(Almost) Non-Interactive Key Exchange from Identity-Based Encryption

O. Blazy C. Chevalier

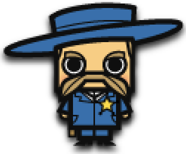
ARES, August 2018

Key Exchange

Alice

$$\xleftrightarrow[Info]{Info}$$

Bob



K_{AB}

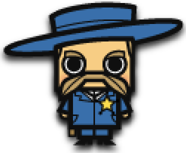
K_{AB}

Non-Interactive Key Exchange

Alice pk_A

K_{AB}

Bob pk_B



K_{AB}

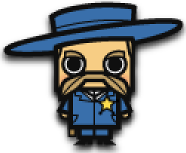
Non-Interactive Key Exchange: Diffie Hellman

Alice: g^a

K_{AB}

g^{ab}

Bob: g^b



K_{AB}

- Various Models Exist

- Various Models Exist
- Consider different kinds of user registration, corruptions, queries

- Various Models Exist
- Consider different kinds of user registration, corruptions, queries
- FHKP13: Inside the same kind of user registration they are (poly) equivalent

- *Register honest user ID* queries: $\mathcal{A}$ supplies id. The challenger runs NIKE.KGen to generate $(\mathsf{pk}, \mathsf{sk})$, records $(\mathit{honest}, \mathsf{id}, \mathsf{pk}, \mathsf{sk})$ and returns pk.

- *Register honest user ID* queries: $\mathcal{A}$ supplies id. The challenger runs NIKE.KGen to generate $(\mathsf{pk}, \mathsf{sk})$, records $(\mathit{honest}, \mathsf{id}, \mathsf{pk}, \mathsf{sk})$ and returns pk.
- *Register corrupt user ID* queries: $\mathcal{A}$ supplies an identity id, pk. The challenger records $(\mathit{corrupt}, \mathsf{id}, \mathsf{pk}, \perp)$.

- *Register honest user ID* queries: $\mathcal{A}$ supplies id.The challenger runs NIKE.KGen to generate $(\mathsf{pk}, \mathsf{sk})$, records $(\mathit{honest}, \mathsf{id}, \mathsf{pk}, \mathsf{sk})$ and returns pk.
- *Register corrupt user ID* queries: $\mathcal{A}$ supplies an identity id, pk. The challenger records $(\mathit{corrupt}, \mathsf{id}, \mathsf{pk}, \perp)$.
- *Corrupt reveal* queries: $\mathcal{A}$ supplies an honest id_A and a corrupted id_B. The challenger runs the NIKE.SharedK algorithm using the honest sk_A and the corrupted pk_B and returns the result.

- *Register honest user ID* queries: $\mathcal{A}$ supplies id.The challenger runs NIKE.KGen to generate $(\mathsf{pk}, \mathsf{sk})$, records $(\mathit{honest}, \mathsf{id}, \mathsf{pk}, \mathsf{sk})$ and returns pk.
- *Register corrupt user ID* queries: $\mathcal{A}$ supplies an identity id, pk. The challenger records $(\mathit{corrupt}, \mathsf{id}, \mathsf{pk}, \perp)$.
- *Corrupt reveal* queries: $\mathcal{A}$ supplies an honest id_A and a corrupted id_B. The challenger runs the NIKE.SharedK algorithm using the honest sk_A and the corrupted pk_B and returns the result.
- *Test* query: $\mathcal{A}$ supplies two honest different id_A, id_B. It either runs NIKE.SharedK using the secret keys or it generates a random key.

HKR vs DKR

User can be registered differently

- Honest Key Registration: When an adversary register a user, his keys have to be well-formed
- Dishonest Key Registration: They do not have to be

HKR vs DKR

User can be registered differently

- Honest Key Registration: When an adversary register a user, his keys have to be well-formed
- Dishonest Key Registration: They do not have to be

Compiler

- We give a compiler from HKR to DKR

Classical Identity-Based Key Encapsulation Mechanism

- $\mathsf{Gen}(\lambda)$ returns $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$ returns $\mathsf{usk}[\mathsf{id}]$ for identity id.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$ returns the key K together with a ciphertext C.
- $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, C)$ returns the decapsulated key K or $\perp$.

