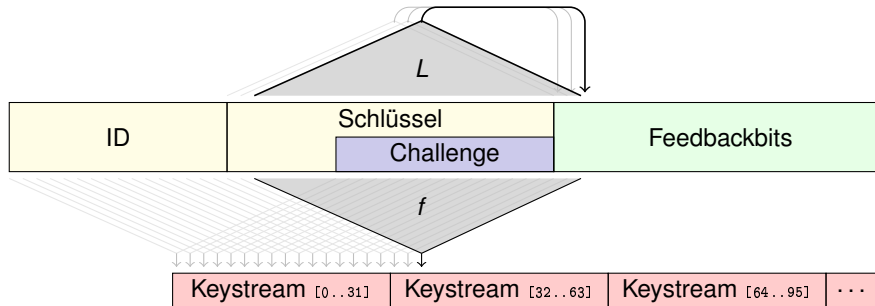
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)
- für die ersten 32 Bit Keystream: Challenge fließt in Registerinhalt ein (gleichzeitig mit Ver-/Entschlüsseln dieser)
- anschließend: konstruiere neue Bits mit Feedbackfunktion L



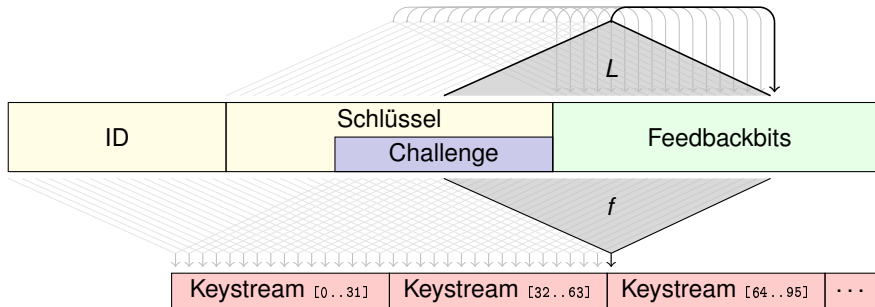
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)
- für die ersten 32 Bit Keystream: Challenge fließt in Registerinhalt ein (gleichzeitig mit Ver-/Entschlüsseln dieser)
- anschließend: konstruiere neue Bits mit Feedbackfunktion L



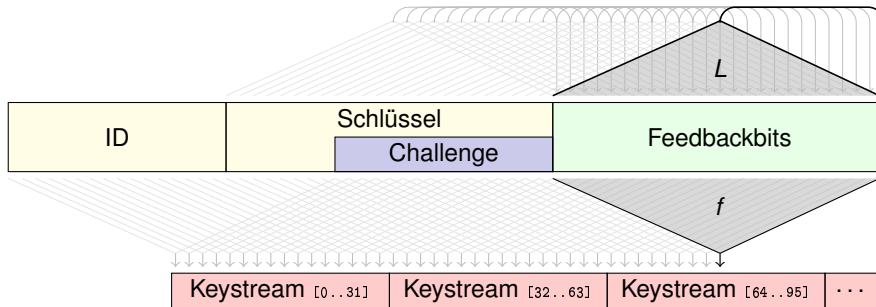
Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)
- für die ersten 32 Bit Keystream: Challenge fließt in Registerinhalt ein (gleichzeitig mit Ver-/Entschlüsseln dieser)
- anschließend: konstruiere neue Bits mit Feedbackfunktion L



Erzeugung des Keystreams

- 48 Bit großes Schieberegister (“Zustand”)
- für die ersten 32 Bit Keystream: Challenge fließt in Registerinhalt ein (gleichzeitig mit Ver-/Entschlüsseln dieser)
- anschließend: konstruiere neue Bits mit Feedbackfunktion L



Angriffe auf Hitag2 – Übersicht

	Methode 1	Methode 2	Methode 3
Passiv Kommunikation mithören (z.B. Abschließen)	X	X	(X) je nach Hersteller
Aktiv mit Transponder kommunizieren	X (kurz)	X (30 Sekunden)	-
Aktiv mit Fahrzeug (erfolglos) kommunizieren	-	-	X 136 Versuche
Benötigte Rechenleistung (vor Ort)	gering	mittel	hoch
Dauer (ca.)	wenige Sek.	1 Min.	wenige Min.
weitere Bemerkungen	setzt unsichere Konf. von Seiten des Herstellers voraus	Berechnung gut parallelisierbar	Berechnung gut parallelisierbar

