secunet

Post-Quanten-Kryptographie

Auswirkungen auf TLS

Agenda

01 Grundlagen

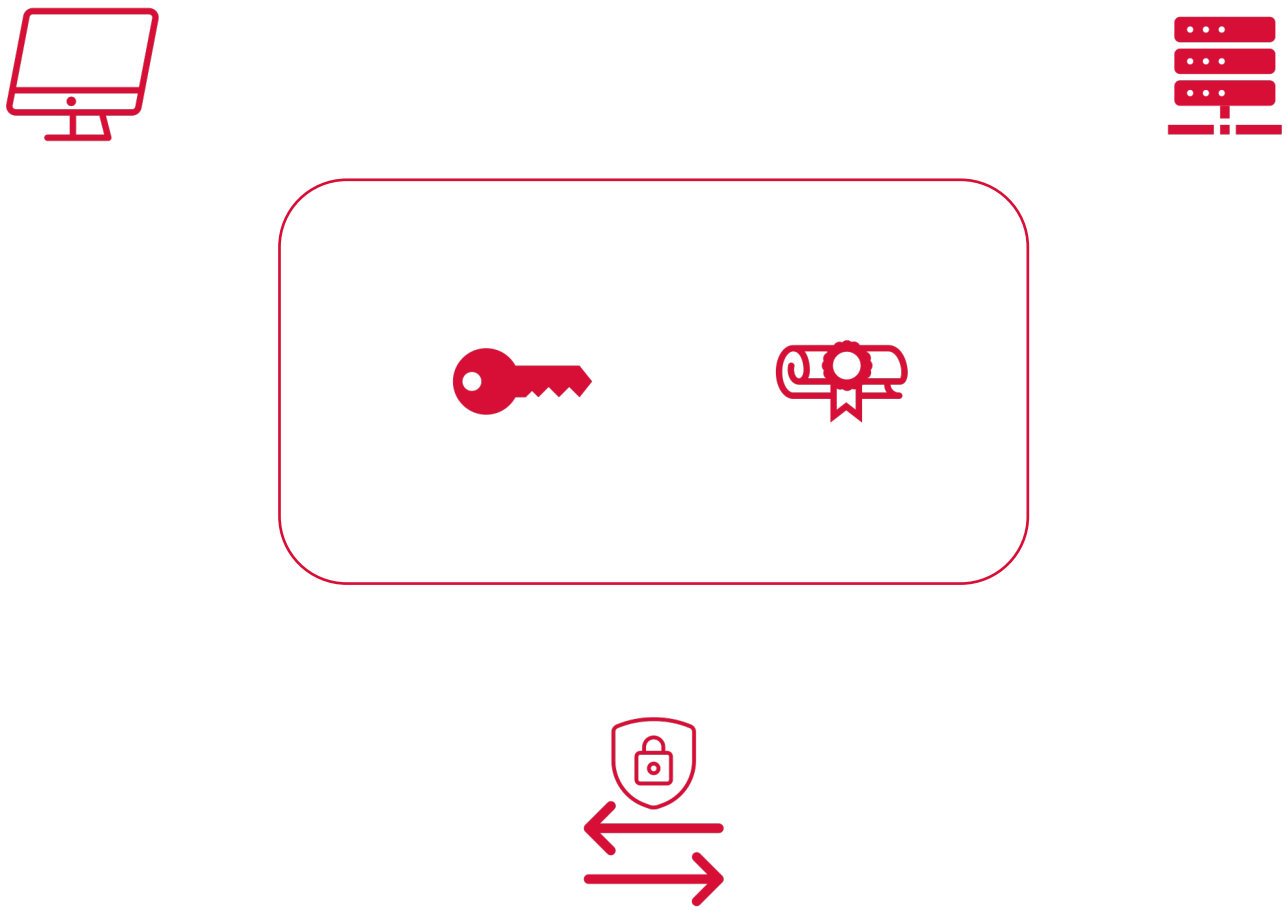
02 Schlüsselaustausch

03 Authentisierung

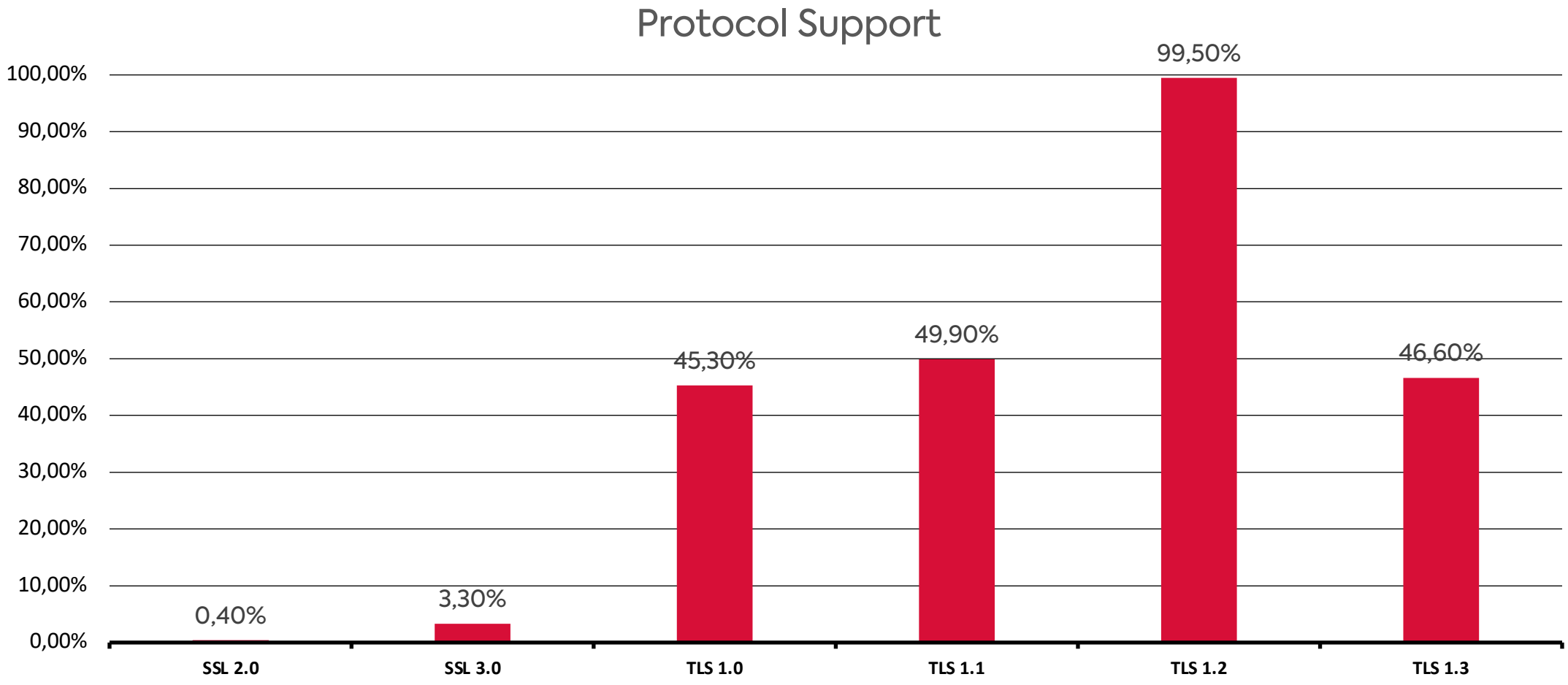
04 Auswirkungen kurz zusammengefasst

Agenda

Grundlagen
TLS

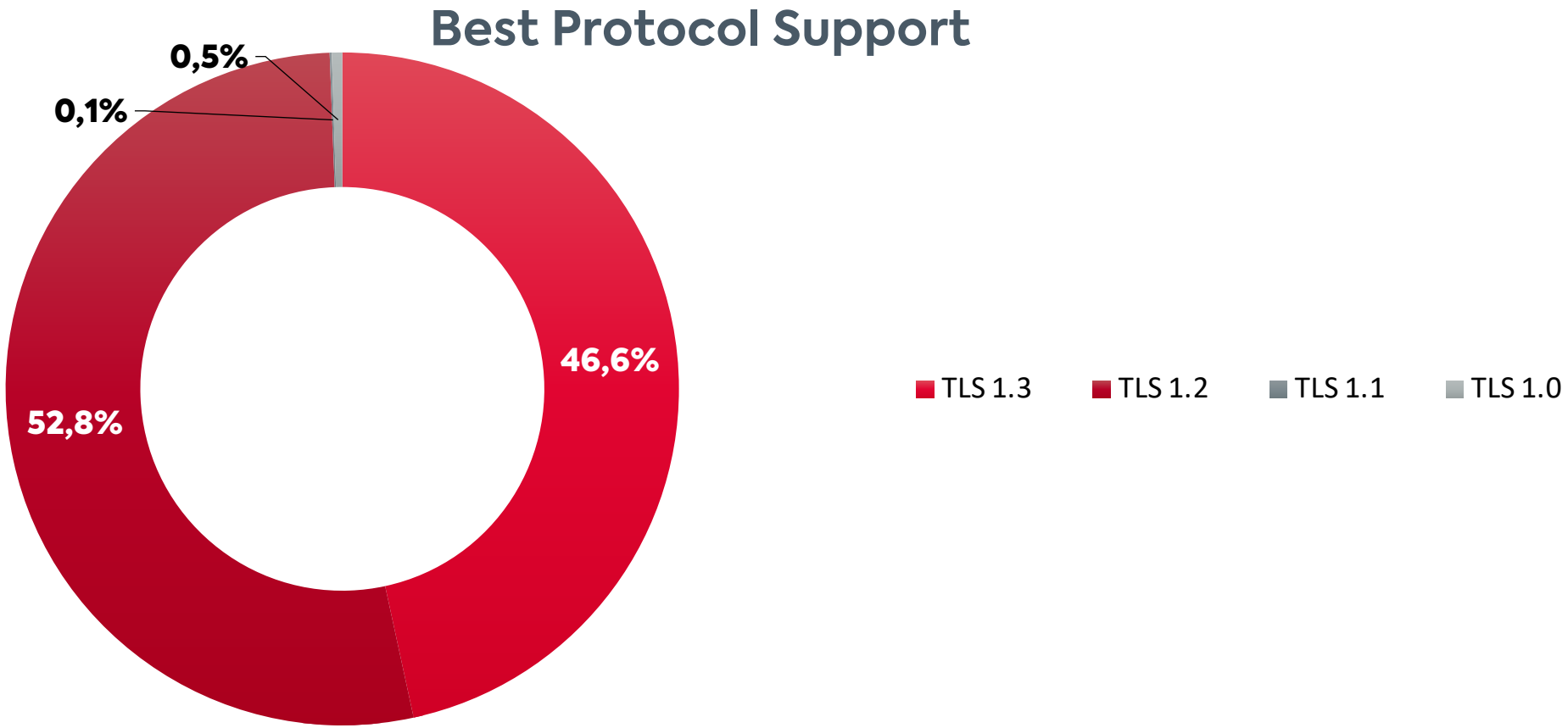


Grundlagen
Verbreitung TLS



www.ssllabs.com/ssl-pulse/

Grundlagen
Verbreitung TLS



www.ssllabs.com/ssl-pulse/

Grundlagen
Bedrohung Quantenalgorithmen

Shor-Algorithmus (1994)

- Effiziente Faktorisierung großer Zahlen (bricht RSA)
- Effiziente Berechnung des diskreten Logarithmus (bricht ECC)
- Polynomielle Laufzeit

