Swoosh: Efficient Lattice-Based Non-Interactive Key Exchange

Phillip Gajland[1,2] & Miguel Quaresma[1]

[1] Max Planck Institute for Security and Privacy

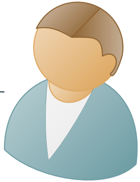
[2] Ruhr-University Bochum

RWC 2024: Real World Crypto Symposium, Toronto, Canada

Based on a USENIX 2024 paper with the same title [GdKQ+24].

$(a, g^a) \xleftarrow{\$} \mathsf{Gen}$

$\xrightarrow{g^a}$ $\xleftarrow{g^b}$



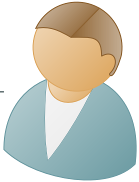
$(b, g^b) \xleftarrow{\$} \mathsf{Gen}$

$(g^b)^a \leftarrow \mathsf{SdK}(a, g^b)$

$(g^a)^b \leftarrow \mathsf{SdK}(b, g^a)$

$(a, g^a) \xleftarrow{\$} \mathsf{Gen}$

$\xrightarrow{\quad g^a \quad}$ $\xleftarrow{\quad g^b \quad}$

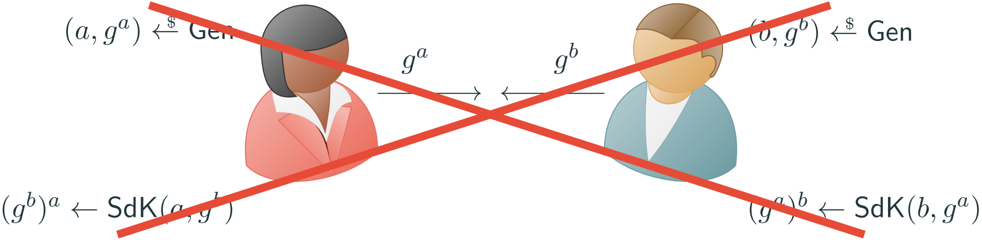


$(b, g^b) \xleftarrow{\$} \mathsf{Gen}$

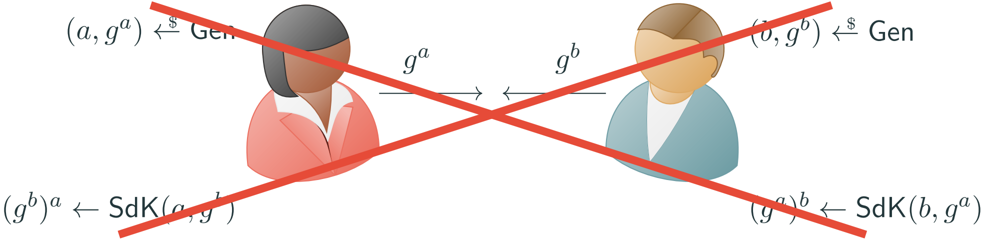
$(g^b)^a \leftarrow \mathsf{SdK}(a, g^b)$

$(g^a)^b \leftarrow \mathsf{SdK}(b, g^a)$

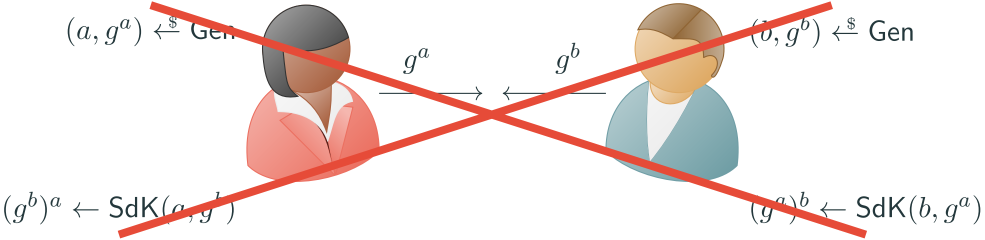
- Diffie-Hellman is widely deployed



- Diffie-Hellman is widely deployed
- Need a post-quantum replacement



- Diffie-Hellman is widely deployed
- Need a post-quantum replacement
- Folklore: *"Lattice-based non-interactive key exchange is impractical"*
  - Impossibility results [GKRS20]



- Diffie-Hellman is widely deployed
- Need a post-quantum replacement
- Folklore: *"Lattice-based non-interactive key exchange is impractical"*
  - Impossibility results [GKRS20]
- Our work: *"It's not **that** bad"*

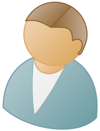
OUTLINE

OUTLINE

- NIKE vs. KEM Applications
- Scheme: Passive-SWOOSH
- Security Model

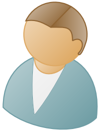
OUTLINE

- NIKE vs. KEM Applications
- Scheme: Passive-Swoosh
- Security Model
- Parameter choices
- Implementation details
- Comparison



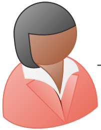
NON-INTERACTIVE KEY EXCHANGE (NIKE)

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

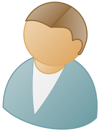


NON-INTERACTIVE KEY EXCHANGE (NIKE)

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

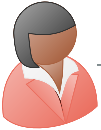


$\xrightarrow{pk_A}$

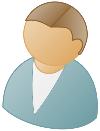


Non-interactive key exchange (NIKE)

