Fundusze Europejskie Wiedza Edukacja Rozwój
Politechnika Wrocławska
Unia Europejska Europejski Fundusz Społeczny

"ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej"

Cryptography

(from algorithmic perspective)

Filip Zagórski

DRAFT: I April 12, 2019

Preface

This lecture notes were written to support students of the course *Cryptography*, a lecture taught at the Department of Computer Science, Wroclaw University of Science and Technology.

The lecture notes are in part inspired by the following books:

(1) Introduction to Modern Cryptography [**KL14**] by Jonathan Katz and Yehuda Lindell

(2) Introduction to Cryptography [**Buc13**] by Johannes A. Buchmann

Contents

Part 1

Secret key cryptography

Chapter 1

Information theoretic secrecy

1.1. Perfect secrecy

Definition 1.1 (Perfect secrecy)**.** *An encryption scheme* $\pi = \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ *over a message space* $\mathcal{M}$ *is* ***perfectly secret*** *if for every probability distribution over* $\mathcal{M}$*, every messsage* $m \in \mathcal{M}$*, and every ciphertext* $c \in \mathcal{C}$ *for which* $P[C = c] > 0$*:*

$$P[M = m | C = c] = P[M = m].$$

Definition 1.2. *An encryption scheme* $\pi = \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ *over a message space* $\mathcal{M}$ *is* ***perfectly secret*** *if and only if for every probability distribution over* $\mathcal{M}$*, every message* $m \in \mathcal{M}$*, and every ciphertext* $c \in \mathcal{C}$*:*

$$P[C = c | M = m] = P[C = c].$$

Definition 1.3 (Perfect indistinguishability)**.** *An encryption scheme* $\pi = \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ *over a message space* $\mathcal{M}$ *is* ***perfectly secret*** *if for every probability distribution over* $\mathcal{M}$*, every* $m_0, m_1 \in \mathcal{M}$*, and every* $c \in \mathcal{C}$*:*

$$P[C = c | M = m_0] = P[C = c | M = m_1].$$

When a given encryption scheme π satisfies definition 1.3, we say that it has **perfectly indistinguishabile encryptions**.

Now, we use the last definition and express it in a different form. To do so, we first define the eavesdroping experiment (game) $\mathsf{PrivK}^{eav}_{\pi,\mathcal{A}}$. The game is played between a player $\mathcal{A}$, called an adversary, and an entity which lets $\mathcal{A}$ interact with the encryption scheme π. (For now the interaction is very limited: an adversary passively waits to see a single ciphertext. A more realistic definiotions will be presented in next chapters).

Definition 1.4 (Eavesdroping experiment)**.** $\mathsf{PrivK}^{eav}_{\mathcal{A},\pi}$

(1) *A secret key is generated* $k \leftarrow \mathsf{Gen}$.

(2) $\mathcal{A}$ *generates two messages* m_0, m_1 *of equal length.*

(3) *A random bit* $b \leftarrow \{0,1\}$ *is picked, and* $\mathcal{A}$ *obtains the encryption* $c \leftarrow \mathsf{Enc}(k, m_b)$.

(4) $\mathcal{A}$ *outputs a bit* b'.

(5) *The outcome of the experiment is equal to* 1 *if* $b = b'$*, and* 0 *otherwise.*

Definition 1.5. *A private key encryption scheme* $\pi = \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ *has indistinguishable encryptions in the presence of an eavesdropper if for any adversary* $\mathcal{A}$

$$P[\mathsf{PrivK}^{eav}_{\mathcal{A},\pi} = 1] = \frac{1}{2}.$$

Although the word perfect is used, schemes that satisfy that definitions are usually not used in practice. The reason for that comes from the following limitations:

(1) a secret key must be at least as long as the message,

(2) a secret key must to be used only once.

Problems

1. Prove or refute: definitions 1.1 and 1.2 are equivalent.
2. Prove or refute: definitions 2 and 3 are equivalent.
3. Prove or refute: definitions 1 and 3 are equivalent.
4. Prove or refute: For every encryption scheme that is perfectly secret it holds that for every distribution over the message space $\mathcal{M}$, every $m, m' \in \mathcal{M}$, and every $c \in \mathcal{C}$:

$$P[M = m | C = c] = P[M = m' | C = c].$$

5. Consider the following definition of perfect secrecy for the encryption of two messages. An encryption scheme $\langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ over a message space $\mathcal{M}$ is perfectly-secret for two messages if for all distributions over $\mathcal{M}$, all $m_0, m_1 \in \mathcal{M}$, and all $c_0, c_1 \in \mathcal{C}$ with $P(C_0 = c_0 \wedge C_1 = c_1) > 0$:

$$P(M_0 = m_0 \wedge M_1 = m_1 | C_0 = c_0 \wedge C_1 = c_1) = P(M_0 = m_0 \wedge M_1 = m_1),$$

   where m_0 and m_1 are sampled independently from the same distribution over $\mathcal{M}$.

   Prove that no encryption scheme satisfies this definition (hint: take $m_0 \neq m_1$ but $c_0 = c_1$).

6. (a) Prove that the *shift cipher* is perfectly secure if only a single character is encrypted.
   (b) Prove that One Time Pad is perfectly secure.

Chapter 2

Pseudorandom objects

2.1. Introduction

In this section we turn our attention into the schemes that are (only) computationally secure (in the previous section we were considering information-theoretic security).

Definition 2.1. *A function $f \geq 0$ is* negligible *if for every polynomial $p(\cdot) > 0$ there exists an $N > 0$ such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.*

Now, we redefine the eavesdropping experiment, to take into account size (n) of the security parameter.

Definition 2.2 (Eavesdroping experiment)**.** $\mathsf{PrivK}^{eav}_{\mathcal{A},\pi}(n)$

(1) *A secret key is generated $k \leftarrow \mathsf{Gen}(1^n)$.*

(2) *$\mathcal{A}$ generates two messages m_0, m_1 of equal length.*

(3) *A random bit $b \leftarrow \{0,1\}$ is picked, and $\mathcal{A}$ obtains the encryption $c \leftarrow \mathsf{Enc}(k, m_b)$.*

(4) *$\mathcal{A}$ outputs a bit b'.*

(5) *The outcome of the experiment is equal to 1 if $b = b'$, and 0 otherwise.*

From now on, we will limit computational power of an adversary. We will be only interested in adversaries who run in polynomial time.

