Fälschungssicherheit für RFID-Tags

Seminar „Smarte Objekte und smarte Umgebungen“

Oliver Zweifel
zweifelo@student.ethz.ch

Betreuer: Christian Floerkemeier

Zusammenfassung. In dieser schriftlichen Ausarbeitung des Seminarvortrags zum Thema „Fälschungssicherheit für RFID-Tags“ werde ich auf die Fälschungsproblematik von RFID-Tags anhand zweier Papers eingehen. Beim Paper „Strengthening EPC Tags Against Cloning“ [1] wird gezeigt, wie billige, in ihrer Funktionalität einfache Tags bis zu einem gewissen Grad fälschungssicher gemacht werden können. Im Paper „Security Analysis of a Cryptographically-Enabled RFID Device“ [2] wird auf Sicherheitsmängel eines teuren, als fälschungssicher angepriesenen Tags eingegangen.

1 Einleitung

Die Grundfunktionalität von RFID-Tags besteht darin, dass ein Tag auf die Anfrage eines Lesegeräts mit seiner gespeicherten Identifikationsnummer (ID) antwortet. Dieses Verhalten eines Tags kann von Angreifern, die einen Tag fälschen wollen, einfach nachgeahmt werden. Einerseits können Tags gekauft werden, bei denen im Nachhinein eine beliebige ID einprogrammiert werden kann. Solche Tags eignen sich hervorragend zum Klonen von Tags. Man braucht dazu lediglich die ID eines bereits existierenden Tags in ein solches einzuprogrammieren. Andererseits können Tags simuliert werden, indem man einen Rechner, der mit einer Antenne verbunden ist, so programmiert, dass er sich wie der zu simulierende Tag verhält. Hervorzuheben ist, dass dabei auf einen physischen Tag gänzlich verzichtet werden kann. Trotz dieser Sicherheitsmängel planen viele Firmen (gerade Hersteller von Luxusgütern), einfache Tags als Echtheitszertifikate für ihre Produkte zu verwenden. Man kann zwar Tags (weitgehend) fälschungssicher machen, allerdings ist das in den meisten Fällen nur möglich, indem man die Tags mit komplexer kryptographischer Funktionaltät versieht. Dies führt jedoch zu erheblich höheren Kosten in der Tag-Produktion [1], und auch der Bandbreitenbedarf der Kommunikation zwischen Tag und Lesegerät steigt beim Einsatz aufwändiger kryptographischer Protokolle beträchtlich. Im Folgenden möchte ich die Problematik anhand zweier, ziemlich unterschiedlicher Papers genauer erläutern.

2 Strengthening EPC Tags Against Cloning [1]

In diesem Paper wird gezeigt, wie man einfache, billige EPC-Tags gegen Klonen bzw. Fälschen schützen kann. Dabei ist dasjenige Klonen gemeint, das realisiert wird, indem man einen Tag auf alle zum Klonen notwendigen Informationen abscannt. Dieser Vorgang wird auch *Skimming* genannt. Der EPC (Electronic Product Code) kann als Nachfolger des klassischen Barcodes betrachtet werden. Wie der Barcode speichert auch der EPC allgemeine Produktinformationen wie z. B. Hersteller, Produkttyp etc. Im Gegensatz zum Barcode beinhaltet der EPC aber noch eine ID, wodurch alle EPC-Tags eindeutig unterschieden werden können. EPC-Tags sind passive Tags, d. h. sie verfügen über keine eigene Energieversorgung und müssen deshalb vom Lesegerät gespiesen werden.

2.1 Rollenumkehr beim Kill-Befehl

Aufgrund ihrer eingeschränkten Funktionalität ist es nicht einfach, EPC-Tags gegen Klonen zu schützen. Die Autoren dieses Papers haben dennoch einen Weg gefunden.

