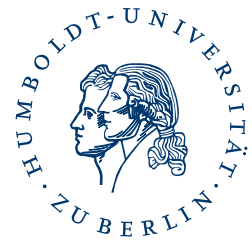
Einführung in die Kryptologie

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

SS 2020

Produktchiffren

- Produktchiffren erhält man durch die sequentielle Anwendung mehrerer Verschlüsselungsverfahren
- Sie können extrem schwer zu brechen sein, auch wenn die einzelnen Komponenten leicht zu brechen sind

Definition

- Seien $KS_1 = (M_1, C_1, E_1, D_1, K_1, S_1)$ und $KS_2 = (M_2, C_2, E_2, D_2, K_2, S_2)$ Kryptosysteme mit $C_1 = M_2$
- Dann ist das ***Produktkryptosystem*** $KS_1 \times KS_2$ von KS_1 und KS_2 definiert als $(M_1, C_2, E, D, K_1 \times K_2, S)$ mit $S = (S_1, S_2)$ und

$$E(k_1, k_2; x) = E_2(k_2, E_1(k_1, x)) \text{ sowie } D(k_1, k_2; y) = D_1(k_1, D_2(k_2, y))$$

für alle $x \in M_1$, $y \in C_2$ und $(k_1, k_2) \in K_1 \times K_2$

- Der Schlüsselraum von $KS_1 \times KS_2$ umfasst also alle Schlüsselpaare $(k_1,k_2) \in K_1 \times K_2$, wobei wir voraussetzen, dass die beiden Schlüssel unabhängig gewählt werden (d.h. es gilt $p(k_1, k_2) = p(k_1)p(k_2)$)

Beispiel

- Man sieht leicht, dass die affine Chiffre $KS = (M, C, K, E, D)$ mit $M = C = \mathcal{A} = \{a_0, \ldots, a_{m-1}\}$ und $K = \mathbb{Z}_m^* \times \mathbb{Z}_m$ das Produkt der
  - multiplikativen Chiffre $KS_1 = (M, C, K_1, E_1, D_1)$ und der
  - additiven Chiffre $KS_2 = (M, C, K_2, E_2, D_2)$ ist
- Für jeden Schlüssel $(k_1, k_2) \in K = K_1 \times K_2 = \mathbb{Z}_m^* \times \mathbb{Z}_m$ gilt

$$E(k_1, k_2; x) = k_1 x + k_2 = E_2(k_2, E_1(k_1, x))$$

- Das ist exakt die affine Chiffre!
- Welche Chiffre ergibt sich, wenn wir die Reihenfolge von KS_1 und KS_2 vertauschen?

Beispiel (Fortsetzung)

- Für $KS_2 \times KS_1$ erhalten wir das Kryptosystem $KS' = (M, C, K', E', D')$, in dem für jeden Schlüssel $(k_2, k_1) \in K' = K_2 \times K_1 = \mathbb{Z}_m \times \mathbb{Z}_m^*$ gilt

$$E'(k_2, k_1; x) = k_1(x + k_2) = k_1 x + k_1 k_2 = E(k_1, k_1 k_2; x)$$

- Die Abbildung $(k_2, k_1) \mapsto (k_1, k_1 k_2)$ ist also eine Bijektion zwischen den Schlüsselräumen K' und K und der Schlüssel (k_2, k_1) realisiert in KS' die gleiche Chiffrierfunktion wie der Schlüssel $(k_1, k_1 k_2)$ in KS
- Zudem können wir jeden Schlüsselgenerator S' für KS' in einen Schlüsselgenerator S für KS transformieren (und auch S wieder zurück in S'), so dass S in KS jede Chiffrierfunktion mit der gleichen Wahrscheinlichkeit erzeugt wie S' in KS'
- Daher können wir die Kryptosysteme $KS = KS_1 \times KS_2$ und $KS' = KS_2 \times KS_1$ als gleich (genauer: äquivalent, siehe Übungen) ansehen, d.h. KS_1 und KS_2 kommutieren ◁

Produktchiffren

Definition

- Ein Kryptosystem $KS = (M, C, K, D, E)$ mit $M = C$ heißt ***endomorph***
- Ein endomorphes Kryptosystem KS heißt ***idempotent***, falls $KS \times KS$ äquivalent zu KS ist (in Zeichen: $KS \times KS = KS$)

Beispiel

- Eine leichte Rechnung zeigt, dass
  - die additive Chiffre,
  - die multiplikative Chiffre und
  - die affine Chiffre idempotent sind
- Dies trifft auch auf
  - die Blocktransposition sowie
  - die Vigenère- und Hill-Chiffre

  zu (siehe Übungen)

◁

Produktchiffren

- Will man durch mehrmalige Anwendung (Iteration) derselben Chiffre eine höhere Sicherheit erreichen, so darf diese nicht idempotent sein
- Man kann versuchen, durch Kombination zweier idempotenter Systeme KS_1 und KS_2 ein System $KS = KS_1 \times KS_2$ zu erhalten, das nicht idempotent ist
- Da KS im Fall $KS_1 \times KS_2 = KS_2 \times KS_1$ wegen

$$\begin{aligned}(KS_1 \times KS_2) \times (KS_1 \times KS_2) &= KS_1 \times (KS_2 \times KS_1) \times KS_2 \\ &= KS_1 \times (KS_1 \times KS_2) \times KS_2 \\ &= (KS_1 \times KS_1) \times (KS_2 \times KS_2) \\ &= KS_1 \times KS_2\end{aligned}$$

idempotent ist, dürfen hierbei KS_1 und KS_2 jedoch nicht kommutieren

Iterierte Blockchiffren

- Ab jetzt werden wir nur noch Blockchiffren über dem Binäralphabet $A = \{0, 1\}$ betrachten und auch der Schlüsselraum wird von der Form $\{0, 1\}^k$ sein, wobei k die Schlüssellänge bezeichnet (einzelne Schlüssel eines Kryptosystems werden wir bis auf weiteres mit K bezeichnen)
- Eine iterierte Blockchiffre wird durch eine Rundenfunktion (*round function*) g und einen Key-Schedule Algorithmus f beschrieben
- Ist N die Rundenzahl, so erzeugt f bei Eingabe eines Schlüssels K eine Folge $f(K) = (K^1, \ldots, K^N)$ von N Rundenschlüsseln K^i für g

Iterierte Blockchiffren

- Mit diesen wird ein Klartext $x = w^0$ durch N-malige Anwendung der Rundenfunktion g zu einem Kryptotext $y = w^N$ verschlüsselt:

$$w^1 := g(K^1, w^0)$$
$$\vdots$$
$$w^N := g(K^N, w^{N-1})$$

- Um y wieder zu entschlüsseln, muss die inverse Rundenfunktion g^{-1} mit umgekehrter Rundenschlüsselfolge $K^N, \ldots, K^1$ benutzt werden:

$$w^{N-1} := g^{-1}(K^N, w^N)$$
$$\vdots$$
$$w^0 := g^{-1}(K^1, w^1)$$

- Beispiele für iterierte Chiffren sind der aus 16 Runden bestehende DES-Algorithmus und der AES mit einer variablen Rundenzahl $N \in \{10, 12, 14\}$, die wir später behandeln werden

Substitutions-Permutations-Netzwerke

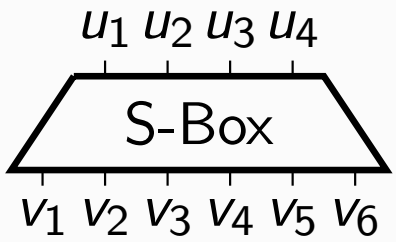
- Als Basisbausteine für die Rundenfunktion von iterierten Blockchiffren eignen sich Substitutionen und Transpositionen besonders gut
- Aus Effizienzgründen sollten die Substitutionen nur eine relativ kleine Blocklänge ℓ haben