Definition 2.3. *A private-key encryption scheme $\pi = \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ has* indistinguishable encryptions in the presence of an eavesdropper *if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function* negl *such that*

$$P[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n),$$

where the probability is taken over the random coins used by $\mathcal{A}$, as well as the random coins used in the experiment (for choosing the key k, ther random bit b, and any random coins used in the encryption process.

2.2. Pseudorandom number generators (PRNG)

Definition 2.4. *Let $l(\cdot)$ be a polynomial and let G be a deterministic polynomial-time algorithm such that for any input $s \in \{0,1\}^n$, algorithm G outputs a string of length $l(n)$. G is a pseudorandom generator if the following conditions hold:*

(1) *(Expansion:) For every n it holds that $l(n) > n$.*

(2) *(Pseudorandomness:) For all PPT (polynomial-time distinguishers) $\mathcal{D}$, there exists a negligible function* negl *such that:*

$$|P[\mathcal{D}(r) = 1] - P[D(G(s)) = 1]| \leq \mathsf{negl}(n),$$

where r is chosen uniformly at random from $\{0,1\}^{l(n)}$, the seed s is chosen uniformly at random from $\{0,1\}^n$, and the probabilities are taken over the random coins used by $\mathcal{D}$ and the choce of r and s.

Definition 2.5. *A generator $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ is* predictable *if there exists an efficient (probabilistic polynomial time) algorithm $\mathcal{A}$ and such an $i : 1 < i < l(n)$ that:*

$$P\left[\mathcal{A}(G(x)_{1\ldots i}) = G(x)_{i+1}\right] > \frac{1}{2} + \varepsilon(n)$$

for a non-negligible function $\varepsilon(n)$.

Definition 2.6. *Let G be a pseudorandom generator with expansion factor l. Define a private-key encryption scheme for messages of length l as follows:*

Gen: *on input 1^n, choose uniformly at random a key $k \leftarrow \{0,1\}^n$*

Enc: *on input: a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{l(n)}$, output the ciphertext $c = \mathsf{Enc}_k(m) = G(k) \oplus m$.*

Dec: *on input: a key* $k \in \{0,1\}^n$ *and a ciphertext* $c \in \{0,1\}^{l(n)}$*, output the plaintext message* $m = \mathsf{Dec}_k(c) = G(k) \oplus c$.

Theorem 2.1. *If* G *is a pseudorandom generator, then the construction presented in Definition 2.5 is a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.*

Problems

1. Let f, g be negligible functions. Show that:
   (a) The function $h(n) = f(n) + g(n)$ is negligible .
   (b) For any positive polynomial p, the function $h(n) = p(n) \cdot f(n)$ is negligible .

2. Show that if G is a pseudorandom generator then G is unpredictable.

3. Let $\mathsf{G} : \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudorandom generator. Design a computationally unbounded distinguisher $\mathcal{D}$ which predicts next bits of G's output with non-negligible advantage.

4. Let G be a pseudorandom generator where $|G(s)| > 2|s|$.
   (a) Define $G'(s) = G(s0^{|s|})$. Is G' necessarily a pseudorandom generator?
   (b) Define $G'(s) = G(s_1 \dots s_{n/2})$, where $s = s_1 \dots s_n$. Is G' necessarily a pseudorandom generator?

5. Let G be a pseudorandom generator and define $G'(s)$ to be the output of G truncated to n bits (where $|s| = n$). Prove that the function $F_k(x) = G'(k) \oplus x$ is not pseudorandom.

6. Consider the following LFSR over $\mathcal{Z}_2$: $z_{i+4} = z_i + z_{i+1} + z_{i+2} + z_{i+3}$ mod 2, for $i \geq 0$. (a) Draw the corresponding LFSR. (b) For all possible vectors (z_0, z_1, z_2, z_3) find a period of the LFSR (period of an i-bit LFSR is the smallest $n > 0$, for which $(z_0, \dots, z_{i-1}) = (z_n, ldots, z_{n+i-1})$). (c) Repeat the exercise for $z_{i+4} = z_i + z_{i+3}$ mod 2.

7. Output of an LFSR can be written in a matrix form:

$$(z_{m+1}, z_{m+2}, \ldots, z_{2m}) = (c_1, c_2, \ldots, c_m) \begin{pmatrix} z_1 & z_2 & z_3 & \cdots & z_m \\ z_2 & z_3 & z_4 & \cdots & z_{m+1} \\ \vdots & \vdots & & & \vdots \\ z_m & z_{m+1} & z_{m+2} & \cdots & z_{2m-1} \end{pmatrix},$$

where $z = \langle z_1, \ldots, z_m \rangle$ corresponds to the initial content of the LFSR and $c = \langle c_1, \ldots, c_m \rangle$ defines which registers are used to compute the next bit.

For an LFRS defined by: $c = \langle 0, 1, 0, 0, 1 \rangle$, a key $z = \langle 0, 1, 0, 1, 1 \rangle$ (initial state) one obtains:

$$(0, 0, 1, 0, 0) = (0, 1, 0, 0, 1) \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Knowing that $c = \langle 1, 0, 0, 1, 0 \rangle$ and $\langle z_6, z_7, z8, z_9, z_{10} \rangle = \langle 0, 0, 1, 1, 0 \rangle$ find the key $z = \langle z_1, z_2, z_3, z_4, z_5 \rangle$.

8. What is the complexity of an attack on A5/1 if in each round all LFSRs are moving (instead of applying a majority rule).

9. Let $G : \{0,1\}^s \to \{0,1\}^n$ be a pseudorandom generator. Which of the following generators are also pseudorandom?
   (a) $G'(x) = G(0)$
   (b) $G'(x) = G(x)||G(x)$
   (c) $G'(x) = G(x)||0$.
   (d) $G'(x) = G(x)_{0,\ldots,n-2}$ ($G'(x)$ takes as its output first $n-1$ bits of $G(x)$)
   (e) $G'(x) = G(k) \oplus 1^n$.

   For each case: prove or refute pseudorandomnes of G'.

10. Show how to construct a variable output-length pseudorandom generator from any pseudorandom function.

11. Let G be a pseudorandom generator and define $G'(s) = G(s)_{1\ldots n}$ (output truncated to the first n bits) for $|s| = n$. Prove that the function $F_k(x) = G'(k) \oplus x$ is not pseudorandom.

