Attacks on Stream Ciphers: A Perspective

Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute, Kolkata
India
palash@isical.ac.in

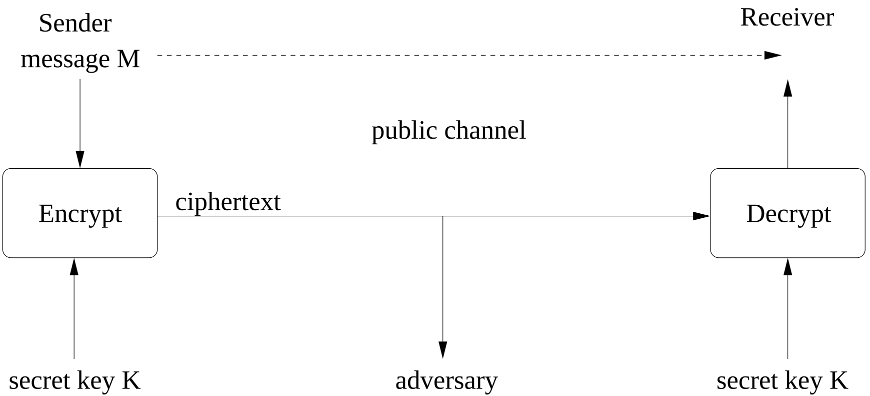
First Asian Workshop on Symmetric Key Cryptography – ASK 2011, 30th August 2011

Overview of the Talk

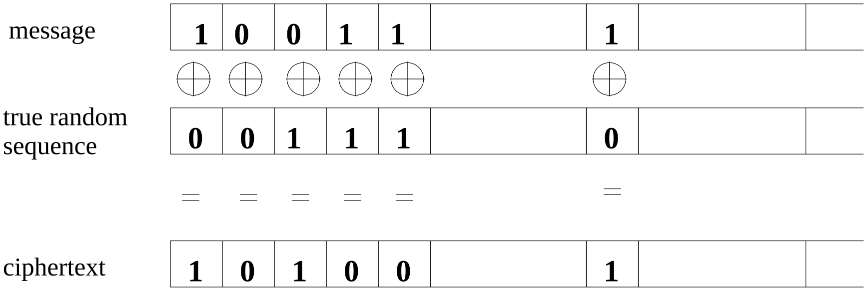
- Background.
- Correlation Attacks.
- Algebraic Attacks.
- Differential Attacks.
- Time/Memory Trade-Off Attacks.

Background.

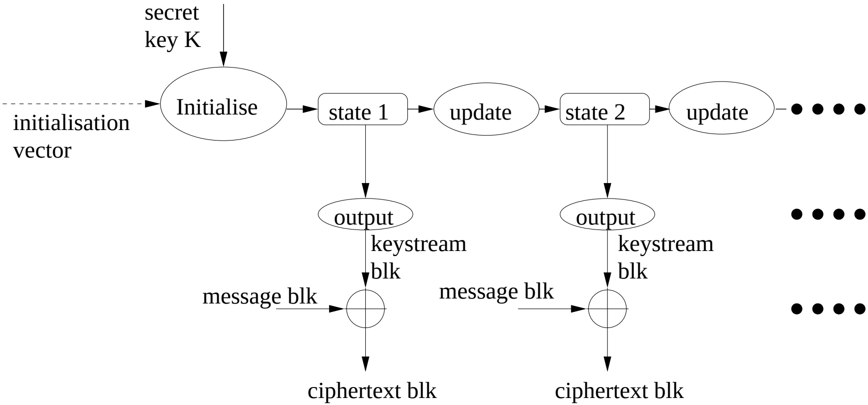
Model of Symmetric Key Encryption



One-Time Pad


message
1 0 0 1 1 1
true random sequence
0 0 1 1 1 0
ciphertext
1 0 1 0 0 1

Model of Additive Stream Cipher



- Key: k bits; IV: (usually) $\leq k$ bits; state: (usually) $\geq 2k$ bits;
- initialise, update, output: functions (deterministic algorithms);
- keystream blk, msg blk, cpr blk: $\geq$ 1 bit.

Self-Synchronizing Stream Cipher

message	m_0	m_1	m_2	$\cdots$	m_i	$\cdots$
keystream	k_0	k_1	k_2	$\cdots$	k_i	$\cdots$
ciphertext	c_0	c_1	c_2	$\cdots$	c_i	$\cdots$

$$c_i = m_i \oplus k_i.$$

Self-Synchronizing Stream Cipher

message	m_0	m_1	m_2	$\cdots$	m_i	$\cdots$
keystream	k_0	k_1	k_2	$\cdots$	k_i	$\cdots$
ciphertext	c_0	c_1	c_2	$\cdots$	c_i	$\cdots$

$$c_i = m_i \oplus k_i.$$

- k_i is completely determined by the secret key K *and* $c_{i-n}, \ldots, c_{i-1}$.
- Correctly receiving n ciphertext bits allow correct generation of the next keystream bit.
- Robust against channel errors: bit flip/drop/insert.

Self-Synchronizing Stream Cipher

message	m_0	m_1	m_2	$\cdots$	m_i	$\cdots$
keystream	k_0	k_1	k_2	$\cdots$	k_i	$\cdots$
ciphertext	c_0	c_1	c_2	$\cdots$	c_i	$\cdots$

$$c_i = m_i \oplus k_i.$$

- k_i is completely determined by the secret key K *and* $c_{i-n}, \ldots, c_{i-1}$.
- Correctly receiving n ciphertext bits allow correct generation of the next keystream bit.
- Robust against channel errors: bit flip/drop/insert.

More generally, m_i is completely determined by the secret key K and the last n ciphertext bits.

Attack Models: Adversarial Access

- **Ciphertext only attack:**
  the attacker has access to only ciphertext(s);

Attack Models: Adversarial Access

- **Ciphertext only attack:**
  the attacker has access to only ciphertext(s);
- **Known plaintext attack:**
  the attacker *knows* $(P_1, C_1), \ldots, (P_t, C_t)$;

Attack Models: Adversarial Access

- **Ciphertext only attack:**
  the attacker has access to only ciphertext(s);
- **Known plaintext attack:**
  the attacker *knows* $(P_1, C_1), \ldots, (P_t, C_t)$;
- **Chosen plaintext attack:**
  the attacker *chooses* $P_1, \ldots, P_t$; receives $C_1, \ldots, C_t$;
  - For additive stream ciphers, this is the same as known plaintext attack.

Attack Models: Adversarial Access (contd.)

- **Known/Chosen IV attack:** (resynchronization attack)
  the attacker *knows/chooses* $\mathrm{IV}_1, \ldots, \mathrm{IV}_t$;
  receives the corresponding keystreams.
  - Obtaining keystreams correspond to known plaintexts.
  - IVs are always known.

Attack Models: Adversarial Access (contd.)

- **Known/Chosen IV attack:** (resynchronization attack)
  the attacker *knows/chooses* $\mathrm{IV}_1, \ldots, \mathrm{IV}_t$;
  receives the corresponding keystreams.
  - Obtaining keystreams correspond to known plaintexts.
  - IVs are always known.
- **Chosen ciphertext attack.**
  the attacker *chooses* $C_1, \ldots, C_t$; receives $P_1, \ldots, P_t$;
  - Not very meaningful for usual additive stream ciphers.
  - Serious threat for self-synchronising stream ciphers.
  - Serious threat for stream ciphers which combine encryption and authentication in a single composite primitive.

Attack Models: Adversarial Goals

- **Key recovery:** the ultimate goal of the adversary.

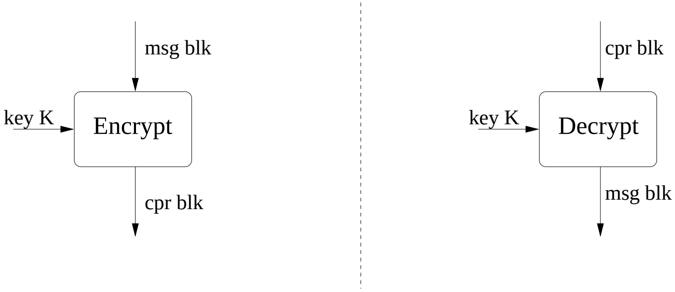
Attack Models: Adversarial Goals

- **Key recovery:** the ultimate goal of the adversary.
- **State recovery:**
  - This allows forward generation of the keystream.
  - If the state update function is invertible, then this allows to move backwards.
  - If the initialisation function is invertible, then this allows key recovery.

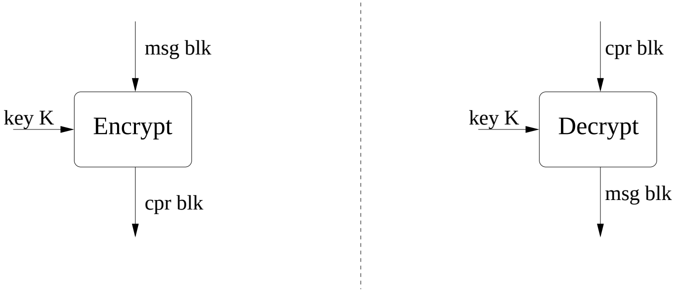
Attack Models: Adversarial Goals

- **Key recovery:** the ultimate goal of the adversary.
- **State recovery:**
  - This allows forward generation of the keystream.
  - If the state update function is invertible, then this allows to move backwards.
  - If the initialisation function is invertible, then this allows key recovery.
- **Distinguishing attack:**
  - Define a test statistic on a bit string such that the values it takes for uniform random strings and for the real keystream are 'significantly' different.
  - Sometimes distinguishing attacks can be converted to key recovery attacks.
  - In case of chosen IV attacks, the goal is to distinguish between the set of keystreams and a set of uniform random strings of the same lengths.

Encrypting Short Fixed Length Strings



Encrypting Short Fixed Length Strings



Block Cipher.

