# Post-Quanten Kryptografie


# Chancen und Risiken in der IT-Sicherheit

**Stefan Schubert**
**Institut für IT-Sicherheitsforschung**

# Stefan Schubert

5. Semester IT-Security

Research Assistant

Institut für IT-Security Research

fh st.pölten

Projektmitarbeit “KIF”

seit einem Jahr

# Gliederung

PK Kryptografie

Quantencomputer

Problemstellung und Lösung

Projekt und Ausblick

# PK-Kryptografie

# PK Kryptografie

**Schlüsselaustauschproblem**

Symmetrische Kryptografie braucht selben Schlüssel. Diesen Schlüssel geheim zu tauschen war sehr schwer.. Jahrelange Suche nach Algorithmen.

**DH - Schlüsselaustausch**

1976 Erster „Public-Key“ Algorithmus von Diffie-Hellman welcher auch nach ihnen bekannt wurde. Schlüsselaustausch Verfahren.

**RSA-Verschlüsselung**

Erster Verschlüsselungsalgorithmus RSA (Rivest Shamir Adelman) im Jahr 1977 (schon vorher „entdeckt“ von einem NSA Mitarbeiter, durfte aber nicht veröffentlicht werden)

**Häufigste Anwendungen**

Schlüsselaustausch und Signaturen

# PK Kryptografie

**PK sehr wichtig**

Das Internet wäre nicht in der heutigen Form möglich, PK-Krypto wird überall eingesetzt, von Banküberweisungen bis Chipkarten.

**Primitiven**

Public-Key Kryptografie basiert auf mathematischen Falltür-Funktionen die in einer Richtung leicht zu rechnen sind, in die andere aber sehr schwer

**Nachteile**

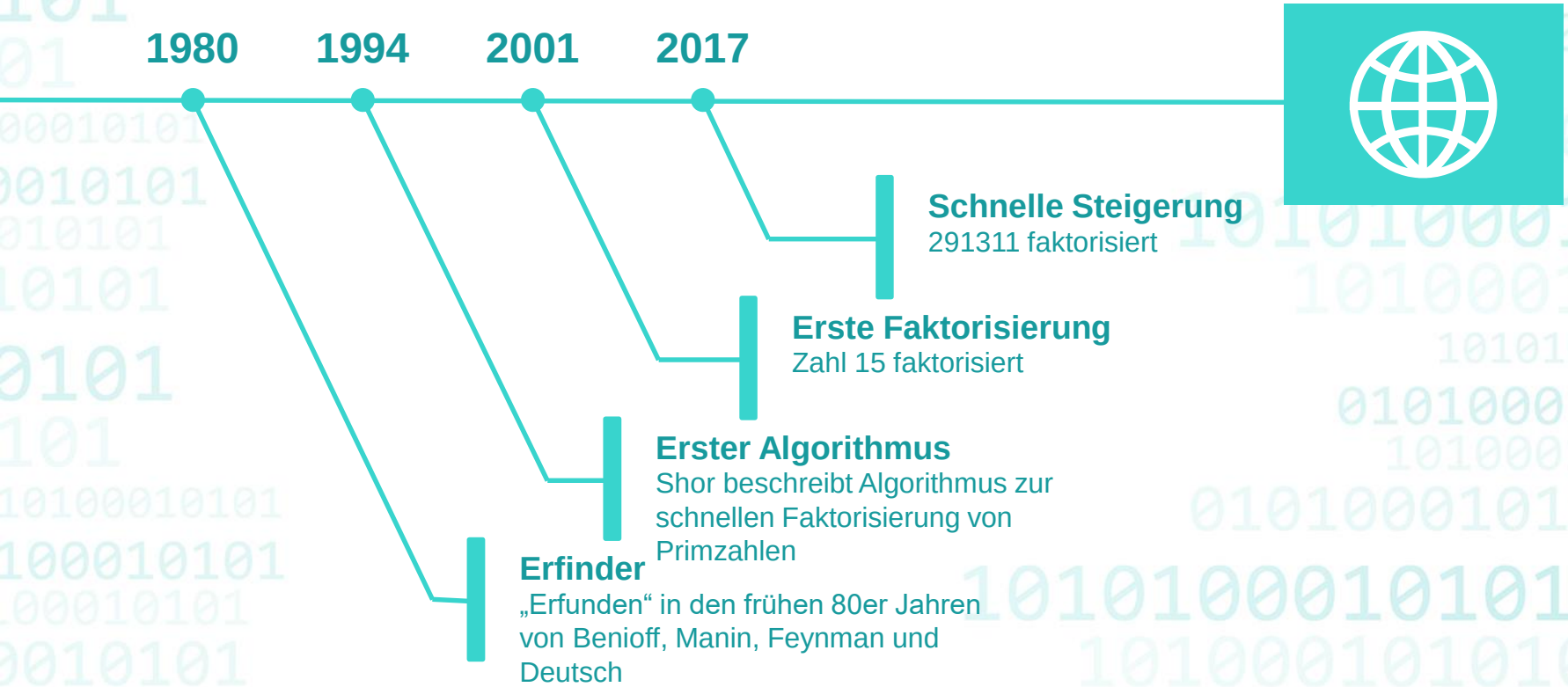
Für Verschlüsselung sehr langsam, deswegen oft Hybrid-Systeme. (ECDH-AES256…)

