



# MERKLE–HELLMAN–KRYPTOSYSTEM

Algorithmus zur Entschlüsselung von Kryptotexten

Projektarbeit im Rahmen der Vorlesung *Algorithm Engineering*

WiSe 2015/16, Friedrich–Schiller–Universität Jena

Sebastian Brühl

## Kryptosystem nach Merkle–Hellman

- 1978 von Ralph C. Merkle und Martin Hellman vorgestellt
- *Hiding Information and Signatures in Trapdoor Knapsacks* [1]
- basiert auf Untersummenproblem  $\rightarrow O(2^n)$
- Entschlüsselung bedeutet Berechnung eines Skalarproduktes
- empfohlene Schlüssellänge um ein Knacken der Verschlüsselung in angemessener Zeit zu verhindern:  $n > 100$
- 1976 – Cray-1 – 80 MFLOPS
- 2016 – raj – 4 TFLOPS
- Gilt die Aussage  $n > 100$  heute noch?

## Vektorisierung

Public Key

2292	1089	211	1625	1283	599	759	315	2597	2463
------	------	-----	------	------	-----	-----	-----	------	------

Klartext

?	?	?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---	---

=

Kryptotext 

6790
------

Bedingung: ? = {0, 1}

## Vektorisierung

Public Key

2292	1089	211	1625	1283	599	759	315	2597	2463
------	------	-----	------	------	-----	-----	-----	------	------

Klartext

0	0	1	1	1	0	1	1	1	0
---	---	---	---	---	---	---	---	---	---

=

Kryptotext 

6790
------

Bedingung: ? = {0, 1}

# Vektorisierung

## Probleme

- Schlüssellängen  $n > 100$  benötigen sehr große Zahlen ( $2^{200}$ )
- Algorithmus in zwei Versionen
- `_64: unsigned long long` (vektorisierbar)
- `_BN: boost::multiprecision::int_512_t` (nicht vektorisierbar)
- Initialisierung/Allozierung des Testvektors
- Padding des Testvektors

# Vektorisierung

- skalare Version: GNU Compiler
- vektorisierte Version: Intel Compiler + OpenMP/SIMD

## Laufzeit in Sekunden



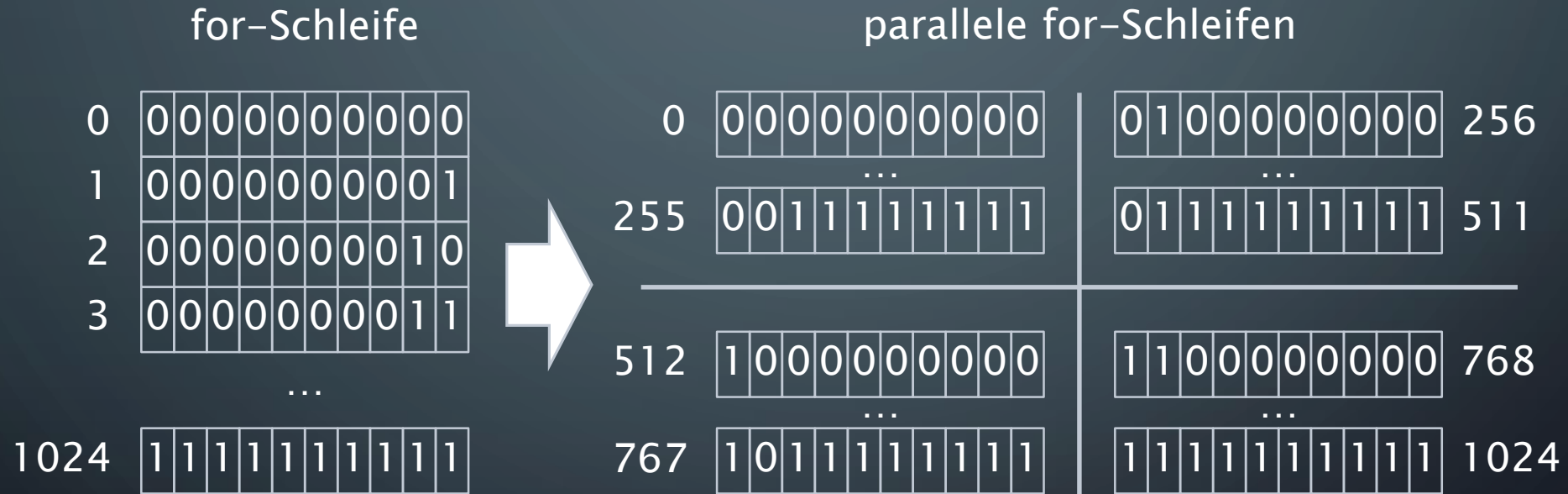
```
#pragma omp for schedule(static)
for(uint64_t n=0; n<n_max; ++n) {
    uint64_t result=0;

    #pragma omp simd aligned(pThreadTestVector,aKeyVector:64) reduction(+:result)
    for(int i=0; i<aKeyLength; ++i) {
        pThreadTestVector[i] = ((n >> i) & 1);
        result += pThreadTestVector[i] * aKeyVector[i];
    }
    if(aDP == result) {
        #pragma omp simd aligned(pThreadTestVector,pThreadPlainVector:64)
        for(int i=0; i<aKeyLength; ++i) {
            pThreadPlainVector[i] = pThreadTestVector[i];
        }
        #pragma omp cancel for
    }
}
```

