Übungen zur Vorlesung Systemsicherheit

Symmetrische Kryptographie

Tilo Müller, Reinhard Tartler, Michael Gernoth

Lehrstuhl Informatik 4

10. November 2010

Organisatorisches

Termine:

- Mi. 12:30 - 14:00 (01.153)
- Fr. 12:30 - 14:00 (01.153)
- Anmeldung über Waffel
- `https://waffel.informatik.uni-erlangen.de/`

Mailingliste:

- Automatischer Abgleich mit Waffel-Teilnehmerlisten
- Gegenseitige Hilfe möglich und erwünscht!
- `mailto:i4syssec@lists.informatik.uni-erlangen.de`

Organisatorisches

- Gruppen zu je 2-3 Studenten
  Anmeldung: `/proj/i4syssec/bin/syssecgroups`
- 5 Blätter mit je 15 Punkten
  (+ mögliche Zusatzpunkte)
- 2-3 Wochen pro Blatt
- Bestanden wenn
  - 50% der Punkte insgesamt
  - 33% der Punkte pro Blatt
- Hauptsächlich Programmieraufgaben
  (plus kurze Wissensfragen / Rätsel)
  - Programmiersprache ist C (GCC + Make)
  - Zu jedem Projekt gehört eine README
  - Wissensfragen als `txt` oder `pdf` (kein `doc` o.ä.)
- Abgabe über SVN (nach `$REPOSITORY/$BLATT/$AUFGABE/`)

Abgabetermine für Übungsaufgaben

Thema	Deadline
Symmetric Cryptography	Fr, 26.11.2010
Asymmetric Cryptography	Fr, 17.12.2010
Library Preloading	Fr, 14.01.2011
Buffer Overflows	Fr, 28.01.2011
Real Exploits	Fr, 11.02.2011

Zu den Aufgaben relevante Dateien finden Sie unter

`/proj/i4syssec/ex/$(BLATT)/$(AUFGABE)`

Aufgabe 1.1

Kryptoanalyse historischer Chiffren

Aufgabe 1.1 | Terminologie

- **Kryptographie**: Wissenschaft über die Verschlüsselung von Informationen
- **Kryptoanalyse**: Methoden zum Brechen von Verschlüsselungen
- **Kryptologie**: Kryptographie + Kryptoanalyse

Zwei Arten der Kryptographie:

- **Symmetrische Verschlüsselung** oder *Secret-Key Kryptographie*: Ein und derselbe geheime Schlüssel zum Ver- bzw. Entschlüsseln.
  $\Rightarrow$ 1. Aufgabenblatt
- **Assymmetrische Verschlüsselung** oder *Public-Key Kryptographie*: Ein allgemein bekannter, öffentlicher Schlüssel zum Verschlüsseln und ein geheimer, privater zum Entschlüsseln.
  $\Rightarrow$ 2. Aufgabenblatt

Aufgabe 1.1 | Historische Chiffren

- Alle *symmetrisch* (Public-Key Verfahren erst seit 1970er)
- Aus heutiger Sicht absolut *unsicher*; selbst von Laien zu brechen
- **Monoalphabetic Cipher**: Jedes Zeichen wird eindeutig auf ein anderes Zeichen des Alphabets abgebildet (z.B. Shift Cipher, Substitution Cipher, Affine Cipher, ...)
- **Polyalphabetic Cipher**: Jedes Zeichen kann je nach Position auf verschiedene Zeichen abgebildet werden (z.B. Viginere Cipher, Hill Cipher, Permutation Cipher, ...)

In Aufgabe 1.1 sind ausschließlich monoalphabetische Chiffren gegeben; diese sind besonders leicht durch **Häufigkeitsanalyse** zu brechen.

Aufgabe 1.1 | Monoalphabetische Chiffren

- **Shift Cipher**: Verschiebe jedes Zeichen X um k Positionen, d.h.:

$$e_k(X) = X + k \mod n$$

$$d_k(Y) = Y - k \mod n$$

Beispiel:

$$n = 26, k = 3$$

$$\text{HELLO } (7, 4, 11, 11, 1) \rightarrow \text{ KHOOR } (10, 7, 14, 14, 17)$$

Aufgabe 1.1 | Monoalphabetische Chiffren

- **Substitution Cipher**:

$$e_k(X) = k(X)$$

$$d_k(Y) = k^{-1}(Y)$$

Wobei k eine bijektive Abbildung des Alphabets auf sich selbst ist.

Beispiel:

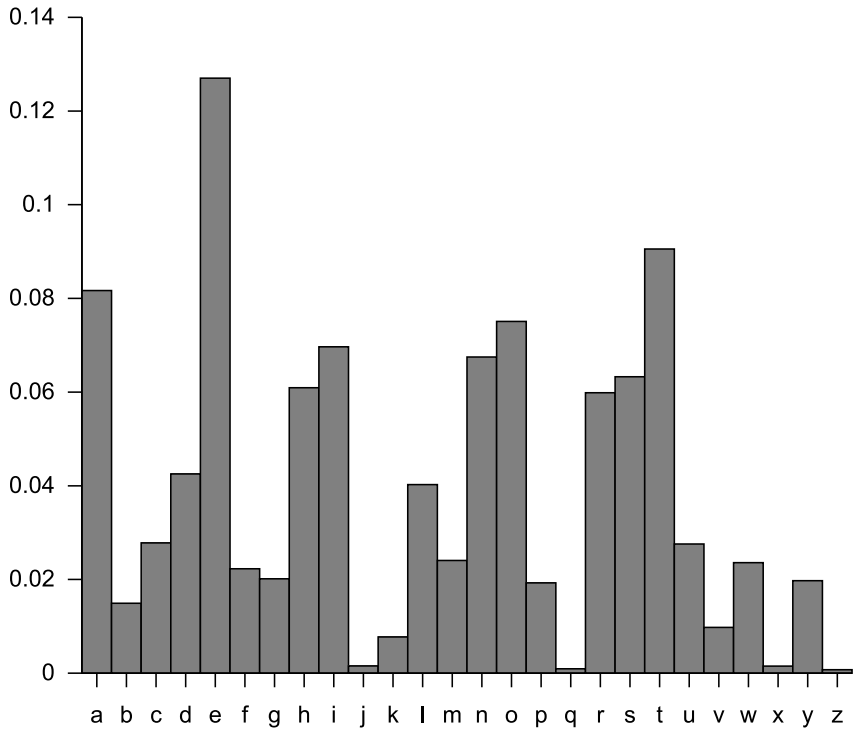
a	b	c	d	e	f	g	h	i	j	k	l	m
f	g	n	e	a	t	x	z	o	i	q	b	h
n	o	p	q	r	s	t	u	v	w	x	y	z
k	y	w	v	c	p	j	l	s	d	m	u	r

D.h. “COMPUTERSCIENCE” $\rightarrow$ “NYHWLJACPNOAKNA”

Aufgabe 1.1 | Häufigkeitsanalyse

- Einen *Shift Cipher* zu brechen ist trivial, da es maximal 26 verschiedene Schlüssel k gibt.
- Bei einem *Substitution Cipher* gibt es dagegen $26! = 4 * 10^{26}$ verschiedene Schlüssel!

$\Rightarrow$ Haüfigkeitsanalyse:



Häufigkeitstabelle für englischen Text. Quelle: Wikipedia

Aufgabe 1.1 | Häufigkeitsanalyse

- Für einen unverschlüsselten, englischen Text gilt
  - $e = 12\%$
  - $t = 9\%$
  - $a = 8\%$
  - $o = 7\%$
  - $i = 7\%$
  - ...
- Berechne die relativen Häufigkeiten aller Buchstaben des verschlüsselten Textes
- Falls z.B. $z = 12\%$ gilt, so gilt (wahrscheinlich) $e(a) = z$
- Damit lassen sich schnell die häufigsten 5-10 Buchstaben ermitteln
- Der Rest ergibt sich nach und nach durch andere Merkmale der Sprache
  (z.B. häufiges “th” im Englischen)

Aufgabe 1.2

Unix Password Hashes

Aufgabe 1.2 | Passwörter zur Anmeldung

Es werden keine Klartextpasswörter, sondern kryptographische Hashes verwendet.

- Ein Salt hilft gegen Wörterbuch-Angriffe
- Passworthash: `f(salt,pw)`
- Salt und Hash werden gemeinsam abgespeichert.
- Bei der Anmeldung wird wieder Salt und Passwort gehasht und mit gespeichertem Ergebnis verglichen.

Aufgabe 1.2 | Gängige Verfahren

- Unix
  - traditionelles Unix `crypt()` (modifiziertes DES)
  - MD5
  - Blowfish (standard in OpenBSD)

`/etc/shadow`

- Windows
  - Windows Lan Manager Hash (LM Hash)
  - DES $2 * 7$ Zeichen
  - Konversion zu Großbuchstaben

Sicherung in SAM (Security Account Manager)

Aufgabe 1.2 | Netzwerkanmeldung

- Hashes über das Netzwerk
- Challenge-Response Verfahren
- Unix
  - NIS
  - LDAP
  - Kerberos
- Windows
  - NTLM (NT Lan Manager)
    - ab NT 3.1
    - Benutzerpasswort wird mit MD4 gehashed, auf 21 Bytes aufgefüllt
    - Challenge Response mit DES und 7 Byte-Stücken des Hashes als Key
    - `http://davenport.sourceforge.net/ntlm.html`
  - Active Directory (LDAP + Kerberos)

Aufgabe 1.2 | Passwort Hash Algorithmen

Grundlegender Aufbau:

```
$<CODE>$<SALT>$<deadbeefHASHdeadbeef>
```

- Gängige `CODE`s sind:
  - MD5: `$1$`
  - SHA256: `$5$`
  - SHA512: `$6$`
  - Blowfish: `$1b$`
- `SALT` ist eine zufällige Zeichenfolge im BASE64 Alphabet: `[A-Za-z0-9./]`

Aufgabe 1.2 | Hinweise

- Sicheres Einlesen von Passwörtern:

```
#include <openssl/evp.h>
int EVP_read_pw_string(char *buf, int length, const char *prompt, int verify);
```

- Erzeugen von Passworthashes:

```
#include <crypt.h>
char *crypt(const char *key, const char *salt);
```

Ende

- Nächste Woche: Aufgabe 1.3, File Encryption
- Jetzt: Betreute Gruppenarbeit / Fragestunde