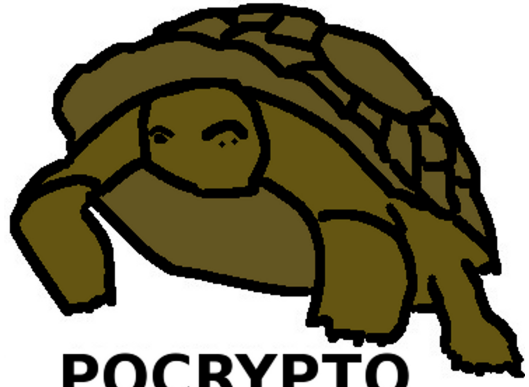
# Kryptographie für Langzeitsicherheit

Tanja Lange




27. November 2015

Bundesdruckerei Seminar

# Alle Kryptographie muss Post-Quanten-Kryptographie sein

- Mark Ketchen, IBM Research, 2012, im Zusammenhang mit Quanten-Computing: "Were actually doing things that are making us think like, 'hey this isn't 50 years off, this is maybe just 10 years off, or 15 years off.' It's within reach."

# Alle Kryptographie muss Post-Quanten-Kryptographie sein

- Mark Ketchen, IBM Research, 2012, im Zusammenhang mit Quanten-Computing: "Were actually doing things that are making us think like, 'hey this isn't 50 years off, this is maybe just 10 years off, or 15 years off.' It's within reach."
- Sprung nach 2022, oder 2027. Es gibt Quantencomputer.
- Shor's Algorithmus löst in Polynomialzeit:
  - Faktorisieren von ganzen Zahlen.
  - Diskrete Logarithmen in endlichen Körpern.
  - Diskrete Logarithmen auf elliptischen Kurven.
- Damit ist alle zur Zeit benutzte Kryptographie mit öffentlichen Schlüsseln gebrochen!!

# Alle Kryptographie muss Post-Quanten-Kryptographie sein

- Mark Ketchen, IBM Research, 2012, im Zusammenhang mit Quanten-Computing: "Were actually doing things that are making us think like, 'hey this isn't 50 years off, this is maybe just 10 years off, or 15 years off.' It's within reach."
- Sprung nach 2022, oder 2027. Es gibt Quantencomputer.
- Shor's Algorithmus löst in Polynomialzeit:
  - Faktorisieren von ganzen Zahlen.
  - Diskrete Logarithmen in endlichen Körpern.
  - Diskrete Logarithmen auf elliptischen Kurven.
- Damit ist alle zur Zeit benutzte Kryptographie mit öffentlichen Schlüsseln gebrochen!!
- Grovers Algorithmus beschleunigt Angriffe mit vollständiger Suche.
- Beispiel: Mit $2^{64}$ quanten Operationion bricht man AES-128.

# Alle Kryptographie muss Post-Quanten-Kryptographie sein

- Mark Ketchen, IBM Research, 2012, im Zusammenhang mit Quanten-Computing: "Were actually doing things that are making us think like, 'hey this isn't 50 years off, this is maybe just 10 years off, or 15 years off.' It's within reach."
- Sprung nach 2022, oder 2027. Es gibt Quantencomputer.
- Shor's Algorithmus löst in Polynomialzeit:
  - Faktorisieren von ganzen Zahlen.
  - Diskrete Logarithmen in endlichen Körpern.
  - Diskrete Logarithmen auf elliptischen Kurven.
- Damit ist alle zur Zeit benutzte Kryptographie mit öffentlichen Schlüsseln gebrochen!!
- Grovers Algorithmus beschleunigt Angriffe mit vollständiger Suche.
- Beispiel: Mit $2^{64}$ quanten Operationion bricht man AES-128.
- Das Internet muss auf Post-Quanten-Verschlüsselung umgestellt werden.

# Vertrauen wächst nur langsam

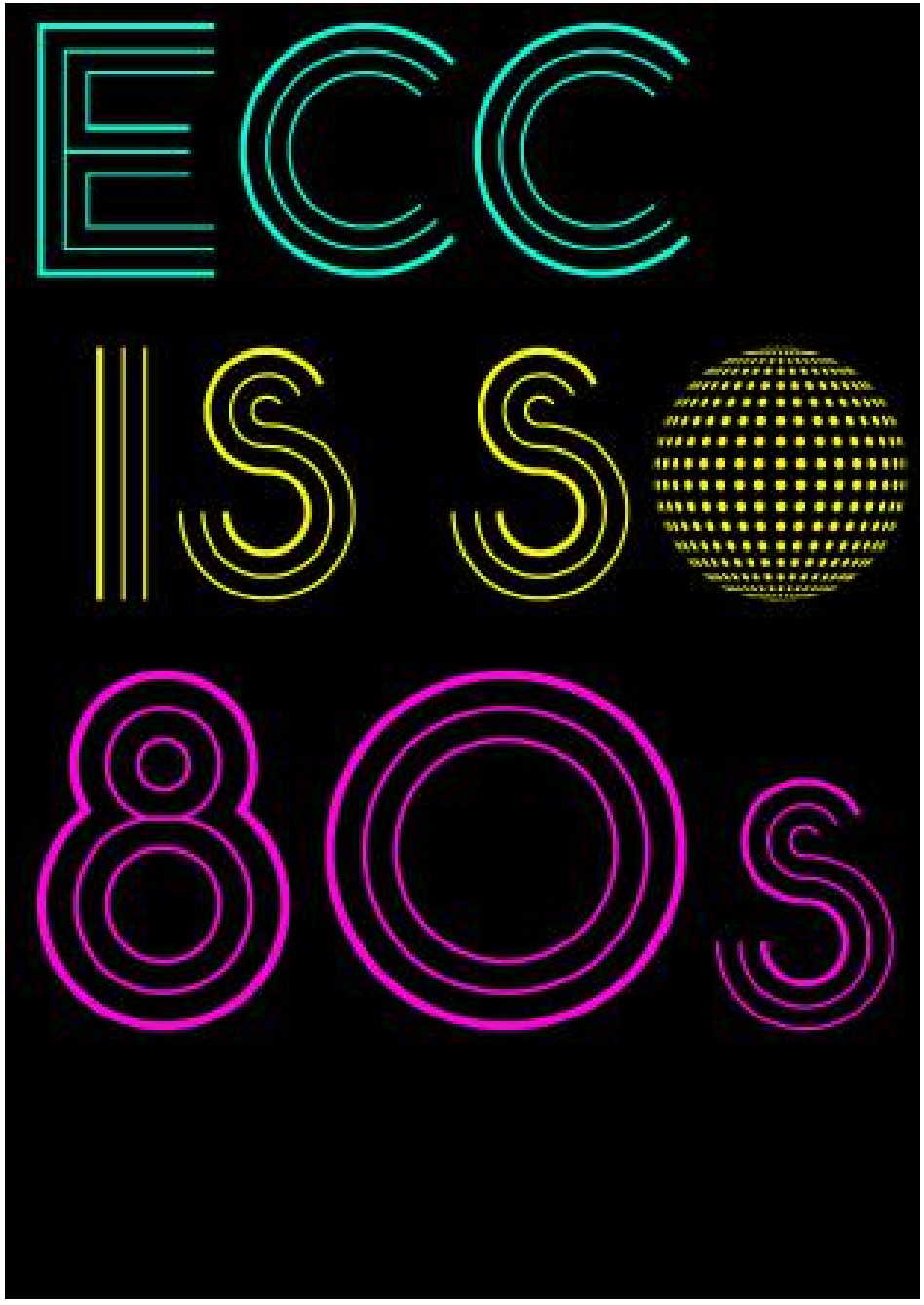
- Zwischen kryptographischem Design und Einsatz braucht es viel Forschung:
  - Untersuche vieler verschiedener Systeme.
  - Finde effiziente Algorithmen für Angriffe.
  - Konzentration auf sichere Kryptosysteme.