12. Let F be a pseudorandom function, and G a pseudorandom generator with expansion factor $l(n) = n+1$. For each of the following encryption schemes, state whether the scheme has indistinguishable encryptions in the presence of an eavesdropper and whether it is CPA-secure. In each case, the shared key is a random $k \in \{0,1\}^n$.
    (a) To encrypt $m \in \{0,1\}^{2n+2}$, parse m as $m_1||m_2$ with $|m_1| = |m_2|$ and send
    $\langle G(k) \oplus m_1, G(k+1) \oplus m_2 \rangle$.
    (b) To encrypt $m \in \{0,1\}^{n+1}$, choose a random $r \leftarrow \{0,1\}^n$ and send $\langle r, G(r) \oplus m \rangle$.
    (c) To encrypt $m \in \{0,1\}^n$, send $m \oplus F_k(0^n)$.
    (d) To encrypt $m \in \{0,1\}^{2n}$, parse m as $m_1||m_2$ with $|m_1| = |m_2|$, then choose $r \leftarrow \{0,1\}^n$ at random, and send $\langle r, m_1 \oplus F_k(r), m_2 \oplus F_k(r+1) \rangle$.

Chapter 3

Computational secrecy

3.1. Chosen plaintext attack

In Chapter 1 we defined notion of perfect secrecy and discussed its limitations (long single-use keys). Then, in Chapter 2 we switched to compuational secrecy. But there is also another problem – all previous definitions are very restrictive with respect to the capabilities of an adversary. In the definition of perfect secrecy, it is assumed that an adversary may only eavesdrop communication, moreover it is assumed that each ciphertext uses a different key.

In this section we remove aforementioned restrictions. But there is a price we pay for that, from now on, we only consider adversaries who are *probabilistic polynomial-time* (PPT) algorithms. On the other side, we will be able to model a larger spectrum of adversarial behaviour:

(1) adversary may be able to learn more about π:
   CPA: by being allowed to ask for encryptions of selected plaintext messages,
   CCA: by being allowed to ask for both: encryptions and decryptions of selected messages.

(2) adversary may be able to learn even more, by obtaining additional information *e.g.,* the time it takes to compute the output. (And so, we will be interested in leakage-resilient schemes.)

Definition 3.1 (CPA experiment)**.** $\mathit{PrivK}^{\mathit{cpa}}_{\mathcal{A},\pi}(n)$

(1) *A key is generated* $k \leftarrow \mathsf{Gen}(1^n)$.

(2) (a) *The adversary* $\mathcal{A}$ *on input* 1^n *is given access to the encryption oracle* $\mathsf{Enc}_k(\cdot)$. $\mathcal{A}$ *generates messages* x_i *and obtains ciphertexts* $c_i = \mathsf{Enc}_k(x_i)$*).*

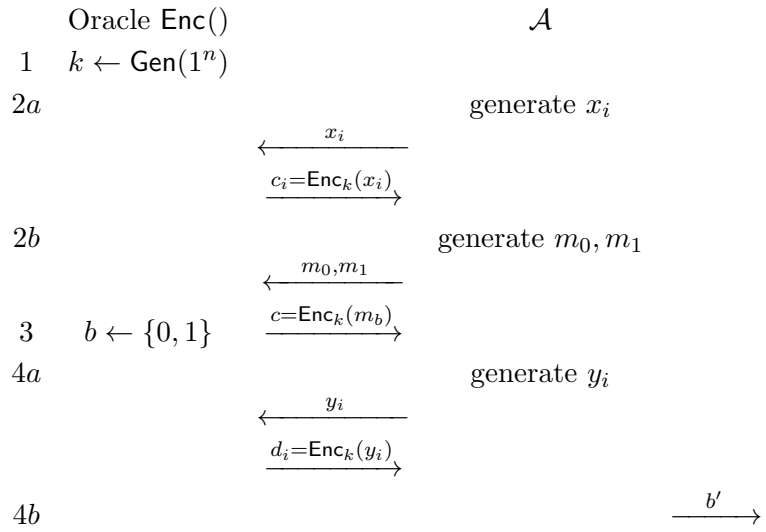
(b) $\mathcal{A}$ *outputs a pair of messages* m_0, m_1 *of equal length.*

(3) *A random bit* $b \leftarrow \{0,1\}$ *is selected, and the (challenge) ciphertext* $c \leftarrow \mathsf{Enc}_k(m_b)$ *is given to* $\mathcal{A}$.

(4) (a) *The adversary* $\mathcal{A}$ *continues to have oracle access to* $\mathsf{Enc}_k(\cdot)$.

(b) $\mathcal{A}$ *outputs a bit* b'.

(5) *The outcome of the experiment is* 1 *(we denote it by* $\mathit{PrivK}^{\mathit{cpa}}_{\mathcal{A},\pi}(n) = 1$*) if* $b = b'$ *and is* 0 *otherwise.*



Definition 3.2 (CPA security)**.** *A private-key encryption scheme* $\pi = \langle \mathsf{Gen}(,)\mathsf{Enc}(,)\mathsf{Dec}() \rangle$ *has indistinguishable encryptions under a chosen-ciphertext attack (is CPA-secure) if for all PPT (probabilistic polynomial-time) adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$P\left[\mathit{PrivK}^{\mathit{cpa}}_{\mathcal{A},\pi}(n) = 1\right] \leq \frac{1}{2} + \mathit{negl}(n).$$

The probability is taken over the random choices made by $\mathcal{A}$ *and random choices made during the* $\mathit{PrivK}^{\mathit{cpa}}_{\mathcal{A},\pi}(n)$ *experiment.*

Definition 3.3 (Counter-mode protoplast)**.** *Let* $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *be a pseudorandom function. We define a private-key encryption scheme* π_F *as follows:*

Gen: *on input* 1^n*, choose uniformly at random a key* $k \leftarrow \{0,1\}^n$*.*

Enc: *on input a key* $k \in \{0,1\}^n$ *and a message* $m \in \{0,1\}^n$*, choose* $r \in \{0,1\}^n$ *uniformly at random and output the ciphertext:*

$$c = \mathsf{Enc}_k(m) = \langle r, F_k(r) \oplus m \rangle$$

Dec: *on input a key* $k \in \{0,1\}^n$ *and a ciphertext* $c = \langle r, s \rangle \in \{0,1\}^n \times \{0,1\}^n$ *output the plaintext:*

$$m = \mathsf{Dec}_k(c) = \mathsf{Dec}_k(\langle r, s \rangle) = F_k(r) \oplus s.$$

Theorem 3.1. *If* F *is a pseudorandom function then construction* π_F *presented as Definition 3.3 is a fixed-length private-key encryption scheme for messages of length* n *that hase indistinguishable encryptions under a chosen-plaintext attack.*

Problems

1. Let $\Pi_1 = \langle Gen_1, Enc_1, Dec_1 \rangle$, $\Pi_2 = \langle Gen_2, Enc_2, Dec_2 \rangle$ be the two private-key encryption schemes. Show how to construct Π – a *CPA-secure* private-key encryption scheme by combining schemes Π_1 and Π_2. You may assume that Π_i is *CPA*-secure but you do not know which one.