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n.$$

$$D : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n.$$

For each $K \in \{0,1\}^k$,

$$D_K(E_K(M)) = M.$$

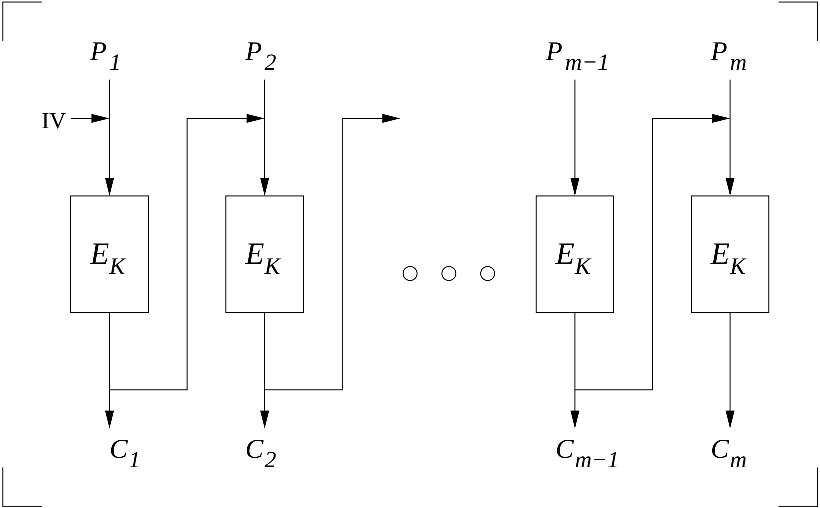
Modes of Operations

message: $M_1, M_2, M_3, \ldots$ (n-bit blocks);
initialization vector: n-bit IV (used as nonce).

Cipher block chaining (CBC) mode:

$C_1 = E_K(M_1 \oplus \mathrm{IV})$;
$C_i = E_K(M_i \oplus C_{i-1})$, $i \geq 2$.

CBC Mode



Modes of Operations (contd.)

message: $M_1, M_2, M_3, \ldots$ (n-bit blocks);
initialization vector: n-bit IV (used as nonce).

Output feedback (OFB) mode:
$Z_1 = E_K(\mathrm{IV}); Z_i = E_K(Z_{i-1}), i \geq 2;$
$C_i = M_i \oplus Z_i, i \geq 1.$

- This is essentially an additive stream cipher.

Modes of Operations (contd.)

message: $M_1, M_2, M_3, \ldots$ (n-bit blocks);
initialization vector: n-bit IV (used as nonce).

Output feedback (OFB) mode:
$Z_1 = E_K(\mathrm{IV})$; $Z_i = E_K(Z_{i-1})$, $i \geq 2$;
$C_i = M_i \oplus Z_i$, $i \geq 1$.

- This is essentially an additive stream cipher.

Cipher feedback (CFB) mode:
$C_1 = M_1 \oplus E_K(\mathrm{IV})$;
$C_i = M_i \oplus E_K(C_{i-1})$, $i \geq 2$.

- Can be used as a self-synchronizing stream cipher in a 1-bit feedback mode.

Modes of Operations (contd.)

message: $M_1, M_2, M_3, \ldots$ (n-bit blocks);
initialization vector: n-bit IV (used as nonce).

Output feedback (OFB) mode:
$Z_1 = E_K(\mathrm{IV})$; $Z_i = E_K(Z_{i-1})$, $i \geq 2$;
$C_i = M_i \oplus Z_i$, $i \geq 1$.

- This is essentially an additive stream cipher.

Cipher feedback (CFB) mode:
$C_1 = M_1 \oplus E_K(\mathrm{IV})$;
$C_i = M_i \oplus E_K(C_{i-1})$, $i \geq 2$.

- Can be used as a self-synchronizing stream cipher in a 1-bit feedback mode.

Counter (CTR) mode:
$C_i = M_i \oplus E_K(\mathrm{nonce}||\mathsf{bin}(i))$, $i \geq 1$.

- Other variants of the CTR mode have been proposed.

Linear Feedback Shift Register

Given (non-zero) initial state $(a_0, \ldots, a_{n-1})$ generates a sequence

$$a_0, a_1, a_2, \ldots, a_i, \ldots$$

where $a_i = c_{n-1}a_{i-1} \oplus \cdots \oplus c_1 a_{i-n+1} + c_0 a_{i-n}$.

Linear Feedback Shift Register

Given (non-zero) initial state $(a_0, \ldots, a_{n-1})$ generates a sequence

$$a_0, a_1, a_2, \ldots, a_i, \ldots$$

where $a_i = c_{n-1}a_{i-1} \oplus \cdots \oplus c_1 a_{i-n+1} + c_0 a_{i-n}$.
Characteristic (connection) polynomial:

$$\tau(x) = x^n \oplus c_{n-1}x^{n-1} \oplus \cdots \oplus c_1 x \oplus c_0.$$

Linear Feedback Shift Register

Given (non-zero) initial state $(a_0, \ldots, a_{n-1})$ generates a sequence

$$a_0, a_1, a_2, \ldots, a_i, \ldots$$

where $a_i = c_{n-1}a_{i-1} \oplus \cdots \oplus c_1 a_{i-n+1} + c_0 a_{i-n}$.
Characteristic (connection) polynomial:

$$\tau(x) = x^n \oplus c_{n-1}x^{n-1} \oplus \cdots \oplus c_1 x \oplus c_0.$$

- If $\tau(x)$ is primitive over $GF(2)$, then the period of $\{a_i\}$ is $2^n - 1$.
- Other well-understood “randomness-like” properties.

Linear Feedback Shift Register

Given (non-zero) initial state $(a_0, \ldots, a_{n-1})$ generates a sequence

$$a_0, a_1, a_2, \ldots, a_i, \ldots$$

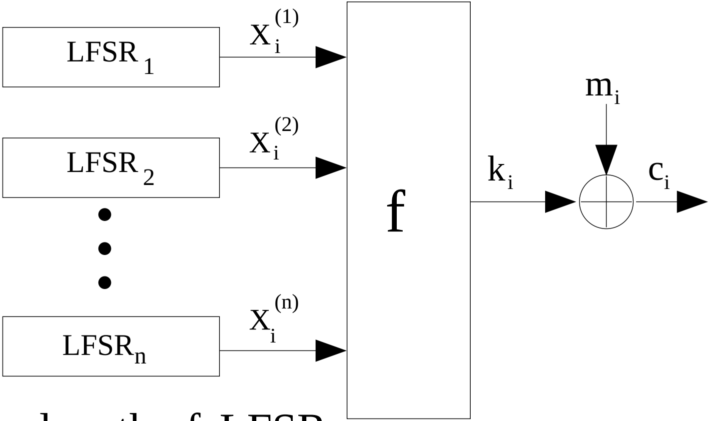
where $a_i = c_{n-1}a_{i-1} \oplus \cdots \oplus c_1 a_{i-n+1} + c_0 a_{i-n}$.
Characteristic (connection) polynomial:

$$\tau(x) = x^n \oplus c_{n-1}x^{n-1} \oplus \cdots \oplus c_1 x \oplus c_0.$$

- If $\tau(x)$ is primitive over $GF(2)$, then the period of $\{a_i\}$ is $2^n - 1$.
- Other well-understood “randomness-like” properties.

- Any bit of the sequence is a linear combination of the first n bits.
- Given any n bits of the sequence, it is easy to get the initial state.
- Unsuitable for direct use in cryptography.

Nonlinear Combiner Model



m_i = length of $LFSR_i$

Correlation Attacks.

Correlation Attack

Suppose

$$\Pr\left[X_1^{(i)} = k_i\right] = p \neq \frac{1}{2}.$$

Divide-and-conquer attack.

- Collect ℓ bits of the keystream.
- From each possible $2^{m_1} - 1$ non-zero initial states of LFSR_1, generate ℓ bits of the LFSR sequence.
- Let s be the number of places where the LFSR sequence equals the keystream sequence.
- If $s \approx \ell p$, then the corresponding state is likely to be the correct intial state.
- If $s \approx \ell/2$, then the corresponding state is unlikely to be the correct intial state.

Correlation Attack (contd.)

- For the attack to work ℓ must be at least $m_1/(1 - H(p))$.
- If $p = 1/2$ the attack does not work.

Correlation Attack (contd.)

- For the attack to work ℓ must be at least $m_1/(1 - H(p))$.
- If $p = 1/2$ the attack does not work.
- But, if $\Pr\left[X_1^{(i)} \oplus X_2^{(i)} = k_i\right] = p \neq \frac{1}{2}$ then the LFSRs 1 and 2 can be attacked simultaneously.

Correlation Attack (contd.)

- For the attack to work ℓ must be at least $m_1/(1-H(p))$.
- If $p = 1/2$ the attack does not work.
- But, if $\Pr\left[X_1^{(i)} \oplus X_2^{(i)} = k_i\right] = p \neq \frac{1}{2}$ then the LFSRs 1 and 2 can be attacked simultaneously.
- In general, if

$$\Pr\left[X_{j_1}^{(i)} \oplus \cdots \oplus X_{j_r}^{(i)} = k_i\right] = p \neq \frac{1}{2}$$

then the LFSRs $j_1, \ldots, j_r$ can be attacked simulatenously.

Correlation Attack (contd.)

- For the attack to work ℓ must be at least $m_1/(1 - H(p))$.
- If $p = 1/2$ the attack does not work.
- But, if $\Pr\left[X_1^{(i)} \oplus X_2^{(i)} = k_i\right] = p \neq \frac{1}{2}$ then the LFSRs 1 and 2 can be attacked simultaneously.
- In general, if

$$\Pr\left[X_{j_1}^{(i)} \oplus \cdots \oplus X_{j_r}^{(i)} = k_i\right] = p \neq \frac{1}{2}$$

  then the LFSRs $j_1, \ldots, j_r$ can be attacked simulatenously.
- Leads to Boolean function design criteria and trade-offs.
  - Balancedness.
  - Correlation immunity (resilience).
  - Algebraic degree.
  - Nonlinearity.
  - Other properties: propagation criteria, strict avalanche criteria, ...

