A Post-Quantum Round-Optimal Oblivious PRF from Isogenies

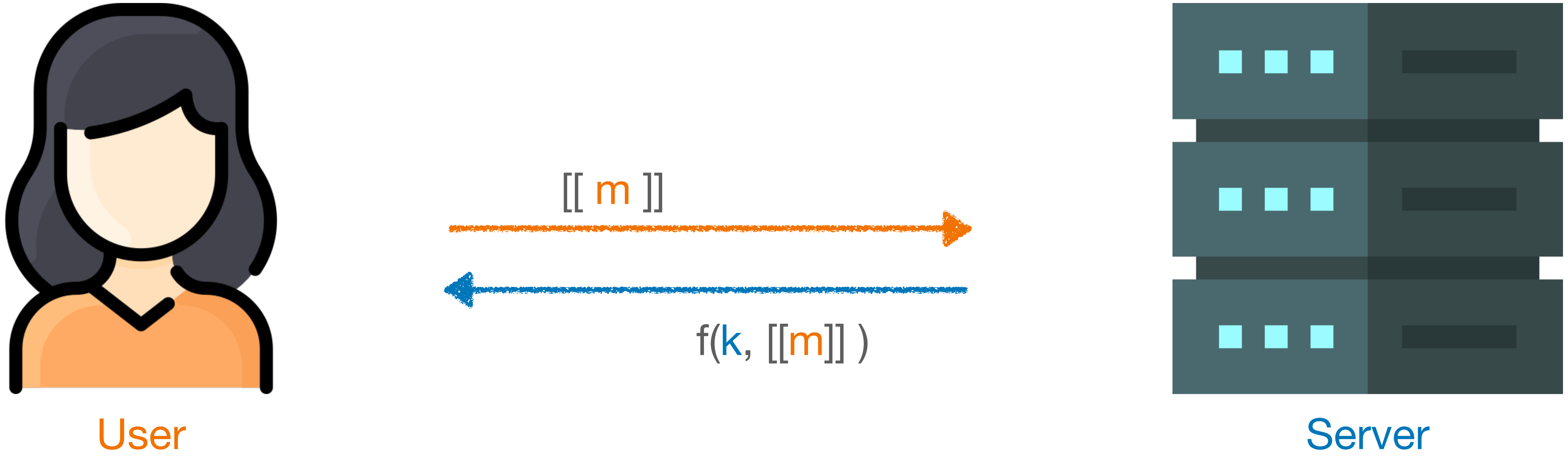
Andrea Basso

16th August, 2023
Selected Areas in Cryptography 2023

UNIVERSITY OF BIRMINGHAM

University of BRISTOL

Oblivious PRF



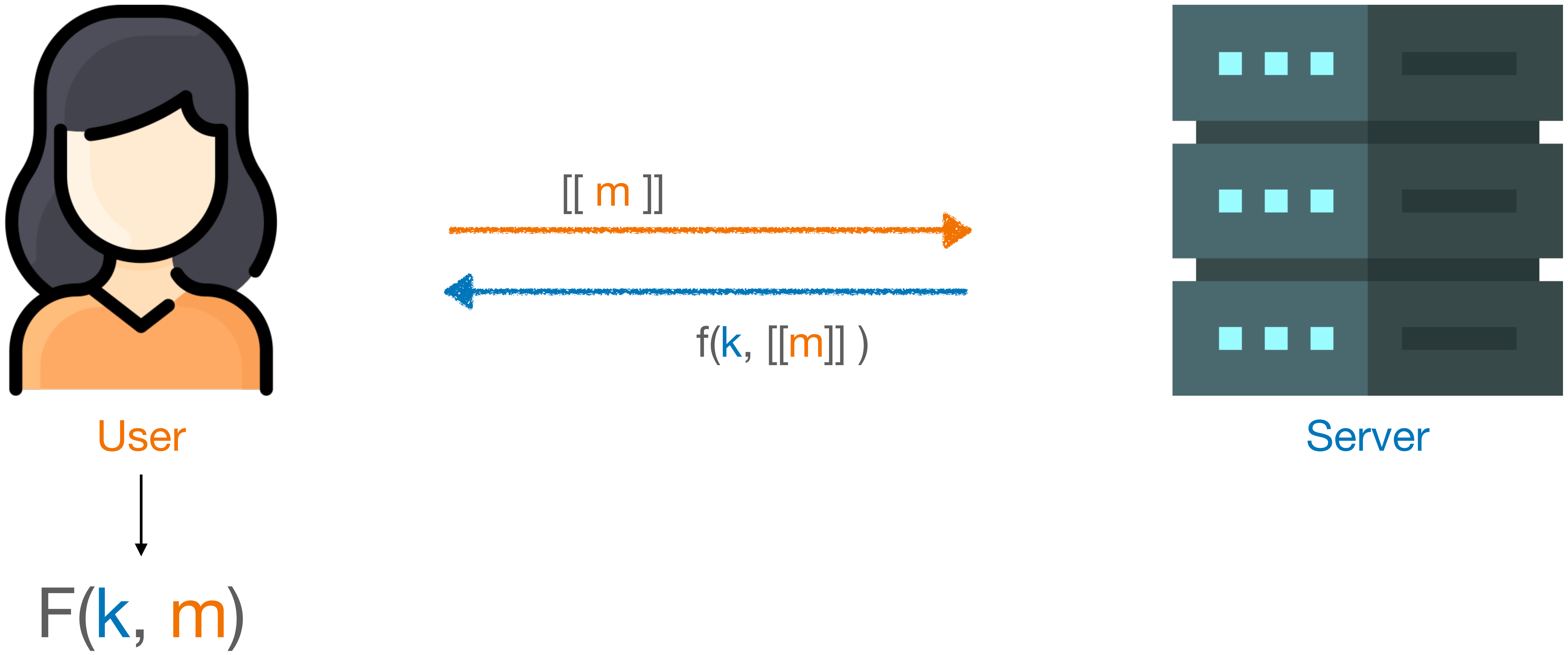
[[m]]

f(k, [[m]])

User

Server

Oblivious PRF



[[m]]

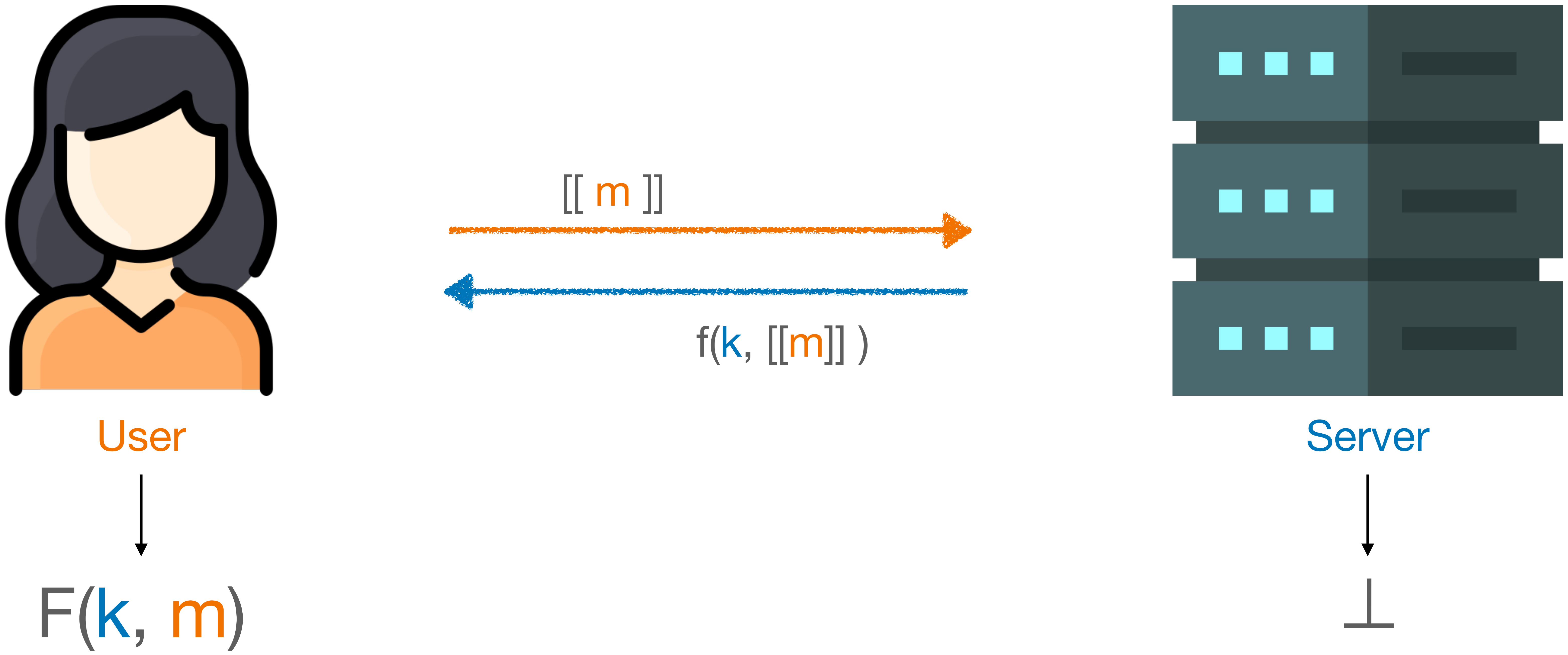
f(k, [[m]])

User

Server

F(k, m)

Oblivious PRF



User

[[m]]

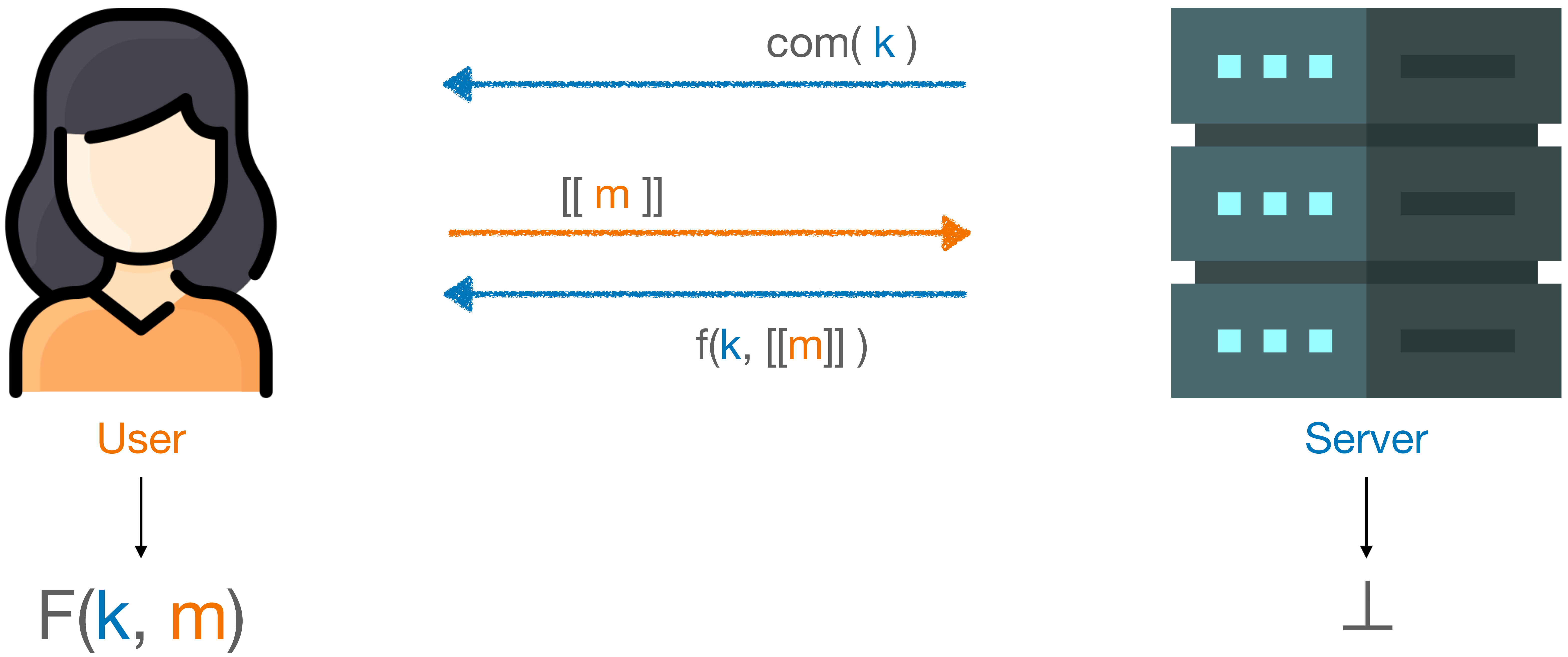
f(k, [[m]])

Server

$F(k, m)$

$\perp$

Oblivious PRF



com(k)

[[m]]

f(k, [[m]])

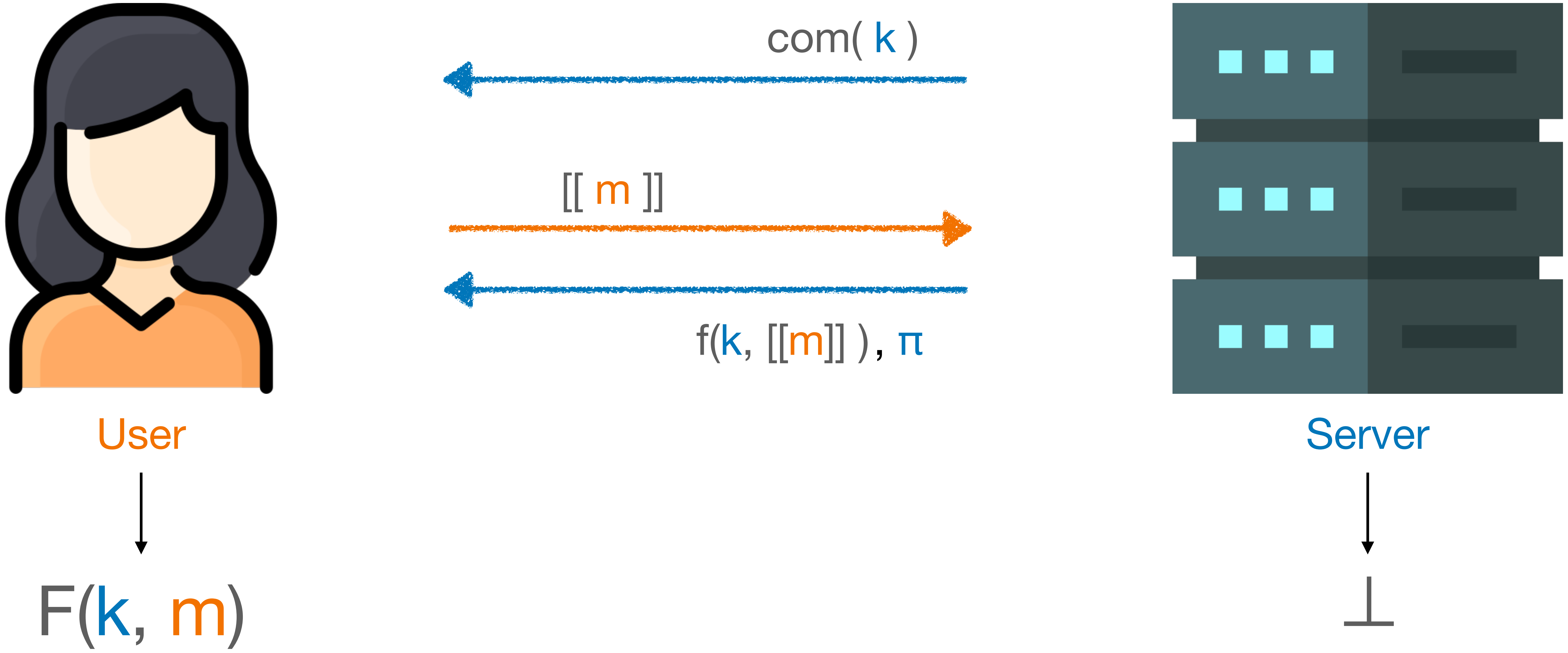
User

Server

F(k, m)

⊥

Oblivious PRF



com(k)

[[m]]

f(k, [[m]]) , π

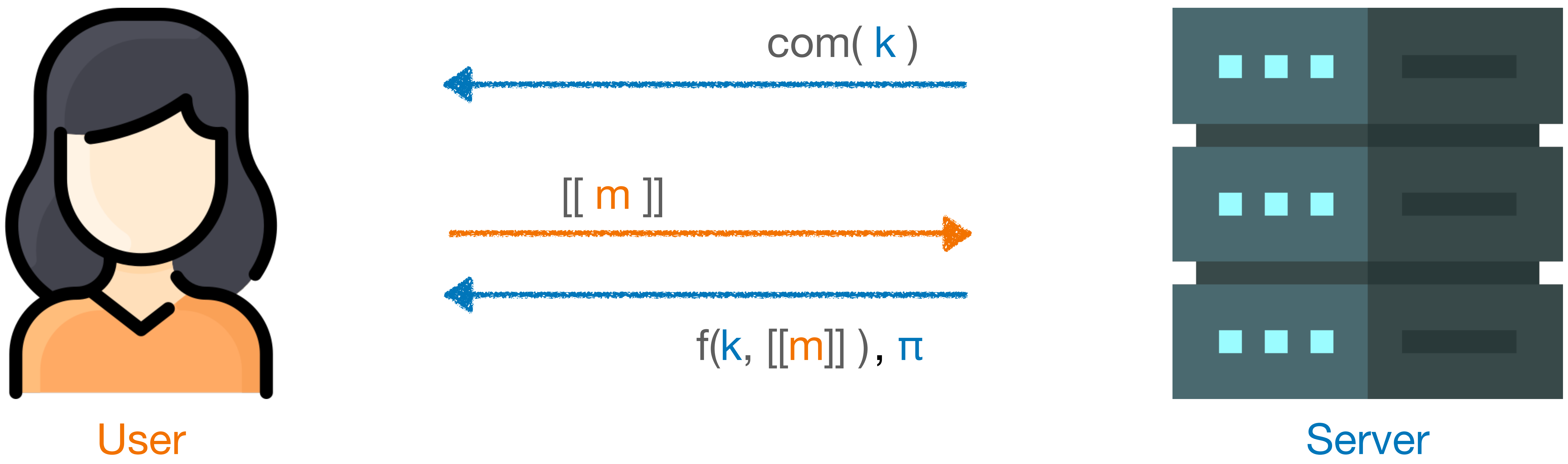
User

Server

F(k, m)

⊥

Oblivious PRF



com(k)

[[m]]

f(k, [[m]]), π

User

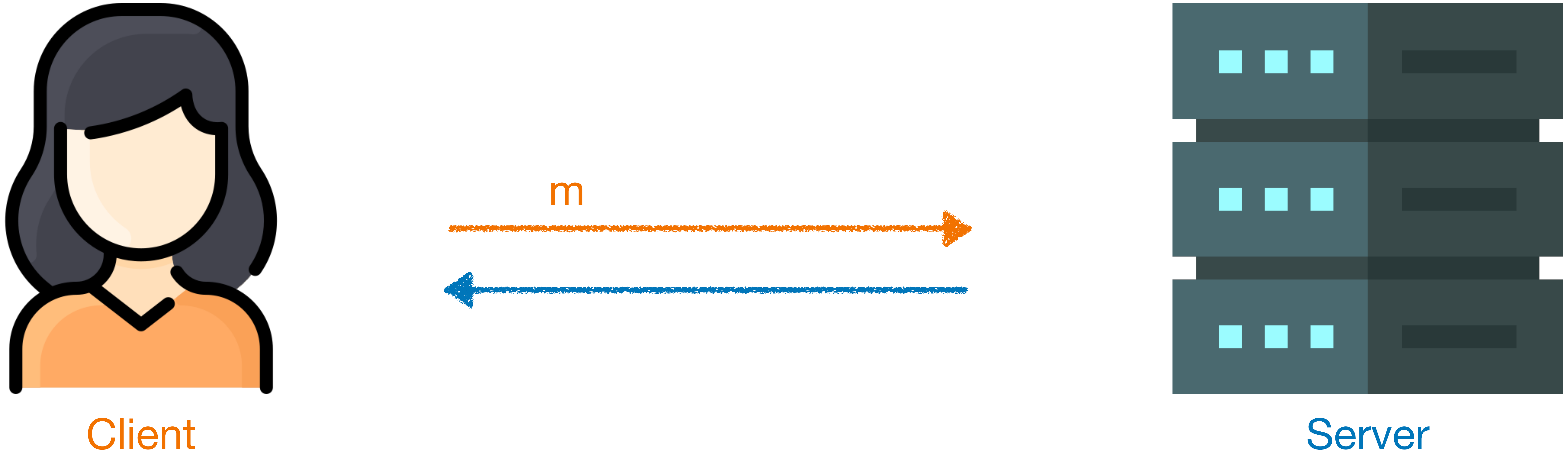
Server

$F(k, m)$

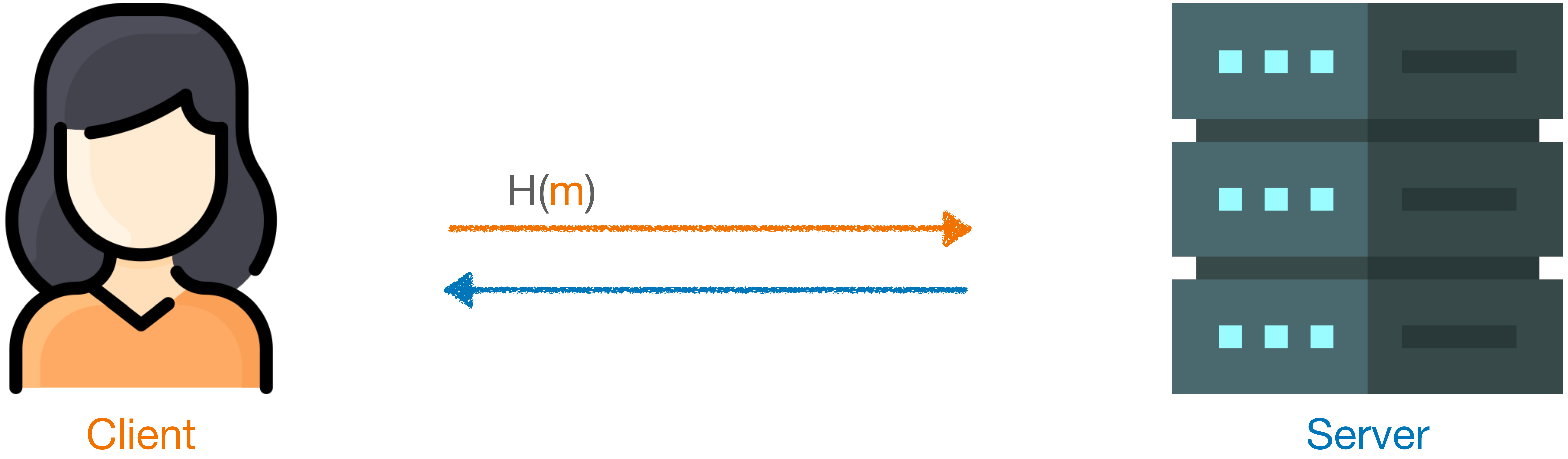
$\perp$

- Password-checking in Microsoft Edge
- OPAQUE
- Privacy pass
- Private-set intersection
- Adaptive OT
- ....

