Stream ciphers

Myrto Arapinis
School of Informatics
University of Edinburgh

October 6, 2016

Symmetric encryption schemes

A symmetric cipher consists of two algorithms

- encryption algorithm $E: \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- decryption algorithm $D: \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

st. $\forall k \in \mathcal{K}$, and $\forall m \in \mathcal{M}$, $D(k, E(k, m)) = m$

Kerckhoff's principle

- The encryption (E) and decryption (D) algorithms are public
- The security relies entirely on the secrecy of the key

Adversarie's capabilities - threat model

The attacker may have access to :

- some ciphertexts $c_1, \ldots, c_n$
- some plaintext/ciphertext pairs $(m_1, c_1), \ldots, (m_n, c_n)$ st. $c_i = E(k, m_i))$
- an encryption oracle - he can maybe trick a user to encrypt messages $m_1, \ldots, m_n$ of his choice
- a decryption oracle - he can maybe trick a user to decrypt ciphertexts $c_1, \ldots, c_n$ of his choice
- unlimited, or polynomial, or realistic ($\leq 2^{80}$) computational power

- A cryptographic scheme is secure under some assumptions, that is against a certain type of attacker
- A cryptographic scheme may be vulnerable to certain types of attacks but not others

What is a good encryption scheme?

An encryption scheme is secure against a given adversary, if this adversary cannot

- recover the secret key k
- recover the plaintext m underlying a ciphertext c
- recover any bits of the plaintext m underlying a ciphertext c
- . . .

The One-Time Pad (OTP)

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

$$
\begin{array}{cccccccccc}
k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\hline
c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

The One-Time Pad (OTP)

- $\mathcal{M} \ = \ \mathcal{C} \ = \ \mathcal{K} \ = \ \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

$$
\begin{array}{cccccccccc}
k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\hline
c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

- Decryption: $\forall k \in \mathcal{K}.\ \forall c \in \mathcal{C}.\ D(k,c) = k \oplus c$

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

$$\begin{array}{cccccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline \\ c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

- Decryption: $\forall k \in \mathcal{K}.\ \forall c \in \mathcal{C}.\ D(k,c) = k \oplus c$

$$\begin{array}{cccccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

$$\begin{array}{cccccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline \\ c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

- Decryption: $\forall k \in \mathcal{K}.\ \forall c \in \mathcal{C}.\ D(k,c) = k \oplus c$

$$\begin{array}{cccccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

- Consistency: $D(k, E(k,m)) = k \oplus (k \oplus m) = m$

Perfect secrecy

Definition

A cipher (E, D) over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$ satisfies perfect secrecy if for all messages $m_1, m_2 \in \mathcal{M}$ of same length ($|m_1| = |m_2|$), and for all ciphertexts $c \in \mathcal{C}$

$$|Pr(E(k, m_1) = c) - Pr(E(k, m_2) = c)| \leq \epsilon$$

where $k \xleftarrow{r} \mathcal{K}$ and ϵ is some "negligible quantity".

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

<u>Proof:</u> We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

$$Pr(E(k, m) = c)$$

where $k \xleftarrow{r} \mathcal{K}$.

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

Proof: We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

$$Pr(E(k, m) = c) \quad = \quad \frac{\#\{k \in \mathcal{K} :\ k \oplus m = c\}}{\#\mathcal{K}}$$

where $k \xleftarrow{r} \mathcal{K}$.

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

Proof: We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

$$\begin{aligned} Pr(E(k,m)=c) &= \frac{\#\{k \in \mathcal{K}:\ k \oplus m = c\}}{\#\mathcal{K}} \\ &= \frac{\#\{k \in \mathcal{K}:\ k = m \oplus c\}}{\#\mathcal{K}} \end{aligned}$$

where $k \xleftarrow{r} \mathcal{K}$.

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

Proof: We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

$$\begin{aligned} Pr(E(k,m)=c) &= \frac{\#\{k \in \mathcal{K}:\ k \oplus m = c\}}{\#\mathcal{K}} \\ &= \frac{\#\{k \in \mathcal{K}:\ k = m \oplus c\}}{\#\mathcal{K}} \\ &= \frac{1}{\#\mathcal{K}} \end{aligned}$$

where $k \xleftarrow{r} \mathcal{K}$.

