Secure and Anonymous Hybrid Encryption from Coding Theory

Edoardo Persichetti

University of Warsaw

06 June 2013

Part I

PRELIMINARIES

Error-Correcting Codes

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.

w-error correcting: exists decoding algorithm that corrects up to w errors occurred on a codeword.

Error-Correcting Codes

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.

w-error correcting: exists decoding algorithm that corrects up to w errors occurred on a codeword.

Hamming weight

Number of non-zero entries: $wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$.

Error-Correcting Codes

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.

w-error correcting: exists decoding algorithm that corrects up to w errors occurred on a codeword.

Hamming weight

Number of non-zero entries: $wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$.

Parity-check matrix

$H \in \mathbb{F}_q^{(n-k)\times n}$ defines the code as follows: $x \in \mathcal{C} \iff Hx^T = 0$.
Systematic form: $(M|I_{n-k})$.

Code-based Public-key Encryption Schemes

McEliece: first cryptosystem using error correcting codes (1978).

Code-based Public-key Encryption Schemes

McEliece: first cryptosystem using error correcting codes (1978).

Based on the hardness of decoding random linear codes.

Code-based Public-key Encryption Schemes

McEliece: first cryptosystem using error correcting codes (1978).

Based on the hardness of decoding random linear codes.

“Dual” version proposed by Niederreiter (1985).

Code-based Public-key Encryption Schemes

McEliece: first cryptosystem using error correcting codes (1978).

Based on the hardness of decoding random linear codes.

"Dual" version proposed by Niederreiter (1985).

Problem (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ *and* $w \in \mathbb{N}$.
Goal: *find a word* $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq w$ *such that* $He^T = y$.

Unique solution and hardness only if w is below a certain threshold (GV bound).

Code-based Public-key Encryption Schemes

McEliece: first cryptosystem using error correcting codes (1978).

Based on the hardness of decoding random linear codes.

"Dual" version proposed by Niederreiter (1985).

Problem (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ *and* $w \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq w$ *such that* $He^T = y$.

Unique solution and hardness only if w is below a certain threshold (GV bound).

If H defines an error-correcting code, we have a trapdoor: special description Δ allows decoding algorithm to correct errors.

Niederreiter, revisited

Key Generation

- Choose w-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Niederreiter, revisited

Key Generation

- Choose w-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Encryption

- Message is a word $e \in \mathbb{F}_2^n$ of weight w.
- $c = He^T$.

Niederreiter, revisited

Key Generation

- Choose w-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Encryption

- Message is a word $e \in \mathbb{F}_2^n$ of weight w.
- $c = He^T$.

Decryption

- Set $e = Decode_\Delta(c)$ and return e.
- Return $\perp$ if decoding fails.

Part II

Hybrid Encryption

Motivation

Purpose of public-key encryption: encrypt key for symmetric scheme.

Motivation

Purpose of public-key encryption: encrypt key for symmetric scheme.

Niederreiter cryptosystem requires use of constant-weight encoding functions to transform symmetric key into fixed-weight string e.

Motivation

Purpose of public-key encryption: encrypt key for symmetric scheme.

Niederreiter cryptosystem requires use of constant-weight encoding functions to transform symmetric key into fixed-weight string e.

Can do this in a more efficient way: build a KEM based on Niederreiter's assumptions.

The KEM-DEM framework

Introduced by Cramer and Shoup (2001), combines the actions of two independent mechanisms.

The KEM-DEM framework

Introduced by Cramer and Shoup (2001), combines the actions of two independent mechanisms.

Key Encapsulation Mechanism (KEM)

- Keygen: generates private key SK and public key PK.
- $\mathrm{Enc}^{KEM}(\mathrm{PK})$: produces a symmetric key K and a ciphertext c_0.
- $\mathrm{Dec}^{KEM}(\mathrm{SK}, c_0)$: returns the symmetric key K (or $\perp$).

The KEM-DEM framework

Introduced by Cramer and Shoup (2001), combines the actions of two independent mechanisms.

Key Encapsulation Mechanism (KEM)

- Keygen: generates private key SK and public key PK.
- $\mathrm{Enc}^{KEM}(\mathrm{PK})$: produces a symmetric key K and a ciphertext c_0.
- $\mathrm{Dec}^{KEM}(\mathrm{SK}, c_0)$: returns the symmetric key K (or $\perp$).