# Vertrauen wächst nur langsam

- Zwischen kryptographischem Design und Einsatz braucht es viel Forschung:
  - Untersuche vieler verschiedener Systeme.
  - Finde effiziente Algorithmen für Angriffe.
  - Konzentration auf sichere Kryptosysteme.
  - Finde effiziente Algorithmen für Nutzer.
  - Untersuche Implementierungen auf echter Hardware.
  - Untersuche Seitenkanalangriffe, Fehler-Angriffe, usw.
  - Konzentration auf sichere, verlässliche Implementierungen.
  - Konzentration auf Implementierungen, die schnell/klein genug sind.
  - Integriere die neuen Systeme sicher in Anwendungen.

# Vertrauen wächst nur langsam

- Zwischen kryptographischem Design und Einsatz braucht es viel Forschung:
  - Untersuche vieler verschiedener Systeme.
  - Finde effiziente Algorithmen für Angriffe.
  - Konzentration auf sichere Kryptosysteme.
  - Finde effiziente Algorithmen für Nutzer.
  - Untersuche Implementierungen auf echter Hardware.
  - Untersuche Seitenkanalangriffe, Fehler-Angriffe, usw.
  - Konzentration auf sichere, verlässliche Implementierungen.
  - Konzentration auf Implementierungen, die schnell/klein genug sind.
  - Integriere die neuen Systeme sicher in Anwendungen.
- Beispiel: ECC wurde in **1985** eingeführt und hat große Vorteile über RSA.
  Erst jetzt, in **2015**, werden robuste Implementierungen fürs Internet eingeführt.
- Forschung in Post-Quanten Kryptographie kann nicht warten, bis es Quantencomputer gibt!


ECC
IS
80s

# Umstieg für vertrauliche Dokumente noch dringender

- Wir wissen, dass alle heutige Kommunikation gespeichert wird (z.B. in Utah) und Quantencomputer köennen später alles brechen – und finden was wir heute geschrieben haben. Das ist ein Problem für medizische Daten, Journalisten, Polizei- und Gerichtsberichte, Staatsgeheimisse, ...


# Post-Quanten Kryptographie für Langzeitsicherheit

- EU-Projekt in Horizon 2020.
- Anfangsdatum 1. März 2015, Laufzeit 3 Jahre.
- 11 Partner aus Industrie und Wissenschaft, TU/e ist Koordinator













# Erwartungen an PQCRYPTO

- Post-Quanten Systeme gibt es schon, aber sie sind nicht hinreichend erforscht. Wir können jetzt zwar Empfehlungen aussprechen, aber diese Systeme sind groß oder langsam (oder beides)

# Erwartungen an PQCRYPTO

- Post-Quanten Systeme gibt es schon, aber sie sind nicht hinreichend erforscht. Wir können jetzt zwar Empfehlungen aussprechen, aber diese Systeme sind groß oder langsam (oder beides) darum haben ein passendes Logo gefunden.

# Erwartungen an PQCRYPTO

- Post-Quanten Systeme gibt es schon, aber sie sind nicht hinreichend erforscht. Wir können jetzt zwar Empfehlungen aussprechen, aber diese Systeme sind groß oder langsam (oder beides) darum haben ein passendes Logo gefunden.
- PQCRYPTO wird ein Portfolio an sicheren Post-Quanten-Systemen erstellen, die Effizienz dieser Systeme verbessern, und wird sie an die verschiedenen Bedürfnisse von Mobil-Telefonen, Clouds und dem Internet anpassen.
- PQCRYPTO wird effiziente und sichere Implementatierungen dieser systeme für eine Reihe an Anwendungen und CPUs erstellen und öffentlich zu Verfügung stellen.

# Arbeitspakete

Technische Arbeitspakete

- WP1: Post-quanten cryptography for small devices
  Leiter: Tim Güneysu, Vize: Peter Schwabe
- WP2: Post-quanten cryptography for the Internet
  Leiter: Daniel J. Bernstein, Vize: Bart Preneel
- WP3: Post-quanten cryptography for the cloud
  Leiter: Nicolas Sendrier, Vize: Lars Knudsen

Nicht-technische Arbeitspakete

- WP4: Management and dissemination
  Leiter: Tanja Lange
- WP5: Standardization
  Leiter: Walter Fumy

# WP1: Post-quanten cryptography for small devices

Leiter: Tim Güneysu, Vize: Peter Schwabe

- Suche nach Post-Quanten Systemen, die in das Energie- und Speicher-Budget von kleinen Geräten passen (z.B. auf Chipkarten mit 8-Bit, 16-Bit, oder 32-Bit Architecturen, verschiedenen Speichergrößen, mit oder ohne Koprozessor).
- Schreibe effiziente Implementierungen für diese Systeme.
- Untersuche und verbessere die Sicherheit gegen Implementierungsangriffe (Seitekanal o.ä.).
- Die Deliverables beinhalten Referenzimplementierungen und optimierte Implementierungen für die ganze Bandbreite an Prozessoren – von kleinen 8-Bit Microcontrollern bis hin so 32-Bit ARM Prozessoren.
- Die Deliverables enthalten auch FPGA und ASIC Designs und eine Analyse der physikalischen Sicherheit.

