Cryptanalysis of ORYX

David Wagner[1]
Leonie Simpson[2]
Ed Dawson[2]
John Kelsey[3]
William Millan[2]
Bruce Schneier[3]

[1]University of California, Berkeley

[2]Information Security Research Centre
Queensland University of Technology

[3]Counterpane Systems
Minneapolis

August 1998

Outline of Presentation

1. Introduction
2. ORYX
3. Known-Plaintext Attack
4. Implementation Issues
5. Results
6. Ciphertext-only Attack
7. Summary and Conclusions

1.1 Background

- Increasing usage of mobile communications
- Cellular communications sent over a radio link - easy to eavesdrop
- Cryptography used for security
  - to protect privacy
  - to prevent fraud

1.2 Mobile Communications Devices

- First Generation - Analog
  - use of encryption - rare
  - level of security offered - low
- Second Generation - Digital
  - use of encryption - common
  - level of security offered - ???

1.3 Security of Current Encryption Standards in Mobile Communication Devices

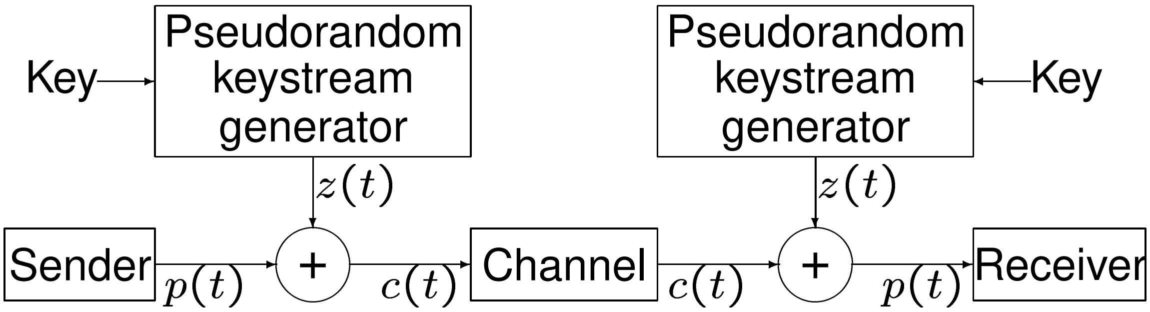
- **Europe - Global Systems Mobile (GSM)**
  - voice privacy algorithm - A5 cipher
    known-plaintext attack: Golić, Eurocrypt'97
  - authentication algorithm - COMP128
    chosen challenge attack: Briceno, Goldberg and Wagner, 1998

- **North America - Telecommunications Industry Association (TIA) Standards**
  - voice privacy algorithm in TDMA systems
    ciphertext-only attack
  - CMEA - block cipher for encrypting control channel
    known-plaintext attack: Wagner, Schneier and Kelsey, CRYPTO'97

2 The ORYX cipher

- stream cipher proposed for protecting cellular data transmissions in North American digital cellular systems

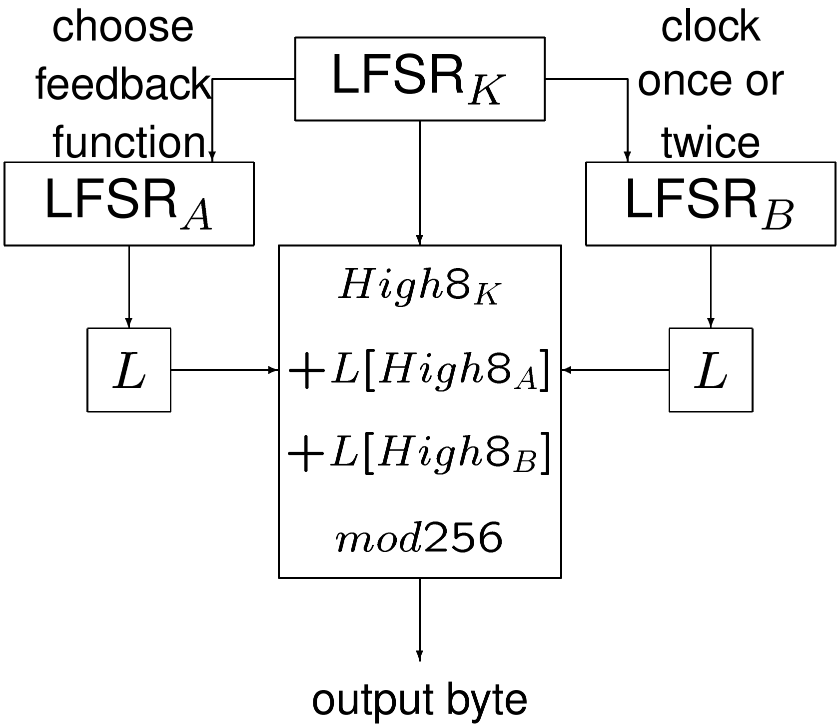
- used as a keystream generator



Encryption and decryption in binary additive stream ciphers.

$$
\begin{aligned}
c(t) &= p(t) \oplus z(t) \quad (1)\\
p(t) &= c(t) \oplus z(t) \quad (2)\\
z(t) &= p(t) \oplus c(t) \quad (3)
\end{aligned}
$$

2.1 ORYX Keystream Generator



- four components:
  - three 32-bit binary linear feedback shift registers (LFSR's)
  - an S-box, contains known permutation, L, of the integers 0-255

2.2 ORYX Keystream Generation

To produce a keystream byte,

1. LFSR$_K$ is stepped once

2. LFSR$_A$ is stepped once
   with one of two different feedback polynomials
   depending on the content of a stage of LFSR$_K$

