Der post-quanten Algorithmus New Hope

EIN NEUES PRINZIP FÜR KRYPTOGRAFISCHEN SCHLÜSSELAUSTAUSCH

Dipl. Math. Xenia Bogomolec, 26.01.2017

Inhalt

- Motivation
  - Persönliche Inspiration
  - Grundsätzliche Notwendigkeit
- Secret Keys
  - Symmetrische Verfahren
  - Asymmetrische Verfahren
  - Hybride Verfahren
- Key Exchange
  - Diffie-Hellmann
  - RSA
  - ECC
- New Hope
  - Peikerts KEM
  - KEM vs. Key Exchange
  - Zu testende Möglichkeiten bei Brute Forcef
  - Man in the Middle
- Googles Test
- Anhang
  - Zufallswerte
  - Fehlerquote
  - Länge der Nachrichten
  - Herleitung Zahlen New Hope
  - Herleitung Zahlen RSA
  - Raffiniertere RSA-Hacks

Motivation

Persönliche Inspiration

Kryptografie, die Verschlüsselung von Nachrichten/Informationen, so dass kein Unbefugter den Inhalt erkennen kann, hat mich schon immer fasziniert.

Als ich las, dass es eine neue Methode für Schlüsselaustausch gibt, bei der der Schlüssel nicht mehr explizit übermittelt wird, und trotzdem beide Seiten denselben Schlüssel berechnen können, dachte ich:

Diese Magie will ich verstehen!!!

Umso begeisterter war ich, als ich rausfand, dass die mathematischen Grundlagen dazu genau aus dem Bereich stammen, in dem ich an der Leibniz Universität in Hannover geforscht habe.

Motivation

Grundsätzliche Notwendigkeit

Einfaches Beispiel: 31313 = 173*181

Ab etwa 100 Stellen wird es jedoch schwierig. Eine 1024-bit Zahl hat in dezimaler Darstellung 309 Stellen, eine 4096-bit Zahl sogar 1234.

Die Sicherheit konventioneller Kryptografischer Methoden beruht auf der Annahme, dass große ganze Zahlen nicht in annehmbarer Zeit in Primfaktoren zerlegt werden können.

Peter Wiliston Shor (* 14. August 1959 in New York), ein amerikanischer Mathematiker und Informatiker, hat jedoch einen Algorithmus zur Faktorisierung ganzer Zahlen für Quanten-Computer entwickelt.

Dieser Algorithmus wird Faktorisierung von großen ganzen Zahlen so schnell berechnen können, dass die konventionellen Kryptografischen Methoden nicht mehr haltbar sein werden.