**Wichtig für Schlüssel an sich**

Kein Schlüsselaustausch mehr notwendig, bzw. Sicher Schlüsselaustausch wird dadurch möglich

# Quantencomputer

# Quantencomputer - Entwicklung



# Quantencomputer

**Vorteile**

Viele Probleme die zuvor sehr schwierig waren können schnell gelöst werden

**Quanten-Suche**

Kann Objekte in einem unsortierten Array schneller finden als ein klassischer Computer
O(n) -> O(√n)

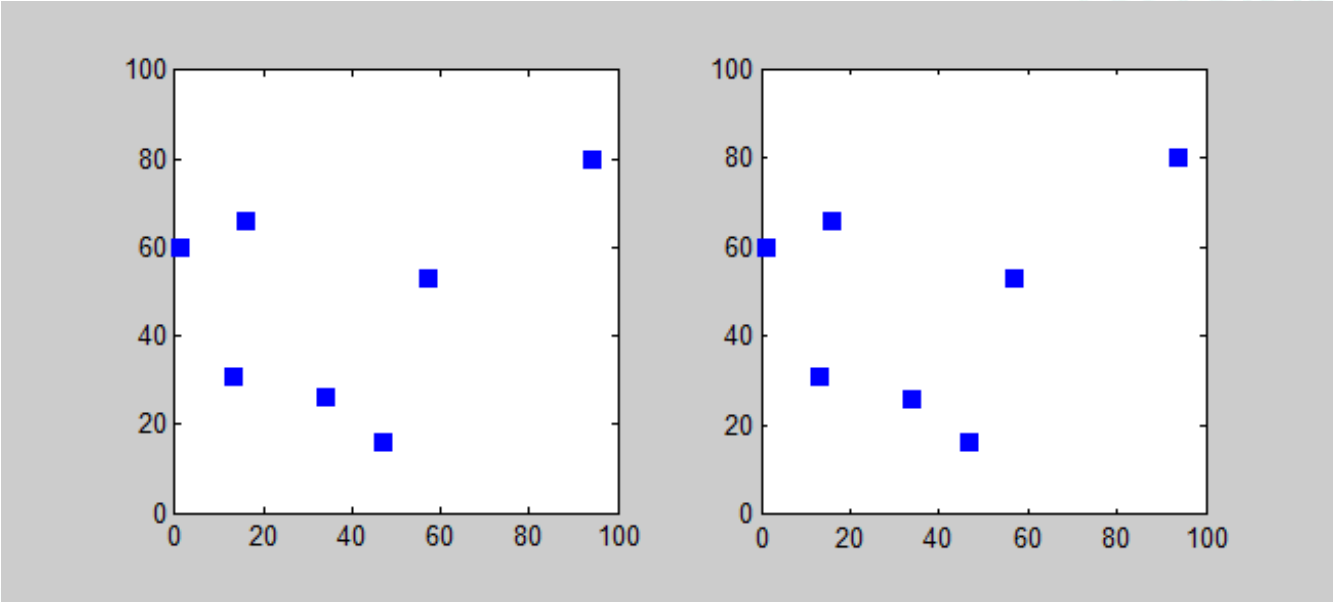