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

Proof: We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

$$\begin{aligned} Pr(E(k,m)=c) &= \frac{\#\{k\in\mathcal{K}:\ k\oplus m=c\}}{\#\mathcal{K}} \\ &= \frac{\#\{k\in\mathcal{K}:\ k=m\oplus c\}}{\#\mathcal{K}} \\ &= \frac{1}{\#\mathcal{K}} \end{aligned}$$

where $k \stackrel{r}{\leftarrow} \mathcal{K}$.

Thus, for all messages $m_1, m_2 \in \mathcal{M}$, and for all ciphertexts $c \in \mathcal{C}$

$$|Pr(E(k,m_1)=c) - Pr(E(k,m_2)=c)| \leq$$

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

Proof: We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

$$\begin{aligned} Pr(E(k,m)=c) &= \frac{\#\{k \in \mathcal{K}:\ k \oplus m = c\}}{\#\mathcal{K}} \\ &= \frac{\#\{k \in \mathcal{K}:\ k = m \oplus c\}}{\#\mathcal{K}} \\ &= \frac{1}{\#\mathcal{K}} \end{aligned}$$

where $k \xleftarrow{r} \mathcal{K}$.

Thus, for all messages $m_1, m_2 \in \mathcal{M}$, and for all ciphertexts $c \in \mathcal{C}$

$$|Pr(E(k,m_1)=c) - Pr(E(k,m_2)=c)| \leq \left| \frac{1}{\#\mathcal{K}} - \frac{1}{\#\mathcal{K}} \right| = 0$$

Limitations of OTP

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext.

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext.
- Getting true randomness!
  - The key should not be guessable from an attacker.

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext.
- Getting true randomness!
  - The key should not be guessable from an attacker.
- Perfect secrecy does not capture all possible attacks

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext.
- Getting true randomness!
  - The key should not be guessable from an attacker.
- Perfect secrecy does not capture all possible attacks
  - OTP is subject to two-time pad attacks
    given $m_1 \oplus k$ and $m_2 \oplus k$, we can compute
    $m_1 \oplus m_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$
    English has enough redundancy s.t. $m_1 \oplus m_2 \rightarrow m_1, m_2$

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext.
- Getting true randomness!
  - The key should not be guessable from an attacker.
- Perfect secrecy does not capture all possible attacks
  - OTP is subject to two-time pad attacks
    given $m_1 \oplus k$ and $m_2 \oplus k$, we can compute
    $m_1 \oplus m_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$
    English has enough redundancy s.t. $m_1 \oplus m_2 \to m_1, m_2$
  - OTP is malleable
    given the ciphertext $c = E(k, m)$ with $m = to\ bob : m_0$, it is possible to compute the ciphertext $c' = E(k, m')$ with
    $m' = to\ eve : m_0$
    $c' := c \oplus "to\ bob : 00 \ldots 00" \oplus "to\ eve : 00 \ldots 00"$

Stream ciphers

- Goal: make the OTP practical

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG)
    $G: \{0,1\}^s \rightarrow \{0,1\}^n$ with $s << n$

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG) $G : \{0,1\}^s \to \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k,m) = G(k) \oplus m$

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG) $G: \{0,1\}^s \rightarrow \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k,m) = G(k) \oplus m$
- Decryption using a PRG G: $D(k,c) = G(k) \oplus c$

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG) $G: \{0,1\}^s \to \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k,m) = G(k) \oplus m$
- Decryption using a PRG G: $D(k,c) = G(k) \oplus c$
- Stream ciphers are subject to two-time pad attacks

Stream ciphers

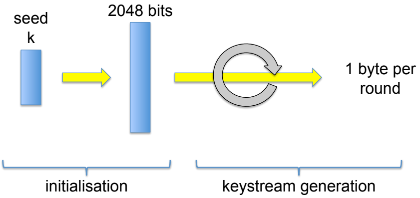
- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG) $G: \{0,1\}^s \rightarrow \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k,m) = G(k) \oplus m$
- Decryption using a PRG G: $D(k,c) = G(k) \oplus c$
- Stream ciphers are subject to two-time pad attacks
- Stream ciphers are malleable

RC4

- Stream cipher invented by Ron Rivest in 1987

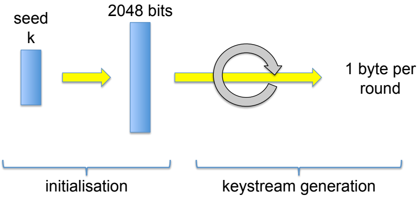
RC4

- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



RC4

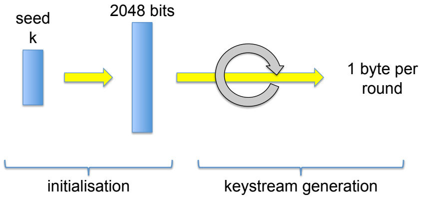
- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.