Fast Correlation Attacks

Coding theory framework:

State S of an LFSR is expanded to sequence **a** which is perturbed by non-linear noise **e** to obtain ciphertext **c** with $p = \Pr[e_i = 0] \neq 1/2$.

Fast Correlation Attacks

Coding theory framework:

State S of an LFSR is expanded to sequence **a** which is perturbed by non-linear noise **e** to obtain ciphertext **c** with $p = \Pr[e_i = 0] \neq 1/2$.

View the expansion of S to **a** as the encoding procedure of a linear code.

Fast Correlation Attacks

Coding theory framework:

State S of an LFSR is expanded to sequence **a** which is perturbed by non-linear noise **e** to obtain ciphertext **c** with $p = \Pr[e_i = 0] \neq 1/2$.

View the expansion of S to **a** as the encoding procedure of a linear code.

Given **c**, using suitable decoding technique to obtain S.

An Iterative Decoding Procedure

- Generation of parity checks: find a number of linear relations that a bit a_i in the sequence $\mathbf{a}$ should satisfy.

An Iterative Decoding Procedure

- Generation of parity checks: find a number of linear relations that a bit a_i in the sequence **a** should satisfy.
  - Shifting, squaring and multiples of the connection polynomial.

An Iterative Decoding Procedure

- Generation of parity checks: find a number of linear relations that a bit a_i in the sequence **a** should satisfy.
  - Shifting, squaring and multiples of the connection polynomial.
- Use **k** as an approximation of **a** and find the number of equations involving a_i that hold for k_i.
- If this number is less than a threshold, then complement k_i.

An Iterative Decoding Procedure

- Generation of parity checks: find a number of linear relations that a bit a_i in the sequence **a** should satisfy.
  - Shifting, squaring and multiples of the connection polynomial.
- Use **k** as an approximation of **a** and find the number of equations involving a_i that hold for k_i.
- If this number is less than a threshold, then complement k_i.
- Iterate the procedure until the sequence satisfies the LFSR recurrence.

An Iterative Decoding Procedure

- Generation of parity checks: find a number of linear relations that a bit a_i in the sequence **a** should satisfy.
  - Shifting, squaring and multiples of the connection polynomial.
- Use **k** as an approximation of **a** and find the number of equations involving a_i that hold for k_i.
- If this number is less than a threshold, then complement k_i.
- Iterate the procedure until the sequence satisfies the LFSR recurrence.
- Works well if the number of taps in the LFSR is small.

Improvements to Correlation Attacks

- Identify an embedded low-rate convolutional code in the LFSR code; use Viterbi algorithm for decoding.

Improvements to Correlation Attacks

- Identify an embedded low-rate convolutional code in the LFSR code; use Viterbi algorithm for decoding.
- Turbo code techniques.
  - Identify "parallel" embedded convolutional code in the LFSR code.
  - The keystream sequence is used to construct received sequences for the convolutional codes.
  - These are decoded using an iterative algorithm.

Improvements to Correlation Attacks

- Identify an embedded low-rate convolutional code in the LFSR code; use Viterbi algorithm for decoding.
- Turbo code techniques.
  - Identify “parallel” embedded convolutional code in the LFSR code.
  - The keystream sequence is used to construct received sequences for the convolutional codes.
  - These are decoded using an iterative algorithm.
- List decoding techniques.

Improvements to Correlation Attacks

A different view: Reconstruction of linear polynomials.

Bit a_i is a linear combination $a_i = \bigoplus_{j=0}^{m_1-1} w_{i,j} a_j$; where $w_{i,j}$s can be computed from $\tau(x)$.

Improvements to Correlation Attacks

A different view: Reconstruction of linear polynomials.

Bit a_i is a linear combination $a_i = \bigoplus_{j=0}^{m_1-1} w_{i,j} a_j$; where $w_{i,j}$s can be computed from $\tau(x)$.

Let $\mathbf{w}_i = (w_{i,0}, \ldots, w_{i,m_1-1})$ and define $A(x) = \bigoplus_{j=0}^{m_1-1} x_j a_j$.

- The values $a_0, \ldots, a_{m_1-1}$ define the polynomial and are unknown.
- Then $A(x)$ is a linear polynomial and $a_i = A(\mathbf{w}_i)$ for $i \geq m_1$.

Improvements to Correlation Attacks

A different view: Reconstruction of linear polynomials.

Bit a_i is a linear combination $a_i = \bigoplus_{j=0}^{m_1-1} w_{i,j}a_j$; where $w_{i,j}$s can be computed from $\tau(x)$.

Let $\mathbf{w}_i = (w_{i,0}, \ldots, w_{i,m_1-1})$ and define $A(x) = \bigoplus_{j=0}^{m_1-1} x_j a_j$.

- The values $a_0, \ldots, a_{m_1-1}$ define the polynomial and are unknown.
- Then $A(x)$ is a linear polynomial and $a_i = A(\mathbf{w}_i)$ for $i \geq m_1$.
- k_i is a noisy output of the *unknown* polynomial $A(x)$ evaluated at the *known* point $\mathbf{w}_i$.

Improvements to Correlation Attacks

A different view: Reconstruction of linear polynomials.

Bit a_i is a linear combination $a_i = \bigoplus_{j=0}^{m_1-1} w_{i,j}a_j$; where $w_{i,j}$s can be computed from $\tau(x)$.

Let $\mathbf{w}_i = (w_{i,0}, \ldots, w_{i,m_1-1})$ and define $A(x) = \bigoplus_{j=0}^{m_1-1} x_j a_j$.

- The values $a_0, \ldots, a_{m_1-1}$ define the polynomial and are unknown.
- Then $A(x)$ is a linear polynomial and $a_i = A(\mathbf{w}_i)$ for $i \geq m_1$.
- k_i is a noisy output of the *unknown* polynomial $A(x)$ evaluated at the *known* point $\mathbf{w}_i$.
- Use of techniques from computational learning theory due to Goldreich, Rubinfeld and Sudan to reconstruct f from the k_is.

Improvements to Correlation Attacks

A different view: Reconstruction of linear polynomials.

Bit a_i is a linear combination $a_i = \bigoplus_{j=0}^{m_1-1} w_{i,j} a_j$; where $w_{i,j}$s can be computed from $\tau(x)$.

Let $\mathbf{w}_i = (w_{i,0}, \ldots, w_{i,m_1-1})$ and define $A(x) = \bigoplus_{j=0}^{m_1-1} x_j a_j$.

- The values $a_0, \ldots, a_{m_1-1}$ define the polynomial and are unknown.
- Then $A(x)$ is a linear polynomial and $a_i = A(\mathbf{w}_i)$ for $i \geq m_1$.
- k_i is a noisy output of the *unknown* polynomial $A(x)$ evaluated at the *known* point $\mathbf{w}_i$.
- Use of techniques from computational learning theory due to Goldreich, Rubinfeld and Sudan to reconstruct f from the k_is.
- The application is not straightforward, there are a few tricks involved.

Other Kinds of Correlations

- Correlations between linear functions of several output bits and linear functions of a subset of LFSR bits.
  - For strong enough correlations, a number of stochastic equations may be derived.
  - If the known keystream sequence is long enough, then the equations can be solved.

Other Kinds of Correlations

- Correlations between linear functions of several output bits and linear functions of a subset of LFSR bits.
  - For strong enough correlations, a number of stochastic equations may be derived.
  - If the known keystream sequence is long enough, then the equations can be solved.
- Keystream (or simply key) correlation: leads to distinguishing attacks.
  - Bias in a particular keystream bit or a linear combination of keystream bits, eg. $\Pr[k_{16} = 0] \neq 1/2$.
    Attack types: multiple keys; or, single key but, multiple IVs.
  - Bias in a subsequence of key bits,
    eg. $\Pr[k_i = k_{i+3}] \neq 1/2$ for all $i \geq 0$.

Some References: Correlation Attacks

- T. Siegenthaler: Decrypting a Class of Stream Ciphers Using Ciphertext Only. IEEE Trans. Computers 34(1): (1985).
- T. Siegenthaler: Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans. on Inf. Th. 30(5): (1984).

Some References: Correlation Attacks

- T. Siegenthaler: Decrypting a Class of Stream Ciphers Using Ciphertext Only. IEEE Trans. Computers 34(1): (1985).
- T. Siegenthaler: Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans. on Inf. Th. 30(5): (1984).
- W. Meier, O. Staffelbach: Fast Correlation Attacks on Certain Stream Ciphers. J. Cryptology 1(3): (1989).
- J. Dj. Golić: Correlation Properties of a General Binary Combiner with Memory. J. Cryptology 9(2): (1996).

Some References: Correlation Attacks

- T. Siegenthaler: Decrypting a Class of Stream Ciphers Using Ciphertext Only. IEEE Trans. Computers 34(1): (1985).
- T. Siegenthaler: Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans. on Inf. Th. 30(5): (1984).
- W. Meier, O. Staffelbach: Fast Correlation Attacks on Certain Stream Ciphers. J. Cryptology 1(3): (1989).
- J. Dj. Golić: Correlation Properties of a General Binary Combiner with Memory. J. Cryptology 9(2): (1996).
- T. Johansson, F. Jönsson: Fast Correlation Attacks Based on Turbo Code Techniques. CRYPTO 1999.
- T. Johansson, F. Jönsson: Fast Correlation Attacks through Reconstruction of Linear Polynomials. CRYPTO 2000.
- M. J. Mihaljevic, M. P. C. Fossorier, H. Imai: Fast Correlation Attack Algorithm with List Decoding and an Application. FSE 2001.

Algebraic Attacks.

Algebraic Attacks: Basic Idea

Let L be the update functions of all the LFSRs.

- Each LFSR is updated using a linear function and let L be the applications of these linear functions to the respective states.
- L is a linear function on the whole state.