# WP2: Post-quanten cryptography for the Internet

Leiter: Daniel J. Bernstein, Vize: Bart Preneel

- Suche nach Post-Quanten Systemen, die für großeInternetsever geeignet sind, die gleichzeitig viele verschienene Verbindungen aufrechterhalten müssen.
- Entwickele sichere und effiziente Implementierungen.
- Binde diese Systeme in Internet-Protokolle ein.
- Die Deliverables enthalten eine Software Bibliothek für alle üblichen Internet-Plattformen, z.B. Prozessoren von großen Servern, kleinere PC und Laptop CPUs, Netbook CPUs (Atom, Bobcat, usw.), und Smartphone CPUs (ARM).
- Das Ziel ist es, hochsichere Post-Quanten-Kryptographie für das Internet bereitzustellen.

# WP3: Post-quanten cryptography for the cloud

Leiter: Nicolas Sendrier, Vize: Lars Knudsen

- Garantiere 50 Jahre an Sicherheit für Daten auf Cloud Servern, selbst wenn die Cloud Provider nicht vertrauenswürdig sind.
- Mache es möglich, dass Nutzer Daten in Cloud Servern teilen und gemeinsam editieren können, unter Nutzer-definierten Sicherheitsstandards.
- Unterstütze fortgeschrittene Cloud Anwendungen wie Suche in veschlüsselten Daten.
- Die Arbeit enthält public-key und symmetrische Kryptographie.
- Hier ist hohe Sicherheit und Geschwindigkeit wichtiger als die Schlüssellänge.

# Auswirkungen von PQCRYPTO

- Empfehlungen von Experten für post-quanten sichere Systeme.
- Die empfohlenen Systeme werden schneller und kleiner durch die PQCRYPTO Forschung.
- Mehr Benchmarking für bessere Vergleiche.
- Krypto-Bibliotheken werden öffentlich bereitgestellt für viele verschiedene Architekturen.
- Workshop und "summer" school zu Post-Quanten-Kryptographie (Spring or summer 2017).
- Mehr Info gibt es online unter http://pqcrypto.eu.org/ und auf twitter https://twitter.com/pqc_eu.

# Stand der Technik in Post-Quanten Verschlüsselung

- Codierungstheorie-basiert: z.B., 1978 McEliece.
  - Angreifer versucht, Fehler in einem zufällig aussehenden Code zu korrigieren.
  - Welche Codes sind gut? Man könnte von Reed–Solomon Codes ausgehen; Scaling hinzufügen? Permutationen? Puncturing? Subcodes? Unterkörper? Wildness? Listdecoding? Höheres Geschlecht für Kurven? Oder LDPC? MDPC? QC-MDPC? QD-MDPC? Rankmetrik? ...
  - Eine Auswahl an Artikeln zu Angriffen:
    1962 Prange; 1981 Omura; 1988 Lee–Brickell; 1988 Leon; 1989 Krouk; 1989 Stern; 1989 Dumer; 1990 Coffey–Goodman; 1990 van Tilburg; 1991 Dumer; 1991 Coffey–Goodman–Farrell; 1993 Chabanne–Courteau; 1993 Chabaud; 1994 van Tilburg; 1994 Canteaut–Chabanne; 1998 Canteaut–Chabaud; 1998 Canteaut–Sendrier; 2008 Bernstein–Lange–Peters; 2009 Bernstein–Lange–Peters–van Tilborg; 2009 Bernstein (post-quanten); 2009 Finiasz–Sendrier; 2010 Bernstein–Lange–Peters; 2011 May–Meurer–Thomae; 2011 Becker–Coron–Joux; 2012 Becker–Joux–May–Meurer; 2013 Bernstein–Jeffery–Lange–Meurer (post-quanten); 2015 May–Ozerov.
  - Wir haben Vertrauen in McEliece (mit größeren Schlüsseln als 1978), aber die Schlüssel sind wirklich groß .
  - QC-MDPC: hat kleinere Schlüssel, aber nicht genug Forschung.
  - Seiten-Kanal-Angriffe? Kompliziertere Protokolle? ...
- Gitter-basierte Verschlüsselung ist noch komplexer.

# Lineare Codes (Folien zum Teil von Tung Chou)

Ein binärer linearer Code $C$ der Länge $n$ und Dimension $k$ ist ein $k$-dimensionaler Unterraum von $\mathbb{F}_2^n$.

$C$ ist normalerweise gegeben als

- Zeilenraum der Generatormatrix $G \in \mathbb{F}_2^{k \times n}$

$$C = \{\mathbf{m}G | \mathbf{m} \in \mathbb{F}_2^k\}$$

- oder als Kern der Paritätsprüfmatrix $H \in \mathbb{F}_2^{(n-k) \times n}$

$$C = \{\mathbf{c} | H\mathbf{c}^\intercal = 0, \ \ \mathbf{c} \in \mathbb{F}_2^n\}$$

Beispiel:

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$\mathbf{c} = (111)G = (10011)$ ist ein Codewort.

# Gewicht, Abstand, Decodier-Problem

- Das Hamming-Gewicht eines Wortes ist die Anzahl der nicht-Null Koordinaten.
- Der Hamming-Abstand zwischen zwei Wörtern in $\mathbb{F}_2^n$ ist die Anzahl der Koordinaten, in denen sie verschieden sind.

Decodierungs-Problem: Finde das nächstgelegene Codeword $\mathbf{c} \in C$ zu einem gegebenen $\mathbf{r} \in \mathbb{F}_2^n$ (wenn eindeutig). Sei $\mathbf{r} = \mathbf{c} + \mathbf{e}$, dann ist das Finden von $\mathbf{e}$ ein äquivalentes Problem.

