HUMBOLDT-UNIVERSITÄT ZU BERLIN

Institut für Informatik
Seminar: Analyse kryptographischer Algorithmen
SS 2002

Der E2-Algorithmus

Ausarbeitung zum Seminarvortrag
von
Heinz-Günter Kuper
kuper@informatik.hu-berlin.de

INHALTSVERZEICHNIS

1 Einführung

E2 – der "**E**fficient **E**ncryption" Algorithmus -- ist eine symmetrische Blockchiffre, die von Nippon Telegraph und Telephone Corporation (NTT) entwickelt wurde als Kandidat für den Advanced Encryption Standard. Die Entwickler des Algorithmus – Kazumaro Aoki, Masayuki Kanda, Tsutomu Matsumoto, Shiho Moriai, Kazuo Ohta, Miyako Ookubo, Youichi Takashima und Hiroki Ueda – haben sich die folgende Ziele gesetzt:

- E2 sollte resistent sein gegen
    - *Brute-Force*-Angriffen,
    - Differentielle Kryptoanalyse,
    - Lineare Kryptoanalyse,
    - *Higher-Order Differential Attack*
    - *Interpolation Attack*
    - *Partitioning Attack* und
    - *Related-Key Attack*.
- Die Geschwindigkeit der Verschlüsselung und Entschlüsselung von E2 sollte schneller sein als die des Single-DES.

E2 ver- und entschlüsselt 128-bit Daten mit Schlüsselgrößen von 128-, 192- und 256-bit. Der Algorithmus besteht aus einem Datenrandomisierungsteil und einem Schlüsselgenerator-Teil.

Die Datenrandomisierung wird in einer Feistel-Chiffre-Struktur realisiert mittels einer 12-rundige Wiederholung der *F*-Funktion, einer initialen Transformation (*IT*-Funktion) und einer finalen Transformation (*FT*-Funktion). Schlüsselgeneration wird erreicht durch eine 9-rundige Wiederholung der *G*-Funktion.

Die Entwickler waren der Meinung, dass schon eine 9-rundige Wiederholung der F-Funktion genug Sicherheit bieten würde gegen *Brute-Force*-Angriffe, differentielle/lineare Kryptoanalyse, *higher order differential attack* und *interpolation attack*, aber haben eine zusätzliche Steigerung der Sicherheit erhofft durch hinzufügen der komplizierten IT- und FT-Funktionen und Erhöhung der Runden von 9 auf 12. Sie behaupteten, dass es keine verkürzte Angriffsmöglichkeiten auf den Algorithmus gäbe und dass E2 nur für eine *Brute-Force*-Angriff anfällig wäre. E2 soll also mit einem 192- bzw. 256-bit Schlüssel sicherer sein als Triple-DES.

Deren ANSI C Implementierung von E2 erreichte schon 30M-bits-pro-Sekunde auf einem Pentium Pro (200 MHz), während eine typische Implementierung in C von DES erreichte nur 28M-bits-pro-Sekunde auf einem DEC Alpha 8400 (300 MHz).

Nach einer Beschreibung des E2-Algorithmus, werden die Ergebnisse einer Analyse des Algorithmus durch NIST vorgeführt. Schließlich wird eine Einführung in die Problematik von *Brute-Force*-Angriffen versucht.

2 Der E2-Algorithmus

Die globale Struktur des Algorithmus basiert auf die Feistel-Chiffre, aufgrund ihrer Bekanntheit und stetiges Widerstand gegen Angriffe und wegen ihrer Symmetrie (das Verfahren zum Ver- und Entschlüsseln ist dasselbe).

Zunächst wird das Verfahren zum Ver- und Entschlüsseln behandelt, danach die Schlüsselgeneration (engl. *key scheduling*) und letztlich die durch den Algorithmus benutzte Funktionen werden erläutert.

