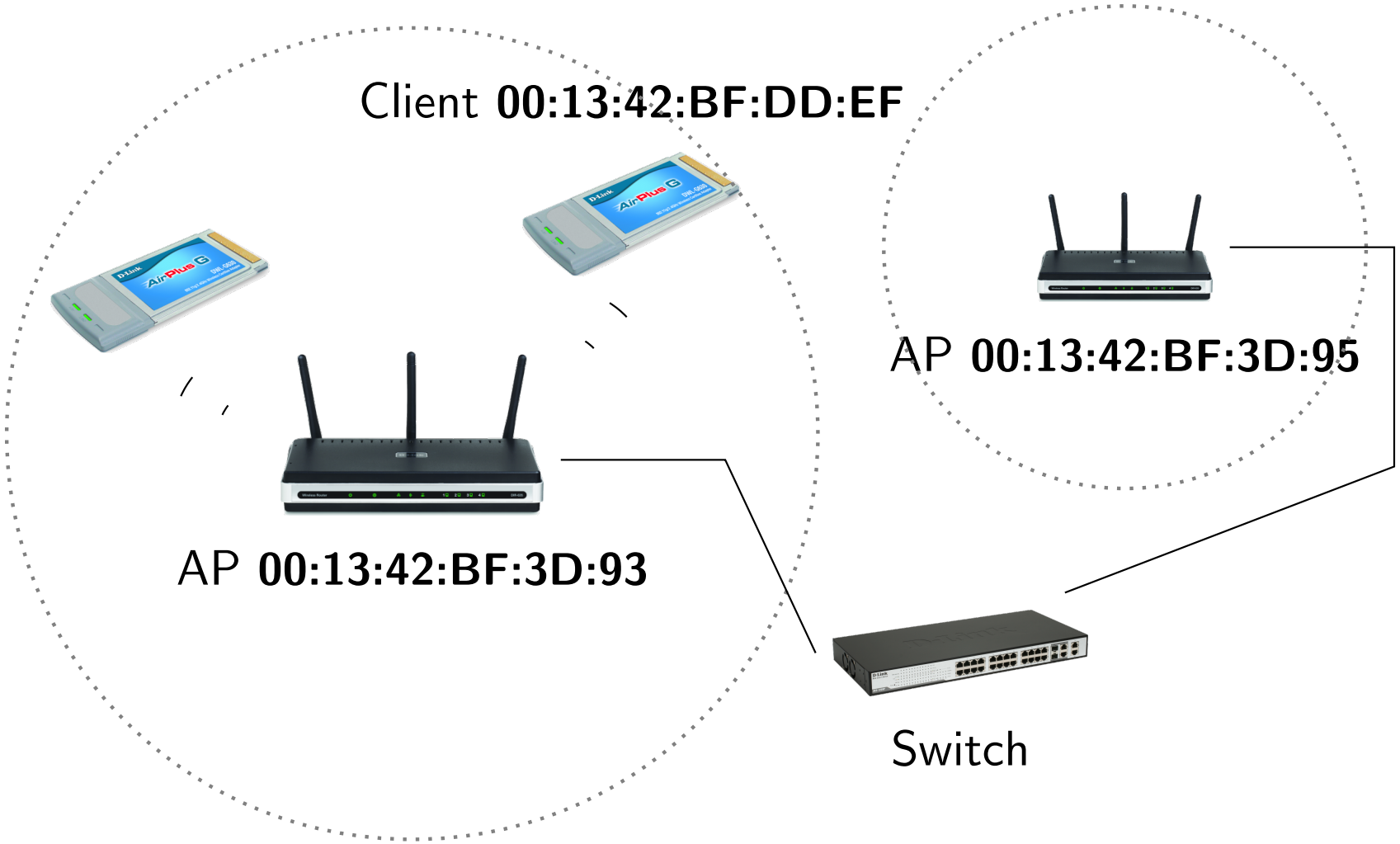
Breaking 104 Bit WEP in less than 60 seconds

Erik Tews Ralf-Philipp Weinmann Andrei Pyshkin
<e_tews,weinmann,pyshkin@cdc.informatik.tu-darmstadt.de>

