Collusion-Resistant Quantum Secure Key Leasing Beyond Decryption

Fuyuki Kitagawa
NTT

Ryo Nishimaki
NTT

Nikhil Pappu
Portland State UNIVERSITY

Secure Key Leasing (SKL)

Secure Key Leasing (SKL)

[AKN+23,APV23]

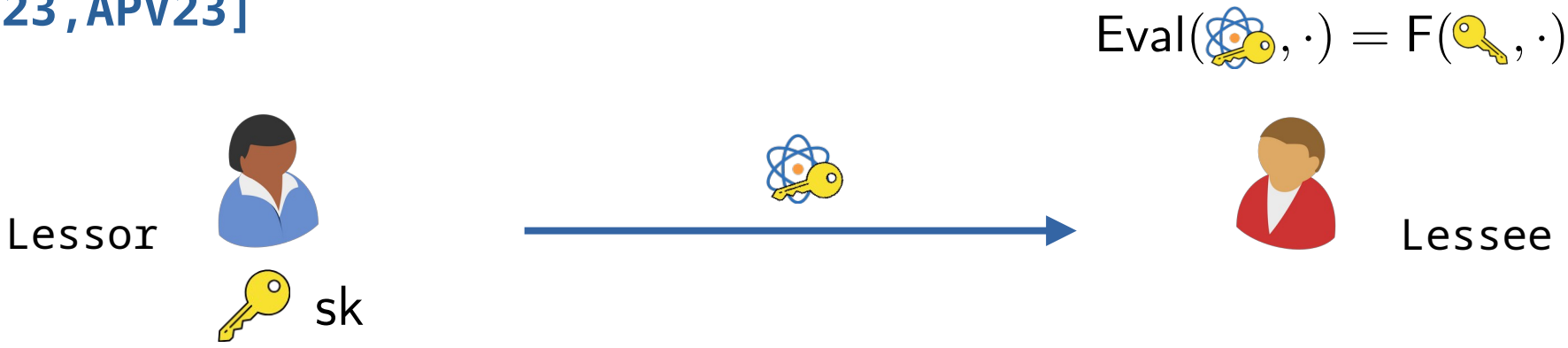
Lessor

sk

Lessee

Secure Key Leasing (SKL)

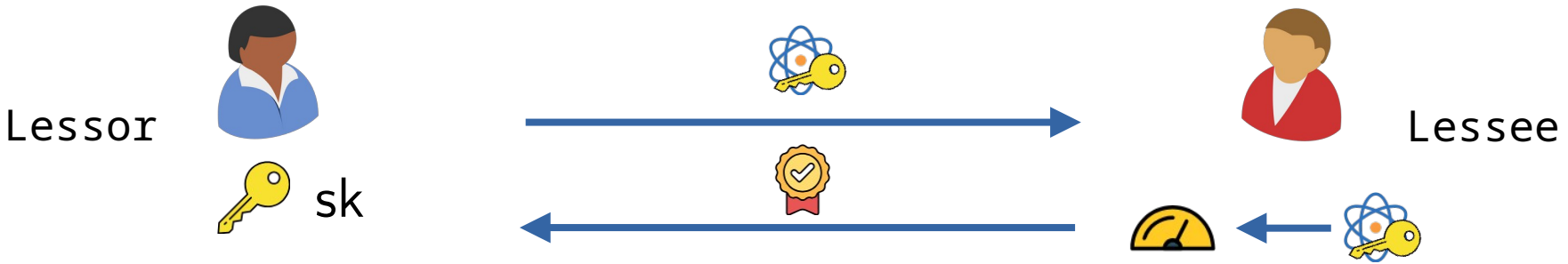
[AKN+23,APV23]



Secure Key Leasing (SKL)

[AKN+23,APV23]