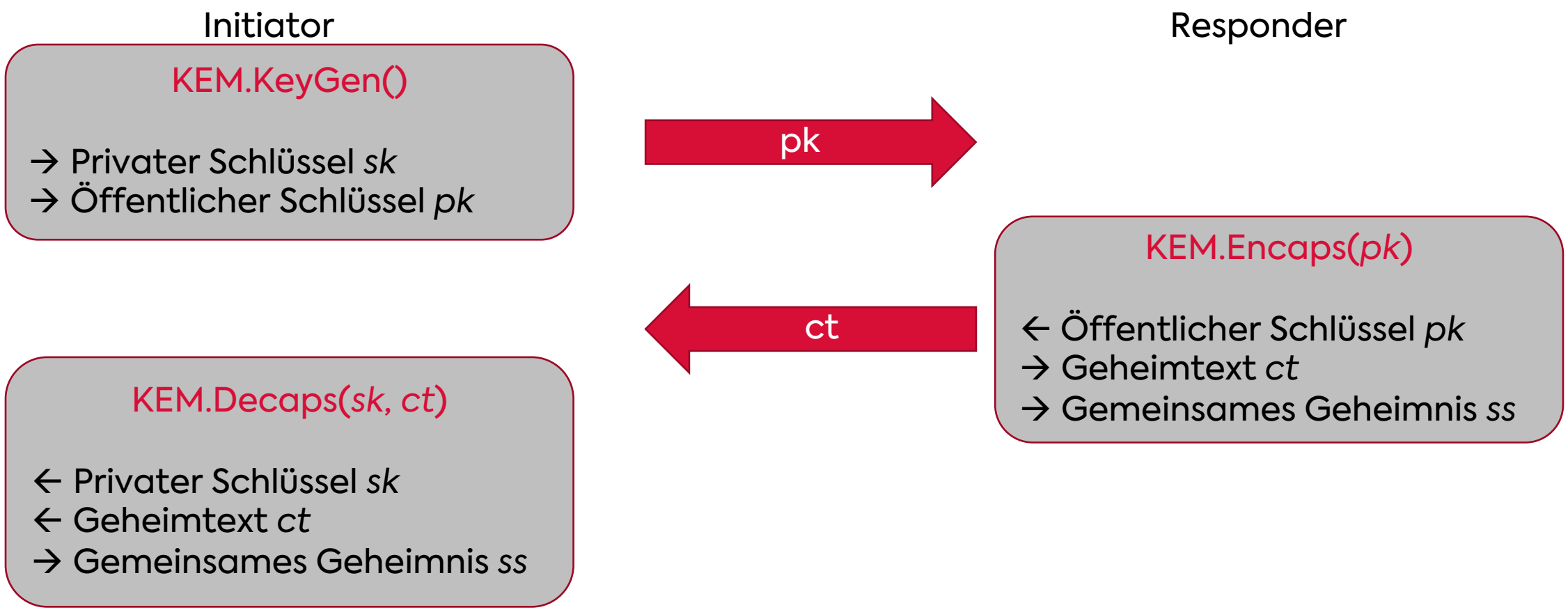
» **Neue quantensichere, asymmetrische Algorithmen werden benötigt**

Grover-Algorithmus (1996)

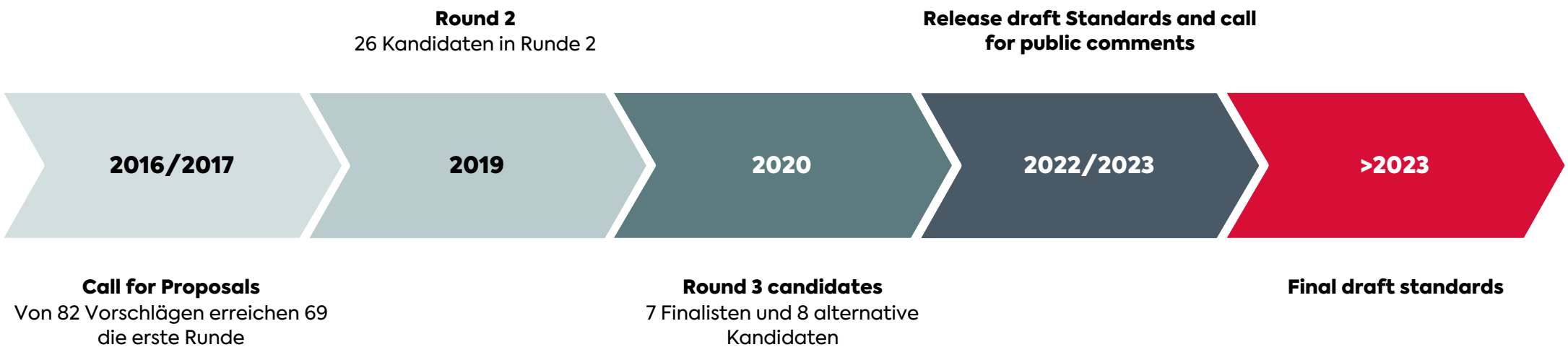
- Schnelle Suche in unsortierten Datenbanken der Größe N in Wurzel N Iterationen
- Halbiert Bitsicherheit symmetrischer Algorithmen

» **Verdoppeln der Länge der symmetrischen Schlüssel bzw. des Outputs von Hashfunktionen**

Grundlagen
Key Encapsulation Mechanism (KEM)



Grundlagen
NIST Standardisierungswettbewerb



Grundlagen
NIST Algorithmen

KEMs

Finalisten	Alternative Kandidaten
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

Signaturalgorithmen

Finalisten	Alternative Kandidaten
CRYSTALS-Dilithium	GeMSS
Falcon	Picnic
Rainbow	SPHINCS+