3. LFSR$_B$ is stepped either once or twice
   depending on the content of a stage of LFSR$_K$

4. The high bytes of LFSR$_K$, LFSR$_A$ and LFSR$_B$
   combined to form keystream byte using the
   combining function:
   $\{High8_K + L[High8_A] + L[High8_B]\} mod 256$

3 Known Plaintext Attack

- Objective: to reconstruct the LFSR initial states
- Assume known:
  - LFSR feedback polynomials
  - Permutation L
  - segment of keystream $Z = \{Z(i)\}_{i=1}^{N}$
- Unknown: initial states of the 3×32-bit LFSRs
- Approach: divide and conquer
  - divide 96 bit initial state into smaller parts
  - guess contents of smaller part
  - check whether guess is correct

3.1 Features of ORYX used in attack

- for each output byte produced, LFSR_A and LFSR_K have been stepped once, and LFSR_B has been stepped once or twice

- output byte at time i is formed from contents of high 8 stages of all three component LFSRs

- stages of LFSR_K which control LFSR_A and LFSR_B are within the high 8 stages of LFSR_K

- divide the 96-bit keyspace and focus on these 24 stages

3.2 Divide and Conquer attack on ORYX

The attack is performed in three stages:

1. Guess contents of high 8 stages of the LFSRs at the time the first keystream byte, $Z(1)$, is produced. Denote this guess $High8(1)$

2. Construct a path of guesses, $High8(i)$, consistent with known keystream, $\{Z(i)\}_{i=2}^{N}$, - an iterative procedure

3. Reconstruct the initial states of the three 32-bit LFSRs

3.2.1 Attack Stage 1 - Guess $High8(1)$

- Guess $High8_A(1)$ and $High8_B(1)$

- Use $Z(1)$, L and the combining function to calculate corresponding contents of $High8_K(1)$

- Our initial 16-bit guess and the first byte of keystream gives 24 bits - a candidate for $High8(1)$

3.2.2 Attack Stage 2 - Construct a path of guesses

In the ith iteration:

- Use candidate $High8(i)$ to form set of predictions
  - $High8_K(i+1)$ and $High8_A(i+1)$ have one unknown bit shifting into them
  - $High8_B(i+1)$ has one or two unknown bits shifting into it
  - 12 possible combinations of input bits
- Evaluate candidate $High8(i)$ by comparing predicted output bytes with known $Z(i+1)$
- Decision rule: If $Z(i+1)$ is the same as
  - one of 12 distinct predictions, then a single set of values exists for $High8(i+1)$

- the predicted output byte for more than one combination of new input bits, then multiple sets of values exist for $High8(i+1)$
  i.e. the path of guesses branches

- no predicted output byte, then guessed contents of $High8(i)$ were obviously incorrect. Go back to last branching point and trace out another path.

• If no path of consistent guesses the length of the known keystream segment can be found, guessed contents of $High8_A(i)$ and $High8_B(i)$ were incorrect - make a new 16-bit guess and repeat the procedure

3.2.3 Attack Stage 3 - Reconstruct LFSR initial states

For a sufficiently long path of consistent guesses,

- Assume the values for $High8_A(1)$, $High8_B(1)$ and $High8_K(1)$ were correct - this gives 8 bits in each LFSR state at the time the first keystream byte is produced

- Each of the consecutive guesses $High8(i)$ for $2 \leq i \leq 25$ gives another bit in the states of LFSR$_K$ and LFSR$_A$ and another one or two bits in the state of LFSR$_B$

- Step back each of the LFSRs to recover the initial states

4 Implementation Issues

Attack depends on identifying wrong guesses.

- Reject $High8(i)$ if $Z(i+1)$ is not the same as any predicted output

- Forming output bytes: 24 bit input, 8 bit output
  2^{24} different inputs, 2^{8} different output bytes
  A particular output byte produced by 2^{16} input combinations

- $Pr($ wrong $High(i+1) \rightarrow Z(i+1)) = \frac{2^{16}-1}{2^{24}-1}$
  ≈ 0.0039

- $Pr($ wrong $High(i+1)$ detected$) \approx 0.9961$

- $Pr($ wrong $High(i)$ detected$) \approx (0.9961)^{12}$
  $= 0.9541$

5 Results

- Experiments performed for keystream lengths $N = 25, 26$ and 27 bytes

- For each keystream length, the attack was performed 1000 times, using psuedorandomly generated LFSR initial states

- Attack considered successful if reconstructed initial states were the same as those actually used

Keystream Length (bytes)	25	26	27
% Success	99.7	99.9	100.0

6 Ciphertext-only attacks:

Possible with some knowledge of the plaintext

- identify a probable plaintext string of at least 7 characters
- slide this along the ciphertext string hoping to find the correct position
- if aligned correctly, obtain a segment of keystream the length of the plaintext string
- Apply the known plaintext attack outlined on this keystream segment
  - every path of guesses ruled out by the end of $N = 7$ bytes of known text $\Rightarrow$ not a valid match
  - Otherwise, assume match is valid. This provides about 35 bits in the 96-bit key

- If three probable word matches can be found, we have over 100 bits of information on the key

- Solve linear equations over GF(2) to recover the 96-bit key

7.1 Summary

- Divide and conquer attack on ORYX is possible, regardless of key schedule
- Attack involves exhaustive search over a 16-bit subkey
- Success depends on obtaining a sufficient amount of keystream - 25 to 27 bytes
- potential for ciphertext-only attack if some plain-text statistics are known

7.2 Conclusion

- ORYX algorithm offers a low level of security
- further evidence of the low level of security offered in second generation mobile telephone devices
- low level of security may be due to adoption of algorithms for widespread use before the algorithms are subjected to public scrutiny
- What will the level of security be in third generation mobile communications devices?