   Assuming that an adversary can win the *CPA* experiment with an advantage ϵ_i for the scheme Π_i, evaluate adversary's advantage for the scheme Π.

2. Show that there exist private-key encryption schemes that have indistinguishable encryptions in the presence of an eavesdropper but do not have indistinguishable multiple encryptions in the presence of an eavesdropper.

3. Prove that ECB mode of encryption does not yield CPA-secure encryption regardles of function F.

4. Consider a variant of CBC where in each encryption IV is increased by 1 (instead of choosing it at random). Show that the variant is not CPA-secure.

Chapter 4

Real-world constructions

4.1. RC4

RC4 [**Riv92**] was designed in 1986 by Ronald Rivest and it was the most popular stream cipher for years. RC4 was the only stream cipher used in WEP (Wired Equivalent Privacy) protocol for securing wireless networks in both standards: 802.11a and 802.11b. It was the default stream cipher in many versions of SSL/TLS protocols, it was estimated that around year 2013 about 85% of the network traffic was encrypted with this cipher. The history of partial breaks of the RC4 may serve as a warning for the future constructions. A series of seemingly innocent breaks [**Gol97, FM00, FMS01, MS01, Mir02**] has lead to the real-world threats [**ABP$^+$13, VP15**]. And not long after the latter two attacks were published, RC4 was abandoned and replaced with more secure alternatives.

RC4 scheme uses two algorithms $\mathsf{KSA}(\mathsf{N}, \mathsf{T})$ which takes a secret key K as an input, and outputs an array (permutation) S of size N. Algorithm $\mathsf{PRGA}(\mathsf{N})$ outputs pseudo-random bytes from S.

The default setting for RC4 is $N = T = 256$, and key-length between 40 and 2048 bits, but in this chapter we analyze its security for arbitrary values N and T. Our analysis follow results presented in [**Mir02, KLZ16**].

Algorithm 1: $\mathsf{KSA}_k(\mathsf{N},\mathsf{T})$

```
1 for i from 0 to N − 1 do
2 |   S[i] := i
3 end
4 j := 0;
5 for i from 0 to T do
6 |   j := (j + S[i mod N] +
  |     K[i mod L]) mod N;
7 |   swap(S[i mod N], S[j mod
  |     N]);
8 end
```

Algorithm 2: $\mathsf{PRGA}_S(\mathsf{N})$

```
1 i := 0;
2 j := 0;
3 while GeneratingOutput do
4 |   i := (i + 1) mod N;
5 |   j := (j + S[i]) mod N;
6 |   swap(S[i], S[j]);
7 |   Z := S[(S[i] + S[j]) mod N];
8 |   output  Z
9 end
```

$K[i]$ returns ith BYTE of the key k. L denotes length of the key in bits.

Original $\mathsf{RC4} = \mathsf{RC4}(\mathsf{N},\ \mathsf{T}) = \mathsf{RC4}(256,256)$ is:

(1) $S := \mathsf{KSA}_k(\mathsf{N},\ \mathsf{N})$

(2) $outputStream \leftarrow \mathsf{PRGA}_S(\mathsf{N})$

$\mathsf{RC4\text{-}RS}(\mathsf{N},\ \mathsf{T})$ is:

(1) $S := \mathsf{KSA\text{-}RS}_k(\mathsf{N},\mathsf{T})$

(2) $outputStream \leftarrow \mathsf{PRGA}_S(\mathsf{N})$

Function RC4-drop[D] drops first D bytes of PRGA output.

Let RC4-mdrop[D] denote RC4 which for each consecutive $D+1$ bytes generated by $PRNG$ it drops first D bytes before it outputs the next byte.

Function RC4-SST repeats the loop of KSA (lines 5-8 as long as SST marking is done, see [**KLZ16**] – it is StoppingRuleKLZ from page 15).

A closer look at KSA* reveals that it is actually so-called *cyclic to random transposition*. If we identify elements $[n]$ with cards then we do the following: at step t exchange card $t \bmod n$ with randomly chosen one. Let $X = \{X\}_{t\geq 0}$ denote the chain corresponding to this shuffling and let $\mathcal{L}(X_t)$ denote the distribution of the chain at time t.

4.1.1. Sign distinguisher for KSA in RC4.

Definition 4.1 (Sign of a permutation)**.** *Let $\sigma = (a_1b_1)\dots(a_kb_k)$ be an n-element permutation with k non-trivial transpositions ($a_i \neq b_i$ for $i =$*

$1, \ldots k$*). We define the sign (parity) of the permutation* σ *as* $sgn(\sigma) = (-1)^k$.

One can look at the sign-change process for the *cyclic to random transposition* as follows: after the table is initialized (with the identity permutation), the sign of S is equal to $+1$ since it is identity so the initial distribution is concentrated in $v_0 = (Pr(\text{sign}(X_0) = +1), Pr(\text{sign}(X_0) = -1)) = (1, 0)$.