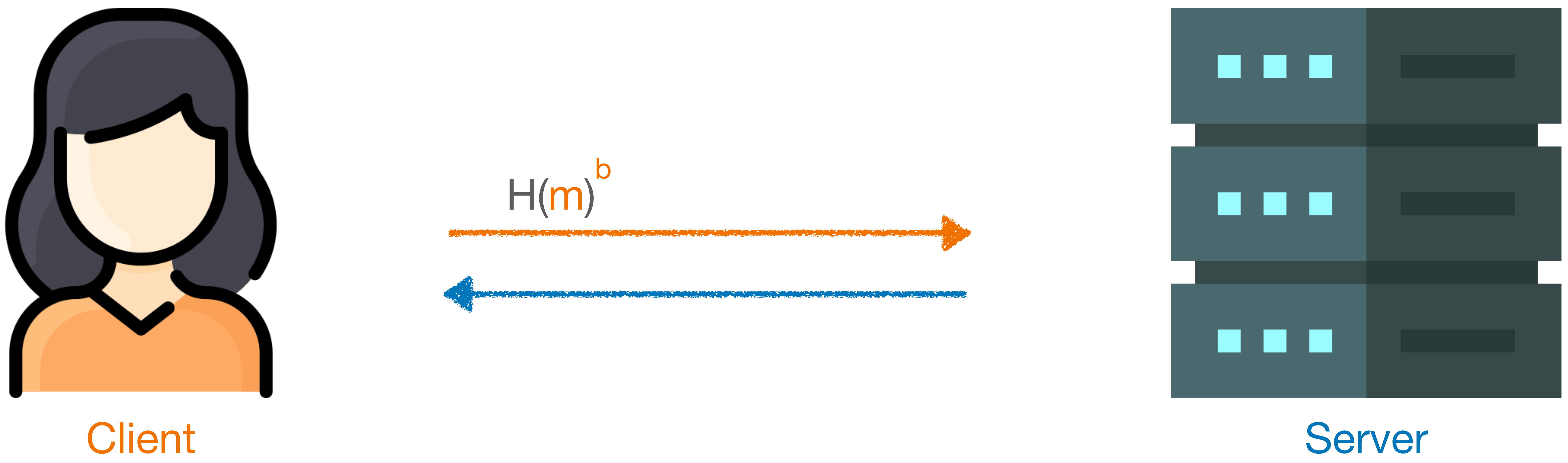
HashDH OPRF



HashDH OPRF



HashDH OPRF

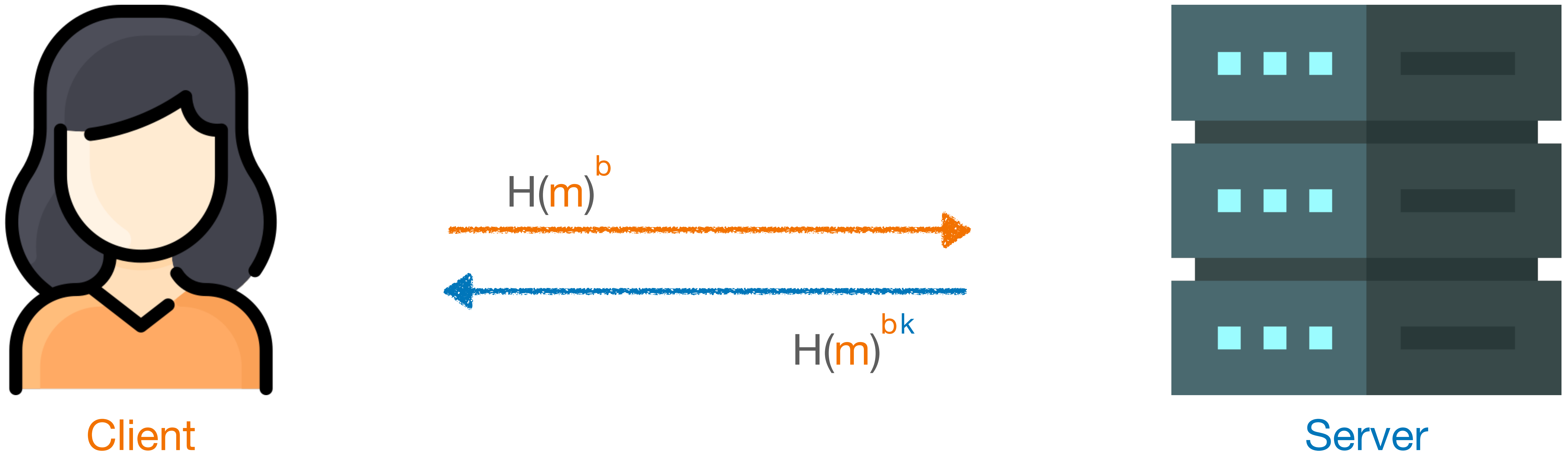


$H(m)^b$

Client

Server

HashDH OPRF



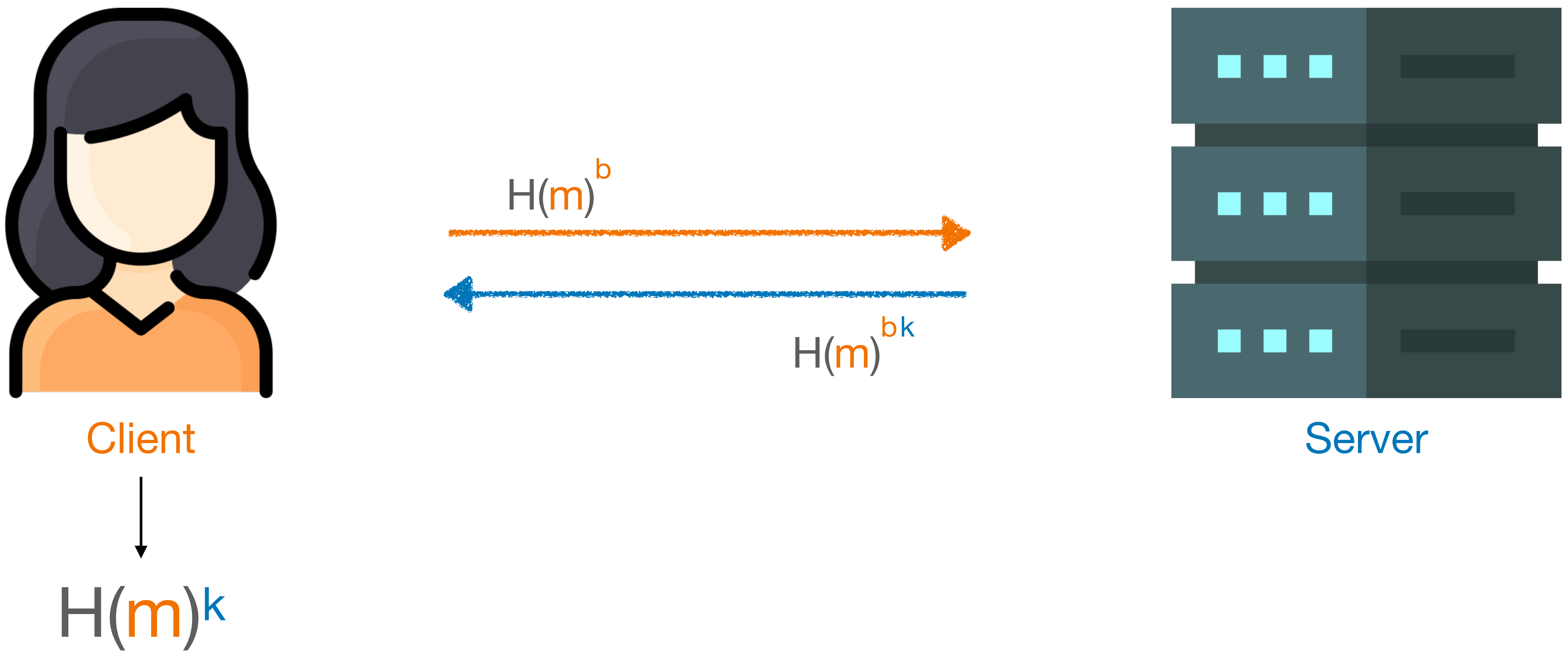
$H(m)^{b}$

$H(m)^{bk}$

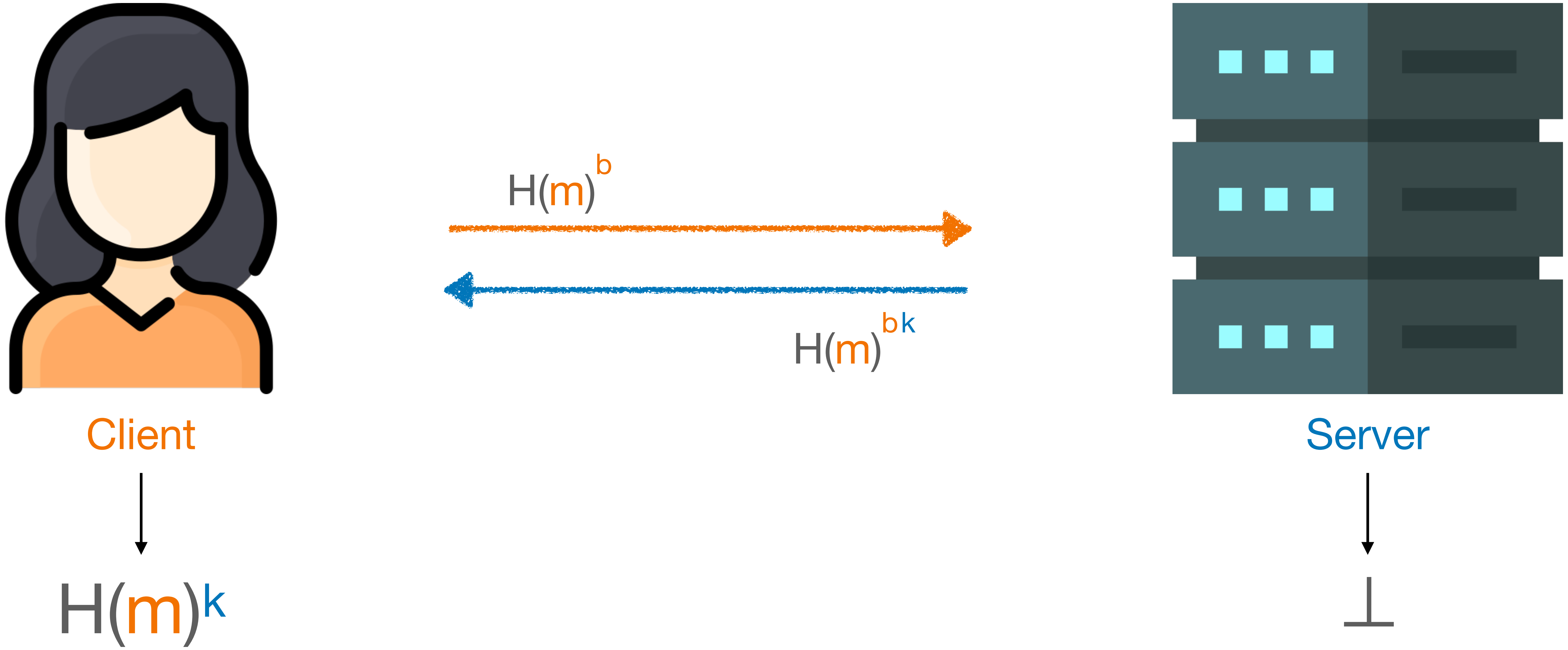
Client

Server

HashDH OPRF



HashDH OPRF



$H(m)^{b}$

$H(m)^{bk}$

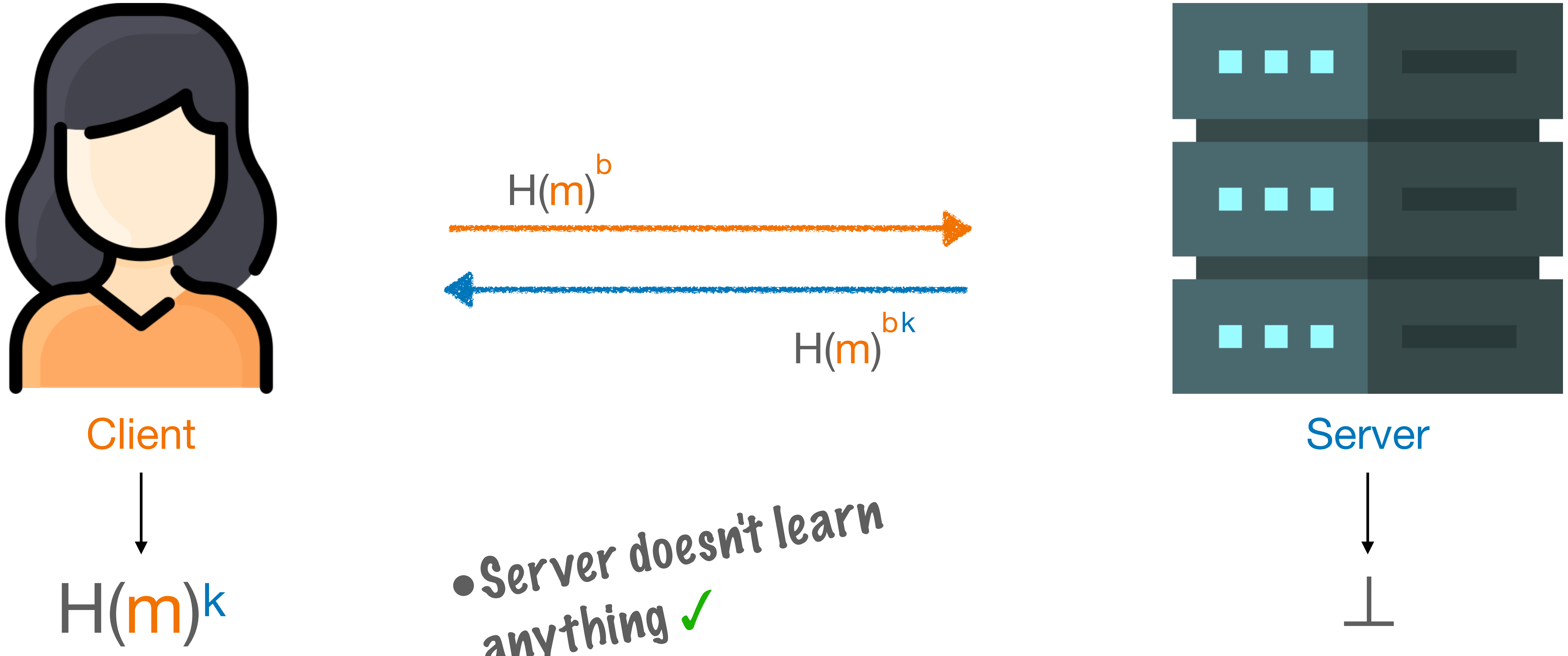
Client

Server

$H(m)^{k}$

$\perp$

HashDH OPRF



$H(m)^b$

$H(m)^{bk}$

Client

$H(m)^k$

Server

$\perp$

- Server doesn't learn anything ✓
- Output is deterministic ✓
- Client only learns one output ✓

Post-quantum OPRFs

- Generic MPC techniques →
  - many rounds (can’t be optimal)
- VOPRF based on lattices [ADDS19] →
  - round optimal
  - feasibility result (> 2^{40} bits of comms)
- VOPRF based on SIDH [BKW20] →
  - six rounds
  - broken by attack on PR and on SIDH
- OPRF based on CSIDH [BKW20] →
  - three rounds (OT required)
  - CSIDH parameters?

Post-quantum OPRFs

- Generic MPC techniques →
  - many rounds (can’t be optimal)

- VOPRF based on lattices [ADDS19] →
  - round optimal
  - feasibility result (> 2^{40} bits of comms)

- VOPRF based on SIDH [BKW20] →
  - six rounds
  - broken by attack on PR and on SIDH

- OPRF based on CSIDH [BKW20] →
  - three rounds (OT required)
  - CSIDH parameters?

The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[d, "k"'] & E_m \arrow[d] \\
E_k \arrow[r] & E_{mk}
\end{tikzcd}$$

The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[d, "k"'] & E_m \arrow[d] \\
E_k \arrow[r] & E_{mk}
\end{tikzcd}$$

The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[d, "k"'] & E_m \arrow[d] \\
E_k \arrow[r] & E_{mk}
\end{tikzcd}$$

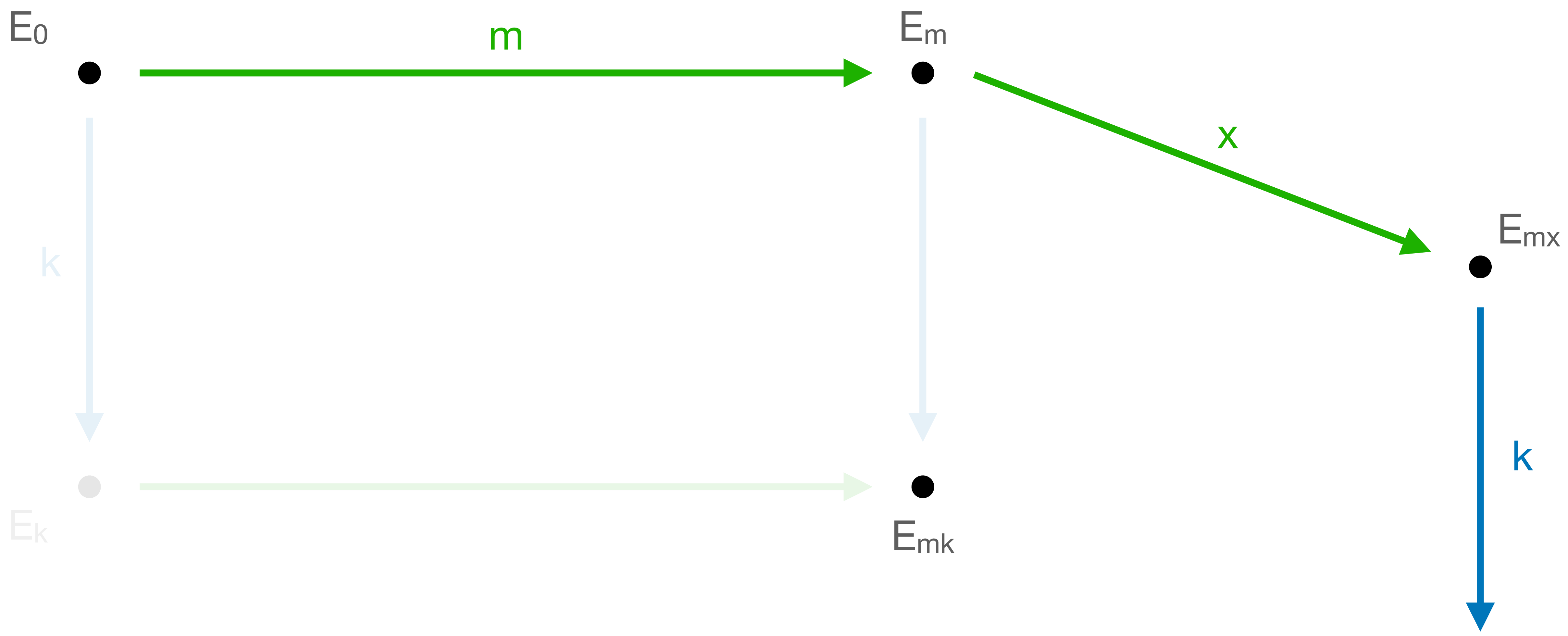
The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[d, "k"'] & E_m \arrow[d] \arrow[dr, "x"] & \\
E_k \arrow[r] & E_{mk} &
\end{tikzcd}$$

The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[d, "k"'] & E_m \arrow[r, "x"] \arrow[d] & E_{mx} \\
E_k \arrow[r] & E_{mk} &
\end{tikzcd}$$

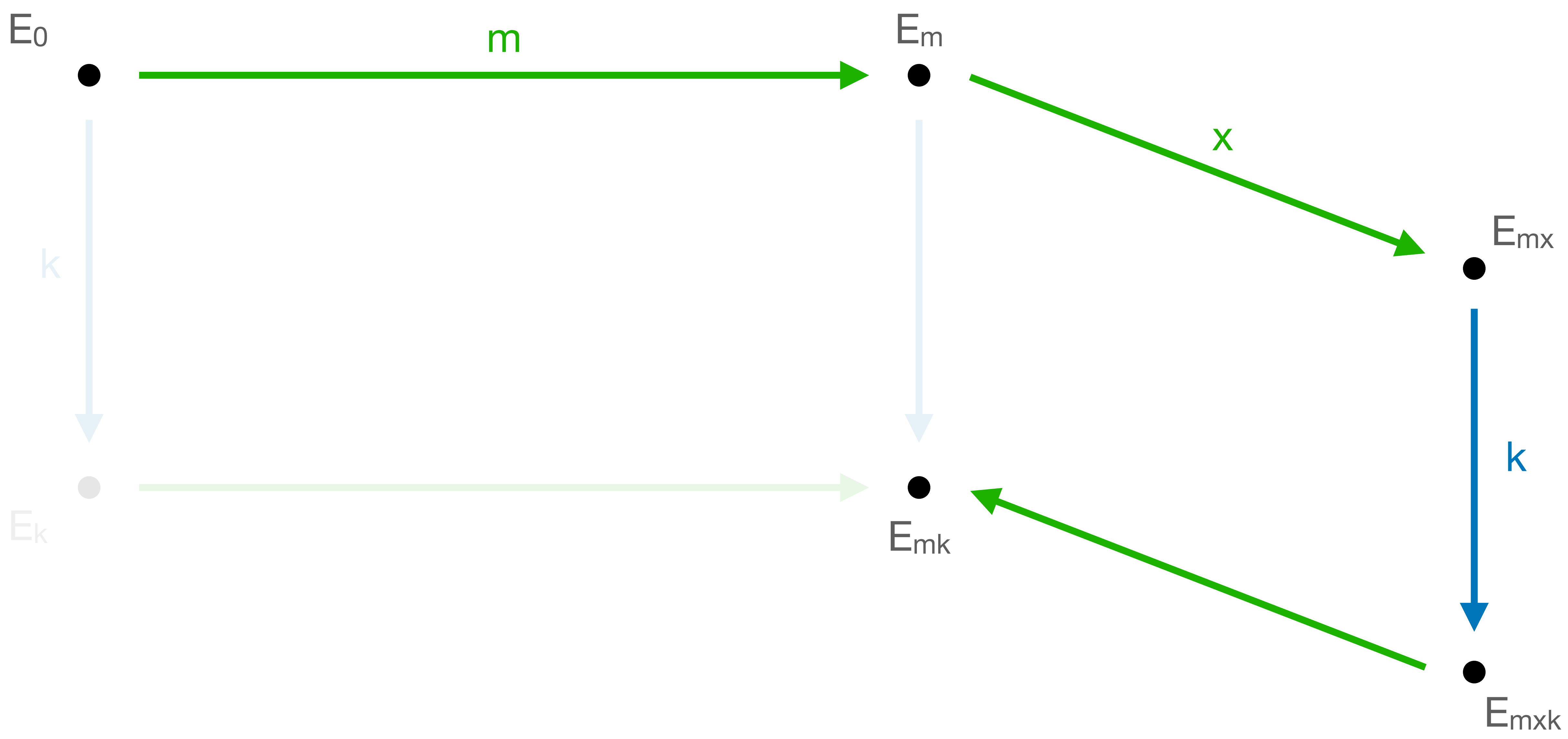
The original OPRF [BKW20]



The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[dd, "k"'] & E_m \arrow[dr, "x"] \arrow[dd] & \\
& & E_{mx} \arrow[dd, "k"] \\
E_k \arrow[r] & E_{mk} & \\
& & E_{mxk}
\end{tikzcd}$$

The original OPRF [BKW20]