RC4

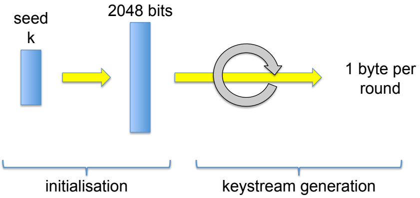
- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.
- Used in HTTPS and WEP

RC4

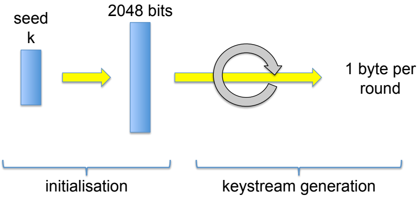
- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.
- Used in HTTPS and WEP
- Weaknesses of RC4:

RC4

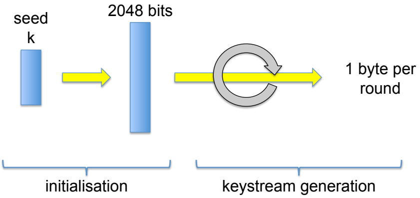
- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.
- Used in HTTPS and WEP
- Weaknesses of RC4:
  - first bytes are biased
    $\longrightarrow$ drop the first to 256 generated bytes

RC4

- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.
- Used in HTTPS and WEP
- Weaknesses of RC4:
  - first bytes are biased
    $\longrightarrow$ drop the first to 256 generated bytes
  - subject to related keys attacks
    $\longrightarrow$ choose randomly generated keys as seeds

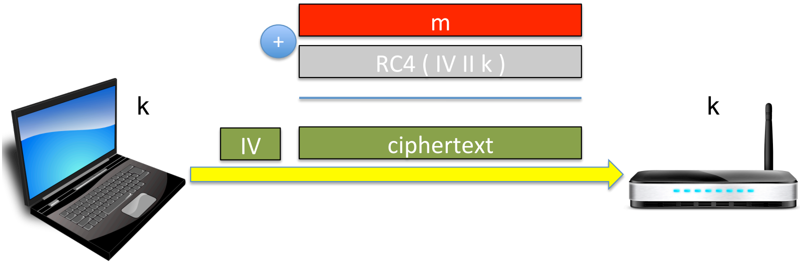
RC4: initialisation

```
for i := 0 to 255 do
    S[i] := i
end
j := 0
for i := 0 to 255 do
    j := (j + S[i] + K[i (mod |K|)]) (mod 256)
    swap(S[i], S[j])
end
i := 0
j := 0
```

RC4: key stream generation

```
while generatingOutput
    i := i + 1(mod 256)
    j := j + S[i](mod 256)
    swap(S[i], S[j])
    output(S[S[i] + S[j](mod 256)])
end
```

WEP uses RC4



Initialisation Vector (IV): 24-bits long string

Weaknesses of WEP

Weaknesses of WEP

- two-time pad attack: IV is 24 bits long, so the key is reused after at most 2^{24} frames
  $\longrightarrow$ use longer IVs

Weaknesses of WEP

- two-time pad attack: IV is 24 bits long, so the key is reused after at most 2^{24} frames
  $\longrightarrow$ use longer IVs

- Fluhrer, Mantin and Shamir (FMS) attack (related keys attack):
  - the keys only differ in the 24 bits IV
  - first bytes of key stream known because standard headers are always sent
  - for certain IVs knowing m bytes of key and keystream means you can deduce byte $m+1$ of key

  $\longrightarrow$ instead of using related IVs, generate IVs using a PRG

Weaknesses of WEP

- two-time pad attack: IV is 24 bits long, so the key is reused after at most 2^{24} frames
  $\longrightarrow$ use longer IVs
- Fluhrer, Mantin and Shamir (FMS) attack (related keys attack):
  - the keys only differ in the 24 bits IV
  - first bytes of key stream known because standard headers are always sent
  - for certain IVs knowing m bytes of key and keystream means you can deduce byte $m+1$ of key

  $\longrightarrow$ instead of using related IVs, generate IVs using a PRG

Weaknesses of TLS

MUST READ **THESE TEN CITIES ARE HOME TO THE BIGGEST BOTNETS**

RC4 NOMORE crypto exploit used to decrypt user cookies in mere hours

Websites using RC4 encryption need to change their protocols as exploits using design flaws are now far easier to perform.