Agenda

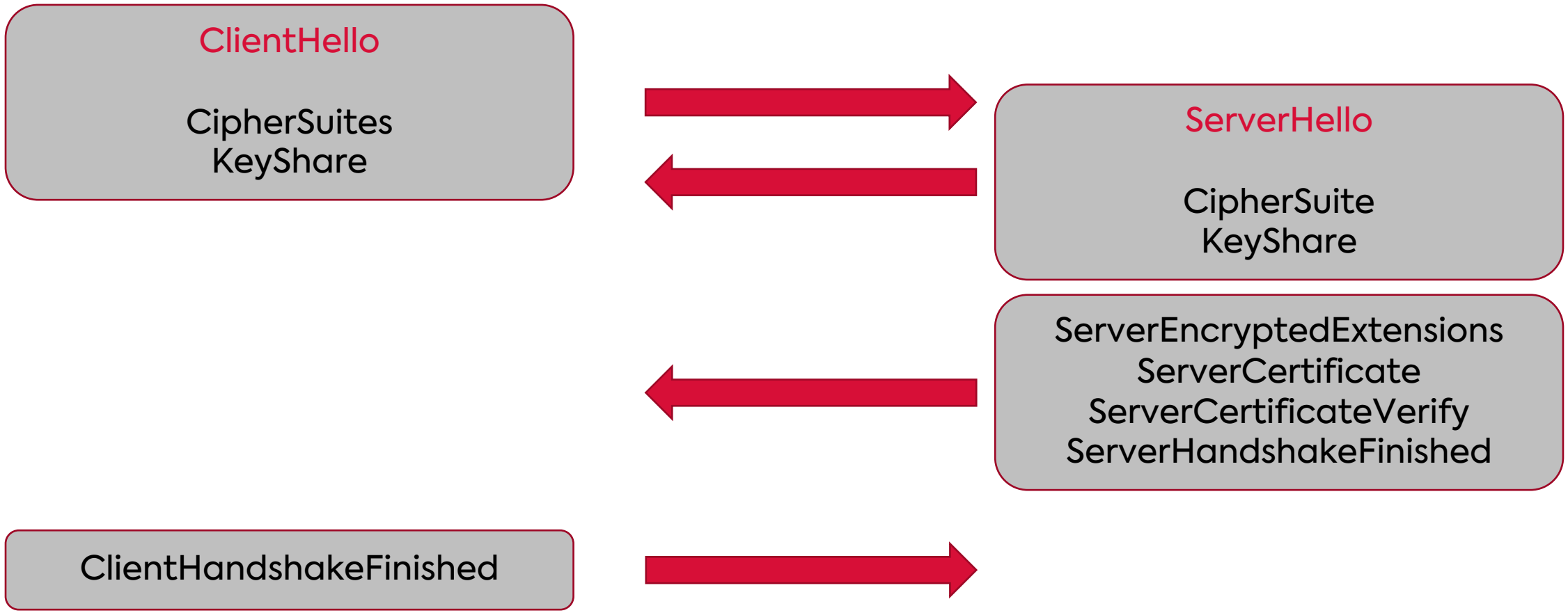
01 Grundlagen

02 **Schlüsselaustausch**

03 Authentisierung

04 Auswirkungen kurz zusammengefasst

Schlüsselaustausch TLS 1.3



Schlüsselaustausch
Hybrides Modell

Anforderungen

- Sicherheit
- Rückwärtskompatibilität
- Performanz
- Latenz

Designfragen

- Negotiation
- Übertragen öffentlicher Schlüssel und Ciphertexte
- Berechnung gemeinsames Geheimnis

Schlüsselaustausch Negotiation

- Jede Kombination wird als NamedGroup repräsentiert
- Jeder Wert steht für ein Algorithmenpaar
- Spezifische Werte sollten von IANA standardisiert werden

```
enum {
      /* Elliptic Curve Groups (ECDHE) */
      secp256r1(0x0017), secp384r1(0x0018),
      secp521r1(0x0019), x25519(0x001D),
      x448(0x001E),

      /* Finite Field Groups (DHE) */
      ffdhe2049(0x0100), ffdhe3072(0x0101),
      ffdhe4096(0x0102), ffdhe6144(0x0103),
      ffdhe8192(0x0104),

      /*Hybrid Key Exchange Methods */
      TBD(0xTBD), ...,

      /* Reserved Code Points */
      ffdhe_private_use(0x01FC..0x01FF),
      hybrid_private_use(0x2F00..0x2FFF),
      ecdhe_private_use(0xFE00..0xFEFF),
      (0xFFFF)
} NamedGroup;
```

Schlüsselaustausch
Übertragen öffentlicher Schlüssel und Ciphertexte

- KEM Schlüssel/Ciphertext wird als KeyShareEntry repräsentiert
- „key_Exchange“ Feld ist Konkatenation der key_Exchange Felder der Algorithmen
- Client key_Exchange: pk aus KeyGen()
- Server key_Exchange: ct aus Encaps()

```
struct {
     NamedGroup group;
     opaque key_Exchange<1...2^16-1>;
} KeyShareEntry;

struct {
     KeyShareEntry client_shares<0...2^16-1>;
} KeyShareClientHello;

struct {
     KeyShareEntry server_share;
} KeyShareServerHello;
```

Schlüsselaustausch
Berechnung gemeinsames Geheimnis

```
concatenated_shared_secret = shared_secret_1 || shared_secret_2
```

Schlüsselaustausch
Offene Fragen

- Große öffentliche Schlüssel/Ciphertexte
- Duplikation von KeyShares
- Resumption
- Fehlerrate

Schlüsselaustausch
Security Considerations

- Öffentliche Schlüssel, Ciphertexte, Geheimnisse sollten konstante Längen haben
- Wenn individuelle gemeinsame Geheimnisse fixe Länge haben, dann ist Konkatenationsansatz sicher
- KEMs müssen auch bei Key Re-use sicher sein (bspw. durch IND-CCA2-Sicherheit)