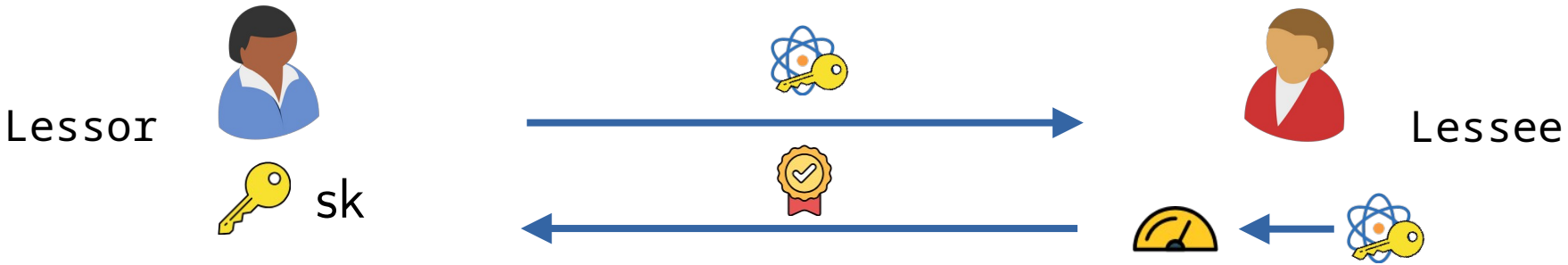
$\mathsf{Eval}(\,,\cdot) = \mathsf{F}(\,,\cdot)$

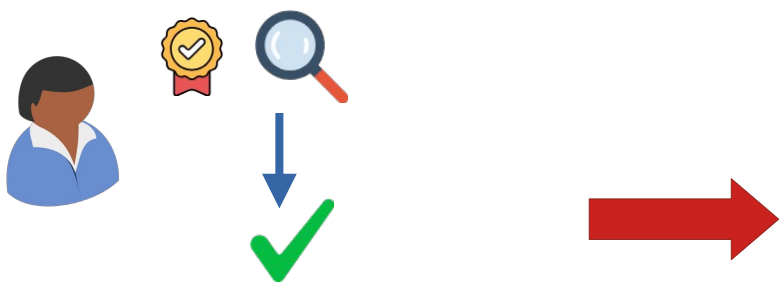


Secure Key Leasing (SKL)

[AKN+23,APV23]

$\mathsf{Eval}(\,,\cdot) = \mathsf{F}(\,,\cdot)$

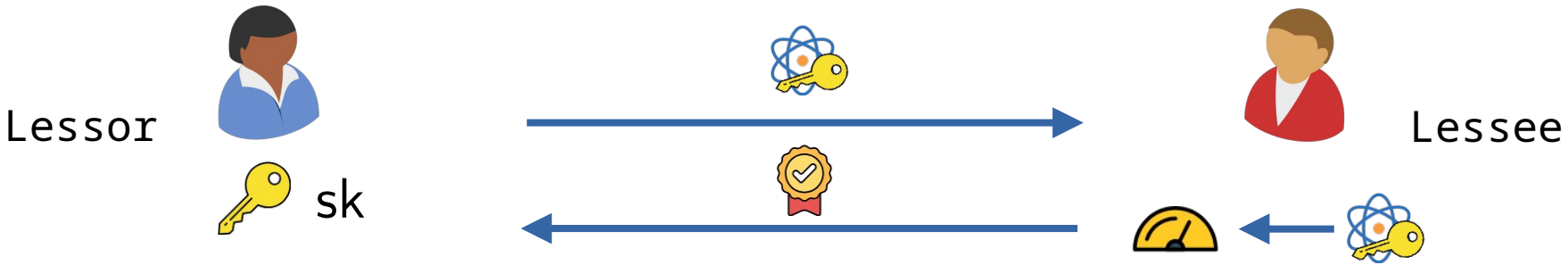


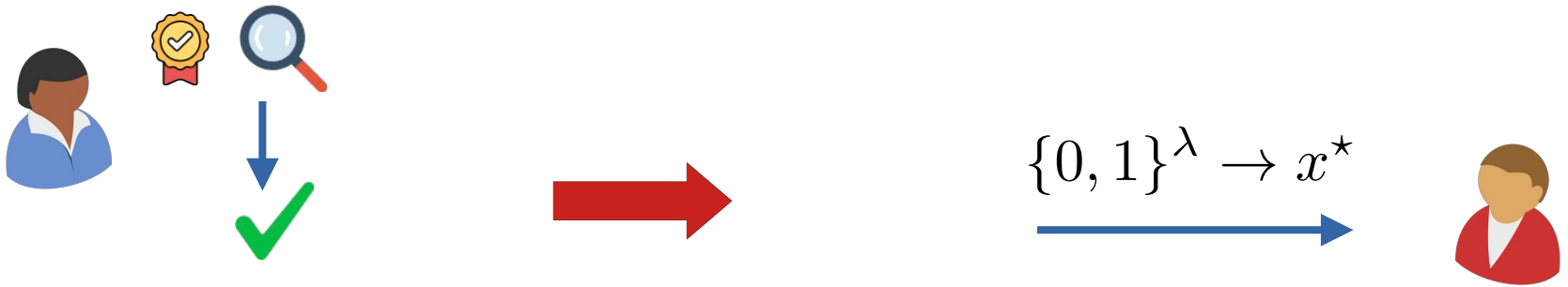


Secure Key Leasing (SKL)

