KINDI

Key EncapsulatIoN and Encryption baseD on LattIces

Rachid El Bansarkhani

TECHNISCHE UNIVERSITÄT DARMSTADT

QuantiCor

KINDI - Introduction

- **IDEA:** Use trapdoor-based construction and inject data into the error term [EDB15,EI17]
- Trapdoor construction: Allows to retrieve secret ***s*** and error ***e*** from (Module- or Ring-)LWE instances:

$$\mathbf{c = A \cdot s + e}$$

- Error Term ***e*** contains all encrypted data ***m***
- Secret ***s*** contains a key
- Security based on pseudorandomness of **c**

„simplified“ KEM and Encryption scheme simultaneously

KINDI

CPA-secure Encryption Scheme

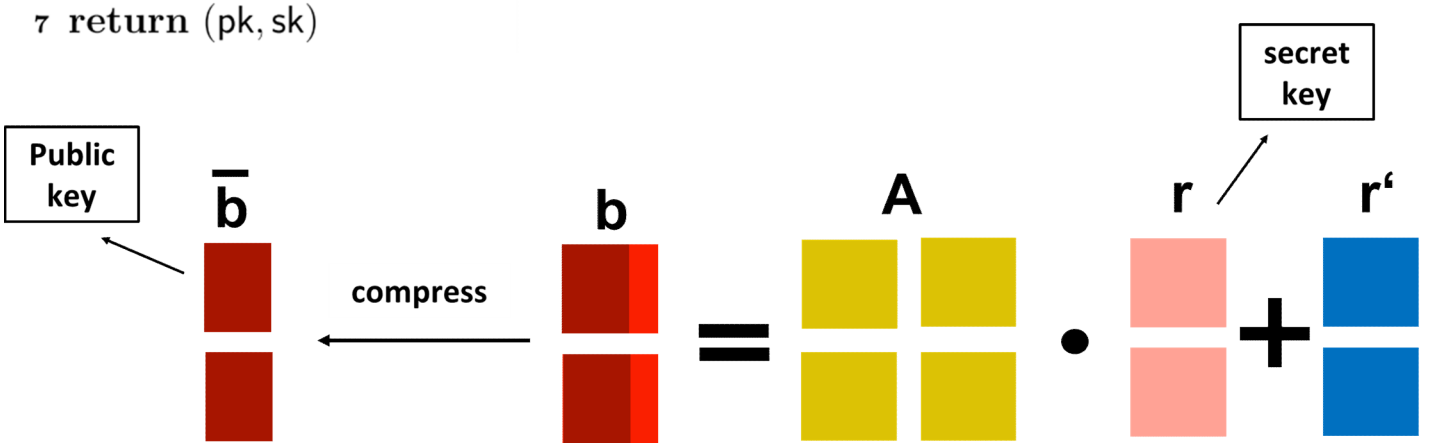
KINDI - Encryption

$\mathsf{KINDI}_{\mathsf{CPA}}.\mathsf{KeyGen}(1^n, p, k, t, \ell)$:

1 $\gamma, \mu \leftarrow \{0,1\}^n$
2 $\mathbf{A} \in \mathcal{R}_q^{\ell \times \ell} \leftarrow \mathsf{Shake}(\mu)$
3 $\mathbf{r}, \mathbf{r}' \in \mathcal{R}_q^{\ell} \leftarrow \mathsf{Shake}_p(\gamma)$
4 $\mathbf{b} = \mathbf{A} \cdot \mathbf{r} + \mathbf{r}'$
5 $\bar{\mathbf{b}} = \mathsf{Compress}(\mathbf{b}, t)$
6 $\mathsf{pk} := (\bar{\mathbf{b}}, \mu),\ \mathsf{sk} := (\mathbf{r}, \bar{\mathbf{b}}, \mu)$
7 **return** $(\mathsf{pk}, \mathsf{sk})$