2.1 Bemerkung zur Notation

B sei ein Vektorraum, der aus 8-bit (1 Byte) Elementen besteht.
D.h. $B := GF(2)^8$

W sei ein Vektorraum, der aus 32-bit (1 Wort) Elementen besteht.
D.h. $W := B^4$

H sei ein Vektorraum, der aus 64-bit (1 halber Block) Elementen besteht.
D.h. $H := B^8$

M sei der Klartext, $M \in H^2$.
C sei der Kryptotext, $C \in H^2$.
K sei der Schlüssel, $K \in H^2$. H^3 bzw. H^4.
k sei der Rundenschlüssel, $k \in H^2$.

2.2 Verschlüsselung

Zuerst erzeugt der Schlüsselgenerator 16 Rundenschlüssel (engl. *subkeys*) $\{k_1,k_2,...,k_{16}\}$ ($k_i \in B^{16}$) aus einem Geheimschlüssel *K*. Datenrandomisierung wird erreicht durch eine initiale Transformation *IT*, eine 12-Runden Feistel-Chiffre mit *F*-Funktion und eine finale Transformation *FT*.

M' wird wie folgt berechnet, wobei *M* der Klartext ist:

$$M' = IT(M,k_{13},k_{14})$$

M' wird in L_0 und R_0 aufgeteilt, d.h. $M' = (L_0,R_0)$, wobei $L_0,R_0 \in H$. Für $r = 1$ bis 12 wird wie folgt weiter berechnet:

$$R_r = L_{r-1} \oplus F(R_{r-1}, k_r)$$

$$L_r = R_{r-1}$$

Sei C' die Verkettung von R_{12} und L_{12}, d.h. $C' = (R_{12},L_{12})$. Der Kryptotext wird schließlich durch

$$C = FT(C', k_{16}, k_{15})$$

berechnet.

2.3 Entschlüsselung

Zuerst werden 16 Rundenschlüssel erzeugt. Die Datenrandomisierung wird durch eine initiale Transformation *IT*, gefolgt von einer 12-Runden Feistel-Chiffre mit *F*-Funktion und eine finale Transformation *FT* erreicht.

Sei *C* der Kryptotext.

$$C' = IT(C,k_{16},k_{15})$$

C' wird aufgeteilt in (R_{12},L_{12}). Für $r = 12$ bis 1 wird wie folgt weiter berechnet:

$$L_{r-1} = R_r \oplus F(L_r,k_r)$$

$$R_{r-1} = L_r$$

Sei M' die Verkettung von L_0 und R_0, d.h. $M' = (L_0,R_0)$. Der Klartext *M* wird schließlich durch

$$M = FT(M',k_{13},k_{14})$$

berechnet.