Data Encapsulation Mechanism (DEM)

- $\mathrm{Enc}^{DEM}(K, m)$: produces the ciphertext c_1.
- $\mathrm{Dec}^{DEM}(K, c_1)$: returns the plaintext m (or $\perp$).

HYBRID ENCRYPTION SCHEME

- Keygen: generates private key SK and public key PK.
- Enc^{HY}(PK,m):
  - Run Enc^{KEM}(PK) and get (K, c_0).
  - Run $\mathrm{Enc}^{DEM}(K, m)$ and get c_1.
  - Final ciphertext $c = (c_0, c_1)$.
- Dec^{HY}(SK, c):
  - Run Dec^{KEM}(SK,c_0) and get K.
  - Run $\mathrm{Dec}^{DEM}(K, c_1)$ and recover m.

Security

Independent components with separate security definitions, however

Security

Independent components with separate security definitions, however

IND-CCA secure KEM + IND-CCA secure DEM $\Longrightarrow$
IND-CCA secure hybrid scheme!

Security

Independent components with separate security definitions, however

$$\text{IND-CCA secure KEM} + \text{IND-CCA secure DEM} \implies \text{IND-CCA secure hybrid scheme!}$$

DEM: usual symmetric encryption IND-CCA requirement.
Can use any symmetric scheme (e.g. one-time pad) + MAC.

Security

Independent components with separate security definitions, however

$$\text{IND-CCA secure KEM + IND-CCA secure DEM} \implies \text{IND-CCA secure hybrid scheme!}$$

DEM: usual symmetric encryption IND-CCA requirement.
Can use any symmetric scheme (e.g. one-time pad) + MAC.

IND-CCA security for KEM

- Get public key PK.
- Perform decryption queries.
- Challenge ciphertext: (K^*, c^*) either honestly obtained ($b = 1$) by $\text{Enc}^{KEM}(\text{PK})$ or by choosing K^* as a random string ($b = 0$).
- Perform decryption queries ($\neq c^*$).
- Return b^*.

$$\text{Adv}_{KEM}(\mathcal{A}, \lambda) = \left|\Pr[b^* = b] - 1/2\right|$$

Niederreiter KEM

Secure in Random Oracle model, makes use of Key Derivation Function (KDF), e.g. SHA-3.

Niederreiter KEM

Secure in Random Oracle model, makes use of Key Derivation Function (KDF), e.g. SHA-3.

Key Generation

- Choose w-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Niederreiter KEM

Secure in Random Oracle model, makes use of Key Derivation Function (KDF), e.g. SHA-3.

Key Generation

- Choose w-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Encryption

- Choose a random word $e \in \mathbb{F}_2^n$ of weight w.
- $K = KDF(e), c_0 = He^T$.

Niederreiter KEM

Secure in Random Oracle model, makes use of Key Derivation Function (KDF), e.g. SHA-3.

Key Generation

- Choose w-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Encryption

- Choose a random word $e \in \mathbb{F}_2^n$ of weight w.
- $K = KDF(e), c_0 = He^T$.

Decryption

- Set $e = Decode_\Delta(c_0)$ and return $K = KDF(e)$.
- Return $KDF(c_0)$ if decoding fails.

Proof of security (sketch)

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for SDP such that

$$Adv_{KEM}(\mathcal{A}, \lambda) \leq Adv_{SDP}(\mathcal{A}', \lambda) + n_{DEC}/N.$$

Proof of security (sketch)

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for SDP such that

$$Adv_{KEM}(\mathcal{A}, \lambda) \leq Adv_{SDP}(\mathcal{A}', \lambda) + n_{DEC}/N.$$

Model KDF as a random oracle $\mathcal{H}$.

Proof of security (sketch)

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for SDP such that

$$Adv_{KEM}(\mathcal{A}, \lambda) \leq Adv_{SDP}(\mathcal{A}', \lambda) + n_{DEC}/N.$$

Model KDF as a random oracle $\mathcal{H}$.

- Game 0: the KEM security game.