Definition

- Für ein Wort $u = u_1 \cdots u_n \in \{0,1\}^n$ und Indizes $1 \leq i \leq j \leq n$ bezeichne $u[i,j]$ das ***Teilwort*** $u_i \cdots u_j$ von u
- Im Fall $n = ml$ bezeichnen wir das Teilwort $u[(i-1)l+1, il]$ auch einfach mit $u_{(i)}$, d.h. es gilt $u = u_{(1)} \cdots u_{(m)}$, wobei $|u_{(i)}| = l$ ist

Substitutions-Permutations-Netzwerke

- Sei $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ eine Substitution, die Binärblöcke u der Länge l in Blöcke $v = \sigma_S(u)$ der Länge l' überführt (auch ***S-Box*** S genannt)



- Für $\sigma_S(u)$ schreiben wir auch einfach $S(u)$
- Durch parallele Anwendung von m Kopien der S-Box S erhalten wir die Substitution $\sigma_{mS} : \{0,1\}^{ml} \to \{0,1\}^{ml'}$ mit

$$\sigma_{mS}(u_1 \cdots u_{ml}) = \sigma_S(u_{(1)}) \cdots \sigma_S(u_{(m)})$$

- Auch hier schreiben wir für $\sigma_{mS}(u_1 \cdots u_{ml})$ auch einfach $S(u_1 \cdots u_{ml})$
- Für die Speicherung einer S-Box S mit $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ auf einem Chip werden $l' \cdot 2^l$ Bit Speicherplatz benötigt (im Fall $l = l'$ also $l2^l$ Bit)
- Für $l = l' = 16$ wären dies beispielsweise 2^{20} Bit, was Smartcard-Anwendungen bereits ausschließen würde
- Für eine Transposition P auf $\{0,1\}^\ell$ bezeichnen wir die zugehörige Permutation auf $[\ell]$ mit π_P oder einfach mit π, falls P aus dem Kontext bekannt ist, d.h. $P(u_1 \cdots u_\ell) = u_{\pi(1)} \cdots u_{\pi(\ell)}$

Substitutions-Permutations-Netzwerke

Definition

- Für natürliche Zahlen $m, l \geq 1$ sei $M = C = \{0,1\}^\ell$ mit $\ell = ml$
- Ein ***Substitutions-Permutations-Netzwerk (SPN)*** wird durch eine S-Box S, eine Blocktransposition P mit Blocklänge $\ell = ml$ und durch eine Funktion $f : \{0,1\}^k \to \{0,1\}^{\ell(N+1)}$ beschrieben
- Hierbei realisiert die S-Box S eine Permutation σ_S auf $\{0,1\}^l$ und N ist die Rundenzahl des SPN
- Die Funktion f transformiert einen (externen) Schlüssel $K \in \{0,1\}^k$ in ein Key-Schedule $f(K) = (K^1, \ldots, K^{N+1})$ von $N+1$ Rundenschlüsseln
- Unter ihnen wird ein Klartext $x \in \{0,1\}^\ell$ durch folgenden Chiffrieralgorithmus in einen Kryptotext $y = E_{f,S,P}(K,x) \in \{0,1\}^\ell$ überführt:

1 $w^0 := x$
2 **for** $r := 1$ **to** $N-1$ **do**
3 $\quad u^r := w^{r-1} \oplus K^r$
4 $\quad v^r := \sigma_{mS}(u^r)$
5 $\quad w^r := P(v^r)$
6 $u^N := w^{N-1} \oplus K^N$
7 $v^N := \sigma_{mS}(u^N)$
8 $y := v^N \oplus K^{N+1}$

Substitutions-Permutations-Netzwerke

Die Chiffrierfunktion $E_{f,S,P}(K, x)$

- Zu Beginn jeder Runde $r \in \{1, \ldots, N\}$ wird w^{r-1} zunächst einer XOR-Operation mit dem Rundenschlüssel K^r unterworfen (*round key mixing*), deren Resultat u^r den S-Boxen zugeführt wird
- Auf die Ausgabe v^r der S-Boxen wird in jeder Runde $r \leq N-1$ die Transposition P angewendet, was die Eingabe w^r für die nächste Runde $r+1$ liefert

1 $w^0 := x$
2 **for** $r := 1$ **to** $N-1$ **do**
3 $\quad u^r := w^{r-1} \oplus K^r$
4 $\quad v^r := S(u^r)$
5 $\quad w^r := P(v^r)$
6 $u^N := w^{N-1} \oplus K^N$
7 $v^N := S(u^N)$
8 $y := v^N \oplus K^{N+1}$

- Am Ende der letzten Runde $r = N$ wird nicht die Transposition P angewandt, sondern der Rundenschlüssel K^{N+1} auf v^N addiert
- Dies wird *whitening* genannt und bewirkt, dass auch für den letzten Chiffrierschritt der Schlüssel benötigt und somit der Gegner an einer partiellen Entschlüsselung des Kryptotexts gehindert wird
- Zudem wird dadurch eine (legale) Entschlüsselung nach fast demselben Verfahren ermöglicht (siehe Übungen)

Beispiel

- Wir betrachten ein SPN SP mit Parametern $l = m = N = 4$ und $k = 32$
- Für f wählen wir die Funktion $f(K) = (K^1, \ldots, K^5)$ mit $K^r = K[4(r-1)+1, 4(r-1)+16]$
- Weiter seien $\sigma_S : \{0,1\}^4 \to \{0,1\}^4$ und $\sigma_P : \{1,\ldots,16\} \to \{1,\ldots,16\}$ die folgenden Permutationen (wobei die Argumente und Werte von σ_S hexadezimal dargestellt sind:

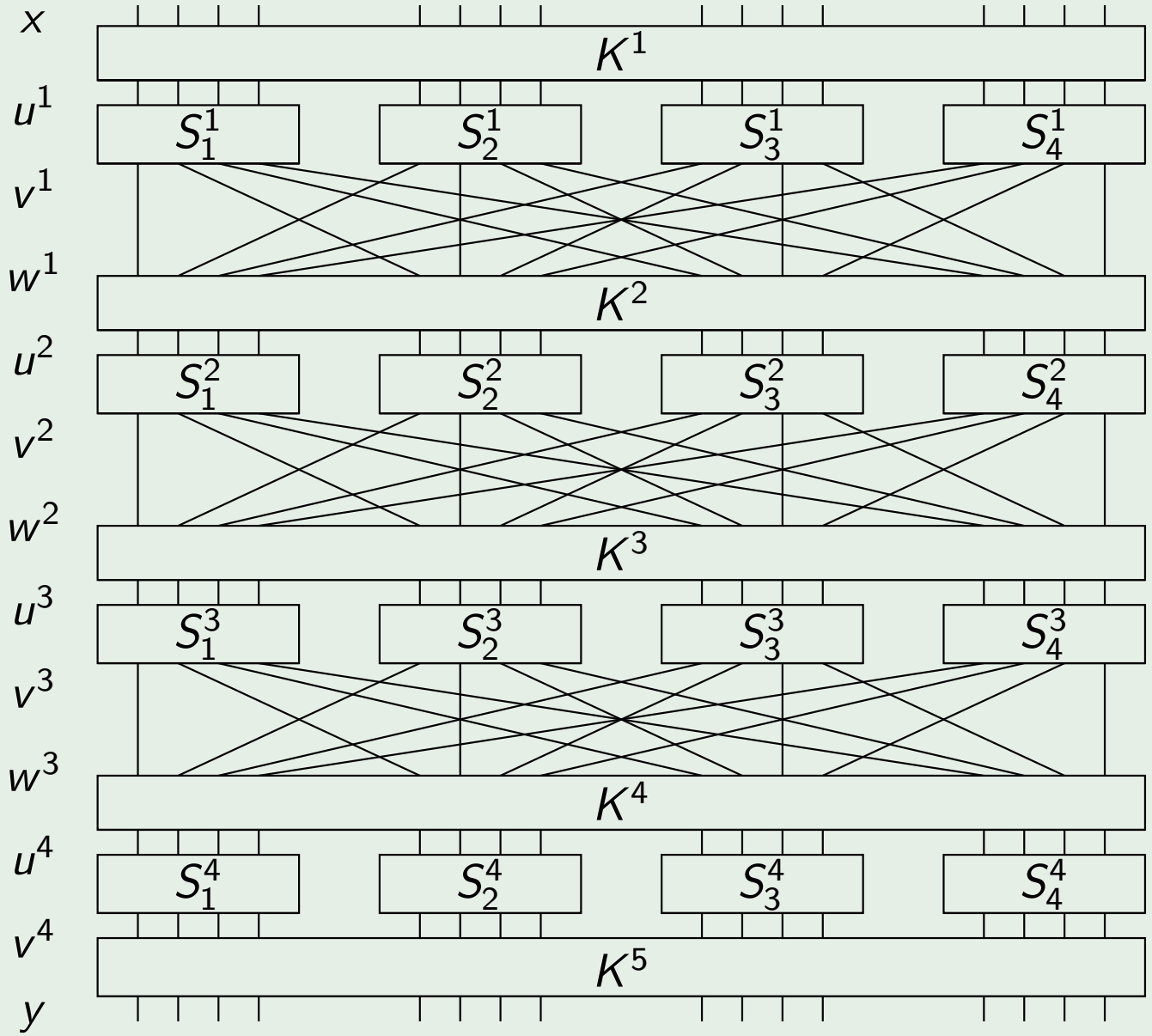
z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\sigma_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