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

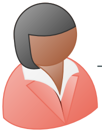


$\xrightarrow{pk_A}$



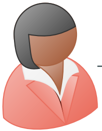
$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

NON-INTERACTIVE KEY EXCHANGE (NIKE)

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$  $\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$  $(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

NON-INTERACTIVE KEY EXCHANGE (NIKE)

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

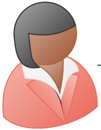


$\xrightarrow{\quad pk_A \quad}$ $\xleftarrow{\quad pk_B \quad}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$



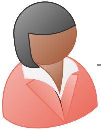
$\xrightarrow{\quad pk_A \quad}$ $\xleftarrow{\quad pk_B \quad}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

NON-INTERACTIVE KEY EXCHANGE (NIKE)

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$



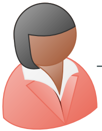
$\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$



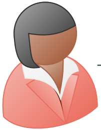
$\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$



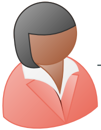
$\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$



$\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

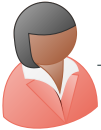
$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

NON-INTERACTIVE KEY EXCHANGE VS. KEY-ENCAPSULATION MECHANISMS

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$



$\xrightarrow{pk_A} \qquad \xleftarrow{pk_B}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

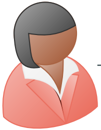
$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

$\xrightarrow{pk_A}$

NON-INTERACTIVE KEY EXCHANGE VS. KEY-ENCAPSULATION MECHANISMS

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$  $\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$  $(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$ $\qquad$ $k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$  $\xrightarrow{pk_A}$  $(c, k_{BA}) \xleftarrow{\$} \mathsf{Enc}(pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

$\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

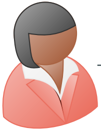
$\xrightarrow{pk_A}$

$\xleftarrow{c}$

$(c, k_{BA}) \xleftarrow{\$} \mathsf{Enc}(pk_A)$

NON-INTERACTIVE KEY EXCHANGE VS. KEY-ENCAPSULATION MECHANISMS

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$



$\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

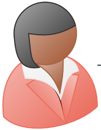
$\xrightarrow{pk_A}$

$\xleftarrow{c}$

$(c, k_{BA}) \xleftarrow{\$} \mathsf{Enc}(pk_A)$

$k_{AB} \leftarrow \mathsf{Dec}(sk_A, c)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$



$\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$

$(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$

$k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$

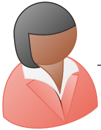
$\xrightarrow{pk_A}$

$\xleftarrow{c}$

$(c, k_{BA}) \xleftarrow{\$} \mathsf{Enc}(pk_A)$

$k_{AB} \leftarrow \mathsf{Dec}(sk_A, c)$

- **Correctness:** $\delta := \Pr[k_{AB} \neq k_{BA}]$ should be small.

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$  $\xrightarrow{pk_A}$ $\xleftarrow{pk_B}$  $(sk_B, pk_B) \xleftarrow{\$} \mathsf{Gen}$

$k_{AB} \leftarrow \mathsf{SdK}(sk_A, pk_B)$ $\qquad$ $k_{BA} \leftarrow \mathsf{SdK}(sk_B, pk_A)$

$(sk_A, pk_A) \xleftarrow{\$} \mathsf{Gen}$  $\xrightarrow{pk_A}$ $\xleftarrow{c}$  $(c, k_{BA}) \xleftarrow{\$} \mathsf{Enc}(pk_A)$

$k_{AB} \leftarrow \mathsf{Dec}(sk_A, c)$

- **Correctness:** $\delta := \Pr[k_{AB} \neq k_{BA}]$ should be small.
- **Security:** k_{AB} and k_{BA} should look random.

- Implicit Authentication using static Diffie-Hellman keys
  - OPTLS [KW16]
- Asynchronous key agreement
  - X3DH [MP16]
- Other
  - EDHOC [SMP24]
  - Group OSCORE [TSP+24]
  - More?

- Implicit Authentication using static Diffie-Hellman keys
  - OPTLS [KW16]
- Asynchronous key agreement
  - X3DH [MP16]
- Other
  - EDHOC [SMP24]
  - Group OSCORE [TSP+24]
  - More?

NIKE VS. KEM APPLICATIONS

- Implicit Authentication using static Diffie-Hellman keys
  - OPTLS [KW16]
- Asynchronous key agreement
  - X3DH [MP16]
- Other
  - EDHOC [SMP24]
  - Group OSCORE [TSP+24]
  - More?

NIKE VS. KEM APPLICATIONS

- Implicit Authentication using static Diffie-Hellman keys
  - OPTLS [KW16]
- Asynchronous key agreement
  - X3DH [MP16]
- Other
  - EDHOC [SMP24]
  - Group OSCORE [TSP+24]
  - More?

NIKE VS. KEM APPLICATIONS

- Implicit Authentication using static Diffie-Hellman keys
  - OPTLS [KW16]
- Asynchronous key agreement
  - X3DH [MP16]
- Other
  - EDHOC [SMP24]
  - Group OSCORE [TSP+24]
  - More?

- Implicit Authentication using static Diffie-Hellman keys
  - OPTLS [KW16]
- Asynchronous key agreement
  - X3DH [MP16]
- Other
  - EDHOC [SMP24]
  - Group OSCORE [TSP+24]
  - More?