Then in each step the sign is unchanged if and only if $i = j$ which happens with probability $1/n$. So the transition matrix P_n of the sign-change process induced by the shuffling process is equal to:

$$M_n := \begin{pmatrix} \frac{1}{n} & 1 - \frac{1}{n} \\ 1 - \frac{1}{n} & \frac{1}{n} \end{pmatrix}.$$

This conclusion corresponds to looking at the distribution of the sign-change process after t steps: $v_0 \cdot P_n^t$, where v_0 is the initial distribution.

$$v_0 \cdot M_n^t = \left(\frac{1}{2} + \frac{1}{2} \left(\frac{2}{n} - 1 \right)^t, \frac{1}{2} - \frac{1}{2} \left(\frac{2}{n} - 1 \right)^t \right).$$

Example 4.1. *Distinguisher's advantage*

For $n = 256$ *(which corresponds to the value of* n *used in RC4) and initial distribution being identity permutation after* $t = n = 256$ *steps one gets:* $v_0 \cdot M_{256}^{256} = (0.567138, 0.432862)$.

k	$+1$	-1	ϵ
0	*.5671382998250798*	*.4328617001749202*	$2^{-3.89672}$
256	*.509015*	*.490985*	$2^{-6.79344}$
512	*.5012105173235390*	*.4987894826764610*	$2^{-9.69016}$
1024	*.5000218258757580*	*.4999781741242420*	$2^{-15.4836}$
2048	*.5000000070953368*	*.4999999929046632*	$2^{-27.0705}$
4096	*.5000000000000007*	*.4999999999999993*	$2^{-50.2442}$
8192			$2^{-96.5918}$
11008			$2^{-128.456}$

Table 1. The advantage (ϵ) of Sign distinguisher of RC4 after discarding initial k bytes.

Problems

1. Construct an algorithm which predicts next bits of *linear congruencial generator*, use this algorithm to construct a distinguisher (statistical test) which can distinguish output generated by an instance of LCG from a random string.
2. Your goal is the same as in the previous problem , but here, the generator is *glibc*'s random().
3. Implement an attack on a modified version of A5/1 where in each round all LFSRs are moving.
4. Design and implement a ciphertext-only attack on a modified version of A5/1 where:
   - in each round all LFSRs are moving,
   - the ouptut is a XOR of the first and the second LFSR,
   - the output is computed only if the output of the third LFSR is equal to 1.
5. Implement an attack on a shrinking generator.
6. Implement algorithms $RC4$ and test the quality of generated random bits depending on the parameters:
   (a) RC4(N, N)-mdrop[D]
   (b) RC4(N, $2N \log N$)-mdrop[D]
   (c) RC4-SST(N)-mdrop[D]

   Repeat experiments for different values of $N = 16, 64, 256$ and for key-lengths: $40, 64, 128$ and for $D = 0, 1, 2, 3$ with

   For statistical tests use any of: TestU01, DieHard, Dieharder.

Part 2

Public key cryptography

Chapter 5

Public key cryptography

5.1. Group theory

In this section we present a series of group-theory definitions, facts and theorems that let you understand how and why RSA cryptosystem works.

Lemma 5.1. *Let $a \in \mathcal{Z}$ and let $b \in \mathcal{Z}_+$ then there exist exactly one pair of integer numbers q, r such that $a = qb + r$ and $0 \leq r < b$.*

Definition 5.1. *We denote by the* greatest common divisor of integers a, b the largest integer such that it divides both a and b: $\gcd(a, b) = \max\{c : c|a \wedge c|b\}$.

We say that a and b are *relatively prime* if $\gcd(a, b) = 1$ (we denote it by $a \perp b$).

Lemma 5.2. *Let a, b be positive integers, then there exists integers X, Y such that $Xa + Yb = \gcd(a, b)$. Moreover $\gcd(a, b)$ is the smallest integer that can be expressed in this way.*

Proof. Let $I = \{\widehat{X}a + \widehat{Y}b : \widehat{X}, \widehat{Y} \in \mathcal{Z}\}$. I is non empty since $a, b \in I$. Let d be the smallest positive integer of I. Let us observe two facts:

(1) d divides every element of I.

(2) d is the common divisor of a and b.

To prove (1): Let us take any $c \in I$, then we can express it as $c = X'a + Y'b$. Let $c = qd + r$, where $0 \leq r < d$. Then:

$$r = c - qd = X'a + Y'b - q(Xa + Yb) = (X' - qX)a + (Y' - Yq)b \in I$$

and there are two options:

if $r \neq 0$**:** then this contradicts the choice of d.

if $r = 0$**:** then $r|c$, so d divides every element of I

To prove (2): since $a \in I$ and $b \in I$ then $d|a$ and $d|b$, so d is a common divisor of a and b. Let us assume (towards contradiction) that there exists such $d' > d$ that d' is also a divisor of a and b, then $d'|Xa + Yb$ but this sum is equal to d and this means that $d'|d$ but $d' > d$.

□

Lemma 5.3. *If $c|ab$ and $\gcd(a, c) = 1$ then $c|b$. In particular, if p is prime and $p|ab$ then either $p|a$ or $p|b$.*

Proof. We know that $c|ab$, so there exists an integer α such that $\alpha c = ab$. From the fact that $\gcd(a, c) = 1$ and from Lemma 5.2 we know that there exist X, Y such that $1 = Xa + Yc$ so:

$$b = Xab + Ycb = X\alpha c + Ycb = c(X\alpha + Yb).$$

□

Lemma 5.4. *If $p|N$ and $q|N$ and $\gcd(p, q) = 1$ then $pq|N$.*

Proof. Let integers a, b be such that: $pa = N$ and $qb = N$. From the Lemma 5.2 there exists integers X, Y such that $Xp + Yq = 1$, then:

$$N = XpN + YqN = Xpqb + Ypaq = pq(Xb + Ya).$$

□

Definition 5.2. *Let a be an integer and N be a positive integer. We say that a is invertible modulo N if there exists such an integer b that $ab = 1 \bmod N$.*

Lemma 5.5. *Let a be an integer and N be a positive integer. a is invertible modulo N if and only if $\gcd(a, N) = 1$.*

Proof. Let a be invertible and b be its inverse ($a \neq 0$). Then from the definition, $ab = 1 \bmod N$, so there exists such an integer c that:

$$ab - 1 = cN \iff ab - cN = 1 \iff \gcd(a, N),$$

where the last equivalence comes from the Lemma 5.2. □

Definition 5.3. *A tuple $\langle G, \circ \rangle$ where G is a set and $\circ : G \times G \to G$ is called a group if the following conditions hold:*

(1) *(closure)* $\forall_{g,h \in G} g \circ h \in G$.

(2) *(existence of identity)* $\exists_{e \in G} \forall_{g \in G} e \circ g = g \circ e = g$.

(3) *(existence of inverse)* $\forall_{g \in G} \exists_{h \in G} g \circ h = e = h \circ g$.

(4) *(associativity)* $\forall_{g_1,g_2,g_3 \in G} (g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.