- $\mathbf{e}$ wird der Fehler-Vektor genannt.
- Es gibt viele Code-Familen mit effizienten Decodier-Algorithmen, z.B. Reed–Solomon Codes, Goppa Codes/alternierende Codes, usw.
- Aber im Allgemeinen ist das Decodierungs-Problem hart: Information-set decoding braucht exponentiell viel Zeit (in $n$).

# Binäre Goppa Codes

Binäre Goppa Codes werden oft wie folgt definiert:

- Eine Liste $L = (a_1, \ldots, a_n)$ mit $a_i \neq a_j$ und $a_i \in \mathbb{F}_q$, genannt der Träger.
- ein quadartfreies Polynom $g(x) \in \mathbb{F}_q[x]$ vom Grad $t$ so dass $g(a) \neq 0$ für alle $a \in L$. $g(x)$ heißt das Goppa Polynom.

Der dazugehörende Goppa Code, $\Gamma(L, g)$, ist die Menge $c = (c_1, \ldots, c_n) \in \mathbb{F}_2^n$ für die gilt:

$$\frac{c_1}{x - a_1} + \frac{c_2}{x - a_2} + \cdots + \frac{c_n}{x - a_n} \equiv 0 \pmod{g(x)}$$

- $\Gamma(L, g)$ kann $t$ Fehler korrigieren.
- $\Gamma(L, g)$ wird in Codierungstheorie-basierter Kryptographie benutzt.

# Das Niederreiter-Kryptosystem

1986 von Harald Niederreiter als Variante des McEliece-Systems vorgeschlagen.

- Public Key: Paritätsprüfmatrix $K \in \mathbb{F}_q^{(n-k)\times n}$ für einen binären Goppa Code – aber ohne das Wissen von Träger und Goppapolynom.
- Verschlüsselung: Klartext $\mathbf{m}$ ist ein $n$-Bit Vector vom Gewicht $t$. Der Ciffretext $\mathbf{c}$ ist ein Vektor von $(n-k)$ Bits:

$$\mathbf{c}^\mathsf{T} = K\mathbf{m}^\mathsf{T}.$$

- Entschlüsselung: Finde einen $n$-Bit Vektor $\mathbf{r}$, so dass

$$\mathbf{c}^\mathsf{T} = K\mathbf{r}^\mathsf{T}$$

gilt, dann decodiere zu $\mathbf{r}$. **Entschlüsseln ist quasi decodieren**.

- Ein passiver Angreifer muss das $t$-Fehler-Korrektur-Problem lösen für den zufällig-aussehenden öffentlichen Schlüssel.

# McBits

Daniel J. Bernstein, Tung Chou, Peter Schwabe, CHES 2013.

- Verschlüsselung ist sowieso superschnell – nur eine Vektor-Matrix-Multiplikation.
- Hauptschritt in der Entschlüsselung ist das Decodieren des Goppa Codes. Die McBits-Software erreicht das in konstanter Laufzeit.
- Enschlüsseln bei $2^{128}$ pre-quanten Sicherheit: $(n; t) = (4096; 41)$ braucht 60493 Ivy Bridge Takte.
- Enschlüsseln bei $2^{263}$ pre-quanten Sicherheit: $(n; t) = (6960; 119)$ braucht 306102 Ivy Bridge Takte.
- Die Beschleunigung durch Grover bedeutet weniger als eine Halbierung des Sicherheitsniveaus, also bringen die letzten Parameter mindestens $2^{128}$ Post-Quanten-Sicherheit.

# SPHINCS: practical stateless hash-based signatures

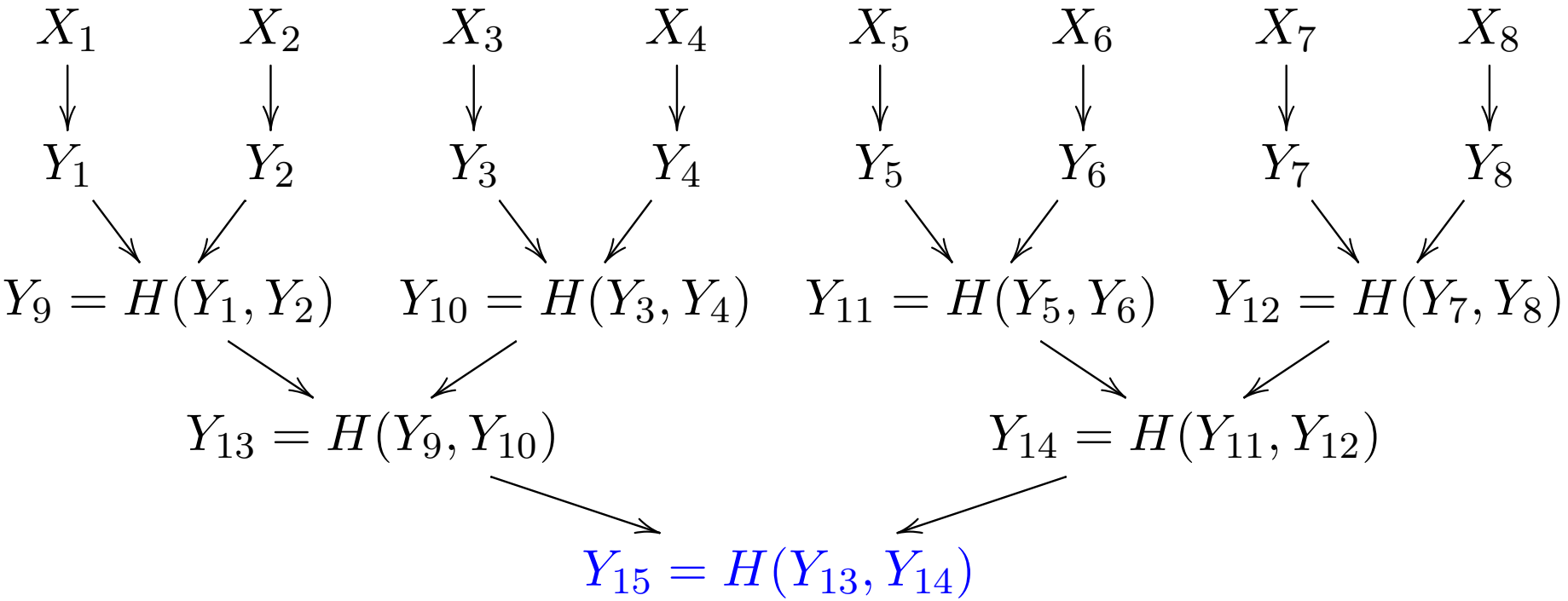
Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing,
Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou,
Peter Schwabe, Zooko Wilcox-O'Hearn