und

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\sigma_P(i)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Substitutions-Permutations-Netzwerke

Beispiel (Fortsetzung)



Substitutions-Permutations-Netzwerke

Beispiel (Schluss)

- Für den Schlüssel $K = 0011\,1010\,1001\,0100\,1101\,0110\,0011\,1111$ liefert f beispielsweise die Rundenschlüssel $f(K) = (K^1, \ldots, K^5)$ mit

$$K^1 = 0011\,1010\,1001\,0100 \qquad K^2 = 1010\,1001\,0100\,1101$$
$$K^3 = 1001\,0100\,1101\,0110 \qquad K^4 = 0100\,1101\,0110\,0011$$
$$K^5 = 1101\,0110\,0011\,1111$$

unter denen der Klartext $x = 0010\,0110\,1011\,0111$ die folgenden Chiffrierschritte durchläuft:

$$\begin{aligned}
x &= 0010\,0110\,1011\,0111 = w^0 \\
w^0 \oplus K^1 &= 0001\,1100\,0010\,0011 = u^1 \\
S(u^1) &= 0100\,0101\,1101\,0001 = v^1 \\
P(v^1) &= 0010\,1110\,0000\,0111 = w^1 \\
&\vdots \\
P(v^3) &= 1110\,0100\,0110\,1110 = w^3 \\
w^3 \oplus K^4 &= 1010\,1001\,0000\,1101 = u^4 \\
S(u^4) &= 0110\,1010\,1110\,1001 = v^4 \\
u^4 \oplus K^5 &= 1011\,1100\,1101\,0110 = y
\end{aligned}$$

◁

Lineare Approximationen

- Sei $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ die funktionale Beschreibung einer S-Box S
- Wählen wir die Eingabe $U = U_1 \cdots U_l$ zufällig unter Gleichverteilung, so gilt für die zugehörige Ausgabe $V = \sigma_S(U) = V_1 \cdots V_{l'}$,

$$\Pr[V = v \mid U = u] = \begin{cases} 1 & \sigma_S(u) = v, \quad \text{für alle } u \in \{0,1\}^l \\ 0 & \text{sonst} \quad\quad\quad\quad\quad \text{und } v \in \{0,1\}^{l'} \end{cases}$$

- Wegen $\Pr[U = u] = 2^{-l}$ folgt

$$\Pr[V = v, U = u] = \begin{cases} 2^{-l} & \sigma_S(u) = v, \\ 0 & \text{sonst} \end{cases}$$

- Wir nennen eine S-Box S linear, wenn σ_S eine lineare Funktion ist, d.h. $\sigma_S(u) = uA$ für eine binäre $(l \times l')$-Matrix A
- In diesem Fall ist jedes Ausgabebit v_j in der Form $v_j = u_{i_1} \oplus \cdots \oplus u_{i_k}$ mit $1 \le i_1 < \cdots < i_k \le l$ darstellbar
- Folglich gilt $\Pr[V_j = U_{i_1} \oplus \cdots \oplus U_{i_k}] = 1$

Lineare Approximationen

- Die Idee bei der linearen Kryptoanalyse ist nun, Gleichungen der Form

  $$V_{j_1} \oplus \cdots \oplus V_{j_{k'}} = U_{i_1} \oplus \cdots \oplus U_{i_k} \oplus c$$

  mit $1 \leq i_1 < \cdots < i_k \leq l$, $1 \leq j_1 < \cdots < j_{k'} \leq l'$ und $c \in \{0,1\}$ zu finden, die mit großer Wahrscheinlichkeit gelten
- Definieren wir für $a \in \{0,1\}^l$ und $b \in \{0,1\}^{l'}$ die Zufallsvariablen

  $$U_a = \bigoplus_{i=1}^{l} a_i U_i \quad \text{und} \quad V_b = \bigoplus_{i=1}^{l'} b_i V_i,$$

  so suchen wir also nach Werten für a, b und c, für die das Ereignis $V_b = U_a \oplus c$ (oder $U_a \oplus V_b = c$) mit großer Wahrscheinlichkeit eintritt
- In diesem Fall lässt sich nämlich der Wert von V_b bei Kenntnis von U_a entsprechend gut vorhersagen
- Wegen $\Pr[U_a \oplus V_b = c] = 1 - \Pr[U_a \oplus V_b = c \oplus 1]$ kommt es nur darauf an, wie stark die Wahrscheinlichkeit $\Pr[U_a \oplus V_b = 0]$ von $1/2$ abweicht
- Die durch das Paar (a, b) beschriebene ***lineare Approximation*** $U_a \oplus V_b$ an die S-Box ist also um so besser, je größer $|\Pr[U_a \oplus V_b = 0] - 1/2|$ ist

Güte einer linearen Approximation an eine S-Box

Definition

- Für eine Zufallsvariable X mit Wertebereich $W(X) = \{0, 1\}$ bezeichne $\varepsilon(X)$ den Wert $\varepsilon(X) = \Pr[X = 0] - 1/2$ (auch ***Bias*** von X genannt)
- Sei $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ die funktionale Beschreibung einer S-Box S und sei $U_a \oplus V_b$ eine durch $(a, b) \in \{0,1\}^l \times \{0,1\}^{l'}$ beschriebene lineare Approximation an S
- Dann ist die ***Güte*** von $U_a \oplus V_b$ definiert als der Absolutbetrag $|\varepsilon(U_a \oplus V_b)|$ des Bias-Wertes von $U_a \oplus V_b$

Lineare Approximationen

Beispiel

- Wir betrachten wieder die S-Box S aus dem letzten Beispiel
- Dann nimmt die Zufallsvariable $(U_1, \ldots, U_4, V_1, \ldots, V_4)$ die 16 Werte in folgender Tabelle jeweils mit Wahrscheinlichkeit $2^{-4} = 1/16$ an

U_1	U_2	U_3	U_4	V_1	V_2	V_3	V_4	$U_3 \oplus U_4 \oplus V_1 \oplus V_4$
0	0	0	0	1	1	1	0	1
0	0	0	1	0	1	0	0	1
0	0	1	0	1	1	0	1	1
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	1	1	1	1	1
0	1	1	0	1	0	1	1	1
0	1	1	1	1	0	0	0	1
1	0	0	0	0	0	1	1	1
1	0	0	1	1	0	1	0	0
1	0	1	0	0	1	1	0	1
1	0	1	1	1	1	0	0	1
1	1	0	0	0	1	0	1	1
1	1	0	1	1	0	0	1	1
1	1	1	0	0	0	0	0	1
1	1	1	1	0	1	1	1	1