TU-Darmstadt

25.11.2007

802.11 Network

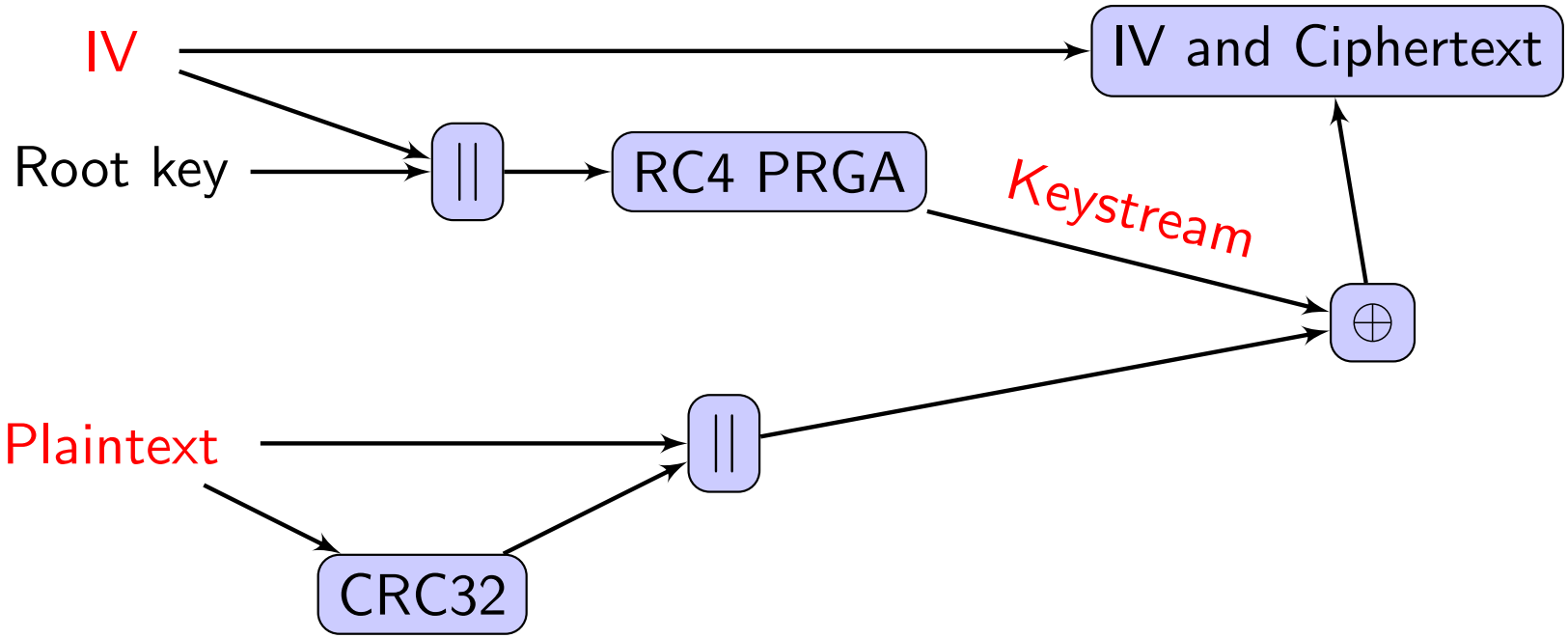


WEP

IEEE 802.11 (1999) pages 61-62:

> *Eavesdropping is a familiar problem to users of other types of wireless technology. IEEE 802.11 specifies a wired LAN equivalent data confidentiality algorithm. Wired equivalent privacy is defined as protecting authorized users of a wireless LAN from casual eavesdropping. This service is intended to provide functionality for the wireless LAN equivalent to that provided by the physical security attributes inherent to a wired medium. . . . The WEP algorithm has the following properties: It is reasonably strong: The security afforded by the algorithm relies on the difficulty of discovering the secret key through a brute-force attack. This in turn is related to the length of the secret key and the frequency of changing keys. WEP allows for the changing of the key (k) and frequent changing of the IV.*

How does WEP work?



So, what do we need to send a single packet?

An examnple

- Per packet key generated by concatenating root key and iv: $K = IV||Rk$
- IV is always transmitted in plaintext.
- root key Rk = *CA FE BA BE 42*, possible per packet keys:

```
48 AC 64 CA FE BA BE 42
27 55 06 CA FE BA BE 42
BC 56 C6 CA FE BA BE 42
```

- The attacker always knows the first 3 bytes

History of WEP

1996 Some first drafts of WEP appeared.

1997 Release of the first final version of IEEE 802.11.

2001 WEP broken by Fluhrer, Mantin, and Shamir, 9,000,000 encrypted packets for 50% success probability.

2004 WEP broken again, by KoreK, 700,000 packets for 50% success probability.

2005 WEP broken again, by KoreK again (chopchop attack), $128 \cdot \mathrm{length}(P)$ packets to decrypt a single packet P.

2005 WEP broken again, by Bittau, fragmentation attack:

- 34 packets for a single full-length keystream
- Decryption of single packes as fast as chopchop
- Realtime decryption of traffic using internet redirection

2007 WEP broken again, by Pyshkin, Tews, Weinmann, with the help of Klein, 35,000 packets for 50% success probability.

The chopchop attack

After you have captured a single packet P.

- Can decrypt the last i bytes of P, with $128 \cdot i$ packets send.
- That gives you the last i bytes of the keystream used to decrypt P too.
- Can not recover the secret key of the network.
- The reverse version of chopchop can give you the next i bytes of the keystream used to encrypt P, with $128 \cdot i$ packets send.
- Works (sometimes) even without associating to the access point.
- For a short packet (70 bytes), and a good connection (800 packets per second), these are only 12 seconds, at a moderate rate (300 packets per second) 30 seconds.

How chopchop works: (nonmathematical version)

We know:

- Every packet is integrity-protected using a CRC32 checksum.
- Selectively flipping bits of the checksum inside an encrypted packet is possible.

So, how can we use this:

- If we shorten a packet by one byte, the checksum becomes invalid.
- If we would know the cleartext of the last byte, we would also know which bits inside the checksum to flip.
- Now we can do the following:
  - Guess that the last byte of cleartext was 0
  - Correct the checksum by flipping the correct bits
  - Send the modified to the accesspoint
  - If it is relayed, the guess was correct
  - If it is not relayed, we guess the last byte to be 1 and start again.