**Nachteile**

Derzeit weiß niemand ob es wirklich je einen riesigen Quantencomputer geben wird der die oben genannten Vorteile auch ausnutzen kann. Falls dies jedoch geschieht werden damit auch einige Felder der Kryptografie nutzlos gemacht -> PK-Kryptografie

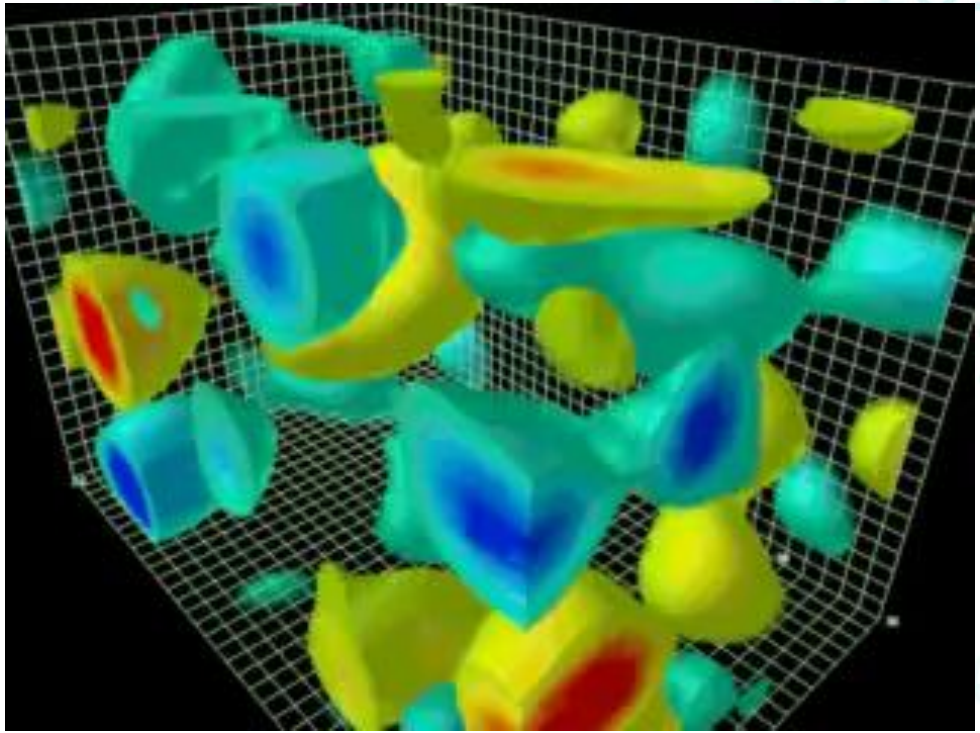
**Quantensimulation**

Sehr wichtig in den Naturwissenschaften zb. Modellierung von Molekül-Systemen oder Darstellung von Quarks