EPC-Tags sind nach dem EPCglobal Class-1 Generation-2 UHF Standard (im Folgenden durch EPCglobal Standard abgekürzt) standardisiert. Dieser Standard schreibt u. a. vor, dass alle EPC-Tags eine sog. Kill-Funktion unterstützen müssen. Diese Kill-Funktion wird vom Tag ausgeführt, indem das Lesegerät einen Kill-Befehl an den Tag schickt. Dabei wird der Tag „zerstört“. Hier ist nicht eine Zerstörung im physikalischen Sinne gemeint, sondern eine permanente Deaktivierung des Tags, sodass dieser nie wieder auf äussere Einflüsse reagieren kann. Damit nicht jeder Angreifer missbräuchlicherweise beliebige Tags eliminieren kann, muss sich das Lesegerät beim Tag authentifizieren, damit der Kill-Befehl vom Tag akzeptiert wird. Dies geschieht, indem der Tag einen Kill-Befehl nur dann akzeptiert, wenn er zusammen mit einer gültigen, 32 Bit grossen Kill-PIN gesendet wurde. Wie bereits oben erwähnt, handelt es sich bei den EPC-Tags um passive Tags. Aus diesem Grunde kann es vorkommen, dass ein Tag nicht mehr genügend Energie zur Verfügung hat, um die Kill-Funktion auszuführen. In diesem Fall wird vom Tag eine Fehlermeldung zurückgegeben. Der EPCglobal Standard spezifiziert aber nicht, wieviel Energie vorhanden sein muss, damit eine Kill-Funktion ausgeführt werden können muss. Diese Tatsache ermöglicht einen Trick: man könnte einen Tag mit einer abgeänderten Kill-Funktion ausstatten, die immer vorgibt, zuwenig Energie zu haben, damit der Tag nie zerstört werden kann. Ein solcher Tag würde dem EPCglobal Standard nicht widersprechen. Man kann auf diese Weise die Kill-PIN, die eigentlich zur Lesegerät-Authentifizierung beim Tag gedacht war, zur Tag-Authentifizierung beim Lesegerät zweckentfremden. Eine solche Zweckentfremdung führt zu einer Rollenumkehr zwischen Tag und Lesegerät. In Teilkapitel 2.2 wird anhand eines einfachen Algorithmus gezeigt, wie diese Rollenumkehr realisiert werden kann.

2.2 Ein einfacher Tag-Authentifizierungs-Algorithmus

Die in Teilkapitel 2.1 beschriebene, abgeänderte Kill-Funktion macht eigentlich nichts weiter, als zu überprüfen, ob die vom Lesegerät gesendete Kill-PIN korrekt ist. Wir definieren eine Funktion PIN-test, welche dieser abgeänderten Kill-Funktion entspricht:

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

Mit Hilfe der Funktion PIN-test kann jetzt ein einfacher Algorithmus zur Tag-Authentifizierung realisiert werden:

```
T:        T ← T_i
T → R:    T
R:        if T = T_x for some 1 ≤ x ≤ N then i ← x
                  else output "unknown tag" and halt
R → T:    PIN-test(K_i)
T → R:    b
R:        if b = 1 then output "valid"
                        else output "invalid"
```

„A → B“ steht für einen Datenfluss von der Einheit A zur Einheit B. „A:“ bedeutet, dass die nachfolgenden Operationen lokal bei der Einheit A ausgeführt werden. Zu Beginn des Algorithmus schickt der Tag seinen EPC an das Lesegerät, welches überprüft, ob dieser EPC bzw. Tag bekannt ist. Der Tag muss dem Lesegerät bekannt sein, da das Lesegerät die Kill-PIN des Tags kennen muss. Ist der Tag unbekannt, so wird er nicht akzeptiert. Ansonsten schickt jetzt das Lesegerät die passende Kill-PIN an den Tag, dieser wendet seine PIN-test Funktion darauf an und schickt anschliessend das Resultat zurück an das Lesegerät. Wenn dieses Resultat „1“ ist, dann wird der Tag akzeptiert. Denn die PIN-test Funktion ergibt nur „1“, wenn der Tag seinerseits die richtige Kill-PIN gespeichert hat. Eine „0“ als Resultat bedeutet, dass auf dem Tag eine falsche Kill-PIN gespeichert ist und es sich deshalb um eine Fälschung handeln muss. In diesem Fall wird der Tag nicht akzeptiert.