Motivation

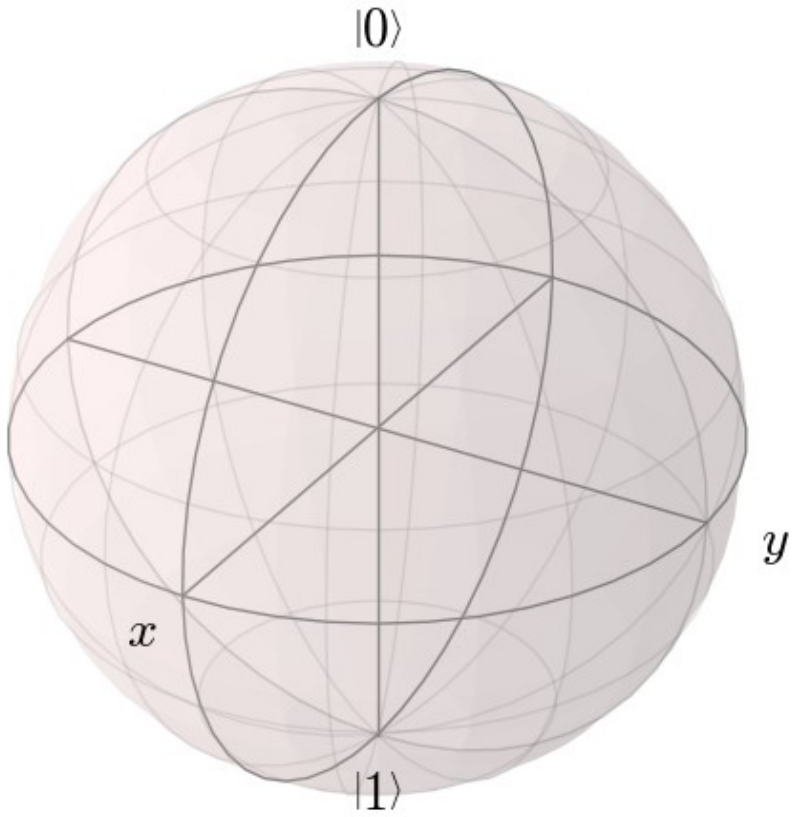
Grundsätzliche Notwendigkeit

2001 wurde Shor's Algorithmus zur Primzahlfaktorisierung mit einem simplen Quantencomputer mit 7 Zuständen durchgeführt.

Binäre Daten (bits):

0 oder 1

Quanten Zustände (qubits):



Secret Keys

Die Basis der Sicherheit jeglicher Kryptografischer Verfahren beruht auf der Geheimhaltung der Keys zum Entschlüsseln von verschlüsselten Daten.

Unterschiedliches Key-Management für

- Symmetrische Verfahren
- Asymmetrische Verfahren
- Hybride Verfahren

Secret Keys
Symmetrische Verfahren

Symmetrische Verfahren benutzen denselben Key für Ver- und Entschlüsselung von Daten.

Alice		Bob
Secret Key	=	Secret Key

→ Geheimer Schlüsselaustausch erforderlich!

Secret Keys

Symmetrische Verfahren

Vorteile:

- Effiziente Algorithmen
- Einfaches Schlüsselmanagement

Nachteile:

- Schlüsselaustausch gefährdet Verschlüsselung

Secret Keys

Asymmetrische Verfahren

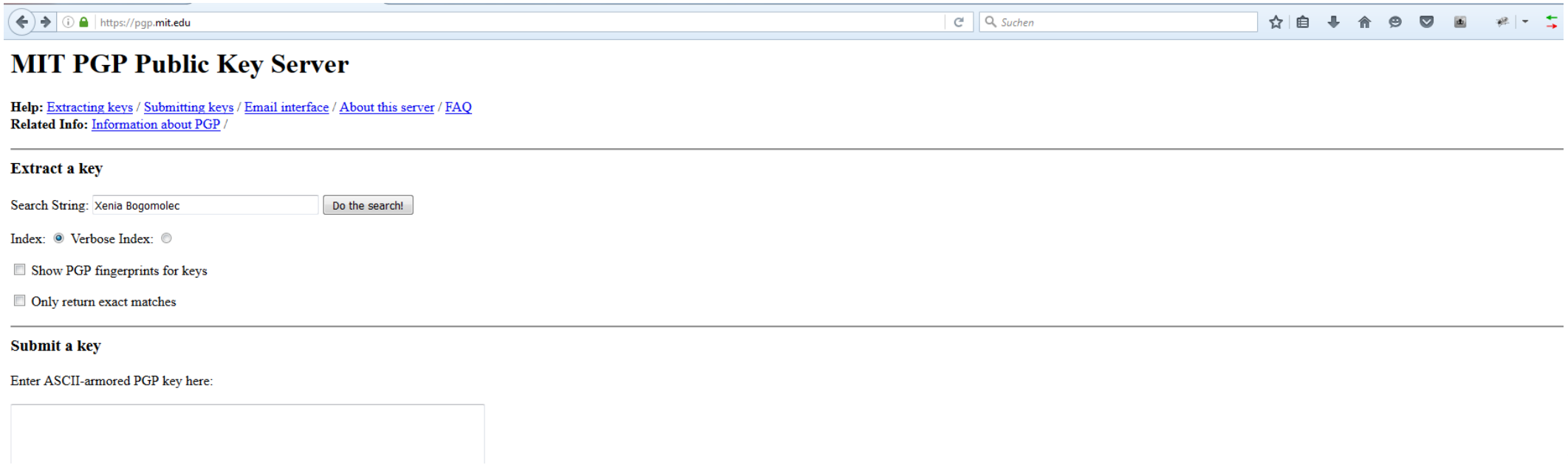
Asymmetrische Verfahren verwenden unterschiedliche Keys zum Ver- und Entschlüsseln.