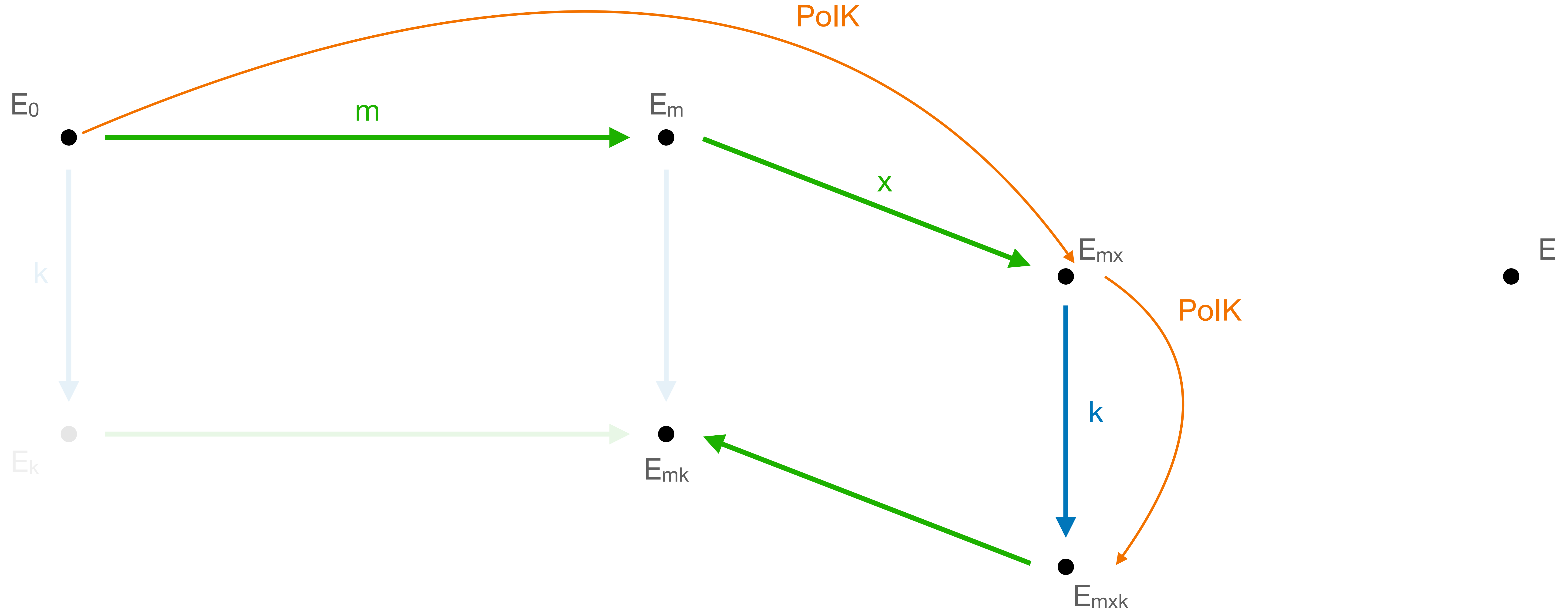
The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[rrd, bend left, "PolK"] \arrow[dd, "k"'] & E_m \arrow[rd, "x"] \arrow[dd] & \\
 & & E_{mx} \arrow[dd, "k"] \\
E_k \arrow[r] & E_{mk} & \\
 & & E_{mxk} \arrow[ul]
\end{tikzcd}$$

The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[dd, "k"'] \arrow[rrd, bend left, "PolK"] & E_m \arrow[rd, "x"] \arrow[dd] & \\
 & & E_{mx} \arrow[dd, "k"] \arrow[dd, bend left, "PolK"] \\
E_k \arrow[r] & E_{mk} & \\
 & & E_{mxk} \arrow[luu, phantom] \arrow[lu]
\end{tikzcd}$$

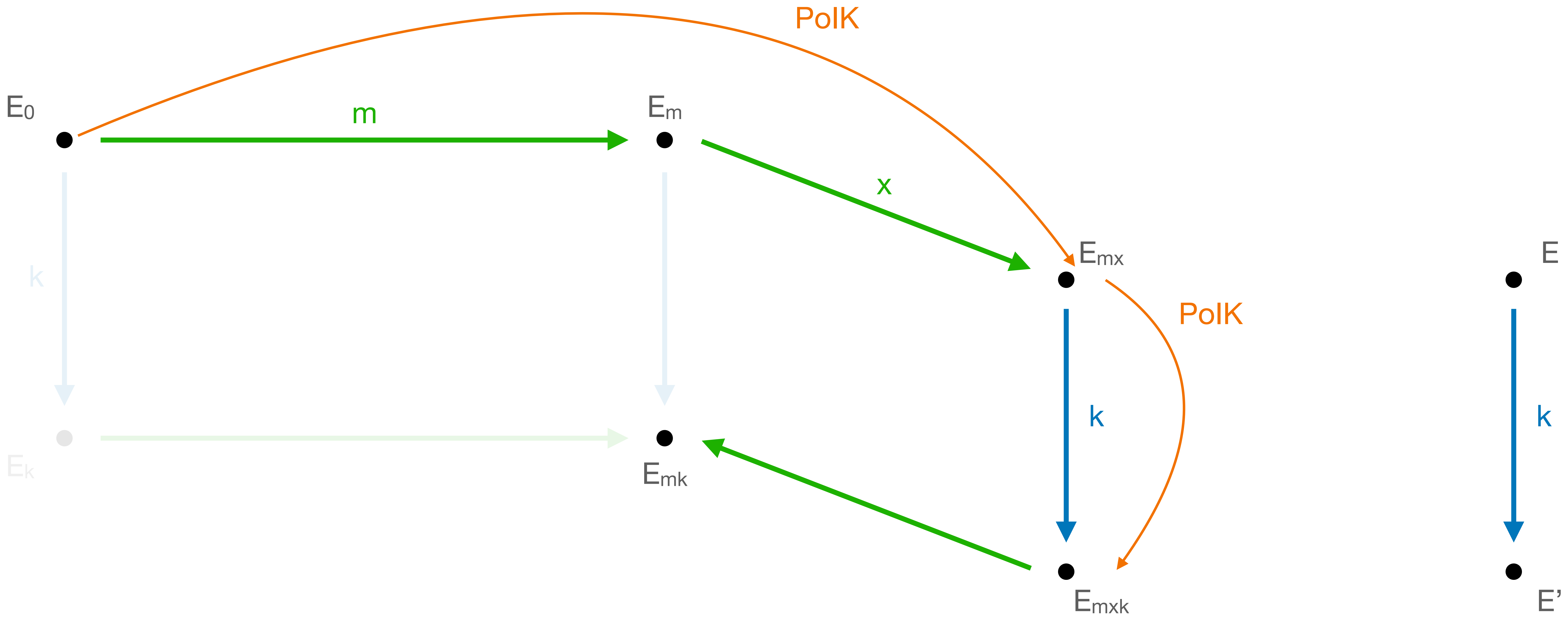
The original OPRF [BKW20]



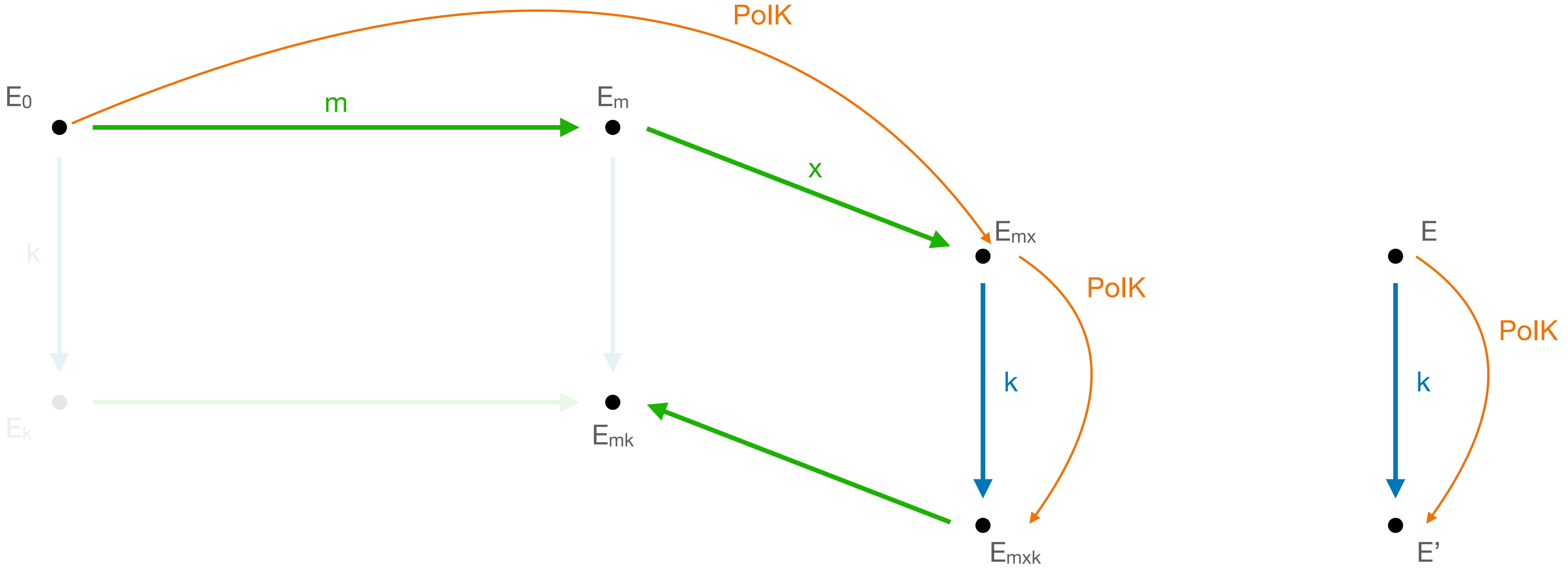
The original OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "m"] \arrow[rrd, bend left, "PolK"] \arrow[dd, "k"] & E_m \arrow[rd, "x"] \arrow[dd] & & \\
& & E_{mx} \arrow[dd, "k"] \arrow[dd, bend left, "PolK"] & E \arrow[dd, "k"] \\
E_k \arrow[r] & E_{mk} & & \\
& & E_{mxk} \arrow[lu] &
\end{tikzcd}$$

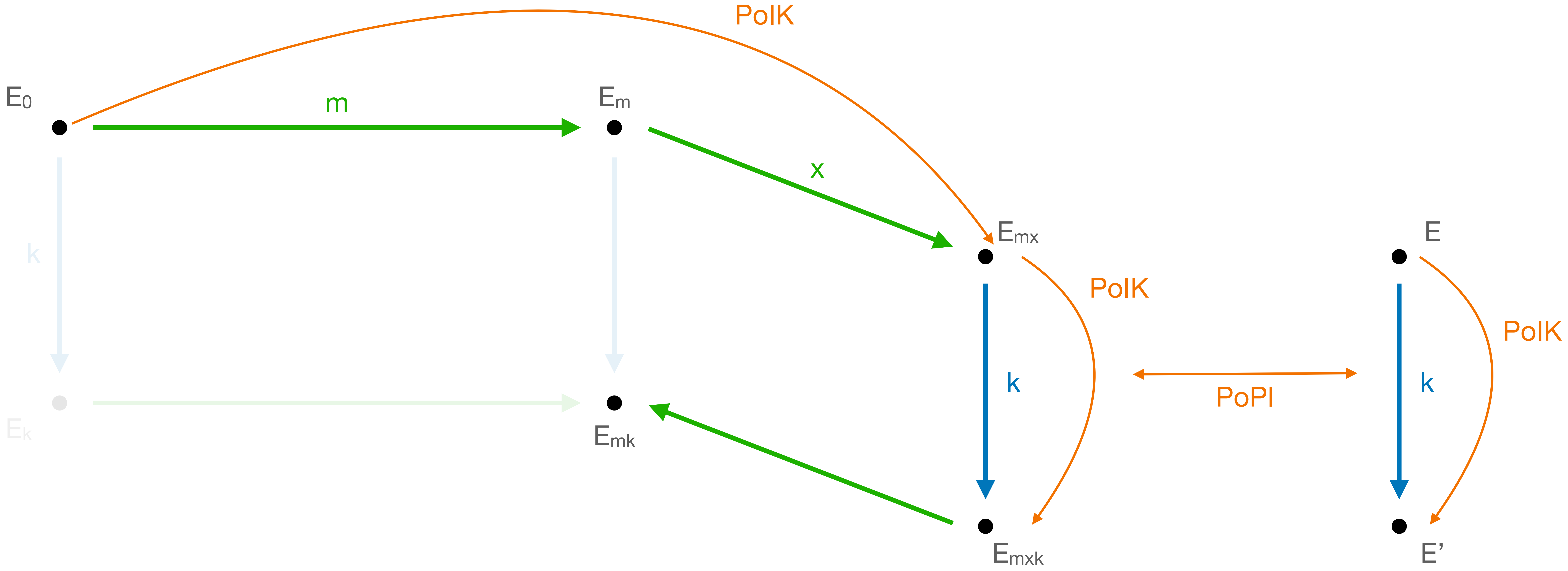
The original OPRF [BKW20]



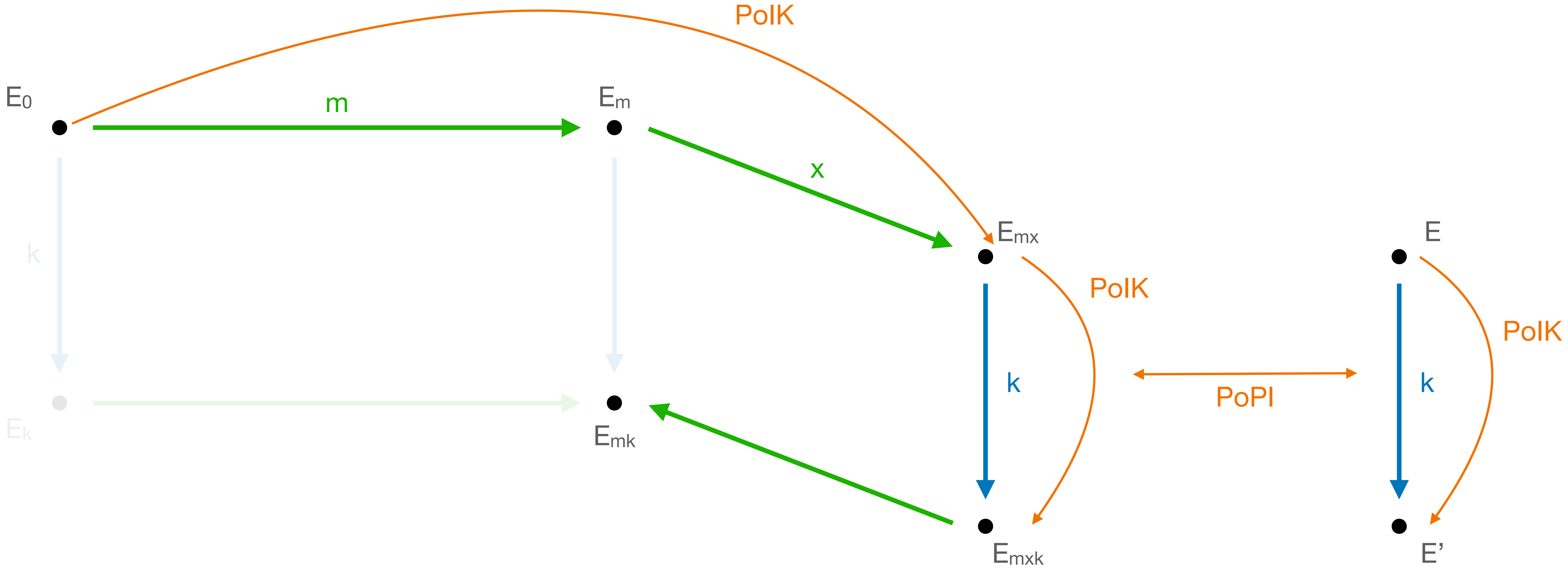
The original OPRF [BKW20]



The original OPRF [BKW20]



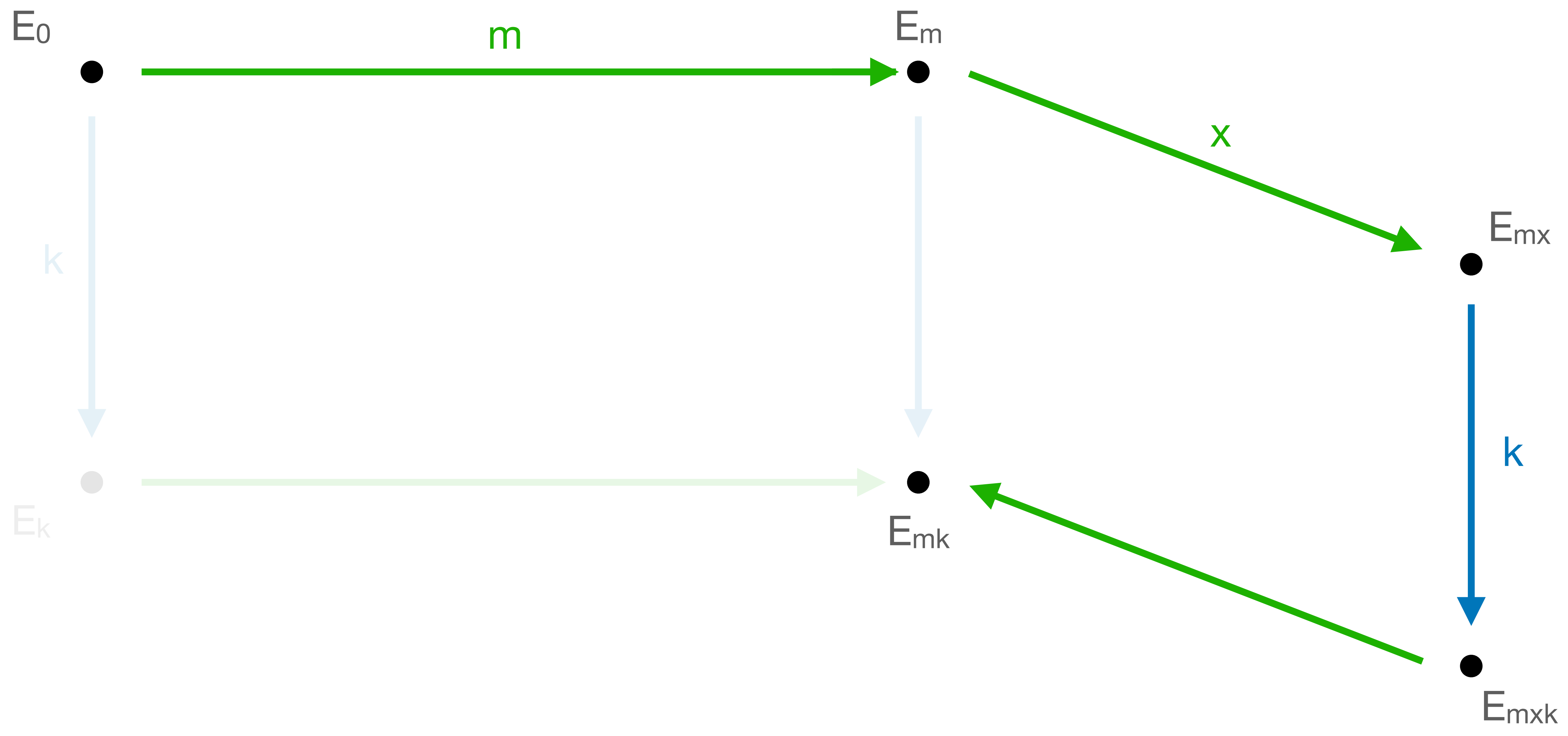
The original OPRF [BKW20]



$$F(k, m) = H(m, j_{mk}, E')$$

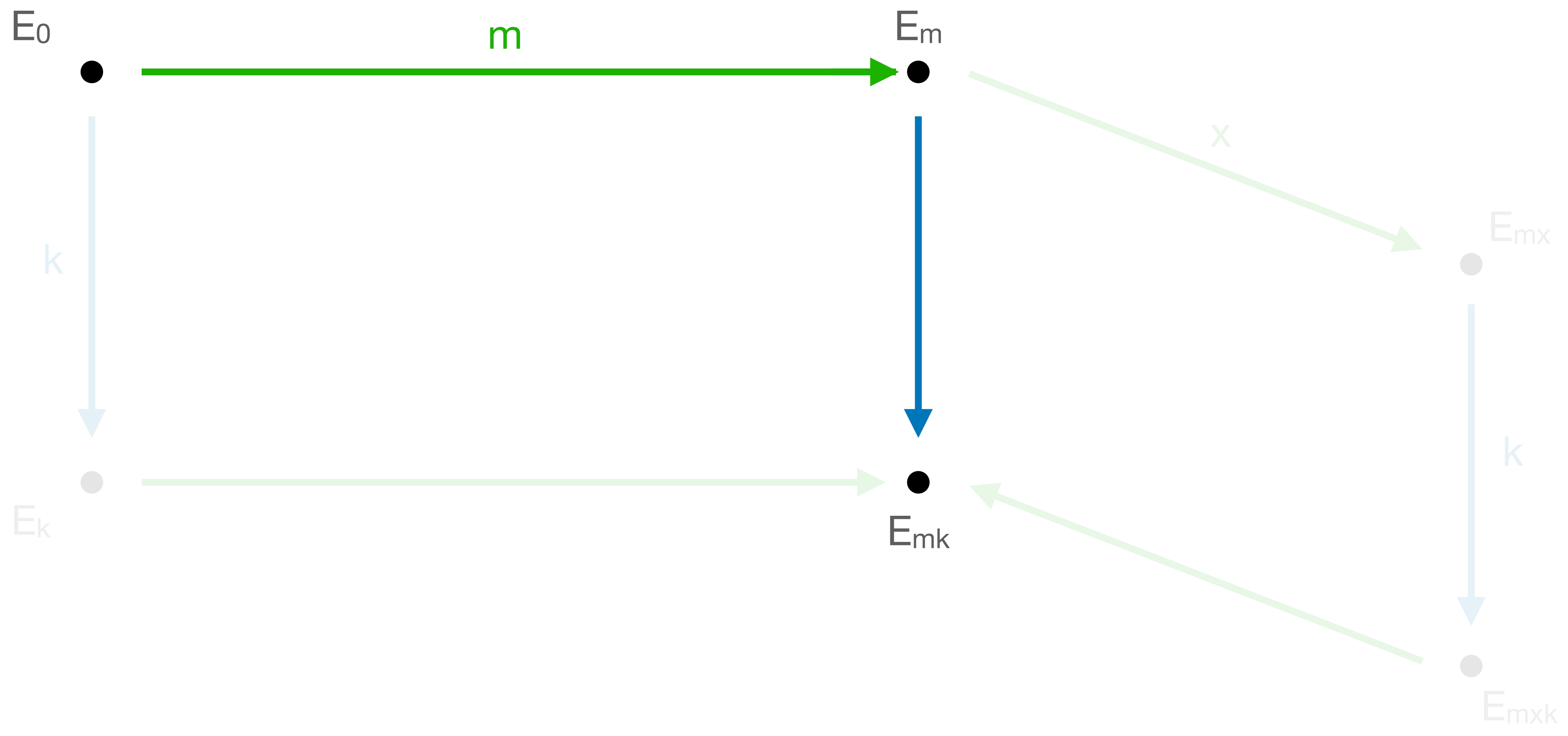
Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs



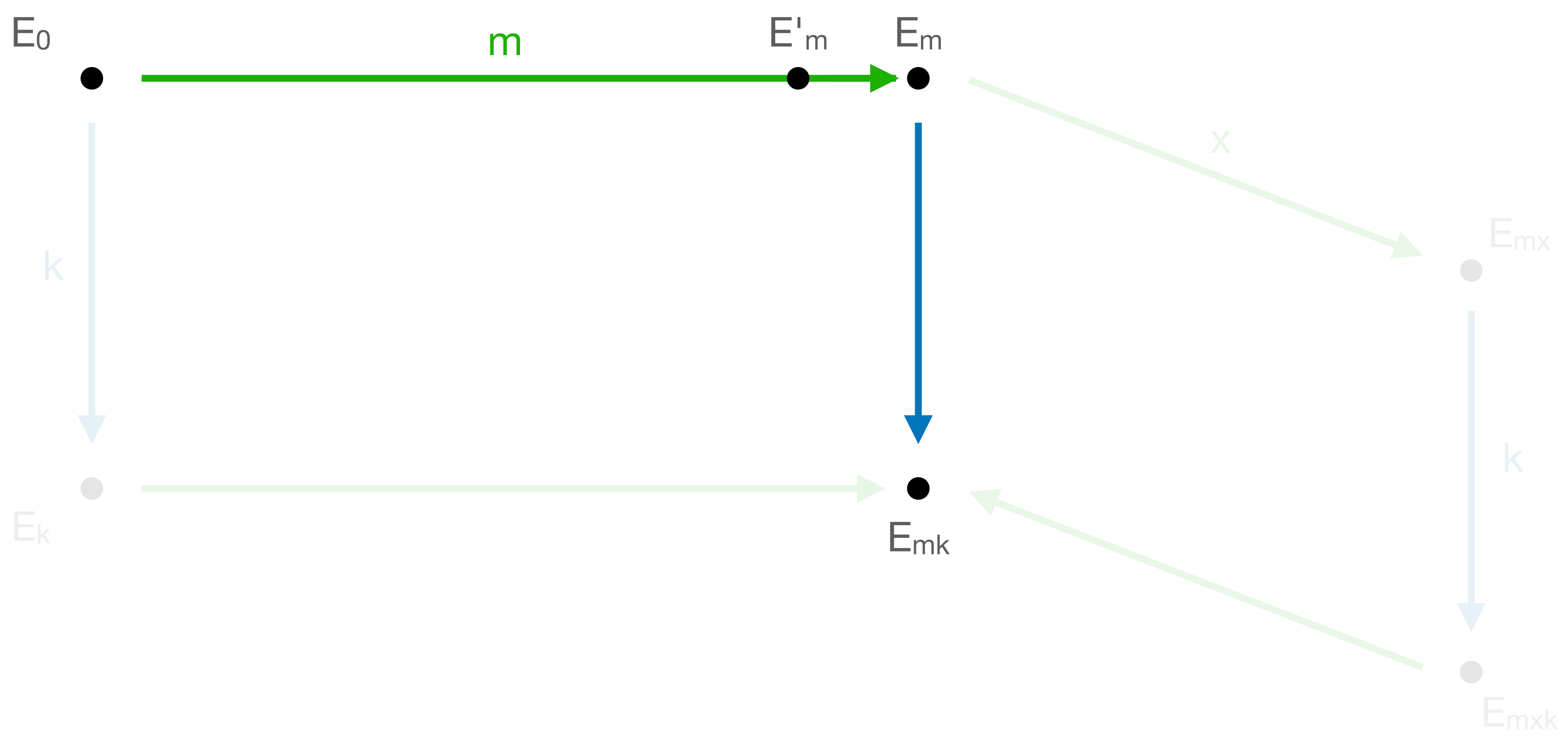
Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs



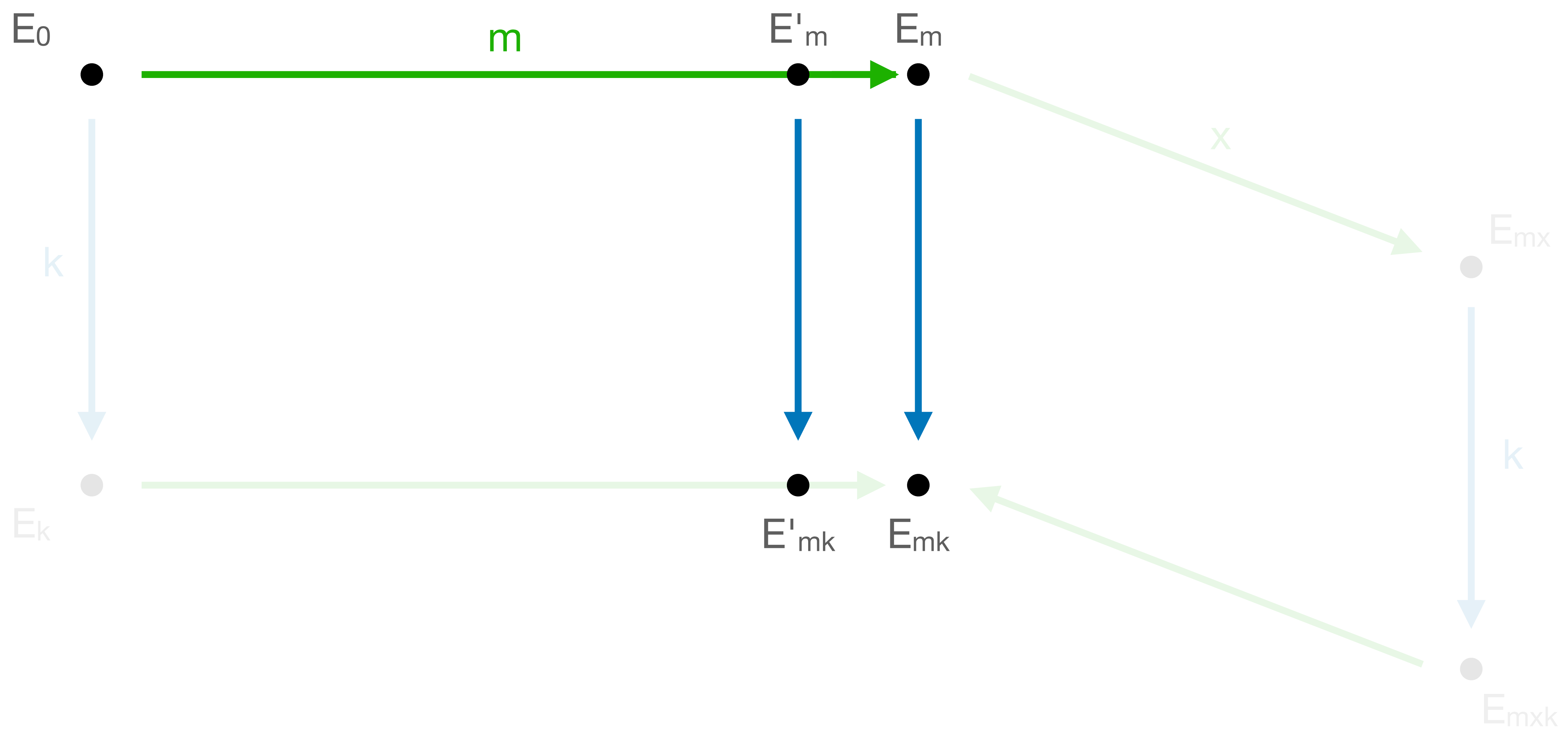
Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs



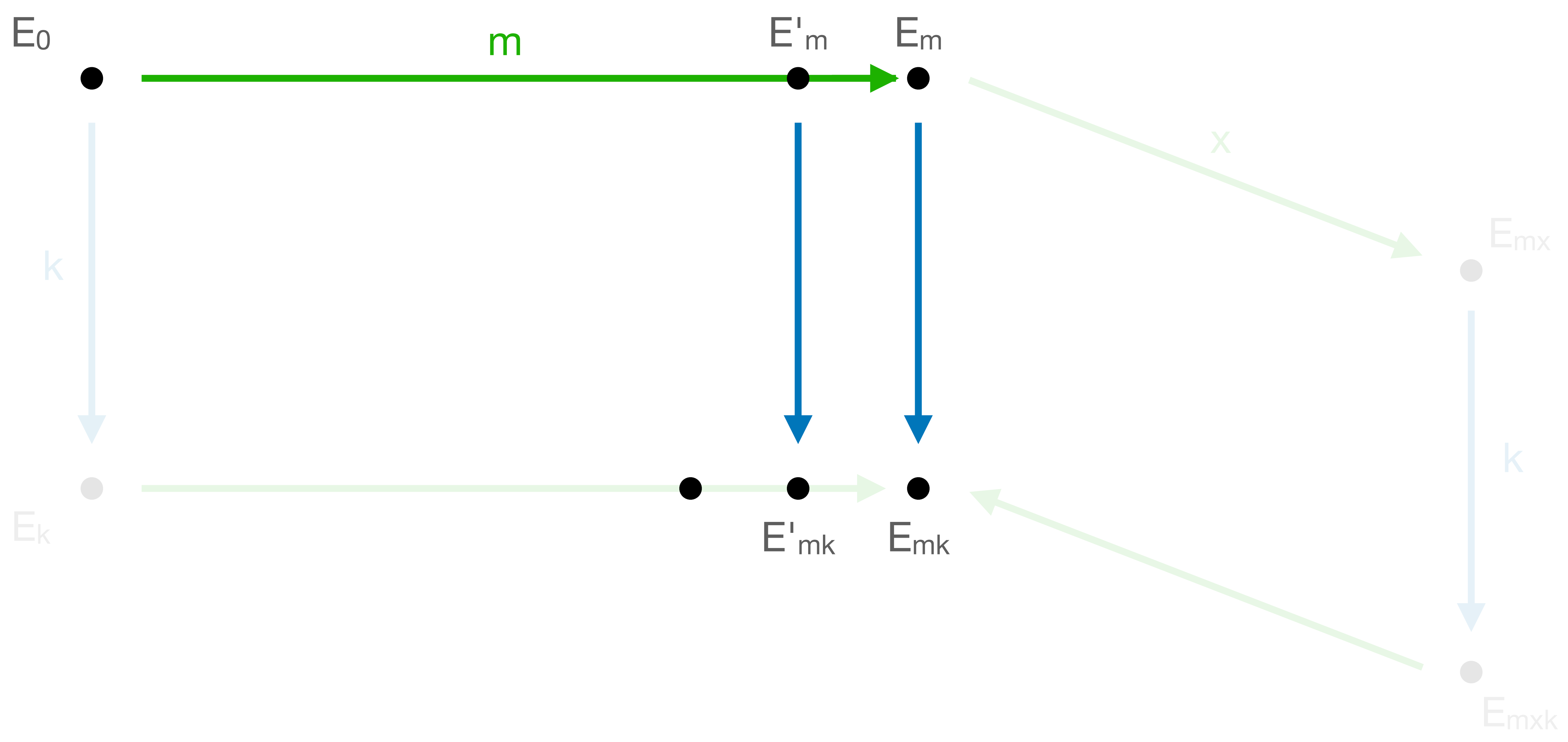
Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs



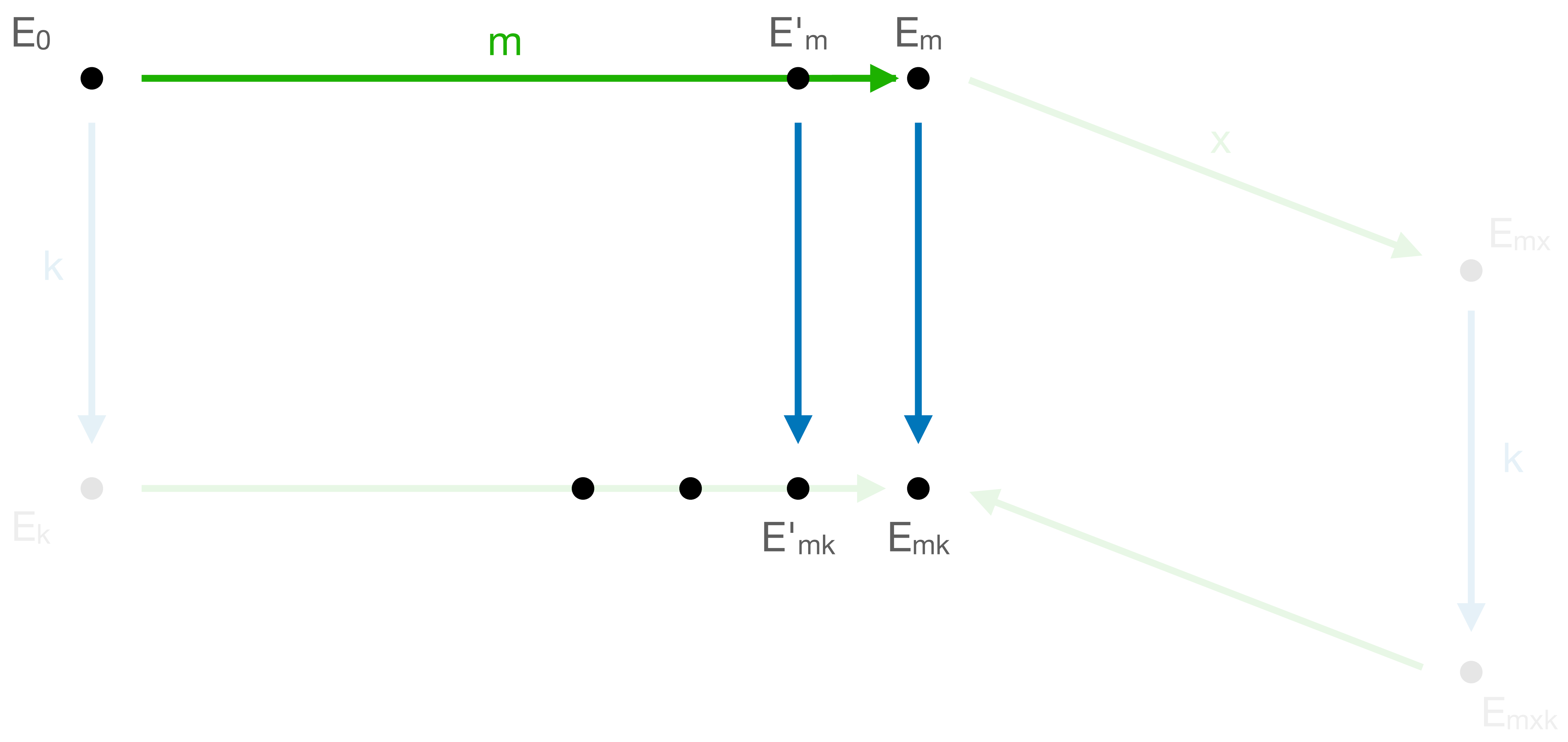
Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs



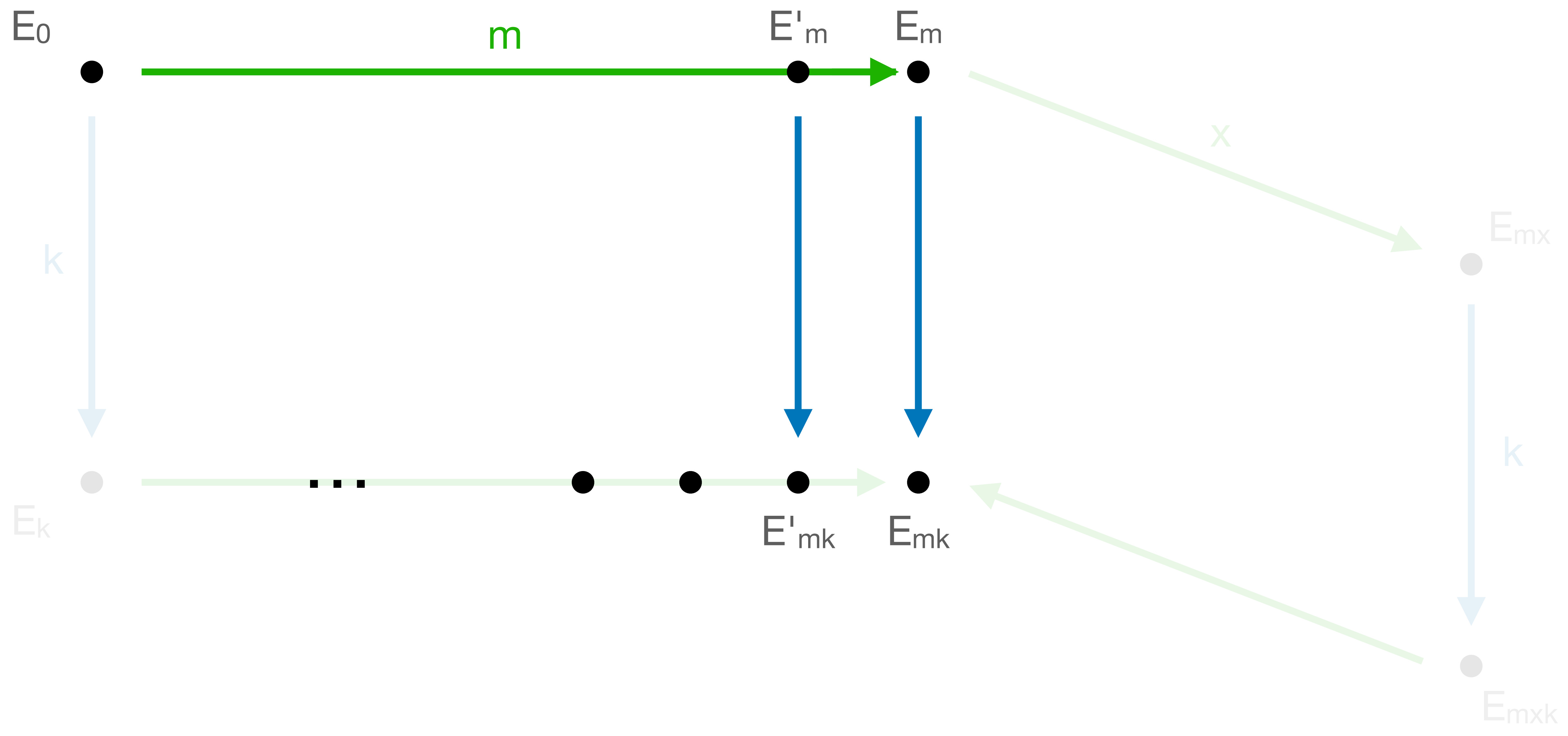
Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs



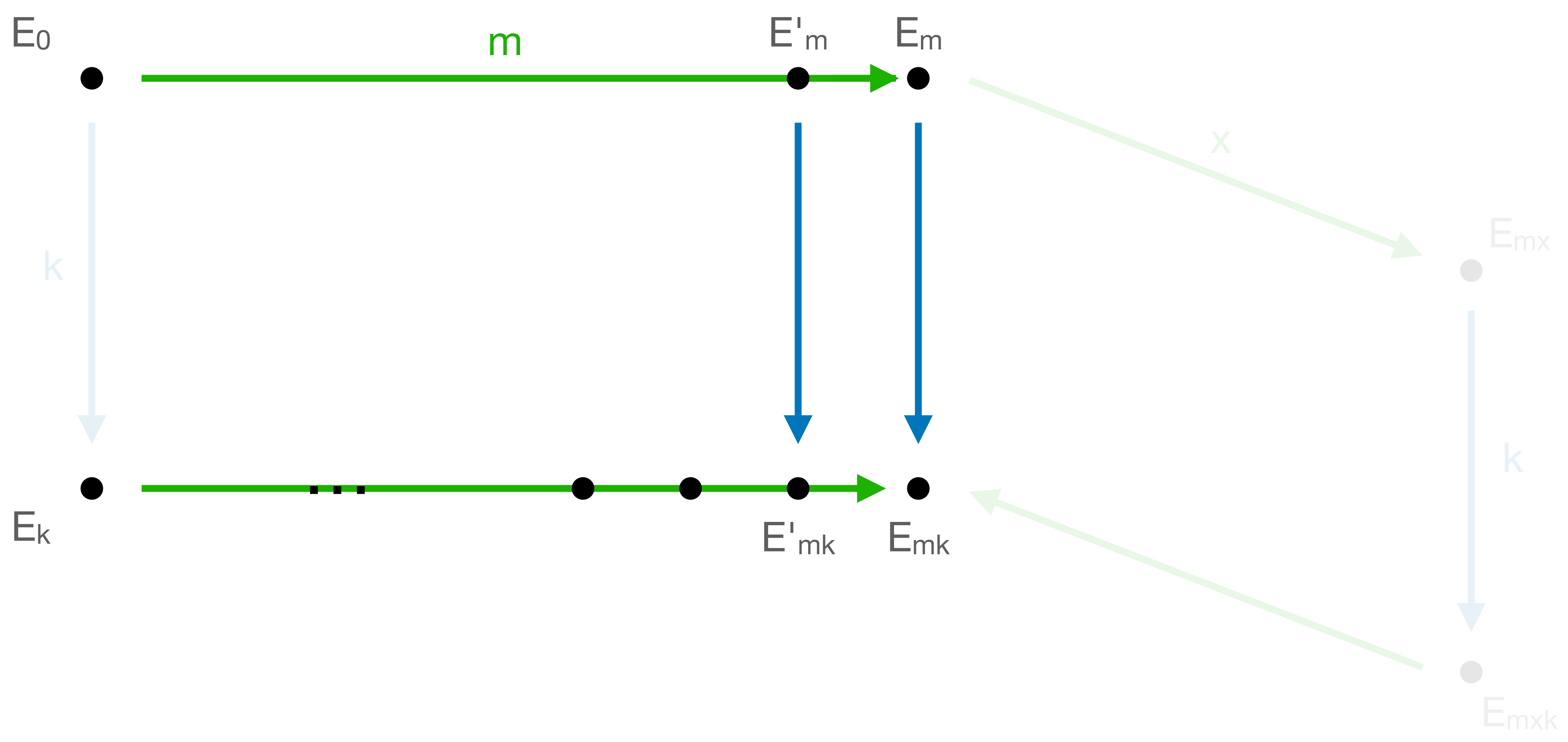
Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs



Breaking pseudorandomness [BKMPS21]