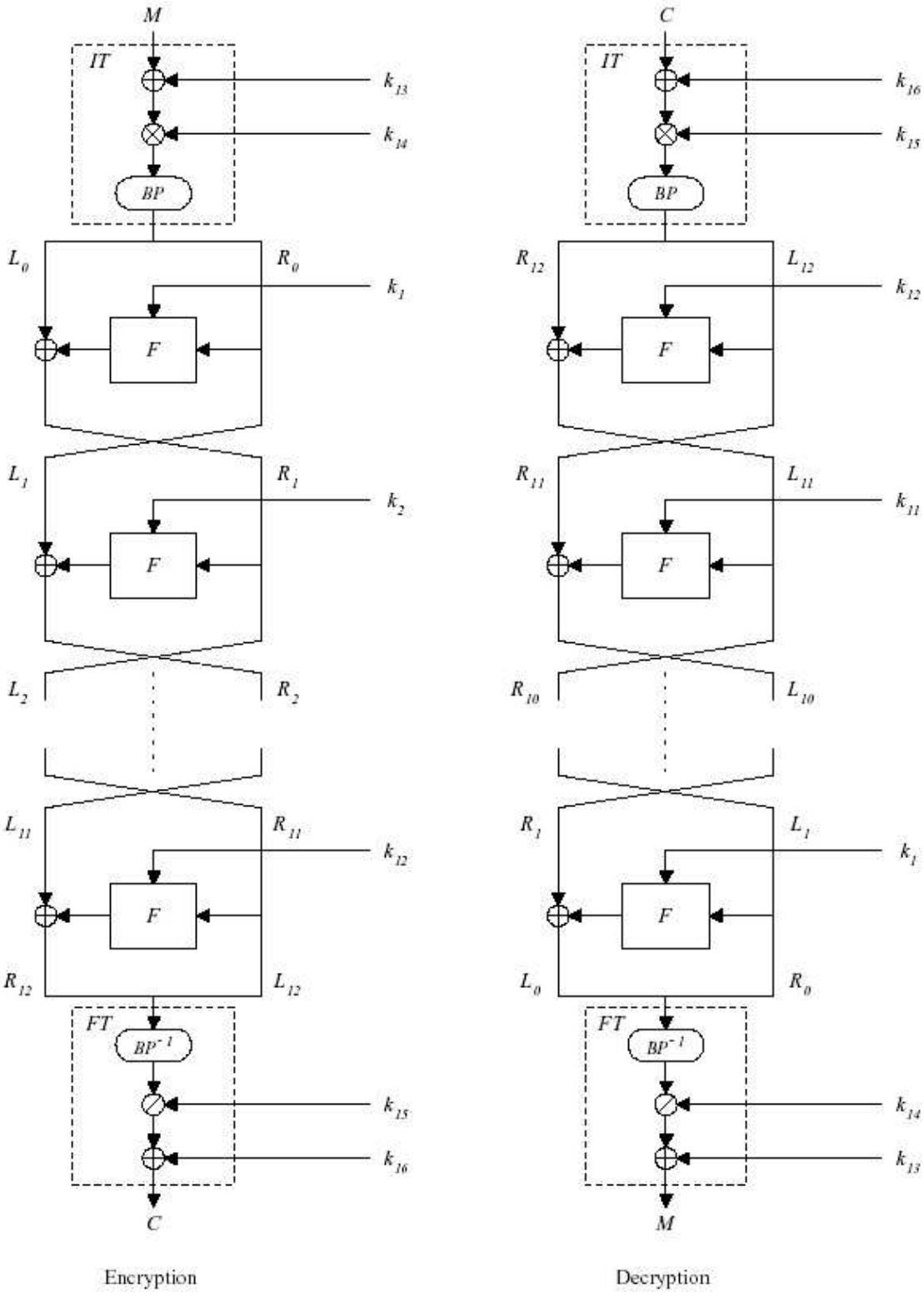

Abbildung 1: Ver- und Entschlüsselung

2.4 Der Schlüsselgenerator

Aus einem Geheimschlüssel $K = (K_1,K_2,K_3,K_4)$, wobei $K_i \in$ H, i = 1,...,4, werden 16 Rundenschlüssel wie folgt erzeugt:

v_{-1} = 0123456789abcdef$_{(hex)}$

$(L_0,(Y_0,v_0)) = G(K,v_{-1})$

$(L_{i+1},(Y_{i+1},v_{i+1})) = G(Y_i,v_i)$ $\quad (i = 0,1,...,7)$

$(l_{4i},l_{4i+1},l_{4i+2},l_{4i+3}) = L_{i+1}$ $\quad (i = 0,1,...,7)$

$(t_i^{(0)},t_i^{(1)},...,t_i^{(7)}) = l_i$ $\quad (i = 0,1,...,31)$

$k_{i+1} = \left(t_{0+(i \bmod 2)}^{\lfloor i/2 \rfloor}, t_{2+(i \bmod 2)}^{\lfloor i/2 \rfloor}, \ldots, t_{30+(i \bmod 2)}^{\lfloor i/2 \rfloor}\right)$ $\quad (i = 0,1,...,15)$

wobei L_i, $Y_i \in \mathrm{H}^4$, $l_i, v_i \in \mathrm{H}$ und $t_i^{(j)} \in \mathrm{B}$.

Für 128-bit Geheimschlüssel werden K_3 und K_4 auf konstante Werte wie folgt gesetzt:

$K_3 = S(S(S(v_{-1})))$
$K_4 = S(K_3)$

Für 192-bit Geheimschlüssel wird nur $K_4 = S(S(S(S(v_{-1}))))$ gesetzt.