Alice		Bob
Secret Key A	≠	Secret Key B
Public Key A	=	Public Key A
Public Key B	=	Public Key B

→ Kein geheimer Schlüsselaustausch nötig!

Secret Keys
Asymmetrische Verfahren

Vorteile:
- Hohe Sicherheit
- Austausch von Public Keys gefährdet Verschlüsselung nicht!
  (Es gibt sogar Public Key Server)



Secret Keys

Asymmetrische Verfahren

Vorteile:

- Hohe Sicherheit
- Austausch von Public Keys gefährdet Verschlüsselung nicht!
  (Es gibt sogar Public Key Server)

Nachteile:

- 2 unterschiedliche Keys pro Kommunikationspartner
- ca. 10 000 Mal langsamer als symmetrische Verfahren
- Lange Schlüssel

Secret Keys

Hybride Verfahren

Kombination von symmetrischem Verfahren und asymmetrischem Verfahren.

Symmetrisch:
Verschlüsselung der Daten mit zufällig gewähltem *Session-Key*.

Asymmetrisch:
Session-Key wird zur Übermittlung mit dem Public Key des Empfängers verschlüsselt.

→ Schlüsselaustausch findet asymmetrisch verschlüsselt statt!

Secret Keys

Hybride Verfahren

Alice		Bob
Session Key	=	Session Key
Secret Key A	≠	Secret Key B
Public Key A	=	Public Key A
Public Key B	=	Public Key B

→ Schlüsselaustausch findet asymmetrisch verschlüsselt statt!

Secret Keys

Hybride Verfahren

OpenSSL ist eine Bibliothek mit unterschiedlichen Cipher-Suiten für hybride Verschlüsselungsverfahren. Mittlerweile werden die darin enthaltenen Methoden als TLS (Transport Layer Security) bezeichnet.

Vorteile:

- Hohe Sicherheit
- Effiziente Algorithmen
- Austausch von Public Keys gefährdet Verschlüsselung nicht!

Nachteile:

- Aufwändiges Schlüsselmanagement
- Lange Schlüssel

Key Exchange

Pre-quantum

Herkömmliche Verfahren für Schlüsselaustausch:

- Diffie-Hellmann (1976)
- RSA (1977)
- Elliptic-Curve (1985)

Sicherheit beruht darauf, dass die Secret Keys nicht in annehmbarer Zeit aus den gesendeten Daten berechnet werden können.

Key Exchange

Pre-quantum

Elliptische Kurven hat man eingesetzt, weil die Berechnungen auf ihnen langsamer sind als auf den ganzen Zahlen. Dadurch kann man mit kürzeren Schlüsseln dasselbe Sicherheitsniveau erreichen wie mit den anderen Verfahren.

Mathematische Grundlagen:

- Diffie-Hellmann:
  Diskrete Exponentiation $a = r^s$

- RSA:
  Produkt großer Primzahlen (Key Exchange) $n = p_1 \cdot p_2$
  Diskrete Exponentiation (Crypto) $a = r^s$

- Elliptic-Curve:
  Multiplikation von Punkten auf elliptischen Kurven $a = r \cdot P$

Key Exchange

Pre-quantum

Umkehr-Funktionen der obigen Funktionen:

- Diskreter Logarithmus
- Faktorisierung von ganzen Zahlen
- Division von Punkten auf elliptischen Kurven

Laufen mit den besten bekannten Algorithmen nicht in annehmbarer Zeit auf pre-quanten Computern.

ABER:
Mit Shor‘s Algorithmus zur Faktorisierung von ganzen Zahlen auf Quantencomputern werden diese Umkehrfunktionen in annehmbarer Zeit berechenbar sein.