Der Algorithmus funktioniert aufgrund der Tatsache, dass der EPC eines Tags zwar leicht in Erfahrung gebracht werden kann, nicht aber dessen Kill-PIN. Denn die Kill-PIN wird vom Tag nie gegen aussen kommuniziert. Ein Angreifer, der ein Tag klonen möchte, könnte höchstens erraten, welche Kill-PIN er seiner Fälschung einprogrammieren sollte. Bei einer 32 Bit grossen Kill-PIN gibt es mehr als 4 Mia Möglichkeiten. Dass der Angreifer die Richtige errät, ist nahezu ausgeschlossen. Der Algorithmus hat aber auch einen gravierenden Nachteil: er funktioniert mit Tags, die zum EPCglobal Standard konform sind. Ein nicht standardkonformer Tag könnte über eine PIN-test Funktion verfügen, welche immer „1“ zurückgibt – egal auf welche Kill-PIN man sie anwendet. Ein solcher Tag würde vom Lesegerät immer akzeptiert werden.

2.3 Ein verbesserter Tag-Authentifizierungs-Algorithmus

Um den in Teilkapitel 2.2 vorgestellten Algorithmus auch auf Tags anwenden zu können, die nicht zum EPCglobal Standard konform sind, muss er ein wenig erweitert werden. Man kann dabei folgenden Trick anwenden: Das Lesegerät schickt anstelle der einzelnen Kill-PIN ein ganzes Set von vielen Kill-PINs an den Tag. Dieses Set besteht aus lauter falschen Kill-PINs, wobei an einer beliebigen Stelle die korrekte Kill-PIN steht. Wir definieren eine Funktion PINSet(i)[q], welche für den Tag i ein solches PIN-Set der Grösse q generiert. Damit kann der Algorithmus aus Teilkapitel 2.2 zu folgendem Algorithmus erweitert werden:

```
T:        T ← T_i
T → R:    T
R:        if T = T_x for some 1 ≤ x ≤ N then i ← x
                else output "unknown tag" and halt
R:        (j, {P_i^(1), P_i^(2), … , P_i^(q)}) ← PINSet(i)[q];
          M ← "valid";

          for n = 1 to q do
R → T:          PIN-test(P_i^(n))
T → R:          b
R:              if b = 1 and n ≠ j then M ← "invalid";
                if b = 0 and n = j then M ← "invalid";

R:        output M;
```

Der Beginn des Algorithmus bleibt unverändert. Wieder überprüft das Lesegerät zuerst, ob der Tag bekannt ist. Ist das der Fall, generiert jetzt aber das Lesegerät dieses PIN-Set. Danach sendet das Lesegerät die erste Kill-PIN des Sets an den Tag, dieser wendet seine PIN-test Funktion darauf an und liefert das Resultat an das Lesegerät zurück. Analog wird derselbe Vorgang für alle anderen Kill-PINs des Sets nacheinander durchgeführt. Am Schluss überprüft das Lesegerät, ob alle Antworten korrekt waren. D. h. genau an der Stelle, wo die korrekte Kill-PIN gesendet wurde, musste mit „1“ geantwortet worden sein und überall sonst mit „0“. Trifft dies zu, wird der Tag akzeptiert. Andernfalls wird der Tag als Fälschung deklariert und abgelehnt.

Wie bereits erwähnt, funktioniert dieser Algorithmus auch bei nicht standardkonformen Tags. Wenn der Algorithmus aber mehrmals auf denselben Tag angewendet wird, ist es wichtig, dass das PIN-Set unverändert bleibt. Ansonsten könnte ein Angreifer nachsehen, welche Kill-PIN in allen Sets vorkommt – und diese wäre ja dann genau die Korrekte. Natürlich kann hier ein Angreifer raten, an welcher Stelle er mit „1“ antworten soll. Aber bei einer Set-Grösse von q würde das nur mit einer Wahrscheinlichkeit von $1/q$ gelingen. Daraus resultiert, dass die Sicherheit des Algorithmus von der Set-Grösse q abhängt. Möchte man q gross wählen, um den Algorithmus möglichst sicher zu machen, muss man mit entsprechend längeren Laufzeiten rechnen. Denn für jede einzelne Kill-PIN des Pin-Sets muss eine Anfrage an den Tag stattfinden, und dieser muss seine PIN-test Funktion darauf anwenden. Deshalb muss man sich darüber im Klaren sein, ob die Sicherheit oder die Geschwindigkeit wichti-