# Hash-basierte Signaturen

- 1979 Lamports Einmalsignaturen.
- Lege eine $k$-Bit Einwegfunktion $G : \{0,1\}^k \to \{0,1\}^k$ und eine Hashfunktion $H : \{0,1\}^* \to \{0,1\}^k$ fest.
- Der geheime Schlüssel ist $X$: $2k$ Strings von jeweils $k$ Bits $x_1[0], x_1[1], \ldots, x_k[0], x_k[1]$. Insgesamt $2k^2$ Bits.
- Der öffentliche Schlüssel ist $Y$: $2k$ Strings von jeweils $k$ Bits $y_1[0], y_1[1], \ldots, y_k[0], y_k[1]$, die $y_i[b] = G(x_i[b])$ erfüllen. Insgesamt $2k^2$ Bits.
- Signatur $S(X, r, m)$ zu einer Nachricht $m$: $r, x_1[h_1], \ldots, x_k[h_k]$ wobei $H(r, m) = (h_1, \ldots, h_k)$.
- Man darf niemals den geheimen Schlüssel für mehr als eine Signatur benutzen.
- Normalerweise wählt man $G = H$ (auf $k$ Bits gekürzt).
- 1979 hat Merkle das Verfahren auf mehrere Signaturen verallgemeinert.

# 8-mal Merkle Hash-Tree

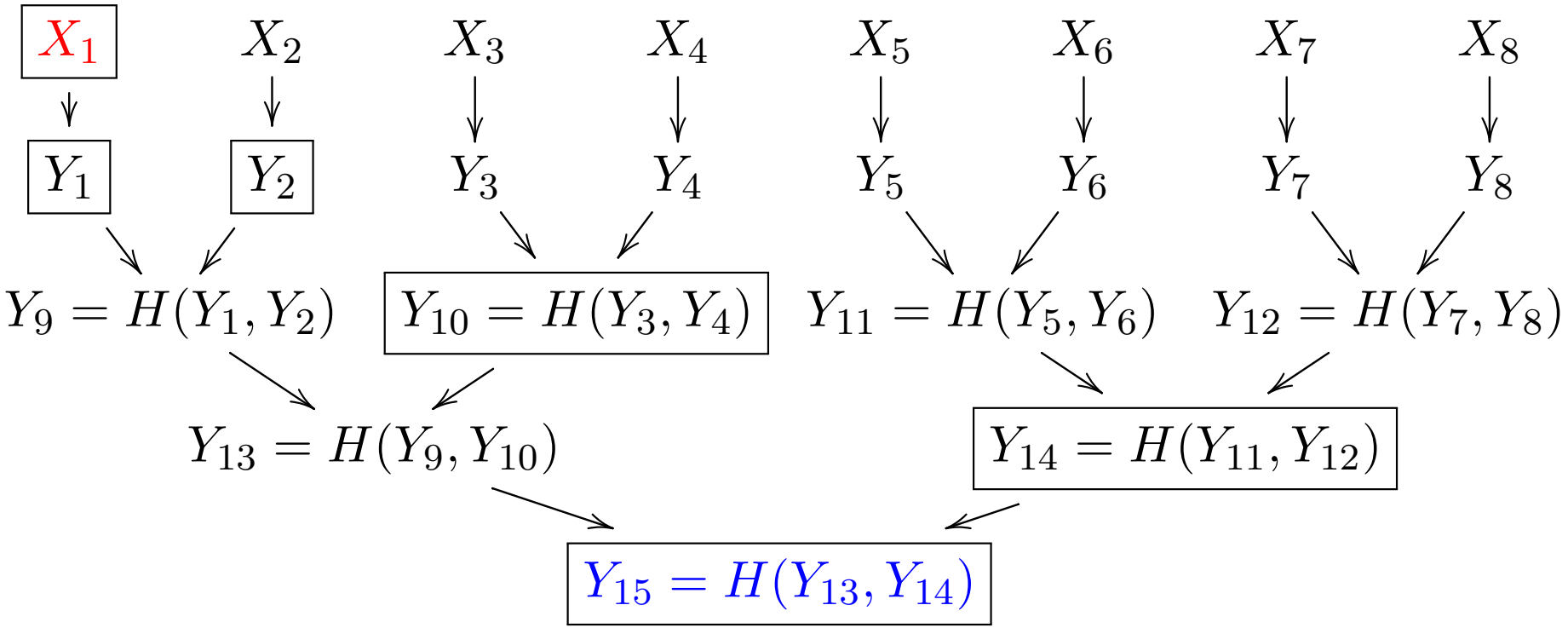
Acht Lamport Einmalschlüssel $Y_1, Y_2, \ldots, Y_8$ mit dazugehöhrenden $X_1, X_2, \ldots, X_8$, wobei $X_i = (x_{i,1}[0], x_{i,1}[1], \ldots, x_{i,k}[0], x_{i,k}[1])$ und $Y_i = (y_{i,1}[0], y_{i,1}[1], \ldots, y_{i,k}[0], y_{i,k}[1])$.




Der öffentliche Schlüssel ist $Y_{15}$.