[AKN+23,APV23]

$\mathsf{Eval}(\,,\cdot) = \mathsf{F}(\,,\cdot)$

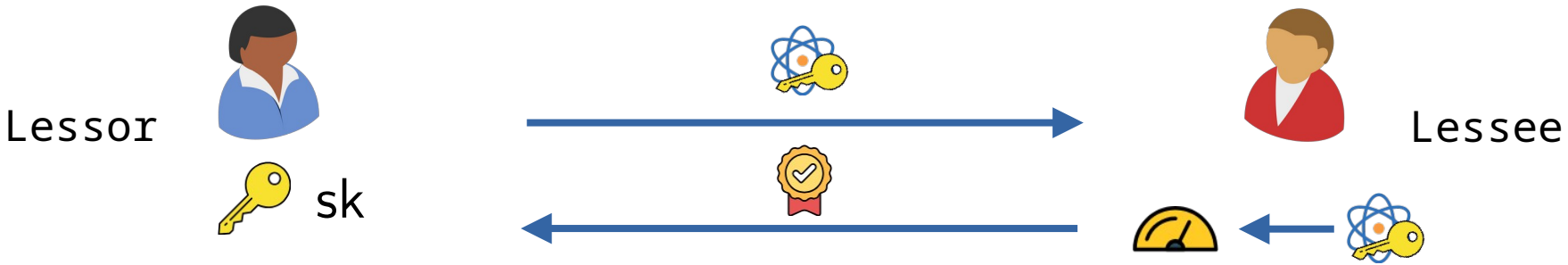


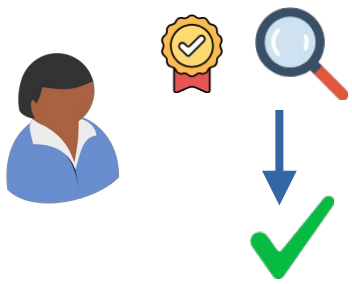


Secure Key Leasing (SKL)

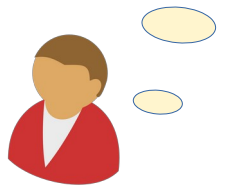
[AKN+23, APV23]

$\mathsf{Eval}(\cdot, \cdot) = \mathsf{F}(\cdot, \cdot)$





$\{0,1\}^\lambda \to x^\star$

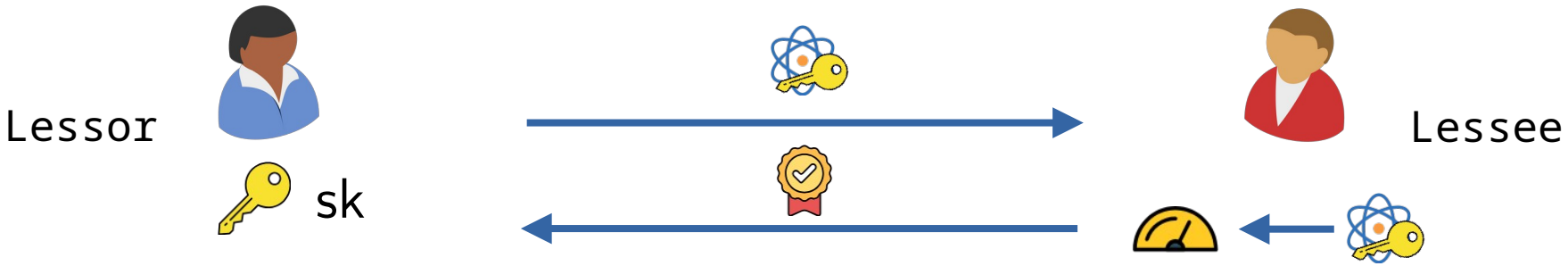


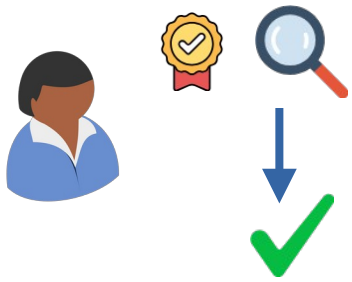
$\mathsf{F}(\mathsf{sk}, x^\star)$??

