(Fälschungs-) Sicherheit bei RFID

Vortrag: Oliver Zweifel

Betreuer: Christian Floerkemeier

- Vorher: Privatsphäre
  - Location Privacy
  - Data Privacy
- Jetzt: Schutz der Tags gegen Fälschung

Übersicht

- Einführung
- Strengthening EPC Tags Against Cloning

  (Ari Juels)
- Security Analysis of a Cryptographically-Enabled RFID Device

  (Steve Bono, Matthew Green, Adam Stubblefield, Ari Juels, Avi Rubin, Michael Szydlo)

Einführung

Grundfunktionalität eines RFID-Tags:
Auf Anfrage mit eigener ID antworten

⇒ Leichte Nachahmung möglich durch

- Programmierbare Tags
- Tag-Simulation

Trotzdem planen viele Firmen (z. B. Hersteller von Luxusgütern) Tags als Echtheitszertifikate für ihre Produkte einzuführen.

Fälschungssicherheit kann mit komplexeren Tags erhöht werden.

→ ist aber mit wesentlich höheren Kosten verbunden

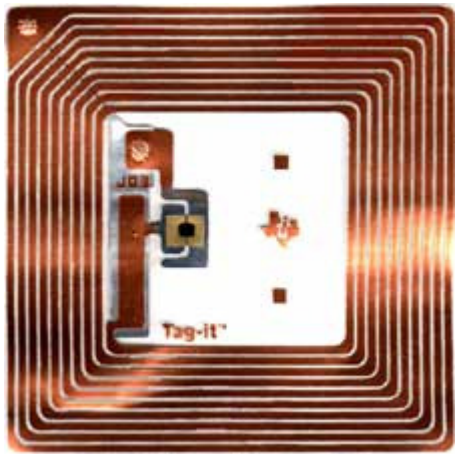
Strengthening EPC Tags Against Cloning

Ziel des Projekts

- Schutz von einfachen EPC Tags gegen Klonen, das mittels Daten, die durch Scannen eines Tags (Skimming) in Erfahrung gebracht werden, realisiert wird.

EPC Tags

- EPC = Electronic Product Code
- Nachfolger des Barcodes
- EPC besteht aus
  - Eindeutiger ID
  - Weiteren Daten (Hersteller, Produkttyp etc.)
- Standardisiert



EPCglobal Standard

- EPCglobal Class-1 Generation-2 UHF Tags müssen über eine Kill-Funktion verfügen.

Kill-Befehl

- „Zerstört“ Tag
- Wird nur akzeptiert, wenn er zusammen mit einer gültigen 32-Bit Kill-PIN geschickt wird
- Kill-Aktion kann nur ausgeführt werden, wenn genügend Energie im Tag vorhanden (ansonsten wird eine Fehlermeldung zurückgegeben)

  Der EPCglobal Standard bestimmt aber nicht, wie viel Energie vorhanden sein muss, damit Kill-Aktion ausgeführt werden kann

  ⇒ Trick möglich:

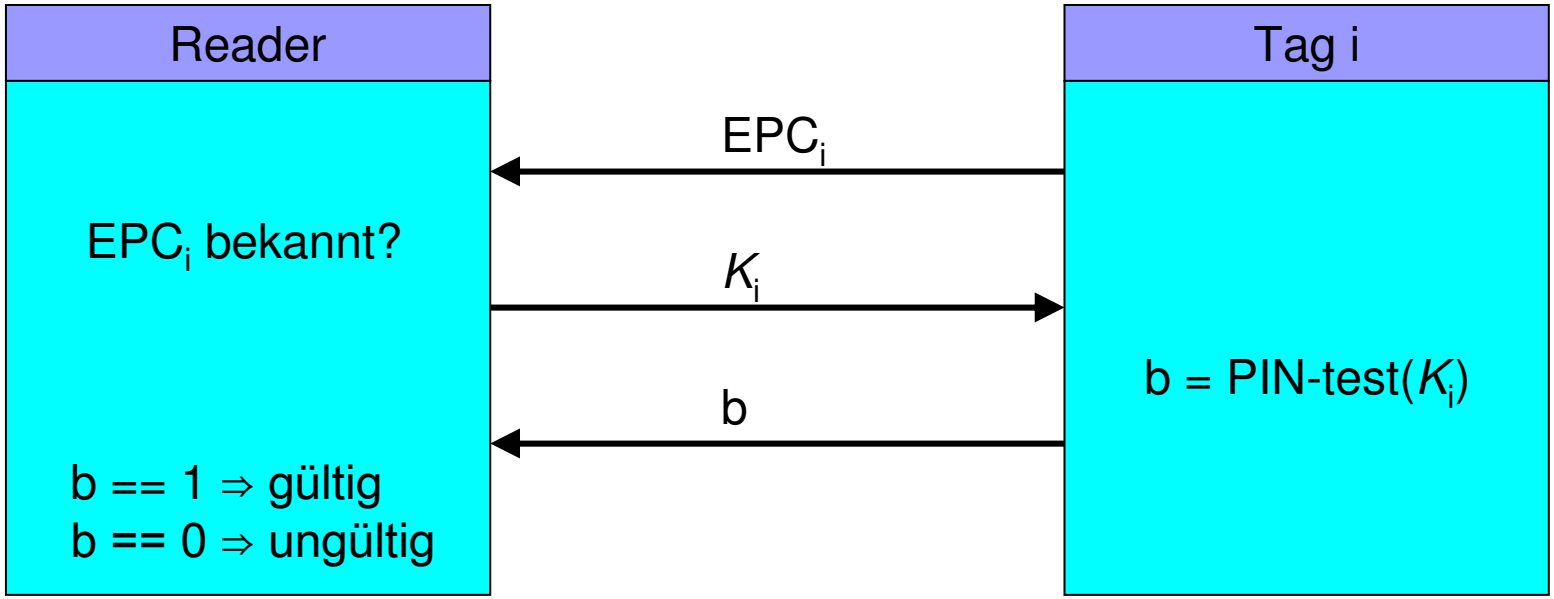
  Auch wenn Tag immer vorgibt, zuwenig Energie zur Verfügung zu haben, widerspricht dieses Verhalten nicht dem EPCglobal Standard. Aber der Kill-Befehl kann auf diese Weise zur Tag-Authentifizierung zweckentfremded werden.

  Rollenumkehr: Reader-Authentifizierung → Tag-Authentifizierung

Algorithmus 1: Einfacher Tag-Authenifizierungs-Algorithmus

Wir definieren Funktion PIN-test(K_i) wie folgt:

$$\text{PIN-test}(K_i) = \begin{cases} 1, \text{ wenn } K_i \text{ korrekter Kill-PIN für Tag i} \\ 0, \text{ sonst} \end{cases}$$



Überlegungen zu Algorithmus 1

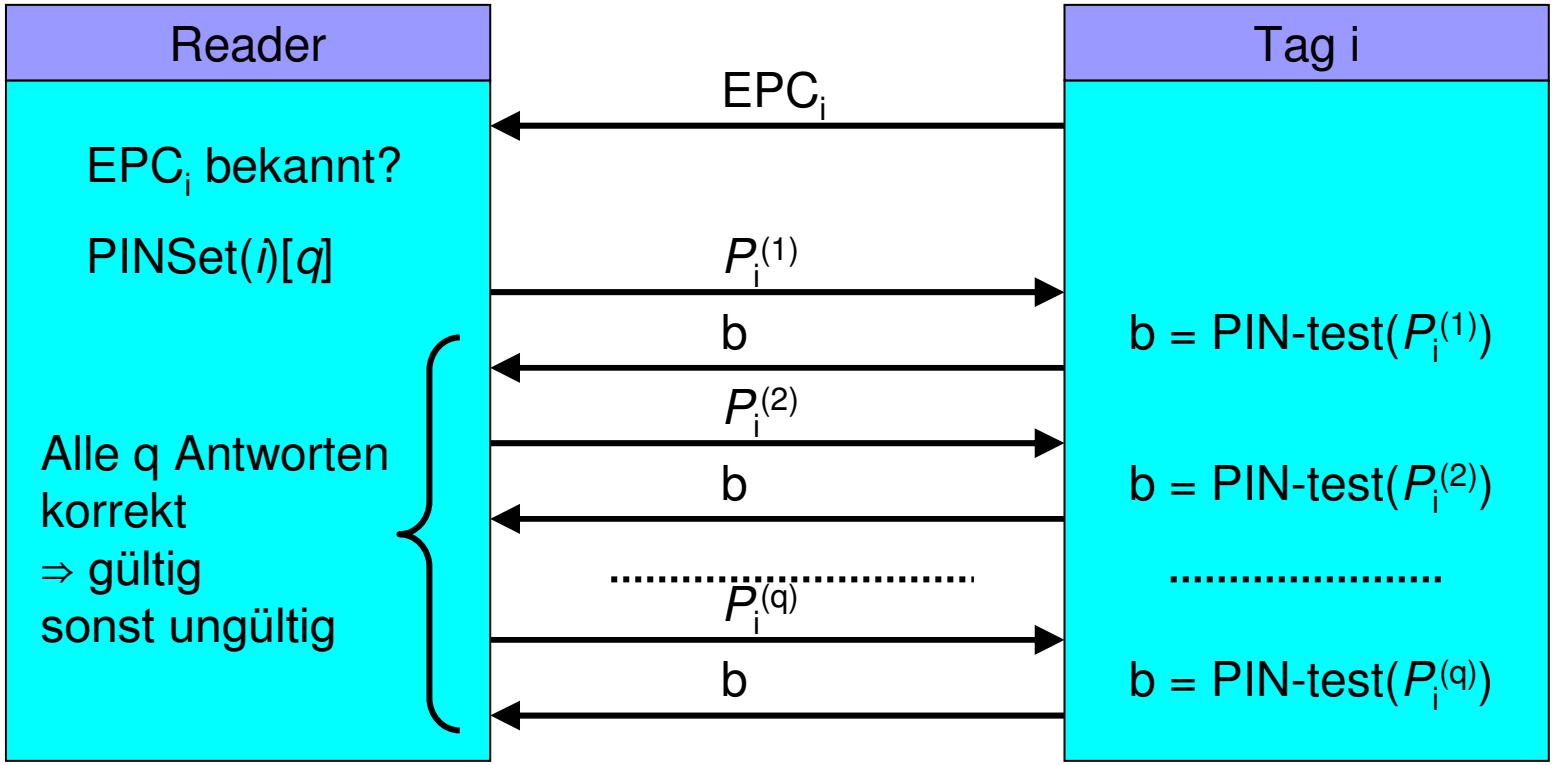
- Funktioniert nur bei EPCglobal konformen Tags.
- Mittels Skimming kann zwar die EPC leicht in Erfahrung gebracht werden, nicht aber den Kill-PIN.

