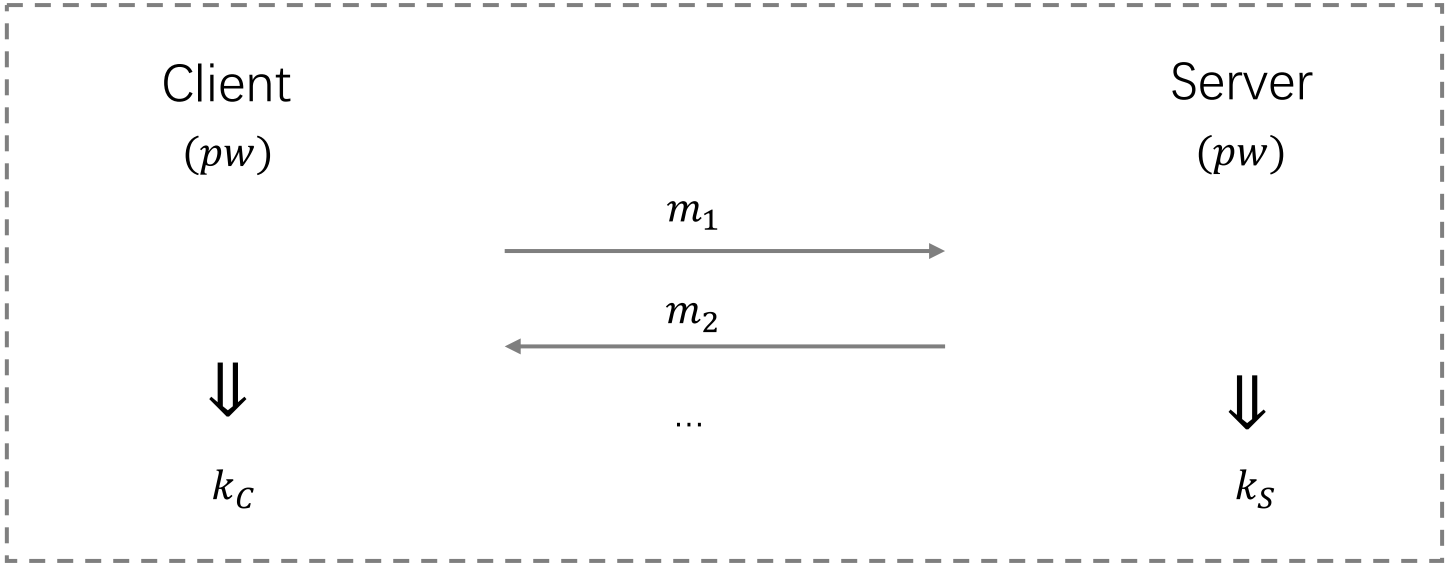
Universal Composable Password Authenticated Key Exchange for the Post-Quantum World

You Lyu, Shengli Liu, Shuai Han
Shanghai Jiao Tong University

Contents

Contents

Password Authenticated Key Exchange



Client

(pw)

Server

(pw)

m_1

m_2

…

$\Downarrow$

k_C

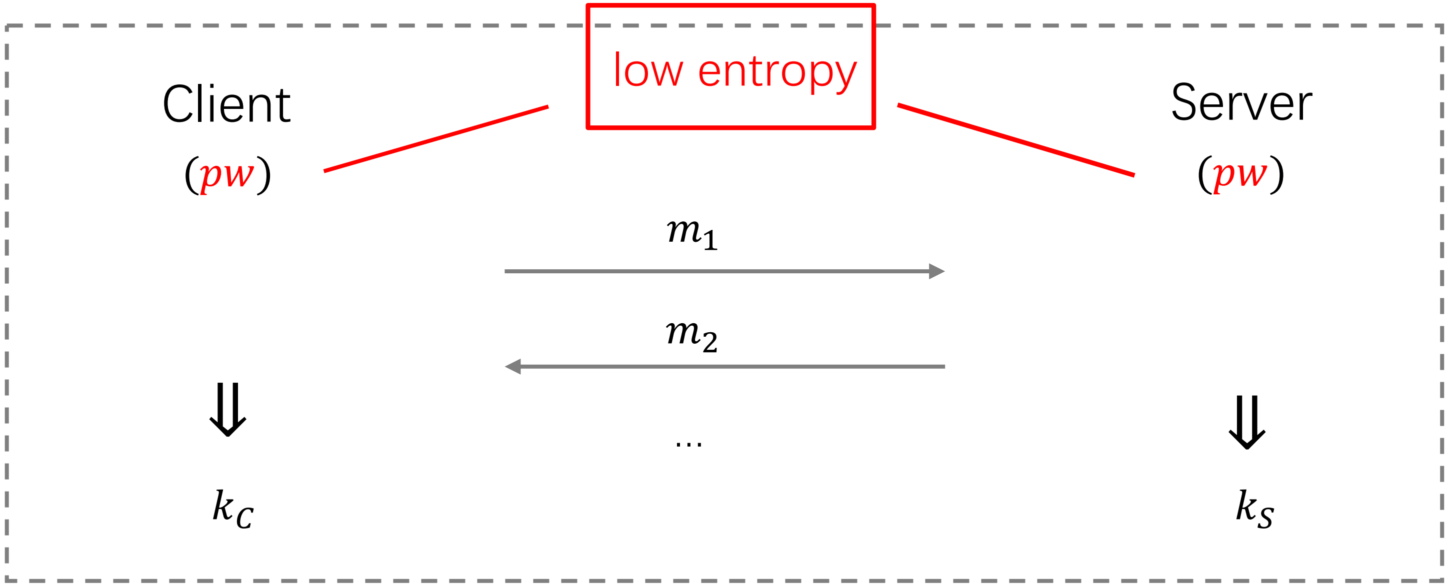
$\Downarrow$

k_S

The Goal of PAKE :

- Authentication：Client and Server can authenticate each other
- Key Exchange： Client and Server can exchange a pseudo-random session key.

Security Requirements

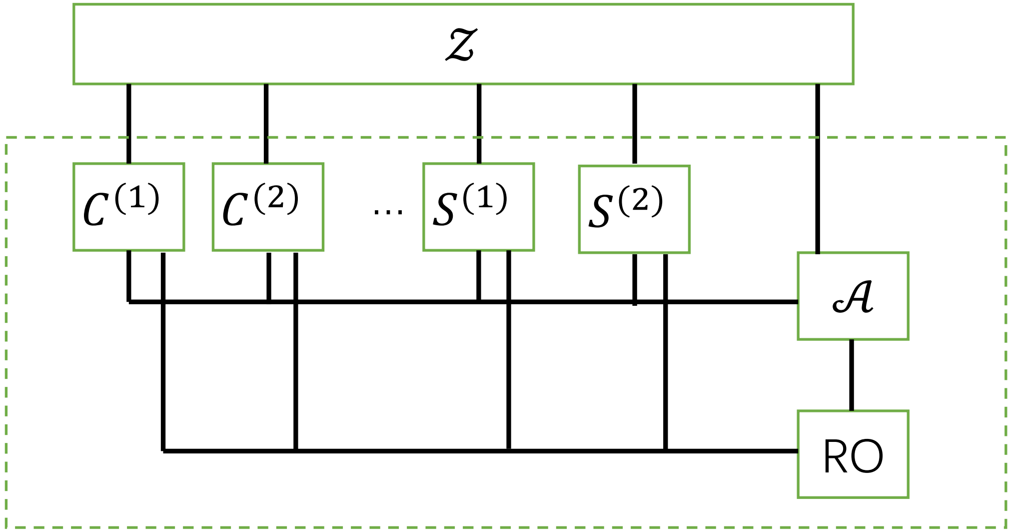


Note that the password only has low entropy.