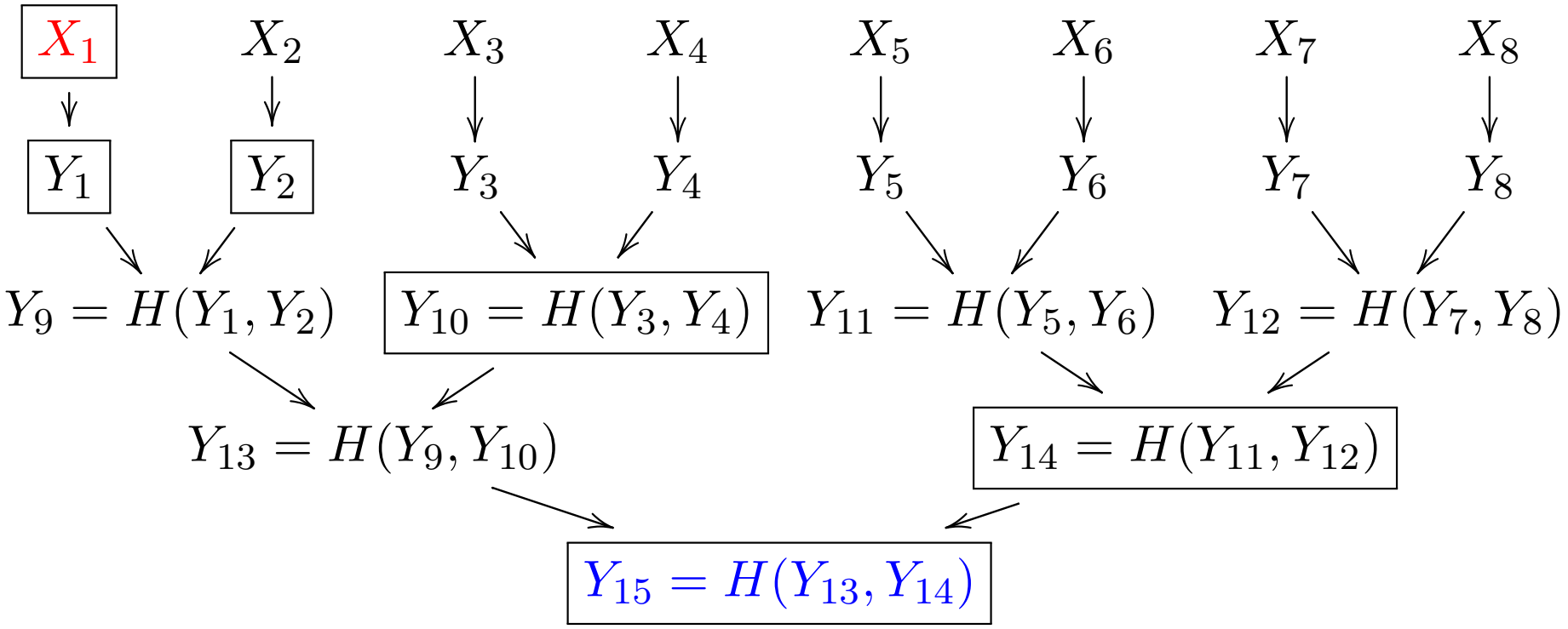
# Signatur im 8-mal Merkle Hash-Tree

Die erste Nachricht hat Signatur $(S(X_1, r, m), Y_1, Y_2, Y_{10}, Y_{14})$.



# Signatur im 8-mal Merkle Hash-Tree

Die erste Nachricht hat Signatur $(S(X_1, r, m), Y_1, Y_2, Y_{10}, Y_{14})$.

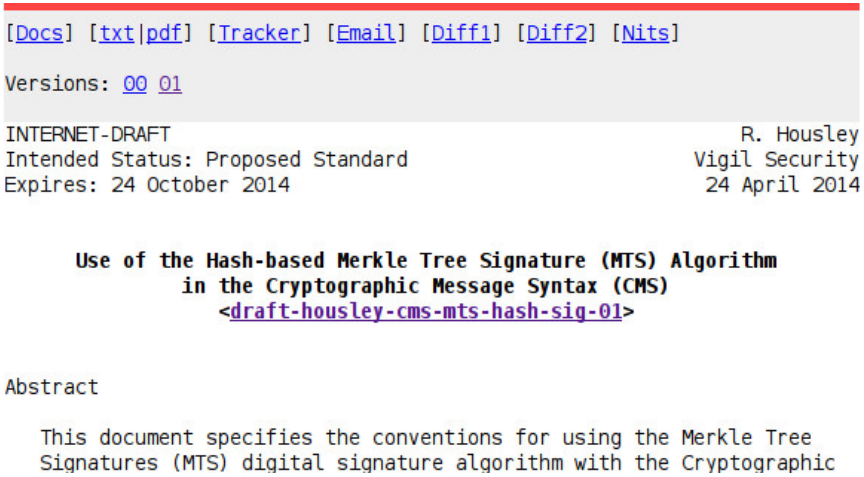



Dazu prüft man die Signatur $S(X_1, r, m)$ auf $m$ gegen $Y_1$ und verlinkt $Y_1$ gegen den öffentlichen Schlüssel $Y_{15}$, indem man $Y'_9 = H(Y_1, Y_2)$ und $Y'_{13} = H(Y'_9, Y_{10})$ berechnet und dann $H(Y'_{13}, Y_{14})$ mit $Y_{15}$ vergleicht.

# Pros und cons

Pros:

- Post-quanten sicher
- Braucht nur eine sichere Hashfunktion
- Kleiner öffentlicher Schlüssel
- Sicherheit gut im Griff
- Schnell
- Bereits für Standards vorgeschlagen http://tools.ietf.org/html/draft-housley-cms-mts-hash-sig-01



[Docs] [txt|pdf] [Tracker] [Email] [Diff1] [Diff2] [Nits]

Versions: 00 01

INTERNET-DRAFT — R. Housley
Intended Status: Proposed Standard — Vigil Security
Expires: 24 October 2014 — 24 April 2014

**Use of the Hash-based Merkle Tree Signature (MTS) Algorithm in the Cryptographic Message Syntax (CMS)**
<draft-housley-cms-mts-hash-sig-01>

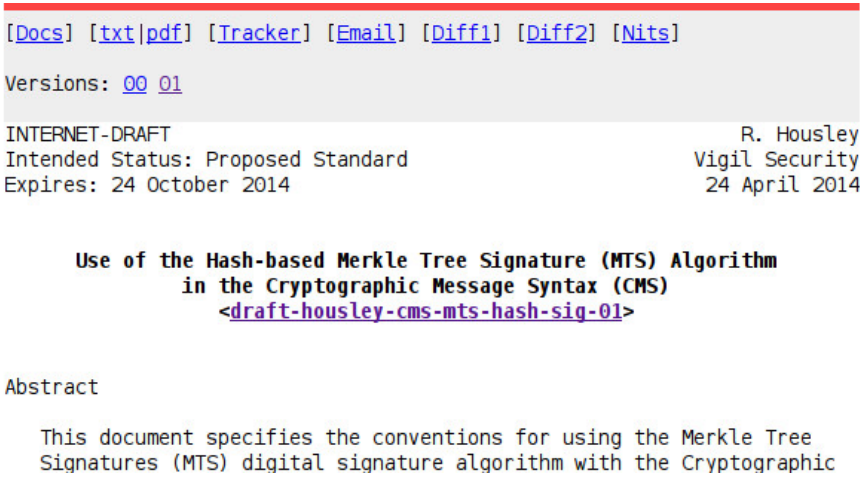
Abstract

This document specifies the conventions for using the Merkle Tree Signatures (MTS) digital signature algorithm with the Cryptographic