# Berechnung klassischer Computer

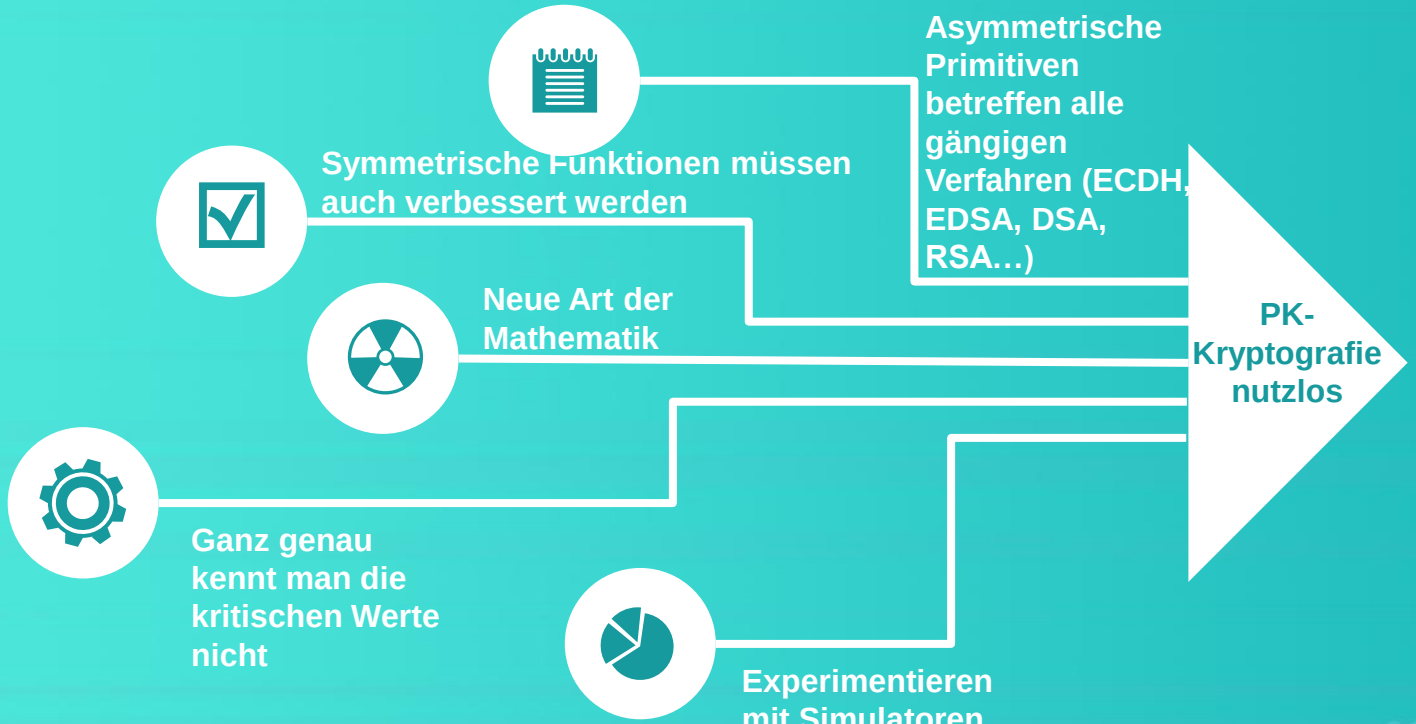


# Berechnung Quantencomputer



# Problemstellung und Lösung

# Problemstellung




**Shors Algorithmus**

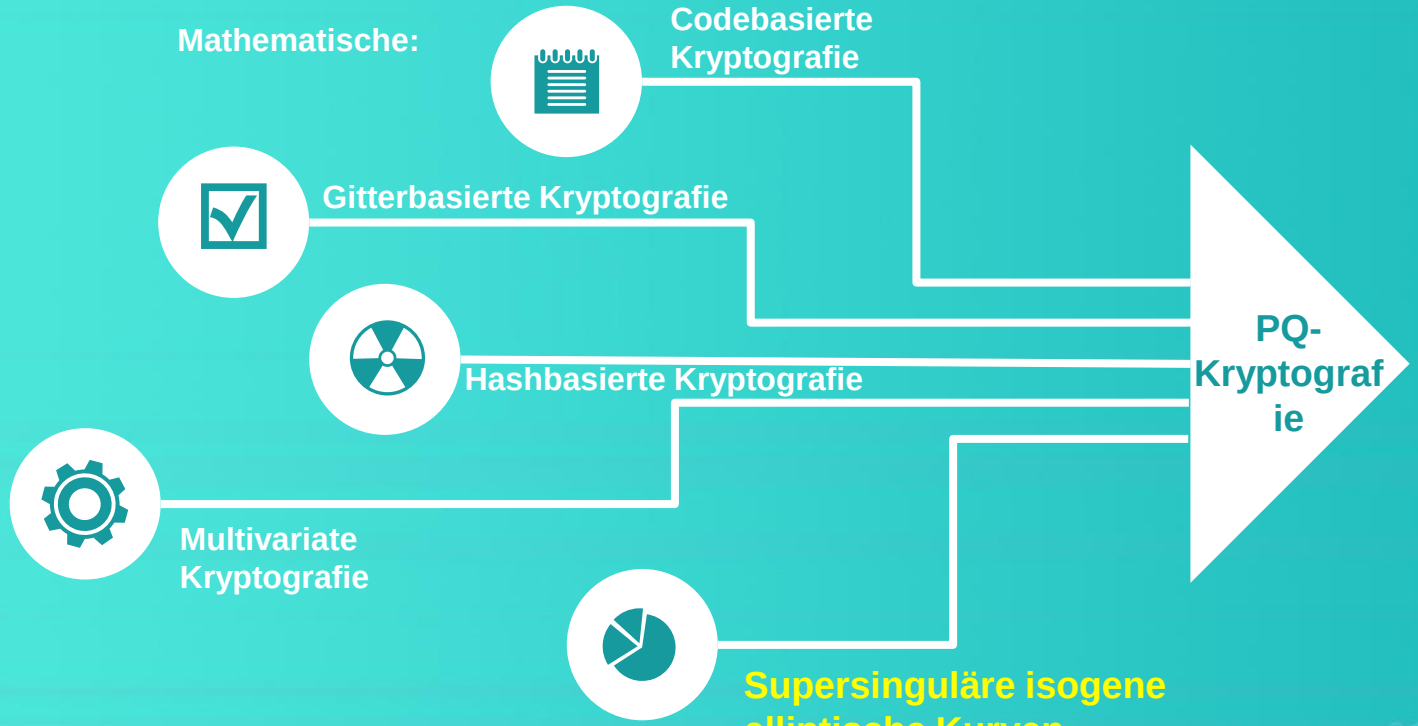
m = # bits

Shors Algorithmus RSA -> $4m^3$time und 2m Qbits

ECC Attacke-> $360m^3$ und 6m Qbits (Proos-Zaika 2004, Roettler-Naehrig-Svare-Lauter 2017)