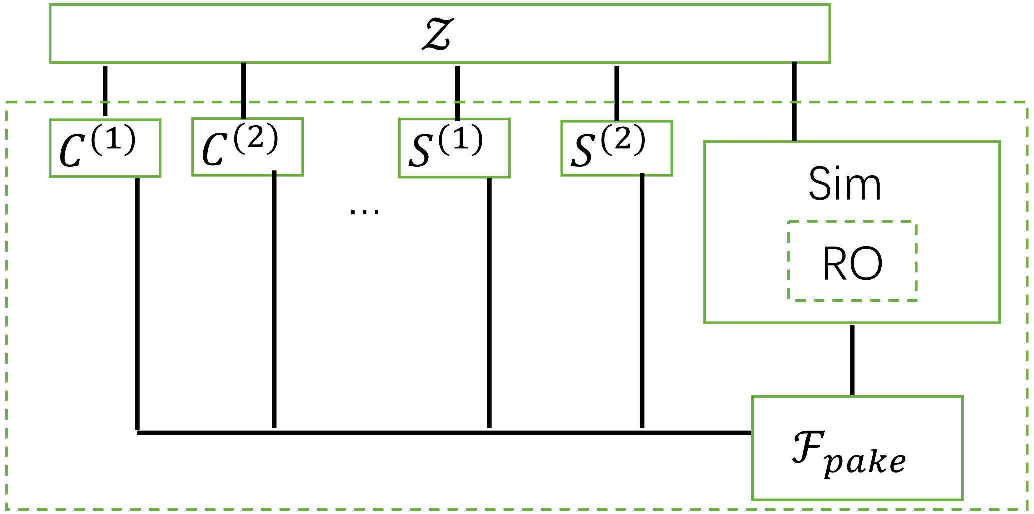
Security Requirements:

- The best strategy for adversary is to implement online-dictionary attack (guess password online)
- Resist offline-dictionary attack

UC-security for PAKE



Real World



Ideal World

Roughly speaking, to prove UC security, we need to construct a simulator Sim s.t.

- Sim can simulate indistinguishable transcript of PAKE protocol in the real world.
- Sim can simulate indistinguishable output session key for each client/server.
- Sim has no information of pw, except with a Testpw() oracle that tells whether a pw is the password client/server uses. Sim can only Testpw once for a client/server instance.

Our Contribution

- New generic construction for UC-secure PAKE in ROM, which implies
  a) UC-secure PAKE in ROM from LWE assumption
  b) UC-secure PAKE in ROM from GA-DDH assumption (the first from isogenies)
- New generic construction for UC-secure PAKE in QROM (the first), which implies
  a) UC-secure PAKE in QROM from LWE assumption (with super-poly modulus q)
  b) UC-secure PAKE in QROM from GA-DDH assumption

Contents

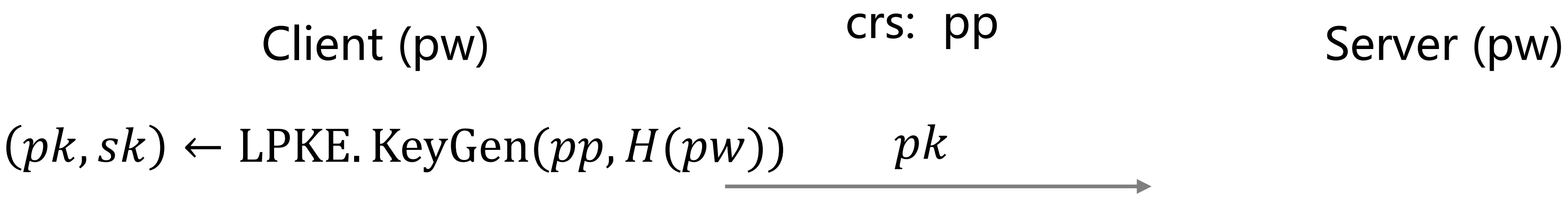
Basic Idea: make PKE associate with pw

We introduce a labeled public key encryption LPKE as our fundamental building block.

- LPKE.Setup: It outputs a public parameter pp and a trapdoor td.
- LPKE.KeyGen(pp, b = H(pw)): It takes as input a label (H(pw) in the PAKE setting) and outputs a key pair (pk, sk)
- LPKE.Enc(pp, pk, b = H(pw), m): It outputs a ciphertext c
- LPKE.Dec(pp, sk, c): It outputs a message m.
- LPKE.Check(td, pk, b): It outputs a bit β indicates whether b is a label of pk.

With LPKE, there is a natural idea to construct a PAKE protocol.

PAKE from LPKE



Client (pw)

crs: pp

Server (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$\xrightarrow{pk}$

PAKE from LPKE

crs: pp

Client (pw)

Server (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$\xrightarrow{\quad pk \quad}$

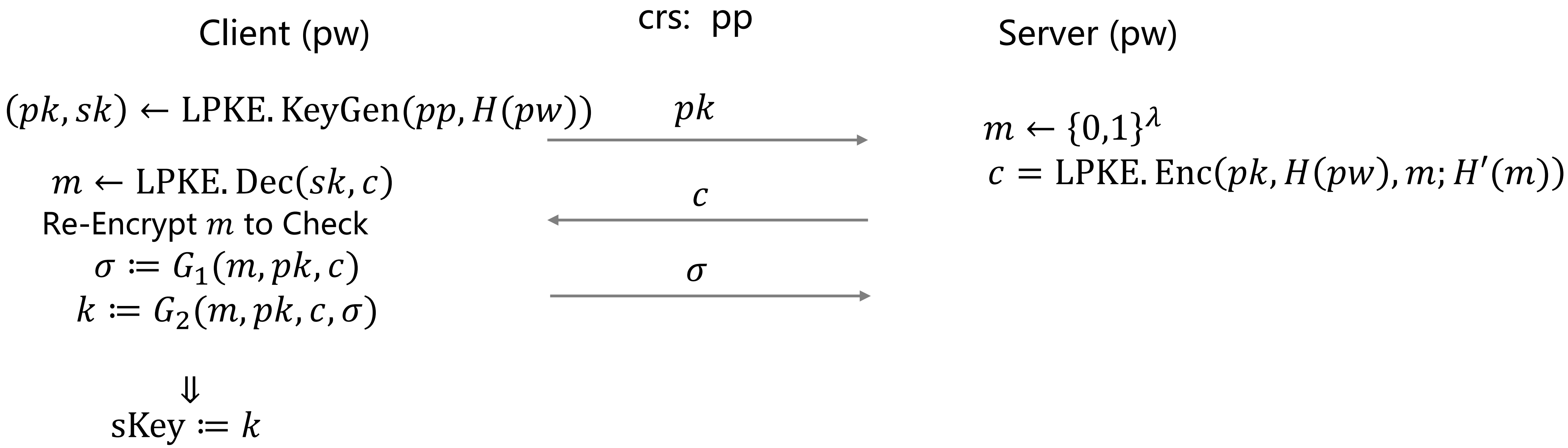
$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