# Parallelisierung

Hauptaufgabe des Algorithmus

- *Testvektor* • *Public Key*  $\stackrel{?}{=} \text{Kryptotext}$
- $2^n$  voneinander unabhängige Iterationen, im Beispiel  $2^{10}=1024$



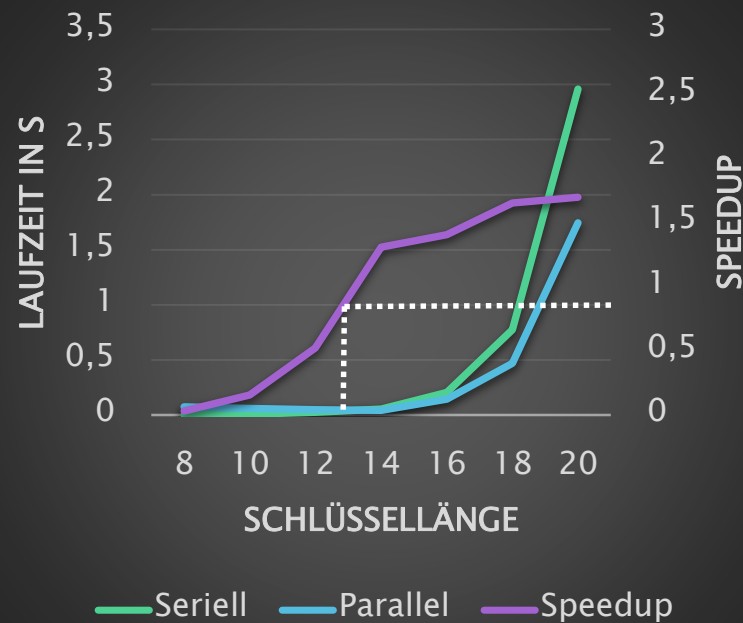
- Problem: Abbruch bei Treffer  $\rightarrow$  `omp cancel for`

# Parallelisierung

## Parallelisierung mittels omp for

```
#pragma omp for schedule(static)
for(uint64_t n=0; n<n_max; ++n) {
    uint64_t result=0;

    #pragma omp simd aligned(pThreadTestVector,aKeyVector:64) reduction(+:result)
    for(int i=0; i<aKeyLength; ++i) {
        pThreadTestVector[i] = ((n >> i) & 1);
        result += pThreadTestVector[i] * aKeyVector[i];
    }
    if(aOP == result) {
        #pragma omp simd aligned(pThreadTestVector,pThreadPlainVector:64)
        for(int i=0; i<aKeyLength; ++i) {
            pThreadPlainVector[i] = pThreadTestVector[i];
        }
        #pragma omp cancel for
    }
}
```