Beispiel (Fortsetzung)

- Um nun $\varepsilon(U_a \oplus V_b)$ zu berechnen, genügt es, die Anzahl $L(a,b)$ der Zeilen zu bestimmen, für die $U_a = V_b$ ist
- Dann gilt $\Pr[U_a \oplus V_b = 0] = \Pr[U_a = V_b] = L(a,b)/16$ und somit

$$\varepsilon(U_a \oplus V_b) = L(a,b)/16 - 1/2 = (L(a,b) - 8)/16$$

- Für $a = 0011$ und $b = 1001$ gibt es z.B. $L(a,b) = 2$ Zeilen (Zeile 5 und Zeile 10) mit $U_a = U_3 \oplus U_4 = V_b = V_1 \oplus V_4$, d.h.
  $\varepsilon(U_3 \oplus U_4 \oplus V_1 \oplus V_4) = (L(a,b) - 8)/16 = -(3/8)$
- Die folgende Tabelle zeigt für alle Werte von a und b (hexadezimal dargestellt) die Anzahlen $L(a,b)$

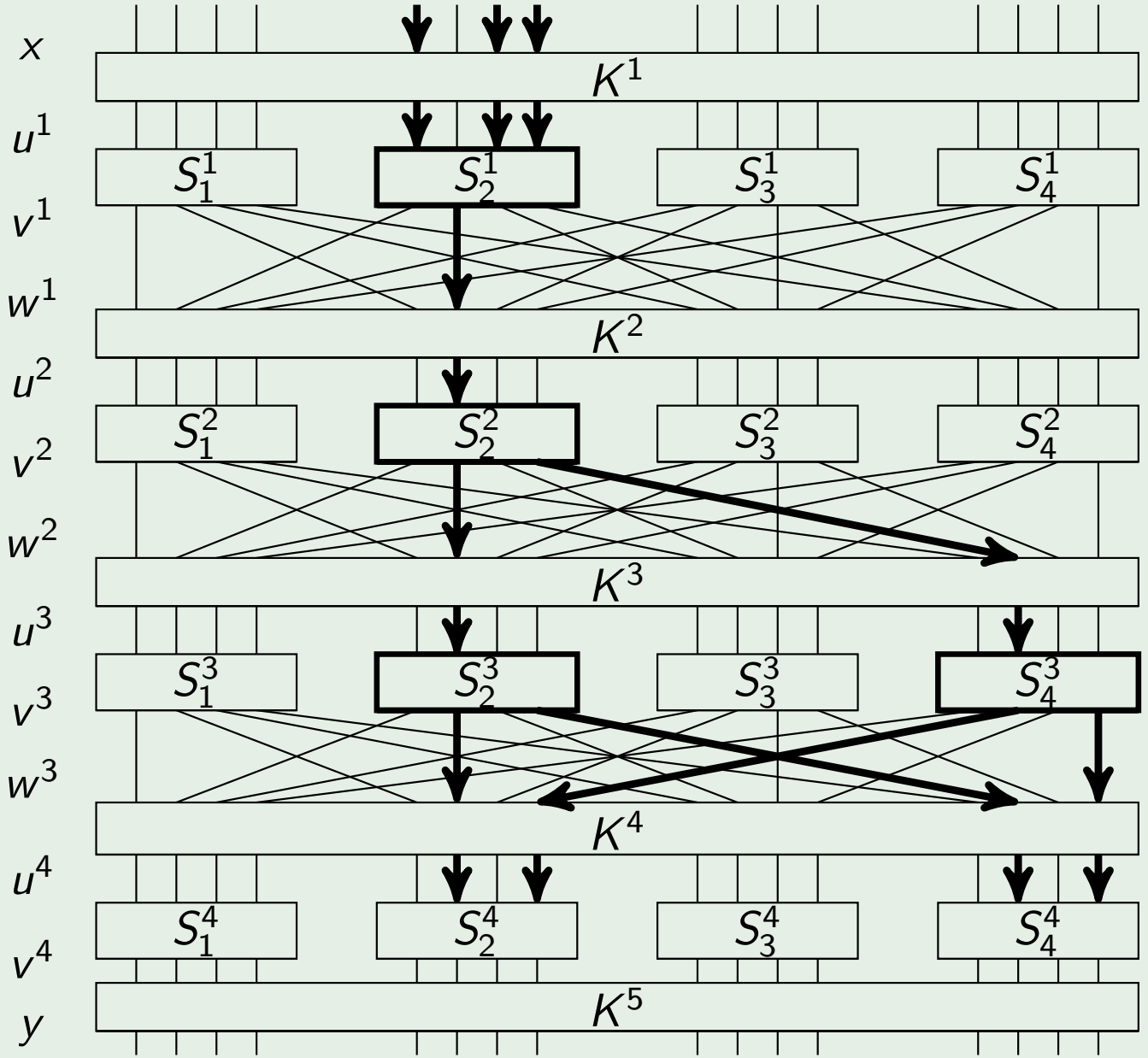
Beispiel (Schluss)

	b															
a	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	8	6	6	8	8	6	14	10	10	8	8	10	10	8	8
2	8	8	6	6	8	8	6	6	8	8	10	10	8	8	2	10
3	8	8	8	8	8	8	8	8	10	**2**	6	6	10	10	6	6
4	8	10	8	6	6	4	6	8	8	6	8	10	10	4	10	8
								⋮								
B	8	12	8	4	12	8	12	8	8	8	8	8	8	8	8	8
								⋮								
F	8	6	4	6	6	8	10	8	8	6	12	6	6	8	10	8

◁

Substitutions-Permutations-Netzwerke

Beispiel (Fortsetzung)



Lineare Kryptoanalyse von SPNs

- Seien $K^1, \ldots, K^5$ die aus K berechneten Rundenschlüssel (diese sind wie K unbekannt, aber konstant)
- Das Ziel besteht zunächst einmal darin, eine lineare Approximation für die Abbildung $x \mapsto u^4$ zu finden, bei der nur die ersten vier Rundenschlüssel $K^1, \ldots, K^4$ benutzt werden
- Hierzu verwenden wir die beiden linearen Approximationen

$$T = U_1 \oplus U_3 \oplus U_4 \oplus V_2 \quad \text{und} \quad T' = U_2 \oplus V_2 \oplus V_4$$

an die S-Box S mit den Bias-Werten

  - $\varepsilon(T) = (L(B,4) - 8)/16 = (12-8)/16 = {}^1\!/_4$ und
  - $\varepsilon(T') = (L(4,5) - 8)/16 = (4-8)/16 = {}^{-1}\!/_4$

(also $\Pr[T=0] = \Pr[T'=1] = 3/4$)

Lineare Kryptoanalyse von SPNs

- Konkret verwenden wir T für die S-Box S_2^1:

  $$T_1 = U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus V_6^1$$

  und T' für die drei S-Boxen S_2^2, S_2^3, S_4^3:

  $$T_2 = U_6^2 \oplus V_6^2 \oplus V_8^2,\ T_3 = U_6^3 \oplus V_6^3 \oplus V_8^3,\ T_4 = U_{14}^3 \oplus V_{14}^3 \oplus V_{16}^3$$

- Nun schalten wir die vier linearen Approximationen $T_1, \ldots, T_4$ an die S-Boxen S_2^1, S_2^2, S_2^3 und S_4^3 zu einer linearen Approximation

  $$L = \underbrace{X_5 \oplus X_7 \oplus X_8}_{X_a \text{ für } a=0\text{B}00} \oplus \underbrace{U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4}_{U_b^4 \text{ für } b=0505} = X_a \oplus U_b^4$$

  an die Abbildung $x \mapsto u^4$ zusammen

- Dann gilt für ein Bit $c \in \{0, 1\}$ die Gleichung

  $$X_a \oplus U_b^4 = T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus c$$

Lineare Kryptoanalyse von SPNs

- An dieser Stelle ergeben sich nun folgende drei Fragen:
  1. Warum gilt die Gleichung $L = T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus c$?
  2. Wie gut ist die lineare Approximation L an die Abbildung $x \mapsto u^4$?
  3. Wie können wir mit ihrer Hilfe einzelne Schlüsselbits bestimmen?