- Implicit Authentication using static Diffie-Hellman keys
  - OPTLS [KW16]
- Asynchronous key agreement
  - X3DH [MP16]
- Other
  - EDHOC [SMP24]
  - Group OSCORE [TSP+24]
  - More?

- Implicit Authentication using static Diffie-Hellman keys
  - OPTLS [KW16]
- Asynchronous key agreement
  - X3DH [MP16]
- Other
  - EDHOC [SMP24]
  - Group OSCORE [TSP+24]
  - More?

LATTICE-BASED NIKE HAS IMPERFECT CORRECTNESS

$$\begin{pmatrix} \underbrace{\vec{s}}_{sk_A} & & & \end{pmatrix} \xleftarrow{\$} \mathsf{Gen}$$

LATTICE-BASED NIKE HAS IMPERFECT CORRECTNESS

$$\left(\underbrace{\boldsymbol{\vec{s}}}_{sk_A}, \underbrace{\boldsymbol{\vec{s}}\,\boldsymbol{A} + \boldsymbol{\vec{e}}}_{pk_A}\right) \xleftarrow{\$} \mathsf{Gen}$$

LATTICE-BASED NIKE HAS IMPERFECT CORRECTNESS

$$\left(\underbrace{\vec{s}}_{sk_A}, \underbrace{\vec{s}\boldsymbol{A} + \vec{e}}_{pk_A}\right) \xleftarrow{\$} \mathsf{Gen}$$

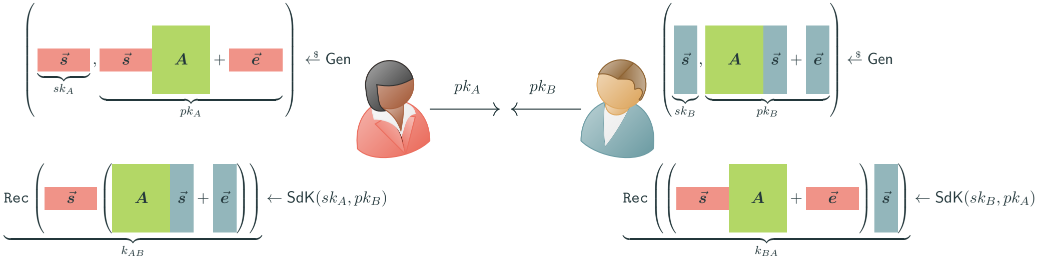
$$\left(\underbrace{\vec{s}}_{sk_B}, \underbrace{\boldsymbol{A}\vec{s} + \vec{e}}_{pk_B}\right) \xleftarrow{\$} \mathsf{Gen}$$

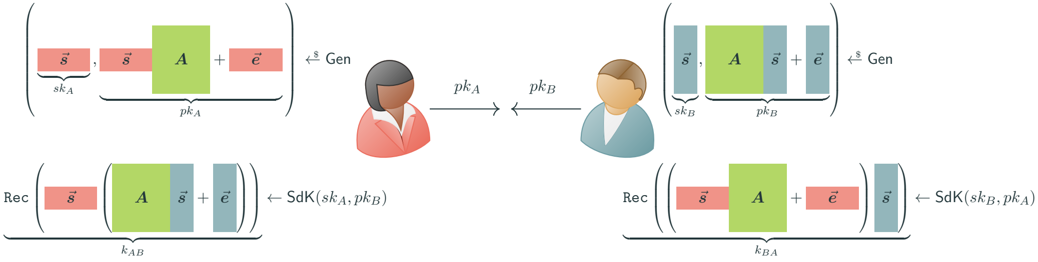
$$\Big(\underbrace{\vec{s}}_{sk_A}, \underbrace{\vec{s}\,\boldsymbol{A} + \vec{e}}_{pk_A}\Big) \xleftarrow{\$} \mathsf{Gen}$$

$$\xrightarrow{pk_A} \quad \xleftarrow{pk_B}$$

$$\Big(\underbrace{\vec{s}}_{sk_B}, \underbrace{\boldsymbol{A}\,\vec{s} + \vec{e}}_{pk_B}\Big) \xleftarrow{\$} \mathsf{Gen}$$

LATTICE-BASED NIKE HAS IMPERFECT CORRECTNESS



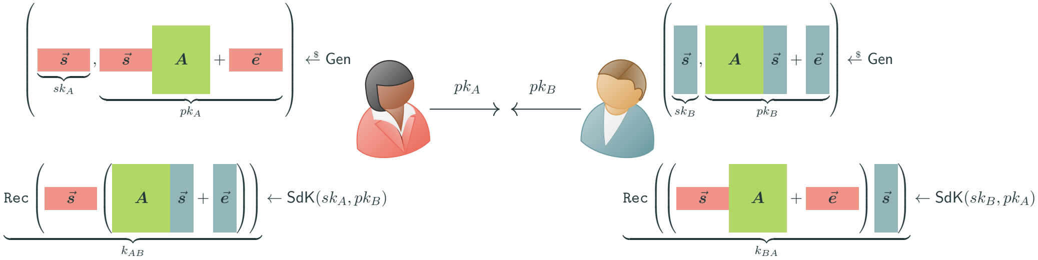


- **Correctness:**

$$\delta := \Pr\left[\mathsf{Rec}\left(\vec{s}\,\boldsymbol{A}\,\vec{s} + \vec{s}\,\vec{e}\right) \neq \mathsf{Rec}\left(\vec{s}\,\boldsymbol{A}\,\vec{s} + \vec{e}\,\vec{s}\right)\right] \leq \frac{4\beta^2 d^2 N}{q}.$$

- **Security:** k_{AB} and k_{BA} look random under the M-LWE assumption.