2.5 Die *IT*-Funktion

Die initiale Transformation wird wie folgt definiert:

$IT: \mathrm{H}^2 \times \mathrm{H}^2 \times \mathrm{H}^2 \rightarrow \mathrm{H}^2$
$(X,A,B) \mapsto BP((X \oplus A) \otimes B)$

Der Binäroperator $\otimes$ ist definiert als

$$Y = X \otimes B, \text{ mit } X,Y,B \in \mathrm{H}^2$$

wobei

$$(x_1, x_2, x_3, x_4) = X \quad (x_i \in W, i = 1,2,3,4)$$
$$(b_1, b_2, b_3, b_4) = B \quad (b_i \in W, i = 1,2,3,4)$$
$$y_i = x_i (b_i \vee 1) \bmod 2^{32} \quad (i = 1,2,3,4)$$
$$Y = (y_1, y_2, y_3, y_4)$$

und *BP* ist eine Byte-Permutation $BP: \mathrm{W}^4 \rightarrow \mathrm{W}^4$.

2.6 Die *F*-Funktion

Wird wie folgt definiert:

$F: \mathrm{H} \times \mathrm{H}^2 \rightarrow \mathrm{H}$
$(X,(K^{(1)},K^{(2)})) \mapsto Y = BRL(S(P(S(X \oplus K^{(1)})) \oplus K^{(2)}))$

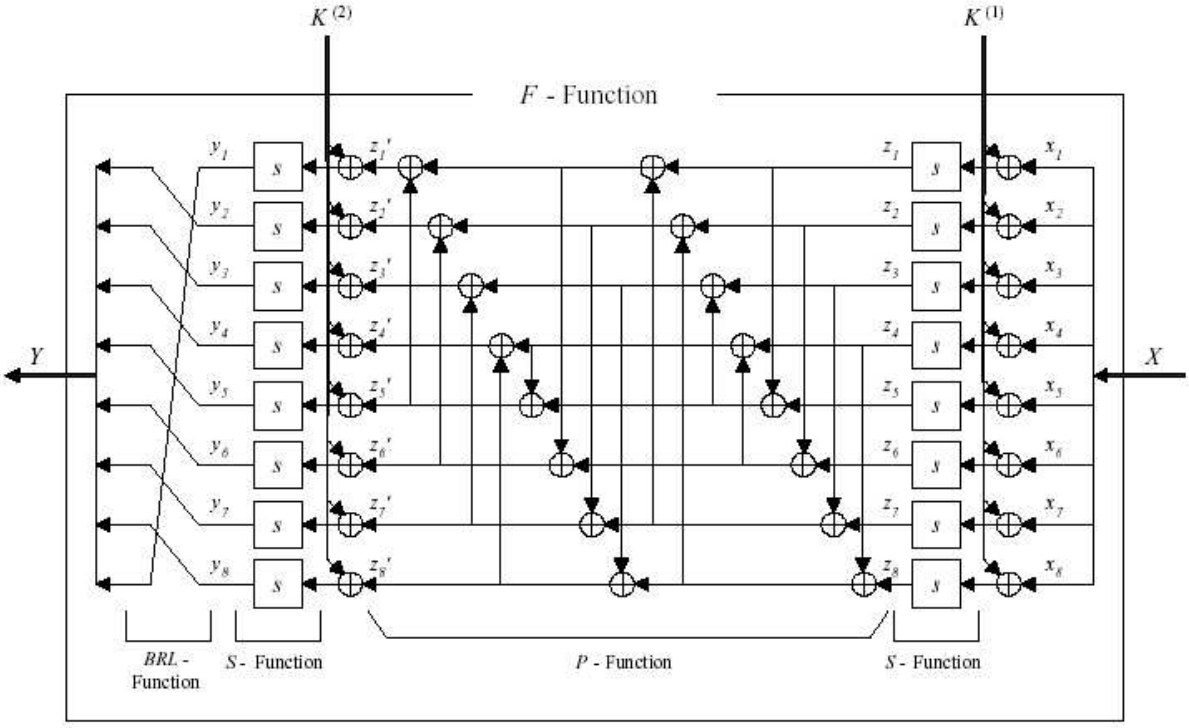

Abbildung 2: Die *F*-Funktion

2.7 Die *FT*-Funktion

Die finale Transformation ist die Inverse der *IT*-Funktion, d.h.