How to correct the checksum: (simplified version)

Every packet has a CRC32 checksum. A receiver takes the decrypted plaintext as a polynominal over $\mathbb{F}_2$ and checks if:

$$P \bmod P_{CRC} = P_{ONE}$$

with:

$$\begin{aligned} P_{ONE} &= X^{31} + X^{30} + X^{29} + X^{28} + X^{27} + X^{26} + X^{25} + X^{24} + \\ &\quad X^{23} + X^{22} + X^{21} + X^{20} + X^{19} + X^{18} + X^{17} + X^{16} + X^{15} + \\ &\quad X^{14} + X^{13} + X^{12} + X^{11} + X^{10} + X^{9} + X^{8} + X^{7} + X^{6} + \\ &\quad X^{5} + X^{4} + X^{3} + X^{2} + X^{1} + 1 \\ P_{CRC} &= X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + \\ &\quad X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1 \end{aligned}$$

How to correct the checksum: (simplified version)

- Let P be the plaintext of the packet and $P = Q \cdot X^8 + P_7$, with Q being the one byte shortened version of P.

$$\begin{aligned} P &= Q \cdot X^8 + P_7 = P_{ONE} \bmod P_{CRC} \\ Q \cdot X^8 &= P_7 + P_{ONE} \bmod P_{CRC} \\ Q &= X^{-8} \cdot (P_7 + P_{ONE}) \bmod P_{CRC} \\ Q + (X^{-8} \cdot (P_7 + P_{ONE})) + P_{ONE} &= P_{ONE} \bmod P_{CRC} \end{aligned}$$

- The correction $(X^{-8} \cdot (P_7 + P_{ONE})) + P_{ONE}$ can be applied to the encrypted packet and the shortened packet will be correct again if P_7 was guessed correctly.

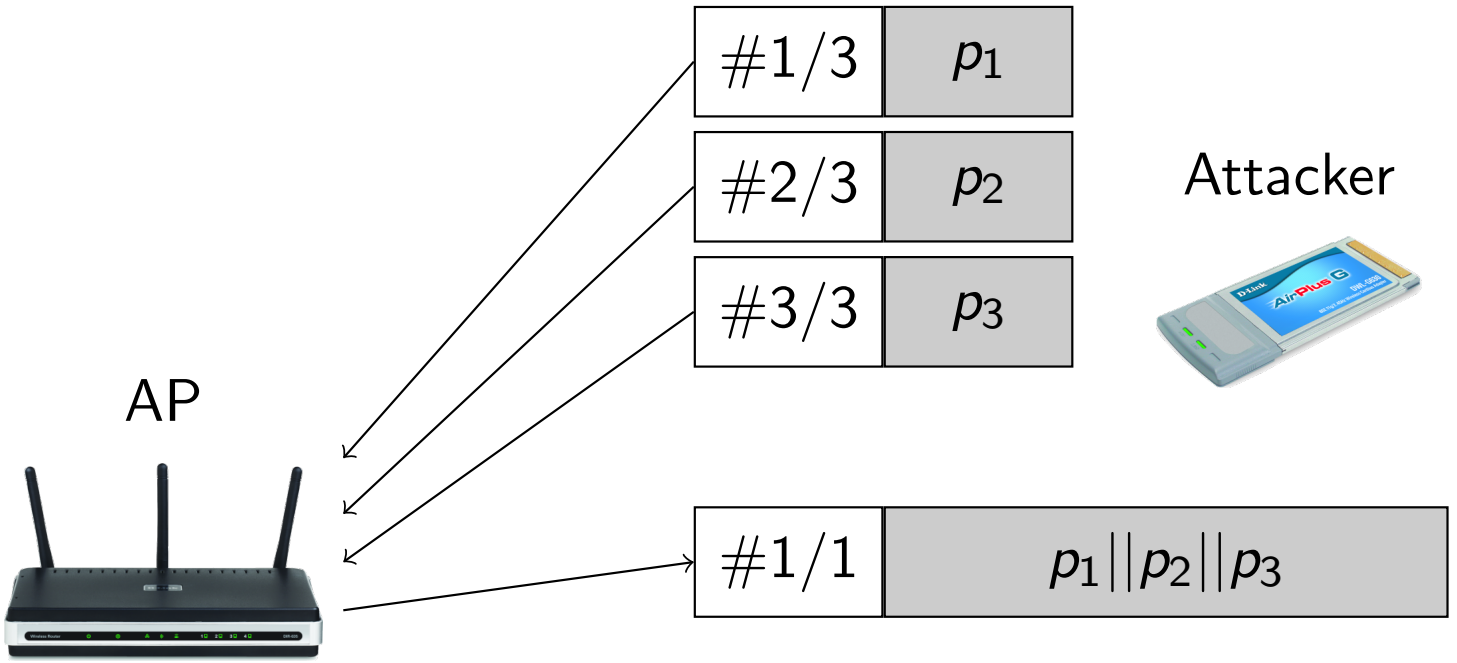
Fragmentation attack (by Bittau and friends)

The 802.11 protocol has build-in fragmentation support:

- Packets are first fragmented, then encrypted
- Each fragment is encrypted totally independant of each other
- Fragmentation information is not integrity protected in any way

Well, how can we use it...

Discovering of keystreams



Real time decryption

We can decrypt WEP-protected traffic in real-time, if the network is connected to the internet:

- Routing information in IP-packets are located at the beginning of the packet.
- Assume we have received a packet P, and we wan't to decrypt it.
- First, we generate the beginning of a packet P' with IP-Header with a destination address under our control (for examnple *130.83.167.48* aka. www.cdc.informatik.tu-darmstadt.de).
- P' is encrypted and sent as fragment 1/2.
- And P is sent as fragment 2/2.
- The accesspoint will reassemble both fragments and send $P'||P$ to *130.83.167.48*.
- No math involved at all.

Summary: Fragmentation

- Incredibly fast, for discovering keystreams (about 1 second)
- Some vendors got buggy fragmentation implementations.
- Some other tricks are possible too.