Key Exchange

Post-quantum

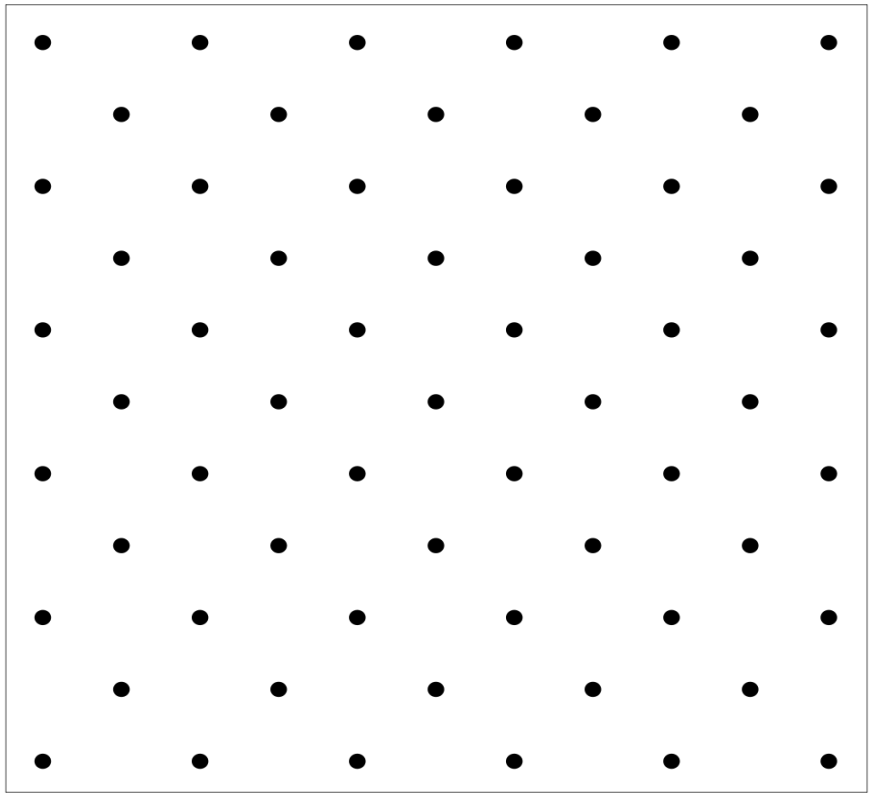
Neue mathematische Grundlagen:
Gitter-basierte Algorithmen

Was ist ein Gitter?

$$\Lambda = \left\{ \sum_{i=1}^{n} a_i v_i \;\middle|\; a_i \in \mathbb{Z} \right\}$$

Key Exchange

Post-quantum



Beispiele für 2-dimensionale Gitter

Key Exchange

Post-quantum

Räume potentieller Lösungen für polynomiale Gleichungen über den ganzen Zahlen können auch als Gitter betrachtet werden.

Bezug zu Datentransformation:
Jeder Character oder String kann auch als ganze Zahl interpretiert werden.

Beipiel für ascii codierten String:
Hallo = 48616c6c6f (hexadezimal) = 310‘872‘140‘911

*Auf ganzen Zahlen werden viele Funktionen zu Einwegfunktionen, die auf rationalen oder reellen Zahlen leicht umkehrbar sind.

Key Exchange

Pre- vs. post-quantum

Die Mächtigkeit der Lösungsräume steht für die Möglichkeiten, die ein Angreifer durchprobieren muss, wenn er den Algorithmus selbst nicht angreifen kann. Das wäre dann ein sogenannter Brute Force Angriff.

Pre-quantum:

$a = r^s$

$n = p_1 \cdot p_2$ → Lösungsräume: Ganze Zahlen oder Punkte auf EC (1-dimensional)

$a = r \cdot P$

Post-quantum:

$a(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 \cdot x + a_0$

$a_i \in Z, \quad i \in \{0, \ldots, n\}$

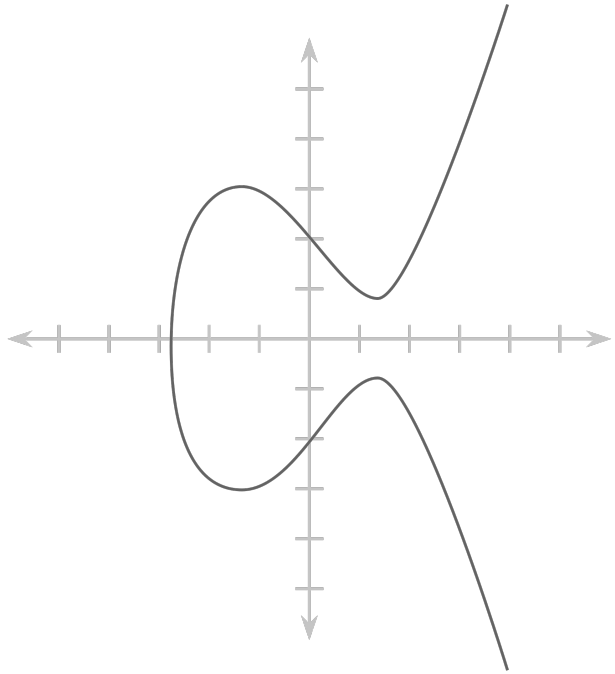
→ Lösungsraum: 2-dimensionales Gitter

Key Exchange

Pre- vs. Post-quantum

Pre-quantum:

und



Post-quantum:

Key Exchange

Gitter

Vermutung als mathematischer Fachbegriff (Conjecture):

1. Stärker als eine einfache Annahme
2. Alles Bekannte deutet darauf hin, dass es so ist
3. Es gibt noch keinen Beweis

Hardness von worst-case Gitter-Problemen ist eine Vermutung

→ Grundlage der Sicherheit für viele post-quanten Probleme

New Hope

Kein expliziter Schlüsselaustausch mehr.

Neues Prinzip: Key Encapsulation Mechanism

→ Chris Peikerts (MIT) KEM mit effizienteren Parametern

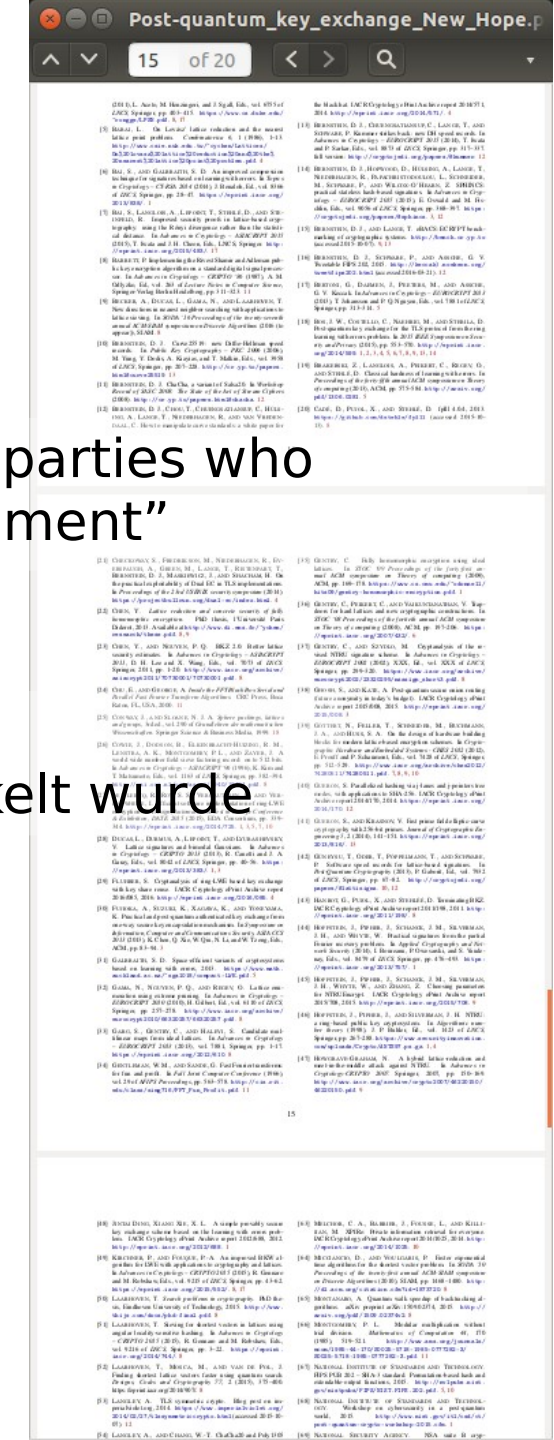
New Hope

Peikerts KEM

„ A simple, lowbandwidth reconciliation technique that allows two parties who ‘approximately agree’ on a secret value to reach exact agreement”

Simple?