ger ist. Wenn man aber die Anforderungen an den Algorithmus nicht darin sehen würde, das Klonen von Tags absolut zu unterbinden, sondern schon damit zufrieden wäre, dass der Algorithmus erkennt, ob geklonte Tags im Umlauf sind, würde eine Set-Grösse $q = 2$ bereits ausreichen. Angenommen, es sind mehrere geklonte Tags in Reichweite des Lesegeräts, und jeder von ihnen kann mit einer Wahrscheinlichkeit von 50% als Fälschung entlarvt werden, dann ist die Wahrscheinlichkeit, dass mindestens ein Tag entlarvt wird, sehr gross. Daraufhin kann man geeignete Massnahmen vornehmen.

2.4 Zusätzliche Überlegungen

Die in Teilkapitel 2.1 beschriebene Zweckentfremdung der Kill-Funktion ist in mehrerer Hinsicht problematisch. Es müssen spezielle Tags produziert werden, bei denen diese abgeänderte Kill-Funktion implementiert ist. Den ursprünglichen Zweck, nämlich die Möglichkeit, Tags zu zerstören, erfüllen diese Tags nicht mehr, obwohl dies in vielen Anwendungen nützlich sein kann. Und was passiert, wenn herkömmliche EPC-Tags in ein solches System gelangen würden? Diese würden gleich reihenweise vom Lesegerät eliminiert. Solche Betrachtungen machen klar, dass die vorgestellten Möglichkeiten, Tags gegen Klonen zu schützen, nicht kompromisslos realisiert werden können.

Weiter muss erwähnt werden, dass immer noch Klon-Angriffe möglich sind, gegen welche die hier vorgestellten Algorithmen machtlos sind. Gelänge es beispielsweise einem Angreifer, in die Datenbank, in der alle Kill-PINs gespeichert sind, einzudringen, so könnte er jeden beliebigen, bekannten Tag perfekt klonen. Es ist auch denkbar, dass man mittels Reverse Engineering den gespeicherten Kill-PIN eines Tags in Erfahrung bringen könnte. Oder eine „Man-in-the-middle“ Attacke, bei der sich der Angreifer in die Kommunikation zwischen dem Tag und dem Lesegerät einschleust. Schliesslich würde auch gewöhnliches Abhören der Kommunikation zwischen dem Tag und dem Lesegerät genügen, um den Kill-PIN des Tags herauszufinden. Allerdings sind gerade die Signale, die vom Tag zum Lesegerät führen, besonders schwach und daher nur schwer abzuhören.

3 Security Analysis of a Cryptographically-Enabled RFID Device [2]

In diesem Paper wird gezeigt, dass auch bei teuren, als fälschungssicher angepriesenen RFID-Tags Vorsicht geboten ist. Dabei werden die Sicherheitsmängel des Texas Instruments DST (Digital Signature Transponder) aufgezeigt, mit dem Ziel, Chip-Hersteller auf die Problematik zu sensibilisieren, sodass in Zukunft solche Mängel im Voraus vermieden werden können. Beim Texas Instruments DST handelt es sich um einen komplexeren Tag, welcher über kryptographische Funktionalität verfügt und dadurch ein Tag-Authentifizierungs-Protokoll basierend auf dem Challenge-Response Prinzip unterstützt. Er wird häufig in elektronischen Wegfahrsperren von Fahrzeugen und als elektronisches Zahlungsmittel verwendet. Als Beispiel wird hier der Speed-

Pass™ des amerikanischen Mineralölkonzerns ExxonMobil genannt, mit welchem man bargeldlos Benzin tanken kann.