Antwort auf Frage 1

- Sei $c = c_1 \oplus c_2 \oplus c_3 \oplus c_4$ mit

$$c_1 = K_5^1 \oplus K_7^1 \oplus K_8^1, \quad c_2 = K_6^2, \quad c_3 = K_6^3 \oplus K_{14}^3, \quad c_4 = K_6^4 \oplus K_8^4 \oplus K_{14}^4 \oplus K_{16}^4$$

- Dann gilt

$$\begin{aligned}
&X_5 \oplus X_7 \oplus X_8 \\
&\quad = U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus c_1 \\
&\quad = T_1 \oplus V_6^1 \oplus c_1 \\
&\quad = T_1 \oplus W_6^1 \oplus c_1 \\
&\quad = T_1 \oplus U_6^2 \oplus c_1 \oplus c_2 \\
&\quad = T_1 \oplus T_2 \oplus V_6^2 \oplus V_8^2 \oplus c_1 \oplus c_2 \\
&\quad = T_1 \oplus T_2 \oplus W_6^2 \oplus W_{14}^2 \oplus c_1 \oplus c_2 \\
&\quad = T_1 \oplus T_2 \oplus U_6^3 \oplus U_{14}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&\quad = T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus V_6^3 \oplus V_8^3 \oplus V_{14}^3 \oplus V_{16}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&\quad = T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus W_6^3 \oplus W_8^3 \oplus W_{14}^3 \oplus W_{16}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&\quad = T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 \oplus \underbrace{c_1 \oplus c_2 \oplus c_3 \oplus c_4}_{c}
\end{aligned}$$

Lineare Kryptoanalyse von SPNs

- Nun zu Frage 2: Wären $T_1, \ldots, T_4$ unabhängig, so würde uns das Piling-up-Lemma (siehe unten) folgende Bias-Werte liefern:

$$2^3(1/4)(-1/4)^3 = -1/32 \text{ für } T_1 \oplus \cdots \oplus T_4 \text{ und } (-1)^{c+1}/32 \text{ für } L$$

- Sind nämlich X_1, X_2 unabhängige Zufallsvariablen mit Wertebereich $W(X_i) = \{0, 1\}$ und Bias $\varepsilon_i = \varepsilon(X_i)$, dann ist

$$\begin{aligned}\Pr[X_1 \oplus X_2 = 0] &= \Pr[X_1 = X_2 = 0] + \Pr[X_1 = X_2 = 1] \\ &= (1/2 + \varepsilon_1)(1/2 + \varepsilon_2) + (1/2 - \varepsilon_1)(1/2 - \varepsilon_2) \\ &= 1/2 + 2\varepsilon_1\varepsilon_2\end{aligned}$$

und $\Pr[X_1 \oplus X_2 = 1] = 1/2 - 2\varepsilon_1\varepsilon_2$, d.h. es gilt $\varepsilon(X_1 \oplus X_2) = 2\varepsilon_1\varepsilon_2$

Diese Beobachtung lässt sich wie folgt verallgemeinern

Lemma (Piling-up Lemma)

Für unabhängige $\{0, 1\}$-wertige Zufallsvariablen $X_1, \ldots, X_n$ mit $\varepsilon_i = \varepsilon(X_i)$ gilt

$$\varepsilon(X_1 \oplus \cdots \oplus X_n) = 2^{n-1} \prod_{i=1}^{n} \varepsilon_i$$

Lemma (Piling-up Lemma)

Für unabhängige $\{0,1\}$-wertige Zufallsvariablen $X_1, \ldots, X_n$ mit $\varepsilon_i = \varepsilon(X_i)$ gilt

$$\varepsilon(X_1 \oplus \cdots \oplus X_n) = 2^{n-1} \prod_{i=1}^{n} \varepsilon_i$$

Beweis. Wir führen den Beweis durch Induktion über n:

Induktionsanfang ($n \leq 2$): Bereits bewiesen

Induktionsschritt ($n \rightsquigarrow n+1$): Nach IV hat die Zufallsvariable $Z = X_1 \oplus \cdots \oplus X_n$ den Bias

$$\varepsilon(Z) = 2^{n-1} \varepsilon(X_1) \cdots \varepsilon(X_n)$$

und daher folgt

$$\begin{aligned} \varepsilon(X_1 \oplus \cdots \oplus X_{n+1}) &= \varepsilon(Z \oplus X_{n+1}) = 2\varepsilon(Z)\varepsilon_{n+1} \\ &= 2^n \varepsilon_1 \cdots \varepsilon_{n+1} \end{aligned}$$

□

Beispiel

- Seien X_1, X_2, X_3 unabhängige Zufallsvariablen mit $\varepsilon(X_i) = 1/4$ für $i = 1, 2, 3$
- Dann liefert das Piling-up Lemma die Bias-Werte $\varepsilon(X_i \oplus X_j) = 1/8$ für $1 \leq i < j \leq 3$
- Da die Zufallsvariablen $Y = X_1 \oplus X_2$ und $Z = X_2 \oplus X_3$ nicht unabhängig sind, lässt sich $\varepsilon(Y \oplus Z)$ nicht mit dem Piling-up-Lemma bestimmen
- Dieses würde nämlich für $Y \oplus Z$ einen Bias-Wert von $2(1/8)^2 = 1/32$ ergeben, wogegen

$$Y \oplus Z = (X_1 \oplus X_2) \oplus (X_2 \oplus X_3) = X_1 \oplus X_3$$

und daher $\varepsilon(Y \oplus Z) = \varepsilon(X_1 \oplus X_3) = 1/8$ ist ◁

Antwort auf Frage 2

- Zwar sind die Zufallsvariablen T_i, aus denen eine lineare Approximation $X_a \oplus U_b^N = T_1 \oplus \cdots \oplus T_k \oplus c$ an die Abbildung $x \mapsto u^N$ gebildet wird, in der Regel nicht unabhängig
- Dennoch zeigt sich in praktischen Anwendungen, dass der Bias-Wert $\varepsilon(T_1 \oplus \cdots \oplus T_k)$ von $T_1 \oplus \cdots \oplus T_k$ meist nicht zu sehr von dem "hypothetischen" Wert $2^{k-1} \prod_{i=1}^{k} \varepsilon(T_i)$ abweicht, welcher sich aus dem Piling-up Lemma ergeben würde
- Daher können wir in unserem Beispiel

  $$\varepsilon(T_1 \oplus \cdots \oplus T_4) \approx -1/32$$

  bzw.

  $$\Pr[U_{0505}^4 = X_{0\mathrm{B}00}] \approx 1/2 + (-1)^{c+1}/32$$

  annehmen

Antwort auf Frage 3

- Wir betrachten zuerst den (für den Gegner günstigen) Fall, dass die lineare Approximation $X_a \oplus U_b^4$ an $x \mapsto u^4$ den Bias-Wert $1/2$ hat (d.h. alle Klartexte x führen auf einen Vektor u^4 mit $x_a \oplus u_b^4 = 0$)
- Wir berechnen für jedes Paar $(x, y) \in M$ das Bit $x_a = x_5 \oplus x_7 \oplus x_8$
- Da wir y und σ_S^{-1} kennen, können wir zudem für jeden Subschlüssel-Kandidaten (engl. candidate subkey) (I, J) für den Teilschlüssel $(K_{(2)}^5, K_{(4)}^5)$ von K^5 aus dem Kryptotext y die zugehörigen u^4-Blöcke

$$u_{(2)}^4(I, J) = \sigma_S^{-1}(y_{(2)} \oplus I) \text{ und } u_{(4)}^4(I, J) = \sigma_S^{-1}(y_{(4)} \oplus J)$$

  zurückrechnen (die beiden anderen u^4-Blöcke werden nicht benötigt)
- Der richtige Kandidat $(I, J) = (K_{(2)}^5, K_{(4)}^5)$ besteht dann für alle Paare (x, y) in M den Gleichheitstest

$$x_a = u_b^4(I, J) \qquad (*)$$

Antwort auf Frage 3 (Fortsetzung)