… im Vergleich zu Methoden, aus denen der KEM entwickelt wurde



New Hope

Peikerts KEM

„ A simple, lowbandwidth reconciliation technique that allows two parties who ‘approximately agree’ on a secret value to reach exact agreement”

Mathematische Grundlage: Ring Learning with Errors

$$
\begin{array}{ccccccccc}
14s_1 & + & 15s_2 & + & 5s_3 & + & 2s_4 & \approx & 8 \ mod\ 17 \\
13s_1 & + & 14s_2 & + & 14s_3 & + & 6s_4 & \approx & 16 \ mod\ 17 \\
6s_1 & + & 10s_2 & + & 13s_3 & + & 1s_4 & \approx & 3 \ mod\ 17 \\
 & & & & & & & \vdots & \\
6s_1 & + & 7s_2 & + & 16s_3 & + & 2s_4 & \approx & 3 \ mod\ 17
\end{array}
$$

New Hope

Peikerts KEM

Voraussetzungen: Polynom vom Grad 1024 mit Koeffizienten aus $\{0, \dots, 12288\}$

Alice	Bob	Alice
$p_a(x) = a(x) \cdot s_a(x) + e_a(x)$	$p_b(x) = a(x) \cdot s_b(x) + e_b(x)$	
Sendet $p_a(x)$ und $a(x)$ zu Bob	$v(x) = p_a(x) \cdot s_b(x) + e_b(x)$ KEM → - Session Key k - Reconciliation string c	
	Sendet $p_b(x)$ und c zu Alice	$w(x) = p_b(x) \cdot s_a(x) + e_a(x)$ Reconciliation string c KEM → - Session Key k

New Hope

KEM vs. Diffie-Hellmann

Alice	Bob	Alice
$p_a(x) = a(x) \cdot s_a(x) + e_a(x)$	$p_b(x) = a(x) \cdot s_b(x) + e_b(x)$	
Sendet $p_a(x)$ und $a(x)$ zu Bob	$v(x) = p_a(x) \cdot s_b(x) + e_b(x)$ *KEM* → - Session Key k - Reconciliation string c	
	Sendet $p_b(x)$ und c zu Alice	$w(x) = p_b(x) \cdot s_a(x) + e_a(x)$ Reconciliation string c → - Session Key k

Alice	Bob	Alice
a, g, p $A = g^a \bmod p$ Sendet A, g, p an Bob	b $B = g^b \bmod p$ $K = A^b \bmod p = g^{ab} \bmod p$ Sendet B an Alice	$K = B^a \bmod p = g^{ba} \bmod p$