Algebraic Attacks: Basic Idea

Let L be the update functions of all the LFSRs.

- Each LFSR is updated using a linear function and let L be the applications of these linear functions to the respective states.
- L is a linear function on the whole state.

Let $(s_0, \ldots, s_{n-1})$ be the n-bit state at time i.
Keystream:

$$\begin{array}{rcl} f(s_0, \ldots, s_{n-1}) & = & k_i \\ f(L(s_0, \ldots, s_{n-1})) & = & k_{i+1} \\ f(L^2(s_0, \ldots, s_{n-1})) & = & k_{i+2} \\ \cdots & \cdots & \cdots \end{array}$$

Each of the expressions on the left have degree $d \triangleq \deg(f)$.

Solving Equations

There are $\sum_{j=1}^{d} \binom{n}{j}$ monomials of degree at most d.

- Replace each monomial by a new variable.
- Solve the resulting system of linear equations.
  - Sufficient number of keystream bits required to get an over-defined system of equations.
- From the solution to the linear system, obtain the solution to the original system.

Solving Equations

There are $\sum_{j=1}^{d} \binom{n}{j}$ monomials of degree at most d.

- Replace each monomial by a new variable.
- Solve the resulting system of linear equations.
  - Sufficient number of keystream bits required to get an over-defined system of equations.
- From the solution to the linear system, obtain the solution to the original system.

Use Gröbner basis based technique to directly solve the system of multivariate polynomial equations over $\mathbb{F}_2$.

- Becomes progressively inefficient as d increases.
- The linearisation technique also essentially computes the Gröbner basis.

Controlling the Degree

Suppose g is a function such that $\deg(f \times g) < \deg(g)$.
Example: $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_1x_2x_3$ and $g(x_1, x_2, x_3) = x_2x_3$.

Controlling the Degree

Suppose g is a function such that $\deg(f \times g) < \deg(g)$.
Example: $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_1 x_2 x_3$ and $g(x_1, x_2, x_3) = x_2 x_3$.

$$
\begin{array}{rcl}
f(s_0, \ldots, s_{n-1}) g(s_0, \ldots, s_{n-1}) & = & k_i \cdot g(s_0, \ldots, s_{n-1}) \\
f(L(s_0, \ldots, s_{n-1})) g(L(s_0, \ldots, s_{n-1})) & = & k_{i+1} \cdot g(L(s_0, \ldots, s_{n-1})) \\
f(L^2(s_0, \ldots, s_{n-1})) g(L^2(s_0, \ldots, s_{n-1})) & = & k_{i+2} \cdot g(L^2(s_0, \ldots, s_{n-1})) \\
\cdots & \cdots & \cdots
\end{array}
$$

Controlling the Degree

Suppose g is a function such that $\deg(f \times g) < \deg(g)$.
Example: $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_1 x_2 x_3$ and $g(x_1, x_2, x_3) = x_2 x_3$.

$$\begin{array}{rcl} f(s_0, \ldots, s_{n-1}) g(s_0, \ldots, s_{n-1}) & = & k_i \cdot g(s_0, \ldots, s_{n-1}) \\ f(L(s_0, \ldots, s_{n-1})) g(L(s_0, \ldots, s_{n-1})) & = & k_{i+1} \cdot g(L(s_0, \ldots, s_{n-1})) \\ f(L^2(s_0, \ldots, s_{n-1})) g(L^2(s_0, \ldots, s_{n-1})) & = & k_{i+2} \cdot g(L^2(s_0, \ldots, s_{n-1})) \\ \cdots & \cdots & \cdots \end{array}$$

If $\deg(g) < d$ *or* $k_j = 0$ (which happens roughly half of the times), then we get a system of equations whose degrees are less than d.

- Finding a “good” g is important.

A General Formulation

Let $\mathbf{s} = (s_0, \ldots, s_{n-1})$. Find a Boolean function $\widehat{f}$ such that for some $\delta \geq 0$

$$\widehat{f}(L^t(\mathbf{s}), \ldots, L^{t+\delta}(\mathbf{s}), k_t, \ldots, k_{t+\delta}) = 0.$$

- For $\delta = 0$, take $\widehat{f} = f$.

A General Formulation

Let $\mathbf{s} = (s_0, \ldots, s_{n-1})$. Find a Boolean function $\widehat{f}$ such that for some $\delta \geq 0$

$$\widehat{f}(L^t(\mathbf{s}), \ldots, L^{t+\delta}(\mathbf{s}), k_t, \ldots, k_{t+\delta}) = 0.$$

- For $\delta = 0$, take $\widehat{f} = f$.

Suppose $\widehat{f}$ can be written as

$$\begin{aligned} & \widehat{f}(L^t(\mathbf{s}), \ldots, L^{t+\delta}(\mathbf{s}), k_t, \ldots, k_{t+\delta}) \\ = \quad & h(L^t(\mathbf{s}), \ldots, L^{t+\delta}(\mathbf{s})) \oplus g(L^t(\mathbf{s}), \ldots, L^{t+\delta}(\mathbf{s}), k_t, \ldots, k_{t+\delta}) \\ = \quad & h_t(\mathbf{s}) \oplus g_t(\mathbf{s}, k_t, \ldots, k_{t+\delta}) \end{aligned}$$

where the degree e of $\mathbf{s}$ in g is less than the degree d of $\mathbf{s}$ in $\widehat{f}$.

A General Formulation (contd.)

Assume that the attacker can find constants $c_0, \ldots, c_{T-1}$ such that

$$\bigoplus_{j=0}^{T-1} c_j h_{t+j}(\mathbf{s}) = 0.$$

A General Formulation (contd.)

Assume that the attacker can find constants $c_0, \ldots, c_{T-1}$ such that

$$\bigoplus_{j=0}^{T-1} c_j h_{t+j}(\mathbf{s}) = 0.$$

Using

$$0 = \widehat{f}(L^t(\mathbf{s}), \ldots, L^{t+\delta}(\mathbf{s}), k_t, \ldots, k_{t+\delta}) = h_t(\mathbf{s}) \oplus g_t(\mathbf{s}, k_t, \ldots, k_{t+\delta})$$

we can write

$$\bigoplus_{j=0}^{T-1} c_j g_{t+j}(\mathbf{s}, k_t, \ldots, k_{t+\delta}) = 0.$$

A General Formulation (contd.)

Assume that the attacker can find constants $c_0, \ldots, c_{T-1}$ such that

$$\bigoplus_{j=0}^{T-1} c_j h_{t+j}(\mathbf{s}) = 0.$$

Using

$$0 = \widehat{f}(L^t(\mathbf{s}), \ldots, L^{t+\delta}(\mathbf{s}), k_t, \ldots, k_{t+\delta}) = h_t(\mathbf{s}) \oplus g_t(\mathbf{s}, k_t, \ldots, k_{t+\delta})$$

we can write

$$\bigoplus_{j=0}^{T-1} c_j g_{t+j}(\mathbf{s}, k_t, \ldots, k_{t+\delta}) = 0.$$

This is an equation of lower degree e in the unknown $\mathbf{s}$.

A General Formulation (contd.)

Finding the constants $c_0, \ldots, c_{T-1}$.

- Choose a "reasonable" value $\mathbf{s}^*$ of $\mathbf{s}$.
- Compute $\hat{k}_t = h_t(\mathbf{s}^*)$ for $t = 0, \ldots, 2T - 1$.

A General Formulation (contd.)

Finding the constants $c_0, \ldots, c_{T-1}$.

- Choose a "reasonable" value $\mathbf{s}^*$ of $\mathbf{s}$.
- Compute $\hat{k}_t = h_t(\mathbf{s}^*)$ for $t = 0, \ldots, 2T - 1$.
- Use Berlekamp-Massey algorithm to find $c_0, \ldots, c_{T-1}$ such that

$$0 = \bigoplus_{j=0}^{T-1} c_j \hat{k}_{t+j}$$

A General Formulation (contd.)

Finding the constants $c_0, \ldots, c_{T-1}$.

- Choose a "reasonable" value $\mathbf{s}^*$ of $\mathbf{s}$.
- Compute $\hat{k}_t = h_t(\mathbf{s}^*)$ for $t = 0, \ldots, 2T - 1$.
- Use Berlekamp-Massey algorithm to find $c_0, \ldots, c_{T-1}$ such that

$$\begin{aligned} 0 &= \bigoplus_{j=0}^{T-1} c_j \hat{k}_{t+j} \\ &= \bigoplus_{j=0}^{T-1} c_j h_{t+j}(\mathbf{s}^*). \end{aligned}$$

A General Formulation (contd.)

Finding the constants $c_0, \ldots, c_{T-1}$.

- Choose a "reasonable" value $\mathbf{s}^*$ of $\mathbf{s}$.
- Compute $\hat{k}_t = h_t(\mathbf{s}^*)$ for $t = 0, \ldots, 2T - 1$.
- Use Berlekamp-Massey algorithm to find $c_0, \ldots, c_{T-1}$ such that

$$\begin{aligned} 0 &= \bigoplus_{j=0}^{T-1} c_j \hat{k}_{t+j} \\ &= \bigoplus_{j=0}^{T-1} c_j h_{t+j}(\mathbf{s}^*). \end{aligned}$$

- Requires $O(T^2)$ time.
- The proof that these $c_0, \ldots, c_{T-1}$ work for all $\mathbf{s}$ is non-trivial.

Some References: Algebraic Attacks

- N. Courtois, W. Meier: Algebraic Attacks on Stream Ciphers with Linear Feedback. EUROCRYPT 2003.

Some References: Algebraic Attacks

- N. Courtois, W. Meier: Algebraic Attacks on Stream Ciphers with Linear Feedback. EUROCRYPT 2003.
- N. Courtois: Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. CRYPTO 2003.
- F. Armknecht, M. Krause: Algebraic Attacks on Combiners with Memory. CRYPTO 2003.
- Frederik Armknecht: Improving Fast Algebraic Attacks. FSE 2004.