## Verteilte Ausführung

- Merkle–Hellman ist eine Blockchiffre
- jeder Block muss einzeln entschlüsselt werden
- Verteilung der Blöcke auf verschiedene Knoten

### 1 Knoten

```
17382782063368765237262445204734040203088222193789395851173452
10238723564189390855970903643896273642629810592211350464402471
17475990107748665711044593982336475481416529010910714520788414
15908301204664955411117444313543931590206542842821515414640570
6987961414368228039787904434979082798126805867157970945712079
10949136815357991306244357081503829821097621379569451381137489
7094481784855885396337418574317731032706247758097529972562272
15978254813751960628654690347092599216041335162975677422391393
12825037750670366559177476180688860637119164166687159567910720
13372980365000799780756056231912669471404272013833375546208052
7103898953525268850285469829944263548120543999460163590047531
16070612675535546961874807733704054194643823103193483850762825
```



### 6 Knoten

```
17382782063368765237262445204734040203088222193789395851173452
10238723564189390855970903643896273642629810592211350464402471
```

```
17475990107748665711044593982336475481416529010910714520788414
15908301204664955411117444313543931590206542842821515414640570
```

```
6987961414368228039787904434979082798126805867157970945712079
10949136815357991306244357081503829821097621379569451381137489
```

```
7094481784855885396337418574317731032706247758097529972562272
15978254813751960628654690347092599216041335162975677422391393
```

```
12825037750670366559177476180688860637119164166687159567910720
13372980365000799780756056231912669471404272013833375546208052
```

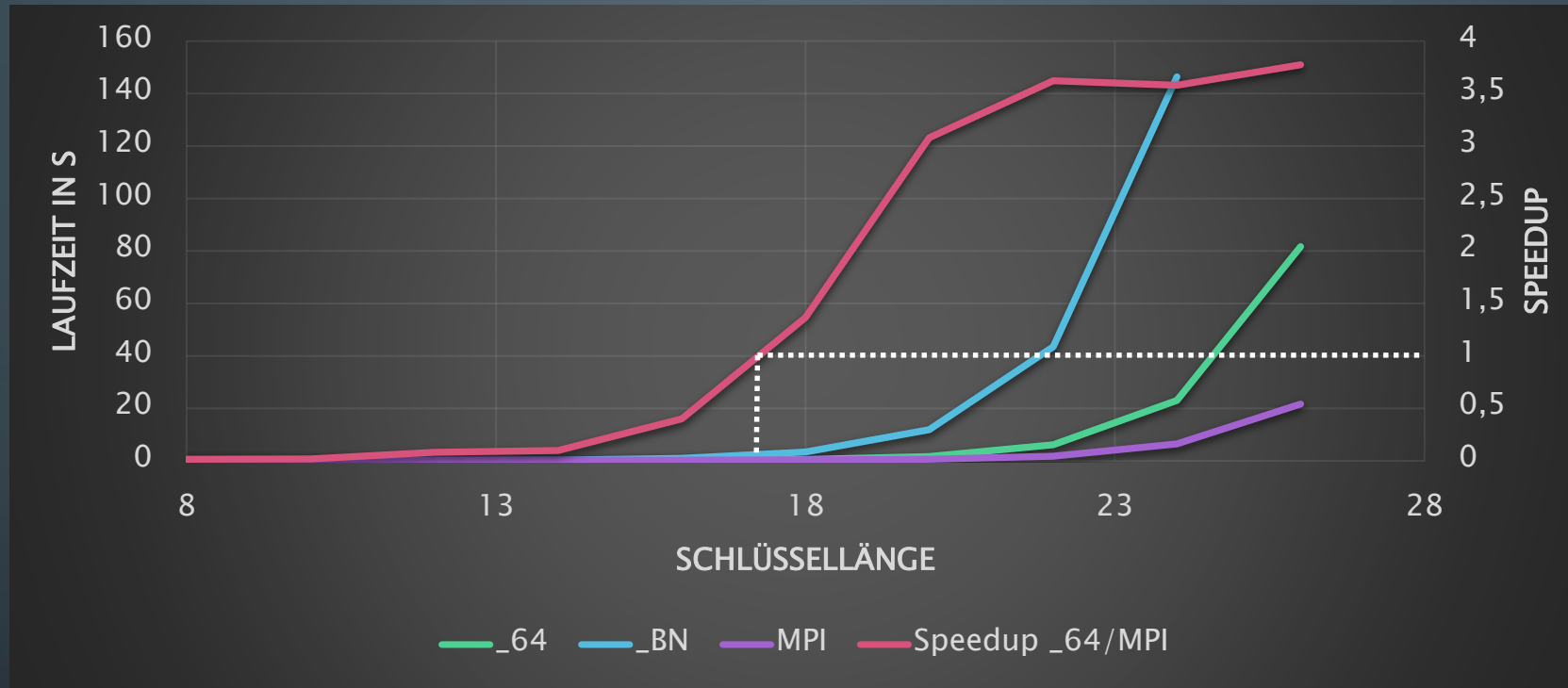
```
7103898953525268850285469829944263548120543999460163590047531
16070612675535546961874807733704054194643823103193483850762825
```