Secure Key Leasing (SKL)

[AKN+23,APV23]

$\mathsf{Eval}(\cdot, \cdot) = \mathsf{F}(\cdot, \cdot)$



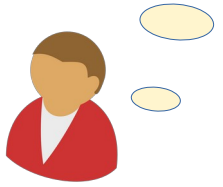


No Cloning Principle

Impossible Classically!

$\mathsf{F}(\mathsf{sk}, x^\star)$??

$\{0,1\}^\lambda \to x^\star$

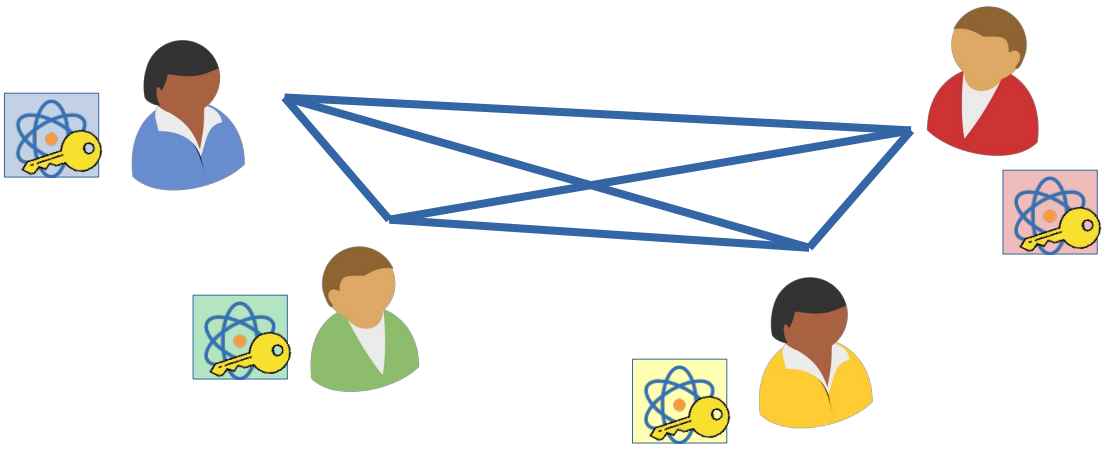


PRF-SKL Application

PRF-SKL Application

Dynamic Distributed Protocols

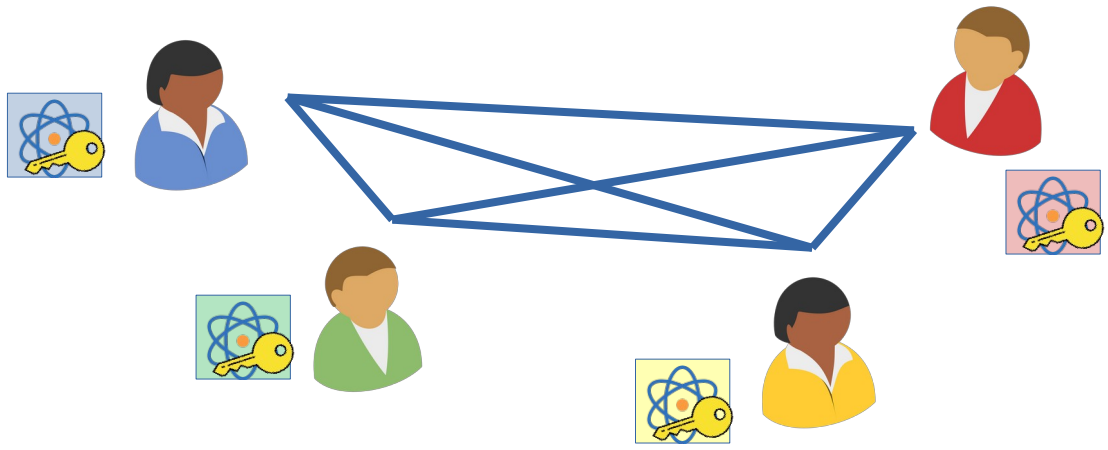
Eg: In **MPC**

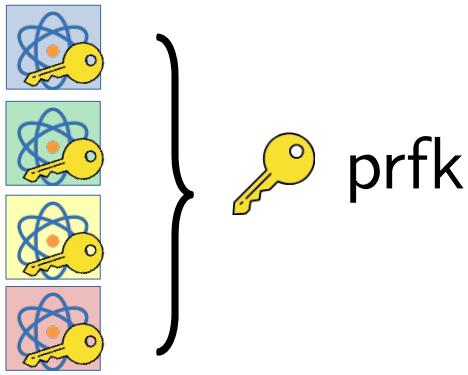