Rq=$\mathbb{Z}_q[x]/\langle x^n+1\rangle$

Drop t least significant bits



KINDI - Encryption

$\mathsf{KINDI}_{\mathsf{CPA}}.\mathsf{Encrypt}(\mathsf{pk}, \mathsf{msg} = \{0,1\}^{n(\ell+1)\log 2p}; \mathsf{coins} = \perp \text{ or } s_1 \in \mathcal{R}_2)$:

1 $s_1 \leftarrow \mathcal{R}_2$

2 $\mathbf{A} \leftarrow \mathsf{Shake}(\mu)$

3 $\bar{\mathbf{p}} = \mathsf{Decompress}(\bar{\mathbf{b}}, t)$

4 $\mathbf{p} = (\bar{\mathbf{p}}_1 + \mathbf{g}, \bar{\mathbf{p}}_2, \ldots, \bar{\mathbf{p}}_\ell)$

5 $\bar{u}, \bar{\mathbf{s}}_1, (\mathbf{s}_2, \ldots, \mathbf{s}_\ell) \in \{0,1\}^{n(\ell+1)\log 2p} \times \mathcal{R}_p \times \mathcal{R}_{2p}^{\ell-1} \leftarrow \mathsf{G}(s_1) := \mathsf{Shake}(s_1)$

6 $\mathbf{s} = (s_1 + 2 \cdot \bar{\mathbf{s}}_1 - [p], \mathbf{s}_2 - [p], \ldots, \mathbf{s}_\ell - [p])^\top$

7 $u = \bar{u} \oplus \mathsf{msg}$

8 $\mathbf{u} = \mathsf{Encode}(u)$

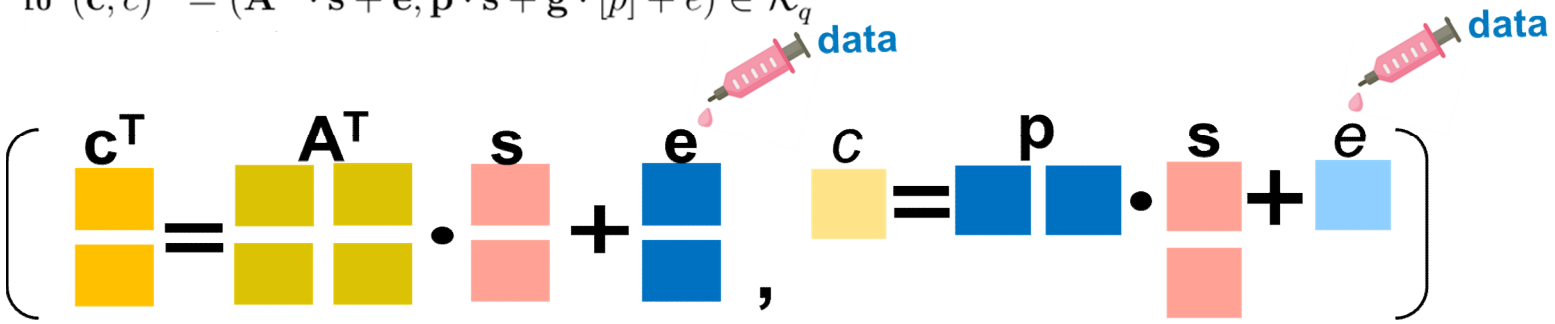
9 $\mathbf{e} = (\mathbf{u}_1 - [p], \ldots, \mathbf{u}_\ell - [p])^\top, \; e = \mathbf{u}_{\ell+1} - [p]$

10 $(\mathbf{c}, c)^\top = (\mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e}, \mathbf{p} \cdot \mathbf{s} + \mathbf{g} \cdot [p] + e) \in \mathcal{R}_q^{\ell+1}$

Shift t bits: Multiply with 2^t

$g = 2^{k-1}$

Centralization to {-p,….,p-1}



KINDI - Encryption

Algorithm 6: $\mathsf{KINDI}_{\mathsf{CPA}}.\mathsf{Decrypt}(\mathsf{sk}, (\mathbf{c}, c))$:

1 $\mathbf{A} \leftarrow \mathsf{Shake}(\mu)$
2 $\bar{\mathbf{p}} = \mathsf{Decompress}(\bar{\mathbf{b}}, t)$
3 $\mathbf{p} = (\bar{\mathbf{p}}_1 + \mathbf{g}, \bar{\mathbf{p}}_2, \ldots, \bar{\mathbf{p}}_\ell)$
4 $\mathbf{v} = c - \mathbf{c} \cdot \mathbf{r}^\top$
5 $s_1 = \mathsf{Recover}(\mathbf{v}) \in \mathcal{R}_2$
6 $\bar{u}, \bar{\mathbf{s}}_1, (\mathbf{s}_2, \ldots, \mathbf{s}_\ell) \in \{0,1\}^{n(\ell+1)\log 2p} \times \mathcal{R}_p \times \mathcal{R}_p^{\ell-1} \leftarrow \mathsf{Shake}(s_1)$
7 $\mathbf{s} = (s_1 + 2 \cdot \bar{\mathbf{s}}_1 - [p], \mathbf{s}_2 - [p], \ldots, \mathbf{s}_\ell - [p])^\top$
8 $(\mathbf{e}, e) = (\mathbf{u}_1 - [p], \ldots, \mathbf{u}_{\ell+1} - [p]) = (\mathbf{c} - \mathbf{A}^\top \cdot \mathbf{s}, c - \mathbf{p} \cdot \mathbf{s}) \bmod q$
9 $\mathsf{msg} = \mathsf{Decode}(\mathbf{u}) \oplus \bar{u}$

Divide by 2^{k-1} and round

- Decryption retrieves secret and the error term back.
- Use message msg or s_1 as a key or both simultaneously.

KINDI

CCA-secure KEM

KINDI - CCA2-secure KEM

$\mathsf{KINDI}_{\mathsf{CCA-KEM}}.\mathsf{Encaps}(\mathsf{pk})$:

1 $s_1 \leftarrow \{0,1\}^n$
2 $d \leftarrow \mathsf{H}(s_1)$
3 $(\mathbf{c}, c)^\top \leftarrow \mathsf{KINDI}_{\mathsf{CPA}}.\mathsf{Encrypt}(\mathsf{pk}, d; s_1)$
4 $K \leftarrow \mathsf{H}'(s_1, (\mathbf{c}, c))$

Few steps more than encryption

- In the random oracle model: **d=0**.
  Essentially: (**c**,*c*)=Encrypt(pk, **0**), **K**=H‘(s_1, (**c**, *c*))
- In the quantum random oracle model: **d=H(s_1)**
  Essentially: (**c**,*c*)=Encrypt(pk, d=H(s_1)), **K**=H‘(s_1, (**c**, *c*))
- Note: **d** can be encrypted, too

KINDI - CCA2-secure KEM

$\mathsf{KINDI}_{\mathsf{CCA-KEM}}.\mathsf{Decaps}(\mathsf{sk}, (\mathbf{c}, c))$:

$(d', s_1') \leftarrow \mathsf{KINDI}_{\mathsf{CPA}}.\mathsf{Decrypt}(\mathsf{sk}, (\mathbf{c}, c))$
if $d' = d := \mathsf{H}(s_1')$
    return $\mathsf{H}'(s_1', (\mathbf{c}, c))$
else
    return $\mathsf{H}'(s, (\mathbf{c}, c))$

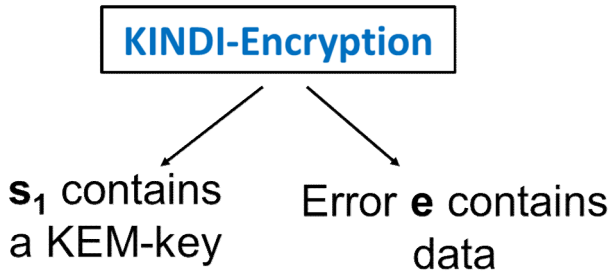
- Due to trapdoor design ciphertext computation/check not required
- RO: Few steps more than Decryption:
  Check d=0 & compute **K**=H‘(s_1, (**c**, *c*))
- qRO: Few steps more than Decryption:
  Check d=H(s_1) & compute **K**=H‘(s_1, (**c**, *c*))