Die Challenge-Response-Authentifizierung des DST funktioniert folgendermassen: Zunächst generiert das Lesegerät eine zufällige, 40 Bit grosse Challenge, welche es an den Tag schickt. Dieser hat seinerseits einen geheimen, 40 Bit grossen Schlüssel gespeichert, mit welchem er die Challenge mittels eines geheimen Algorithmus verschlüsselt. Die least significant 24 Bits des Chiffrats werden dann zurück an das Lesegerät geschickt. Da auch das Lesegerät den geheimen Schlüssel kennt, kann es verifizieren, ob das empfangene Chiffrat vom Tag mit dem korrekten Schlüssel erzeugt wurde.

Die Autoren des Papers sind wie folgt vorgegangen: Zuerst mussten sie den geheimen Verschlüsselungsalgorithmus des Tags mittels Reverse Engineering herausfinden. Mit Hilfe dieses Algorithmus konnten sie dann in einem zweiten Schritt den Schlüssel eines beliebigen Tags knacken. Schliesslich simulierten sie noch einige Tags, um ihre Resultate zu verifizieren.

3.1 Reverse Engineering

Um den geheimen Verschlüsselungsalgorithmus herauszufinden, hatten die Autoren des Papers nur wenige Hilfsmittel zur Verfügung. Einerseits hatten sie die in Abb. 1 dargestellte, grobe schematische Darstellung der Verschlüsselungslogik, die von Texas Instruments selbst einmal veröffentlicht wurde. Andererseits verfügten sie über echte DSTs, bei denen sie zwar den geheimen Schlüssel selbst einprogrammieren konnten, sonst aber nur mittels des Inputs und des daraus resultierende Outputs auf deren Funktionsweise schliessen konnten („Black-Box“ Zugriff).