# Lösungsansätze

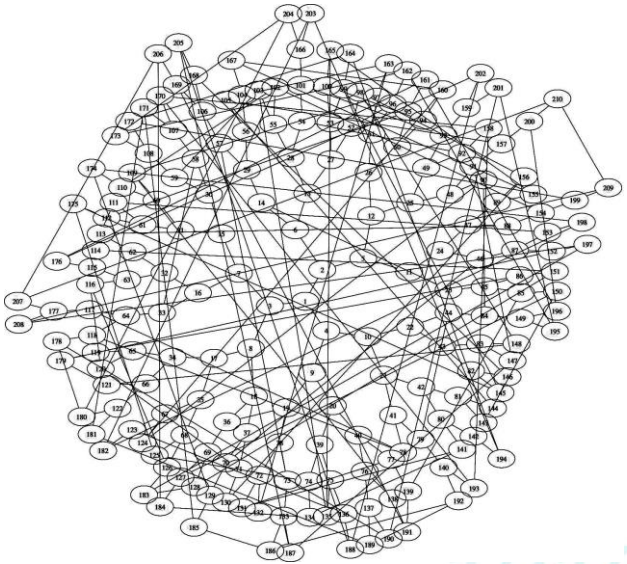
**Physikalische zb. Quantenkryptographie und Funkkanaldaten**



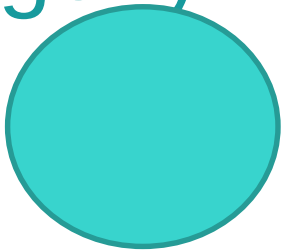
# Projekt und Ausblick

# Supersinguläre isogene elliptische Kurven

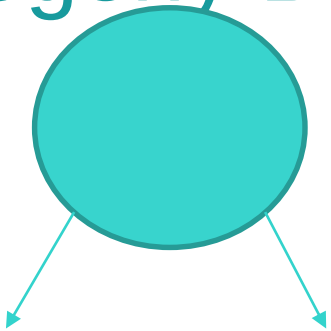
**Berechnet Beziehungen zwischen bestimmten elliptischen Kurven -> dadurch höhere Komplexität (Expander Graph)**