A history of the PTW attack:

- 2004 KoreK showed his attack on WEP, which was able to recover a 104 bit key with probability 50% with about 700,000 packets sniffed of a network.
- 2005 Andreas Klein gave a first talk about a new analysis of the RC4 stream cipher.
- Feb. 2007 We started looking at the Klein analysis and tried to find out if this can be used in WEP networks.
- Apr. 2007 We finished our analysis and published aircrack-ptw, a new WEP cracking tool based on an advanced version of the *Klein attack*. Now only 35,000 - 40,000 packets are needed for 50% success probabiltiy.

The RC4 stream cipher

- Invented in 1987 by Ron Rivest (also known as Ron's Code 4), kept as a trade secret.
- 1994 sourcecode of RC4 appeared on Usenet.
- Very efficient, just uses 8 bit operations and random memory access.
- Arbitrary keylengths from 8 to 2048 bit in 8 bit steps.
- Used in a lot of protocols and applicatoins.

RC4 Sourcecode

RC4-KSA

```
1   for i ← 0 to 255 do
2     S[i] ← i
3   end
4   j ← 0
5   for i ← 0 to 255 do
6     j ← j+S[i]+K[i mod len(K)] mod 256
7     swap(S, i, j)
8   end
9   i ← 0
10  j ← 0
```

RC4-PRGA

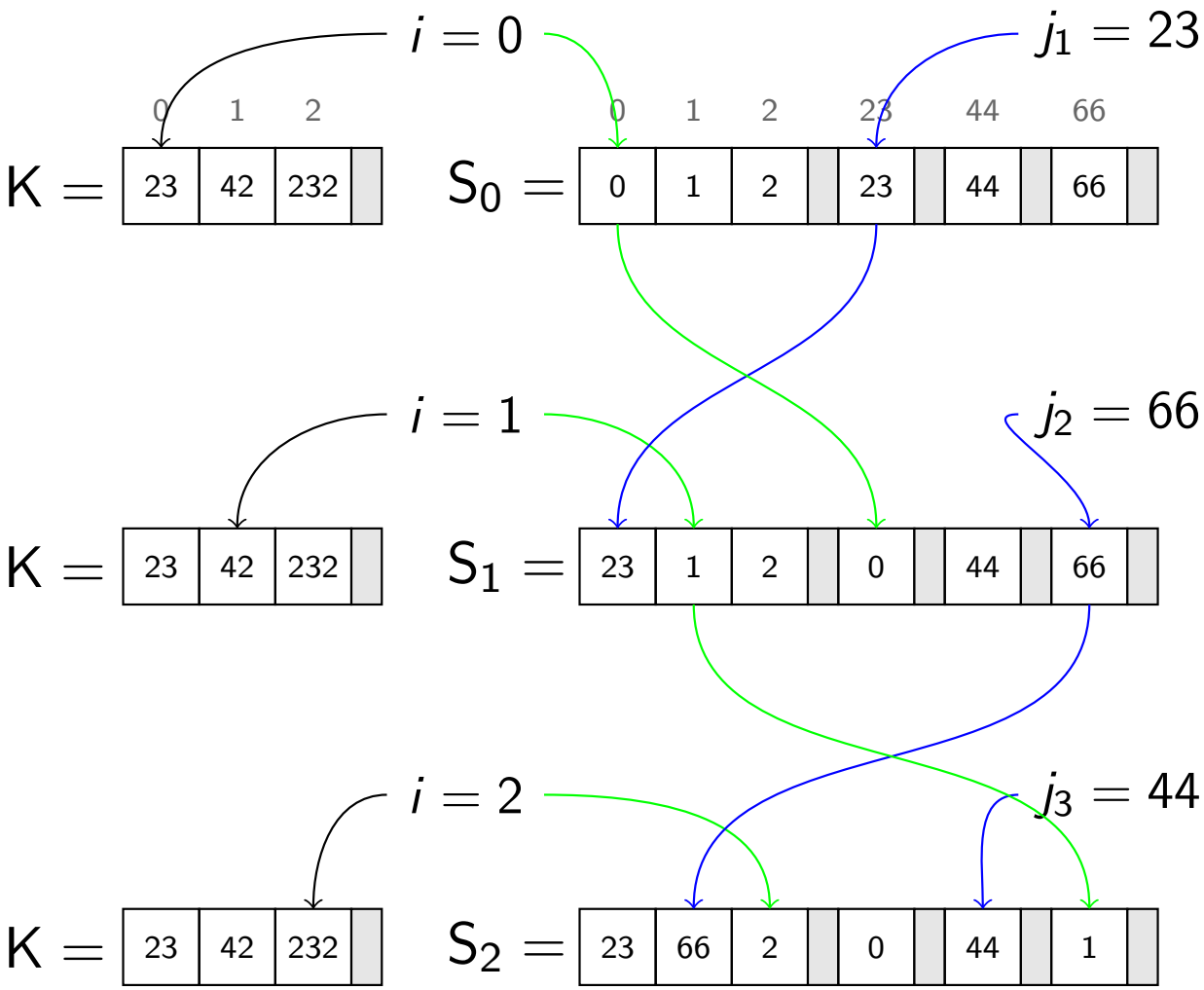
```
1   i ← i + 1 mod 256
2   j ← j + S[i] mod 256
3   swap(S, i, j)
4   return S[ S[i] + S[j] mod 256 ]
```