$\xleftarrow{\quad c \quad}$

Fujisaki-Okamoto transform

PAKE from LPKE



crs: pp

Client (pw)

Server (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

pk

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

c

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\sigma := G_1(m, pk, c)$

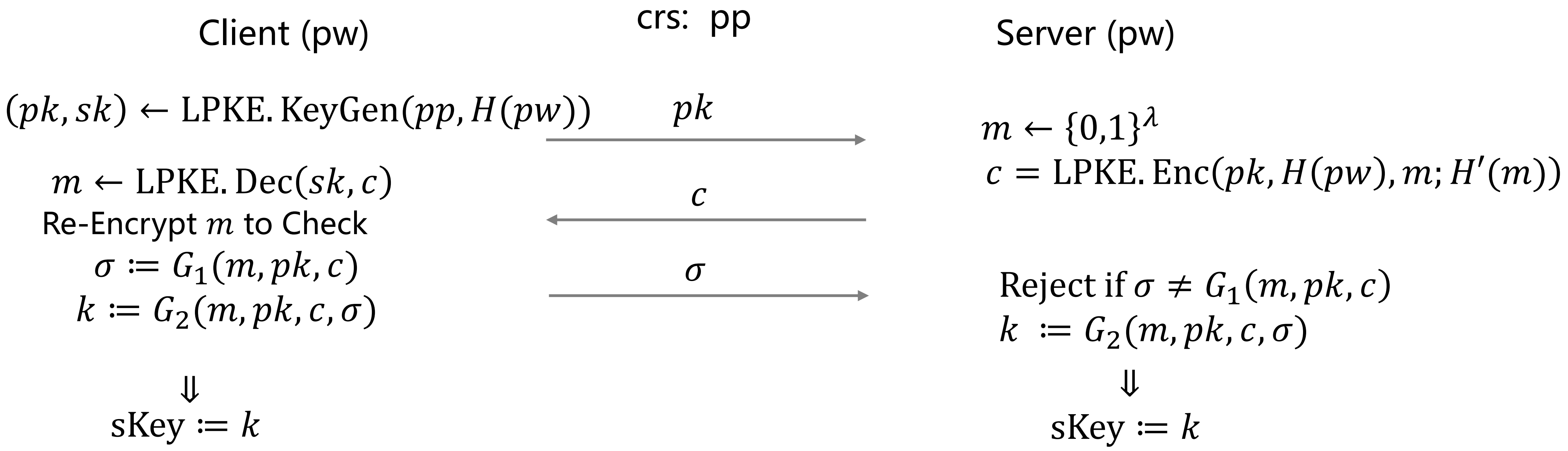
σ

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

PAKE from LPKE



crs: pp

Client (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\sigma := G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

$\xrightarrow{\;pk\;}$

$\xleftarrow{\;c\;}$

$\xrightarrow{\;\sigma\;}$

Server (pw)

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

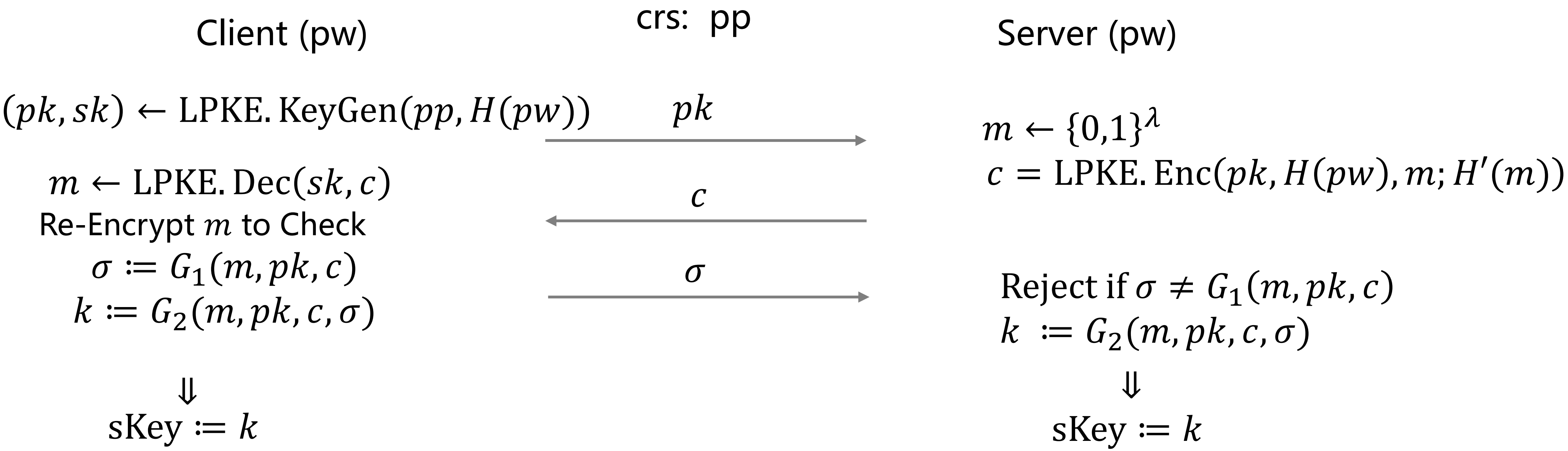
Reject if $\sigma \neq G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

PAKE from LPKE



crs: pp

Client (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\sigma := G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

$\xrightarrow{pk}$

$\xleftarrow{c}$

$\xrightarrow{\sigma}$

Server (pw)

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

Reject if $\sigma \neq G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

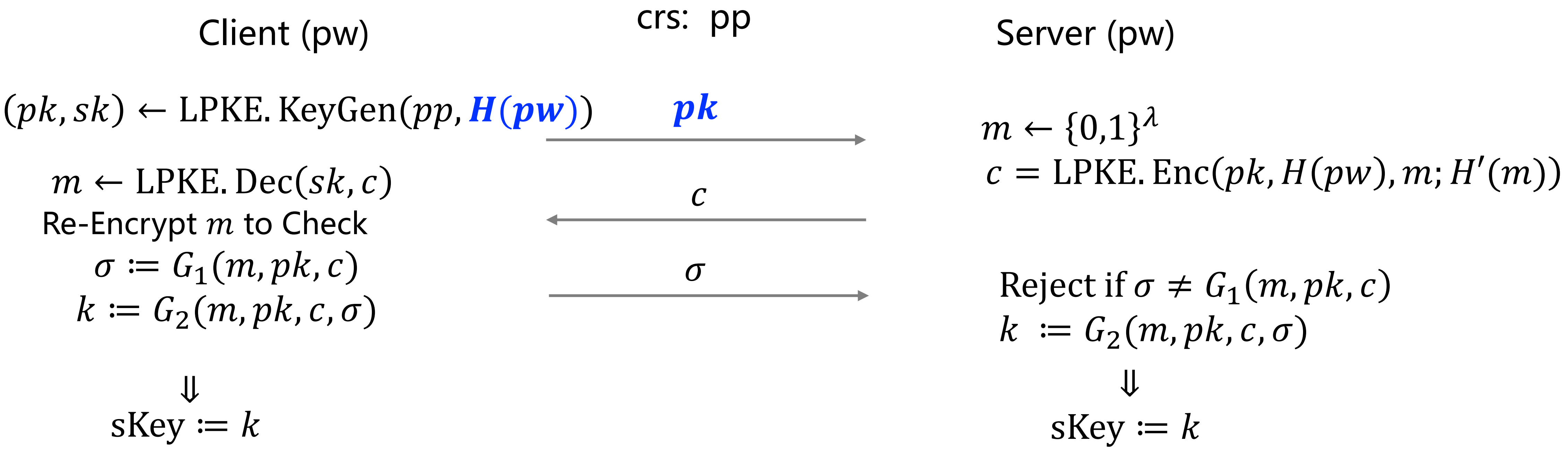
$\mathrm{sKey} := k$

To prove UC security with LPKE of PAKE, here are two key points:

1. What security properties are needed for LPKE ?

2. How the simulator works?

Simulation for the First Message



Client (pw)

crs: pp

Server (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, \boldsymbol{H(pw)})$

$\boldsymbol{pk}$ $\longrightarrow$

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

$\longleftarrow$ c

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\sigma := G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

σ $\longrightarrow$

Reject if $\sigma \neq G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

$\Downarrow$

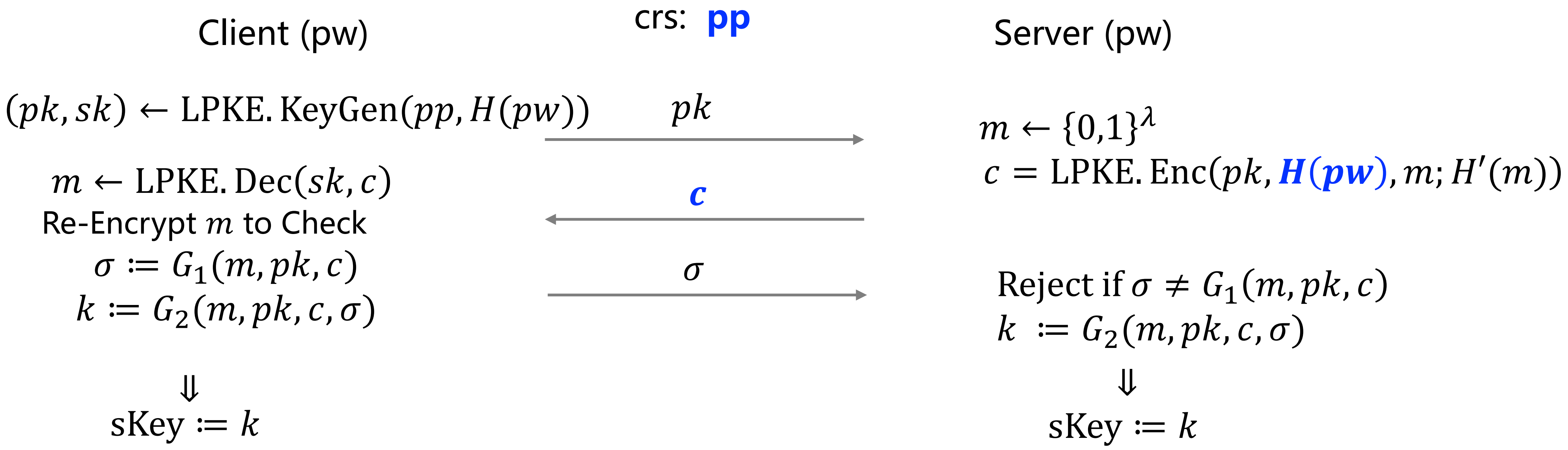
$\mathrm{sKey} := k$

To simulate the first message pk,

Required LPKE property: For every tag b, $\{pk : (pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(b)\} \approx_c \{pk : pk \leftarrow_{\$} \mathcal{PK}\}$

Simulation of the first message: Sim can simulate pk by $pk \leftarrow_{\$} \mathcal{PK}$

Simulation for the Second Message



crs: **pp**

Client (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\sigma := G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

$\xrightarrow{pk}$

$\xleftarrow{\boldsymbol{c}}$

$\xrightarrow{\sigma}$

Server (pw)

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, \boldsymbol{H(pw)}, m; H'(m))$

Reject if $\sigma \neq G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

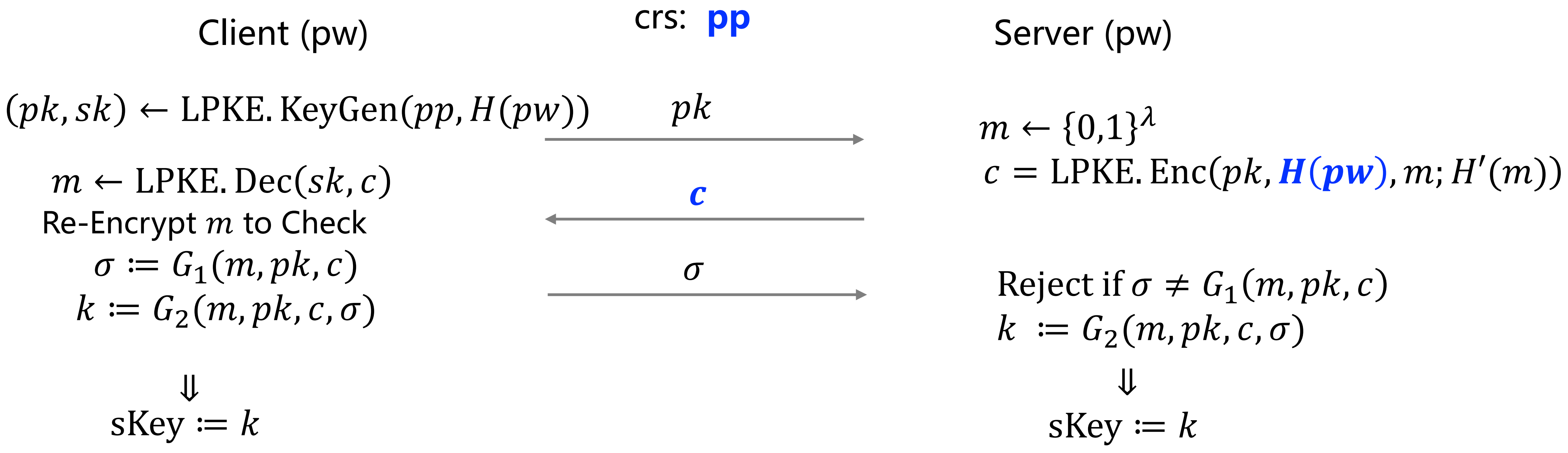
To simulate the second message c upon Sim receiving pk from the adversary $\mathcal{A}$,

Required LPKE property: For every pk, there is at most one pw s.t. $H(pw)$ is the label of pk

Extract the password embedded in pk with trapdoor td:

Sim can search all RO query $H(pw)$ to find LPKE.Check(td, pk, H(pw)) = 1

Simulation for the Second Message



Client (pw)

crs: **pp**

Server (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$\xrightarrow{pk}$

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, \boldsymbol{H(pw)}, m; H'(m))$

$\xleftarrow{\boldsymbol{c}}$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\sigma := G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\xrightarrow{\sigma}$