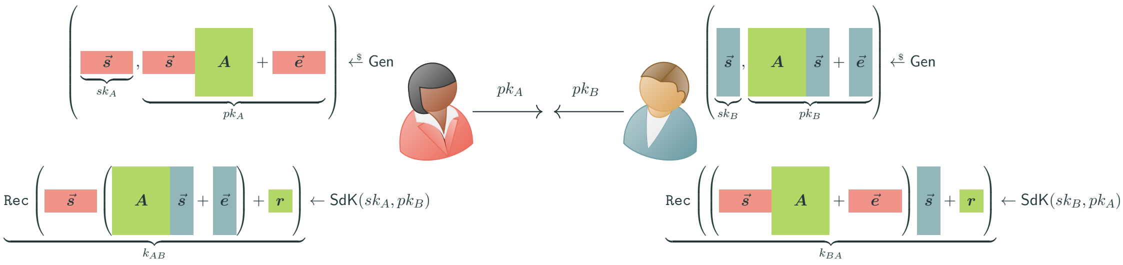
LATTICE-BASED NIKE HAS IMPERFECT CORRECTNESS



- **Semi-Malicious Correctness:** (we add $\boldsymbol{r} := \mathsf{H}(pk_A, pk_B)$)

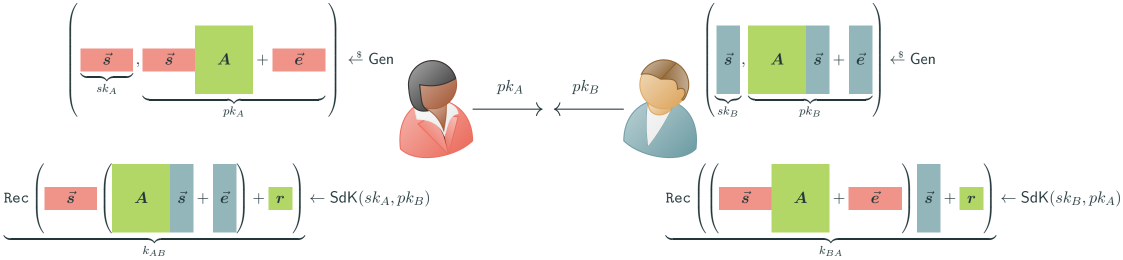
- **Security:** k_{AB} and k_{BA} look random under the M-LWE assumption.

LATTICE-BASED NIKE HAS IMPERFECT CORRECTNESS



- **Semi-Malicious Correctness:** (we add $\boxed{\boldsymbol{r}} := \mathsf{H}(pk_A, pk_B)$)

- **Security:** k_{AB} and k_{BA} look random under the M-LWE assumption.

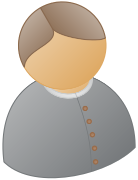


- **Semi-Malicious Correctness:** (we add $r := \mathsf{H}(pk_A, pk_B)$)

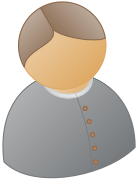
$$\delta := \Pr\left[\mathsf{Rec}\left(\vec{s}\,A\,\vec{s} + \vec{s}\,\vec{e} + r\right) \neq \mathsf{Rec}\left(\vec{s}\,A\,\vec{s} + \vec{e}\,\vec{s} + r\right)\right] \lesssim Q_{\mathsf{H}} \cdot \frac{4\beta^2 d^2 N}{q}.$$

- **Security:** k_{AB} and k_{BA} look random under the M-LWE assumption.

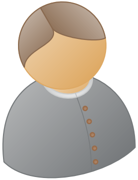
SECURITY MODEL [CKS08]



$b \xleftarrow{\$} \{0, 1\}$



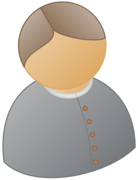
$b \xleftarrow{\$} \{0,1\}$



$$\downarrow$$

$$b'$$

$b \xleftarrow{\$} \{0, 1\}$



if $b = b'$

return win

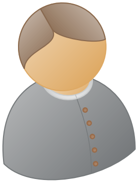
$$\downarrow$$

b'

$$\xleftarrow{\quad\texttt{RegHonUsr(ID)}\quad}$$

$b \xleftarrow{\$} \{0, 1\}$

$$\xrightarrow{\quad pk \quad}$$



$\downarrow$

b'

if $b = b'$

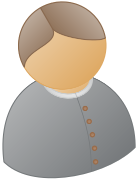
return win

SECURITY MODEL [CKS08]