Notation

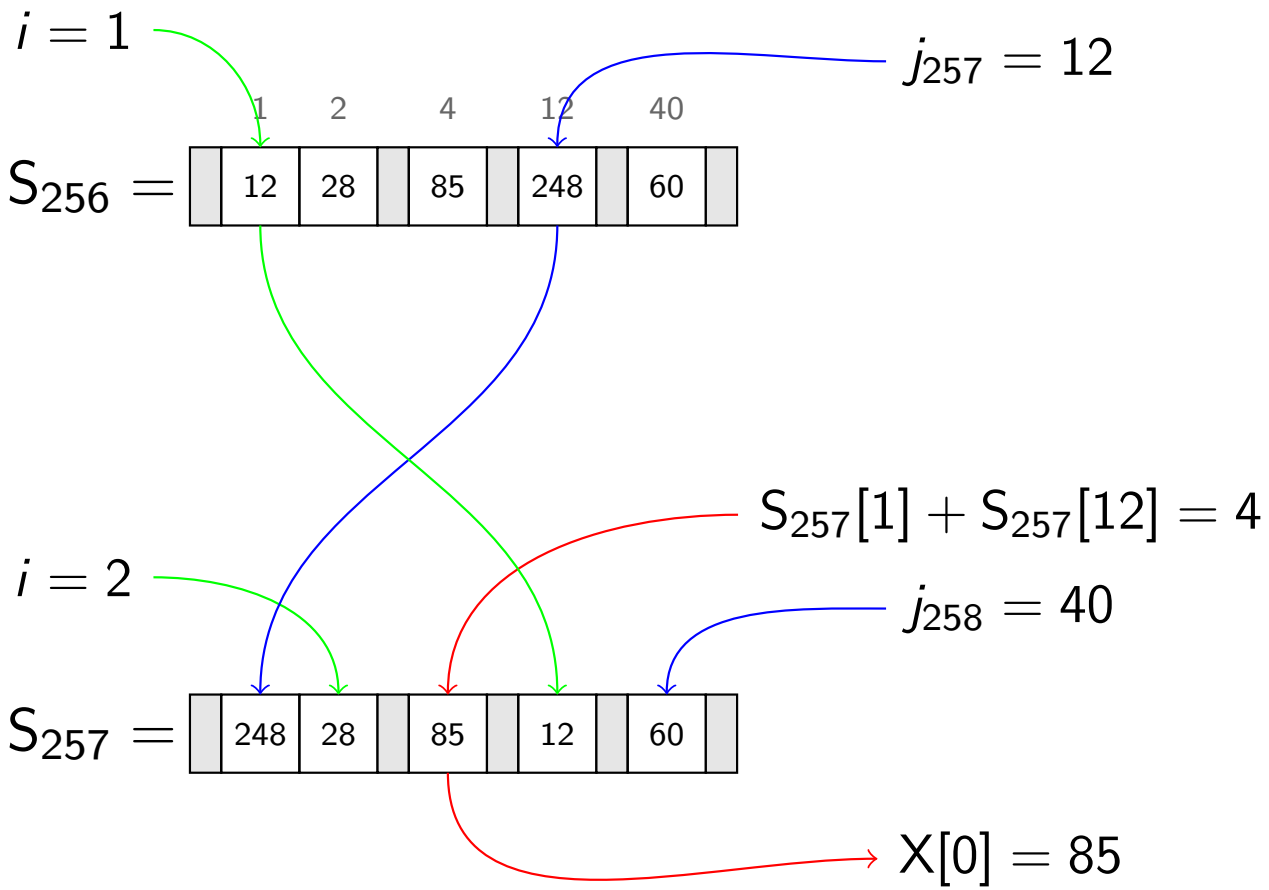
We will use the following notation:

- S_k is the state of S, after exactly k swaps have been executed on S.
- j_k is the state of j, after k swaps on S.
- $l = S^{-1}[k] \Longleftrightarrow k = S[l]$

An example RC4-KSA K = 23, 42, 232, 11



An example RC4-PRGA K = 23, 42, 232, 11



We know

- First 3 bytes of the per packet key are always known to the attacker.
- Attacker therefore knows j_0, j_1, j_2, j_3, S_0, S_1, S_2, S_3, and the value of i after 3 swaps.
- Attacker sometimes knows a lof keystreambytes.

An attacker can for example compute the following value from an intercepted packet:

$$f_3 : S_3^{-1}[3 - X[2] \bmod 256] - (S_3[3] + j_3) \bmod 256$$

Klein's analysis

$$f_3 : S_3^{-1}[3 - X[2] \bmod 256] - (S_3[3] + j_3) \bmod 256$$

Let $X = X[0]||X[1]||...$ be a keystream generated by RC4 using the key $K = K[0]||K[1]||....$