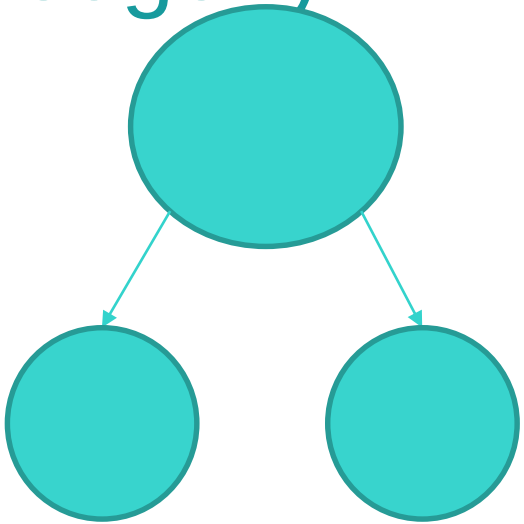
# Supersingular isogeny Diffie Hellman (SIDH)



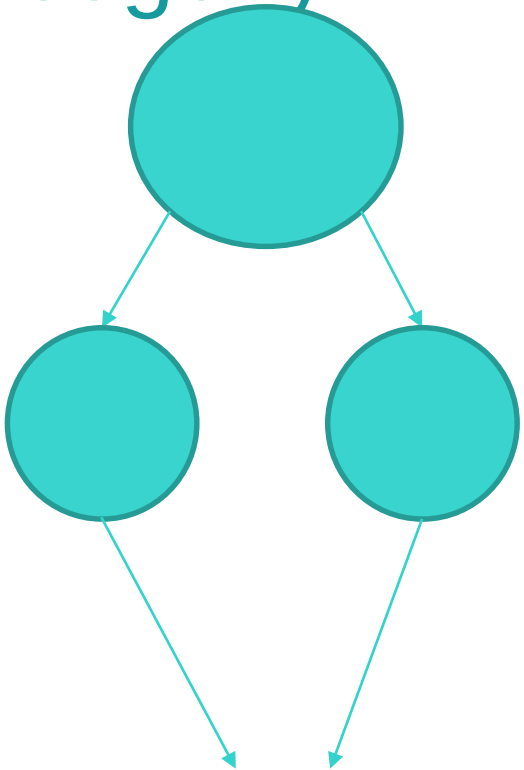
# Supersingular isogeny Diffie Hellman (SIDH)



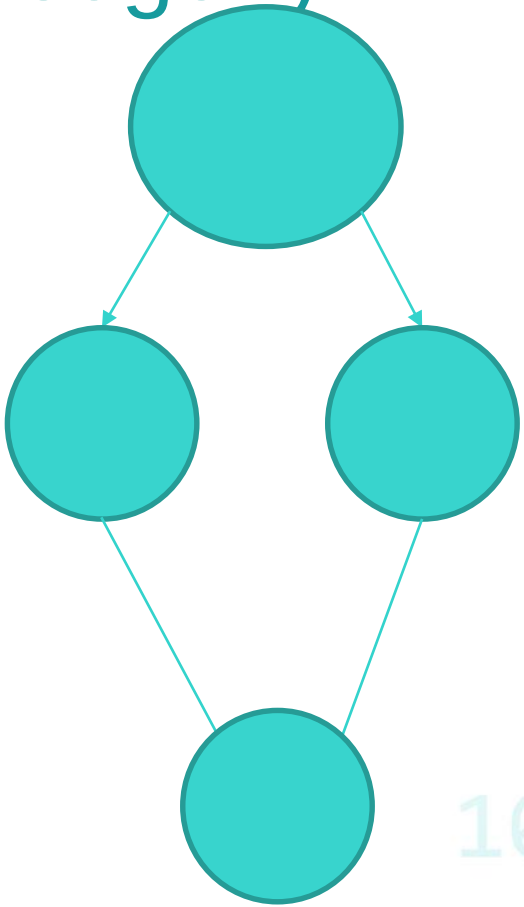
# Supersingular isogeny Diffie Hellman (SIDH)