Some References: Algebraic Attacks

- N. Courtois, W. Meier: Algebraic Attacks on Stream Ciphers with Linear Feedback. EUROCRYPT 2003.
- N. Courtois: Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. CRYPTO 2003.
- F. Armknecht, M. Krause: Algebraic Attacks on Combiners with Memory. CRYPTO 2003.
- Frederik Armknecht: Improving Fast Algebraic Attacks. FSE 2004.
- N. Courtois, A. Klimov, J. Patarin, A. Shamir: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. EUROCRYPT 2000.
- C. Diem: The XL-Algorithm and a Conjecture from Commutative Algebra. ASIACRYPT 2004.
- G. Ars, J.-C. Faugére, H. Imai, M. Kawazoe, M. Sugita: Comparison Between XL and Gröbner Basis Algorithms. ASIACRYPT 2004.

Differential Attacks.

Trivium: A Counter-Point to Correlation and Algebraic Attacks

State: $(s_1^{(i)}, \ldots, s_{288}^{(i)})$: (Super-script i is omitted for simplicity.)

- State update function is non-linear.
- Output function is linear.

Trivium: A Counter-Point to Correlation and Algebraic Attacks

State: $(s_1^{(i)}, \ldots, s_{288}^{(i)})$: (Super-script i is omitted for simplicity.)

- State update function is non-linear.
- Output function is linear.

$t_1 = s_{66} \oplus s_{93}$; $t_2 = s_{162} \oplus s_{177}$; $t_3 = s_{243} \oplus s_{288}$;
$k_i = t_1 \oplus t_2 \oplus t_3$;

Trivium: A Counter-Point to Correlation and Algebraic Attacks

State: $(s_1^{(i)}, \ldots, s_{288}^{(i)})$: (Super-script i is omitted for simplicity.)

- State update function is non-linear.
- Output function is linear.

$t_1 = s_{66} \oplus s_{93}$; $t_2 = s_{162} \oplus s_{177}$; $t_3 = s_{243} \oplus s_{288}$;
$k_i = t_1 \oplus t_2 \oplus t_3$;

$t_1 = t_1 \oplus s_{91} \cdot s_{92} \oplus s_{171}$;
$t_2 = t_2 \oplus s_{175} \cdot s_{176} \oplus s_{264}$;
$t_3 = t_3 \oplus s_{286} \cdot s_{287} \oplus s_{69}$;

Trivium: A Counter-Point to Correlation and Algebraic Attacks

State: $(s_1^{(i)}, \ldots, s_{288}^{(i)})$: (Super-script i is omitted for simplicity.)

- State update function is non-linear.
- Output function is linear.

$t_1 = s_{66} \oplus s_{93}$; $t_2 = s_{162} \oplus s_{177}$; $t_3 = s_{243} \oplus s_{288}$;
$k_i = t_1 \oplus t_2 \oplus t_3$;

$$t_1 = t_1 \oplus s_{91} \cdot s_{92} \oplus s_{171};$$
$$t_2 = t_2 \oplus s_{175} \cdot s_{176} \oplus s_{264};$$
$$t_3 = t_3 \oplus s_{286} \cdot s_{287} \oplus s_{69};$$

$$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92});$$
$$(s_{94}, s_{95} \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176});$$
$$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287});$$

Derivatives

Given an n-variable Boolean function $f(\mathbf{x})$ and $\mathbf{a} \in \{0,1\}^n$, the derivative of f at $\mathbf{a}$ is defined to be a Boolean function

$$\Delta_{\mathbf{a}} f(x) \triangleq f(\mathbf{x} \oplus \mathbf{a}) \oplus f(\mathbf{x}).$$

Derivatives

Given an n-variable Boolean function $f(\mathbf{x})$ and $\mathbf{a} \in \{0,1\}^n$, the derivative of f at $\mathbf{a}$ is defined to be a Boolean function

$$\Delta_{\mathbf{a}} f(x) \triangleq f(\mathbf{x} \oplus \mathbf{a}) \oplus f(\mathbf{x}).$$

Extension:

$$\Delta^{(2)}_{\mathbf{a}_1,\mathbf{a}_2} f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{a}_1 \oplus \mathbf{a}_2) \oplus f(\mathbf{x} \oplus \mathbf{a}_1) \oplus f(\mathbf{x} \oplus \mathbf{a}_2) \oplus f(\mathbf{x}).$$

Derivatives

Given an n-variable Boolean function $f(\mathbf{x})$ and $\mathbf{a} \in \{0,1\}^n$, the derivative of f at $\mathbf{a}$ is defined to be a Boolean function

$$\Delta_{\mathbf{a}} f(x) \triangleq f(\mathbf{x} \oplus \mathbf{a}) \oplus f(\mathbf{x}).$$

Extension:

$$\Delta^{(2)}_{\mathbf{a}_1,\mathbf{a}_2} f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{a}_1 \oplus \mathbf{a}_2) \oplus f(\mathbf{x} \oplus \mathbf{a}_1) \oplus f(\mathbf{x} \oplus \mathbf{a}_2) \oplus f(\mathbf{x}).$$

Other direction: $f(\mathbf{x} \oplus \mathbf{a}_1 \oplus \mathbf{a}_2) = \Delta^{(2)}_{\mathbf{a}_1,\mathbf{a}_2} f(\mathbf{x}) \oplus \Delta_{\mathbf{a}_1} f(\mathbf{x}) \oplus \Delta_{\mathbf{a}_2} f(\mathbf{x}) \oplus f(\mathbf{x})$.

Derivatives

Given an n-variable Boolean function $f(\mathbf{x})$ and $\mathbf{a} \in \{0,1\}^n$, the derivative of f at $\mathbf{a}$ is defined to be a Boolean function

$$\Delta_{\mathbf{a}} f(x) \triangleq f(\mathbf{x} \oplus \mathbf{a}) \oplus f(\mathbf{x}).$$

Extension:

$$\Delta^{(2)}_{\mathbf{a}_1,\mathbf{a}_2} f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{a}_1 \oplus \mathbf{a}_2) \oplus f(\mathbf{x} \oplus \mathbf{a}_1) \oplus f(\mathbf{x} \oplus \mathbf{a}_2) \oplus f(\mathbf{x}).$$

Other direction: $f(\mathbf{x} \oplus \mathbf{a}_1 \oplus \mathbf{a}_2) = \Delta^{(2)}_{\mathbf{a}_1,\mathbf{a}_2} f(\mathbf{x}) \oplus \Delta_{\mathbf{a}_1} f(\mathbf{x}) \oplus \Delta_{\mathbf{a}_2} f(\mathbf{x}) \oplus f(\mathbf{x})$.

$$f(x \oplus \mathbf{a}_1 \oplus \cdots \oplus \mathbf{a}_n) = \bigoplus_{i=0}^{n} \bigoplus_{1 \leq j_1 < \cdots < j_i \leq n} \Delta^{(i)}_{\mathbf{a}_{j_1},\ldots,\mathbf{a}_{j_i}} f(\mathbf{x}).$$

Derivatives (contd.)

Properties.

- $\deg(\Delta_{\mathbf{a}} f) < \deg(f)$.
- $\Delta^{(2)}_{\mathbf{a}_1,\mathbf{a}_2} f(\mathbf{x}) = \Delta^{(2)}_{\mathbf{a}_2,\mathbf{a}_1} f(\mathbf{x})$.
- $\Delta_{\mathbf{a}}(f \oplus g) = \Delta_{\mathbf{a}} f \oplus \Delta_{\mathbf{a}} g$.
- $\Delta_{\mathbf{a}}(f(\mathbf{x})g(\mathbf{x})) = f(\mathbf{x} \oplus \mathbf{a})\Delta_{\mathbf{a}} g(\mathbf{x}) \oplus (\Delta_{\mathbf{a}} f(\mathbf{x}))g(\mathbf{x})$.
- If $\mathbf{a} \in \{0,1\}^n$ is such that $\mathrm{supp}(\mathbf{a}) \subset \{1,\ldots,i\}$, then

$$\Delta_{\mathbf{a}}(x_1 \cdots x_i f(x_{i+1},\ldots,x_n)) = f(x_{i+1},\ldots,x_n)\Delta_{\mathbf{a}}(x_1 \cdots x_i).$$

Derivatives (contd.)

Properties.

- $\deg(\Delta_{\mathbf{a}} f) < \deg(f)$.
- $\Delta^{(2)}_{\mathbf{a}_1,\mathbf{a}_2} f(\mathbf{x}) = \Delta^{(2)}_{\mathbf{a}_2,\mathbf{a}_1} f(\mathbf{x})$.
- $\Delta_{\mathbf{a}}(f \oplus g) = \Delta_{\mathbf{a}} f \oplus \Delta_{\mathbf{a}} g$.
- $\Delta_{\mathbf{a}}(f(\mathbf{x})g(\mathbf{x})) = f(\mathbf{x} \oplus \mathbf{a})\Delta_{\mathbf{a}} g(\mathbf{x}) \oplus (\Delta_{\mathbf{a}} f(\mathbf{x}))g(\mathbf{x})$.
- If $\mathbf{a} \in \{0,1\}^n$ is such that $\mathrm{supp}(\mathbf{a}) \subset \{1,\ldots,i\}$, then

$$\Delta_{\mathbf{a}}(x_1 \cdots x_i f(x_{i+1},\ldots,x_n)) = f(x_{i+1},\ldots,x_n)\Delta_{\mathbf{a}}(x_1 \cdots x_i).$$

  - Nothing special about $x_1 \cdots x_i$; easy modification for the monomial $x_{j_1} \cdots x_{j_i}$.

Derivatives (contd.)