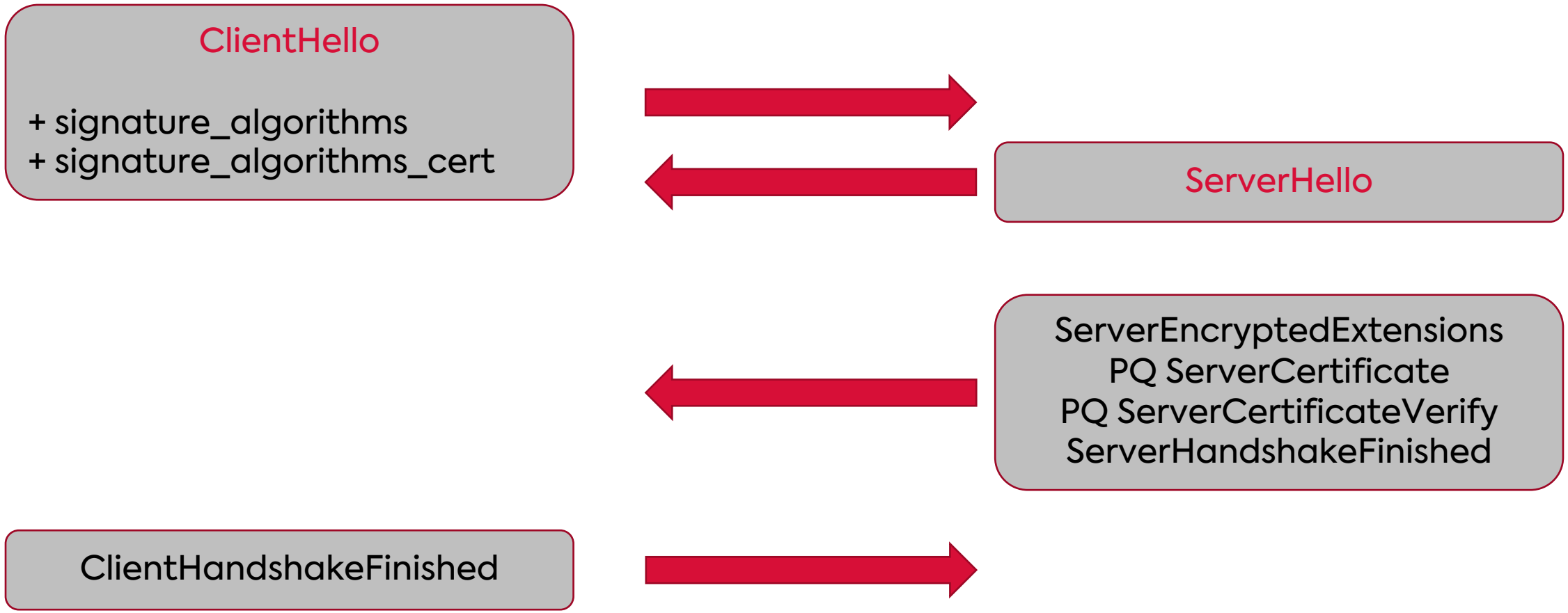
Schlüsselaustausch
Benchmarks – Constrained Devices

Algorithmen	Handshake Read (Bytes)	Handshake Write (Bytes)	Handshake Total (Bytes)	Speed (ms)
Kyber	3 897	1 543	5 440	89,82
FrodoKEM	18 552	15 991	34 544	295,81
Saber	3 897	1 351	5 248	87,69
NTRU	4 039	1 589	5 628	241,34
BIKE	7 773	5 323	13 096	199,66
SIKE	3 295	821	4 116	8 580,58

» Saber und Kyber eignen sich i.A. am besten. FrodoKEM für sicherheitskritische Anwendungen empfohlen.

Agenda

Authentisierung
TLS 1.3 mit PQ Zertifikat



Authentisierung Herausforderungen

- Zusätzliche Latenz
- Kommunikationsoverhead
- Einbindung existierender Hardware

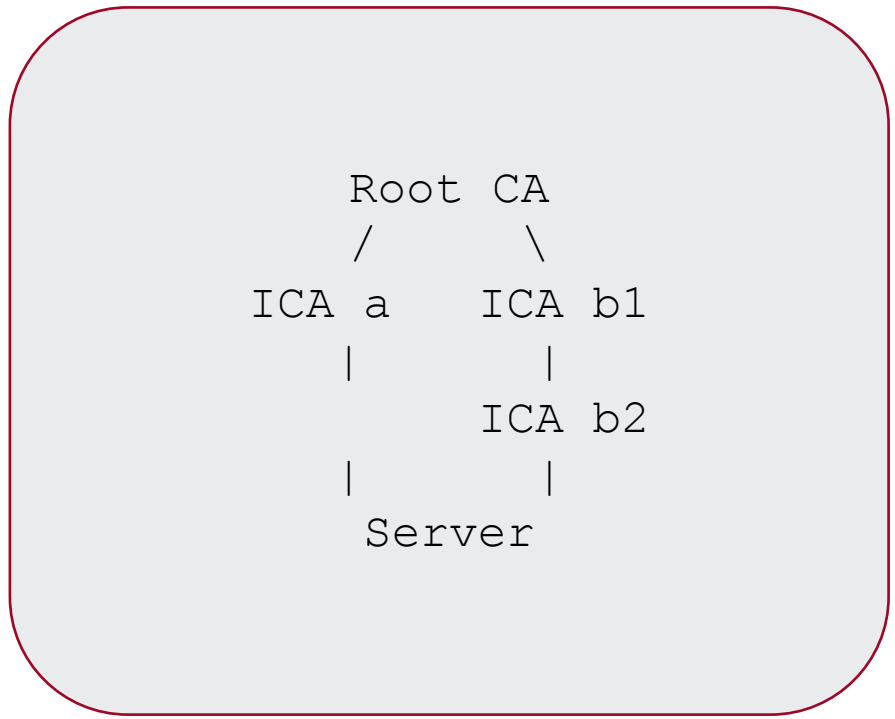
Authentisierung
Benchmarks Signaturalgorithmen