By Charlie Osborne for Zero Day | July 20, 2015 -- 10:21 GMT (11:21 BST) | Topic: Security

1 | f 33 | in 96

RELATED STORIES

Security
Accenture acquires Defense Point Security to boost US federal defenses

Security
Facebook rolls out opt-in encryption for 'secret' Messenger chats

Security
Feds subpoena, gag encrypted chat firm Open Whisper Systems

Security
Insulin pump vulnerabilities could lead to overdose

Linear Feedback Shift Registers (LFSRs)

Linear Feedback Shift Registers (LFSRs)

- $\mathcal{K} = \{0, 1\}^s$

Linear Feedback Shift Registers (LFSRs)

- $\mathcal{K} = \{0, 1\}^s$
- Main data structure: register R of s bits

Linear Feedback Shift Registers (LFSRs)

- $\mathcal{K} = \{0,1\}^s$
- Main data structure: register R of s bits
- Initialisation: $R := k$

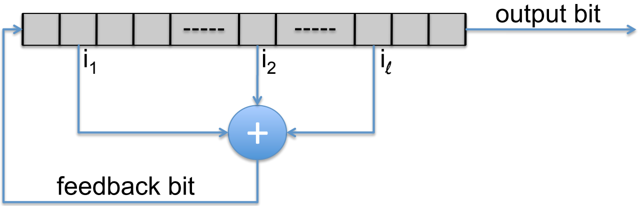
Linear Feedback Shift Registers (LFSRs)

- $\mathcal{K} = \{0,1\}^s$
- Main data structure: register R of s bits
- Initialisation: $R := k$
- Keystream generation: 1-bit output per round

  taps: $i_1, i_2, \ldots i_\ell$

  feedback bit: $R[i_1] \oplus R[i_2] \oplus \cdots \oplus R[i_\ell]$

  output bit: $R[s]$



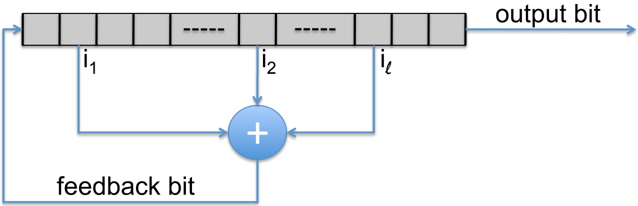
Linear Feedback Shift Registers (LFSRs)

- $\mathcal{K} = \{0,1\}^s$
- Main data structure: register R of s bits
- Initialisation: $R := k$
- Keystream generation: 1-bit output per round

  taps: $i_1, i_2, \ldots i_\ell$

  feedback bit: $R[i_1] \oplus R[i_2] \oplus \cdots \oplus R[i_\ell]$

  output bit: $R[s]$



- Broken LFSR-based stream ciphers:
  - DVD encryption: CSS (2 LFSRs)
  - GSM encryption: A5 (3 LFSRs)
  - Bluetooth encryption: E0 (4 LFSRs)

Content Scrambling System (CSS) uses LFSRs

Content Scrambling System (CSS) uses LFSRs

- $\mathcal{K} = \{0,1\}^{40}$

Content Scrambling System (CSS) uses LFSRs

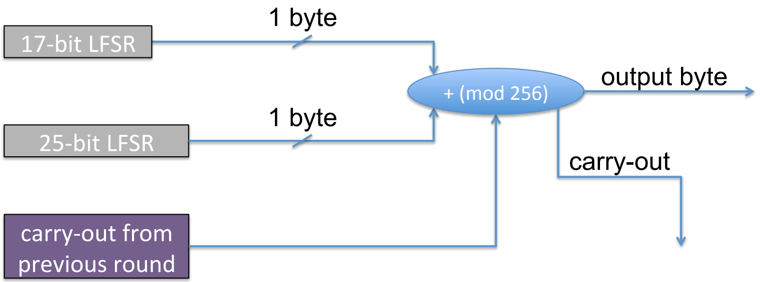
- $\mathcal{K} = \{0,1\}^{40}$
- Data structures: 17-bits LFSR (R_{17}) and 25-bits LFSR (R_{25})