- Der richtige Kandidat $(I, J) = (K^5_{(2)}, K^5_{(4)})$ erfüllt dann für alle Paare (x, y) in M den Gleichheitstest

$$x_a = u^4_b(I, J) \qquad (*)$$

- Dagegen besteht von den falschen Kandidaten $(I, J) \neq (K^5_{(2)}, K^5_{(4)})$ nur etwa die Hälfte den Test $(*)$
- Falls wir also alle Subkey-Kandidaten (I, J) dem Gleichheitstest $(*)$ für eine hinreichend große Menge von Klartext-Kryptotext-Paaren (x, y) unterziehen, können wir den richtigen Kandidaten daran erkennen, dass er als einziger alle Tests besteht

Antwort auf Frage 3 (Schluss)

- Nun zum Fall, dass der Bias-Wert ε der linearen Approximation $L = X_a \oplus U_b^4$ an die Abbildung $x \mapsto u^4$ zwar nicht gleich $1/2$ ist, aber genügend weit von Null abweicht
- In diesem Fall besteht der richtige Kandidat $(I, J) = (K_{(2)}^5, K_{(4)}^5)$ bei einer repräsentativen Auswahl M von Klartext-Kryptotext-Paaren ungefähr einen Anteil von $(1/2 + \varepsilon)$ der durchgeführten Tests
- Dagegen bestehen die falschen Kandidaten etwa die Hälfte aller Tests
- Falls wir also genügend viele Paare (x, y) kennen, können wir den richtigen Kandidaten nun daran erkennen, dass die Anzahl der von ihm bestandenen Tests am weitesten von $\|M\|/2$ abweicht

Algorithmus LinearAttack

- Der Algorithmus LinearAttack (siehe nächste Folie) ermittelt für jeden Subkey-Kandidaten (I, J) die Anzahl $\alpha(I, J)$ der Paare $(x, y) \in M$, für die er den Gleichheitstest $x_a = u_b^4(I, J)$ besteht
- Ausgegeben wird derjenige Kandidat (I, J), der die Abweichung der Anzahl $\alpha(I, J)$ von $\|M\|/2$ maximiert
- Im allgemeinen werden für eine erfolgreiche lineare Attacke circa $t \approx c\varepsilon^{-2}$ Klartext-Kryptotext-Paare benötigt
- Dabei ist c eine „kleine“ Konstante (im aktuellen Beispiel reichen $t \approx 8000$ Paare; d.h. $c \approx 8$, da $\varepsilon^{-2} \approx 1024$ ist)

Algorithmus LinearAttack

1 **for** $(I, J) := (0,0)$ **to** (F,F) **do** $\alpha(I, J) := 0$
2 **for** each $(x, y) \in M$ **do**
3 **for** $(I, J) := (0,0)$ **to** (F,F) **do**
4 $v^4_{(2)} := I \oplus y_{(2)}$
5 $v^4_{(4)} := J \oplus y_{(4)}$
6 $u^4_{(2)} := \sigma_S^{-1}(v^4_{(2)})$
7 $u^4_{(4)} := \sigma_S^{-1}(v^4_{(4)})$
8 **if** $x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16} = 0$ **then**
9 $\alpha(I, J) := \alpha(I, J) + 1$
10 $max := -1$
11 **for** $(I, J) := (0,0)$ **to** (F,F) **do** $\beta(I, J) := |\alpha(I, J) - {}^t/_2|$
12 **if** $\beta(I, J) > max$ **then**
13 $max := \beta(I, J)$
14 $maxkey := (I, J)$
15 **output**(maxkey)

Differentielle Kryptoanalyse von SPNs

- Wie die lineare hat auch die differentielle Kryptoanalyse das Ziel, den unbekannten Schlüssel K zu finden
- Für die Durchführung wird jedoch frei wählbarer Klartext benötigt
- Genauer basiert der Angriff auf einer Menge M von t Doppelpaaren (x, x^*, y, y^*) mit der Eigenschaft, dass $E_K(x) = y$ und $E_K(x^*) = y^*$ ist und alle Klartext-Paare (x, x^*) die gleiche Differenz $x' = x \oplus x^*$ bilden

Definition

- Seien $u, u^* \in \{0,1\}^l$ Eingaben für eine S-Box $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ und seien $v = \sigma_S(u)$ und $v^* = \sigma_S(u^*)$ die zugehörigen Ausgaben
- Dann heißt $u' = u \oplus u^*$ die ***Eingabedifferenz*** (input-xor) und $v' = \sigma_S(u) \oplus \sigma_S(u^*)$ die ***Ausgabedifferenz*** (output-xor) von (u, u^*)
- Für eine vorgegebene Eingabedifferenz $a' \in \{0,1\}^l$ sei weiter

$$\Delta(a') = \{(u, u^*) \mid u \oplus u^* = a'\} = \{(u, u \oplus a') \mid u \in \{0,1\}^l\}$$

die Menge aller Eingabepaare, die die Differenz a' realisieren

Differentielle Kryptoanalyse von SPNs

- Berechnen wir für alle Paare $(u, u^*) \in \Delta(a')$ die zugehörigen Ausgabedifferenzen, so verteilen sich diese auf die $2^{l'}$ möglichen Werte in $\{0,1\}^{l'}$
- Man beachte, dass im Fall einer affinen S-Box $\sigma_S(u) = uA \oplus w$ alle Paare $(u, u^*) \in \Delta(a')$ auf dieselbe Ausgabedifferenz führen:

  $$\sigma_S(u) \oplus \sigma_S(u^*) = (u \oplus u^*)A = u'A = a'A$$

  (hierbei ist A eine $(l \times l')$-Matrix und $w \in \{0,1\}^{l'}$ ein Vektor)
- Ist S nicht affin, können die Eingabepaare $(u, u^*) \in \Delta(a')$ zu unterschiedlichen Ausgabedifferenzen $\sigma_S(u) \oplus \sigma_S(u^*)$ führen
- Um einer differentiellen Kryptoanalyse widerstehen zu können, sollten die auftretenden Ausgabedifferenzen möglichst gleichmäßig verteilt sein

Differentielle Kryptoanalyse von SPNs

Definition

- Ein ***Differential*** für eine S-Box $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ ist ein Paar $(a', b') \in \{0,1\}^l \times \{0,1\}^{l'}$
- Dabei heißt a' die ***Eingabe-*** und b' die ***Ausgabedifferenz*** des Differentials
- Die Anzahl der Eingabepaare (u, u^*), die die Eingabedifferenz a' in die Ausgabedifferenz b' überführen, bezeichnen wir mit $D(a', b')$, d.h.

$$D(a', b') = \|\{(u, u^*) \in \Delta(a') \mid \sigma_S(u) \oplus \sigma_S(u^*) = b'\}\|$$

- Der ***Weitergabequotient*** (engl. propagation ratio) von S für ein Differential (a', b') ist

$$Q(a', b') = \frac{D(a', b')}{2^l}$$

- $Q(a', b')$ ist also die (bedingte) Wahrscheinlichkeit

$$\Pr[V' = b' | U' = a'] = \Pr[\underbrace{\sigma_S(U) \oplus \sigma_S(U^*)}_{V'} = b' | \underbrace{U \oplus U^*}_{U'} = a'],$$

dass zwei zufällig gewählte Eingaben U und U^* die Ausgabedifferenz $V' = b'$ erzeugen, wenn sie die Eingabedifferenz $U' = a'$ haben