Theorem

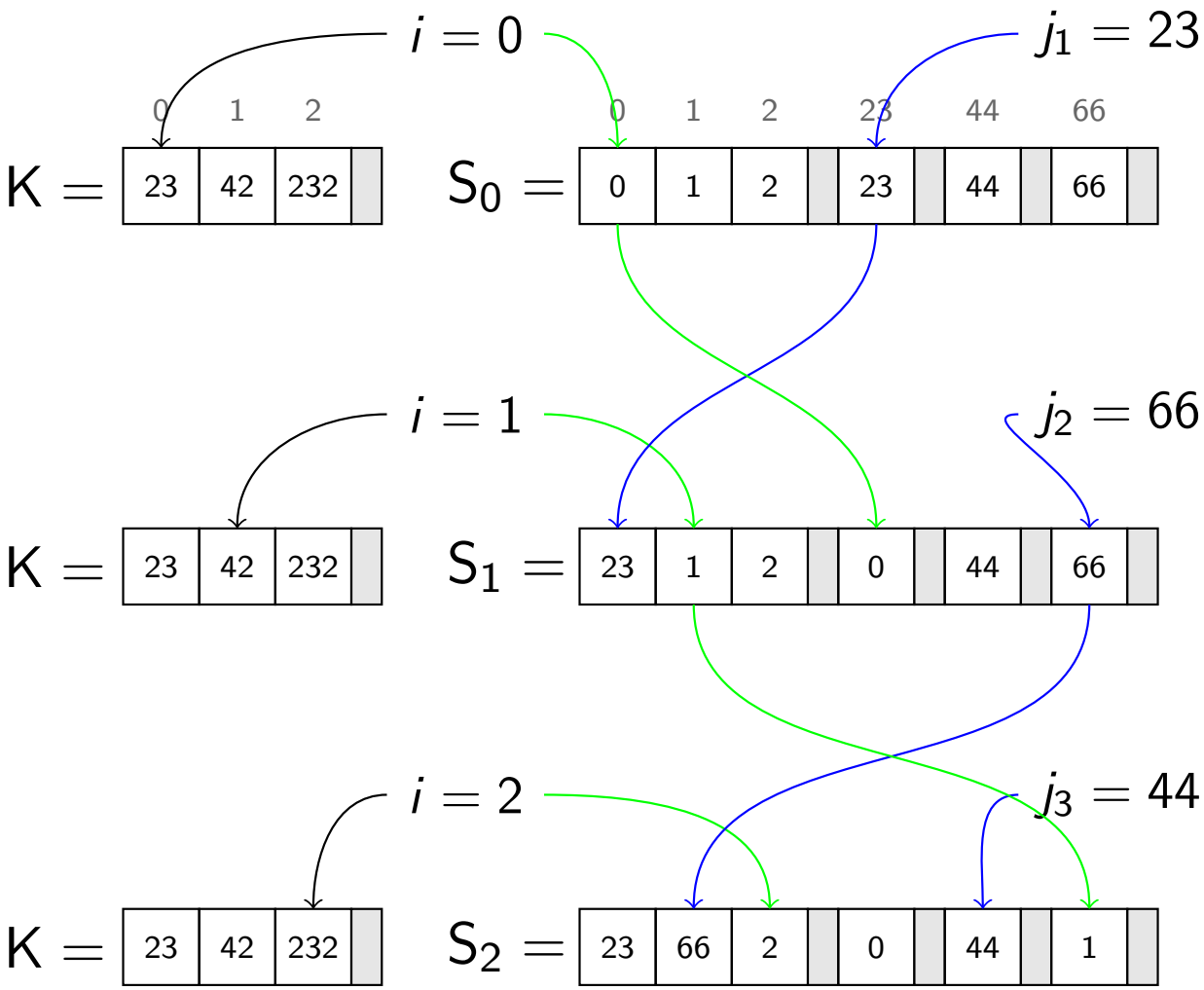
The function $f_3 : (\mathbb{Z}/256\mathbb{Z})^4 \rightarrow \mathbb{Z}/256\mathbb{Z}$ *has the following properties:*

- $\text{Prob}(f_3(K[0], K[1], K[2], X[2]) = K[3]) \approx \frac{1.36}{256}$
- *and for every* $a \neq K[3]$*:*
  $\text{Prob}(f_3(K[0], K[1], K[2], X[2]) = a) < \frac{1}{256}$

And f_3 *can be computed efficiently.*

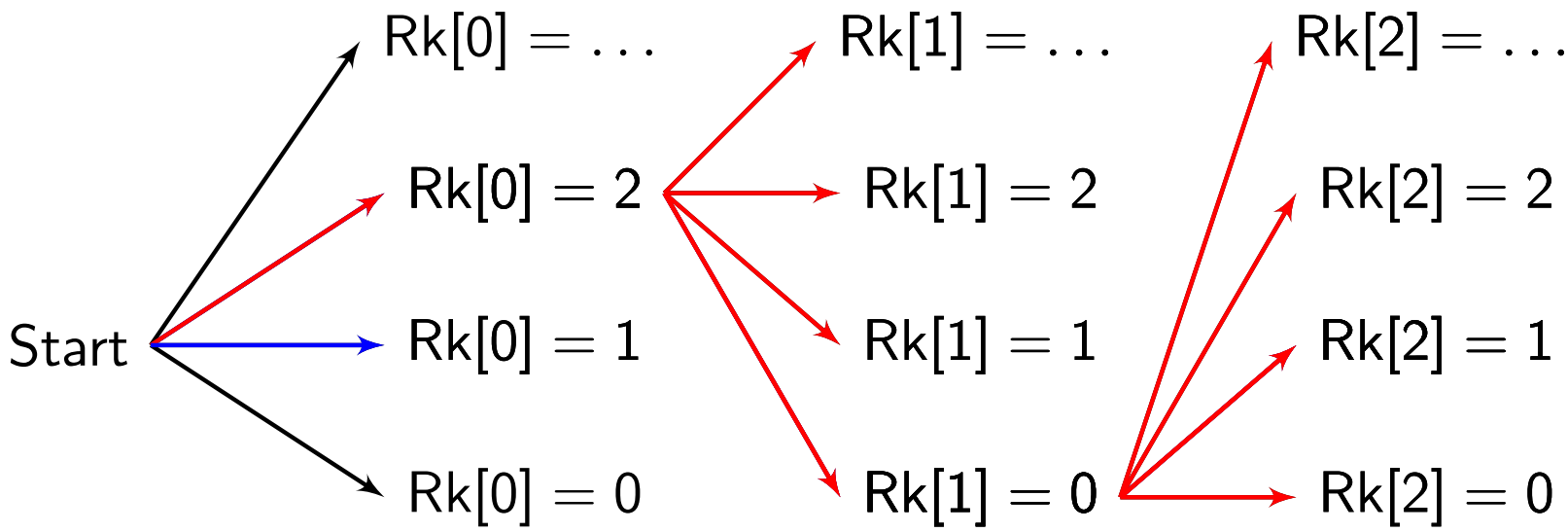
Similar functions can be found to determine keybyte $i + 1$ if the first i keybytes are known.