- Let $C[\mathbf{a}_1, \ldots, \mathbf{a}_i]$ be the set of all linear combinations of $\mathbf{a}_1, \ldots, \mathbf{a}_i$. Then

$$\Delta^{(i)}_{\mathbf{a}_1,\ldots,\mathbf{a}_i} f(\mathbf{x}) = \bigoplus_{\mathbf{c} \in C[\mathbf{a}_1,\ldots,\mathbf{a}_i]} f(\mathbf{x} \oplus \mathbf{c}).$$

- Let $C[\mathbf{a}_1, \ldots, \mathbf{a}_i]$ be the set of all linear combinations of $\mathbf{a}_1, \ldots, \mathbf{a}_i$. Then
$$\Delta^{(i)}_{\mathbf{a}_1,\ldots,\mathbf{a}_i} f(\mathbf{x}) = \bigoplus_{\mathbf{c}\in C[\mathbf{a}_1,\ldots,\mathbf{a}_i]} f(\mathbf{x} \oplus \mathbf{c}).$$
- If $\mathbf{a}_i$ is linearly dependent on $\mathbf{a}_1, \ldots, \mathbf{a}_{i-1}$, then $\Delta^{(i)}_{\mathbf{a}_1,\ldots,\mathbf{a}_i} f(\mathbf{x}) = 0$.

Using Derivatives

Suppose $f(x_1, \ldots, x_n)$ can be written as

$$f(x_1, \ldots, x_n) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) \oplus h(x_1, \ldots, x_n)$$

where $x_1 \cdots x_i$ does not divide any monomial of $h(x_1, \ldots, x_n)$.

Using Derivatives

Suppose $f(x_1, \ldots, x_n)$ can be written as

$$f(x_1, \ldots, x_n) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) \oplus h(x_1, \ldots, x_n)$$

where $x_1 \cdots x_i$ does not divide any monomial of $h(x_1, \ldots, x_n)$.

Let $\mathbf{a}_1, \ldots, \mathbf{a}_i$ be linearly independent vectors such that $\mathrm{supp}(\mathbf{a}_1), \ldots, \mathrm{supp}(\mathbf{a}_i) \subset \{1, \ldots, i\}$.

Using Derivatives

Suppose $f(x_1, \ldots, x_n)$ can be written as

$$f(x_1, \ldots, x_n) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) \oplus h(x_1, \ldots, x_n)$$

where $x_1 \cdots x_i$ does not divide any monomial of $h(x_1, \ldots, x_n)$.

Let $\mathbf{a}_1, \ldots, \mathbf{a}_i$ be linearly independent vectors such that $\mathrm{supp}(\mathbf{a}_1), \ldots, \mathrm{supp}(\mathbf{a}_i) \subset \{1, \ldots, i\}$. Then

$$g(x_{i+1}, \ldots, x_n) = \Delta_{\mathbf{a}_1, \ldots, \mathbf{a}_i} f(x_1, \ldots, x_n)$$

Using Derivatives

Suppose $f(x_1, \ldots, x_n)$ can be written as

$$f(x_1, \ldots, x_n) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) \oplus h(x_1, \ldots, x_n)$$

where $x_1 \cdots x_i$ does not divide any monomial of $h(x_1, \ldots, x_n)$.

Let $\mathbf{a}_1, \ldots, \mathbf{a}_i$ be linearly independent vectors such that $\mathrm{supp}(\mathbf{a}_1), \ldots, \mathrm{supp}(\mathbf{a}_i) \subset \{1, \ldots, i\}$. Then

$$\begin{aligned} g(x_{i+1}, \ldots, x_n) &= \Delta_{\mathbf{a}_1, \ldots, \mathbf{a}_i} f(x_1, \ldots, x_n) \\ &= \bigoplus_{\mathbf{c} \in C[\mathbf{a}_1, \ldots, \mathbf{a}_i]} f(\mathbf{x} \oplus \mathbf{c}). \end{aligned}$$

Using Derivatives

Suppose $f(x_1, \ldots, x_n)$ can be written as

$$f(x_1, \ldots, x_n) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) \oplus h(x_1, \ldots, x_n)$$

where $x_1 \cdots x_i$ does not divide any monomial of $h(x_1, \ldots, x_n)$.

Let $\mathbf{a}_1, \ldots, \mathbf{a}_i$ be linearly independent vectors such that $\mathrm{supp}(\mathbf{a}_1), \ldots, \mathrm{supp}(\mathbf{a}_i) \subset \{1, \ldots, i\}$. Then

$$\begin{aligned} g(x_{i+1}, \ldots, x_n) &= \Delta_{\mathbf{a}_1, \ldots, \mathbf{a}_i} f(x_1, \ldots, x_n) \\ &= \bigoplus_{\mathbf{c} \in C[\mathbf{a}_1, \ldots, \mathbf{a}_i]} f(\mathbf{x} \oplus \mathbf{c}). \end{aligned}$$

Nothing special about $x_1 \cdots x_i$; easy modification for $x_{j_1} \cdots x_{j_i}$.

Maxterm: $x_{j_1} \cdots x_{j_i}$ is a maxterm if the corresponding g is of degree 1.

Using Derivatives (contd.)

Maxterm: $x_{j_1} \cdots x_{j_i}$ is a maxterm if the corresponding g is of degree 1.
Observation: If f is a random polynomial of degree d, then with high probability every degree $(d-1)$ monomial is a maxterm.

Using Derivatives (contd.)

Maxterm: $x_{j_1} \cdots x_{j_i}$ is a maxterm if the corresponding g is of degree 1.
Observation: If f is a random polynomial of degree d, then with high probability every degree $(d-1)$ monomial is a maxterm.

Suppose $x_1 \cdots x_i$ is a maxterm.

$$f(\mathbf{x}) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) + h(\mathbf{x}).$$

Using Derivatives (contd.)

Maxterm: $x_{j_1} \cdots x_{j_i}$ is a maxterm if the corresponding g is of degree 1.
Observation: If f is a random polynomial of degree d, then with high probability every degree $(d-1)$ monomial is a maxterm.

Suppose $x_1 \cdots x_i$ is a maxterm.

$$f(\mathbf{x}) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) + h(\mathbf{x}).$$

- Constant term of g is obtained by setting $x_{i+1}, \ldots, x_n$ to 0 and XORing together the values of f for all possible choices of $x_1, \ldots, x_i$.

Using Derivatives (contd.)

Maxterm: $x_{j_1} \cdots x_{j_i}$ is a maxterm if the corresponding g is of degree 1.
Observation: If f is a random polynomial of degree d, then with high probability every degree $(d-1)$ monomial is a maxterm.

Suppose $x_1 \cdots x_i$ is a maxterm.

$$f(\mathbf{x}) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) + h(\mathbf{x}).$$

- Constant term of g is obtained by setting $x_{i+1}, \ldots, x_n$ to 0 and XORing together the values of f for all possible choices of $x_1, \ldots, x_i$.
- The coefficient of x_j in g ($j > i$) is obtained by setting x_j to 1, all other $x_{i+1}, \ldots, x_n$ to 0 and XORing together the values of f for all possible choices of $x_1, \ldots, x_i$.

Using Derivatives (contd.)

Maxterm: $x_{j_1} \cdots x_{j_i}$ is a maxterm if the corresponding g is of degree 1.
Observation: If f is a random polynomial of degree d, then with high probability every degree $(d-1)$ monomial is a maxterm.

Suppose $x_1 \cdots x_i$ is a maxterm.

$$f(\mathbf{x}) = x_1 \cdots x_i g(x_{i+1}, \ldots, x_n) + h(\mathbf{x}).$$

- Constant term of g is obtained by setting $x_{i+1}, \ldots, x_n$ to 0 and XORing together the values of f for all possible choices of $x_1, \ldots, x_i$.
- The coefficient of x_j in g ($j > i$) is obtained by setting x_j to 1, all other $x_{i+1}, \ldots, x_n$ to 0 and XORing together the values of f for all possible choices of $x_1, \ldots, x_i$.

Nothing special about $x_1 \cdots x_i$; easy modification for $x_{j_1} \cdots x_{j_i}$.

Attacking Stream Ciphers With IV: Pre-Processing

Consider a stream cipher with secret key $K = (\kappa_1, \ldots, \kappa_n)$ and IV$= (v_1, \ldots, v_m)$.

Attacking Stream Ciphers With IV: Pre-Processing

Consider a stream cipher with secret key $K = (\kappa_1, \ldots, \kappa_n)$ and IV$= (v_1, \ldots, v_m)$. Any keystream bit k can be written as

$$k_t = f_t(K, \mathrm{IV}) = f(\kappa_1, \ldots, \kappa_n, v_1, \ldots, v_m).$$

Attacking Stream Ciphers With IV: Pre-Processing

Consider a stream cipher with secret key $K = (\kappa_1, \ldots, \kappa_n)$ and IV$= (v_1, \ldots, v_m)$. Any keystream bit k can be written as

$$k_t = f_t(K, \mathrm{IV}) = f(\kappa_1, \ldots, \kappa_n, v_1, \ldots, v_m).$$

Suppose f behaves like a random polynomial of degree d. Then with high probability every degree $(d-1)$ monomial consisting only of IV bits is a maxterm.

Attacking Stream Ciphers With IV: Pre-Processing

Consider a stream cipher with secret key $K = (\kappa_1, \ldots, \kappa_n)$ and IV$= (v_1, \ldots, v_m)$. Any keystream bit k can be written as

$$k_t = f_t(K, \mathrm{IV}) = f(\kappa_1, \ldots, \kappa_n, v_1, \ldots, v_m).$$

Suppose f behaves like a random polynomial of degree d. Then with high probability every degree $(d-1)$ monomial consisting only of IV bits is a maxterm.

Choose n such maxterms. In a pre-processing stage, the corresponding linear functions $g_1, \ldots, g_n$ are obtained ensuring that each g_j depends on at least one key bit.