Reject if $\sigma \neq G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

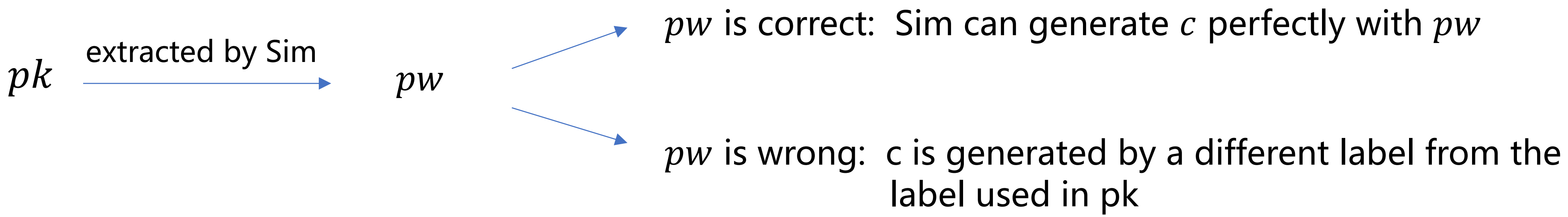
$\Downarrow$

$\mathrm{sKey} := k$

$\Downarrow$

$\mathrm{sKey} := k$

To simulate the second message c upon Sim receiving pk from the adversary $\mathcal{A}$,

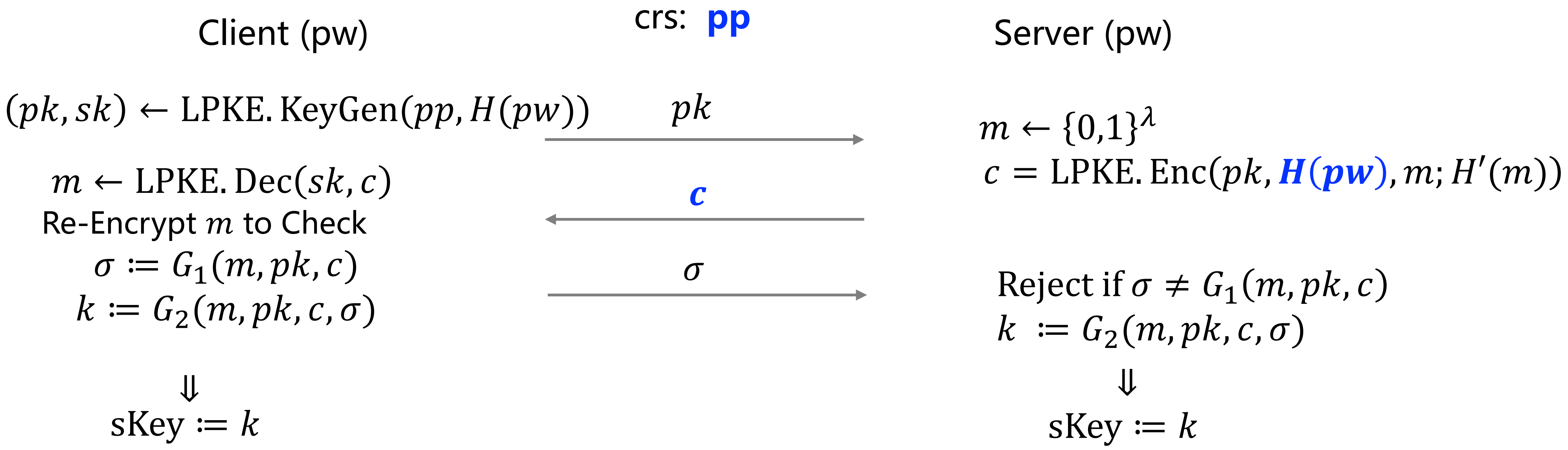


$pk \xrightarrow{\text{extracted by Sim}} pw$

pw is correct: Sim can generate c perfectly with pw

pw is wrong: c is generated by a different label from the label used in pk

Simulation for the Second Message



Client (pw)

crs: **pp**

Server (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$\xrightarrow{pk}$

$m \leftarrow \{0,1\}^\lambda$

$c = \mathrm{LPKE.Enc}(pk, \boldsymbol{H(pw)}, m; H'(m))$

$\xleftarrow{\boldsymbol{c}}$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\sigma := G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\xrightarrow{\sigma}$

Reject if $\sigma \neq G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

$\Downarrow$

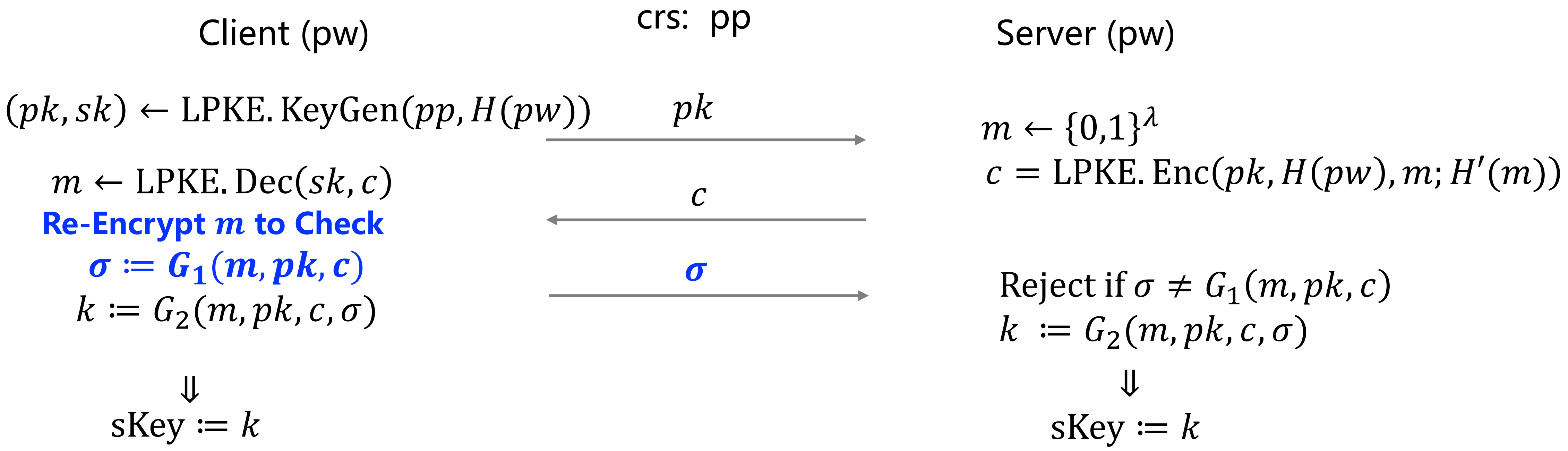
$\mathrm{sKey} := k$

To simulate the second message c upon Sim receiving pk from the adversary $\mathcal{A}$, pw is wrong:

Required LPKE property: For every pk, b satisfied LPKE.Check(td, pk, b) = 0, for every message m, it holds that $\{\mathrm{LPKE.Enc}(pk, b, m)\} \approx_s \{c : c \leftarrow_\$ \mathcal{CT}\}$

Simulation of c: Sim can simulate c by $c \leftarrow_\$ \mathcal{CT}$

Simulation for the Third Message



crs: pp

Client (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\boldsymbol{\sigma \coloneqq G_1(m, pk, c)}$

$k \coloneqq G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} \coloneqq k$

pk →

← c

$\boldsymbol{\sigma}$ →

Server (pw)