KINDI - Design Features

- Trapdoor construction - recovery of secret and error vectors for inspection:
  - Message recovery
  - Have errors and secrets correct format?
  - Has ciphertext been altered?
- Security: Constructions based on module-LWE [LS15] in the (quantum-)RO
  - For module rank equal to 1: ring-LWE
  - Fine grained usage of dimensions for better security-efficiency tradeoffs
  - Higher protection against dense sub-lattice attacks [KF17] etc.

KINDI - Design Features

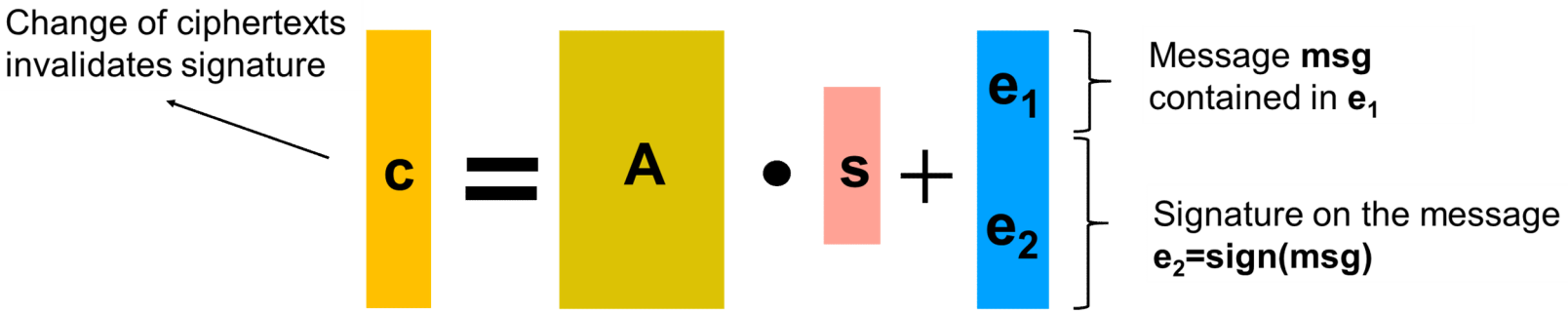
- Simplicity of design for encryption and KEM
- KINDI-Encryption scheme always by design encrypts a message and key simultaneously



- Thus, few steps needed to obtain CCA2-secure KEMs [HHK17]

KINDI - Design Features

- Huge amount of data can be encrypted at small ciphertext sizes
  - $(l+1)\cdot n\cdot \log 2p$ bits of data or $\log 2p$ bits per coefficient
  - Low message expasion $\log 2p/\log q$, e.g. 4 or lower possible
  - Increasing parameters l, n, p → more data and security
  - Allows to encrypt bundles of session keys, signatures, etc.
  - Suitable for sign-then-encrypt scenarios