If G is finite, we call $|G|$ the order of the group. If for every pair of elements $g, h \in G$ $g \circ h = h \circ g$, we call such a group abelian.

To shorten notation, we will skip the group operation $\circ$ whenever it is clear form the context and we will write $a \circ b = ab$. Moreover, we will denote the group identity e by 1.

Lemma 5.6. *Let $a, b, c \in G$, if $ac = bc$ then $a = b$. In particular if $ac = a$ then c is the identity in G.*

Theorem 5.1. *Let G be a finite abelian group and let $m = |G|$ be the order of the group. Then for every element $g \in G : g^m = 1$.*

Proof. Let us fix an element $g \in G$. Let $g_1, g_2, \ldots, g_m$ be the elements of G. We claim that:

$$g_1 g_2 \ldots g_m = (gg_1)(gg_2) \ldots (gg_m).$$

If $gg_i = gg_j$ then from the Lemma 5.6 we know that $g_i = g_j$. From the fact that G is abelian:

$$g_1 \ldots g_m = g^m (g_1 \ldots g_m),$$

then again from the Lemma 5.6 we have that $g^m = 1$.

□

Lemma 5.7. *Let G be a finite group with order $m = |G| > 1$. Then for any $g \in G$ and for any integer i*

$$g^i = g^{[i \bmod m]}.$$

Proof. Let $i = qm + r$ where $0 \leq r < m$ (Lemma 5.1), so $r = [i \bmod m]$. Then, from the Theorem 5.1 we have the following:

$$g^i = g^{qm+r} = g^{qm} \cdot g^r = (g^m)^q g^r = g^r.$$

□

The simple consequence of the last Lemma 5.7 is the following Corollary 5.2, called Fermat's little theorem.

Corollary 5.2. *[Fermat's little theorem] Let p be prime and let a be an integer. Then*

$$a^{p-1} = 1 \bmod p.$$

Proof. Let $G = \langle Z_p^*, * \rangle$, where $Z_p^* = \{a : a \in \mathcal{Z}_+, a < p, \gcd(a, p) = 1\}$, where $*$ operation is multiplication modulo p. To prove the statement we just need to observe that the order of G is equal to the number of integers that are co-prime with p, so the order of $|G| = m = p - 1$ which together with Lemma 5.7 ends the proof. □

Definition 5.4 (Euler's totient function)**.** *Let*

$$\varphi(n) = |\{a : a \in \mathcal{Z}_+, a < n, \gcd(a, n) = 1\}|,$$

we call $\varphi(n)$ Euler's totient function and corresponds to the number of positive integers smaller than n that are relatively prime with n.

Corollary 5.3. *[Euler's theorem] If $a, n \in \mathcal{Z}_+$ are co-prime then*

$$a^{\varphi(n)} = 1 \bmod n$$

5.2. RSA

Lemma 5.8. *Let G be a finite group with order $m = |G| > 1$. Let $e > 0$ be an integer. Let f_e be the function: $f_e : G \to G$ defined by $f_e(g) = g^e$. If $\gcd(e, m) = 1$ then f_e is a permutation (bijection) on G. Moreover if $d = [e^{-1} \bmod m]$ then f_d is the inverse of f_e.*

Proof. From the Lemma 5.5 if $\gcd(e, m) = 1$ we have that e is invertible modulo m. Let d be the modular inverse of e modulo m then, for any $g \in G$:

$$f_d(f_e(g)) = f_d(g^e) = (g^e)^d = g^{ed} = g^{[ed \bmod m]} = g^1 = g.$$

□

Example 5.4. *Let $p = 3$ and $q = 11$ and $N = pq = 33$. Let us define a group $G = \{x \in Z_+ : \gcd(x, N) = 1\}$.*

The order of $m = |G| = \varphi(N) = (p-1)(q-1) = 20$.

$G = \{1, 2, 4, 5, 7, 8, 10, 13, 14, 16, 17, 19, 20, 23, 25, 26, 28, 29, 31, 32\}$.

Example 5.5. *Let us continue with the previous example, let us set $e = 3$ as in Lemma 5.8. Let us now try to find the corresponding d: $ed = 1 \bmod m$. In order to do that, let us note that the inverse exists iff and only if $\gcd(e, m) = 1$ (Lemma 5.5). On the other hand, from the Lemma 5.2 we know that there exist integers X, Y such that*

$$Xe + Ym = \gcd(e, m) = 1.$$

So that means that

$$Xe = 1 \bmod m,$$

so we can find modular inverse of e modulo m by computing the extended Euclidean algorihm. Here

$$\{1, \{7, -1\}\} = \mathsf{ExtendedGCD}(3, 20) = \mathsf{ExtendedGCD}(e, m),$$

So $d = 7$. If we define $f_e[x\_] := \mathsf{PowerMod}[x, e, N]$ and $f_d[x] := \mathsf{PowerMod}[x, d, N]$ and apply this function to the set G, we obtain:

$G_e = Map[fe, G] = \{1, 8, 31, 26, 13, 17, 10, 19, 5, 4, 29, 28, 14, 23, 16, 20, 7, 2, 25, 32\}$.

And then we obtain the original set G by applying f_d to G_e: $G = Map[f_d, G_e]$.

5.2.1. Textbook RSA.

Definition 5.5 (Textbook RSA)**.**

$\mathsf{Gen}(1^n)$**:**

(1) *generate two primes p, q of equal length such that $N = pq$ and N is an n-bit integer.*
(2) *generate at random e such that $\gcd(e, \varphi(N)) = 1$.*
(3) *compute e's modular inverse d modulo $\varphi(N)$ ($ed = 1 \bmod \varphi(N)$).*
(4) *return a pair of public/private keys $\langle K_{pub}, K_{priv} \rangle$.*
   K_{pub}**:** *– the public key: $\langle N, e \rangle$.*
   K_{priv}**:** *– the private key: $\langle N, d \rangle$.*

$\mathsf{Enc}(K_{pub}, x)$**:** *for a message* $x \in Z_N^*$ *return a ciphertext:*

$$c := \mathsf{Enc}(K_{pub}, x) = \mathsf{Enc}_{K_{pub}}(x) = \mathsf{Enc}(\langle N, e\rangle, x) = x^e \bmod N.$$