Identity-Independant 2-Tier IBKEM

- $\mathsf{Gen}(\lambda)$ returns $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$ returns $\mathsf{usk}[\mathsf{id}]$ for identity id.
- Enc:
  - $\mathsf{Enc}_1(\mathsf{mpk}, \perp)$
    Returns C, r
  - $\mathsf{Enc}_2(\mathsf{mpk}, \mathsf{id}, r)$
    Returns K
- $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, C)$ returns the decapsulated key K or $\perp$.

Security: PR-ID-CPA

Procedure Initialize:

$(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Gen}(\lambda)$

Return mpk

Procedure USKGen(id):

(query forbidden to $\mathcal{A}$ for id^*)

$\mathcal{Q}_{ID} \leftarrow \mathcal{Q}_{ID} \cup \{\mathsf{id}\}$

Return

$\mathsf{usk}[\mathsf{id}] \xleftarrow{\$} \mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$

Procedure Enc(id^*):

(one query only)

$(\mathsf{K}^*, \mathsf{C}^*) \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*)$

$\boxed{\mathsf{K}^* \xleftarrow{\$} \mathcal{K}; \mathsf{C}^* \xleftarrow{\$} \mathsf{C}}$

Return $(\mathsf{K}^*, \mathsf{C}^*)$

Procedure Finalize(β):

Return $(\mathsf{id}^* \notin \mathcal{Q}_{ID}) \wedge \beta$

High level view of Boneh Franklin

Gen(λ): $\mathsf{msk} = x \xleftarrow{\$} \mathbb{Z}_p, \mathsf{mpk} = g^x \in \mathbb{G}$ Returns $(\mathsf{mpk}, \mathsf{msk})$. USKGen($\mathsf{msk}$, id): $\mathsf{usk}[\mathsf{id}] = \mathcal{H}(\mathsf{id})^x \in \mathbb{G}$ Returns $\mathsf{usk}[\mathsf{id}]$	Enc(mpk, id): $r \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{C} = g^r$ $K = e(\mathcal{H}(\mathsf{id})^r, \mathsf{mpk})$ Returns $\mathsf{K} = K$ and C. Dec($\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C}$): $\mathsf{K} = e(\mathsf{C}, \mathsf{usk}[\mathsf{id}])$ Returns K.

Splitting Boneh Franklin

Gen(λ):

$\mathsf{msk} = x \stackrel{\$}{\leftarrow} \mathbb{Z}_p, \mathsf{mpk} = g^x \in \mathbb{G}$
Returns $(\mathsf{mpk}, \mathsf{msk})$.

USKGen(msk, id):

$\mathsf{usk}[\mathsf{id}] = \mathcal{H}(\mathsf{id})^x \in \mathbb{G}$
Returns $\mathsf{usk}[\mathsf{id}]$

Enc(mpk, id):

- $\mathsf{Enc}_1(\mathsf{mpk}, \perp)$
  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, $\mathsf{C} = g^r$
  Returns C, r
- $\mathsf{Enc}_2(\mathsf{mpk}, \mathsf{id}, r)$
  $K = e(\mathcal{H}(\mathsf{id})^r, \mathsf{mpk})$
  Returns K

Returns $\mathsf{K} = K$ and C.

Dec($\mathsf{usk}[\mathsf{id}]$, id, C):

$\mathsf{K} = e(\mathsf{C}, \mathsf{usk}[\mathsf{id}])$
Returns K.

High level (partial) view of Boneh Gentry Hamburg

USKGen(msk, id): For $j \in [\![1, \lambda]\!]$:
$R_j = \mathcal{H}(\mathrm{id}, j) \in J(N), w \xleftarrow{\$} F_K(\mathrm{id}, j)$, and a such that $u^a R_j \in QR(N)$
let $\{z_0, z_1, z_2, z_3\}$ be the four square roots: sets $r_j = z_w$
Returns usk[id] $= \{r_j\}$

Enc(mpk, id):

- $\mathrm{Enc}_1(\mathrm{mpk}, \perp)$
  Picks $r \xleftarrow{\$} \mathbb{Z}_N, S = r^2 \mod N$, Computes $\tau = \mathcal{Q}'(N, u, R, S)$
  sets $k = \left(\frac{\tau(r)}{N}\right)$, $\mathsf{C} = (S, k)$, Returns C, r
- $\mathrm{Enc}_2(\mathrm{mpk}, \mathrm{id}, r)$
  $\forall j \in [\![1, \lambda]\!]$, compute $g_j = \mathcal{Q}'(N, u, \mathcal{H}(\mathrm{id}, j), S)$
  Then $\forall j \in [\![1, \lambda]\!]$, compute $k_j = \left(\frac{g_j(s)}{N}\right)$
  Returns $K = k_1 || \ldots || k_\lambda$

Return $\mathsf{K} = K$ and C.