Angriffe auf Hitag2 – Übersicht

	Methode 1	Methode 2	Methode 3
Passiv Kommunikation mithören (z.B. Abschließen)	X	X	(X) je nach Hersteller
Aktiv mit Transponder kommunizieren	X (kurz)	X (30 Sekunden)	-
Aktiv mit Fahrzeug (erfolglos) kommunizieren	-	-	X 136 Versuche
Benötigte Rechenleistung (vor Ort)	gering	mittel	hoch
Dauer (ca.)	wenige Sek.	1 Min.	wenige Min.
weitere Bemerkungen	setzt unsichere Konf. von Seiten des Herstellers voraus	Berechnung gut parallelisierbar	Berechnung gut parallelisierbar

Ausgenutzte Schwächen

- Aufgezeichnete Challenge lässt sich auf Transponder “replayen”
- Gleiche Challenge \Rightarrow gleicher Keystream
- Einige Fahrzeughersteller schützen Speicher nicht vor Auslesen
- Redundanz eines Befehls darf beliebig lang sein

Ausgenutzte Schwächen

- Aufgezeichnete Challenge lässt sich auf Transponder “replayen”
- Gleiche Challenge \Rightarrow gleicher Keystream
- Einige Fahrzeughersteller schützen Speicher nicht vor Auslesen
- Redundanz eines Befehls darf beliebig lang sein

Idee

- Versuche Lesebefehl für Speicherbereich der ID (11000) zu erraten
- Haben wir ihn gefunden, kennen wir einen Teil des Keystreams
- Wir senden ihn mit mehr Redundanz und erhalten mehr Keystream
- \Rightarrow beliebig viel Keystream auslesbar
- Senden dann Lesebefehl für kritischen Speicher, entschlüssle Antwort

Beliebigen Lesebefehl (*1***) finden

- Sende zufällige Befehle (wegen Redundanz 10 Bit lang)
- War es ein Lesebefehl (*1***), erfolgt eine 32 Bit lange Antwort
- Im Schnitt 64 Versuche benötigt (16 aus 1024 Möglichkeiten $\Rightarrow \frac{1}{64}$)

Beliebigen Lesebefehl (*1***) finden

- Sende zufällige Befehle (wegen Redundanz 10 Bit lang)
- War es ein Lesebefehl (*1***), erfolgt eine 32 Bit lange Antwort
- Im Schnitt 64 Versuche benötigt (16 aus 1024 Möglichkeiten $\Rightarrow \frac{1}{64}$)

Bestimmten Lesebefehl (11000) finden

- Das ist (unter Lesebefehlen) mit Wahrscheinlichkeit $\frac{1}{16}$ der Fall
- Nehme den Erfolgsfall an (Falls später Widerspruch \Rightarrow suche weiter)
- Als Antwort erhalten wir die verschlüsselte ID

Beliebigen Lesebefehl (*1***) finden

- Sende zufällige Befehle (wegen Redundanz 10 Bit lang)
- War es ein Lesebefehl (*1***), erfolgt eine 32 Bit lange Antwort
- Im Schnitt 64 Versuche benötigt (16 aus 1024 Möglichkeiten $\Rightarrow \frac{1}{64}$)

Bestimmten Lesebefehl (11000) finden

- Das ist (unter Lesebefehlen) mit Wahrscheinlichkeit $\frac{1}{16}$ der Fall
- Nehme den Erfolgsfall an (Falls später Widerspruch \Rightarrow suche weiter)
- Als Antwort erhalten wir die verschlüsselte ID
- Finde $10 + 32 = 42$ Keystreambits, da Klartext nun bekannt

Beliebigen Lesebefehl (*1***) finden

- Sende zufällige Befehle (wegen Redundanz 10 Bit lang)
- War es ein Lesebefehl (*1***), erfolgt eine 32 Bit lange Antwort
- Im Schnitt 64 Versuche benötigt (16 aus 1024 Möglichkeiten $\Rightarrow \frac{1}{64}$)