Algorithmen	Sign Mean lokal (ms)	Verify Mean lokal (ms)	Sign Mean cloud (ms)	Verify Mean cloud (ms)
RSA 3072	3,19	0,06	2,39	0,04
ECDSA 384	1,32	1,05	1,28	0,93
Dilithium II	0,82	0,16	0,41	0,12
Falcon 512	5,22	0,05	6,50	0,07
Picnic L1FS	4,09	3,25	3,17	2,39
SPHINCS+ SHA256_128-f-simple	93,37	3,92	62,7	2,50
Rainbow Ia	0,34	0,83	0,015	0,48

» Dilithium und Rainbow haben kompetetive Performanz. Falcon schnelle Verifikation aber langsames Signieren.

Authentisierung Zertifikatsgröße

Algorithmen	Eine ICA (kB)	Zwei ICA (kB)	CertificateVerify (kB)
RSA 3072	1,63	2,44	0,38
ECDSA 384	1,34	2,15	0,05
Dilithium II	6,90	10,42	2,04
Falcon 512	3,54	5,37	0,69
Picnic L1FS	66,20	99,57	30,03
SPHINCS+ SHA256_128-f-simple	34,46	51,74	16,98
Rainbow Ia	116,86	175,35	0,06



Dilithium und Falcon eignen sich.

Authentisierung
Handshake Performanz

Algorithmen	Handshake 50th percentile (ms)	Handshake 95th percentile (ms)	Latency 50th percentile (%)	Latency 95th percentile (%)
RSA 3072	131,54	227,26	0	0
Dilithium II	140,20	232,51	6,58	2,31
Falcon 512	141,22	235,46	8,12	3,49
Picnic L1FS	634,90	985,88	382,63	333,79
SPHINCS+ SHA256-128f-simple	533,15	904,98	320,49	298,19

» Dilithium und Falcon könnten in Zertifikaten eingesetzt werden.

Authentisierung Übersicht

Algorithmen

- Falcon
- Dilithium

Kleine Anpassungen

- Neue Identifikatoren für signature_algorithms_cert und signature_algorithms
- Neue Identifikatoren für X.509

Authentisierung
Hybride Signaturen

Mehrere Zertifikate

- TLS erweitern
- Evtl. Certificate-Nachricht
- CertificateVerify: Konkatenieren der Signaturen

Mehrere Schlüssel in einem Zertifikat

- Algorithmen individuell oder gemeinsam
- Wie behandelt CA hybride Signaturen?
- CertificateVerify: Konkatenieren der Signaturen

Agenda

01 Grundlagen

02 Schlüsselaustausch

03 Authentisierung

04 **Auswirkungen kurz zusammengefasst**

Auswirkungen kurz zusammengefasst Übersicht

Zertifikate

- Hybride Zertifikate / PQ-only Zertifikate
- Größe nimmt zu, evtl. neue Erweiterungen

TLS Handshake

- Signaturen konkatenieren oder Datenstrukturen anpassen
- Handshake-Dauer nimmt zu

PKI

- Hybride PKIs / mehrere Zertifikate
- Proof-of-Possession?

TLS Standard

- Algorithmenpaare statt elliptische Kurven
- Resumption, … ?

NIST Algorithmen

- Dilithium, Falcon; Kyber, SABER, Frodo

Vielen Dank! Fragen?