# Pros und cons

Pros:

- Post-quanten sicher
- Braucht nur eine sichere Hashfunktion
- Kleiner öffentlicher Schlüssel
- Sicherheit gut im Griff
- Schnell
- Bereits für Standards vorgeschlagen http://tools.ietf.org/html/draft-housley-cms-mts-hash-sig-01



[Docs] [txt|pdf] [Tracker] [Email] [Diff1] [Diff2] [Nits]

Versions: 00 01

INTERNET-DRAFT — R. Housley
Intended Status: Proposed Standard — Vigil Security
Expires: 24 October 2014 — 24 April 2014

**Use of the Hash-based Merkle Tree Signature (MTS) Algorithm in the Cryptographic Message Syntax (CMS)**
<draft-housley-cms-mts-hash-sig-01>

Abstract

This document specifies the conventions for using the Merkle Tree Signatures (MTS) digital signature algorithm with the Cryptographic

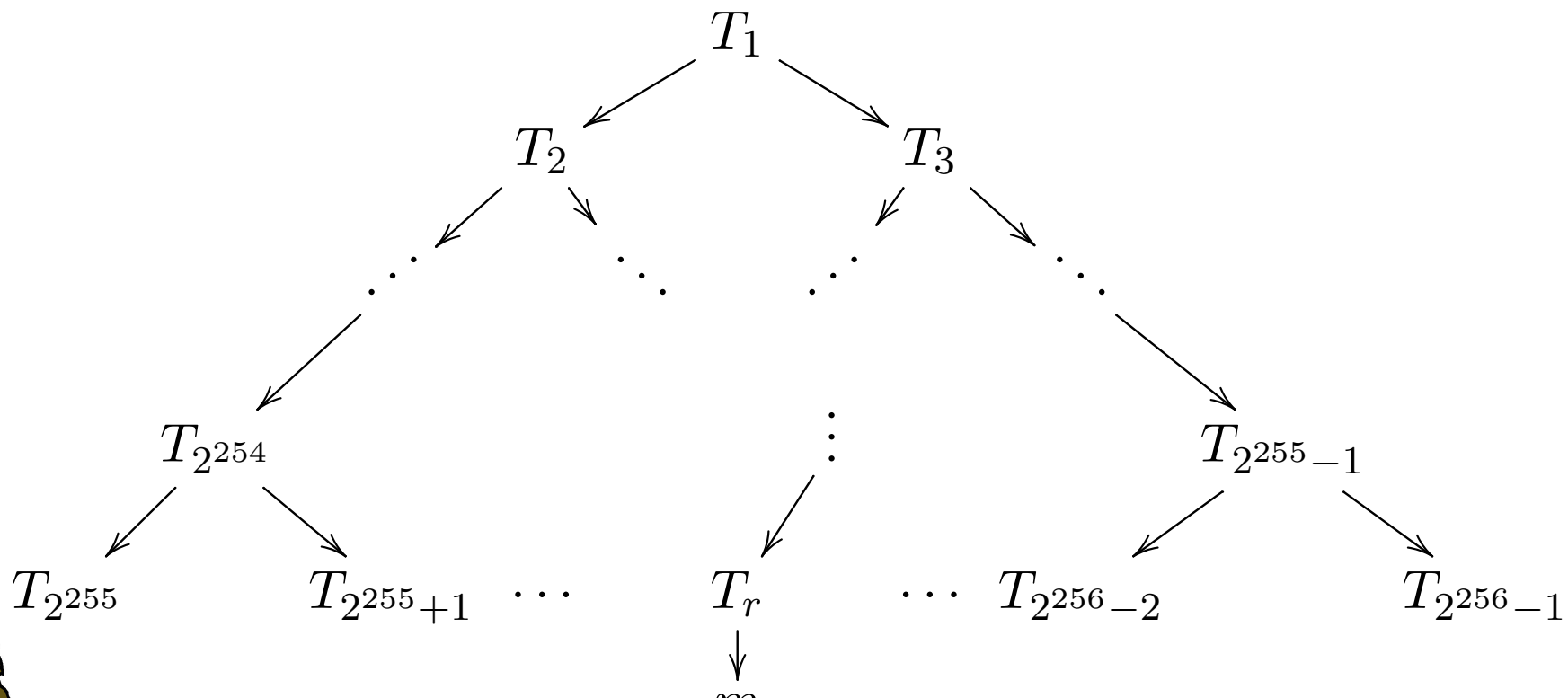
Cons:

- Große Signaturen und geheime Schlüssel
- Benötigt Wissen vom Zustand (stateful)
  Adam Langley "for most environments it's a huge foot-cannon."


ELIMINATE
THE STATE

# Große Bäume (1987 Goldreich), Schlüssel auf Abruf (Levin)

Signierer wählt zufällig $r \in \{2^{255}, 2^{255}+1, \ldots, 2^{256}-1\}$,
benutzt Einmalschlüssel $T_r$ um die Nachricht zu signieren;
benutzt Einmalschlüssel $T_i$ um $(T_{2i}, T_{2i+1})$ für $i < 2^{255}$ zu signieren.
Erzeugt $i$-ten geheimen Schlüssel als $H_k(i)$, wobei $k$ ein Hauptschlüssel ist.



# Tut, aber Signaturen sind sehr lang

0.6 MB für Hash-basierte Goldreich Signaturen mit Winternitz-16 Einmalsignaturen (kürzer als Lamport).

Würde in typischen Anwendungen den Verkehr dominieren, und führt für typische Netze zu so großer Verzögerung, dass Nutzer es merken.