$b \xleftarrow{\$} \{0, 1\}$

$\xleftarrow{\mathtt{RegHonUsr}(\mathtt{ID})}$

$\xrightarrow{\quad pk \quad}$



if $b = b'$

return win

$\xleftarrow{\mathtt{TestQue}(\mathtt{ID}_1, \mathtt{ID}_2)}$

$\downarrow$

b'

$$\xleftarrow{\texttt{RegHonUsr}(\texttt{ID})}$$

$b \xleftarrow{\$} \{0,1\}$

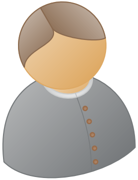
$$\xrightarrow{pk}$$

if $b = 0$

$\quad k := \mathsf{SdK}(sk_1, pk_2)$

if $b = 1$

$\quad k \xleftarrow{\$} \mathcal{SKS}$



$$\xleftarrow{\texttt{TestQue}(\texttt{ID}_1, \texttt{ID}_2)}$$

$$\xrightarrow{k}$$

$\downarrow$

b'

if $b = b'$

$\quad$ **return** win

$\texttt{RegHonUsr}(\texttt{ID})$ $\longleftarrow$

$b \xleftarrow{\$} \{0,1\}$

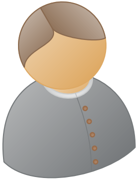
pk $\longrightarrow$

if $b = 0$

$k := \mathsf{SdK}(sk_1, pk_2)$

if $b = 1$

$k \xleftarrow{\$} \mathcal{SKS}$



Only for DKR

if $b = b'$

return win

$\texttt{TestQue}(\texttt{ID}_1, \texttt{ID}_2)$ $\longleftarrow$

$\downarrow$ b'

k $\longrightarrow$

Security model [CKS08]

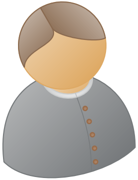
$$b \xleftarrow{\$} \{0,1\}$$

if $b = 0$

$\quad k := \mathsf{SdK}(sk_1, pk_2)$

if $b = 1$

$\quad k \xleftarrow{\$} \mathcal{SKS}$



$\xleftarrow{\texttt{RegHonUsr}(\texttt{ID})}$

$\xrightarrow{pk}$

$\xleftarrow{\texttt{RegCorUsr}(\texttt{ID}, pk')}$

Only for DKR

if $b = b'$

return win

$\xleftarrow{\texttt{TestQue}(\texttt{ID}_1, \texttt{ID}_2)}$

$\xrightarrow{k}$

$\downarrow$

b'

SECURITY MODEL [CKS08]

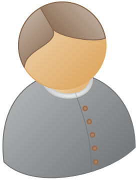
$b \xleftarrow{\$} \{0,1\}$

if $b = 0$

  $k := \mathsf{SdK}(sk_1, pk_2)$

if $b = 1$

  $k \xleftarrow{\$} \mathcal{SKS}$



$\xleftarrow{\texttt{RegHonUsr}(\texttt{ID})}$

$\xrightarrow{pk}$

$\xleftarrow{\texttt{RegCorUsr}(\texttt{ID}, pk')}$

$\xleftarrow{\texttt{RevCorQue}(\texttt{ID}_1, \texttt{ID}_2)}$

$\xrightarrow{\mathsf{SdK}(sk_1, pk_2')}$

$\xleftarrow{\texttt{TestQue}(\texttt{ID}_1, \texttt{ID}_2)}$

$\xrightarrow{k}$

$\left.\begin{array}{l} \\ \\ \\ \end{array}\right\}$ Only for DKR

$\downarrow$

b'

if $b = b'$

  return win

- Diffie-Hellman has active security "almost for free" due to perfect correctness
- Passive-SWOOSH needs:
  - Semi-malicious correctness
  - Non-interactive zero-knowledge (NIZK) proof of knowledge
    - *"I know the corresponding secret key of the public key"*

- Diffie-Hellman has active security "almost for free" due to perfect correctness
- Passive-SWOOSH needs:
  - Semi-malicious correctness
  - Non-interactive zero-knowledge (NIZK) proof of knowledge
    - *"I know the corresponding secret key of the public key"*

- Diffie-Hellman has active security "almost for free" due to perfect correctness
- Passive-SWOOSH needs:
    - Semi-malicious correctness
    - Non-interactive zero-knowledge (NIZK) proof of knowledge
        - *"I know the corresponding secret key of the public key"*

- Diffie-Hellman has active security "almost for free" due to perfect correctness
- Passive-Swoosh needs:
  - Semi-malicious correctness
  - Non-interactive zero-knowledge (NIZK) proof of knowledge
    - *"I know the corresponding secret key of the public key"*

- Diffie-Hellman has active security "almost for free" due to perfect correctness
- Passive-SWOOSH needs:
  - Semi-malicious correctness
  - Non-interactive zero-knowledge (NIZK) proof of knowledge
    - *"I know the corresponding secret key of the public key"*

PARAMETERS AND IMPLEMENTATION

Parameter	Description	Value
β	upper bound on $\lVert\vec{\boldsymbol{s}}\rVert_\infty = \lVert\vec{\boldsymbol{e}}\rVert_\infty$	1
q	prime modulus	$2^{214} - 255$
d	dim of $\mathcal{R}_q := \mathbb{Z}_q[X]/(X^d + 1)$	256
l	# factors $X^d + 1$ splits into $\mod q$	128
N	height of the $\boldsymbol{A}$ matrix	32
n	lattice dimension	8192
χ	secret / noise distribution (ternary)	$p(-1) = 25\%$ $p(0) = 50\%$ $p(1) = 25\%$

- Passive-SWOOSH (Semi-malicious correctness) implementation in Rust and Jasmin[1]
  - NIZK not included
  - Performance penalty depends on context
- Main optimisation targets
  - Key Generation Gen
  - Shared Key Derivation SdK