Proof of security (sketch)

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for SDP such that

$$Adv_{KEM}(\mathcal{A}, \lambda) \leq Adv_{SDP}(\mathcal{A}', \lambda) + n_{DEC}/N.$$

Model KDF as a random oracle $\mathcal{H}$.

- Game 0: the KEM security game.
- Game 1: halt if challenge ciphertext $c_0^* = He^{*T}$ had been previously queried.

Proof of security (sketch)

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for SDP such that

$$Adv_{KEM}(\mathcal{A}, \lambda) \leq Adv_{SDP}(\mathcal{A}', \lambda) + n_{DEC}/N.$$

Model KDF as a random oracle $\mathcal{H}$.

- Game 0: the KEM security game.
- Game 1: halt if challenge ciphertext $c_0^* = He^{*T}$ had been previously queried.
- Game 2: generate c_0^* at beginning and halt if $\mathcal{H}$ queried at e^*. Use adversary $\mathcal{A}'$ as a simulator.

Proof of security (sketch)

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for SDP such that

$$Adv_{KEM}(\mathcal{A}, \lambda) \leq Adv_{SDP}(\mathcal{A}', \lambda) + n_{DEC}/N.$$

Model KDF as a random oracle $\mathcal{H}$.

- Game 0: the KEM security game.
- Game 1: halt if challenge ciphertext $c_0^* = He^{*T}$ had been previously queried.
- Game 2: generate c_0^* at beginning and halt if $\mathcal{H}$ queried at e^*. Use adversary $\mathcal{A}'$ as a simulator.

Simulation possible thanks to modification in the decryption algorithm.

THE SIMULATOR

$\mathcal{A}'$ has to solve an instance (H, y, w) of SDP. Interaction with $\mathcal{A}$:

The simulator

$\mathcal{A}'$ has to solve an instance (H, y, w) of SDP. Interaction with $\mathcal{A}$:

Key Generation

Set PK$=H$ and give PK to $\mathcal{A}$.

The simulator

$\mathcal{A}'$ has to solve an instance (H, y, w) of SDP. Interaction with $\mathcal{A}$:

Key Generation

Set PK$= H$ and give PK to $\mathcal{A}$.

Challenge Queries

Set $c^* = y$ and K^* random string and give (K^*, c^*) to $\mathcal{A}$.

The simulator

$\mathcal{A}'$ has to solve an instance (H, y, w) of SDP. Interaction with $\mathcal{A}$:

Key Generation

Set PK$= H$ and give PK to $\mathcal{A}$.

Challenge Queries

Set $c^* = y$ and K^* random string and give (K^*, c^*) to $\mathcal{A}$.

Random Oracle Queries

Receive query e and compute $s = He^T$. If $s = y$ then win the game and halt. Otherwise, generate K at random.

The simulator

$\mathcal{A}'$ has to solve an instance (H, y, w) of SDP. Interaction with $\mathcal{A}$:

Key Generation

Set PK$= H$ and give PK to $\mathcal{A}$.

Challenge Queries

Set $c^* = y$ and K^* random string and give (K^*, c^*) to $\mathcal{A}$.

Random Oracle Queries

Receive query e and compute $s = He^T$. If $s = y$ then win the game and halt. Otherwise, generate K at random.

Decryption Queries

Receive query c_0 and reply with a random string K.

The simulator

$\mathcal{A}'$ has to solve an instance (H, y, w) of SDP. Interaction with $\mathcal{A}$:

Key Generation

Set PK$= H$ and give PK to $\mathcal{A}$.

Challenge Queries

Set $c^* = y$ and K^* random string and give (K^*, c^*) to $\mathcal{A}$.

Random Oracle Queries

Receive query e and compute $s = He^T$. If $s = y$ then win the game and halt. Otherwise, generate K at random.

Decryption Queries

Receive query c_0 and reply with a random string K.

Use of tables to guarantee integrity.

Part III

Anonymity

Introduction

Increasingly important notion in the community.

Introduction

Increasingly important notion in the community.

Key Privacy vs Data Privacy

Increasingly important notion in the community.

Key Privacy vs Data Privacy

IK-CCA SECURITY FOR PKE

- Get two public keys PK_0 and PK_1.
- Perform decryption queries (for either).
- Choose message m. Challenge ciphertext: $c^* = \mathsf{Enc}(\mathsf{PK}_b, m)$ for $b \in \{0, 1\}$.
- Perform decryption queries ($\neq c^*$).
- Return b.