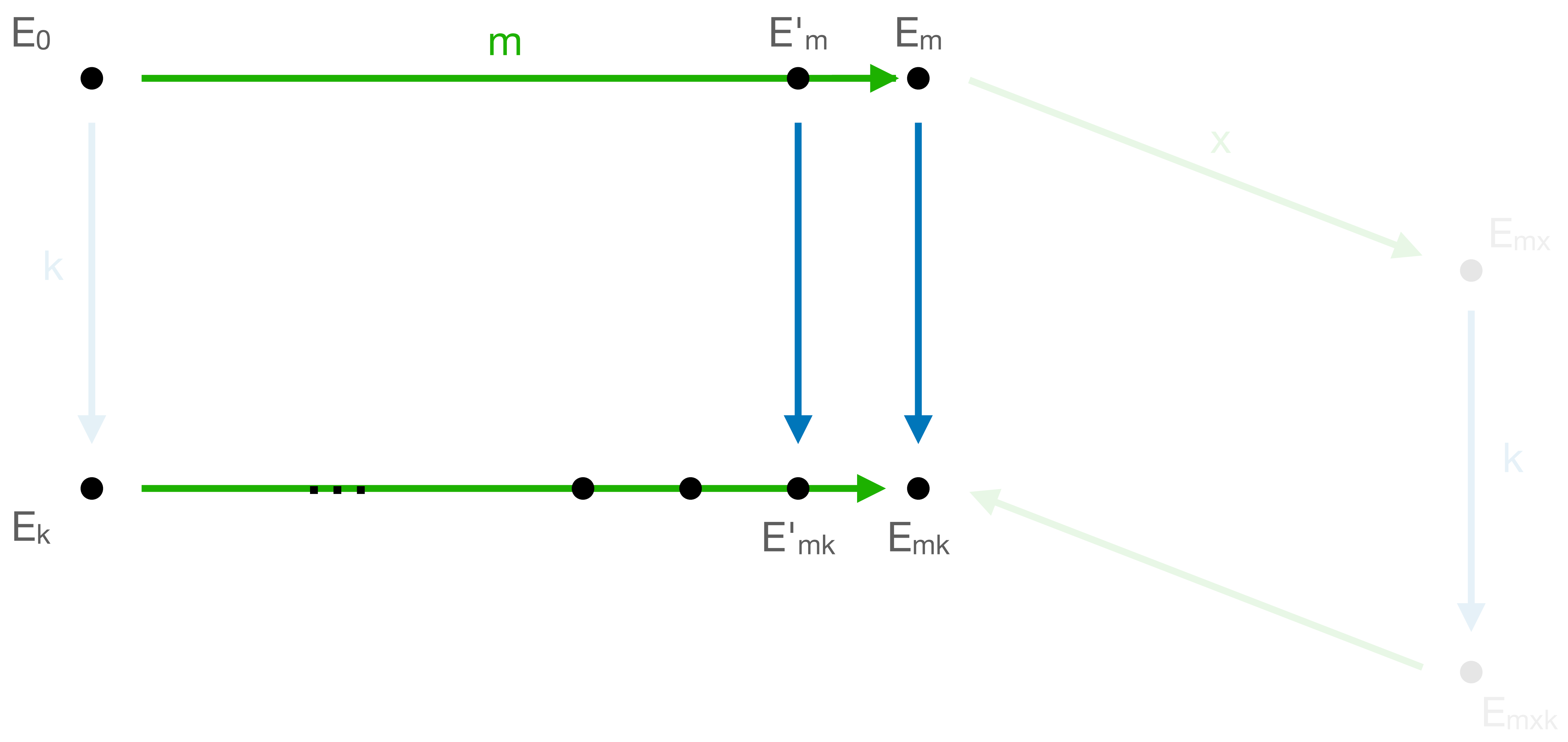
Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs



Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs

Part 1

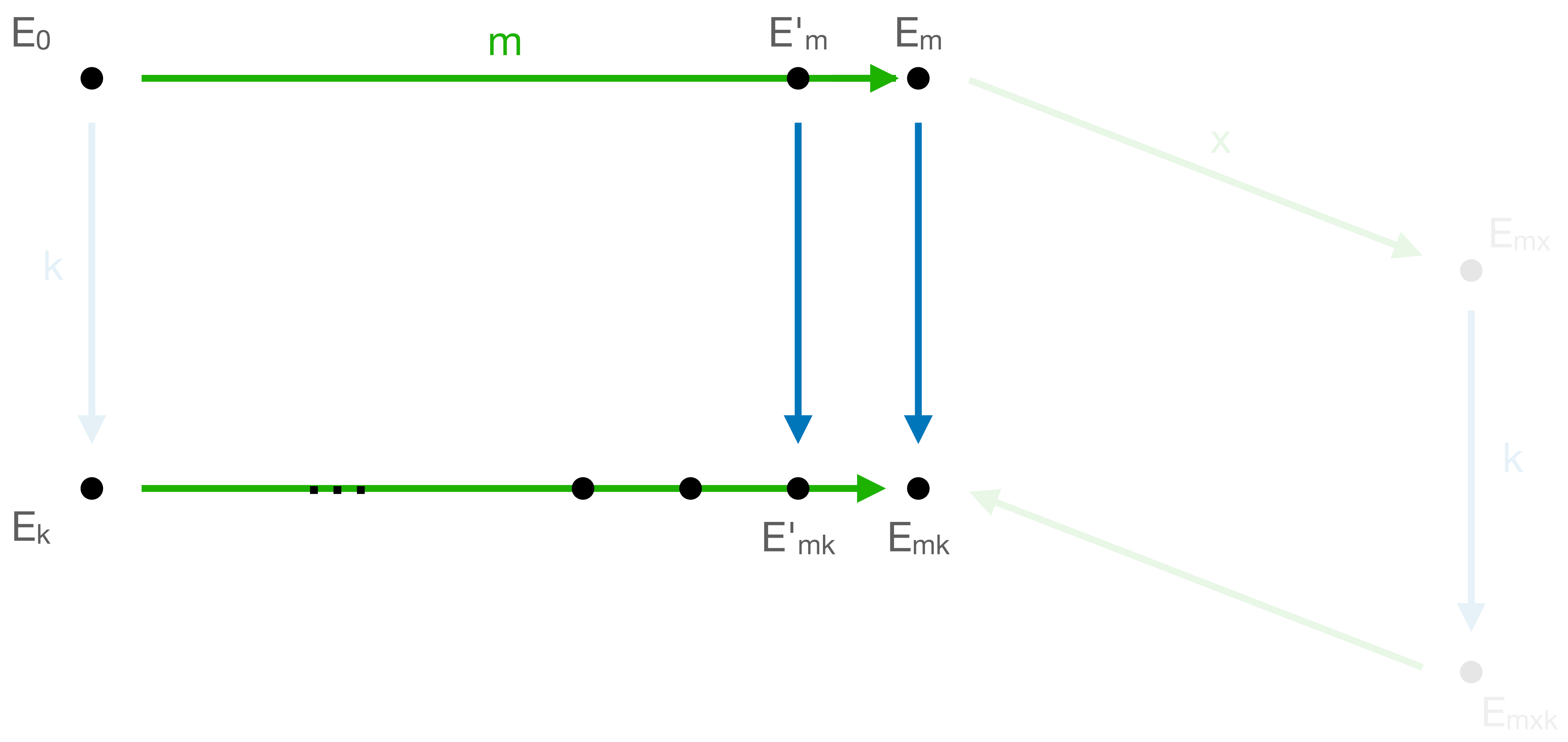


Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs

Part 1

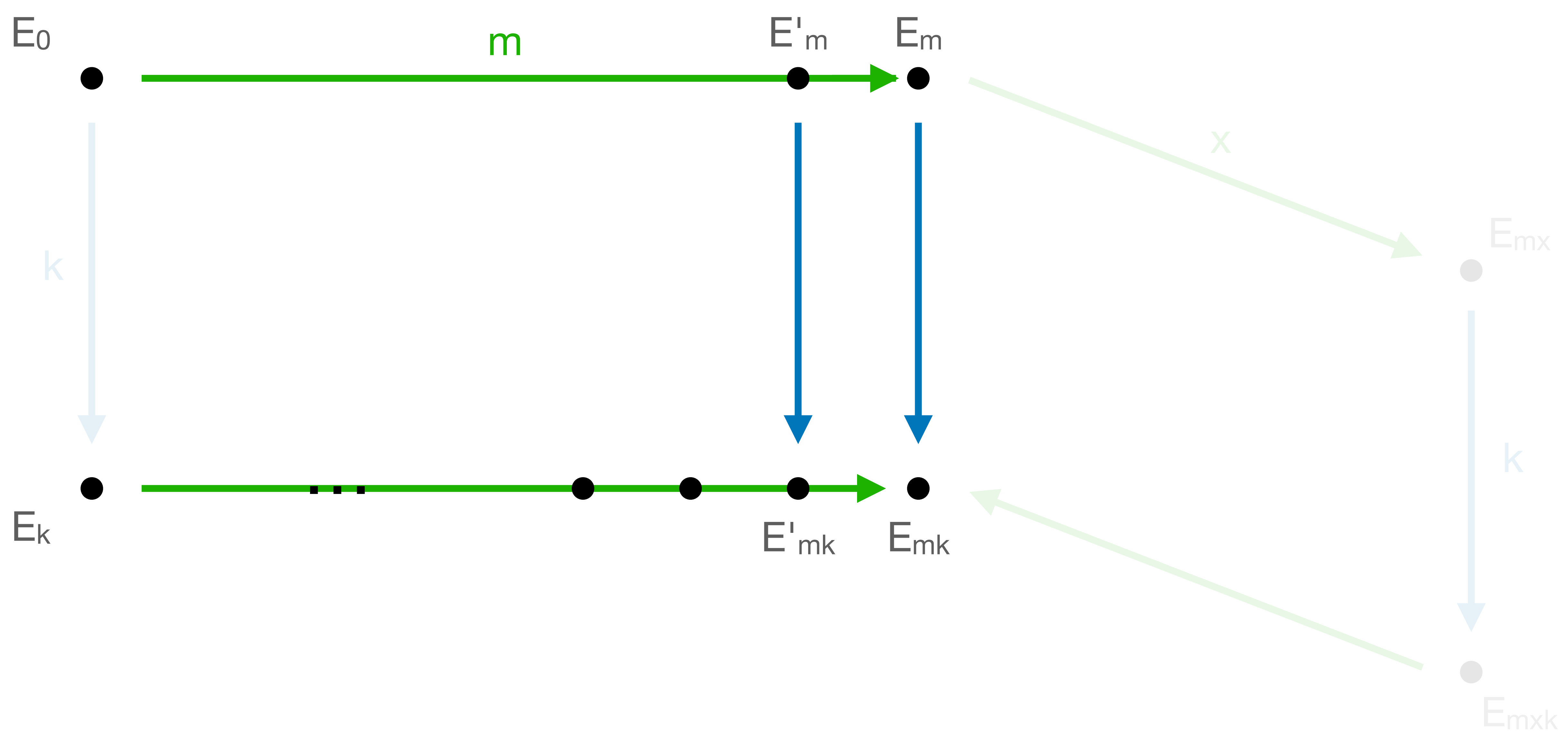
Part 2



Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs

Part 1



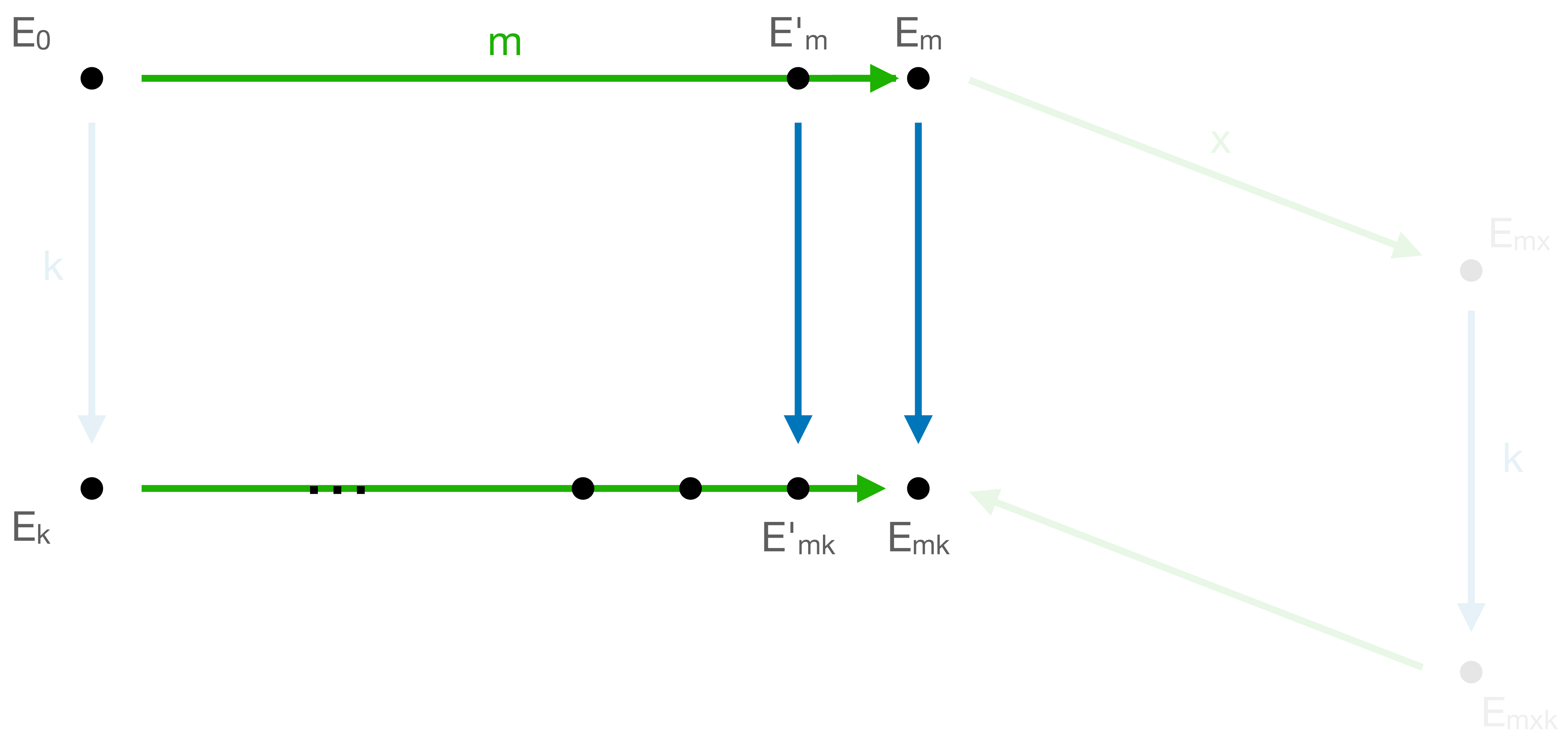
Part 2

- Repeat the attack 3 times

Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs

Part 1



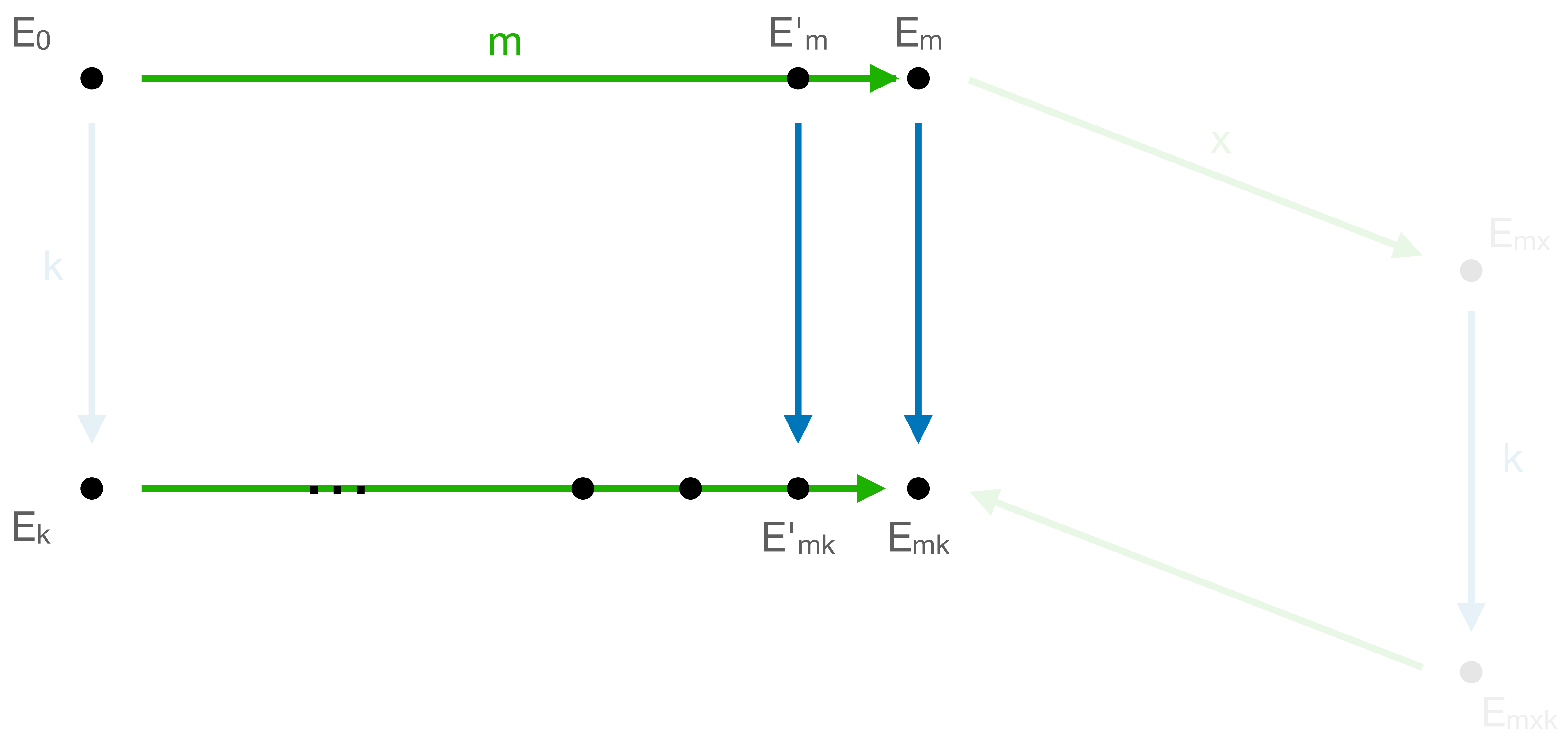
Part 2

- Repeat the attack 3 times
- Find a basis on E_k

Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs

Part 1



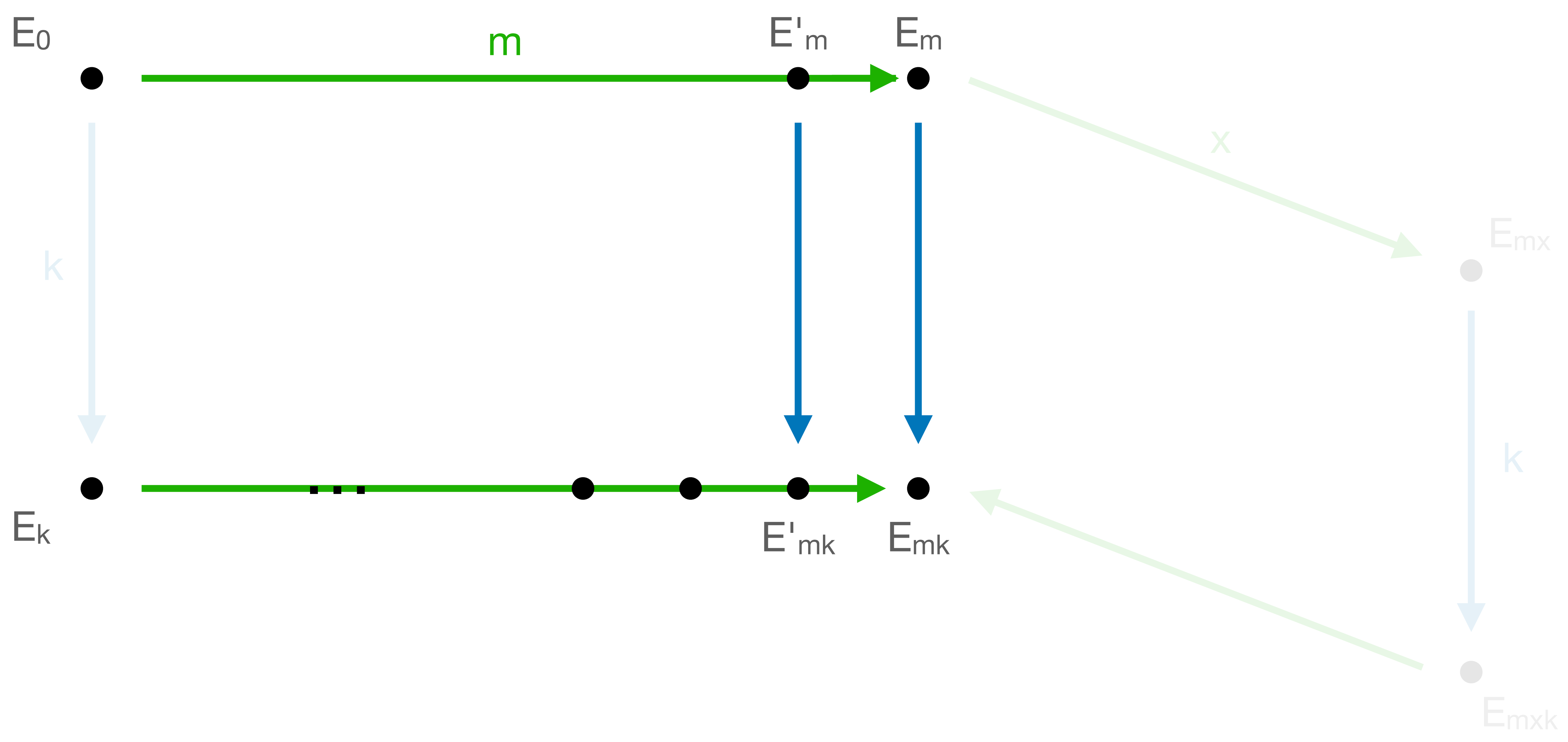
Part 2

- Repeat the attack 3 times
- Find a basis on E_k
- Evaluate the PRF on *any* message

Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs

Part 1



Part 2

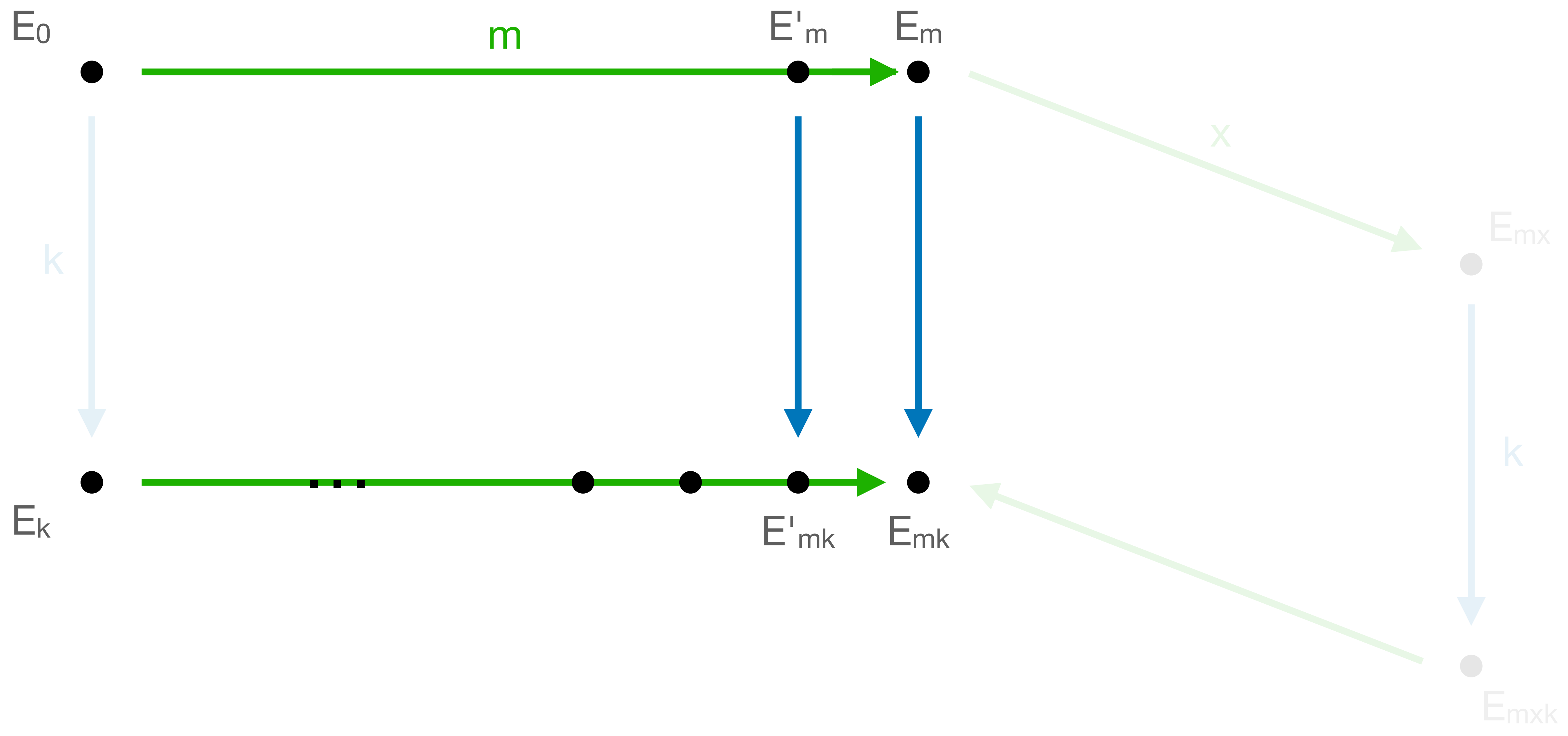
- Repeat the attack 3 times
- Find a basis on E_k
- Evaluate the PRF on *any* message

The server can check the degree with the PoK!