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

Reject if $\sigma \neq G_1(m, pk, c)$

$k \coloneqq G_2(m, pk, c, \sigma)$

$\Downarrow$

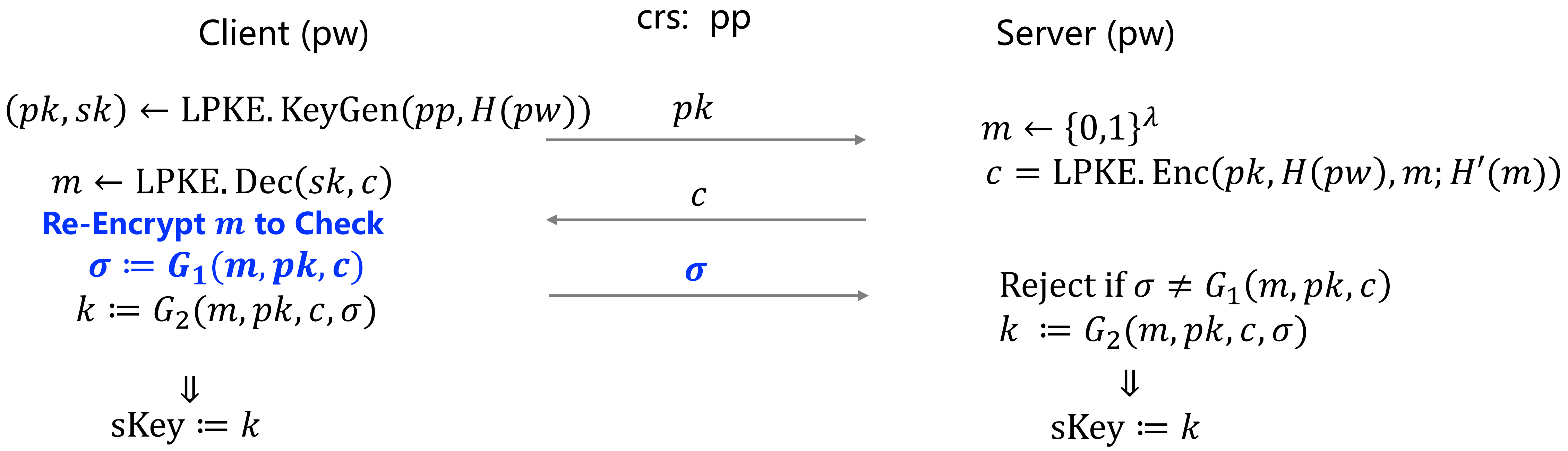
$\mathrm{sKey} \coloneqq k$

To simulate the third message σ upon Sim receiving c from the adversary $\mathcal{A}$,

Required LPKE property: CPA security and weak spreadness (for security of FO-transformation)

Simulation for σ: Note that Sim does not have sk corresponding to pk.
Here Sim uses a technique similar to FO-transformation to extract m from c

Simulation for the Third Message



crs: pp

Client (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

$\boldsymbol{\sigma := G_1(m, pk, c)}$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

$\xrightarrow{pk}$

$\xleftarrow{c}$

$\xrightarrow{\boldsymbol{\sigma}}$

Server (pw)

$m \leftarrow \{0,1\}^\lambda$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

Reject if $\sigma \neq G_1(m, pk, c)$

$k := G_2(m, pk, c, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

To simulate the third message σ upon Sim receiving c from the adversary $\mathcal{A}$,

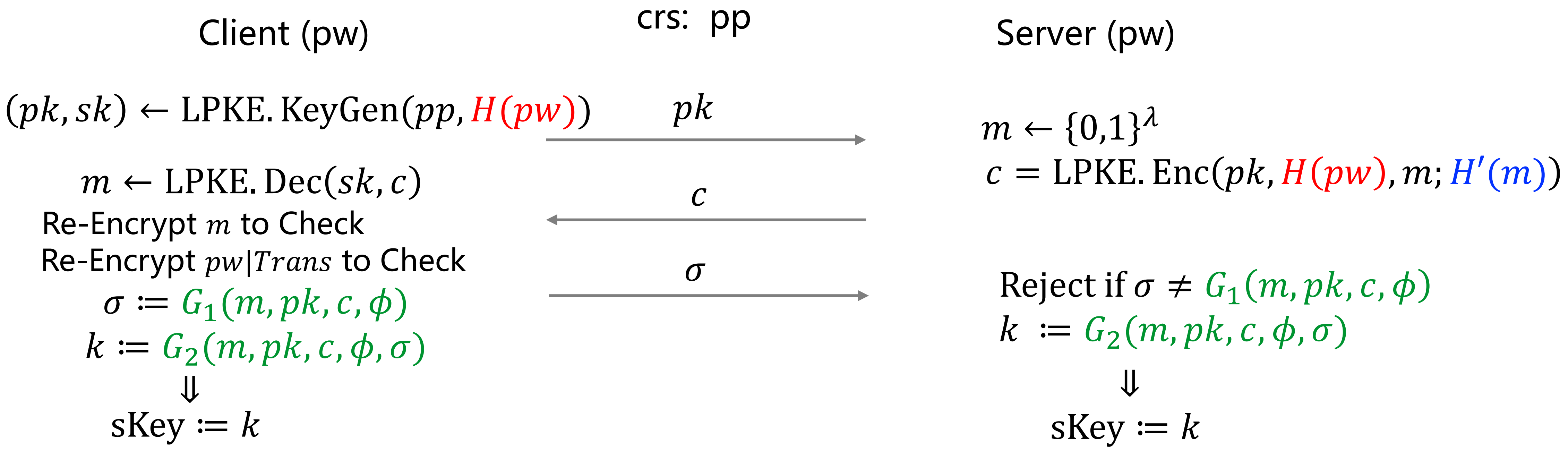
Required LPKE property: CPA security and weak spreadness (for security of FO-transformation)

Simulation for σ: Search RO queries $H(pw)$ and $H'(m)$ s.t. $c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

If exists such pw and m, then Sim can generate σ perfectly with m and pw

Otherwise, Sim can reject c by setting $sKey := \perp$

Upgrade Our Construction from ROM to QROM



crs: pp

Client (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$\xrightarrow{pk}$

Server (pw)