$$X = FT(IT(X,A,B),A,B)$$

2.8 Die *BRL*-Funktion

Die *Byte-Rotate-Left*-Funktion wird wie folgt definiert:

$$BRL: \mathrm{H} \rightarrow \mathrm{H}$$
$$(b_1, b_2,...,b_8) \mapsto (b_2,..., b_8, b_1)$$

2.9 Die *S*-Funktion

Die *S*-Funktion ist Teil der *F*-Funktion und wird mittels s-Boxen definiert:

$$S: \mathrm{H} \rightarrow \mathrm{H}$$
$$(x_1, x_2,...,x_8) \mapsto (s(x_1), s(x_2),...,s(x_8))$$

2.10 Die *s*-Boxen

Die *s*-Funktion wird wie folgt definiert:

$$s: B \to B$$
$$x \mapsto \text{Affine}(\text{Power}(x,127),97,225)$$

wobei

$$\text{Power}(x,e) = x^e \text{ in } GF(2^8)$$
$$\text{Affine}(y,a,b) = ay + b \pmod{2^8}$$

2.11 Die *P*-Funktion

Die *P*-Funktion ist Teil der *F*-Funktion und wird mittels einer Matrizengleichung definiert:

$$P: H \to H$$

$$\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_8 \end{pmatrix} \mapsto \begin{pmatrix} z_1' \\ z_2' \\ \vdots \\ z_8' \end{pmatrix} = P \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_8 \end{pmatrix}$$

wobei die Matrix *P* wie folgt gegeben ist:

$$P = \begin{pmatrix} 0\,1\,1\,1\,1\,1\,1\,0 \\ 1\,0\,1\,1\,0\,1\,1\,1 \\ 1\,1\,0\,1\,1\,0\,1\,1 \\ 1\,1\,1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,1\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,1\,1\,0\,0\,1 \end{pmatrix}$$

Diese Matrix kann auch mittels der folgenden Formel berechnet werden:

$$z_i' = \bigoplus_{j=0}^{7} t_{ij} z_j = \bigoplus_{t_{ij}=1} z_j$$

wobei t_{ij} das Element der *i*-ten Reihe und *j*-ten Spalte in der Matrix *P*.

2.12 Die *G*-Funktion

Die *G*-Funktion ist Teil des Schlüsselgenerators und wird wie folgt definiert:

$$G: \mathrm{H}^4 \times \mathrm{H} \rightarrow \mathrm{H}^4 \times (\mathrm{H}^4 \times \mathrm{H})$$
$$((X_1, X_2, X_3, X_4), U_0) \mapsto ((U_1, U_2, U_3, U_4), (Y_1, Y_2, Y_3, Y_4), V)$$

wobei

$$Y_i = f(X_i) \qquad (i = 1,2,3,4)$$
$$U_i = f(U_{i-1}) \oplus Y_i \qquad (i = 1,2,3,4)$$
$$V = U_4$$

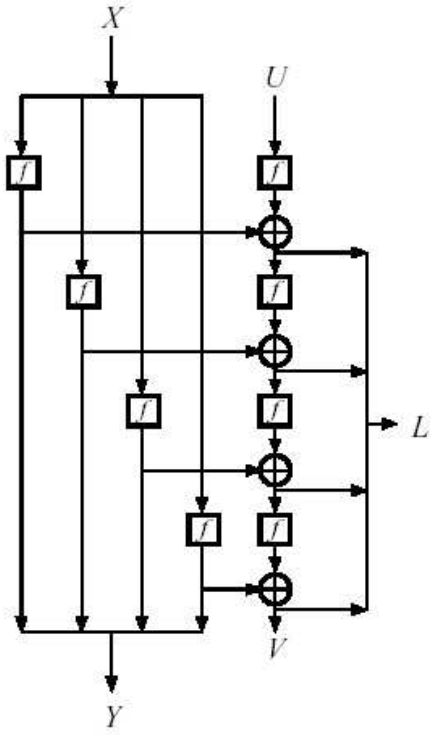

Abbildung 3: Die *G*-Funktion

2.13 Die *f*-Funktion

Die *f*-Funktion ist Teil der *G*-Funktion und wird wie folgt definiert:

$$f: \mathrm{H} \rightarrow \mathrm{H}$$
$$X \mapsto P(S(X))$$

3 NIST-Evaluierung des E2-Algorithmus

Obwohl die Evaluierung des NIST (National Institute of Standards and Technology) keine gravierende Sicherheitsmangel bei E2 feststellen könnte, wurde der Algorithmus nicht in der zweiten Runde des AES-Wettbewerbs aufgenommen. Die Gründe hierfür waren

- seine Langsamkeit im Vergleich zu Rijndael und Twofish,
- ein fehlender *on-the-fly* Schlüsselgenerator, der eine Implementierung auf *low-end* Smart-Cards ausschliest und
- ein hoher Bedarf an ROM.

Die Beurteilung wird unten weiter erläutert.

3.1 32-bit-Leistung

Da nur einfache RISC-Operationen benutzt wurden, sollte die Leistung des Algorithmus ziemlich gleich sein über verschiedener Prozessortypen. Die Geschwindigkeit wurde als angemessen eingestuft mit 410 *clocks per block* in Assembler-Sprache (E2 war damit die 4. schnellste auf dem Pentium und die 6. schnellste auf dem Pentium Pro/II) und eine einheitliche Leistung über verschiedene 32-bit CPUs.

3.2 Smart-Card-Leistung

Das Erzeugen von Rundenschlüssel *on-the-fly* wurde explizit durch die Entwickler des E2-Algorithmus ausgeschlossen (da sie es als ein Sicherheitsrisiko einschätzten) und daher benötigte die Rundenschlüssel 256 Bytes von RAM. Insgesamt braucht der Algorithmus also ungefähr 300 Bytes von RAM, was der Größteil der herkömmlichen Smart-Card-CPUs ausschließt. Daher wäre E2 nicht akzeptabel als ein vebreiteter Smart-Card-Standard. NIST schätzte, dass der Code und die Tabellen in weniger als 2K von ROM reinpassen sollte.

3.3 Hardware-Leistung

Die Hardware-Implementierung von E2 ist aufwendig: eine parallel Version benötigt sechszehn 256-Byte ROMs. Multiplikation wird nur in den initialen und finalen Transofrmationen gebraucht, aber dafür muss ein komkpletter Multiplikationsschaltkreis eingebaut werden, der aber nur wenig benutzt wird. Die NIST-Untersucher waren der Meinung, dass Hardware-Überlegungen nicht von höchster Wichtigkeit für die E2-Entwickler waren. Die für die Speicherung der Rundenschlüssel benötigte 256 Bytes von RAM führte auch zu einer Vergrößerung der Hardware.

4 *Brute-Force*-Angriffe

Es werden viele Metriken benutzt und Abschätzungen gemacht um heraus zu stellen, wie sicher ein Algorithmus tatsächlich ist. Gerade wegen der Komplexität dieser Algorithmen, sind diese Abschätzungen (z.B. bezüglich des *Linear Differential Attack*, *Higher Order Attack*, usw.) oft sehr subjektiv. Aber eine Angriffsmöglichkeit besteht immer: die des *Brute-Force*-Angriffs. Die Debatte um *Brute-Force*-Angriffe zentriert auf dem DES-Algorithmus [Schneier1996] und dies ist wiederum interessant für den E2-Algorithmus, da die Entwickler den Algorithmus selber in der Kategorie von DES klassifizieren (aufgrund seiner Feistel-Chiffre-Struktur).

In diesem Abschnitt wird der *Brute-Force*-Angriff näher untersucht und einige interessante Ideen von Bruce Schneier vorgestellt.