  ⇒ Angreifer muss Kill-PIN erraten

  (bei 32-Bit PIN: $2^{32} \approx 4$ Mia Möglichkeiten)

Algorithmus 2

Trick:
Generiere Set aus falschen Kill-PINs und füge an beliebiger Stelle den wahren Kill-PIN ein.
→ Funktion PINSet(i)[q] = (j, {$P_i^{(1)}$, $P_i^{(2)}$, … , $P_i^{(q)}$})



Überlegungen zu Algorithmus 2

- Funktioniert auch bei nicht EPCglobal konformen Tags.
- $\{P_i^{(n)}\}$ muss bei jeder Ausführung des Algorithmus gleich sein.
- Angreifer kann raten, welcher PIN im Set der Wahre ist. Bei einer Set-Grösse q gelingt ihm das mit einer Wahrscheinlichkeit von 1/q.
- Bei grossem q langsam

  $\Rightarrow$ Tradeoff zwischen hoher Sicherheit und Geschwindigkeit

Zusammenfassung

Schutz von herkömmlichen EPC Tags gegen Klonen.
Trick: Rollenumkehr bei Kill-Befehl

- Algorithmus 1:

  simpel, funktioniert aber nur bei EPCglobal konformen Tags
- Algorithmus 2:

  funktioniert auch bei nicht EPCglobal konformen Tags

Ungelöste Sicherheitsprobleme

- Eindringen in Datenbank
- Reverse Engineering
- Man-in-the-middle Attacke
- Abhören