Bestimmten Lesebefehl (11000) finden

- Das ist (unter Lesebefehlen) mit Wahrscheinlichkeit $\frac{1}{16}$ der Fall
- Nehme den Erfolgsfall an (Falls später Widerspruch \Rightarrow suche weiter)
- Als Antwort erhalten wir die verschlüsselte ID
- Finde $10 + 32 = 42$ Keystreambits, da Klartext nun bekannt
- Sende Befehl erneut mit 7 Redundanzen (benötigt 40 Keystreambits)
- Finde $40 + 32 = 72$ Keystreambits, entdecke ggf. Widerspruch

Angriffe auf Hitag2 – Übersicht

	Methode 1	Methode 2	Methode 3
Passiv Kommunikation mithören (z.B. Abschließen)	X	X	(X) je nach Hersteller
Aktiv mit Transponder kommunizieren	X (kurz)	X (30 Sekunden)	-
Aktiv mit Fahrzeug (erfolglos) kommunizieren	-	-	X 136 Versuche
Benötigte Rechenleistung (vor Ort)	gering	mittel	hoch
Dauer (ca.)	wenige Sek.	1 Min.	wenige Min.
weitere Bemerkungen	setzt unsichere Konf. von Seiten des Herstellers voraus	Berechnung gut parallelisierbar	Berechnung gut parallelisierbar

Angriffe auf Hitag2 – Übersicht

	Methode 1	Methode 2	Methode 3
Passiv Kommunikation mithören (z.B. Abschließen)	X	X	(X) je nach Hersteller
Aktiv mit Transponder kommunizieren	X (kurz)	X (30 Sekunden)	-
Aktiv mit Fahrzeug (erfolglos) kommunizieren	-	-	X 136 Versuche
Benötigte Rechenleistung (vor Ort)	gering	mittel	hoch
Dauer (ca.)	wenige Sek.	1 Min.	wenige Min.
weitere Bemerkungen	setzt unsichere Konf. von Seiten des Herstellers voraus	Berechnung gut parallelisierbar	Berechnung gut parallelisierbar

Angriffe auf Hitag2 – Übersicht

	Methode 1	Methode 2	Methode 3
Passiv Kommunikation mithören (z.B. Abschließen)	X	X	(X) je nach Hersteller
Aktiv mit Transponder kommunizieren	X (kurz)	X (30 Sekunden)	-
Aktiv mit Fahrzeug (erfolglos) kommunizieren	-	-	X 136 Versuche
Benötigte Rechenleistung (vor Ort)	gering	mittel	hoch
Dauer (ca.)	wenige Sek.	1 Min.	wenige Min.
weitere Bemerkungen	setzt unsichere Konf. von Seiten des Herstellers voraus	Berechnung gut parallelisierbar	Berechnung gut parallelisierbar

Ausgenutzte Schwächen

- Nur 48 Bit Schlüssellänge
- Filterfunktion f hängt nur manchmal von den letzten 14 Bits ab
- Die ersten 16 Schlüsselbits werden nicht mit Challenge kombiniert

Ausgenutzte Schwächen

- Nur 48 Bit Schlüssellänge
- Filterfunktion f hängt nur manchmal von den letzten 14 Bits ab
- Die ersten 16 Schlüsselbits werden nicht mit Challenge kombiniert

Idee (erstmal sehr grob...)

- Führe einige Authentifikationsversuche durch
- Betrachte alle 2^{34} Registerinhalte, deren letzte 14 Bits 0 sind
- Ausschlussverfahren: Finde Registerinhalte, deren produzierter Keystream das korrekte 0xFFFFFFFF entschlüsseln würde
- Unter allen Schlüsselkandidaten (üblicherweise < 100) bruteforcen

Schlüssel zerlegen

- Teile Schlüsselraum in $16 + 18 + 14$ Bits auf: K_L, K_M, K_R
- K_L beeinflusst Registerinhalt unabhängig von Challenge
- K_R beeinflusst bei manchen Challenges das 32. Keystreambit *nicht*