Breaking pseudorandomness [BKMPS21]

Pseudorandomness: after n interactions, an attacker cannot generate n+1 PRF outputs

Part 1



Part 2

- Repeat the attack 3 times
- Find a basis on E_k
- Evaluate the PRF on *any* message

The server can check the degree with the PoK!

Actual complexity: sub-exponential

Countermeasures?

It seems hard to prevent an attacker from recovering a basis on E_k

Countermeasures?

It seems hard to prevent an attacker from recovering a basis on E_k

Validate more

Ensure that the client submits valid message isogenies

Countermeasures?

It seems hard to prevent an attacker from recovering a basis on E_k

Validate more

Ensure that the client submits valid message isogenies

↓

The protocol is oblivious

Countermeasures?

It seems hard to prevent an attacker from recovering a basis on E_k

Validate more

Ensure that the client submits valid message isogenies

↓

The protocol is oblivious

Update values

Use dynamic values for server’s computations

Countermeasures?

It seems hard to prevent an attacker from recovering a basis on E_k

Validate more

Ensure that the client submits valid message isogenies

↓

The protocol is oblivious

Update values

Use dynamic values for server's computations

↓

The PRF needs to be deterministic

Countermeasures?

It seems hard to prevent an attacker from recovering a basis on E_k

Validate more

Ensure that the client submits valid message isogenies

↓

The protocol is oblivious

Update values

Use dynamic values for server's computations

↓

The PRF needs to be deterministic

Scale parameters

Attack is sub exponential

Countermeasures?

It seems hard to prevent an attacker from recovering a basis on E_k

Validate more

Ensure that the client submits valid message isogenies

↓

The protocol is oblivious

Update values

Use dynamic values for server's computations

↓

The PRF needs to be deterministic

Scale parameters

Attack is sub exponential

↓

$p > 2^{16,000}$

Countermeasures?

It seems hard to prevent an attacker from recovering a basis on E_k

Validate more

Ensure that the client submits valid message isogenies

↓

The protocol is oblivious

Update values

Use dynamic values for server's computations

↓

The PRF needs to be deterministic

Scale parameters

Attack is sub exponential

↓

$p > 2^{16,000}$

Idea: make the basis on E_k not enough for an attack

An efficient countermeasure

[BKW20]

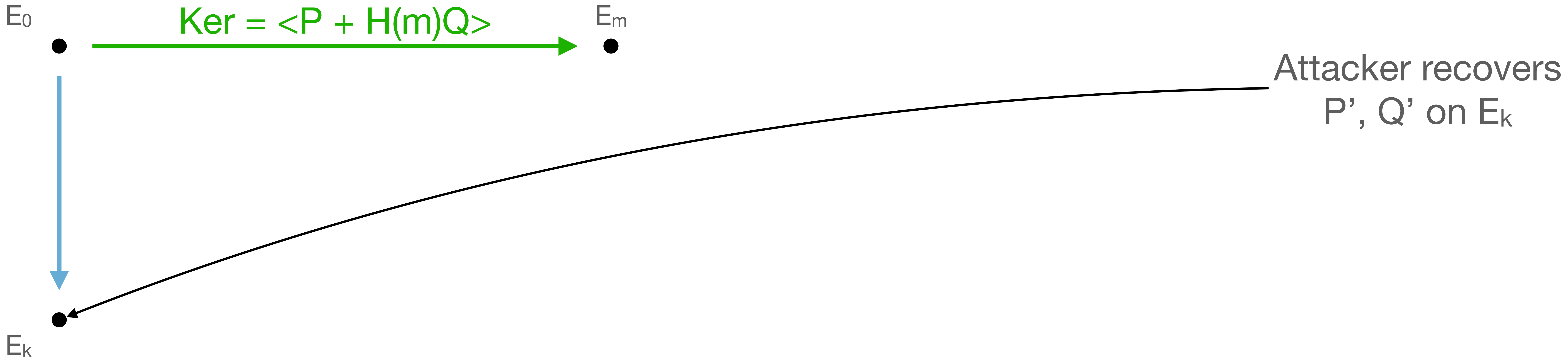
An efficient countermeasure

$$\begin{tikzcd} E_0 \arrow[r, "\mathrm{Ker} = <P + H(m)Q>"] & E_m \end{tikzcd}$$

[BKW20]

An efficient countermeasure

[BKW20]



Attacker recovers
P’, Q’ on E_k

An efficient countermeasure

[BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "\mathrm{Ker} = <P + H(m)Q>"] \arrow[d] & E_m \\
E_k \arrow[r] & E_{mk}
\end{tikzcd}$$

Attacker recovers P', Q' on E_k

↓

Can evaluate the PRF on any message

An efficient countermeasure

[BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "Ker = <P + H(m)Q>"] \arrow[d] & E_m \\
E_k \arrow[r] & E_{mk}
\end{tikzcd}$$

Attacker recovers P', Q' on E_k

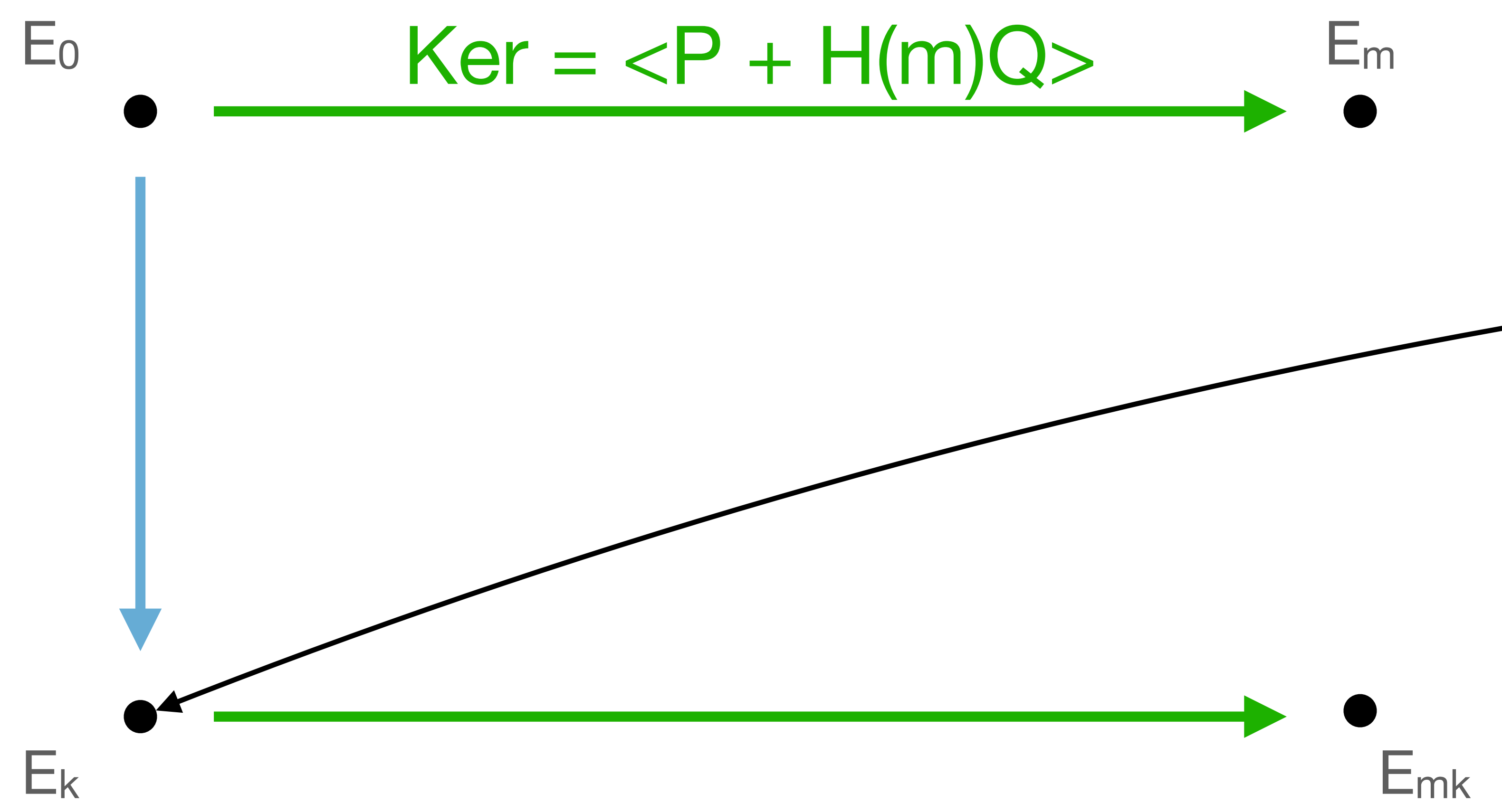
$\downarrow$

Can evaluate the PRF on any message

Our countermeasure

An efficient countermeasure

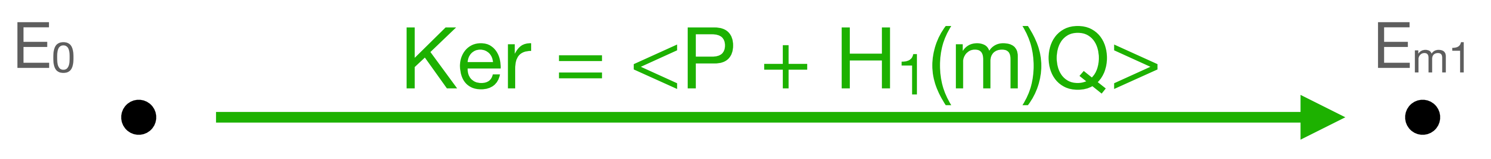
[BKW20]



Attacker recovers P', Q' on E_k

↓

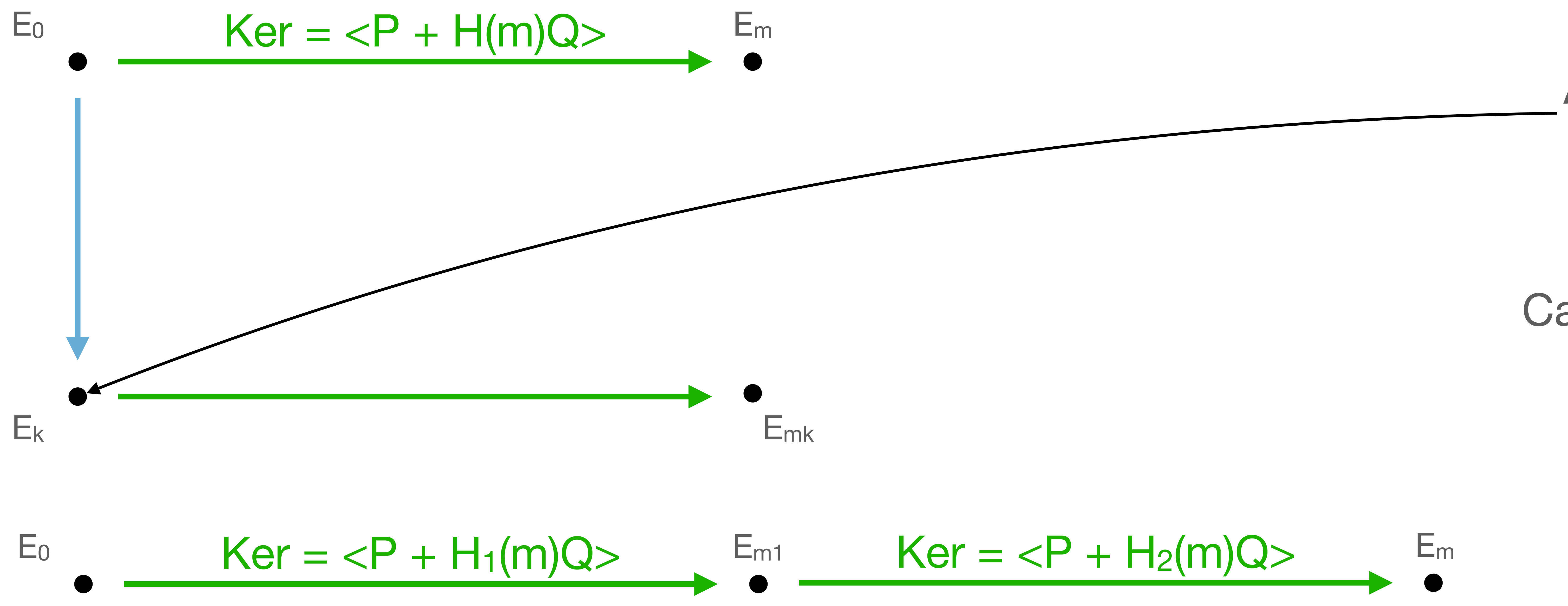
Can evaluate the PRF on any message



Our countermeasure

An efficient countermeasure

[BKW20]


E_0
$\text{Ker} = \langle P + H(m)Q \rangle$
E_m
E_k
E_{mk}
E_0
$\text{Ker} = \langle P + H_1(m)Q \rangle$
E_{m1}
$\text{Ker} = \langle P + H_2(m)Q \rangle$
E_m

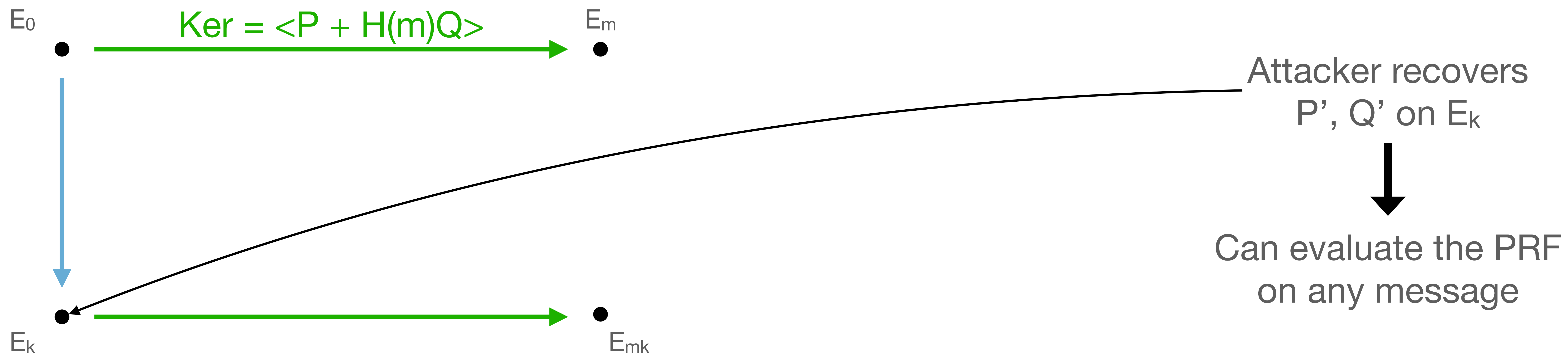
Attacker recovers P’, Q’ on E_k

Can evaluate the PRF on any message

Our countermeasure

An efficient countermeasure

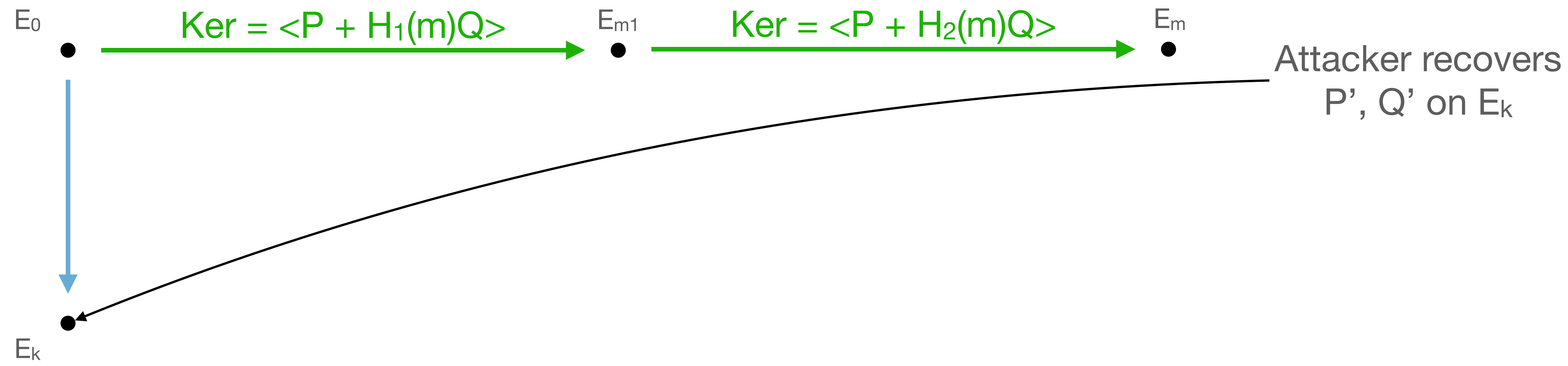
[BKW20]



Attacker recovers P', Q' on E_k

Can evaluate the PRF on any message

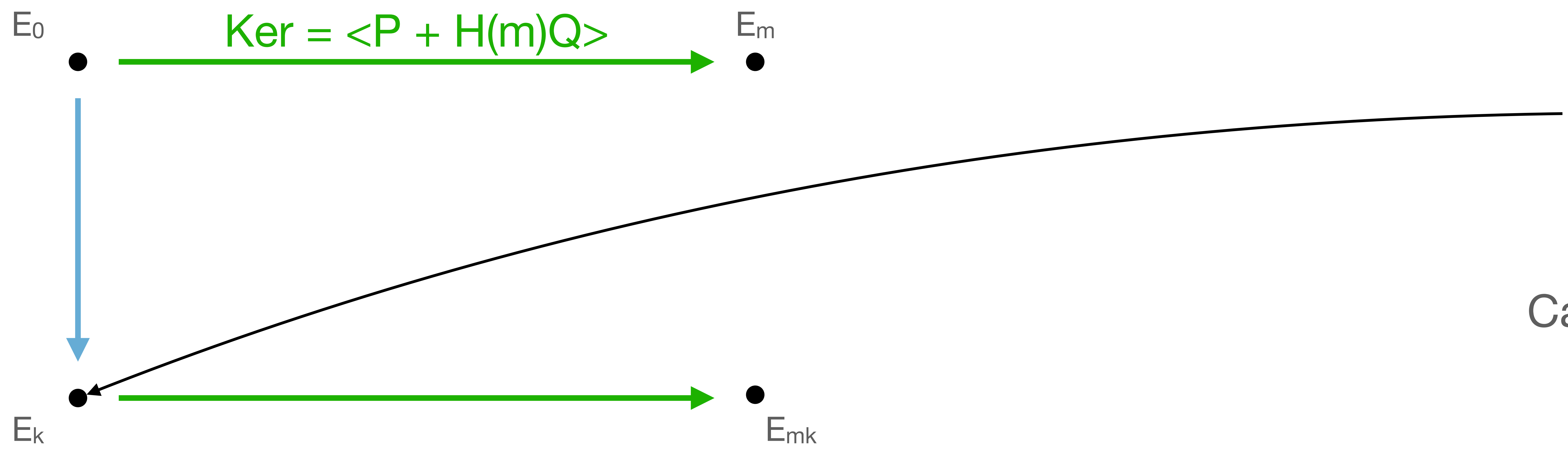
Our countermeasure



Attacker recovers P', Q' on E_k

An efficient countermeasure

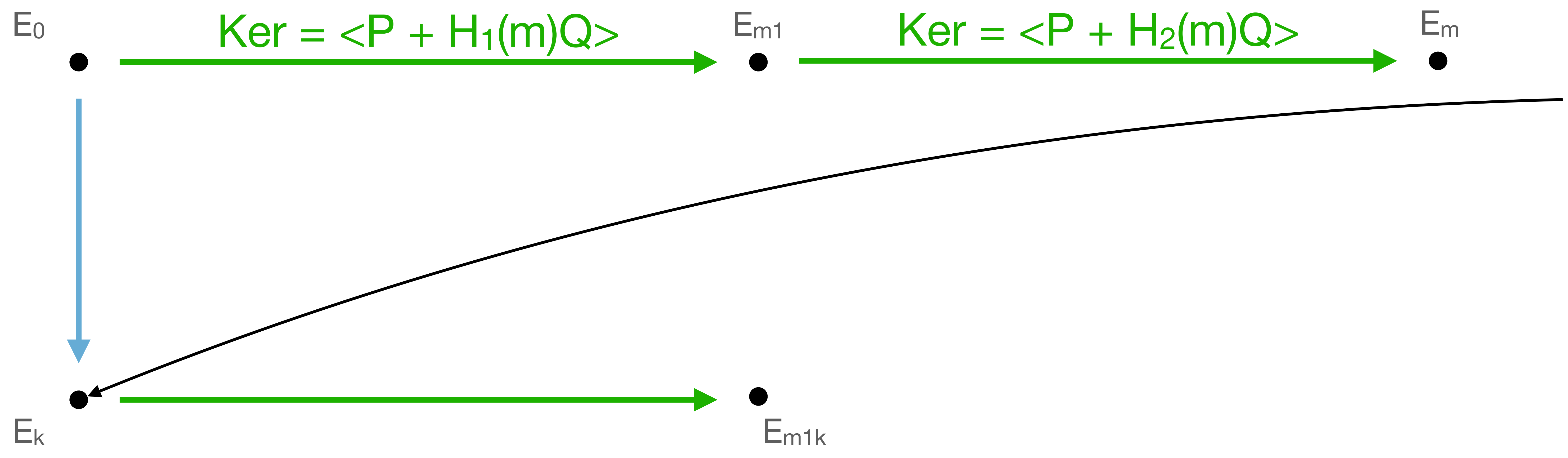
[BKW20]



Attacker recovers P', Q' on E_k

Can evaluate the PRF on any message

Our countermeasure



Attacker recovers P', Q' on E_k

Can only evaluate the first half

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

Longer isogenies

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

Longer isogenies

❌

only works for one party

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

Longer isogenies

❌

only works for one party

Masked-degree isogenies
[Mor22,FMP23]

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

Longer isogenies

❌

only works for one party

Masked-degree isogenies [Mor22,FMP23]

❌

hard to build proofs

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

Longer isogenies

❌

only works for one party

Masked-degree isogenies [Mor22,FMP23]

❌

hard to build proofs

Masked torsion points [Fou22,FMP23]

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

Longer isogenies

❌

only works for one party

Masked-degree isogenies
[Mor22,FMP23]

❌

hard to build proofs

Masked torsion points
[Fou22,FMP23]

✅

it works

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

Longer isogenies

❌

only works for one party

Masked-degree isogenies [Mor22,FMP23]

❌

hard to build proofs

Masked torsion points [Fou22,FMP23]

