Übungen zur Vorlesung Systemsicherheit

Asymmetrische Kryptographie

Tilo Müller, Reinhard Tartler, Michael Gernoth

Lehrstuhl Informatik 1 + 4

24. November 2010

Aufgabenblatt 2

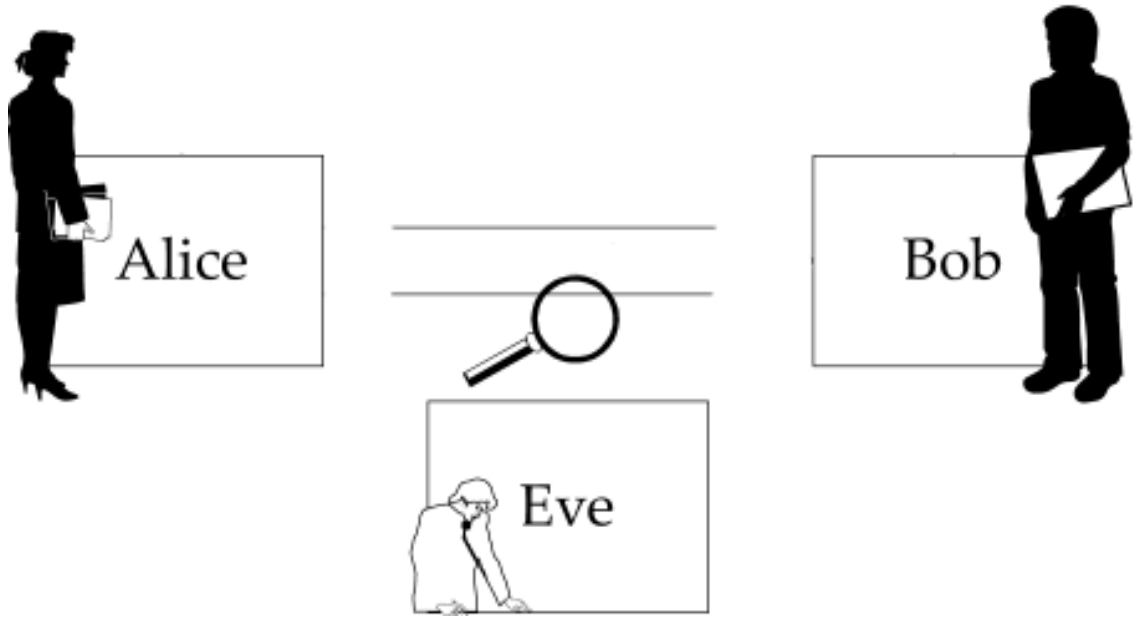
Asymmetrische Kryptographie

Aufgabe 2.3

Diffie Hellman

Aufgabe 2.3 | Key Exchange

- Problem der *Schlüsselverteilung*: Wie einigen sich Alice und Bob auf einen geheimen Schlüssel?
- Früher (bis 1970er): geheime *Codebücher*; Schutz und Verteilung solcher Codebücher war äußerst schwierig. Beispiel: Enigma.
- Ziel: Festlegung eines gemeinsamen, geheimen Schlüssels *ohne* zuvor gemeinsames Wissen zu haben.



Aufgabe 2.3 | Diffie Hellman

Lösung (Stanford, 1976): *Diffie Hellman Key Exchange*

- **Alice** und **Bob**: Einigen sich auf Primzahl **p** und Primitivwurzel **g** (öffentlich)
- **Alice**: Wählt Geheimnis **a** und verschickt $A = g^a \bmod p$ (*Einwegfunktion*)
- **Bob**: Wählt Geheimnis **b** und verschickt $B = g^b \bmod p$ (*Einwegfunktion*)
- **Alice** und **Bob**: Berechnen je gemeinsamen, geheimen Schlüssel **K**

$$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p$$

$$K = B^a \bmod p = (g^b \bmod p)^a \bmod p = g^{ba} \bmod p = g^{ab} \bmod p$$

Anmerkung: p, g, A und B sind öffentlich; es kann aber nicht ohne weiteres auf a und b geschlossen werden (diskrete Logarithmen) und damit nicht auf K.

Aufgabe 2.3 | Diffie Hellman / small numbers

Aufgabenstellung: Implementierung von D-H mit kleinen Zahlen

1. **Schlüssel-Austausch** (bei interner Wahl von $a = 5, b = 8$):

```
$ ./alice <p> <g>
$ ./alice 29 11
A: 14
B> 16
>> Secret key: 23
```

```
$ ./bob <p> <g>
$ ./bob 29 11
A> 14
B: 16
>> Secret key: 23
```

2. **Schlüssel-Wiederherstellung** (ohne Kenntnis von a, b):

```
$ ./dh_crack <p> <g>
$ ./dh_crack 29 11
A> 14
B> 16
>> Secret key: 23
```

Aufgabe 2.4

Zusatzaufgabe

Aufgabe 2.4a | Prime Factorization

Aufgabe:

- $n = 156322800124910087613791861224242082460240865839176093260082641651422812534909549523860138887220426 3$
- Faktorisieren Sie den RSA Modulus n in die Primfaktoren p und q.

Hinweise:

- n ist hier wesentlich größer als in Aufgabe 2.1, dennoch kleiner als echte RSA Moduli.
- Die Primzahlen p und q haben in etwa die gleiche Anzahl von Stellen.
- Sie müssen das Programm zur Primfaktorzerlegung nicht selbst schreiben.

Aufgabe 2.4b | Network-compatible Challenge/Response

Aufgabe: Netzwerk-fähige Implementierung von Aufgabe 2.2b.
Hinweis: Es steht Ihnen frei beliebige Bibliotheken einzubinden.

1. **Server**:

```
$ ./server <private-server-key> <public-client-key> <port>
$ ./server privserver.pem pubclient.pem 7893
Listening on port 7893 ...
Connection accepted
Authentication successfull
Send AES message: Hello from server!
Received AES message: Hello from client.
Good bye.
```

2. **Client**:

```
$ ./client <private-client-key> <public-server-key> <ip> <port>
$ .client privclient.pem pubserver.pem 127.0.0.1 7893
Connecting to 127.0.0.1:7893 ...
Connection established
Authentication successfull
Received AES message: Hello from server!
Send AES message: Hello from client.
Connection closed.
```

Ende

- Nächste Woche: Aufgabenblatt 3
- Library Preloading & Overlay Filesystems (nur 10 Pkt.)
- Jetzt: Betreute Gruppenarbeit / Fragestunde