Security Analysis of a Cryptographically-Enabled RFID Device

Ziel des Projekts

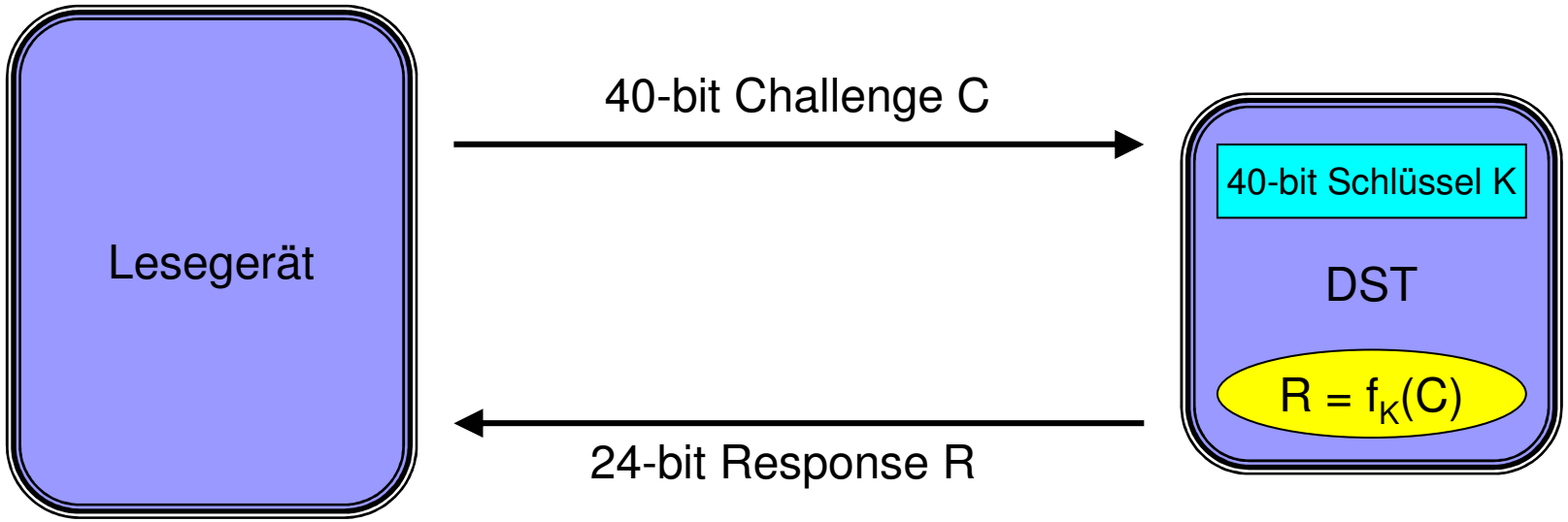
- Sicherheitsmängel des TI DST aufzeigen
- Chip-Entwickler auf Problematik sensibilisieren, sodass in Zukunft solche Probleme im Voraus vermieden werden können

Nicht Ziel: Hack-Anleitung

Texas Instruments DST

- DST = Digital Signature Transponder
- Unterstützt Authentifizierungs-Protokoll (Challenge-Response)
- Typische Einsatzgebiete
  - KFZ-Wegfahrsperren
  - Elektronisches Zahlungsmittel
    z. B. ExxonMobil SpeedPass™

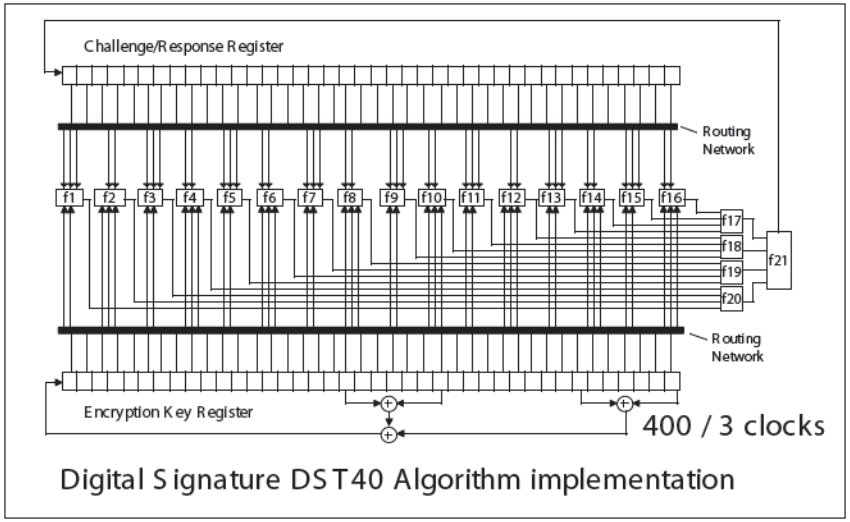
TI DST Challenge-Response Authentifizierung