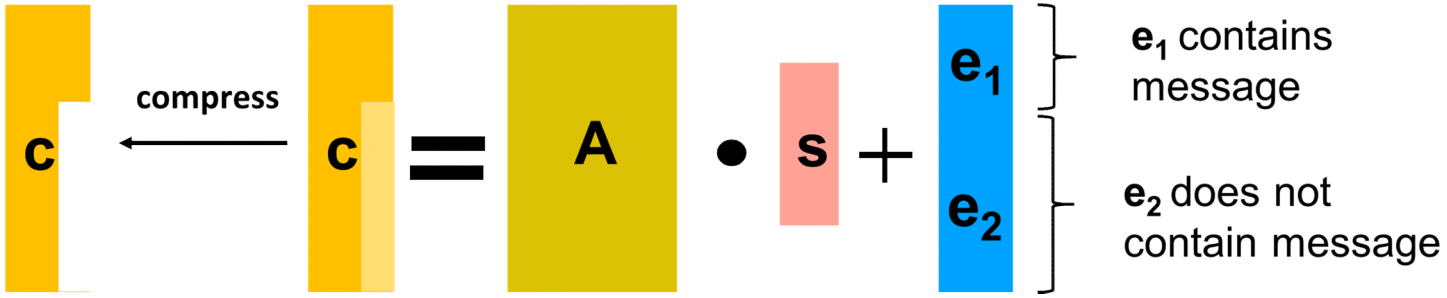
KINDI - Design Features

- KINDI encryption scheme can easily be turned to be CCA2-secure beside the proposed KEM variant:
  - Slightly larger parameters
  - Essentially check size of coefficients
  - Add $F(s_1,e)$ to the ciphertext with F=RO
- Authenticated key exchange is obtained via generic transformations

- Trapdoor constructions are used in many advanced primitives ranging from group signatures to attribute-based encryption

KINDI - Design Features

- Possible Modifications: Ciphertext can also be compressed in case the complete bandwidth is not used



- Secret and public keys can be generated from small seeds, if sizes are critical

KINDI - Technical Features

- Highly efficient implementation even at a high security level, also suitable for IoT
- Use of power-of-two modulo and domain size, e.g. for error distrib.:
  - No rejection sampling and no waste of random bits
  - No expensive modulo operations, one use of „AND“
- Polynomial multiplication: FFT multiplication in cyclotomic ring x^n+1 for small $n=2^k$ such as n=256, 512. Reusage of the FFT subroutine for any change of parameters l, p, q. Extendable to NTT case.
- Usage of FIPS 202 standardized Shake for random bit generation and random oracle instantiations

KINDI - Technical Features

- Constant time implementation:
  Computations independent from secret elements

- Additional implementation: Improved running times via parallelization, e.g.

  - AVX parallelization of the FFT
  - Vectorized Keccak: faster generation of random bits and key derivation

KINDI - Results

Scheme	Sizes (bytes)	Timings (Spec. AVX Impl.)	Timings (Spec. Ref. Impl.)	Bit Security (conservative)	NIST Category
Encryption KINDI-256-3	PK 1184 SK 1472 CT 1792	Keygen 104308 Encrypt 122648 Decrypt 151723	Keygen 203096 Encrypt 247793 Decrypt 312211	>160	3
KEM KINDI-256-3	PK 1184 SK 1472 CT 1792	Encaps 133888 Decaps 162070	Encaps 260137 Decaps 323947	>160	3
Encryption KINDI-512-2	PK 1456 SK 1712 CT 2496	Keygen 113082 Encrypt 142950 Decrypt 187097	Keygen 214064 Encrypt 280420 Decrypt 377962	>220	5
KEM KINDI-512-2	PK 1456 SK 1712 CT 2496	Encaps 160150 Decaps 202458	Encaps 306043 Decaps 397147	>220	5

KINDI - Design Features

- Current implementation and documentation at: http://kindi-kem.de/
- [HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. IACR Cryptology ePrint Archive, 2017:604. 2017.
- [LS15] Adeline Langlois and Damien Stehlé. Worst-case to Average-case Reductions for Module Lattices. Designs, Codes and Cryptography. 2015
- [EDB15] Rachid El Bansarkhani, Ozgür Dagdelen, and Johannes A. Buchmann. Augmented Learning with Errors: The Untapped Potential of the Error Term. Financial Cryptography and Data Security. 2015.
- [EI17] El Bansarkhani Rachid. Lara - A Design Concept for Lattice-based Encryption. Cryptology ePrint Archive, Report 2017/049. 2017.

KINDI - Design Features

- [KF17] Paul Kirchner and Pierre-Alain Fouque. Revisiting Lattice Attacks on overstretched NTRU Parameters. EUROCRYPT. 2017