New Hope

Zu testende Möglichkeiten bei Brute Force Angriff

$$s(x)\,bei\,New\,Hope: 1024^{12289} = 3{,}76 \cdot 10^{36993}$$

$$p_1, p_2\,bei\,RSA: \leq 3{,}13 \cdot 10^{1232}$$

New Hope
Man in the Middle

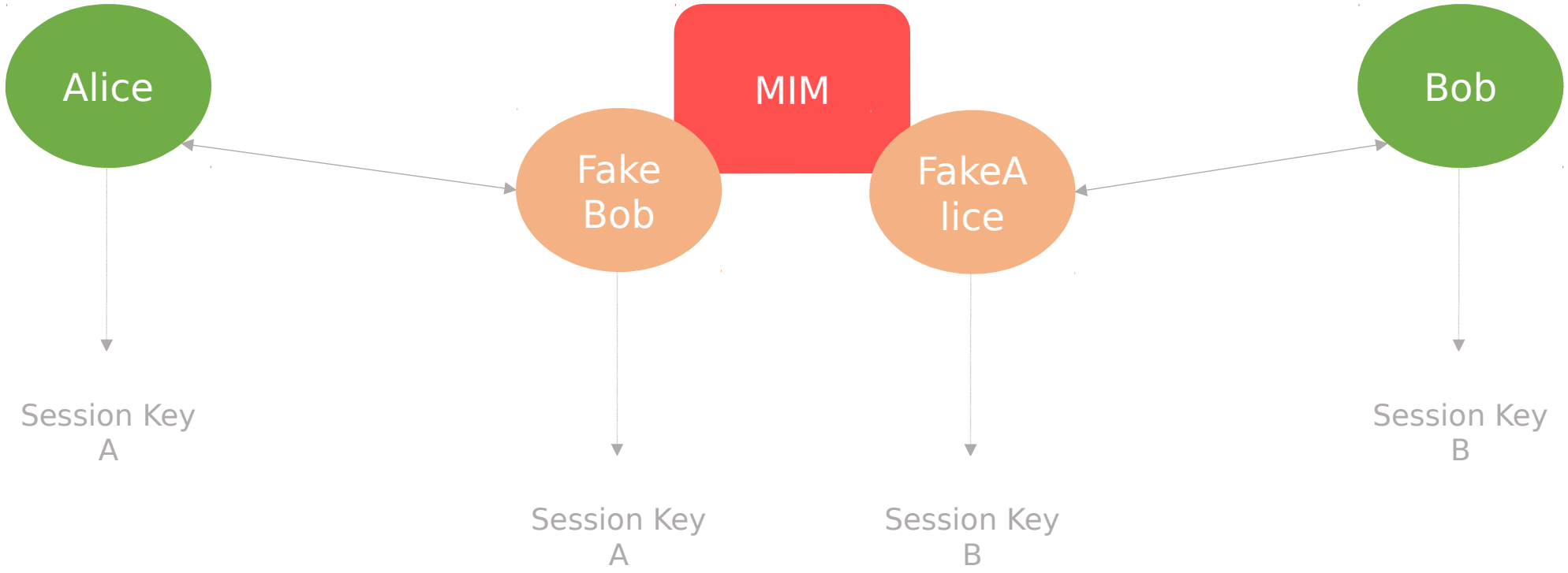
Man in the Middle ist möglich bei

- KEM nicht authentifiziert
  (✓bei aktueller Implementation)
- Polynom $a(x)$ bekannt
  (Wahlweise unterschiedlich)
- Parameter für Berechnungen bekannt (1024, 12289)
  (✓öffentlich, sehr bewusst gewählt, damit der KEM schnell genug läuft)

ABER: Google bettet den KEM in ein ECC-Verfahren ein

New Hope

Man in the Middle



Googles Test

Das post-quanten Konzept New Hope für Schlüsselaustausch wurde 2016 auf einigen Verbindungen zwischen Chrome und den Google Servern getestet.

Eine Testphase von 2 Jahren sollte die Community dazu herausfordern, den New Hope Algorithmus anzugreifen.

Um die User nicht zu gefährden, wurde New Hope in ein ECC-Verfahren eingebettet, der Algorithmus heisst CECPQ1.

Die Testphase wurde nach wenigen Monaten abgebrochen.

Googles Test

Ende der Testphase

Zwei Wissenschaftler veröffentlichten am 21.11.2016 eine algorithmische Lösung für das 'closest vector problem in lattices'. Es wurde jedoch am 24.11.2016 wegen eines Fehlers zurückgezogen.