Differentielle Kryptoanalyse von SPNs

Beispiel

- Betrachten wir die S-Box $\sigma_S : \{0,1\}^4 \to \{0,1\}^4$ aus obigem Beispiel, so erhalten wir für $a' = 1011$ folgende Menge von Eingabepaaren

$$\Delta(a') = \{(0000, 1011), \ldots, (1111, 0100)\},$$

- Diese führen auf die folgenden Ausgabedifferenzen $v' = \underbrace{\sigma_S(u)}_{v} \oplus \underbrace{\sigma_S(u^*)}_{v*}$:

u	u^*	v	v^*	v'
0000	1011	1110	1100	0010
0001	1010	0100	0110	0010
0010	1001	1101	1010	0111
0011	1000	0001	0011	0010
0100	1111	0010	0111	0101
0101	1110	1111	0000	1111
0110	1101	1011	1001	0010
0111	1100	1000	0101	1101

u	u^*	v	v^*	v'
1000	0011	0011	0001	0010
1001	0010	1010	1101	0111
1010	0001	0110	0100	0010
1011	0000	1100	1110	0010
1100	0111	0101	1000	1101
1101	0110	1001	1011	0010
1110	0101	0000	1111	1111
1111	0100	0111	0010	0101

Beispiel (Fortsetzung)

- Die Ausgabedifferenz $b' = 0010$ kommt also $D(a', 0010) = 8$ Mal vor, während die Differenzen 0101, 0111, 1101 und 1111 je zwei Mal und die übrigen Werte überhaupt nicht vorkommen (siehe Zeile `B` in nachfolgender Tabelle)
- Führen wir diese Berechnungen für jede der $2^4 = 16$ Eingabedifferenzen $a' \in \{0,1\}^4$ aus, so erhalten wir die folgenden Werte für die Häufigkeiten $D(a', b')$ der Ausgabedifferenz b' bei Eingabedifferenz a' (a' und b' sind hexadezimal dargestellt):

Differentielle Kryptoanalyse von SPNs

Beispiel (Schluss)

- Die Tabelle zeigt die Häufigkeiten $D(a', b')$ der Ausgabedifferenzen b' der S-Box S für eine Auswahl von Eingabedifferenzen $a' \in \{0,1\}^4$

	b'															
a'	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
⋮								⋮								
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
⋮								⋮								
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

◁

Differentielle Kryptoanalyse von SPNs

- Wir versuchen nun, für bestimmte S-Boxen S_i^r Differentiale (a', b') zu finden, so dass die Eingabedifferenz dieser Differentiale mit der (permutierten) Ausgabedifferenz in der vorhergehenden Runde übereinstimmt (siehe nächste Folie)
- Falls dies gelingt, können wir diese Differentiale zu einer so genannten ***Differentialspur*** (differential trail) zusammensetzen
- Falls die ausgewählten S-Boxen S_i^r (diese werden auch als ***aktiv*** bezeichnet) den zugeordneten Differentialen (a'_i, b'_i) unabhängig voneinander folgen, lässt sich der Weitergabequotient der Spur als das Produkt der Weitergabequotienten der beteiligten Differentiale berechnen
- Obwohl die Unabhängigkeit i.a. nicht gegeben ist, weicht der tatsächliche Wert in praktischen Anwendungen kaum von diesem hypothetischen Wert ab

Differentielle Kryptoanalyse von SPNs

Beispiel

- Betrachten wir das SPN SP aus dem vorigen Beispiel, so lassen sich folgende Differentiale zu einer Spur kombinieren:
  - für S_2^1 das Differential $(1011, 0010) = (\mathtt{B}, 2)$ mit $Q(\mathtt{B}, 2) = 1/2$
  - für S_3^2 das Differential $(0100, 0110) = (4, 6)$ mit $Q(4, 6) = 3/8$ und
  - für S_2^3 und S_3^3 das Differential $(0010, 0101) = (2, 5)$ mit $Q(2, 5) = 3/8$
- Gemäß dieser Spur führt also die Klartextdifferenz

  $$x' = 0000\,1011\,0000\,0000$$

  mit hypothetischer Wahrscheinlichkeit $1/2(3/8)^3$ auf die Differenz

  $$(v^3)' = 0000\,0101\,0101\,0000,$$

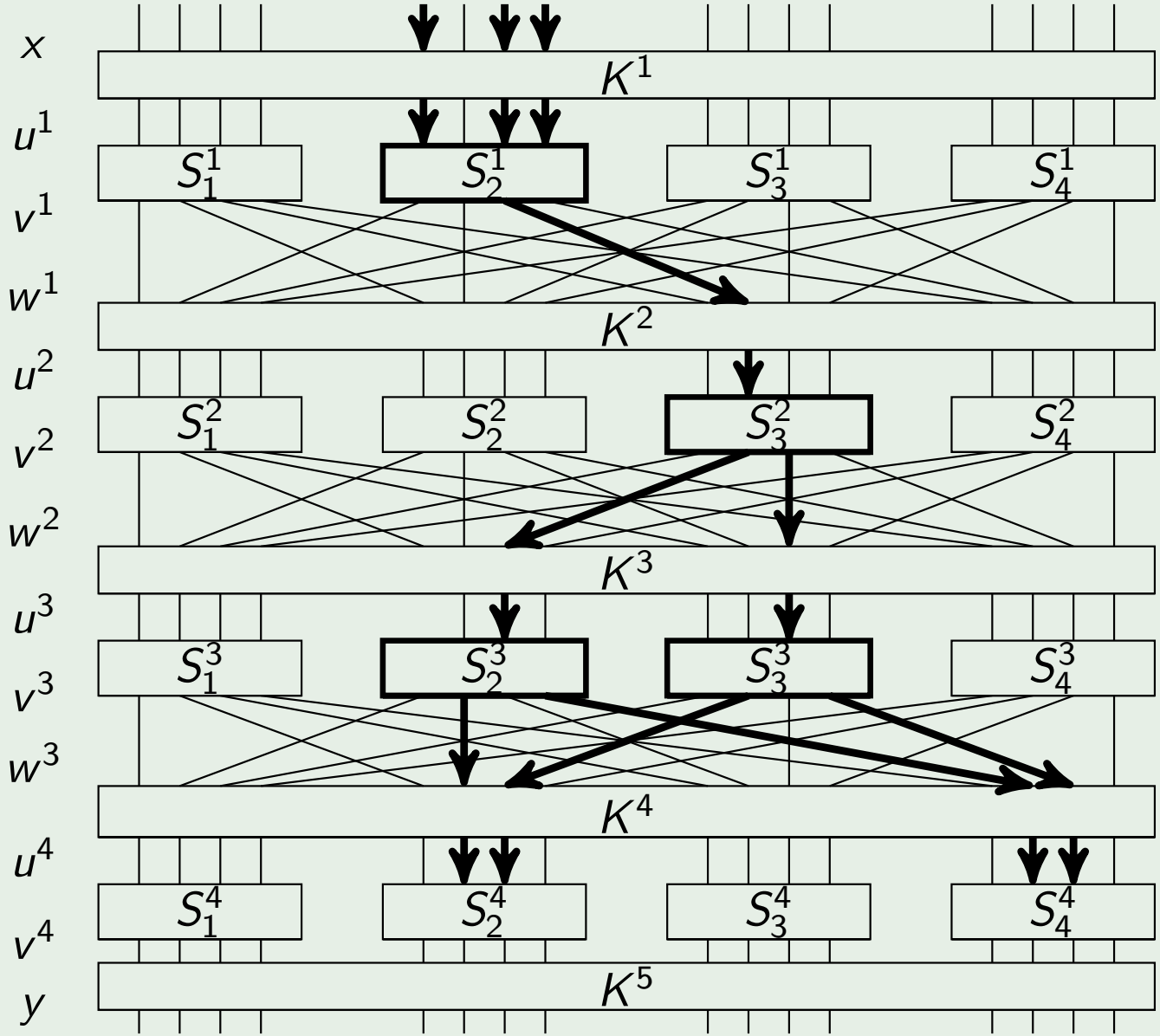
  und diese führt mit Wahrscheinlichkeit 1 auf die Differenz

  $$(u^4)' = 0000\,0110\,0000\,0110$$

- Das Differential $(a', b') = (0000\,1011\,0000\,0000, 0000\,0110\,0000\,0110)$ für die Abbildung $x \mapsto u^4$ hat also einen hypothetischen Weitergabequotienten von $\varepsilon = Q(a', b') = 1/2(3/8)^3 = 27/1024 \approx 0,026$ ◁