# Tut, aber Signaturen sind sehr lang

0.6 MB für Hash-basierte Goldreich Signaturen mit Winternitz-16 Einmalsignaturen (kürzer als Lamport).

Würde in typischen Anwendungen den Verkehr dominieren, und führt für typische Netze zu so großer Verzögerung, dass Nutzer es merken.

Beispiel:
Debian erwartet häufige Updates.
Mindestens eine neue Signatur pro Upgrade.
Typisches Upgrade: nur ein oder ein paar Pakete.
1.2 MB mittlere Paketgröße.
mit 0.08 MB Median.

# Tut, aber Signaturen sind sehr lang

0.6 MB für Hash-basierte Goldreich Signaturen mit Winternitz-16 Einmalsignaturen (kürzer als Lamport).

Würde in typischen Anwendungen den Verkehr dominieren, und führt für typische Netze zu so großer Verzögerung, dass Nutzer es merken.

Beispiel:
Debian erwartet häufige Updates.
Mindestens eine neue Signatur pro Upgrade.
Typisches Upgrade: nur ein oder ein paar Pakete.
1.2 MB mittlere Paketgröße.
mit 0.08 MB Median.

Beispiel:
HTTPS schickt typischerweise mehrere Signaturen pro Seite.
Die durchschnittliche Seite in Alexa Top 1000000 hat 1.8 MB.

# Neu: SPHINCS-256

**Tragbare Größen.**
0.041 MB Signaturen.
0.001 MB öffentliche Schlüssel.
0.001 MB geheime Schlüssel.

**Tragbare Geschwindigkeiten.**
Benchmarks unserer public-domain Software auf Haswell:
51.1 Millionen Takte zum Signieren. (RSA-3072: 14.2 Millionen.)
1.5 Millionen Takte zum Verifizieren. (RSA-3072: 0.1 Millionen.)
3.2 Millionen Takte zur Schlüsselerzeugung. (RSA-3072: 950 Millionen.)

# Neu: SPHINCS-256

**Tragbare Größen.**
0.041 MB Signaturen.
0.001 MB öffentliche Schlüssel.
0.001 MB geheime Schlüssel.

**Tragbare Geschwindigkeiten.**
Benchmarks unserer public-domain Software auf Haswell:
51.1 Millionen Takte zum Signieren. (RSA-3072: 14.2 Millionen.)
1.5 Millionen Takte zum Verifizieren. (RSA-3072: 0.1 Millionen.)
3.2 Millionen Takte zur Schlüsselerzeugung. (RSA-3072: 950 Millionen.)

**Für $2^{128}$ Post-Quanten-Sicherheit gebaut,**
selbst wenn ein Nutzer mehr als $2^{50}$ Nachrichten signiert:
$2^{20}$ Nachrichten/Sekunde für mehr als $30$ Jahre.
Ja, wir haben quanten Angriffe analysiert.

# Zutaten für SPHINCS (und SPHINCS-256)

Baumhöhe nur 60.

Blätter sind keine Einmalsignaturen sondern "Paarmalsignaturen".

Optimierte Signaturgröße *plus* Schlüssellänge.
Neue HORST "Paarmalsignatur", besser als HORS.

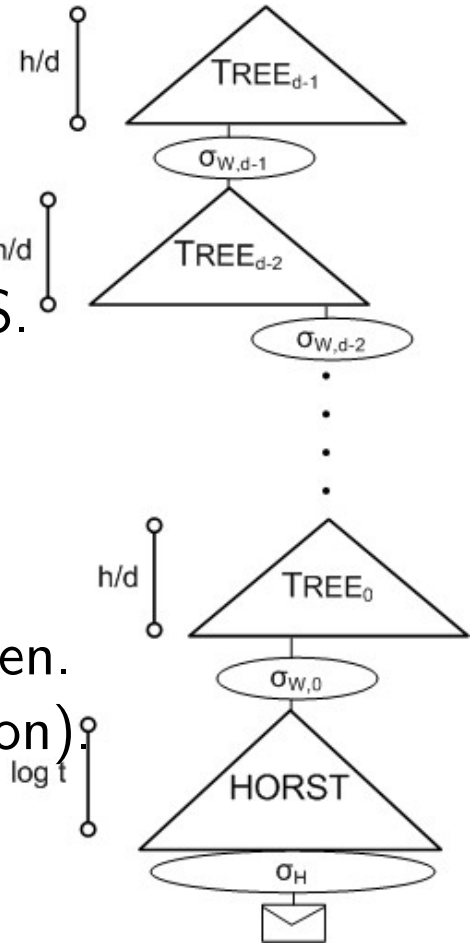
12-stufiger Hyper-Baum wie in GMSS.

Masken wie in XMSS und XMSS$^{MT}$,
um Beweise im Standard-Model zu bekommen.

Optimierte Hash-Geschwindigkeit für kurze Eingaben.
Benutze "sponge hash" (mit ChaCha12 Permutation).

Benutze schnelle Stromchiffre (wieder ChaCha12).

Vectorisiere die Hash-Software und ChaCha12.




Mehr Details im Paper: sphincs.cr.yp.to

# Initial recommendations of long-term secure post-quantum systems

Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe Bos,
Johannes Buchmann, Wouter Castryck, Orr Dunkelman,
Tim Güneysu, Shay Gueron, Andreas Hülsing,
Tanja Lange, Mohamed Saied Emam Mohamed,
Christian Rechberger, Peter Schwabe, Nicolas Sendrier,
Frederik Vercauteren, Bo-Yin Yang

# Erste Empfehlungen von PQCRYPTO

- **Symmetrische Verschlüsselung** Grover, 256-Bit Schlüssel:
  - AES-256
  - Salsa20 mit 256-Bit Schlüssel

  Forschung: Serpent-256, ...
- **Symmetrische Authentisierung** Informationstheoretische MACs (no Grover):
  - GCM mit 96-Bit nonce und 128-Bit Ausgabe
  - Poly1305
- **Public-key Verschlüsselung** McEliece mit binären Goppa Codes:
  - Länge $n = 6960$, Dimension $k = 5413$, $t = 119$ Fehler

  Forschung: QC-MDPC, Stehlé-Steinfeld NTRU, ...
- **Public-key Signaturen** Hash-basiert (minimale Annahmen):
  - XMSS mit Parametern aus dem CFRG Draft
  - SPHINCS-256

  Forschung: HFEv-, ...