[1] https://github.com/jasmin-lang/jasmin/

- Passive-SWOOSH (Semi-malicious correctness) implementation in Rust and Jasmin[1]
  - NIZK not included
  - Performance penalty depends on context
- Main optimisation targets
  - Key Generation Gen
  - Shared Key Derivation SdK

[1] https://github.com/jasmin-lang/jasmin/

- Passive-Swoosh (Semi-malicious correctness) implementation in Rust and Jasmin[1]
  - NIZK not included
  - Performance penalty depends on context
- Main optimisation targets
  - Key Generation Gen
  - Shared Key Derivation SdK

[1] https://github.com/jasmin-lang/jasmin/

- Passive-SWOOSH (Semi-malicious correctness) implementation in Rust and Jasmin[1]
  - NIZK not included
  - Performance penalty depends on context
- Main optimisation targets
  - Key Generation Gen
  - Shared Key Derivation SdK

[1] https://github.com/jasmin-lang/jasmin/

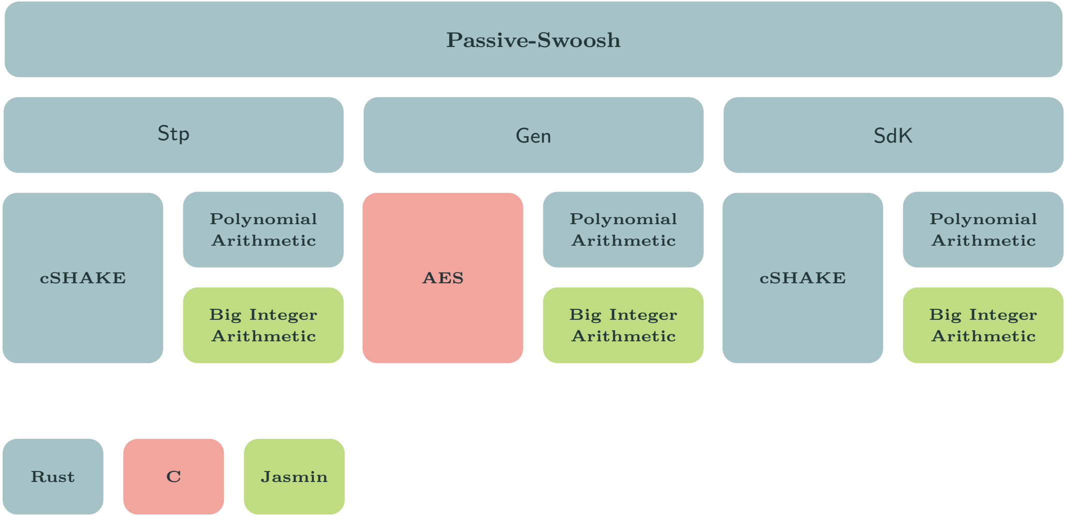
- Passive-SWOOSH (Semi-malicious correctness) implementation in Rust and Jasmin[1]
  - NIZK not included
  - Performance penalty depends on context
- Main optimisation targets
  - Key Generation Gen
  - Shared Key Derivation SdK

[1] https://github.com/jasmin-lang/jasmin/

- Passive-SWOOSH (Semi-malicious correctness) implementation in Rust and Jasmin[1]
  - NIZK not included
  - Performance penalty depends on context
- Main optimisation targets
  - Key Generation Gen
  - Shared Key Derivation SdK

[1] https://github.com/jasmin-lang/jasmin/

IMPLEMENTATION: OVERVIEW



- Key Generation Gen
  - Polynomial multiplication:
    - Number Theoretic Transform (NTT) $\approx 10\%$
    - Big Integer Arithmetic
  - Noise sampling $\vec{s}$, $\vec{e}$
- Shared Key Derivation SdK
  - Random offset computation $r := \mathsf{H}(pk_A, pk_B)$

- Key Generation Gen
  - Polynomial multiplication:
    - Number Theoretic Transform (NTT) $\approx 10\%$
    - Big Integer Arithmetic
  - Noise sampling $\vec{s}$, $\vec{e}$
- Shared Key Derivation SdK
  - Random offset computation $r := \mathsf{H}(pk_A, pk_B)$

- Key Generation Gen
  - Polynomial multiplication:
    - Number Theoretic Transform (NTT) $\approx 10\%$
    - Big Integer Arithmetic
  - Noise sampling $\vec{s}$, $\vec{e}$
- Shared Key Derivation SdK
  - Random offset computation $r := \mathsf{H}(pk_A, pk_B)$

- Key Generation Gen
  - Polynomial multiplication:
    - Number Theoretic Transform (NTT) $\approx 10\%$
    - Big Integer Arithmetic
  - Noise sampling $\vec{s}$, $\vec{e}$
- Shared Key Derivation SdK
  - Random offset computation $r := \mathsf{H}(pk_A, pk_B)$

- Key Generation Gen
  - Polynomial multiplication:
    - Number Theoretic Transform (NTT) $\approx 10\%$
    - Big Integer Arithmetic
  - Noise sampling $\vec{s}$, $\vec{e}$
- Shared Key Derivation SdK
  - Random offset computation $r := \mathsf{H}(pk_A, pk_B)$

- Key Generation Gen
  - Polynomial multiplication:
    - Number Theoretic Transform (NTT) $\approx 10\%$
    - Big Integer Arithmetic
  - Noise sampling $\vec{s}$, $\vec{e}$
- Shared Key Derivation SdK
  - Random offset computation $r := \mathsf{H}(pk_A, pk_B)$