Differentielle Kryptoanalyse von SPNs

Beispiel (Fortsetzung)



Differentielle Kryptoanalyse von SPNs

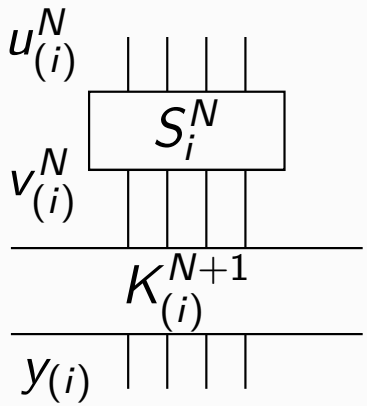
- Sei nun (a', b') ein Differential für die Abbildung $x \mapsto u^4$ mit einem hypothetischen Weitergabequotienten $\varepsilon = Q(a', b')$
- Weiter sei M eine Menge von t Doppelpaaren (x, x^*, y, y^*), die
  - alle mit dem gleichen unbekannten Schlüssel K erzeugt wurden und
  - zusätzlich die Bedingung $x' = x \oplus x^* = a'$ erfüllen
- Dann wird ca. ein ε-Anteil dieser Paare der gewählten Differentialspur folgen (solche Paare werden als ***richtige Doppelpaare*** bezeichnet)
- Alle richtigen Paare führen unter K auf Blöcke u^4 und $(u^4)^*$ mit
$$(u^4)' = u^4 \oplus (u^4)^* = b'$$
- Ein Großteil der falschen Doppelpaare lässt sich daran erkennen, dass die Kryptotext-Differenzen y' nicht die erwarteten $0'$-Blöcke aufweisen (im aktuellen Beispiel sind dies die Blöcke $y'_{(1)}$ und $y'_{(3)}$)
- Es empfiehlt sich, diese Doppelpaare auszufiltern, da sie (wie alle falschen Doppelpaare) nur „Hintergrundrauschen“ erzeugen und somit die Bestimmung des Schlüssels eher behindern

Differentielle Kryptoanalyse von SPNs

Beobachtung

- Für die Ausgabe $v_{(i)}^N$ der S-Box S_i^N in Runde N gilt

$$v_{(i)}^N = y_{(i)} \oplus K_{(i)}^{N+1}$$



- und die Eingabe $u_{(i)}^N$ der S-Box S_i^N in Runde N ist

$$u_{(i)}^N = \sigma_S^{-1}(v_{(i)}^N) = \sigma_S^{-1}(y_{(i)} \oplus K_{(i)}^{N+1})$$

- Falls die S-Box S_i^N nicht affin ist, hängt die aus den Kryptotextblöcken $y_{(i)}$ und $(y_{(i)})^*$ zurückgerechnete Eingabedifferenz

$$(u_{(i)}^N)' = u_{(i)}^N \oplus (u_{(i)}^N)^* = \sigma_S^{-1}(y_{(i)} \oplus K_{(i)}^{N+1}) \oplus \sigma_S^{-1}((y_{(i)})^* \oplus K_{(i)}^{N+1})$$

  von dem Schlüsselblock $K_{(i)}^{N+1}$ ab
- Ist also (x, x^*, y, y^*) ein richtiges Paar, so sind neben $y_{(i)}$ und $y_{(i)}^*$ auch die Eingabedifferenzen $(u_{(i)}^N)' = b'_{(i)}$ von S_i^N bekannt

Differentielle Kryptoanalyse von SPNs

Beobachtung (Fortsetzung)

- Ist also (x, x^*, y, y^*) ein richtiges Paar, so sind neben $y_{(i)}$ und $y^*_{(i)}$ auch die Eingabedifferenzen $(u^N_{(i)})' = b'_{(i)}$ von S^N_i bekannt
- Folglich kommen nur solche Subkey-Werte J für $K^{N+1}_{(i)}$ infrage, für die

$$\sigma_S^{-1}(y_{(i)} \oplus J) \oplus \sigma_S^{-1}(y^*_{(i)} \oplus J) = b'_{(i)} \qquad (*)$$

  ist
- Erfüllt J Gleichung $(*)$, so sagen wir auch, J ist mit dem Doppelpaar (x, x^*, y, y^*) ***konsistent***

- Gemäß dieser Beobachtung lassen sich mit jedem richtigen Doppelpaar einige Kandidaten für den Rundenschlüsselblock $K^{N+1}_{(i)}$ ausschließen
- Ist M hinreichend groß, lässt sich der richtige Schlüsselblock daran erkennen, dass er mit den meisten Doppelpaaren konsistent ist

Differentielle Kryptoanalyse von SPNs

Wir führen nun mit der Spur aus obigem Beispiel einen Angriff mittels differentieller Analyse auf das SPN SP durch

Beispiel

- Der Algorithmus DifferentialAttack (siehe nächste Folie) bestimmt für jeden Subschlüssel-Kandidaten (I, J) für $(K^5_{(2)}, K^5_{(4)})$ die Anzahl $\gamma(I, J)$ aller Doppelpaare (x, x^*, y, y^*) in M, die mit (I, J) konsistent sind
- Dabei bleiben alle als falsch erkannten Paare unberücksichtigt (siehe Zeile 3)
- Ausgegeben wird der Kandidat (I, J) mit dem größten γ-Wert
- Im allgemeinen werden für eine erfolgreiche differentielle Attacke circa $t \approx c\varepsilon^{-1}$ Klartext-Kryptotext-Doppelpaare benötigt
- Dabei ist ε der Weitergabequotient der Spur und c eine Konstante (im Beispiel reichen $t \approx 80$ Doppelpaare, wobei $\varepsilon^{-1} \approx 38$ ist, d.h. $c \approx 2$) $\triangleleft$

Algorithmus DifferentialAttack

1 **for** $(I, J) := (0,0)$ **to** (F,F) **do** $\gamma(I, J) := 0$
2 **for** each $(x, x^*, y, y^*) \in M$ **do**
3 **if** $y_{(1)} = y^*_{(1)}$ und $y_{(3)} = y^*_{(3)}$ **then**
4 **for** $(I, J) := (0,0)$ **to** (F,F) **do**
5 $v^4_{(2)} := I \oplus y_{(2)}$; $v^4_{(4)} := J \oplus y_{(4)}$
6 $u^4_{(2)} := \sigma_S^{-1}(v^4_{(2)})$; $u^4_{(4)} := \sigma_S^{-1}(v^4_{(4)})$
7 $(v^4_{(2)})^* := I \oplus y^*_{(2)}$; $(v^4_{(4)})^* := J \oplus y^*_{(4)}$
8 $(u^4_{(2)})^* := \sigma_S^{-1}((v^4_{(2)})^*)$; $(u^4_{(4)})^* := \sigma_S^{-1}((v^4_{(4)})^*)$
9 $(u^4_{(2)})' := u^4_{(2)} \oplus (u^4_{(2)})^*$; $(u^4_{(4)})' := u^4_{(4)} \oplus (u^4_{(4)})^*$
10 **if** $(u^4_{(2)})' = 0110$ und $(u^4_{(4)})' = 0110$ **then** $\gamma(I, J) := \gamma(I, J) + 1$
11 $max := -1$
12 **for** $(I, J) := (0,0)$ **to** (F,F) **do**
13 **if** $\gamma(I, J) > max$ **then**
14 $max := \gamma(I, J)$; $maxkey := (I, J)$
15 **output**(maxkey)