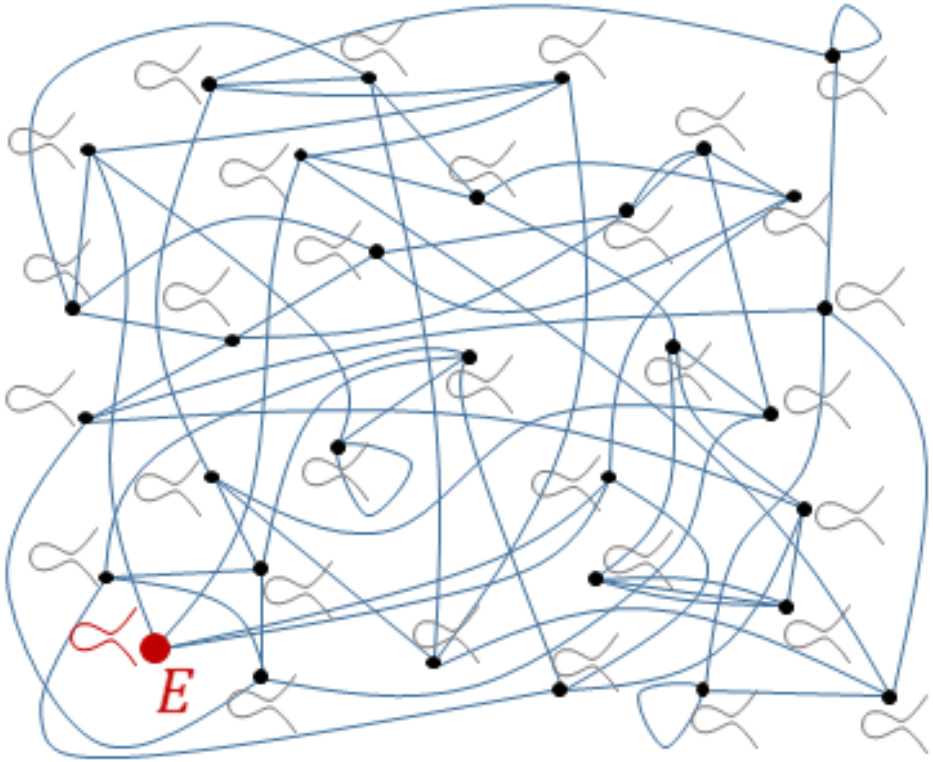
# Supersingular isogeny Diffie Hellman (SIDH)



# Supersingular isogeny Diffie Hellman (SIDH)



# Supersingular isogeny Diffie Hellman (SIDH)



# Vorteile isogene Kryptosysteme

**01**

**Schlüssellänge**

Diese Art von Kryptografie hat sehr kurze Schlüssel (Vgl. ECDH und andere PQ-Algorithmen)

.

**02**

**Flexibel**

Alle kryptografischen Anforderungen werden erfüllt: Vertraulichkeit (Verschlüsselung), Integrität (Hash), Authentizität (Signatur)

**03**

**Weniger Speicherintensiv**

Speziell wichtig in Umgebungen wo vielleicht genug Rechenkraft da ist aber nicht genug Speicher wie bei Embedded-Systems, Smart Cards und Java-Cards.

**04**

**Blockchain**

Speicherprobleme auch zb. in Blockchain-Systemen (Blockchain zu groß und kann nicht mehr dezentral gespeichert werden). Auch hier sind kleine Schlüsselgrößen wichtig.