Theoretical background (Klein)



Analysis of the Klein attack

It is basically a decision tree



The PTW attack

$$f_4 : S_3^{-1}[4 - X[3] \bmod 256] - (S_3[4] + S_3[3] + j_3) \bmod 256$$

Theorem

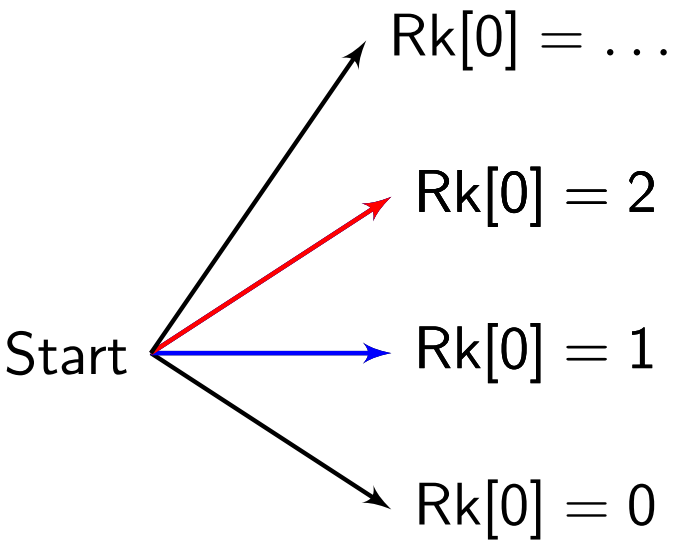
The function $f_4 : (\mathbb{Z}/256\mathbb{Z})^4 \rightarrow \mathbb{Z}/256\mathbb{Z}$ *has the following properties:*

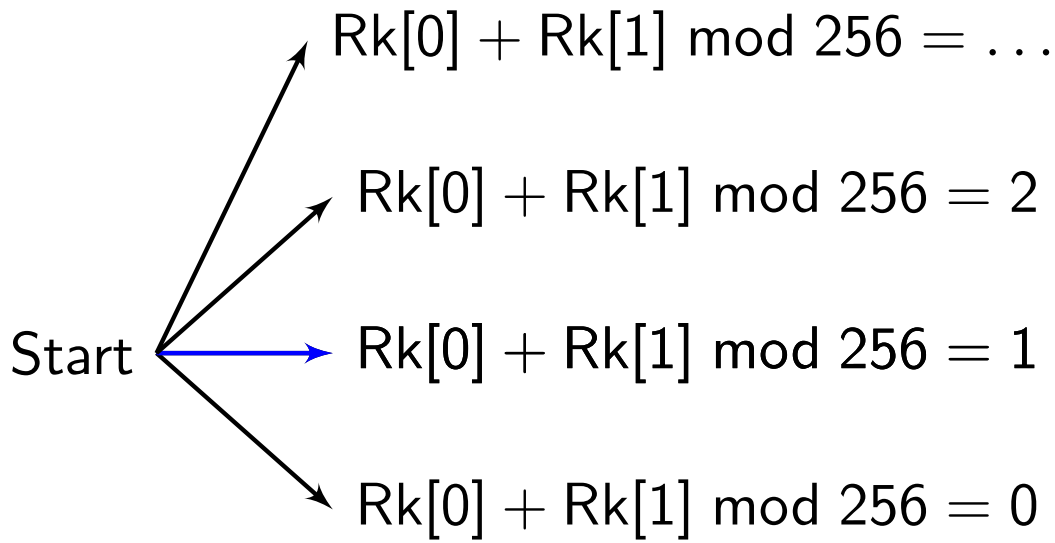
- $\text{Prob}(f_4(K[0], K[1], K[2], X[3]) = K[3] + K[4] \bmod 256) \approx \frac{1.34}{256}$
- *and for every* $a \neq K[3] + K[4] \bmod 256$*:*
  $\text{Prob}(f_4(K[0], K[1], K[2], X[3]) = a) < \frac{1}{256}$

