CSA – FAQ

von [gme]

v0.5beta
Diese FAQ befindet sich sozusagen im Betastadium. Es gibt keine Gewähr auf Vollständigkeit noch 100% sachliche Korrektheit. Es werden bestimmt noch ca. drölf Millionen Fehler vorhanden sein…aber Hauptsache das Ding is' erst mal online und verschimmelt nicht mehr auf meiner HD…
Es fehlen:
-Beschreibung des Streamcipher
-Beschreibung des Blockcipher
-Beschreibung des Keyschedulers
-1000 Dinge, die mir jetzt gerade nicht einfallen…

1. Basic-FAQ

Was heisst CSA ?

Common Scrambling Algorithmus.

Wo wird er angewendet?

Mit dem CSA verschlüsseln alle digitalen Pay-TV Sender, die DVB konform ausstrahlen ihre Sendungen.

Ich dachte es wird mit Irdeto(2)/Seca(2)/Nagravision/etc... verschlüsselt?!

Diese senderabhängigen Verfahren verschlüsseln lediglich den eigentlichen Key des CSA, Control Word oder Common Key genannt.

Wieso so umständlich?

Dieses Verfahren wurde eingeführt um eine gewisse Basis zu sichern und dem ‚proprietären Wildwuchs' entgegenzuwirken. Somit sollte (mehr oder wenig) jedes DVB Programm auf jedem DVB Receiver empfang- & decodierbar sein. Außerdem wird das sog. Simulcrypt, das gleichzeitige verwenden mehrerer Verschlüsselungsverfahren für ein Programm möglich ohne denselben Videostream mehrmals senden zu müssen.

Was kann man damit hacken?

Mit dem CSA selber? Nix. Der CSA ist wie der Name schon sagt ‚nur' ein Algorithmus, eine Rechenvorschrift, die sagt wie Daten mittels eines Schlüssels verändert werden sollen um vor unautorisiertem Zugriff geschützt zu sein.

Ich dachte Pay-TV wäre schon geknackt, die machen doch überall so einen Aufstand wegen der Schwarzseher?

‚Geknackt' sind nur die drüberliegenden Systeme (Irdeto usw) und davon noch nichtmal alle. Eigentlich sind auch nicht diese Systeme geknackt, sondern nur die Karten unsicher, so dass man die Keys (für das jeweilige System) auslesen kann und oft auch den Algorithmus. Mit sicheren Karten blieben viele Bildschirme schwarz.

Was würde denn passieren wenn der CSA ‚geknackt' wäre?

Das wäre der heilige Gral des DVB Pay-TV und ist auch als „Streamhack" bekannt. Damit wären auf einen Schlag alle DVB konformen (den CSA benutzenden) Sender offen und es müsste einiger Aufwand getrieben werden, das wieder rückgängig zu machen. Viele Leute lachen darüber, andere schwärmen davon. Wenn es jemand schon geschafft hat, dann ist der bis jetzt sehr still geblieben – meiner Meinung nach das Klügste was man in diesem Fall tun kann, zurücklehnen und genießen...

Wenn es so wichtig ist, wieso hält man den Algorithmus dann nicht geheim?

Das hat man getan, über viele Jahre war der CSA ein wohlbehütetes Geheimnis des DVB Konsortiums. Er wurde nur gegen einen NDA (non Disclosure Agreement) vergeben und auch dann nur, wenn der User ‚bone fide' (übersetzbar mit ‚in guter Absicht') handelt. Außerdem durfte er nur in Hardware implementiert werden um reverse Engineering so schwer wie möglich zu machen.

Wie ist der CSA denn publik geworden?

Das passierte nicht alles auf einmal, irgendwann (~2000 ?) tauchten Patentscripte über den CSA auf (die ‚berühmte' UK PA GB2 322 994/995), darin wird der CSA weitgehend beschrieben, allerdings fehlten wichtige Einzelheiten wie z.B. Permutationstabellen und Substitutions-Tabellen (S-Boxen). Im Herbst 2002 ging es dann Schlag auf Schlag, es tauchte eine Software namens FreeDec auf, die es möglich machte mit einer sog. Budget Karte von TwinHan PayTV zu sehen. Zuvor mussten Karten dafür die nötige Hardware, den ECD (European Common Descrambler) an Board haben. Leider tauchten zu der Software keine Sourcecodes auf und so passierte, was passieren musste (was die Entwickler des CSA vorrausgesehen hatten und ihn deshalb nur in Hardware implementieren lassen wollten): FreeDec wurde disassembliert und analysiert, nach wenigen Wochen (harter Arbeit!) waren die letzten unbekannten Details gelüftet. Woher der Autor von FreeDec (‚Christof') die Informationen hat, ist reine Spekulation. Angeblich soll er sie aus den Patentscripten haben, wenn das stimmt muss er genauere Scripte gehabt haben, als die allgemein bekannten.