4.1 Prinzipien eines *Brute-Force*-Angriffs

Typischerweise ist ein *Brute-Force*-Angriff ein *known-plaintext* Angriff: d.h. es wird ein (Ausschnitt des) Kryptotexts und des dazugehörenden Klartexts benötigt. Weiterhin beeinflüssen zwei Parameter die Geschwindigkeit eines *Brute-Force*-Angriffs:

1. Die Anzahl der zu untersuchenden Schlüssel.
2. Die Geschwindigkeit, mit der jeden Schlüssel untersucht wird.

Obwohl die Schlüssel eines bestimmten Algorithmus vielleicht 10-mal schneller getestet werden können als die eines anderen, kann dieser Faktor ignoriert werden, da wir uns im Bereich von extrem grossen Zahlen bewegen.

4.2 Hardware-*Brute-Force*-Angriff

Ein *Brute-Force*-Angriff kann optimal in Hardware implementiert werden, besonders wenn parallele Prozessoren eingesetzt werden, da jeder Prozessor eine Untermenge des Schlüsselraums untersuchen kann. In 1994 untersuchte Michael Wiener wie teuer es wäre, spezialisierte Hardware für einen *Brute-Force*-Angriff zusammen zu stellen. Er entwarf spezialisierte Prozessoren, Platinen und Gehäuse (engl. *racks*) für einen Angriff auf DES. Wiener schätzte, dass man für US $ 1 Million ein Gerät zusammenbauen könnte, das einen 56-bit DES-Schlüssel in durchschnittlich 3,5 Stunden (maximal 7 Stunden) knacken könnte. Und der Preis/Geschwindigkeit-Verhältnis ist linear. Mit einkalkulieren von Moore's Law ist die unten angegebene Tabelle 1 schon eine konservative Einschätzung.

Kosten (US $)	Länge der Schlüssel (bits)					
	40	**56**	**64**	**80**	**112**	**128**
100.000	2 s	35 h	1 J	70.000 J	10^{14} J	10^{19} J
1 Million	0,2 s	3,5 h	37 Tage	7.000 J	10^{13} J	10^{18} J
10 Million	0,02 s	21 min	4 Tage	700 J	10^{12} J	10^{17} J
100 Million	2 ms	2 min	9 h	70 J	10^{11} J	10^{16} J
1 Milliarde	0,2 ms	13 s	1 h	7 J	10^{10} J	10^{15} J
10 Milliarde	0,02 ms	1 s	5,4 min	245 Tage	10^{9} J	10^{14} J
100 Milliarde	2µs	0,1 s	32 s	24 Tage	10^{8} J	10^{13} J
1 Billion	0,2 µs	0,01 s	3 s	2,4 Tage	10^{7} J	10^{12} J
10 Billion	0,02 µs	1 ms	0,3 s	6 h	10^{6} J	10^{11} J

Tabelle 1: Geschätzte Durchschnittszeit für einen Hardware- *Brute-Force*-Angriff in 1995

4.3 Software-*Brute-Force*-Angriff

Ohne sonderangefertigte Hardware und hochparallele Rechner, sind *Brute-Force*-Angriffe bedeutend schwieriger. Ein Software-Angriff ist ungefähr 1000-mal langsamer als ein Hardware-Angriff. Aber das Internet stellt gerade ein riesiges Netzwerk von Computern bereit, die ein riesiges Potential an Rechenkraft darstellen. Die weltweite Computerausstattung wurde 1996 auf 200 Millionen Computer – von Cray-Mainframes bis Subnotebooks – geschätzt. Auch wenn es möglich wäre, diese gesamte Rechenkraft zum knacken einen 128-bit Schlüssel zu koordinieren, und jeder Computer eine Million Verschlüsselungen pro Sekunde durchführen könnte, würde es immerhin die millionfache des Alters des Universums dauern bis zum Gewinnen des Schlüssels.