Attacking Stream Ciphers With IV: Pre-Processing

Consider a stream cipher with secret key $K = (\kappa_1, \ldots, \kappa_n)$ and IV$= (v_1, \ldots, v_m)$. Any keystream bit k can be written as

$$k_t = f_t(K, \mathrm{IV}) = f(\kappa_1, \ldots, \kappa_n, v_1, \ldots, v_m).$$

Suppose f behaves like a random polynomial of degree d. Then with high probability every degree $(d-1)$ monomial consisting only of IV bits is a maxterm.

Choose n such maxterms. In a pre-processing stage, the corresponding linear functions $g_1, \ldots, g_n$ are obtained ensuring that each g_j depends on at least one key bit.

Let A be an $n \times n$ matrix representing these linear functions. It can be ensured with high probability that A is invertible.

Attacking Stream Ciphers With IV: On-line

Suppose $v_1 \cdots v_{d-1}$ be a maxterm and $g(K, v_d, \ldots, v_m)$ be the corresponding linear function.

Attacking Stream Ciphers With IV: On-line

Suppose $v_1 \cdots v_{d-1}$ be a maxterm and $g(K, v_d, \ldots, v_m)$ be the corresponding linear function. Let $\mathbf{a}_1, \ldots, \mathbf{a}_{d-1} \in \{0,1\}^{n+m}$ be l.i. with $\mathrm{supp}(\mathbf{a}_j)$ among the indices of $v_1, \ldots, v_{d-1}$; and let $\mathbf{b}_j$ be the restriction of $\mathbf{a}_j$ to the last m bits.

Attacking Stream Ciphers With IV: On-line

Suppose $v_1 \cdots v_{d-1}$ be a maxterm and $g(K, v_d, \ldots, v_m)$ be the corresponding linear function. Let $\mathbf{a}_1, \ldots, \mathbf{a}_{d-1} \in \{0,1\}^{n+m}$ be l.i. with $\mathrm{supp}(\mathbf{a}_j)$ among the indices of $v_1, \ldots, v_{d-1}$; and let $\mathbf{b}_j$ be the restriction of $\mathbf{a}_j$ to the last m bits. Then

$$\begin{aligned} g(K, v_d, \ldots, v_m) &= \bigoplus_{\mathbf{c} \in C[\mathbf{a}_1, \ldots, \mathbf{a}_{d-1}]} f((K, \mathrm{IV}) \oplus \mathbf{c}) \\ &= \bigoplus_{\mathbf{d} \in C[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]} f(K, \mathrm{IV} \oplus \mathbf{b}). \end{aligned}$$

Attacking Stream Ciphers With IV: On-line

Suppose $v_1 \cdots v_{d-1}$ be a maxterm and $g(K, v_d, \ldots, v_m)$ be the corresponding linear function. Let $\mathbf{a}_1, \ldots, \mathbf{a}_{d-1} \in \{0,1\}^{n+m}$ be l.i. with $\mathrm{supp}(\mathbf{a}_j)$ among the indices of $v_1, \ldots, v_{d-1}$; and let $\mathbf{b}_j$ be the restriction of $\mathbf{a}_j$ to the last m bits. Then

$$\begin{aligned} g(K, v_d, \ldots, v_m) &= \bigoplus_{\mathbf{c} \in C[\mathbf{a}_1, \ldots, \mathbf{a}_{d-1}]} f((K, \mathrm{IV}) \oplus \mathbf{c}) \\ &= \bigoplus_{\mathbf{d} \in C[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]} f(K, \mathrm{IV} \oplus \mathbf{b}). \end{aligned}$$

Obtaining the outputs of f on 2^{d-1} chosen IVs gives the value of $g(K, 0, \ldots, 0)$ for the unknown K.

Attacking Stream Ciphers With IV: On-line

Suppose $v_1 \cdots v_{d-1}$ be a maxterm and $g(K, v_d, \ldots, v_m)$ be the corresponding linear function. Let $\mathbf{a}_1, \ldots, \mathbf{a}_{d-1} \in \{0,1\}^{n+m}$ be l.i. with $\mathrm{supp}(\mathbf{a}_j)$ among the indices of $v_1, \ldots, v_{d-1}$; and let $\mathbf{b}_j$ be the restriction of $\mathbf{a}_j$ to the last m bits. Then

$$\begin{aligned} g(K, v_d, \ldots, v_m) &= \bigoplus_{\mathbf{c} \in C[\mathbf{a}_1, \ldots, \mathbf{a}_{d-1}]} f((K, \mathrm{IV}) \oplus \mathbf{c}) \\ &= \bigoplus_{\mathbf{d} \in C[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]} f(K, \mathrm{IV} \oplus \mathbf{b}). \end{aligned}$$

Obtaining the outputs of f on 2^{d-1} chosen IVs gives the value of $g(K, 0, \ldots, 0)$ for the unknown K.
Obtain the values of $g_1(K, 0, \ldots, 0), \ldots, g_n(K, 0, \ldots, 0)$. Use the previously computed A^{-1} to solve the system of linear equations and obtain the secret key K.

Feasibility and Computational Complexity

- Exponential in d in both the pre-processing and the online phases.
    - Works well when d is small.

Feasibility and Computational Complexity

- Exponential in d in both the pre-processing and the online phases.
  - Works well when d is small.
- Polynomial in n in both the pre-processing and the online phases;
  - pre-processing: $O(n^3)$ to compute A^{-1}.
  - on-line: $O(n^2)$ to solve using A^{-1}.

Feasibility and Computational Complexity

- Exponential in d in both the pre-processing and the online phases.
  - Works well when d is small.
- Polynomial in n in both the pre-processing and the online phases;
  - pre-processing: $O(n^3)$ to compute A^{-1}.
  - on-line: $O(n^2)$ to solve using A^{-1}.
- Variants of the attack have been proposed.

Some References: Differential Attacks

- X. Lai. Higher Order Derivatives and Differential Cryptanalysis. Communications and Cryptography, 1992.
- A.Canteaut, M. Videau: Degree of Composition of Highly Nonlinear Functions and Applications to Higher Order Differential Cryptanalysis. EUROCRYPT 2002.

Some References: Differential Attacks

- X. Lai. Higher Order Derivatives and Differential Cryptanalysis. Communications and Cryptography, 1992.
- A.Canteaut, M. Videau: Degree of Composition of Highly Nonlinear Functions and Applications to Higher Order Differential Cryptanalysis. EUROCRYPT 2002.
- I. Dinur, A. Shamir: Cube Attacks on Tweakable Black Box Polynomials. EUROCRYPT 2009.
- J.-P. Aumasson, I. Dinur, W. Meier, A. Shamir: Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium.
- I. Dinur, A. Shamir: Breaking Grain-128 with Dynamic Cube Attacks. FSE 2011.

Time/Memory Trade-Off Attacks

Inverting a One-Way Function

Let S be a finite set with $\#S = N$ and

$$f : S \to S$$

be a one-way function.

Inverting a One-Way Function

Let S be a finite set with $\#S = N$ and

$$f : S \rightarrow S$$

be a one-way function.

Inversion problem: Given target y, find x such that $f(x) = y$.

Inverting a One-Way Function

Let S be a finite set with $\#S = N$ and

$$f : S \to S$$

be a one-way function.

Inversion problem: Given target y, find x such that $f(x) = y$.

- Memory N; time constant.
  - Pre-compute a table of all N pairs (x, y) such that $f(x) = y$.
  - Store the table sorted on the second column.
  - Given a target y_0, look up the table to find a pre-image.

Let S be a finite set with $\#S = N$ and

$$f : S \to S$$

be a one-way function.

Inversion problem: Given target y, find x such that $f(x) = y$.

- Memory N; time constant.
  - Pre-compute a table of all N pairs (x, y) such that $f(x) = y$.
  - Store the table sorted on the second column.
  - Given a target y_0, look up the table to find a pre-image.
- Memory constant; time N.
  - Given target y_0, compute $f(x)$ for each $x \in S$ until y_0 is obtained.

Time/Memory Trade-Off

$$f : S \to S.$$

Basic idea.

- Perform a one-time computation of N invocations of f.
- Store a table of size M.
- Given a particular target y_0, in time T obtain a pre-image.

$$f : S \rightarrow S.$$

Basic idea.

- Perform a one-time computation of N invocations of f.
- Store a table of size M.
- Given a particular target y_0, in time T obtain a pre-image.

Trade-Off Curve:

$$TM^2 = N^2.$$

$$f : S \to S.$$

Basic idea.

- Perform a one-time computation of N invocations of f.
- Store a table of size M.
- Given a particular target y_0, in time T obtain a pre-image.

Trade-Off Curve:

$$TM^2 = N^2.$$

A trade-off point: $T = M = N^{2/3}$.

Time/Memory Trade-Off

$$f : S \to S.$$

Basic idea.

- Perform a one-time computation of N invocations of f.
- Store a table of size M.
- Given a particular target y_0, in time T obtain a pre-image.

Trade-Off Curve:

$$TM^2 = N^2.$$

A trade-off point: $T = M = N^{2/3}$.
Pre-computation time is N which would make the attack inadmissible.

Multiple Targets/Data

$$f : S \to S.$$

Given: $y_1, \ldots, y_D$.
Goal: Invert *any one* of these points, i.e., obtain an x such that $f(x) = y_i$ for some i.

Multiple Targets/Data

$$f : S \rightarrow S.$$

Given: $y_1, \ldots, y_D$.
Goal: Invert *any one* of these points, i.e., obtain an x such that $f(x) = y_i$ for some i.
Modified Trade-Off Curve:

$$TM^2D^2 = N^2; 1 \leq D^2 \leq T; P = N/D.$$

Multiple Targets/Data

$$f : S \to S.$$

Given: $y_1, \ldots, y_D$.
Goal: Invert *any one* of these points, i.e., obtain an x such that $f(x) = y_i$ for some i.
Modified Trade-Off Curve:

$$TM^2D^2 = N^2; 1 \leq D^2 \leq T; P = N/D.$$

- Pre-computation time: $P = N/D$.
- Memory M and online time satisfy the equation $TM^2 = (N/D)^2$.
- A trade-off point: $D = N^{1/4}$; $P = N^{3/4}$; $T = M = N^{1/2}$.
- All the parameters D, P, T, M are less than N which makes the attack admissible.