Nun, den Streamhack gibt es bisher nicht, wofür also der ganze Aufstand?!

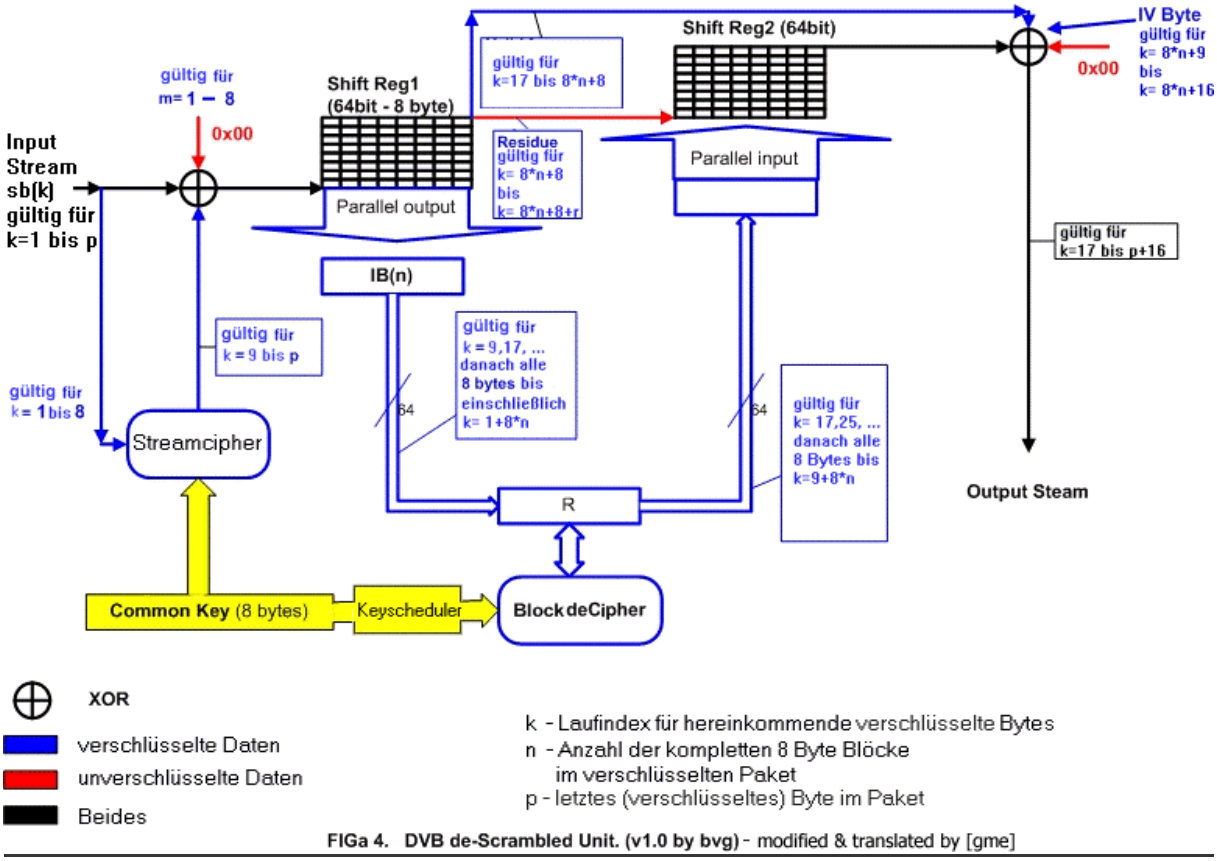
Mit dem CSA als Software-Implementation ist es jetzt möglich mit praktisch jeder DVB Karte PayTV zu entschlüsseln – vorrausgesetzt man hat die gültigen Keys für den Algorithmus, der die Control Words liefert (Irdeto, usw...)