- Key Generation Gen
  - Polynomial multiplication:
    - Number Theoretic Transform (NTT) $\approx 10\%$
    - Big Integer Arithmetic
  - Noise sampling $\vec{s}$, $\vec{e}$
- Shared Key Derivation SdK
  - Random offset computation $\boldsymbol{r} := \mathsf{H}(pk_A, pk_B)$

- Schoolbook multiplication is inefficient: $\mathcal{O}(d^2)$
- NTT is more efficient: $\mathcal{O}(d \log(d))$
  - In-place NTT reduces memory usage
  - NTT is part of the scheme (i.e. matrix $\boldsymbol{A}$ and public keys in NTT domain)

- Schoolbook multiplication is inefficient: $\mathcal{O}(d^2)$
- NTT is more efficient: $\mathcal{O}(d\log(d))$
    - In-place NTT reduces memory usage
    - NTT is part of the scheme (i.e. matrix $\boldsymbol{A}$ and public keys in NTT domain)

- Noise sampling $\vec{s}$, $\vec{e}$
    - Coefficients sampled from *Centered Binomial Distribution* (CBD)
    - CBD generated from output of a ***Pseudo-Random Function*** **(PRF)**
    - **Ternary** coefficients computed from two bits using signed reduction modulo 3
- Random offset computation $r := \mathsf{H}(pk_A, pk_B)$
    - Rejection sampling on output of an *eXtendable Output Function* (XOF)
- Instantiation of primitives dictates performance: AES256-CTR[2] & cSHAKE-256

[2] https://bench.cr.yp.to/impl-stream/aes256ctr.html

- Noise sampling $\vec{s}$, $\vec{e}$
  - Coefficients sampled from *Centered Binomial Distribution* (CBD)
  - CBD generated from output of a ***Pseudo-Random Function* (PRF)**
  - **Ternary** coefficients computed from two bits using signed reduction modulo 3
- Random offset computation $r := \mathsf{H}(pk_A, pk_B)$
  - Rejection sampling on output of an *eXtendable Output Function* (XOF)
- Instantiation of primitives dictates performance: AES256-CTR[2] & cSHAKE-256

[2] https://bench.cr.yp.to/impl-stream/aes256ctr.html

- Noise sampling $\vec{s}$, $\vec{e}$
  - Coefficients sampled from *Centered Binomial Distribution* (CBD)
  - CBD generated from output of a ***Pseudo-Random Function*** **(PRF)**
  - **Ternary** coefficients computed from two bits using signed reduction modulo 3
- Random offset computation $\boldsymbol{r} := \mathsf{H}(pk_A, pk_B)$
  - Rejection sampling on output of an ***eXtendable Output Function*** **(XOF)**
- Instantiation of primitives dictates performance: AES256-CTR[2] & cSHAKE-256

[2] https://bench.cr.yp.to/impl-stream/aes256ctr.html

- Noise sampling $\vec{s}$, $\vec{e}$
  - Coefficients sampled from *Centered Binomial Distribution* (CBD)
  - CBD generated from output of a ***Pseudo-Random Function* (PRF)**
  - **Ternary** coefficients computed from two bits using signed reduction modulo 3
- Random offset computation $\boldsymbol{r} := \mathsf{H}(pk_A, pk_B)$
  - Rejection sampling on output of an ***eXtendable Output Function* (XOF)**
- Instantiation of primitives dictates performance: AES256-CTR[2] & cSHAKE-256

[2] https://bench.cr.yp.to/impl-stream/aes256ctr.html

Scheme (variant)	**Assumption**	**PQ** [3]	**NI** [4]	**Size (bytes)**		**Cycles**	
				c	pk	Gen	Enc + Dec **or** SdK
ECDH (`X25519`)	CDH	✗	✓	—	32	28 187	87 942
CRYSTALS-Kyber (`Kyber-768`)	M-LWE	✓	✗	1 088	1 184	200 302	539 108
Classic McEliece (`mceliece348864`)	Binary Goppa Codes	✓	✗	96	261 120	46 715 060	143 178
CTIDH (`CTIDH-1024`)	CSIDH	✓	✓	—	128	469 520 000	511 190 000
This work (Passive-Swoosh)	M-LWE	✓	✓	—	221 184	146 920 890	10 612 666

[3] Post-quantum

[4] Non-interactive

COMPARISON OF SELECT POST-QUANTUM KEMS AND NIKES

Scheme (variant)	Assumption	PQ [3]	NI [4]	Size (bytes)		Cycles	
				c	pk	Gen	Enc + Dec or SdK
ECDH (`X25519`)	CDH	✗	✓	—	32	28 187	87 942
CRYSTALS-Kyber (`Kyber-768`)	M-LWE	✓	✗	1 088	1 184	200 302	539 108
Classic McEliece (`mceliece348864`)	Binary Goppa Codes	✓	✗	96	261 120	46 715 060	143 178
CTIDH (`CTIDH-1024`)	CSIDH	✓	✓	—	128	469 520 000	511 190 000
This work (Passive-Swoosh)	M-LWE	✓	✓	—	221 184	146 920 890	10 612 666

- Diffie-Hellman is most efficient but not PQ secure
- PQC KEMs are faster than Passive-Swoosh but require interaction
- PQ NIKEs: trade-off between key size and speed