Content Scrambling System (CSS) uses LFSRs

- $\mathcal{K} = \{0,1\}^{40}$
- Data structures: 17-bits LFSR (R_{17}) and 25-bits LFSR (R_{25})
- Initialisation: $R_{17} := 1||K[0-15]$
  $R_{25} := 1||K[16-39]$

Content Scrambling System (CSS) uses LFSRs

- $\mathcal{K} = \{0,1\}^{40}$
- Data structures: 17-bits LFSR (R_{17}) and 25-bits LFSR (R_{25})
- Initialisation: $R_{17} := 1||K[0-15]$
  $R_{25} := 1||K[16-39]$
- Keystream generation: 1-byte output per round



Weaknesses in CSS

Can be broken in time 2^{17}. The idea of the attack is as follows:

Weaknesses in CSS

Can be broken in time 2^{17}. The idea of the attack is as follows:

- Because of structure of MPEG-2, first 20 bytes of plaintext are known

Weaknesses in CSS

Can be broken in time 2^{17}. The idea of the attack is as follows:

- Because of structure of MPEG-2, first 20 bytes of plaintext are known
- Hence also first 20 bytes of keystream are known

Weaknesses in CSS

Can be broken in time 2^{17}. The idea of the attack is as follows:

- Because of structure of MPEG-2, first 20 bytes of plaintext are known
- Hence also first 20 bytes of keystream are known
- Given output of 17 bit LFSR, can deduce output of 25 bit LFSR by subtraction

Weaknesses in CSS

Can be broken in time 2^{17}. The idea of the attack is as follows:

- Because of structure of MPEG-2, first 20 bytes of plaintext are known
- Hence also first 20 bytes of keystream are known
- Given output of 17 bit LFSR, can deduce output of 25 bit LFSR by subtraction
- Hence try all 2^{17} possibilities for 17 bit LFSR and if generated 25 bit LFSR produces observed keystream, cipher is cracked

Android BitCoin attack

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS SIGN IN

RANDOM THEFT —

All Android-created Bitcoin wallets vulnerable to theft

Android Java SecureRandom function flaw undermines security of Android wallets.

LEE HUTCHINSON - 8/12/2013, 3:15 PM

Xurxo Martinez

Bitcoin.org released a security advisory over the weekend warning the Bitcoin community that any Bitcoin wallet generated on any Android device is insecure and open to theft. The insecurity appears to stem from a flaw in the Android Java SecureRandom class, which under certain circumstances can

Modern stream ciphers

Project eStream: project to "identify new stream ciphers suitable for widespread adoption", organised by the EU ECRYPT network
$\longrightarrow$ HC-128, Rabbit, Salsa20/12, SOSEMANUK,
Grain v1, MICKEY 2.0, Trivium

Modern stream ciphers

Project eStream: project to "identify new stream ciphers suitable for widespread adoption", organised by the EU ECRYPT network
$\longrightarrow$ HC-128, Rabbit, Salsa20/12, SOSEMANUK,
Grain v1, MICKEY 2.0, Trivium

Conjecture

These eStream stream ciphers are "secure"

Concluding remarks

Concluding remarks

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition

Concluding remarks

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition

Concluding remarks

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition
- The design of crypto primitives is subtle and error prone.
  $\longrightarrow$ use standardised publicly know primitives

Concluding remarks

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition
- The design of crypto primitives is subtle and error prone.
  $\longrightarrow$ use standardised publicly know primitives
- Crypto primitives are secure under a precisely defined threat model.
  $\longrightarrow$ respect the security assumptions of the crypto primitives

Concluding remarks

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition
- The design of crypto primitives is subtle and error prone.
  $\longrightarrow$ use standardised publicly know primitives
- Crypto primitives are secure under a precisely defined threat model.
  $\longrightarrow$ respect the security assumptions of the crypto primitives
- Many attacks dues to poor implementations of cryptography