$\mathsf{Dec}(K_{priv}, c)$**:** *for a ciphertext* $c \in Z_N^*$ *return a plaintext:*

$$x := \mathsf{Dec}(K_{priv}, c) = \mathsf{Dec}_{K_{priv}}(c) = \mathsf{Dec}_{\langle N,d\rangle}(c) = c^d \bmod N.$$

Problems

1. Show that *Textbook RSA* is not CPA-secure.
2. Show how one can decipher *RSA OAEP*.
3. Let $N = pq$ and let $[N, e_1], [N, e_2]$ be public keys of Alice and Bob respectively. Show that if Eve sends encrypted messages to Alice $c_1 = m^{e_1} \bmod N$ and Bob $c_2 = m^{e_2} \bmod N$ and you intercept them then you can recover m from c_1 and c_2. What is the success probability of your attack?.
4. Let $N = pq$ be a product of two distinct primes. Show that if $\phi(N)$ and N are known, then it is possible to compute p and q in polynomial time.
5. Let $N = pq$ be a product of two distinct primes. Show that if N and an integer d such that $3d = 1 \bmod \phi(N)$ are known, then it is possible to compute p and q in polynomial time.
6. Let $|N|$ denotes the length of binary representation of N.
   (a) Show that if $N = M^e$ for some integers $M, e > 1$ then $e \leq |N| + 1$.
   (b) Given N and e with $2 \leq e \leq |N| + 1$, show how to determine in $poly(|N|)$ time whether there exists an integer M with $N = M^e$.
   (c) Given N, show how to let test in $poly(|N|)$ time whether N is a perfect power.

Chapter 6

Factoring algorithms

In this section we discuss the hardness of integer factorization (factoring). It is conjectured that on a classical computers this problem is infeasible. While for quantum computers, Shor presented an algorithm [**Sho99**] which runs in polynomial-time.

It is also important to note that a given problem may be easy to be solved if one generates its instances in "wrong" way *e.g.,* selects parameters from a wrong sub-space.

Definition 6.1 (Factoring experiment $\mathsf{Factor}_{\mathcal{S},\mathcal{A}}(n)$)**.**

(1) *Select at random elements* $p, q \in \mathcal{S}(n)$*. Compute* $N = pq$*.*

(2) $\mathcal{A}$ *is given* N *and outputs* p', q'*.*

(3) *The output of the experiment is defined to be* 1 *if* $N = p'q'$ *(and* $p', q' > 1$*) and* 0 *otherwise.*

Example 6.1. *Let* $\mathcal{S} = \mathbf{Z}$ *be the set of integers, then* $\mathcal{S}(n)$ *denotes the set of* n*-bit integers. It is easy to see that* $P[\mathsf{Factor}_{\mathbf{Z},\mathcal{A}}(n) = 1] \geq \frac{3}{4}$ *for the following simple adversary* $\mathcal{A}(N)$*:*

(1) *if* $2|N$
then: $p' = 2$, $q' = N/2$
else: $p' = 1$, $q' = N$

(2) *return* $\langle p', q' \rangle$

On the other hand, it is believed that the factoring experiment is "hard" for the case when one selects factors from the set of prime numbers.

We describe two classical algorithms for finding integer factorization of an integer n: *Dixon's random squares method* [**Dix81**] and *Quadratic Sieve*.

The best factoring algorithms are sub-exponential, to compare them, the following notation is used:

Definition 6.2. *Let n, u, v be real numbers, and let $n > e$. Then*

$$L_n[u, v] = e^{v(\log n)^u (\log\log n)^{1-u}}.$$

6.1. Dixon's random squares method

We want to factor an integer n. The idea behind this algorithm is to find two numbers x, y such that

$$x^2 = y^2 \bmod n,$$

but at the same time $x \neq \pm y \bmod n$. In order to find such numbers, the algorithm sets a bound B and then limits it attention only to those numbers x which have only "small" prime factors.

Definition 6.3 (Factor base)**.** *Let B be a positive number, we call a set*

$$\mathcal{F}(B) = \{p \in \mathsf{Primes} : p \leq B\}$$

a factor base B. Moreover we call integers x for which each of its prime factor is in $\mathcal{F}(B)$ B-smooth.

Example 6.2. *Let $n = 2\ 058\ 769$. Let $B = 19$, then $\mathcal{F}(B) = \{2, 3, 5, 7, 11, 13, 17, 19\}$. By $k = |\mathcal{F}(B)| = 8$.*

The algorithm collects a set of k numbers z_i such that $z_i^2 \bmod n$ is B-smooth. Let's say that the following numbers were selected:

$$\begin{aligned}
456 &= 2^3 3^1 19^1 \\
9075 &= 3^1 5^2 11^2 \\
81600 &= 2^6 3^1 5^2 17^1 \\
176256 &= 2^7 3^4 17^1 \\
230400 &= 2^1 03^2 5^2 \\
279072 &= 2^5 3^3 17^1 19^1 \\
337535 &= 5^1 11^1 17^1 19^2 \\
371712 &= 2^1 03^1 11^2 \\
459800 &= 2^3 5^2 11^2 19^1 \\
834632 &= 2^3 17^2 19^2 \\
961875 &= 3^4 5^4 19^1 \\
1184832 &= 2^6 3^2 11^2 17^1
\end{aligned}$$

Now, we may observe that:

$$x = 456 \cdot 279\ 072 \cdot 1\ 184\ 832 \bmod n = 823\ 404,$$

and the corresponding right-hand sides:

$$y = (2^3 3^1 19^1) \cdot (2^5 3^3 17^1 19^1) \cdot (2^6 3^2 11^2 17^1) \bmod n = 2^7 3^3 11^1 17^1 19^1 = 1\ 985\ 323$$

need to satisfy:

$$x^2 = y^2 \bmod n$$

and so $(x - y)(x + y) = 0 \bmod n$. Therefore:

$$GCD[x - y, n] = 1993,$$

it's non-trivial divisor.

The complexity of Dixon's algorithm is $L_n[1/2, 2\sqrt{2}]$.

6.2. Quadratic Sieve

The Quadratic Sieve [**Pom84**] algorithm tries to find numbers x, y that satisfy the same congruence as in Dixon's algorithm. Thanks to the optimized way of selecting those numbers, QS offers a visible speed up over the Dixon's algorithm. Its running time is equal to $L_n[1/2, 1]$.