ANONYMITY FOR CODE-BASED SCHEMES

"Plain" Niederreiter (or McEliece) scheme: not secure.

Anonymity for code-based schemes

"Plain" Niederreiter (or McEliece) scheme: not secure.

IND-CPA "randomized" variant by Nojima et al.: IK-CPA secure (Yamakawa et al., 2007).

Anonymity for code-based schemes

"Plain" Niederreiter (or McEliece) scheme: not secure.

IND-CPA "randomized" variant by Nojima et al.: IK-CPA secure (Yamakawa et al., 2007).

What about hybrid encryption?

Anonymity for code-based schemes

"Plain" Niederreiter (or McEliece) scheme: not secure.

IND-CPA "randomized" variant by Nojima et al.: IK-CPA secure (Yamakawa et al., 2007).

What about hybrid encryption? Unfortunately

$$\text{IK-CCA secure KEM + IK-CCA secure DEM} \not\Longrightarrow \text{IK-CCA secure hybrid scheme}$$

(Mohassel, 2010)

Anonymity for code-based schemes

"Plain" Niederreiter (or McEliece) scheme: not secure.

IND-CPA "randomized" variant by Nojima et al.: IK-CPA secure (Yamakawa et al., 2007).

What about hybrid encryption? Unfortunately

$$\text{IK-CCA secure KEM + IK-CCA secure DEM} \not\Longrightarrow \text{IK-CCA secure hybrid scheme}$$

(Mohassel, 2010)

We prove IK-CCA security for our scheme directly.

Proof of security (sketch)

Alternative definition of Adv

$$\mathsf{Adv}'_{IND-CCA}(\mathcal{A},\lambda) = \left|\Pr[b^* = 1|b = 1] - \Pr[b^* = 1|b = 0]\right|.$$

Equivalent since $\mathsf{Adv}'_{IND\text{-}CCA}(\mathcal{A},\lambda) = 2 \cdot \mathsf{Adv}_{IND\text{-}CCA}(\mathcal{A},\lambda)$.

Proof of security (sketch)

Alternative definition of Adv

$$\mathsf{Adv}'_{IND-CCA}(\mathcal{A},\lambda) = \left|\Pr[b^* = 1|b = 1] - \Pr[b^* = 1|b = 0]\right|.$$

Equivalent since $\mathsf{Adv}'_{IND\text{-}CCA}(\mathcal{A},\lambda) = 2 \cdot \mathsf{Adv}_{IND\text{-}CCA}(\mathcal{A},\lambda)$.

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for IND-CCA such that

$$Adv_{IK\text{-}CCA}(\mathcal{A},\lambda) \leq Adv'_{IND\text{-}CCA}(\mathcal{A}',\lambda) + n_{DEC}/2N.$$

Proof of security (sketch)

Alternative definition of Adv

$$\mathsf{Adv}'_{IND-CCA}(\mathcal{A},\lambda) = \left|\Pr[b^* = 1|b = 1] - \Pr[b^* = 1|b = 0]\right|.$$

Equivalent since $\mathsf{Adv}'_{IND\text{-}CCA}(\mathcal{A},\lambda) = 2 \cdot \mathsf{Adv}_{IND\text{-}CCA}(\mathcal{A},\lambda)$.

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for IND-CCA such that

$$Adv_{IK\text{-}CCA}(\mathcal{A},\lambda) \leq Adv'_{IND\text{-}CCA}(\mathcal{A}',\lambda) + n_{DEC}/2N.$$

Model KDF as a random oracle $\mathcal{H}$.

Proof of security (sketch)

Alternative definition of Adv

$$\mathsf{Adv}'_{IND-CCA}(\mathcal{A},\lambda) = \Big|\Pr[b^* = 1|b = 1] - \Pr[b^* = 1|b = 0]\Big|.$$

Equivalent since $\mathsf{Adv}'_{IND\text{-}CCA}(\mathcal{A},\lambda) = 2 \cdot \mathsf{Adv}_{IND\text{-}CCA}(\mathcal{A},\lambda)$.