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

$\xleftarrow{c}$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

Re-Encrypt $pw|Trans$ to Check

$\sigma := G_1(m, pk, c, \phi)$

$k := G_2(m, pk, c, \phi, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

$\xrightarrow{\sigma}$

Reject if $\sigma \neq G_1(m, pk, c, \phi)$

$k := G_2(m, pk, c, \phi, \sigma)$

$\Downarrow$

$\mathrm{sKey} := k$

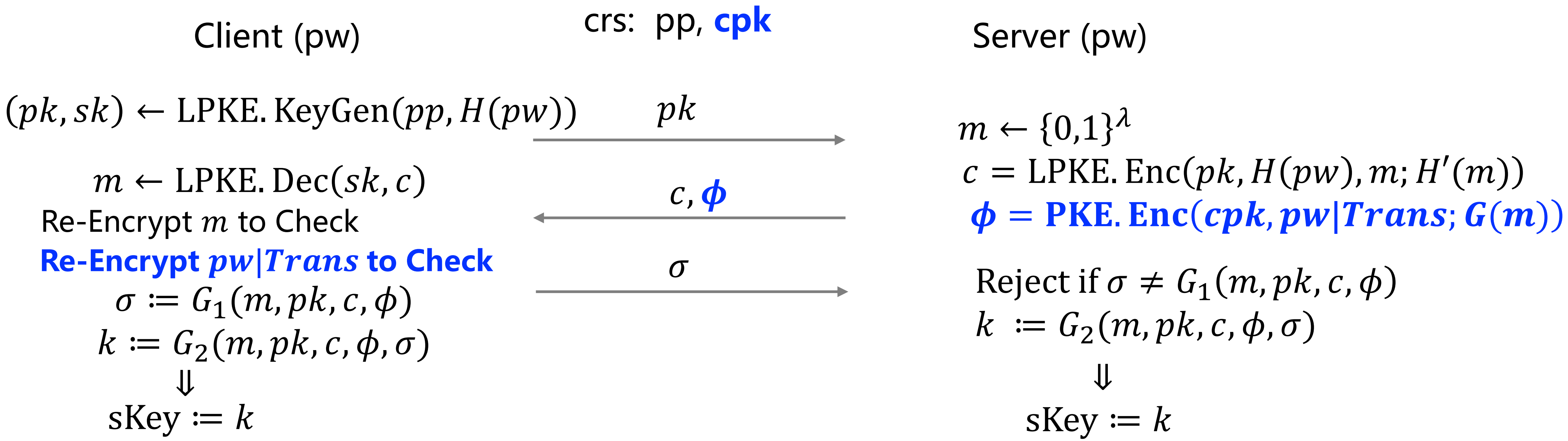
In our construction, the usage of RO can be divided into three functionalities.

1. The red RO is used to extract the password from public key pk and c

2. The blue RO is used to the FO-transformation and extract the message m from c

3. The green RO serves as pseudo-random functions.

Upgrade Our Construction from ROM to QROM



Client (pw)

crs: pp, **cpk**

Server (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$\xrightarrow{pk}$

$m \leftarrow \{0,1\}^\lambda$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

$\boldsymbol{\phi} = \mathbf{PKE.Enc}(\boldsymbol{cpk}, \boldsymbol{pw|Trans}; \boldsymbol{G(m)})$

$\xleftarrow{c, \boldsymbol{\phi}}$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

Re-Encrypt $pw|Trans$ to Check

$\sigma \coloneqq G_1(m, pk, c, \phi)$

$k \coloneqq G_2(m, pk, c, \phi, \sigma)$

$\Downarrow$

$\mathrm{sKey} \coloneqq k$

$\xrightarrow{\sigma}$

Reject if $\sigma \neq G_1(m, pk, c, \phi)$

$k \coloneqq G_2(m, pk, c, \phi, \sigma)$

$\Downarrow$

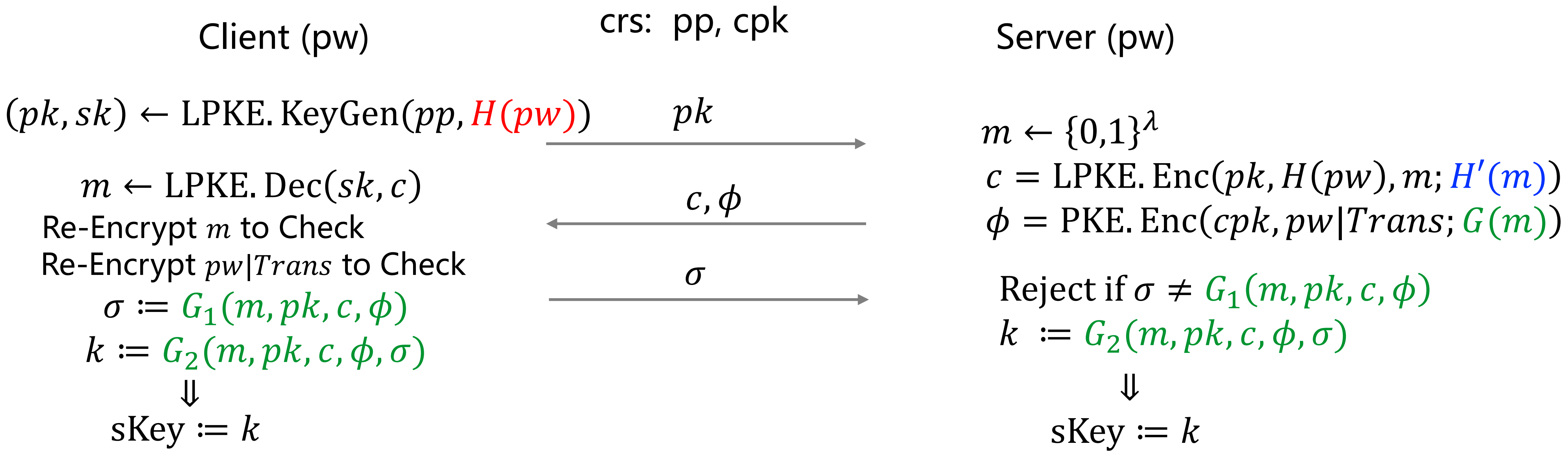
$\mathrm{sKey} \coloneqq k$

We first add a CCA-secure PKE encryption in the second message. The randomness is derived from m.

When Sim receives the second message c,ϕ, it can first decrypt ϕ to obtain pw, then extract message m through ciphertext c.

Now the extraction of m from c becomes a standard FO-transformation technique.

Upgrade Our Construction from ROM to QROM



Client (pw)

crs: pp, cpk

Server (pw)

$(pk, sk) \leftarrow \mathrm{LPKE.KeyGen}(pp, H(pw))$

$m \leftarrow \mathrm{LPKE.Dec}(sk, c)$

Re-Encrypt m to Check

Re-Encrypt $pw|Trans$ to Check

$\sigma \coloneqq G_1(m, pk, c, \phi)$

$k \coloneqq G_2(m, pk, c, \phi, \sigma)$

$\Downarrow$

$\mathrm{sKey} \coloneqq k$

$\xrightarrow{pk}$

$\xleftarrow{c, \phi}$

$\xrightarrow{\sigma}$

$m \leftarrow \{0,1\}^\lambda$

$c = \mathrm{LPKE.Enc}(pk, H(pw), m; H'(m))$

$\phi = \mathrm{PKE.Enc}(cpk, pw|Trans; G(m))$

$\text{Reject if } \sigma \neq G_1(m, pk, c, \phi)$

$k \coloneqq G_2(m, pk, c, \phi, \sigma)$

$\Downarrow$

$\mathrm{sKey} \coloneqq k$

1. The red RO is used to extract the password from public key pk
2. The blue RO is used to the FO-transformation and extract the message from c
3. The green RO serves as pseudo-random functions.

The blue RO can be adapted into QROM using online-extractable technique in [EC: DFMS21]
The green RO can be proven in QROM using the O2H Lemma [C: AHU19]
The red RO seems hard to adapt into QROM.