✅

it works

needs new PoIK

One more attack to prevent

The SIDH attacks fully break the BKW OPRF

Need to introduce SIDH countermeasures

Longer isogenies

❌

only works for one party

Masked-degree isogenies [Mor22,FMP23]

❌

hard to build proofs

Masked torsion points [Fou22,FMP23]

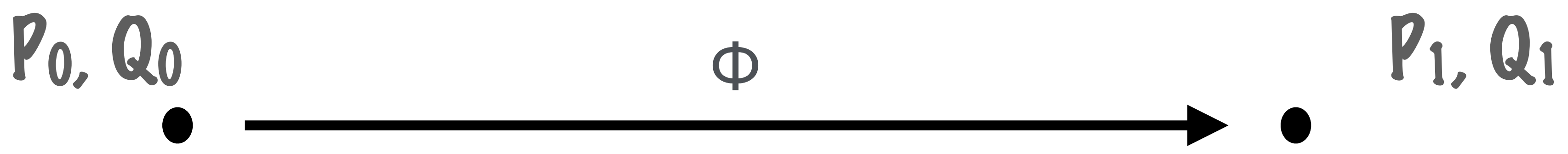
✅

it works

needs new PoIK

$p \approx 2^{6000}$

PolK with masked torsion



PolK with masked torsion

$$\begin{tikzcd}P_0, Q_0 \arrow[r, "\phi"] & {[a]P_1, [a]Q_1}\end{tikzcd}$$

PolK with masked torsion

$$\begin{tikzcd} \overset{P_0, Q_0}{\bullet} \arrow[r, "\phi"] \arrow[d] & \overset{[a]P_1, [a]Q_1}{\bullet} \\ \bullet \arrow[r] & \bullet \arrow[u] \end{tikzcd}$$

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d] & {[a]P_1, [a]Q_1} \\
\bullet \arrow[r] & \bullet \arrow[u]
\end{tikzcd}$$

challenges from {-1, 0, 1}

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d] & {[a]P_1, [a]Q_1} \\
\bullet \arrow[r] & \bullet \arrow[u]
\end{tikzcd}$$

challenges from $\{-1, 0, 1\}$

soundness error = 2/3
⇒ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d] & {[a]P_1, [a]Q_1} \\
P_2, Q_2 \arrow[r] & \bullet \arrow[u]
\end{tikzcd}$$

challenges from {-1, 0, 1}

soundness error = 2/3

⇒ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d] & {[a]P_1, [a]Q_1} \\
P_2, Q_2 \arrow[r] & P_3, Q_3 \arrow[u]
\end{tikzcd}$$

challenges from $\{-1, 0, 1\}$

soundness error = 2/3
⇒ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d] & {[a]P_1, [a]Q_1} \\
P_2, Q_2 \arrow[r] & P_3, Q_3 \arrow[u]
\end{tikzcd}$$

$a = a_1 \times a_2 \times a_3$

challenges from $\{-1, 0, 1\}$

soundness error = 2/3
⇒ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d] & {[a]P_1, [a]Q_1} \\
{[a_1]P_2, [a_1]Q_2} \arrow[r] & P_3, Q_3 \arrow[u]
\end{tikzcd}$$

$a = a_1 \times a_2 \times a_3$

challenges from $\{-1, 0, 1\}$

soundness error = 2/3
⇒ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d] & {[a]P_1, [a]Q_1} \\
{[a_1]P_2, [a_1]Q_2} \arrow[r] & {[a_2]P_3, [a_2]Q_3} \arrow[u]
\end{tikzcd}$$

$a = a_1 \times a_2 \times a_3$

challenges from {-1, 0, 1}

soundness error = 2/3
⇒ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d, "{[a_1]}"'] & {[a]P_1, [a]Q_1} \\
{[a_1]P_2, [a_1]Q_2} \arrow[r] & {[a_2]P_3, [a_2]Q_3} \arrow[u]
\end{tikzcd}$$

$a = a_1 \times a_2 \times a_3$

challenges from $\{-1, 0, 1\}$

soundness error = 2/3
⇒ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d, "{[a_1]}"'] & {[a]P_1, [a]Q_1} \\
{[a_1]P_2, [a_1]Q_2} \arrow[r, "{[a_2]}"] & {[a_2]P_3, [a_2]Q_3} \arrow[u]
\end{tikzcd}$$

$a = a_1 \times a_2 \times a_3$

challenges from {-1, 0, 1}

soundness error = 2/3

⇒ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd} P_0, Q_0 \arrow[r, "\phi"] \arrow[d, "{[a_1]}"'] & {[a]P_1, [a]Q_1} \\ {[a_1]P_2, [a_1]Q_2} \arrow[r, "{[a_2]}"] & {[a_2]P_3, [a_2]Q_3} \arrow[u, "{[a_3]}"'] \end{tikzcd}$$

$a = a_1 \times a_2 \times a_3$

challenges from $\{-1, 0, 1\}$

soundness error = 2/3

$\Rightarrow$ need 1.7λ repetitions

PoIK with masked torsion

$$\begin{tikzcd}
P_0, Q_0 \arrow[r, "\phi"] \arrow[d, "{[a_1]}"'] & {[a]P_1, [a]Q_1} \\
{[a_1]P_2, [a_1]Q_2} \arrow[r, "{[a_2]}"] & {[a_2]P_3, [a_2]Q_3} \arrow[u, "{[a_3]}"']
\end{tikzcd}$$

$a = a_1 \times a_2 \times a_3$

challenges from {-1, 0, 1}

soundness error = 2/3

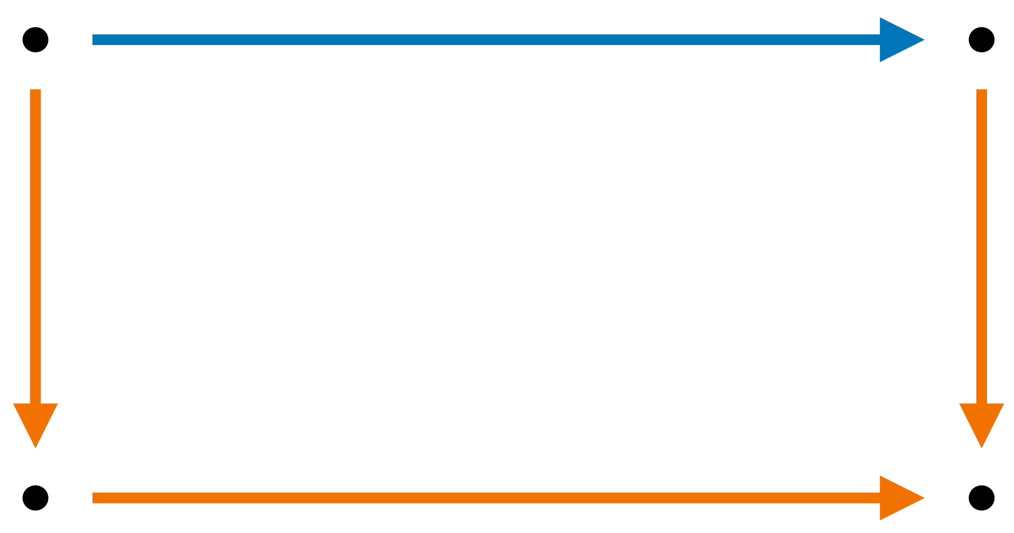
$\Rightarrow$ need 1.7λ repetitions

$p \approx \operatorname{ord} P, Q \times \deg \Phi \times \deg \rightarrow$

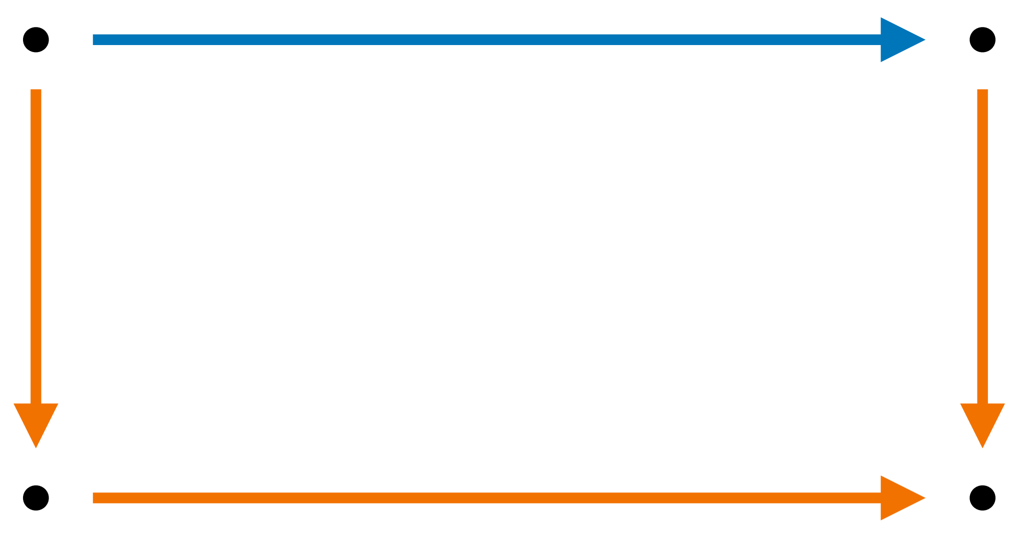
$\approx 2^{9000}$

Verifiability

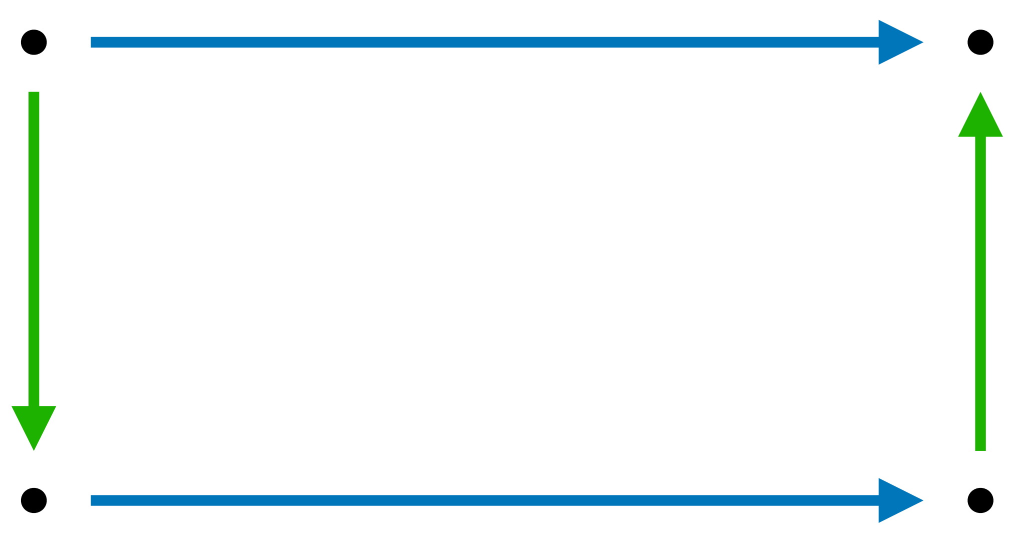
[BKW20] uses 3 proofs:



Server’s isogeny



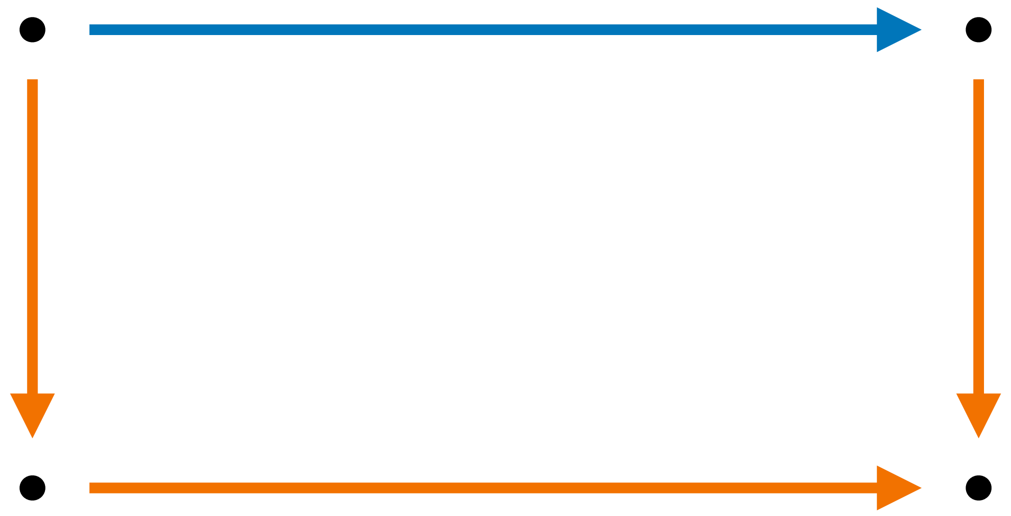
Server’s commitment



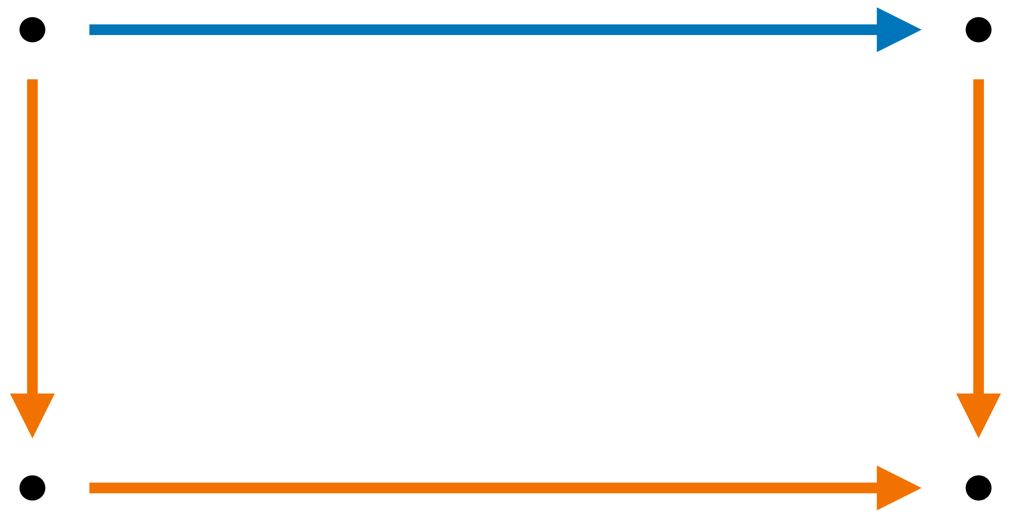
Isogeny is parallel
to commitment

Verifiability

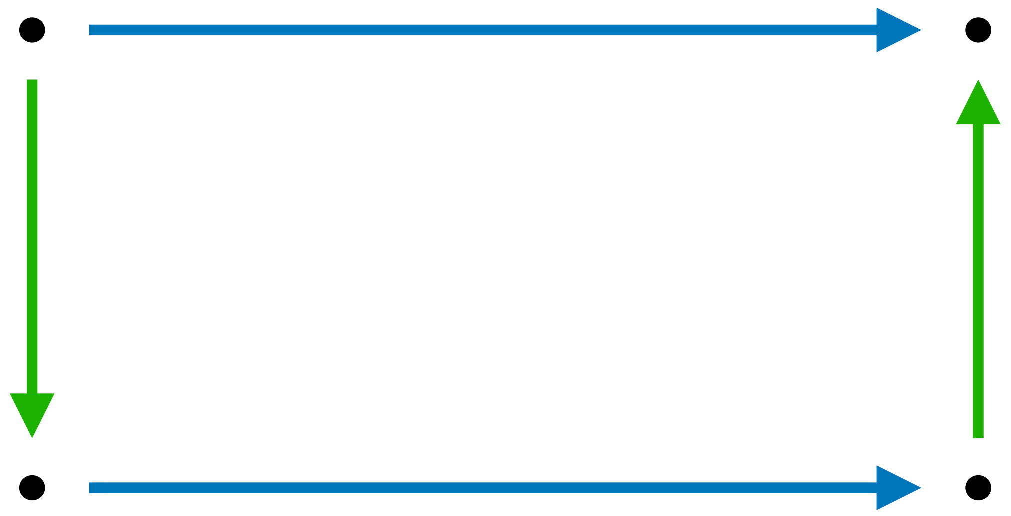
[BKW20] uses 3 proofs:



Server’s isogeny



Server’s commitment

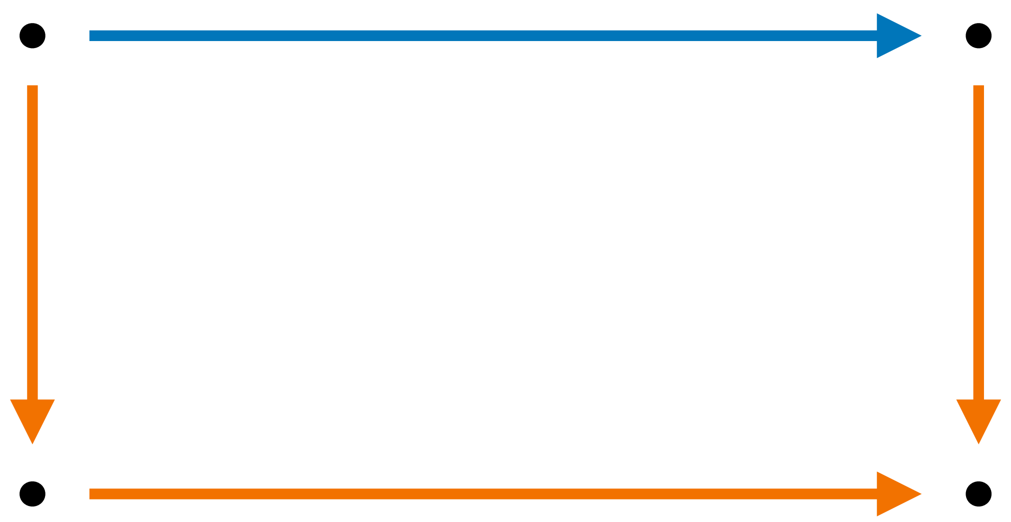


Isogeny is parallel
to commitment

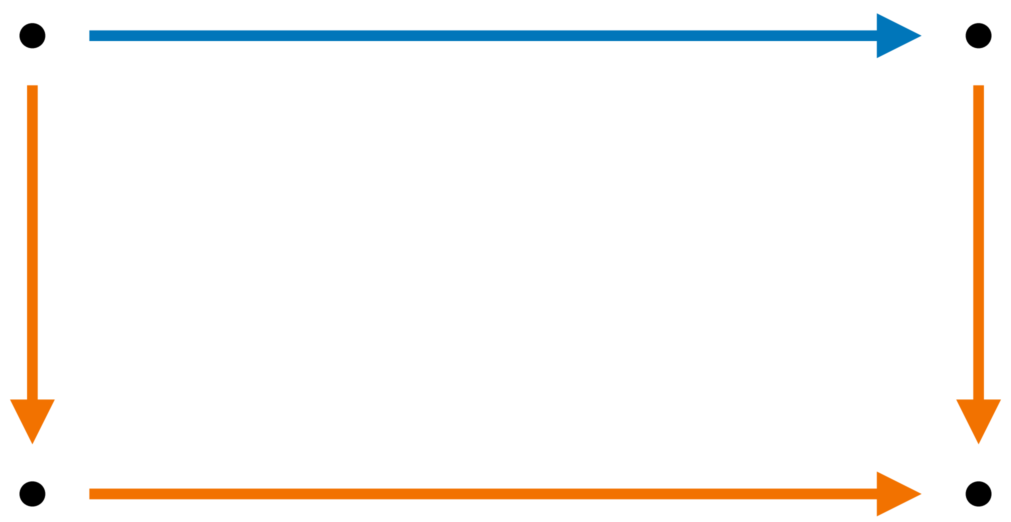
Interactive (5 rounds)

Verifiability

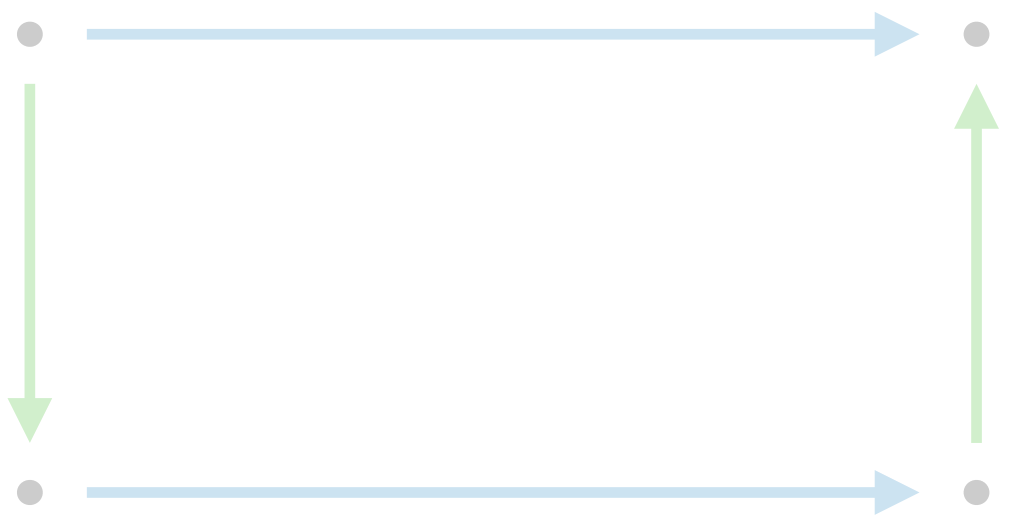
[BKW20] uses 3 proofs:



Server’s isogeny



Server’s commitment

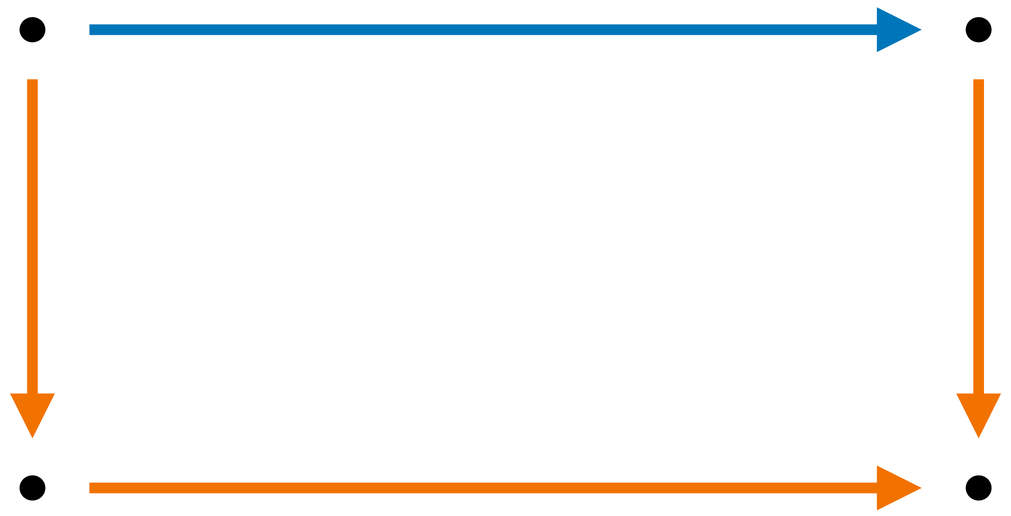


Isogeny is parallel
to commitment

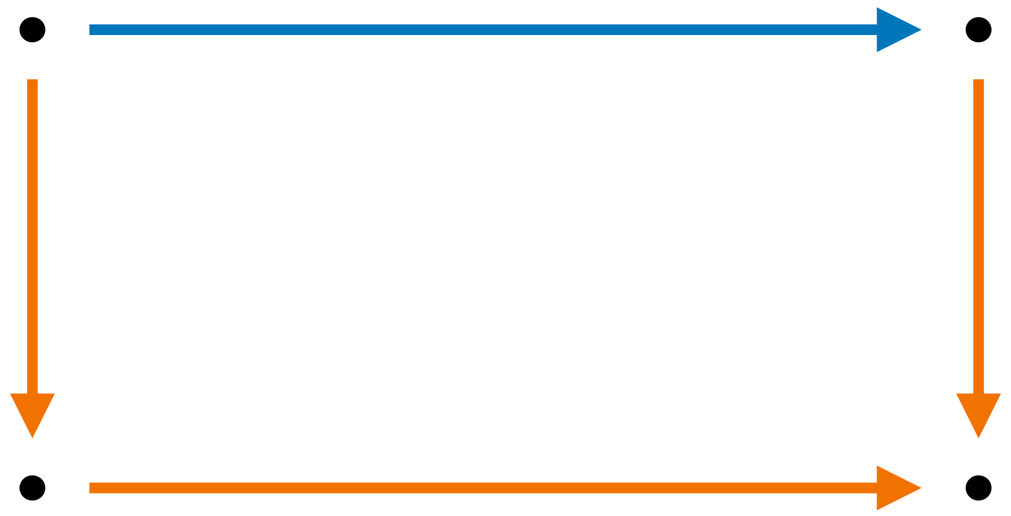
Interactive (5 rounds)