4.4 Die chinesische Lotterie

Quisquater und Desmedt haben die chinesische Lotterie als hochparalleles Kryptoanalysesystem vorgeschlagen. Stellen wir uns vor, dass in jedem verkauften Radio und Fernseher ein *brute-force*, Millionen-Tests-pro-Sekunde Chip eingebaut würde. Jeder Chip wird programmiert, automatisch nach dem Empfang des Klartext/Kryptotext-Paars per Funk eine unterschiedliche Menge von Schlüsseln zu untersuchen. Eine Zeit wird vereinbart, zu der jedes Gerät eingeschaltet wird zum Empfang der Daten. Nach einiger Zeit (siehe Tabelle 2) erscheint der Schlüssel auf dem Display, und der 'Gewinner' meldet sich bei der entsprechenden Stelle. Das Verfahren wäre einfacher, wenn jedes Gerät Zufallschlüssel probiert, aber dadurch wäre der Angriff 39% langsamer.

Land	Bevölkerung	Anzahl Fernsehr/Radio	Dauer	
			56-bit	64-bit
China	1.190.431.000	257.000.000	280 s	20 h
U.S.A.	260.714.000	739.000.000	97 s	6,9 h
Iraq	19.890.000	4.730.000	4,2 h	44 d
Israel	5.051.000	3.640.000	5,5 h	58 d

Tabelle 2: Chinesische Lotterie Angriffeinschätzung

4.5 Thermodynamische Grenze

Eine Folge des 2. Gesetzes der Thermodynamik ist, dass eine gewisse Energie benötigt wird zur Darstellung von Information. Das Setzen eines einzigen Bits durch Änderung des Zustands eines Systems bedarf mindestens k*T* Energie, wobei *T* die absolute Temperatur des Systems und k die Boltzmankonstante sind.

Da k = $1{,}38 \times 10^{-16}$ erg/° Kelvin und die Umgebungstemperatur des Universums 3,2° K beträgt, würde ein idealer Computer mit einer Betriebstemperatur von 3,2° K ungefähr $4{,}4 \times 10^{-16}$ erg aufbrauchen pro Setzen oder Löschen eines Bits. Sollte man der Computer kälter als die kosmische Hintergrundsstrahlung betreiben wollen, müsste man mehr Energie in eine Wärmepumpe investieren.

Die jährliche Energieausgabe der Sonne ist ungefähr $1{,}21 \times 10^{41}$ erg, was genug Energie wäre um ungefähr $2{,}7 \times 10^{56}$ einzelne Bitänderungen auf dem idealen Computer zu erreichen (genug um einen 187-bit Zähler durch alle Werte durchlaufen zu lassen). Sollte man eine Dyson-Sphäre um die Sonne bauen und ihre gesamte Energie für 32 Jahre aufsammeln, hätte man genug Kraft um einen Computer bis 2^{192} zählen zu lassen (aber ohne irgendwelche andere Berechnungen).

Hätte man eine Supernova als Energiequelle (10^{51} erg in einem Jahr), könnte der Computer bis 2^{219} zählen.

Aus diesen Berechnungen kann man schliessen, dass ein *Brute-Force*-Angriff gegen einen 256-bit Schlüssel unmöglich wäre, es sei denn Computer werden nicht aus Materie konstruiert und existieren irgendwo anders als im Raum.

5 Quellen

Gladman, B.
AES Algorithm Efficiency,
http://fp.gladman.plus.com/cryptography_technology/aes/index.htm

Nechvatal, J.; Barker, E.; Dodson, D.; Dworkin, M.; Foti, J.; Roback, E.
Status Report on the First Round of the Development of the Advanced Encryption Standard,
http://csrc.nist.gov/encryption/aes/round1/r1report.htm

Nippon Telegraph and Telephone Corporation,
Specification of E2 – a 128-bit Block Cipher,
http://info.isl.ntt.co.jp/e2/E2spec.pdf

Nippon Telegraph and Telephone Corporation,
Supporting Document on E2 (Updated),
http://info.isl.ntt.co.jp/e2/E2sup_update.pdf

Schneier, B.
Applied Cryptography: Protocols, Algorithms and Source Code in C,
John Wiley & Sons, Inc.: New York, 1996 (2. Auflage)

Schneier, B.; Kelsey, J.; Whiting, D.; Wagner, D., Hall, C.; Ferguson, N.
Performance Comparison of the AES Submissions (Version 2.0),
http://www.counterpane.com/aes-performance.pdf