High level (partial) view of Gentry Peikert Vaikuntanathan

USKGen(msk, id):

For a fresh id, $\forall i \in [\![1, \lambda]\!]$ let $u_i = \mathcal{H}(\mathrm{id}||i)$.
Sets $\mathsf{usk}[\mathrm{id}]_i = f_A^{-1}(A(u_i))$. Returns $\mathsf{usk}[\mathrm{id}]$.

Enc(mpk, id):

- $\mathsf{Enc}_1(\mathsf{mpk}, \bot)$
  For $i \in [\![1, \lambda]\!]$:
  Picks $s_i \xleftarrow{\$} \mathbb{Z}_q^n, v_i \xleftarrow{\$} \mathbb{Z}_q$ uniformly, and sets $p_i = \mathbf{A}^\top s_i + x \in \mathbb{Z}_q^m$ with an x sampled in χ^m. $\mathsf{C} = \{p_i, v_i\}$
  Returns $\mathsf{C}, r = \{s_i\}$
- $\mathsf{Enc}_2(\mathsf{mpk}, \mathrm{id}, r)$
  $\forall i \in [\![1, \lambda]\!]$, sets a bit $k_i = \left((|v_i - \mathcal{H}(\mathrm{id}||i)s_i) \leq \frac{q-1}{4}\right)$.
  Returns $K = k_1|| \dots ||k_\lambda$

Returns $\mathsf{K} = K$ and C.

Summary

	Hypothesis	Post Quantum
Cocks / BGH	DQR	✗

Summary

	Hypothesis	Post Quantum
Cocks / BGH	DQR	✗
Boneh Franklin	DDH	✗

Summary

	Hypothesis	Post Quantum
Cocks / BGH	DQR	✗
Boneh Franklin	DDH	✗
GPV	LWE	✓

Summary

	Hypothesis	Post Quantum
Cocks / BGH	DQR	✗
Boneh Franklin	DDH	✗
GPV	LWE	✓
GHP	Rank-Metric	✓

- IND-ID-CCA + Publicly Verifiable

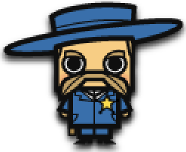
- IND-ID-CCA + Publicly Verifiable
- PR-ID-CCA + SS-NIZK of validity

- IND-ID-CCA + Publicly Verifiable
- PR-ID-CCA + SS-NIZK of validity
- PR-ID-CPA + extractable SS-NIZK of validity

First case: IND-CCA + Publicly Verifiable

Alice C_A

Bob C_B



$$K_A = \mathsf{Dec}(\mathsf{C}_B, \mathsf{usk}[\mathsf{id}_A])$$
$$K'_B = \mathsf{Enc}_2(\mathsf{C}_A, r_A, \mathsf{id}_B)$$
$$K'_B \oplus K_A$$

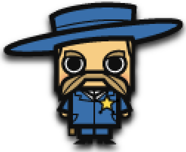
$$= K_{AB} =$$

$$K'_A = \mathsf{Enc}_2(\mathsf{C}_B, r_B, \mathsf{id}_A)$$
$$K_B = \mathsf{Dec}(\mathsf{C}_A, \mathsf{usk}[\mathsf{id}_B])$$
$$K_B \oplus K'_A$$

Second case: PR-IND-CCA + SS-NIZK

Alice C_A, Π_A

Bob C_B, Π_B



$$K_A = \mathrm{Dec}(C_B, \mathsf{usk}[\mathrm{id}_A])$$
$$K'_B = \mathrm{Enc}_2(C_A, r_A, \mathrm{id}_B)$$