[3]Post-quantum
[4]Non-interactive

SUMMARY

Contributions:

- M-LWE based NIKE, with strong correctness and proof in the QROM.
  - Generic transformation from passive to active security using NIZKs.
- Optimised implementation of Passive-SWOOSH, written in Rust and Jasmin.
  - Parameters achieving 120 bits of security against quantum adversaries.
  - Smaller public keys than Classic McEliece KEM and faster than CTIDH NIKE.

ia.cr/2023/271 — github.com/MQuaresma/pswoosh

phillip.gajland@{mpi-sp.org,rub.de}
miguel.quaresma@mpi-sp.org
p4i11ip, miguel__mq

Contributions:

- M-LWE based NIKE, with strong correctness and proof in the QROM.
  - Generic transformation from passive to active security using NIZKs.
- Optimised implementation of Passive-SWOOSH, written in Rust and Jasmin.
  - Parameters achieving 120 bits of security against quantum adversaries.
  - Smaller public keys than Classic McEliece KEM and faster than CTIDH NIKE.

ia.cr/2023/271 — github.com/MQuaresma/pswoosh

phillip.gajland@{mpi-sp.org,rub.de}
miguel.quaresma@mpi-sp.org
p4i11ip, miguel__mq

Contributions:

- M-LWE based NIKE, with strong correctness and proof in the QROM.
  - Generic transformation from passive to active security using NIZKs.
- Optimised implementation of Passive-Swoosh, written in Rust and Jasmin.
  - Parameters achieving 120 bits of security against quantum adversaries.
  - Smaller public keys than Classic McEliece KEM and faster than CTIDH NIKE.

ia.cr/2023/271 — github.com/MQuaresma/pswoosh

phillip.gajland@{mpi-sp.org,rub.de}
miguel.quaresma@mpi-sp.org
p4i11ip, miguel__mq

Contributions:

- M-LWE based NIKE, with strong correctness and proof in the QROM.
  - Generic transformation from passive to active security using NIZKs.
- Optimised implementation of Passive-SWOOSH, written in Rust and Jasmin.
  - Parameters achieving 120 bits of security against quantum adversaries.
  - Smaller public keys than Classic McEliece KEM and faster than CTIDH NIKE.

ia.cr/2023/271 — github.com/MQuaresma/pswoosh

phillip.gajland@{mpi-sp.org,rub.de}
miguel.quaresma@mpi-sp.org
p4i11ip, miguel__mq

Contributions:

- M-LWE based NIKE, with strong correctness and proof in the QROM.
  - Generic transformation from passive to active security using NIZKs.
- Optimised implementation of Passive-Swoosh, written in Rust and Jasmin.
  - Parameters achieving 120 bits of security against quantum adversaries.
  - Smaller public keys than Classic McEliece KEM and faster than CTIDH NIKE.

ia.cr/2023/271 — github.com/MQuaresma/pswoosh

phillip.gajland@{mpi-sp.org,rub.de}
miguel.quaresma@mpi-sp.org
p4i11ip, miguel__mq

Contributions:

- M-LWE based NIKE, with strong correctness and proof in the QROM.
  - Generic transformation from passive to active security using NIZKs.
- Optimised implementation of Passive-SWOOSH, written in Rust and Jasmin.
  - Parameters achieving 120 bits of security against quantum adversaries.
  - Smaller public keys than Classic McEliece KEM and faster than CTIDH NIKE.

ia.cr/2023/271 — github.com/MQuaresma/pswoosh

phillip.gajland@{mpi-sp.org,rub.de}
miguel.quaresma@mpi-sp.org
p4i11ip, miguel__mq

[ABC+22] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions.

[BBC+21] Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, Benjamin Smith, and Jana Sotáková. CTIDH: faster constant-time CSIDH. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):351–387, 2021. https://tches.iacr.org/index.php/TCHES/article/view/9069.

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228, New York, NY, USA, April 24–26, 2006. Springer, Heidelberg, Germany.

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 127–145, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[GdKQ+24] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. Swoosh: Efficient lattice-based non-interactive key exchange. In *33rd USENIX Security Symposium (USENIX Security 24)*, Philadelphia, PA, August 2024. USENIX Association.

[GKRS20] Siyao Guo, Pritish Kamath, Alon Rosen, and Katerina Sotiraki. Limits on the efficiency of (ring) LWE based non-interactive key exchange. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 374–395, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.

[KW16] Hugo Krawczyk and Hoeteck Wee. The OPTLS protocol and TLS 1.3. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 81–96, Saarbruecken, Germany, March 2016. IEEE.

[MP16] Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol (revision 1). Part of the Signal Protocol Documentation, 2016. https://signal.org/docs/specifications/x3dh/x3dh.pdf.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.

[SMP24] Göran Selander, John Preu*ss* Mattsson, and Francesca Palombini. Ephemeral Diffie-Hellman Over COSE (EDHOC). Internet-Draft draft-ietf-lake-edhoc-23, Internet Engineering Task Force, January 2024. Work in Progress.

[TSP+24] Marco Tiloca, Göran Selander, Francesca Palombini, John Preu*ss* Mattsson, and Rikard Höglund. Group Object Security for Constrained RESTful Environments (Group OSCORE). Internet-Draft draft-ietf-core-oscore-groupcomm-21, Internet Engineering Task Force, March 2024. Work in Progress.