**05**

**Nachteile**

Geschwindigkeit da höherer Rechenaufwand, Signaturen kaum erforscht

# Verbesserungen

**Bis 2017 sehr langsam und keine statischen Schlüssel möglich**
**-> Mit Einführung von SIKE (NIST-Kandidat) kein Problem mehr**

## Performance on x64

| Primitive | Quantum sec. | Problem | Speed | Comm. |
|---|---|---|---|---|
| Classical | | | | |
| RSA 3072 | ~0 bits | factoring | 4.6 ms | 0.8 KB |
| ECDH NIST P-256 | ~0 bits | EC dlog | 1.4 ms | 0.1 KB |
| Passively secure key-exchange | | | | |
| SIDHp503 | 84 bits | isogenies | 10.3 ms | 0.7 KB |
| SIDHp751 | 125 bits | isogenies | 31.5 ms | 1.1 KB |
| IND-CCA secure KEMs | | | | |
| Kyber | 161 bits | M-LWE | 0.07 ms | 1.2 KB |
| FrodoKEM | 103–150 bits | LWE | 1.2–2.3 ms | 9.5–15.4 KB |
| SIKEp503 | 84 bits | isogenies | 10.1 ms | 0.4 KB |
| SIKEp751 | 125 bits | isogenies | 30.5 ms | 0.6 KB |

very fast slow very small large

(*) Obtained on 3.4GHz Intel Haswell (Kyber) or Skylake (FrodoKEM and SIKE).

# Signal Protokoll

| Heute | Morgen |
| --- | --- |
| ECDH | SIDH |
| AES256 | **AES512** |
| HMAC + SHA256 | **HMAC + SHA3** |

# Projekt KIF

## Evaluierung Mathematik

SIDH (Schlüsselaustausch mit Isogenie) wurde evaluiert und der Stand der Technik mittels zweier Arbeiten genau dargestellt.

## Evaluierung Programmierung

C und Assembler-Algorithmus mit effizienter mathematischer Implementierung über die Microsoft-Library

## Prototypen

Erste Implementierungen von Teilen des Algorithmus in Java, mit Geschwindigkeits und Speicheroptimierung für den Einsatz in Hardware Token, Smart Cards etc.

## Effizienzsteigerung

Verschiedene Verfahren werden eingesetzt um den Algorithmus noch weiter zu optimieren

# Optimierungsbeispiel

## V. Comparing Java BigInteger and Huldra-Library BigInteger

The direct benchmark comparison in table I shows the real advantage of using the external BigInt Library instead of the native BigInteger class. The benchmark was done with an Intel i7 6700HQ 2.6 GHz processor with four processing cores. The following calculations have been done:

- **addition**: Adding two 100.000 digit numbers 100.000 times
- **subtraction**: Subtracting two 100.000 digit numbers 100.000 times
- **multiplication**: Multiplication of a 300 digit number 1000 times
- **many small multiplications**: Calculation of 50.000 factorial
- **big multiplication**: Multiplication of two 500.000 digit numbers
- **division**: Divide two 400.000 digit numbers 1000 times

TABLE I
Speed comparison between BigInteger(Java) and BigInt(Huldra-Library) in seconds

| | BigInteger | BigInt |
|---|---|---|
| addition | 1.840s | 0.832s |
| subtraction | 1.287s | 0.574s |
| multiplication | 0.714s | 0.479s |
| many small multiplications | 0.852s | 0.417s |
| big multiplication | 0.279s | 0.129s |
| division | 2.608s | 2.069s |

The chart shows improvement across the board. In addition to mathematical improvements, parallelization and multithreading this leads to a significant speedup in the SIDH algorithm.

# Ausblick

**Volle Implementierung**

Fertigstellung eines effizienten Authentifizierungsalgorithmus mit Supersingulären-Elliptischen Kurven.

**Testreihen**

Vollständige Testreihen auf verschiedenen Geräten für Vergleiche

**Fortschritte weiter verfolgen**

Weitere Evaluierung und Implementierung der bis dahin erreichten Forschungsergebnisse

**Blockchain**

Implementierung in eine sichere Blockchain

# Schlusswort

**Sicherheitstechnologie braucht zu lange um sich durchzusetzen !**

**Beispiel DES**
**Eingeführt 1977**

**Gebrochen 1998**

**Weiter in Verwendung**

**Solche Dinge sollten nicht mehr passieren !**


Thank you!

Stefan.Schubert@fhstp.ac.at