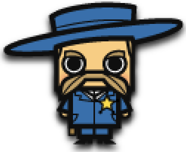
$$K'_A = \mathrm{Enc}_2(C_B, r_B, \mathrm{id}_A)$$
$$K_B = \mathrm{Dec}(C_A, \mathsf{usk}[\mathrm{id}_B])$$

$$K'_B \oplus K_A \quad = \quad K_{AB} \quad = \quad K_B \oplus K'_A$$

Third case: PR-IND-CPA + SS-NIZK

Alice $C_A, \Pi_A(r_A)$

Bob $C_B, \Pi_B(r_B)$



$$K_A = \mathrm{Dec}(C_B, \mathrm{usk}[\mathrm{id}_A])$$
$$K'_B = \mathrm{Enc}_2(C_A, r_A, \mathrm{id}_B)$$
$$K'_B \oplus K_A$$

$$= \quad K_{AB} \quad =$$

$$K'_A = \mathrm{Enc}_2(C_B, r_B, \mathrm{id}_A)$$
$$K_B = \mathrm{Dec}(C_A, \mathrm{usk}[\mathrm{id}_B])$$
$$K_B \oplus K'_A$$

Summary

	Hypothesis	Post Quantum
Cocks / BGH	DQR	✗

Summary

	Hypothesis	Post Quantum
Cocks / BGH	DQR	✗
Boneh Franklin	DDH	✗

Summary

	Hypothesis	Post Quantum
Cocks / BGH	DQR	✗
Boneh Franklin	DDH	✗
GPV	LWE	✓

Summary

	Hypothesis	Post Quantum
Cocks / BGH	DQR	✗
Boneh Franklin	DDH	✗
GPV	LWE	✓
GHP	Rank-Metric	✓

Adapted IBKEM with auxiliary input

Gen(λ):

$i \in [\![1, k]\!]$, picks $\alpha_i, \beta_i \in \mathbb{Z}_p$, $a \in \mathbb{Z}_p$.
Sets $g_{0i} = g^{\beta_i}, g_{1i} = g^{\alpha_i}, g_1 = g^a$,
$\mathsf{msk} = \{a, (\alpha_i, \beta_i)\}, \mathsf{mpk} = g, g_1, (g_{0i}, g_{1i})$.
Returns (mpk, msk).

USKGen(msk, id):

$h_0 = g$, and $\forall i \in [\![1, k]\!]$, $h_i = (h_{i-1}^{\alpha_i^{\mathsf{id}_i} \beta_i^{1-\mathsf{id}_i}})$.
Sets $\mathsf{aux}_{\mathsf{id}} = (h_1, \ldots, h_k)$,$\mathsf{usk}[\mathsf{id}] = h_k^a$.
Returns $\mathsf{usk}[\mathsf{id}], \mathsf{aux}_{\mathsf{id}}$

Enc(mpk, id):

- $\mathsf{Enc}_1(\mathsf{mpk}, \perp)$
  Picks $r \xleftarrow{\$} \mathbb{Z}_q$, and sets $\mathsf{C} = g^r$. C
  Returns C, r
- $\mathsf{Enc}_2(\mathsf{mpk}, \mathsf{id}, r, \mathsf{aux}_{\mathsf{id}})$
  $K = e(g_1, h_k)^r$

Returns $\mathsf{K} = K$ and C.

Dec($\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C}, \mathsf{aux}_{\mathsf{id}}$):

Returns $\mathsf{K} = e(\mathsf{C}, \mathsf{sk}_{\mathsf{id}})$.

Conclusion

- ✓ Compiling HKR to DKR

Conclusion

- ✓ Compiling HKR to DKR
- ✓ Almost NIKE from Special IBE

Conclusion

- ✓ Compiling HKR to DKR
- ✓ Almost NIKE from Special IBE
- ✓ Exist based on most classical assumptions

Conclusion

- ✓ Compiling HKR to DKR
- ✓ Almost NIKE from Special IBE
- ✓ Exist based on most classical assumptions
- ➘ Construct such IBE outside ROM

Conclusion

- ✓ Compiling HKR to DKR
- ✓ Almost NIKE from Special IBE
- ✓ Exist based on most classical assumptions
- ➘ Construct such IBE outside ROM
- $\Rrightarrow$ Using ROM, obtain a real ID-NIKE

Conclusion

- ✓ Compiling HKR to DKR
- ✓ Almost NIKE from Special IBE
- ✓ Exist based on most classical assumptions
- Construct such IBE outside ROM
- $\Rrightarrow$ Using ROM, obtain a real ID-NIKE
- $\Rrightarrow$ In the SM, propose better IBE with auxiliary input