TMTO on Stream Ciphers

Let $g(S)$ denote the keystream obtained by starting from state S.
Assume that the output function produces a single bit.

TMTO on Stream Ciphers

Let $g(S)$ denote the keystream obtained by starting from state S. Assume that the output function produces a single bit.

State-to-keystream map:

$$f : s\text{-bit state } S \mapsto s\text{-bit prefix of } g(S).$$

TMTO on Stream Ciphers

Let $g(S)$ denote the keystream obtained by starting from state S.
Assume that the output function produces a single bit.
State-to-keystream map:

$$f : s\text{-bit state } S \mapsto s\text{-bit prefix of } g(S).$$

Data: Given a keystream of length $D + s - 1$, shift a window of size s to construct D s-bit strings $y_1, \ldots, y_D$.
Inverting f on any of these targets gives an internal state.

TMTO on Stream Ciphers

Let $g(S)$ denote the keystream obtained by starting from state S.
Assume that the output function produces a single bit.
State-to-keystream map:

$$f : s\text{-bit state } S \mapsto s\text{-bit prefix of } g(S).$$

Data: Given a keystream of length $D + s - 1$, shift a window of size s to construct D s-bit strings $y_1, \ldots, y_D$.
Inverting f on any of these targets gives an internal state.

- Search space: $N = 2^s$.
- Trade-off point: $D = 2^{s/4}$; $P = 2^{3s/4}$; $T = M = 2^{s/2}$.

TMTO on Stream Ciphers

Let $g(S)$ denote the keystream obtained by starting from state S.
Assume that the output function produces a single bit.
State-to-keystream map:

$$f : s\text{-bit state } S \mapsto s\text{-bit prefix of } g(S).$$

Data: Given a keystream of length $D + s - 1$, shift a window of size s to construct D s-bit strings $y_1, \ldots, y_D$.
Inverting f on any of these targets gives an internal state.

- Search space: $N = 2^s$.
- Trade-off point: $D = 2^{s/4}$; $P = 2^{3s/4}$; $T = M = 2^{s/2}$.

- Suppose K is k bits long.
- If $s < 2k$, then $T = 2^{s/2} < 2^k$.
- *Ignoring pre-computation time*, this is an attack.
- **Counter-measure:** state size must be double that of secret key size.

TMTO on Stream Ciphers (contd.)

Consider a stream cipher with IV.
Suppose IVs are v bits long.
A one-way function:

$$f : (K, \mathrm{IV}) \mapsto (k + v)\text{-bit prefix of the keystream.}$$

TMTO on Stream Ciphers (contd.)

Consider a stream cipher with IV.
Suppose IVs are v bits long.
A one-way function:

$$f : (K, \mathrm{IV}) \mapsto (k + v)\text{-bit prefix of the keystream.}$$

- Search space: $N = 2^{k+v}$.
- Trade-off point: $D = 2^{(k+v)/4}$; $P = 2^{3(k+v)/4}$; $T = M = 2^{(k+v)/2}$.

TMTO on Stream Ciphers (contd.)

Consider a stream cipher with IV.
Suppose IVs are v bits long.
A one-way function:

$$f : (K, \mathrm{IV}) \mapsto (k + v)\text{-bit prefix of the keystream.}$$

- Search space: $N = 2^{k+v}$.
- Trade-off point: $D = 2^{(k+v)/4}$; $P = 2^{3(k+v)/4}$; $T = M = 2^{(k+v)/2}$.

- *Ignoring pre-computation time*, if $v < k$, then $T < 2^k$ and we have a valid attack.
- **Counter-measure:** IV should be at least as large as the key.

TMTO on Stream Ciphers (contd.)

Consider a stream cipher with IV.
Suppose IVs are v bits long.
A one-way function:

$$f : (K, \mathrm{IV}) \mapsto (k+v)\text{-bit prefix of the keystream.}$$

- Search space: $N = 2^{k+v}$.
- Trade-off point: $D = 2^{(k+v)/4}$; $P = 2^{3(k+v)/4}$; $T = M = 2^{(k+v)/2}$.

- *Ignoring pre-computation time*, if $v < k$, then $T < 2^k$ and we have a valid attack.
- **Counter-measure:** IV should be at least as large as the key.
- If $v < k/3$, then $P < 2^k$ and we have a valid attack *even considering pre-computation*.

Multi-User Setting

- A secure stream cipher will become popular and will be widely deployed.
- Users will choose random secret keys.
- Encryption will be done using the secret key and an IV.
- Restriction on the IV: should not be repeated for the same key.
- To obtain higher security, a user may choose a secret key for each session.
  - Each message in a session would be encrypted using a distinct IV.
  - Same restriction: do not repeat IV for the same key.

Multi-User (In)security

Set IV to a fixed value v and define the map

$$f : K \to \text{ first } k \text{ bits of } \mathrm{SC}_K(v).$$

Multi-User (In)security

Set IV to a fixed value v and define the map

$$f : K \to \text{ first } k \text{ bits of } \mathrm{SC}_K(v).$$

- Suppose $k = 80$: Get 2^{20} users to encrypt messages using the same IV and obtain the first 80 bits of the keystream.
  - No violation of IV usage; same IV used, but, for *different* keys.
  - This gives 2^{20} targets.

Multi-User (In)security

Set IV to a fixed value v and define the map

$$f : K \rightarrow \text{ first } k \text{ bits of } \mathrm{SC}_K(v).$$

- Suppose $k = 80$: Get 2^{20} users to encrypt messages using the same IV and obtain the first 80 bits of the keystream.
  - No violation of IV usage; same IV used, but, for *different* keys.
  - This gives 2^{20} targets.
- Inverting f on any one of these targets will give the corresponding secret key.
- Trade-off point: $P = 2^{60}$; $D = 2^{20}$; $T = M = 2^{40}$.
- A very realistic attack: 80-bit security is inadequate!

Multi-User (In)security

Set IV to a fixed value v and define the map

$$f : K \rightarrow \text{ first } k \text{ bits of } \mathrm{SC}_K(v).$$

- Suppose $k = 80$: Get 2^{20} users to encrypt messages using the same IV and obtain the first 80 bits of the keystream.
  - No violation of IV usage; same IV used, but, for *different* keys.
  - This gives 2^{20} targets.
- Inverting f on any one of these targets will give the corresponding secret key.
- Trade-off point: $P = 2^{60}$; $D = 2^{20}$; $T = M = 2^{40}$.
- A very realistic attack: 80-bit security is inadequate!
  - No counter-measures; using random IVs may actually make it easier for the attacker to mount the attack.

Multi-User (In)security

Set IV to a fixed value v and define the map

$$f : K \rightarrow \text{ first } k \text{ bits of } \mathrm{SC}_K(v).$$

- Suppose $k = 80$: Get 2^{20} users to encrypt messages using the same IV and obtain the first 80 bits of the keystream.
  - No violation of IV usage; same IV used, but, for *different* keys.
  - This gives 2^{20} targets.
- Inverting f on any one of these targets will give the corresponding secret key.
- Trade-off point: $P = 2^{60}$; $D = 2^{20}$; $T = M = 2^{40}$.
- A very realistic attack: 80-bit security is inadequate!
  - No counter-measures; using random IVs may actually make it easier for the attacker to mount the attack.
- Works for all k; but, the effect is less dramatic.

Some References: TMTO Attacks

- M. E. Hellman: A Cryptanalytic Time-Memory Trade-Off. IEEE Trans. on Infor. Th., 26 (1980).

Some References: TMTO Attacks

- M. E. Hellman: A Cryptanalytic Time-Memory Trade-Off. IEEE Trans. on Infor. Th., 26 (1980).
- S. H. Babbage: Improved Exhaustive Search Attacks on Stream Ciphers. European Convention on Security and Detection, IEE Conference publication No. 408, IEE, 1995.
- J. Dj. Golic: Cryptanalysis of Alleged A5 Stream Cipher. EUROCRYPT 1997.
- Alex Biryukov, Adi Shamir: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. ASIACRYPT 2000.

Some References: TMTO Attacks

- M. E. Hellman: A Cryptanalytic Time-Memory Trade-Off. IEEE Trans. on Infor. Th., 26 (1980).
- S. H. Babbage: Improved Exhaustive Search Attacks on Stream Ciphers. European Convention on Security and Detection, IEE Conference publication No. 408, IEE, 1995.
- J. Dj. Golic: Cryptanalysis of Alleged A5 Stream Cipher. EUROCRYPT 1997.
- Alex Biryukov, Adi Shamir: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. ASIACRYPT 2000.
- J. Hong, P. Sarkar: New Applications of Time Memory Data Tradeoffs. ASIACRYPT 2005.
- S. Chatterjee, A. Menezes and P. Sarkar. Another Look at Tightness. SAC 2011, to appear.

Summary

- A brief background on stream ciphers.
  - Additive and self-synchornizing stream ciphers.
  - Attack models and goals.
  - Block cipher modes of operations.
  - LFSR and non-linear combiner model.
- Correlation Attacks.
- Algebraic Attacks.
- Chosen IV differential attacks.
- (In)security in the Multi-User Setting.
  - TMTO attacks on stream ciphers.
  - Inadequacy of 80-bit security in the multi-user setting.

Summary

- A brief background on stream ciphers.
  - Additive and self-synchornizing stream ciphers.
  - Attack models and goals.
  - Block cipher modes of operations.
  - LFSR and non-linear combiner model.
- Correlation Attacks.
- Algebraic Attacks.
- Chosen IV differential attacks.
- (In)security in the Multi-User Setting.
  - TMTO attacks on stream ciphers.
  - Inadequacy of 80-bit security in the multi-user setting.

We have left out a lot of topics including some *important* ones.

Thank you for your attention!