# Verteilte Ausführung



- Verteilung mittels MPI
- beispielsweise 6 Knoten
- Knoten 0 übernimmt Steuerung
- Alle Knoten sind an der Berechnung beteiligt

## Benchmark – Speedup



- ca.  $22 \cdot 10^9$  Operationen/s (MPI)
- Laufzeit MPI bei einer Schlüssellänge von 100:  $80 \cdot 10^{12}$  Jahre
- Alter des Universums:  $14 \cdot 10^9$  Jahre

Quellcode: <https://github.com/sblatgithub/knapsack>

The screenshot shows the GitHub repository page for 'sblatgithub/knapsack'. The repository is titled 'Implementation of Merkle-Hellman Knapsack Cryptosystem'. It has 26 commits, 1 branch, 0 releases, and 1 contributor. The current branch is 'master'. The repository is public and has 1 watch, 0 stars, and 0 forks. The commit history is listed below, showing the most recent commit by sblatgithub: 'improved vector alignment in ResolveDP\_64\_PV2' (4 days ago). Other recent commits include updates to README.md, CMakeLists.txt, and test.cpp.

File	Commit Message	Time
<a href="#">.gitignore</a>	implemented StringToBits	a month ago
<a href="#">CMakeLists.txt</a>	added vectorization report flag	a month ago
<a href="#">COMMENTS.md</a>	Update COMMENTS.md	29 days ago
<a href="#">README.md</a>	Update README.md	a month ago
<a href="#">knapsackLib.cpp</a>	improved vector alignment in ResolveDP_64_PV2	4 days ago
<a href="#">knapsackLib.hpp</a>	improved vector alignment in ResolveDP_64_PV2	4 days ago
<a href="#">main.cpp</a>	improved vector alignment in ResolveDP_64_PV2	4 days ago
<a href="#">test.cpp</a>	full implementation of Merkle-Hellman added, method signatures unified	10 days ago
<a href="#">test.hpp</a>	full implementation of Merkle-Hellman added, method signatures unified	10 days ago

## Quellen

- [1] Ralph C. Merkle, Martin E. Hellman. "Hiding Information and Signatures in Trapdoor Knapsacks"  
In IEEE TRANSACTIONS ON INFORMATION THEORY, VOL: IT-24, NO. 5, SEPTEMBER 1978.