Außerdem kann der Algorithmus analysiert werden und vielleicht findet sich ja doch die eine oder andere Schwachstelle.

2. detaillierte Beschreibung

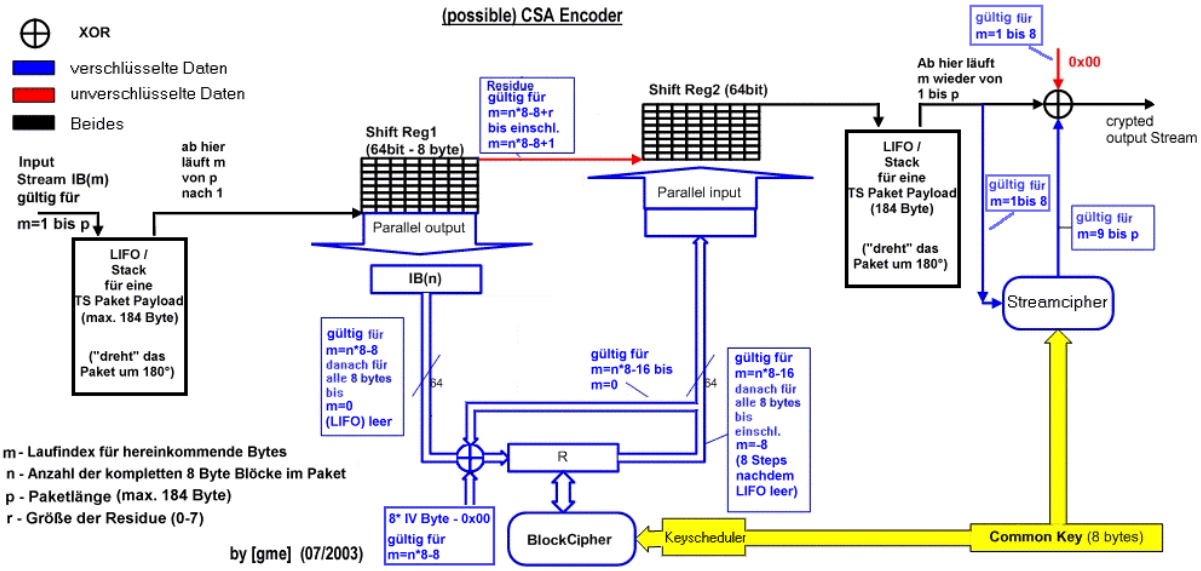
Du bist bis hierher vorgedrungen, also nehme ich an, dass die Basic-FAQ noch nicht alle deine Fragen beantwortet hat. Vielleicht können dir die folgenden Seiten ja weiterhelfen (vorsicht, es wird teilweise bestimmt ein wenig theoretisch :))

2.1 Wie sieht der CSA aus?



FIGa 4. DVB de-Scrambled Unit. (v1.0 by bvg) - modified & translated by [gme]

Dies ist eine Übersicht über den CSA Descrambler (der Scrambler sieht anders aus). Der CSA ist die (fiese) Kombination zweier eigener Algorithmen, eines Stream- und eines Blockciphers. Jeder der beiden würde für sich schon eine nette Verschlüsselung abgeben, aber die Entwickler mussten sie unbedingt kombinieren...