Alexander Kulpe

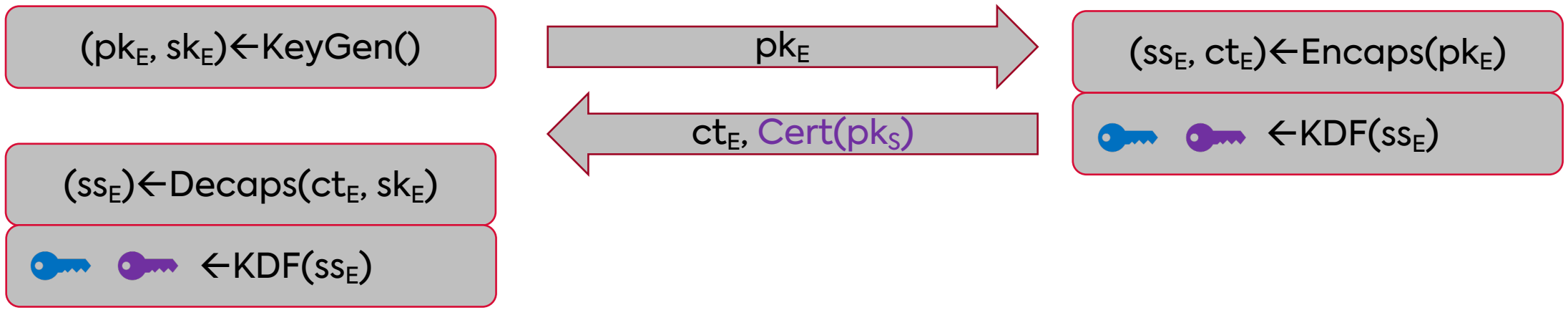
Praktikant

secunet Security Networks AG

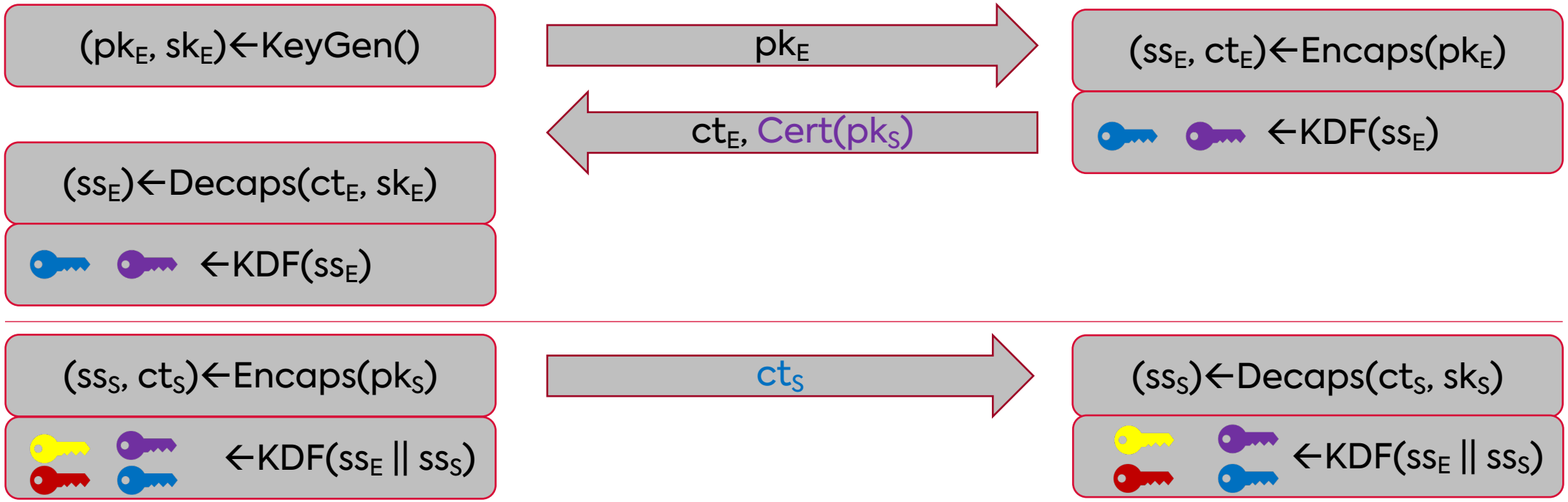
alexander.kulpe@secunet.com

secunet

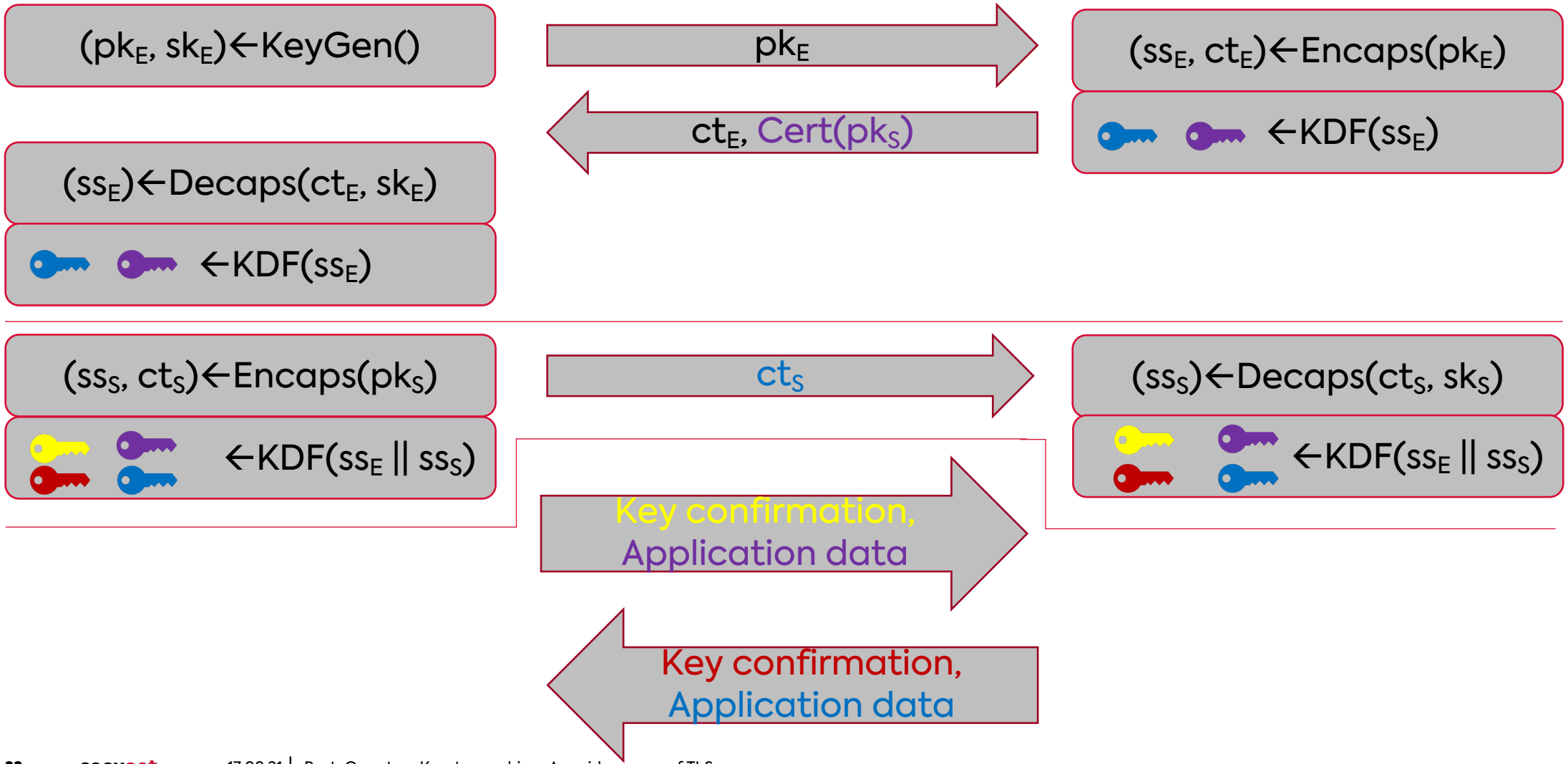
Alternative: KEMTLS
Vereinfachter Protokollablauf