Schlüssel zerlegen

- Teile Schlüsselraum in $16 + 18 + 14$ Bits auf: K_L, K_M, K_R
- K_L beeinflusst Registerinhalt unabhängig von Challenge
- K_R beeinflusst bei manchen Challenges das 32. Keystreambit *nicht*

Vorgehen: Für alle linken Teilschlüssel (2^{16} Iterationen):

Schlüssel zerlegen

- Teile Schlüsselraum in $16 + 18 + 14$ Bits auf: K_L, K_M, K_R
- K_L beeinflusst Registerinhalt unabhängig von Challenge
- K_R beeinflusst bei manchen Challenges das 32. Keystreambit *nicht*

Vorgehen: Für alle linken Teilschlüssel (2^{16} Iterationen):

- Initialisiere leere Tabelle $\mathbb{F}^{18} \rightarrow \mathbb{F}$ // $\mathbb{F} = \{0, 1\}$

Schlüssel zerlegen

- Teile Schlüsselraum in $16 + 18 + 14$ Bits auf: K_L, K_M, K_R
- K_L beeinflusst Registerinhalt unabhängig von Challenge
- K_R beeinflusst bei manchen Challenges das 32. Keystreambit *nicht*

Vorgehen: Für alle linken Teilschlüssel (2^{16} Iterationen):

- Initialisiere leere Tabelle $\mathbb{F}^{18} \rightarrow \mathbb{F}$ // $\mathbb{F} = \{0, 1\}$
- Für alle möglichen mittleren Registerteilinhalt Y (2^{18} Iterationen):
 - Falls restlicher Registerinhalt irrelevant (s.o.):
 - Setze Rest auf 0, errechne Keystreambits 0..17 (B) und Bit 32 (b_{32})
 - Trage in Tabelle ein: $Y \oplus B \mapsto b_{32}$ // $Y \oplus B = K_M \oplus \text{Challengechiff}$

Schlüssel zerlegen

- Teile Schlüsselraum in $16 + 18 + 14$ Bits auf: K_L, K_M, K_R
- K_L beeinflusst Registerinhalt unabhängig von Challenge
- K_R beeinflusst bei manchen Challenges das 32. Keystreambit *nicht*


Vorgehen: Für alle linken Teilschlüssel (2^{16} Iterationen):

- Initialisiere leere Tabelle $\mathbb{F}^{18} \rightarrow \mathbb{F}$ // $\mathbb{F} = \{0, 1\}$
- Für alle möglichen mittleren Registerteilinhalt Y (2^{18} Iterationen):
 - Falls restlicher Registerinhalt irrelevant (s.o.):
 - Setze Rest auf 0, errechne Keystreambits 0..17 (B) und Bit 32 (b_{32})
 - Trage in Tabelle ein: $Y \oplus B \mapsto b_{32}$ // $Y \oplus B = K_M \oplus \text{Challengegechiffert}$
- Für alle möglichen K_M und aufgezeichneten Challenges ($2^{18} \cdot 136$):
 - Ermittle $K_M \oplus \text{Challengegechiffert}$, schlage in Tabelle nach
 - Vorhanden, aber b_{32} passt nicht auf Fahrzeugantwort $\Rightarrow K_M$ war falsch

Hitag2 äußerst unsicher!

- Designschwächen wären vermeidbar gewesen:
 - Begrenzung der Redundanzkodierung
 - Befehle nur mit Jumper akzeptieren
 - Mehr als 48 Bit
 - **Standard wie AES statt Eigenentwicklung**
- Bessere Wegfahrsperre (HitagAES) bereits im Angebot
(KFZ-Hersteller kaufen trotzdem noch immer Hitag2)
- RFID-Schutzhülle schützt vor Angriffen 1 und 2
- Fahrzeughersteller kann Angriff 3 deutlich erschweren:
 - Sperre nach z.B. 3 fehlgeschlagenen Zugriffen
 - Danach Entsperren mit mechanischem Schlüssel

Im Falle eines Diebstahls zahlt Versicherung

-  Roel Verdult, Flavio D. Garcia, and Josep Balasch, *Gone in 360 seconds: Hijacking with hitag2*, Proceedings of the 21st USENIX Conference on Security Symposium (Berkeley, CA, USA), Security'12, USENIX Association, 2012, pp. 37–37.