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for IND-CCA such that

$$Adv_{IK\text{-}CCA}(\mathcal{A},\lambda) \leq Adv'_{IND\text{-}CCA}(\mathcal{A}',\lambda) + n_{DEC}/2N.$$

Model KDF as a random oracle $\mathcal{H}$.

- Game 0: the KEM security game.

Proof of security (sketch)

Alternative definition of Adv

$$\mathsf{Adv}'_{IND-CCA}(\mathcal{A},\lambda) = \left|\Pr[b^* = 1|b = 1] - \Pr[b^* = 1|b = 0]\right|.$$

Equivalent since $\mathsf{Adv}'_{IND\text{-}CCA}(\mathcal{A},\lambda) = 2 \cdot \mathsf{Adv}_{IND\text{-}CCA}(\mathcal{A},\lambda)$.

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for IND-CCA such that

$$Adv_{IK\text{-}CCA}(\mathcal{A},\lambda) \leq Adv'_{IND\text{-}CCA}(\mathcal{A}',\lambda) + n_{DEC}/2N.$$

Model KDF as a random oracle $\mathcal{H}$.

- Game 0: the KEM security game.
- Game 1: halt if challenge ciphertext $c^* = \mathsf{Enc}(\mathsf{PK}_b, m)$ had been previously queried.

Proof of security (sketch)

Alternative definition of Adv

$$\mathrm{Adv}'_{IND-CCA}(\mathcal{A},\lambda) = \left|\Pr[b^* = 1|b = 1] - \Pr[b^* = 1|b = 0]\right|.$$

Equivalent since $\mathrm{Adv}'_{IND\text{-}CCA}(\mathcal{A},\lambda) = 2 \cdot \mathrm{Adv}_{IND\text{-}CCA}(\mathcal{A},\lambda)$.

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for IND-CCA such that

$$Adv_{IK\text{-}CCA}(\mathcal{A},\lambda) \leq Adv'_{IND\text{-}CCA}(\mathcal{A}',\lambda) + n_{DEC}/2N.$$

Model KDF as a random oracle $\mathcal{H}$.

- Game 0: the KEM security game.
- Game 1: halt if challenge ciphertext $c^* = \mathrm{Enc}(\mathrm{PK}_b, m)$ had been previously queried.
- Game 2: return additional random string m' together with c^*.

Proof of security (sketch)

Alternative definition of Adv

$$\mathrm{Adv}'_{IND-CCA}(\mathcal{A},\lambda) = \left|\Pr[b^* = 1|b = 1] - \Pr[b^* = 1|b = 0]\right|.$$

Equivalent since $\mathrm{Adv}'_{IND\text{-}CCA}(\mathcal{A},\lambda) = 2 \cdot \mathrm{Adv}_{IND\text{-}CCA}(\mathcal{A},\lambda)$.

Theorem

Let $\mathcal{A}$ be an adversary for KEM and $N = |\mathcal{W}_{n,q,w}|$. There exists an adversary $\mathcal{A}'$ for IND-CCA such that

$$Adv_{IK\text{-}CCA}(\mathcal{A},\lambda) \leq Adv'_{IND\text{-}CCA}(\mathcal{A}',\lambda) + n_{DEC}/2N.$$

Model KDF as a random oracle $\mathcal{H}$.

- Game 0: the KEM security game.
- Game 1: halt if challenge ciphertext $c^* = \mathrm{Enc}(\mathrm{PK}_b, m)$ had been previously queried.
- Game 2: return additional random string m' together with c^*.
- Game 3: set challenge ciphertext $c^* = \mathrm{Enc}(\mathrm{PK}_b, m')$.
  Use adversary $\mathcal{A}'$ as a simulator.

Part IV

Conclusions

Conclusions

First KEM based directly on coding theory problem.

Conclusions

First KEM based directly on coding theory problem.

Simple construction and tight security proof.

Conclusions

First KEM based directly on coding theory problem.

Simple construction and tight security proof.

Extending (Yamakawa et al., 2007), obtains IK-CCA security.

Conclusions

First KEM based directly on coding theory problem.

Simple construction and tight security proof.

Extending (Yamakawa et al., 2007), obtains IK-CCA security.

Implementation?

Merci beaucoup

Thank you

Grazie