Upgrade Our Construction from ROM to QROM

crs: pp, cpk

Client (pw)

$(pk, sk) \leftarrow \mathrm{eLPKE.KeyGen}(pp, pw)$

$m \leftarrow \mathrm{eLPKE.Dec}(sk, c)$

Re-Encrypt m to Check

Re-Encrypt $pw|Trans$ to Check

$\sigma \coloneqq G_1(m, pk, c, \phi)$

$k \coloneqq G_2(m, pk, c, \phi, \sigma)$

$\Downarrow$

$\mathrm{sKey} \coloneqq k$

$\xrightarrow{\quad pk \quad}$

$\xleftarrow{\quad c, \phi \quad}$

$\xrightarrow{\quad \sigma \quad}$

Server (pw)

$m \leftarrow \{0,1\}^{\lambda}$

$c = \mathrm{eLPKE.Enc}(pk, pw, m; H'(m))$

$\phi = \mathrm{PKE.Enc}(cpk, pw|Trans; G(m))$

Reject if $\sigma \neq G_1(m, pk, c, \phi)$

$k \coloneqq G_2(m, pk, c, \phi, \sigma)$

$\Downarrow$

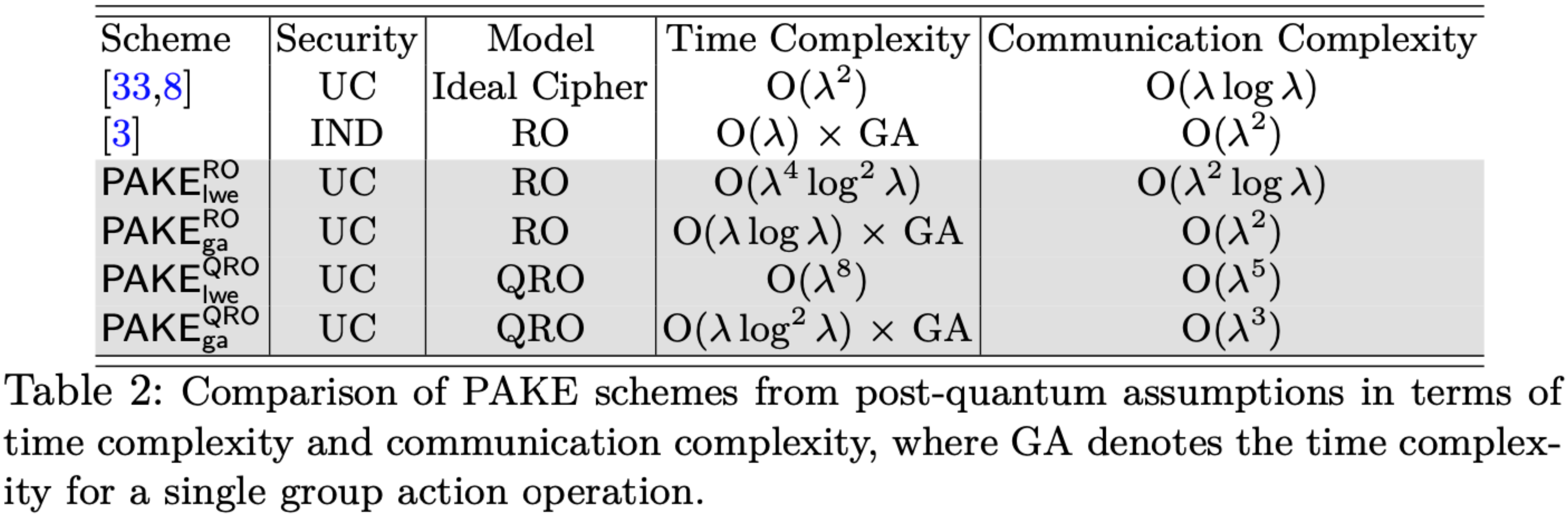
$\mathrm{sKey} \coloneqq k$

High Level Idea:

1. Remove the usage of RO
2. Enhance the underlying LPKE such that it can extract pw only with trapdoor td (and without the help of RO)

Contents

Efficiency Comparison



Scheme	Security	Model	Time Complexity	Communication Complexity
[33,8]	UC	Ideal Cipher	$O(\lambda^2)$	$O(\lambda \log \lambda)$
[3]	IND	RO	$O(\lambda) \times GA$	$O(\lambda^2)$
$\text{PAKE}_{\text{lwe}}^{\text{RO}}$	UC	RO	$O(\lambda^4 \log^2 \lambda)$	$O(\lambda^2 \log \lambda)$
$\text{PAKE}_{\text{ga}}^{\text{RO}}$	UC	RO	$O(\lambda \log \lambda) \times GA$	$O(\lambda^2)$
$\text{PAKE}_{\text{lwe}}^{\text{QRO}}$	UC	QRO	$O(\lambda^8)$	$O(\lambda^5)$
$\text{PAKE}_{\text{ga}}^{\text{QRO}}$	UC	QRO	$O(\lambda \log^2 \lambda) \times GA$	$O(\lambda^3)$

Table 2: Comparison of PAKE schemes from post-quantum assumptions in terms of time complexity and communication complexity, where GA denotes the time complexity for a single group action operation.

Our LWE-based PAKE in QROM might not be practical, but we are the first to achieve this.

Our other PAKE protocols are practical.

Conclusion

- In this paper, we propose generic constructions for UC-secure PAKE from LPKE in ROM/QROM.
- These constructions admit four specific PAKE schemes with UC security in ROM or QROM, based on LWE or GA-DDH.
- For more information, please refer to the full version our paper.

  https://eprint.iacr.org/2024/374.pdf

Thanks!   Questions?

Instantiation LPKE from LWE Assumption

Adopt from the Regev encryption scheme

Setup: $(\mathbf{A}, \mathbf{T}) \leftarrow \mathrm{TrapGen}$. With trapdoor $\mathbf{T}$, one can solve the LWE problem.

LPKE.KeyGen($H(pw)$): $sk \coloneqq s, \quad pk \coloneqq \mathbf{A}^T s + e - H(pw)$

LPKE.Enc($pk, m, H(pw)$): $p \coloneqq pk + H(pw)$. $c_1 \coloneqq \mathbf{A}r,\ c_2 \coloneqq < p, r > + m \times \frac{q}{2}$

LPKE.Dec($sk = s, c = (c_1, c_2)$): $m \coloneqq \lceil c_2 - < s, c_1 > \rfloor_q$

LPKE.Test($\mathbf{T}, pk, pw$): solve $pk - H(pw) = \mathbf{A}^T s + e$ and check whether e is small enough

If pk and c contain different labels, then c is encrypted by a uniform p and thus uniform by the leftover hash lemma.

It is also unlikely for two random vector v1, v2 s.t. v1 – v2 is close to the lattice $\Lambda(\mathbf{A}^T)$.
(Uniqueness of labels contained in pk)

Construction of extractable LPKE

We construct the extractable LPKE from a bit-by-bit approach. Suppose pw has λ bits.

Setup will generate a crs (along with its trapdoor) and 2λ uniform strings $\{v_1^0, v_1^1, \ldots, v_\lambda^0, v_\lambda^1\}$

eLPKE.KeyGen generates λ public keys, the i-th public key is generated by label $v_i^{pw_i}$

eLPKE.Enc chooses random $z_1, z_2, \ldots, z_\lambda$ s.t. $m = z_1 \oplus z_2 \oplus \cdots \oplus z_\lambda$. Then encrypt z_i using pk_i with label $v_i^{pw_i}$

Now Sim can extract pw from $\overrightarrow{pk} = pk_1, \ldots, pk_\lambda$ via a bit-by-bit approach.

When $\overrightarrow{pk}$ and $\vec{c}$ use different labels, at least one z_i in c_i becomes uniform and thus the whole m is uniform.

But now the ciphertext $\vec{c}$ leaks too much information of pw.

So we additionally require Ciphertext Randomness in case of Random Messages for underlying LPKE. (and this is why our PAKE in QROM requires super-polynomial modulus q)