Vorgehensweise

1. Reverse Engineering (Bestimmung des Algorithmus)
2. Key Cracking
3. Tag-Simulation

Reverse Engineering

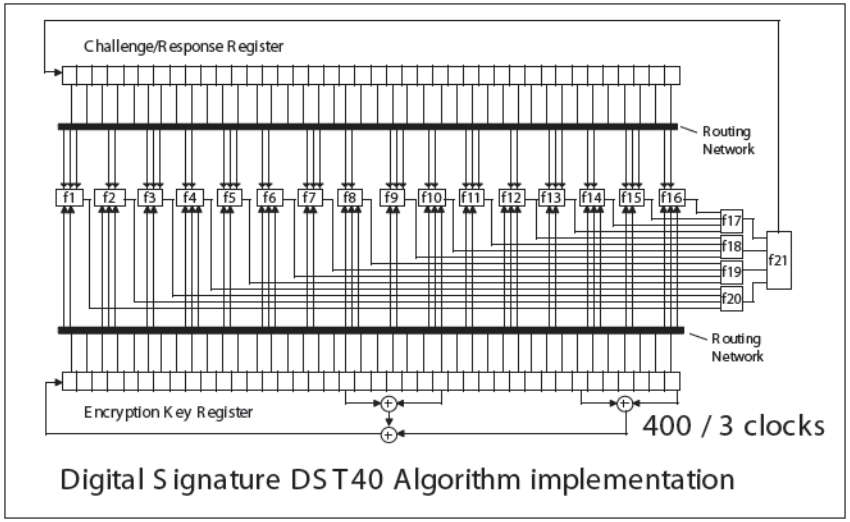


feedback shift register

Hilfsmittel:

- Grobe schematische Darstellung der Verschlüsselungslogik
- „Black-Box“ Zugriff auf echten DST

Reverse Engineering



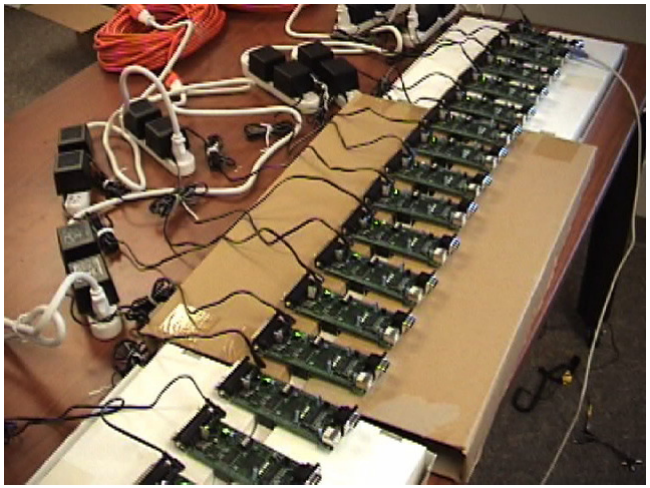
Key = 0-Vektor
C' = alle Bits in Challenge C ausser dem letzten

Inhalt des C/R-Registers nach einer Runde:
$C_0 = 0|\ C'$ oder $C_1 = 1|\ C'$

analog für C_1

Annahme C_0 trifft zu:
⇒ wenn man kompletten Algo auf C_0 (statt auf C) anwendet, bekommt man dasselbe Resultat wie bei der Anwendung auf C, einfach um ein Bit nach rechts verschoben.
Falls Annahme falsch, dann gibt es ein sehr unterschiedliches Resultat

Key Cracking



- Schlüssel ist nur 40 Bit gross
- Challenge 40 Bit, aber Response nur 24 Bit (least significant 24 Bit)
  ⇒ damit Schlüssel eindeutig berechnet werden kann sind daher 2 abgehörte Challenge-Response-Paare erforderlich
- Array von 16 FPGAs knacken Schlüssel in weniger als einer Stunde (Brute Force)

Tag-Simulation

Laptop + DAC Board + Antenne

- Erfolgreiches Knacken einer KFZ-Wegfahrsperre
- Tanken mit der „SpeedPass™“-Karte

Weitere Überlegungen

- Warum haben TI ihren Chip nicht sicherer gemacht?
  - Laufzeit
  - Kosten

Meine Einschätzung

- Problematik abhängig vom Einsatzgebiet.

  Beim Autoschlüssel ist sie weit weniger gravierend als beim elektronischen Zahlungsmittel.

Zusammenfassung

1. Algorithmus herausfinden (Reverse Engineering)
2. Schlüssel knacken (Brute Force, da nur 40-Bit Schlüssel)
3. Tag simulieren (mit relativ geringem Hardwareaufwand möglich)

Schlussfolgerung

- TI hätten Sicherheitsmängel vermeiden können, wenn sie eine grössere Schlüssellänge gewählt hätten.
  - Reisepass 56 Bit
- Experiment zeigt, dass die alleinige Verheimlichung eines Algorithmus kein zuverlässiger Weg zu mehr Sicherheit ist.

Das war‘s…

- Fragen?
- Kritik?