Example 6.3. *Let $n = 2\ 058\ 769$ (the same as in the Example 6.2). The algorithm starts from defining two entities:*

(1) $m = \lfloor \sqrt{n} \rfloor$

(2) $f(x) := (x+m)^2 - n$

Then, the two parameters are used:

- B *– which defines the size of "sieving" set – we are again looking for* B*-smooth numbers.*
- C *– that defines the size of the interval we are working on: we perform computations for numbers in the set* $\{-C, -C+1, \ldots, -1, 0, 1, \ldots, C\}$.

Because we evaluate $f(x)$ *for* $x < 0$*, we extend the definition of* $\mathcal{F}(B)$*, by adding* $\{-1\}$ *to this set. Now, we try to find, which of the numbers* $f(x)$ *for* $x \in \{-C, \ldots, C\}$ *are* B*-smooth. Let* $B = 20$ *and* $C = 30$ *then:*

$$\begin{array}{lrrl}
\lambda_1 & f(-22) = & -65\ 025 & = (-1)^1 3^2 5^2 17^2 \\
\lambda_2 & f(-7) = & -22\ 440 & = (-1)^1 2^3 3^1 5^1 11^1 17^1 \\
\lambda_3 & f(-1) = & -5\ 280 & = (-1)^1 2^5 3^1 5^1 11^1 \\
\lambda_4 & f(1) = & 456 & = 2^3 3^1 19^1 \\
\lambda_5 & f(4) = & 9\ 075 & = 3^1 5^2 11^2 \\
\lambda_6 & f(29) = & 81\ 600 & = 2^6 3^1 5^2 17^1
\end{array}$$

This table is found by trial division, for the elements of the factor base. After the table is found, a system of linear equations is built up. The solution of the system gives the answer to the question, which equation one should multiply in order to find correct x *and* y.

We have:

$$\begin{cases}
\lambda_1 + \lambda_2 + \lambda_3 = 0 \bmod 2 & (for -1) \\
3\lambda_2 + 5\lambda_3 + 3\lambda_4 + 6\lambda_6 = 0 \bmod 2 & (for\ 2) \\
2\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 = 0 \bmod 2 & (for\ 3) \\
2\lambda_1 + \lambda_2 + \lambda_3 + 2\lambda_5 + 2\lambda_6 = 0 \bmod 2 & (for\ 5) \\
\lambda_2 + \lambda_3 + 2\lambda_5 = 0 \bmod 2 & (for\ 11) \\
2\lambda_1 + \lambda_2 + \lambda_6 = 0 \bmod 2 & (for\ 17) \\
\lambda_4 = 0 \bmod 2 & (for\ 19)
\end{cases}$$

A non-trivial solution of the above system of equations is: $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6) = (0, 1, 1, 0, 1, 1)$.

This translates into the following:

$$x = f(-7)f(-1)f(4)f(29) \bmod n = 1427 \cdot 1433 \cdot 1438 \cdot 1462 \bmod n = 1\ 008\ 826,$$

$$y = 2^7 3^2 5^3 11^2 17 \bmod n = 1\ 804\ 033.$$

And now, we may compute n's non-trivial factors:

$$p = \mathsf{GCD}[x + y, n] = 1\ 033,$$

$$q = \mathsf{GCD}[x - y, n] = 1\ 993.$$

Problems

1. Show that if the running time of a factoring algorithm is upper bounded by $L_n[0, v]$ then it runs in polynomial time. Show that if the running time is $L_n[1, v]$ then this algorithm is exponential.
2. Draw the function $f(n) = L_{2^n}[x, y]$ for $n \in \{1, 2, \ldots, 4096\}$ and $(x, y) = (1/2, 1)$, $(x, y) = (1/3, (64/9)^{1/3})$.
3. Factor 11111 using the quadratic sieve.
4. Use the $p-1$ method to factor (a) $n = 138277151$, (b) $n = 18533588383$.
5. Estimate the running time of the $p - 1$ method.

Bibliography

[ABP+13] Nadhem AlFardan, Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob C N Schuldt, *On the Security of RC4 in TLS*, Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13) (Washington, D.C.), USENIX, 2013, pp. 305–320.

[Buc13] Johannes Buchmann, *Introduction to cryptography*, Springer Science & Business Media, 2013.

[Dix81] John D Dixon, *Asymptotically fast factorization of integers*, Mathematics of computation **36** (1981), no. 153, 255–260.

[FM00] Scott R Fluhrer and David a. McGrew, *Statistical Analysis of the Alleged RC4 Keystream Generator*, Fast Software Encryption, 7th International Workshop (2000), 19–30.

[FMS01] S Fluhrer, I Mantin, and A Shamir, *Weaknesses in the key scheduling algorithm of RC4*, Selected areas in cryptography (2001).

[Gol97] J Golic, *Linear Statistical Weakness of Alleged RC4 Keystream Generator*, Advances in Cryptology — EUROCRYPT '97 (Walter Fumy, ed.), Lecture Notes in Computer Science, vol. 1233, Springer Berlin Heidelberg, Berlin, Heidelberg, jul 1997.

[Hey02] Howard M Heys, *A tutorial on linear and differential cryptanalysis*, Cryptologia **26** (2002), no. 3, 189–221.

[KL14] Jonathan Katz and Yehuda Lindell, *Introduction to modern cryptography*, CRC press, 2014.

[KLZ16] Michal Kulis, Pawel Lorek, and Filip Zagorski, *Randomized stopping times and provably secure pseudorandom permutation generators*, International Conference on Cryptology in Malaysia, Springer, 2016, pp. 145–167.

[Mir02] Ilya Mironov, *(Not So) Random Shuffles of RC4*, Advances in Cryptology—CRYPTO 2002 (2002).

[MS01] Itsik Mantin and Adi Shamir, *A Practical Attack on Broadcast RC4*, Fast Software Encryption, 8th International Workshop, Yokohama, Japan (2001), 152–164.

[Pom84] Carl Pomerance, *The quadratic sieve factoring algorithm*, Workshop on the Theory and Application of of Cryptographic Techniques, Springer, 1984, pp. 169–182.

[Riv92] Ronald L Rivest, *The rc4 encryption algorithm. rsa data security*, Inc., March **12** (1992), 9–2.

[Sho99] Peter W Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM review **41** (1999), no. 2, 303–332.

[VP15] Mathy Vanhoef and Frank Piessens, *All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS*, USENIX Security Symposium, 2015.