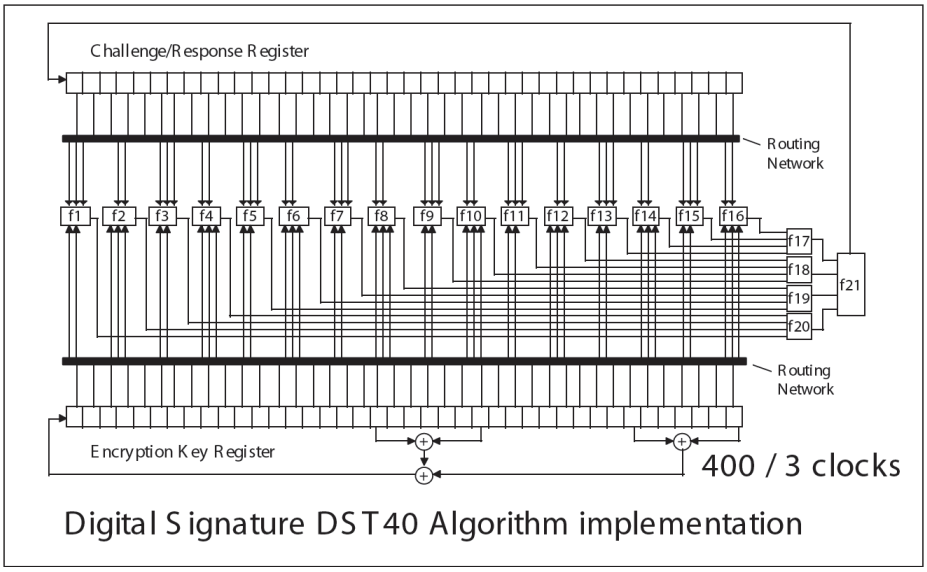

Abb. 1. Schematische Darstellung der Verschlüsselungslogik

In das 40 Bit grosse Challenge/Response-Register wird die Challenge hineingeschrieben und im Laufe des Algorithmus in die Response umgewandelt. In das Encryption Key Register wird der geheime Schlüssel hineinkopiert. Es handelt sich dabei um sog. „feedback shift registers“, denn der Algorithmus läuft über mehrere Runden, wobei der Inhalt dieser Register in jeder Runde um ein Bit nach rechts geschoben wird. Links wird jeweils ein neues Bit eingefügt. Beim Encryption Key Register wird das einzufügende Bit mittels mehrerer XOR-Operatoren aus dem vorherigen Zustand des Encryption Key Registers berechnet. Beim Challenge/Response-Register ist das einzufügende Bit das Resultat einer komplizierten Logik, die aus den logischen Einheiten f1 bis f21 besteht. Diese Logik hat sowohl Bits des Encryption Key Registers als auch Bits des Challenge/Response-Registers als Input. Wegen der zwischengeschalteten Routing Networks weiss man aber nicht, welche Bits genau als Input dienen. Es ist auch nichts über die logische Funktionsweise der Einheiten f1 bis f21 bekannt. Um jetzt die Funktionsweise der gesamten Verschlüsselungslogik herauszufinden, bedienten sich die Autoren des Papers mehrerer Tricks. Einen davon möchte ich im Folgenden detailliert vorstellen:

Wenn man den Schlüssel aus lauter 0en wählt, bleibt er über den ganzen Algorithmus hinweg unverändert. Denn 0 XOR 0 ergibt wieder 0. Also wird nach jeder Runde wieder eine 0 links eingefügt. Auf diese Weise kann man die Logik, bestehend aus den Einheiten f1 bis f21, unabhängig vom Schlüssel betrachten. Nennen wir die ersten 39 Bit der 40 Bit grossen Challenge C C'. Zu Beginn des Algorithmus steht die Challenge C im Challenge/Response-Register. Nach der ersten Runde wird C um ein Bit nach rechts geschoben und links kommt eine 0 oder eine 1 hinein. Nennen wir den Inhalt des Challengs/Response-Registers nach der ersten Runde C_0, falls eine 0 eingefügt wurde, und C_1, falls eine 1 eingefügt wurde. D. h. $C_0 = 0|C'$ und $C_1 = 1|C'$. Wir nehmen jetzt einmal an, C_0 wäre der wirkliche Inhalt des Challenge/Response-Registers nach der ersten Runde. Wenn man den gesamten Algorithmus (inkl. erste Runde) auf C_0 anwendet, bekommt man fast dasselbe Resultat, wie wenn man den Algorithmus auf die ursprüngliche Challenge C anwenden würde. Es ist einfach um ein Bit nach rechts verschoben, da man gewissermassen eine Runde zu weit gegangen ist. War aber die Annahme falsch, und C_1 wäre der wirkliche Inhalt des Challenge/Response-Registers nach der ersten Runde gewesen, dann käme mit grosser Wahrscheinlichkeit ein deutlich anderes Resultat heraus. Dieses eine falsche Bit würde im Laufe des Algorithmus im Challenge/Response-Register nach rechts laufen und würde dadurch immer wieder in die Logik eingeflochten. Analog sieht es natürlich für den Fall aus, wenn C_1 der wirkliche Inhalt des Challenge/Response-Registers nach der ersten Runde ist. Auf diese Weise ist es möglich, das Resultat der Logik, bestehend aus den Einheiten f1 bis f21, nach der ersten Runde eindeutig zu berechnen.

Mit Hilfe dieses Tricks und noch vielen weiteren, aber ähnlich funktionierenden Tricks, kann man die Wahrheitstabellen der Einheiten f1 bis f21 herausfinden und auch auf die Funktionsweise der Routing Networks schliessen. Schliesslich kann aus diesen Informationen der gesamte Verschlüsselungsalgorithmus hergeleitet werden.