Verifiability

[BKW20] uses 3 proofs:

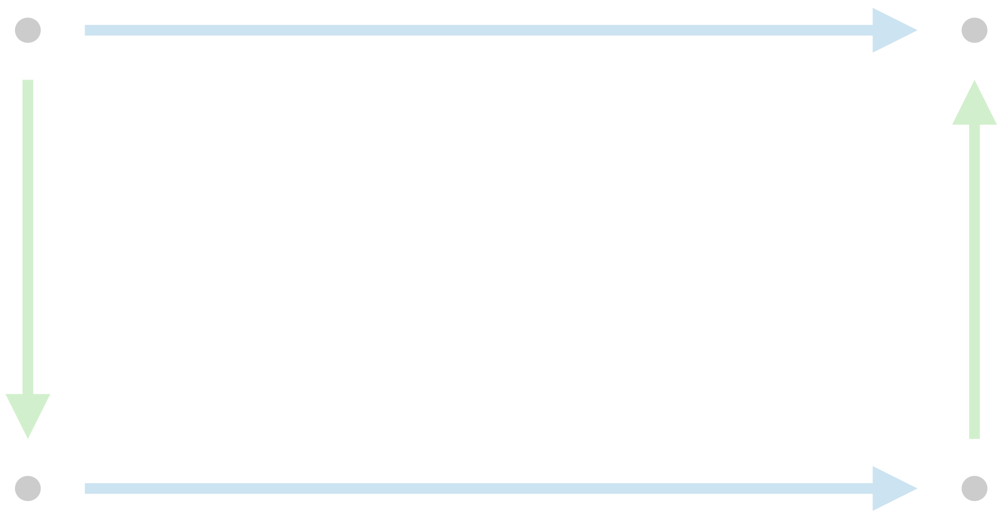


Server’s isogeny



Server’s commitment

Run together

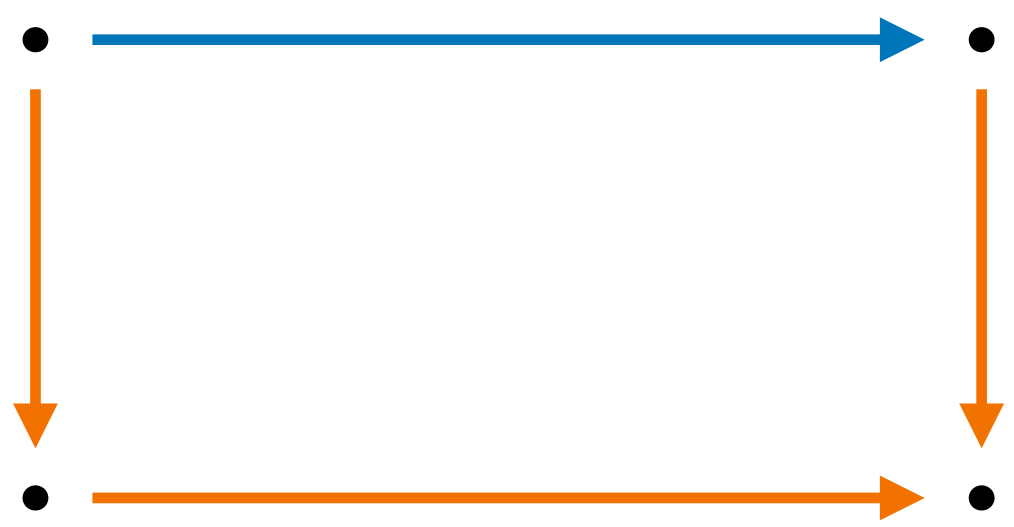


Isogeny is parallel
to commitment

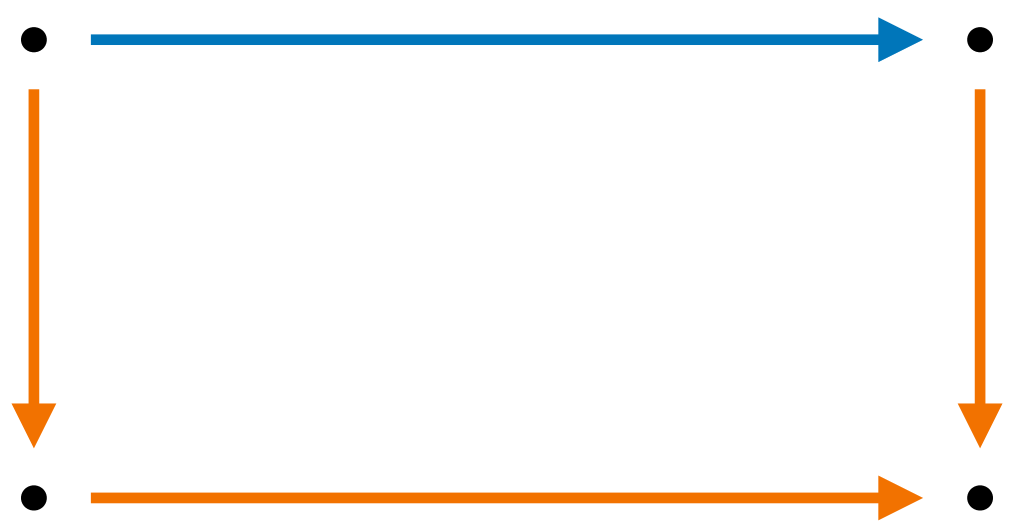
Interactive (5 rounds)

Verifiability

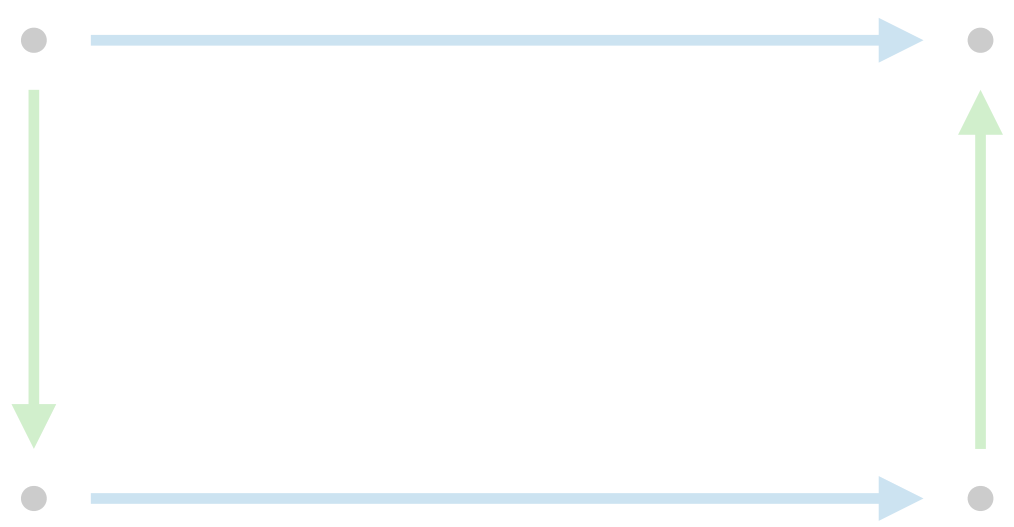
[BKW20] uses 3 proofs:



Server’s isogeny



Server’s commitment



Isogeny is parallel
to commitment

Run together

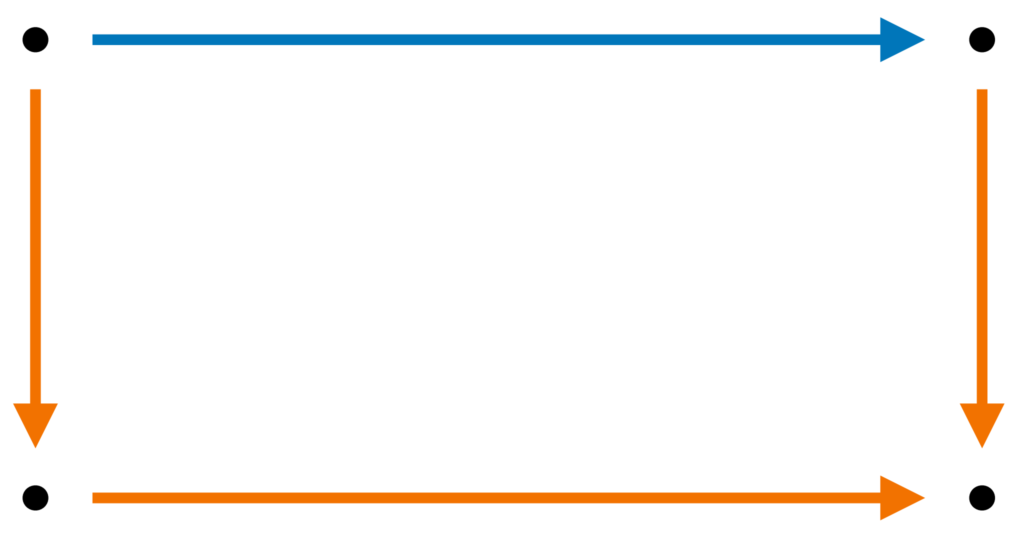
↓

Prove “parallelness" when
revealing horizontal isogeny

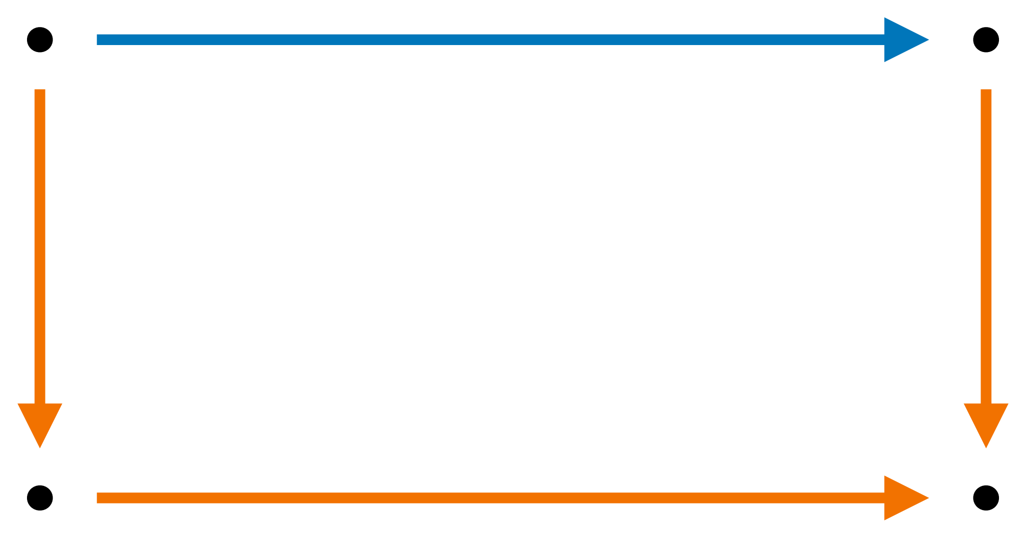
Interactive (5 rounds)

Verifiability

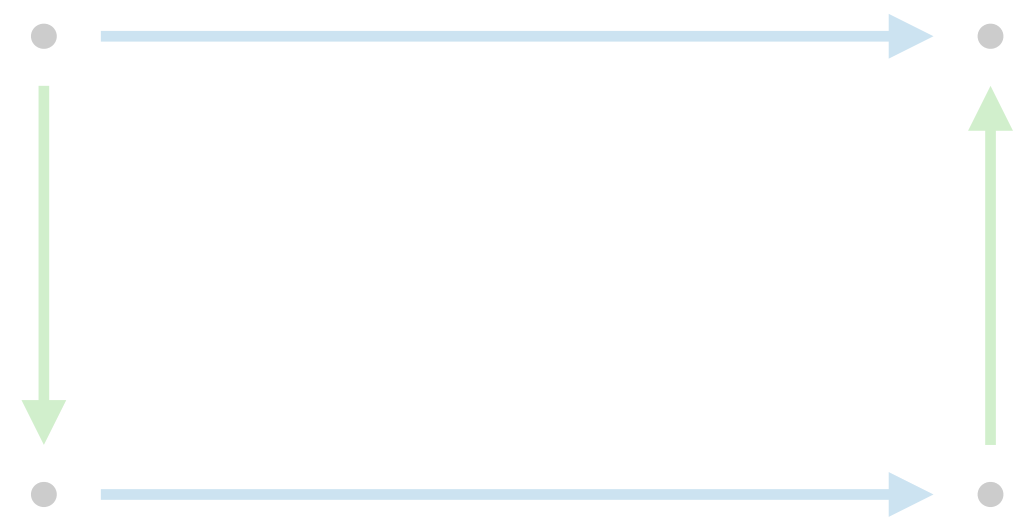
[BKW20] uses 3 proofs:



Server’s isogeny



Server’s commitment



Isogeny is parallel
to commitment

Interactive (5 rounds)

Run together

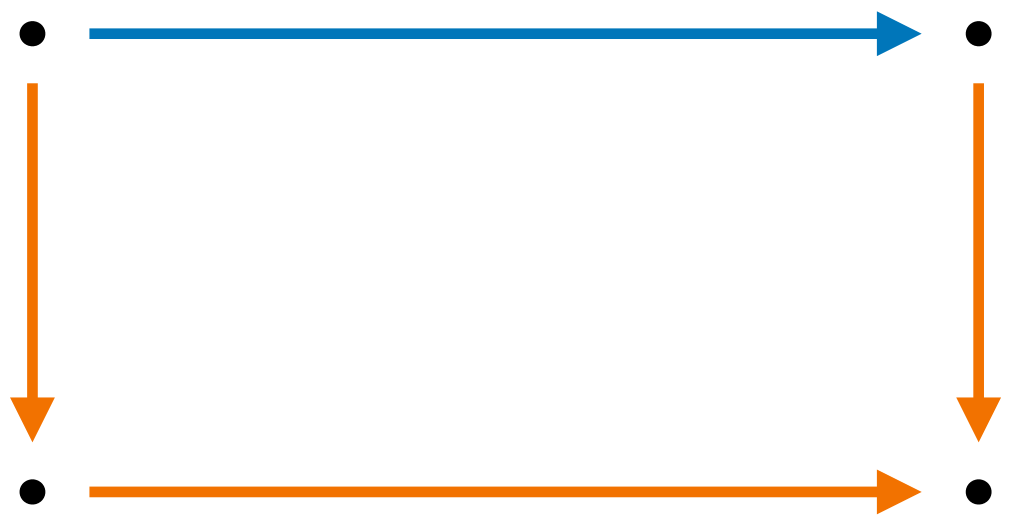
↓

Prove “parallelness" when
revealing horizontal isogeny

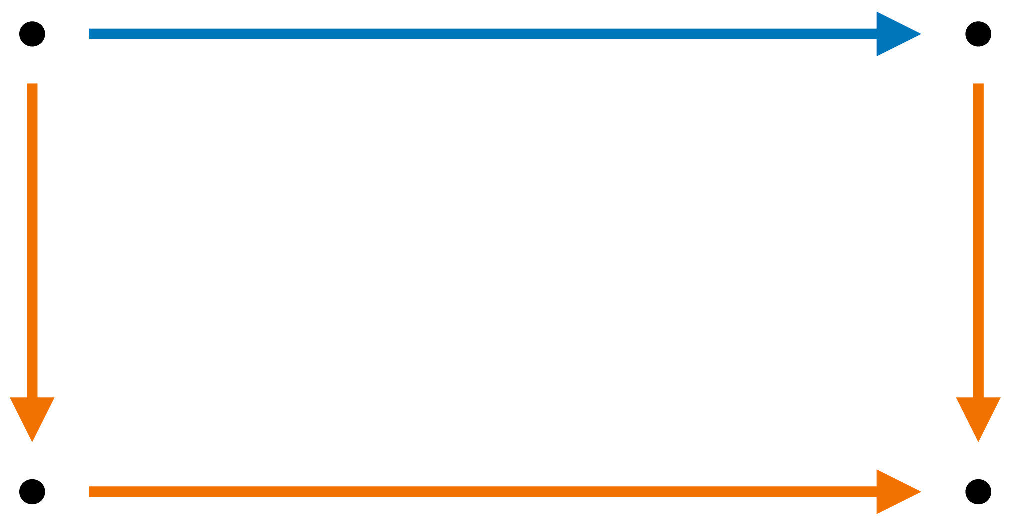
Non-interactive

Verifiability

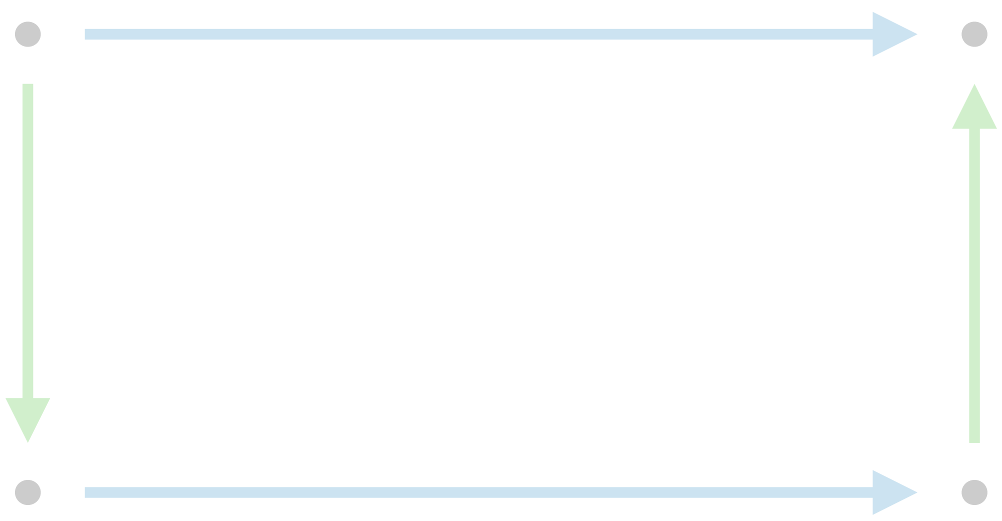
[BKW20] uses 3 proofs:



Server's isogeny



Server's commitment



Isogeny is parallel
to commitment

Run together

↓

Prove "parallelness" when
revealing horizontal isogeny

Non-interactive

Saves computations

Interactive (5 rounds)

Putting it all together

Putting it all together

- One-more unpredictability countermeasure

Putting it all together

- One-more unpredictability countermeasure

more efficient than original

new security assumption

Putting it all together

- One-more unpredictability countermeasure

more efficient than original

new security assumption

- Integrated SIDH countermeasures

Putting it all together

- One-more unpredictability countermeasure

more efficient than original

new security assumption

- Integrated SIDH countermeasures

novel proof of isogeny knowledge

prime is still large

Putting it all together

- One-more unpredictability countermeasure

more efficient than original

new security assumption

- Integrated SIDH countermeasures

novel proof of isogeny knowledge

prime is still large

- New PoPI

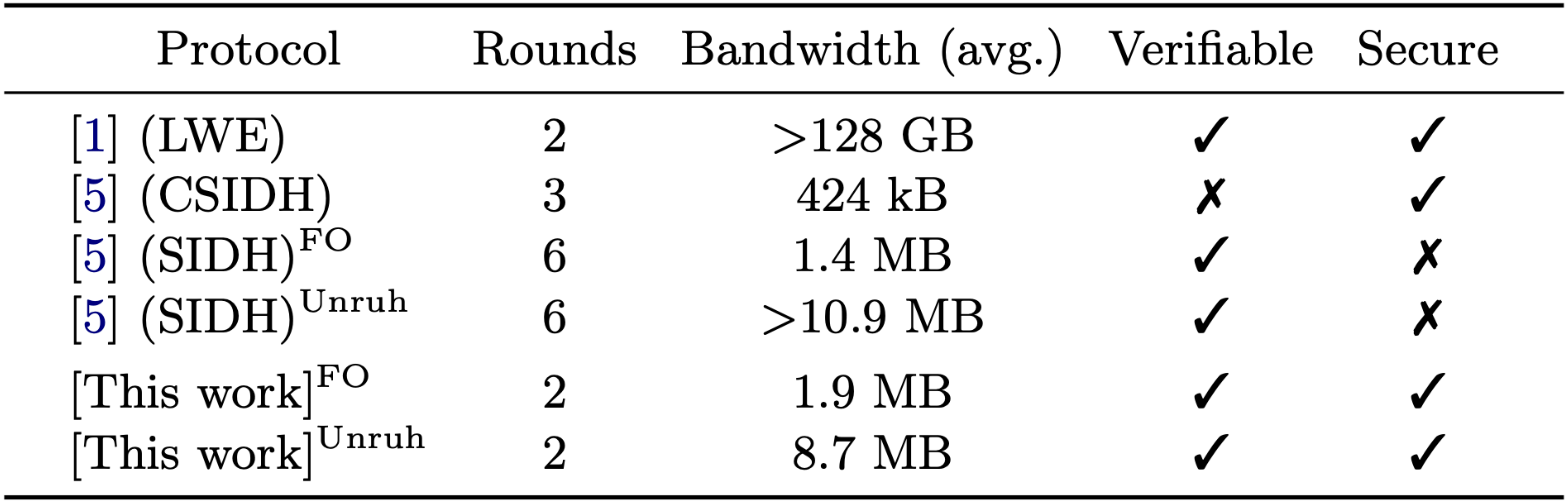
Putting it all together

- One-more unpredictability countermeasure more efficient than original new security assumption

- Integrated SIDH countermeasures novel proof of isogeny knowledge prime is still large

- New PoPI more efficient than original round optimal

Results



Protocol	Rounds	Bandwidth (avg.)	Verifiable	Secure
[1] (LWE)	2	>128 GB	✓	✓
[5] (CSIDH)	3	424 kB	✗	✓
[5] (SIDH)$^{\mathrm{FO}}$	6	1.4 MB	✓	✗
[5] (SIDH)$^{\mathrm{Unruh}}$	6	>10.9 MB	✓	✗
[This work]$^{\mathrm{FO}}$	2	1.9 MB	✓	✓
[This work]$^{\mathrm{Unruh}}$	2	8.7 MB	✓	✓