Am 28.11.2016 kündigte Adam Langley das vorzeitige Beenden des New Hope Tests an mit folgenden Begründungen:

1) Die Veröffentlichung des oben genannten Papers wurde als erreichtes Ziel betrachtet.
2) Auf sehr langsamen Verbindungen könnte der Schlüsselaustausch zu sehr ins Gewicht fallen.
3) Es wird angenommen, dass es bis heute nur sehr einfache Quanten-Computer gibt
4) Die Einbindung von New Hope in TLS 1.3 könnte etwas zu komplex sein.
   (TLS 1.3 consists of one less round trip than TLS 1.2)

Anhang
Zufallswerte

Die Wahl der Polynome $s(x), e(x)$ in $p(x) = a(x) \cdot s(x) + e(x)$ erfolgt aufgrund einer sogenannten zentrierten Binomialverteilung.
$a(x)$ wird für jede Session mit einem SHAKE-128 aus einem 256-bit seed neu generiert.
Die Sicherheitsminderung im Vergleich zu von einem Rauschgenerator * erzeugten Zufallswerten ist so gering, dass man sie vernachlässigen kann.

RNG** darf durch PRNG*** ersetzt werden!

* Physikalische Quelle für Zufallswerte.
** Random Number Generator
*** Pseudo Random Number Generator

(Jeder reine Software-basierte Generator erzeugt nur Pseudo-Zufallswerte.)

Anhang
Fehlerquote

Die Wahrscheinlichkeit, dass Alice eine anderen Session-Key als Bob berechnet ist kleiner als

$$2^{-60} = 8{,}67 \cdot 10^{-19}$$

Also würde bei nicht mal einer von einer Trillion Verbindungen “Datensalat” anstatt sinnvoller Daten ankommen. Diesen Fehler würde man abfangen, indem einfach eine neue Session iniziiert wird.

Anhang
Länge der Nachrichten

Sender	Nachricht
Alice	1824 Bytes
Bob	2048 Bytes

Eine Zahl aus {0,…,12289} kann in zwei Bytes dargestellt werden (FFFF = 65535).
Für ein Polynom vom Grad 1024 bräuchte man somit $2 \cdot 1024 = 2048$ Bytes.
Diese Größe kann jedoch mittels einer Zahlentheoretischen Transformation reduziert werden.

Anhang

Herleitung Zahlen New Hope

Möglichkeiten für $s(x), e(x)$ in $p(x) = a(x) \cdot s(x) + e(x)$: $1024^{12289} = 3{,}76 \cdot 10^{36993}$
New Hopes Parameter für Berechnungen:

- 1024 ist die Dimension des Ringes in dem die Berechnungen stattfinden (höchster Grad eines Polynoms)
- 12289 ist der Modulus
  (Koeffizienten des Polynoms sind aus $\{0,\ldots, 12288\}$)

Bei modularen Berechnungen kann eine Zahl/Polynom ein Produkt größerer Zahlen/Polynome sein.

Beispiel: $5 \cdot 6 = 30 \equiv 2 \, mod \, 7$

Anhang
Herleitung Zahlen RSA

Möglichkeiten für $p_{1,}p_2$ bei einem 4096-bit public RSA key $n=p_1\cdot p_2$:

Bytes	Hexadezimal	Formel dezimal
1	FF	2^8-1
2	FFFF	$2^{2\cdot 8}-1$
512	512 X FF	$2^{512\cdot 8}-1$

Wenn wir p_1 finden ist p_2 klar. 0 und 1 kommen nicht in Frage. Falls $p_1=2$ ist, ist auch n gerade, was man sofort sehen würde. Alle anderen geraden Zahlen und alle auf 5 endenden Zahlen sind keine Primzahlen, somit bleiben höchstens

$$\left(2^{512\cdot 8}-2\right)/2-\left(2^{512\cdot 8}/5\right)=\left(2^{512\cdot 8-1}-1\right)-\left(2^{512\cdot 8}/5\right)=3{,}13\cdot 10^{1232}$$

Zahlen zu testen.

Anhang

Raffiniertere RSA-Hacks

Die richtigen Primzahlen $p_{1,} p_2$ könnten mittels Primzahllisten ermittelt werden, dann brauchen nicht mehr alle eben errechneten Möglichkeiten durchgetestet zu werden.
Die aktuell grösste bekannte Primzahl ist $2^{74'207'2810}-1$.

Wenn die Primzahlfaktoren nahe bei $\sqrt{n}$ liegen, können sie in kurzer Zeit mit probalistischen Verfahren gefunden werden.