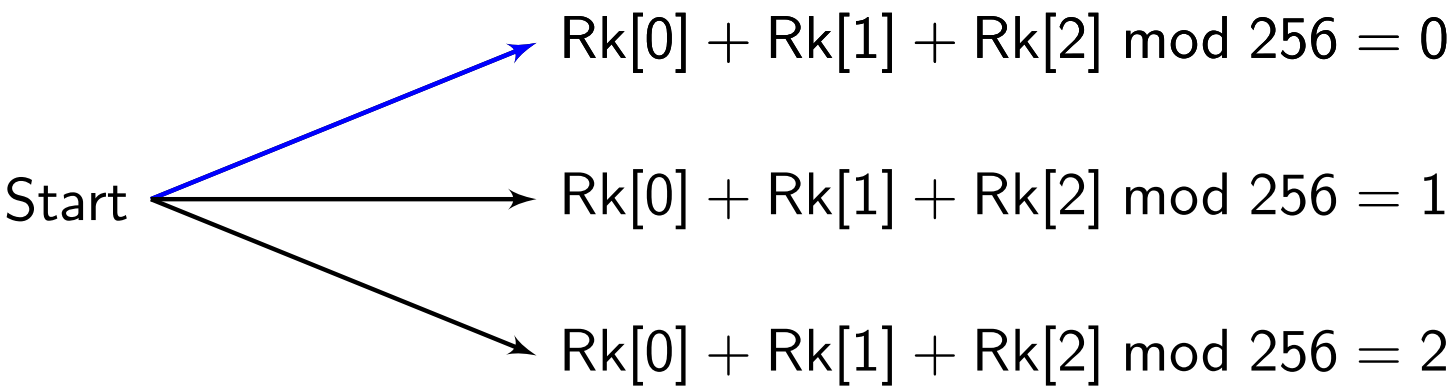
And f_4 *can be computed efficiently.*

Similar functions can be found to determine the sum of the keybytes $i + 1$ to $i + l$, if the first i keybytes are known.

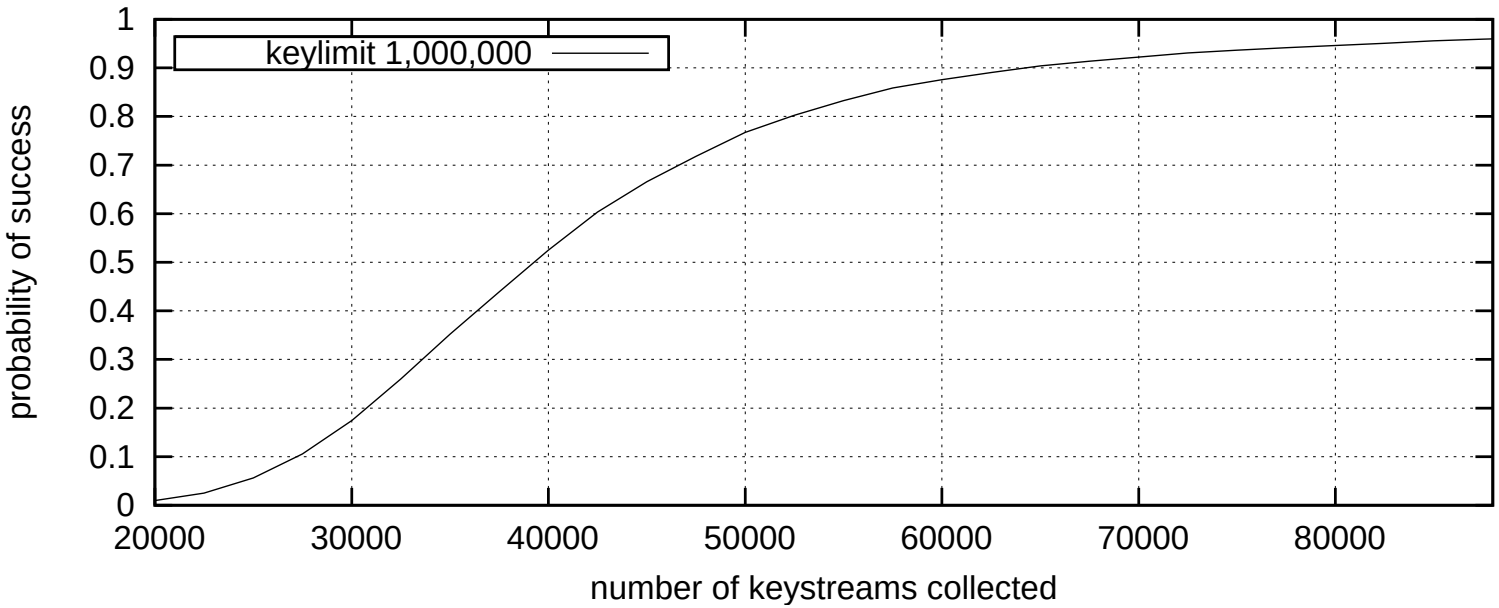
Analysis of the PTW attack







Some more numbers



- If you got 40,000 ARP packets, you can recover the secret key with a probability of about 50%
- With 85,000 ARP packets, you can recover the secret key with a probability of 95%
- With a good connection, you can get 40,000 packets in less that 60 seconds

A passive version

- Until now, we assumed that an attacker can recover as much keystream as needed.
  - This is true for an active attack, where an attacker uses arp-injection.
  - The PTW attack can be applied in a passive way too, just by listening to encrypted IPv4 traffic.
- It is efficient enough for a PDA or mobile phone.
- If the key is restricted to a special charset, it is even faster.

What happened after april 2007?

After we went public, some more people published their ideas which improve our attack.

Many Keystream Bytes of RC4 Leak Secret Key Information written by *Subhamoy Maitra and Goutam Pau*. They did an independant analysis of the Klein attack and had similar ideas to ours. Additionally, they found another correlation in RC4 which could help to improve our attack.

Passive-only Key Recovery Attacks on RC4 written by *Serge Vaudenay and Martin Vuagnoux*. They wrote a very similar attack to our attack, which uses some additional information from the following keystream bytes.

A Study on the Tews, Weinmann, Pyshkin Attack against WEP written by *Yuko Ozasa and Yoshiaki Fujikawa and Toshihiro Ohigashi and Hidenori Kuwakado and Masakatu Morii*. They did an analysis of our attack and found some similar improvements. Perhaps the number of needed packets for any success rate can be cut into half, in the next months.

Implementation is available in the 1.0-dev version of *aircrack-ng* at:

http://www.aircrack-ng.org/

Still questions?

Erik Tews
aircrack-ptw@cdc.informatik.tu-darmstadt.de