Alternative: KEMTLS
Vereinfachter Protokollablauf



Alternative: KEMTLS
Vereinfachter Protokollablauf



KEMTLS vs.. TLS
Performanz ohne ICA

	Key Exchange (Bytes)	Authentication (Bytes)	Cert pk (Bytes)	Cert sig (Bytes)	sum	ICA cert sig	ICA cert sig	Sum	Root CA pk
TLS 1.3	SIKE 405	Falcon 609	Falcon 897	GeMSS 32	2 024	GeMSS 352 180	GeMSS 32	354 236	GeMSS 352 180
KEMTLS	SIKE 405	SIKE 209	SIKE 196	GeMSS 32	842	GeMSS 352180	GeMSS 32	353 054	GeMSS 352 180

	Client sent req. (excl/incl ICA cert)	Client recv. Resp. (excl/incl ICA cert)	Server HS done
TLS 1.3	154,9 259,0	186,0 290,2	123,1 227,1
KEMTLS	190,4 293,3	256,6 359,5	193,4 296,3

31,1 ms latency, 1000 Mbps bandwith

	Client sent req. (excl/incl ICA cert)	Client recv. Resp. (excl/incl ICA cert)	Server HS done
TLS 1.3	473,7 10 936,3	669,8 11 902,5	277,5 10 348,1
KEMTLS	496,8 10 859,5	723,0 11 861,0	330,8 10 331,7

194,6 ms latency, 10 Mbps bandwith

» TLS 1.3 hier etwas besser als KEMTLS.

KEMTLS vs.. TLS
Performanz mit ICA

	Key Exchange (Bytes)	Authentication (Bytes)	Cert pk (Bytes)	Cert sig (Bytes)	sum	ICA cert sig	ICA cert sig	Sum	Root CA pk
TLS 1.3	SIKE 405	Falcon 690	Falcon 897	XMSS^{MT} 979	2 971	XMSS^{MT} 32	GeMSS 32	3 035	GeMSS 352 180
KEMTLS	SIKE 405	SIKE 209	SIKE 196	XMSS^{MT} 979	1 789	XMSS^{MT} ^{32}	GeMSS 32	1 853	GeMSS 352 180

	Client sent req. (excl/incl ICA cert)	Client recv. Resp. (excl/incl ICA cert)	Server HS done
	165,8	196,9	134,0
TLS 1.3	166,2	197,3	134,4
	202,1	268,8	205,6
KEMTLS	202,3	269,1	205,9

31,1 ms latency, 1000 Mbps bandwith

	Client sent req. (excl/incl ICA cert)	Client recv. Resp. (excl/incl ICA cert)	Server HS done
	482,1	678,4	285,8
TLS 1.3	482,5	678,8	286,2
	505,8	732,0	339,7
KEMTLS	506,1	732,4	340,1

194,6 ms latency, 10 Mbps bandwith

» TLS 1.3 hier besser als KEMTLS.

KEMTLS vs. TLS
Performanz mit MLWE/MSIS

	Key Exchange (Bytes)	Authentication (Bytes)	Cert pk (Bytes)	Cert sig (Bytes)	sum	ICA cert sig	ICA cert sig	Sum	Root CA pk
TLS 1.3	Kyber 1 536	Dilithium 2 044	Dilithium 1 184	Dilithium 2 044	6 808	Dilithium 1 184	Dilithium 2 044	10 036	Dilithium 1 184
KEMTLS	Kyber 1 536	Kyber 736	Kyber 800	Dilithium 2 044	5 116	Dilithium 1 184	Dilithium 2 044	8 344	Dilithium 1 184

	Client sent req. (excl/incl ICA cert)	Client recv. Resp. (excl/incl ICA cert)	Server HS done
TLS 1.3	64,3 64,8	95,5 96,0	33,3 33,8
KEMTLS	63,4 63,9	95,0 95,5	32,7 33,2

31,1 ms latency, 1000 Mbps bandwith

	Client sent req. (excl/incl ICA cert)	Client recv. Resp. (excl/incl ICA cert)	Server HS done
TLS 1.3	411,6 415,9	852,4 854,7	446,1 448,0
KEMTLS	399,2 418,9	835,1 864,2	439,9 447,6

194,6 ms latency, 10 Mbps bandwith

» KEMTLS hier etwas besser als TLS 1.3.

KEMTLS vs. TLS
Performanz mit NTRU

	Key Exchange (Bytes)	Authentication (Bytes)	Cert pk (Bytes)	Cert sig (Bytes)	sum	ICA cert sig	ICA cert sig	Sum	Root CA pk
TLS 1.3	NTRU 1 398	Falcon 690	Falcon 897	Falcon 690	3 675	Falcon 897	Falcon 690	5 262	Falcon 897
KEMTLS	NTRU 1 398	NTRU 699	NTRU 699	Falcon 690	3 486	Falcon 897	Falcon 690	5 073	Falcon 897

	Client sent req. (excl/incl ICA cert)	Client recv. Resp. (excl/incl ICA cert)	Server HS done
TLS 1.3	65,1 65,6	96,3 96,9	34,1 34,7
KEMTLS	63,6 64,2	95,2 95,8	32,9 33,5

31,1 ms latency, 1000 Mbps bandwith

	Client sent req. (excl/incl ICA cert)	Client recv. Resp. (excl/incl ICA cert)	Server HS done
TLS 1.3	398,1 406,7	662,2 842,8	269,2 443,5
KEMTLS	396,2 400,0	593,4 835,4	200,6 440,2

194,6 ms latency, 10 Mbps bandwith

» KEMTLS hier besser als TLS 1.3.