3.2 Key Cracking

Nachdem sie den geheimen Verschlüsselungsalgorithmus herausgefunden hatten, versuchten die Autoren des Papers, den Schlüssel eines beliebigen Tags zu knacken. Da die Challenge aus 40 Bits besteht, die Response aber nur aus 24 Bits, sind zwei abgehörte Challenge/Response-Paare erforderlich, um den Schlüssel eindeutig berechnen zu können. Damit konnten sie mit Hilfe von 16 parallel geschalteten FPGAs (siehe Abb. 2) den Schlüssel eines Tags in weniger als einer Stunde knacken. Die FPGA probieren dabei „brute-force“ alle möglichen Schlüsselbelegungen durch. Bei einer Schlüssellänge von nur 40 Bit ist dieses Vorgehen durchaus praktikabel.

Abb. 2. Knacken eines Schlüssels mit Hilfe von 16 FPGAs

3.3 Simulation von Tags

In einem letzten Schritt verifizierten die Autoren des Papers ihre in den Teilkapiteln 3.1 und 3.2 beschriebenen Resultate, indem sie echte Tags simulierten. Dazu brauchten sie nur einen Laptop, eine RFID-Antenne und ein Gerät, welches die digitalen Signale des Laptops in analoge Signale für die Antenne umwandelt. Damit konnten sie ein Auto mit elektronischer Wegfahrsperre starten, und es ist ihnen gelungen, mit dem früher erwähnten SpeedPass™ gratis Benzin zu tanken.

3.4 Zusätzliche Überlegungen

Warum haben Texas Instruments ihren Tag nicht sicherer gemacht? Allein durch die Wahl einer grösseren Schlüssellänge hätte erheblich mehr Sicherheit erlangt werden können. Das Problem hierbei ist, dass der Algorithmus in diesem Fall über wesentlich mehr Runden ablaufen müsste, was die Authentifizierung sehr langsam machen würde. Man möchte schliesslich nicht 15 Sekunden oder länger warten, bis man sein Auto

starten kann. Optimal wäre die Wahl eines öffentlich bekannten, aber als sicher anerkannten Algorithmus anstelle des geheimen Algorithmus. Diese Lösung würde allerdings eine wesentlich komplexere Logik auf dem Chip erfordern, was die Chip-Herstellung verteuern würde. Meiner Meinung nach ist das Problem bei der elektronischen Wegfahrsperre nicht so gravierend, da sie nicht die einzige Sicherheitsmassnahme eines Fahrzeugs ausmacht. Auch ein Auto ohne Wegfahrsperre ist nicht so leicht zu stehlen. Beim elektronischen Zahlungsmittel ist es schlimmer. Hier reicht es, den Tag von jemand anderem zu klonen, um auf dessen Kosten Einkäufe zu tätigen. Schliesslich sei noch erwähnt, dass Texas Instruments auch bessere Tags im Angebot haben. In Anbetracht dieser Tatsache könnte das Ganze auch so ausgelegt werden, dass die Kunden von Texas Instruments zu sehr gespart haben, indem sie einen billigen Tag gekauft haben.

4 Schlussbemerkungen

In den beiden vorgestellten Papers geht es um unterschiedliche Typen von Tags, die auch unterschiedlich eingesetzt werden. Der Schutz vor Fälschungen ist aber bei beiden Tag-Typen ein wichtiges Thema. Die im Paper „Strengthening EPC Tags Against Cloning“ [1] vorgestellte Lösung ist absolut gesehen sicher nicht optimal, in Anbetracht der Einfachheit von EPC-Tags könnte sie aber in bestimmten Anwendungsszenarien durchaus von Bedeutung sein. Bei den im Paper „Security Analysis of a Cryptographically-Enabled RFID Device” [2] genannten Tags ist es offenbar u. a. eine Kostenfrage, wie sicher sie wirklich sind. Man ist heute durchaus in der Lage, wesentlich fälschungssicherere Tags zu produzieren.

5 Literatur

[1] Ari Juels. Strengthening EPC Tags Against Cloning. Manuscript published at http://www.rsasecurity.com/rsalabs/node.asp?id=2780, 2005

[2] Steve Bono, Matthew Green, Adam Stubblefield, Ari Juels, Avi Rubin, Michael Szydlo. Security Analysis of a Cryptographically-Enabled RFID Device. Submitted for publication, see http://www.rsasecurity.com/rsalabs/node.asp?id=2838, 2005