Dies ist der zugehörige Scrambler, bzw. meine Interpretation von ihm.
Bevor ich jetzt den Vorgang des Dekodierens beschreibe, ist es vielleicht sinnvoller, den Vorgang des Kodierens zu betrachten.
Zuerst wird der CK bzw. das CW für das TS Paket gesetzt. Für den Blockcipher generiert der Keyscheduler (auf den ich später noch eingehen werde) die 56 Byte große Keytabelle, die aus dem CK und 6 aus ihm per Permutation abgeleiteten Keys besteht.
Im Streamcipher wird ein Reset durchgeführt, alle internen Register des Streamciphers auf 0 gesetzt und der CK auf die 2 internen Schieberegister verteilt (NICHT die 64 Bit Register auf dem Bild oben, siehe Beschreibung vom Streamcipher).
Die unverschlüsselte TS Payload kommt zuerst in einen LIFO Speicher (bzw. Stack oder Stapel), aus dem sie sofort wieder ausgelesen wird. Das ganze hat den Sinn die Reihenfolge der Daten umzukehren, das letzte Byte ist jetzt das erste und umgekehrt.
So werden die Daten in das 1. Schieberegister geladen. Ist eine nicht glatt durch 8 teilbare Anzahl vorhanden, so wird der Rest (die Residue, die beim ‚durch 8 teilen' bleibt) als erstes durch das 1. Schieberegister in das 2. geschoben. Danach kommt der erste 8Byte Datenblock (im folgenden Block genannt). Sind diese 8 Byte komplett im 1. Register, werden sie parallel durch das XOR Gate in das Register R geschoben. Der erste Block (die ersten 8 Bytes, die eigentlich die letzten vor der Residue bzw. der letzte Block sind, da die Daten ja umgedreht wurden) werden im XOR Gate byteweise mit dem Initialisierungsverktor (IV) geXORt. Dieser Vektor ist beim CSA = 0, könnte aber auch jeden anderen Wert zwischen 0 und 255 haben.
Der erste Block steht nun im Register R, dort wird er mit dem Blockcipher verschlüsselt. Dafür ist Zeit bis der nächste Block komplett in Schieberegister 1 steht. Währenddessen wurde auch ein evtl. vorhandene Rest aus dem 2. Schieberegister in den 2. Stack geschoben. Bevor der 2. Block nun vom Schieberegister nach R geschoben wird, wird er mit dem 1. schon verschlüsselten Block geXORt (byteweise, Block 2 Byte 1 mit Block1 Byte 1 usw.). Danach steht der veränderte 2. Block in R und der mit dem Blockcipher verschlüsselte Block 1 in Schieberegister 2. Von dort wird er nun byteweise auf den 2. Stapel geschoben.
Das ganze wiederholt sich bis alle Daten im Stapel liegen. Von dort werden sie wieder ausgelesen um die ursprüngliche Reihenfolge wiederherzustellen.
Nun ist der Streamcipher dran, der wurde ja am Anfang schon mit dem CK geladen (Reset), der erste Block (nun wirklich die Bytes 1-8) wird als zusätzlicher Initialisierungsvektor für den Streamcipher benutzt und während der sog. Initialisierungphase in den Streamcipher geschoben. Gleichzeitig passiert er UNVERÄNDERT (xor 0) das XOR Gate und geht auf die weitere Reise. Sind diese 8 Bytes durch, schaltet der Streamcipher in die Generierungsphase, von jetzt an wird jedes Byte mit dem Output des Streamciphers geXORt, auch der zuvor unkodierte Rest (Residue).

Das wars. Nun wird das ganze zu einem handlichen TS Paket verschnürt und auf die lange Reise zum Empfänger geschickt.

Beim Empfänger werden die ersten 8 Bytes als Startwert für den Streamcipher benutzt, der zuvor zurückgesetzt und mit dem CW geladen wurde. Für die nächsten 8 Bytes (und den Rest des Pakets) schaltet der Streamcipher – wie schon beim Senden - in die Generierungsphase. Die hereinkommenden Bytes werden ab jetzt mit dem Output des Streamciphers geXORt bevor sie in das Schieberegister 1 gelangen.
Ist das Schieberegister mit 8 Bytes gefüllt, werden diese 8 Bytes parallel zum Blockcipher transferiert und dort entschlüsselt. Der entschlüsselte Block wird, sobald Schieberegister 1 wieder komplett mit Daten gefüllt ist oder der Residue bis zur Position 8 geschoben wurde, parallel ins Schieberegister 2 geladen. Dort werden sie byteweise herausgeschoben und mit dem aus Schieberegister 1 seriell herausgeschobenen folgenden Block geXORt. Falls der aktuelle Block der letzte ist, wird der Block mit dem IV (0) geXORt. Die Residue (falls vorhanden) wird aus Schieberegister 1 direkt in das Schieberegister 2 geschoben und folgt somit direkt dem letzten Block. Die Residue wird ebenfalls mit 0 (bzw. nicht) geXORt.
Danach ist die Entschlüsselung komplett.

...wird demnächst fortgesetzt... – [gme] 10.11.2003