PRF-SKL Application

Dynamic Distributed Protocols

Eg: In **MPC**

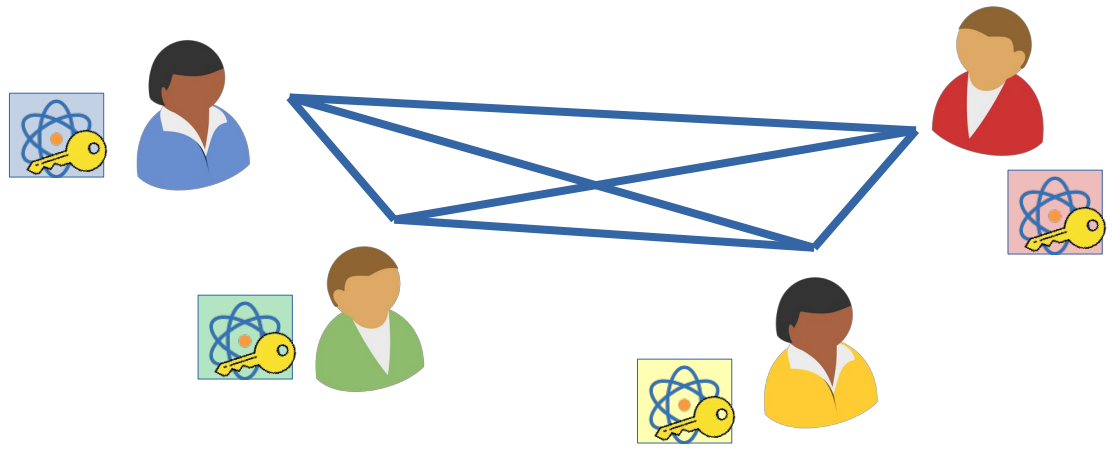




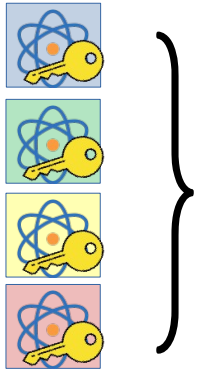
PRF-SKL Application

Dynamic Distributed Protocols

Eg: In **MPC**



Generate **correlated** randomness

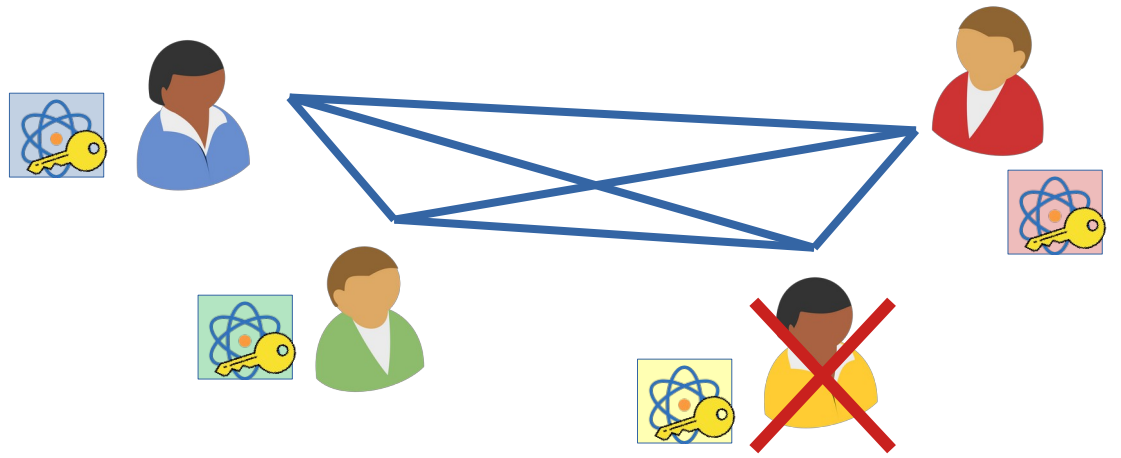


prfk

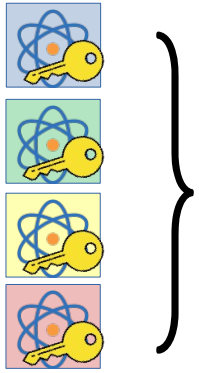
PRF-SKL Application

Dynamic Distributed Protocols

Eg: In **MPC**



Generate **correlated** randomness

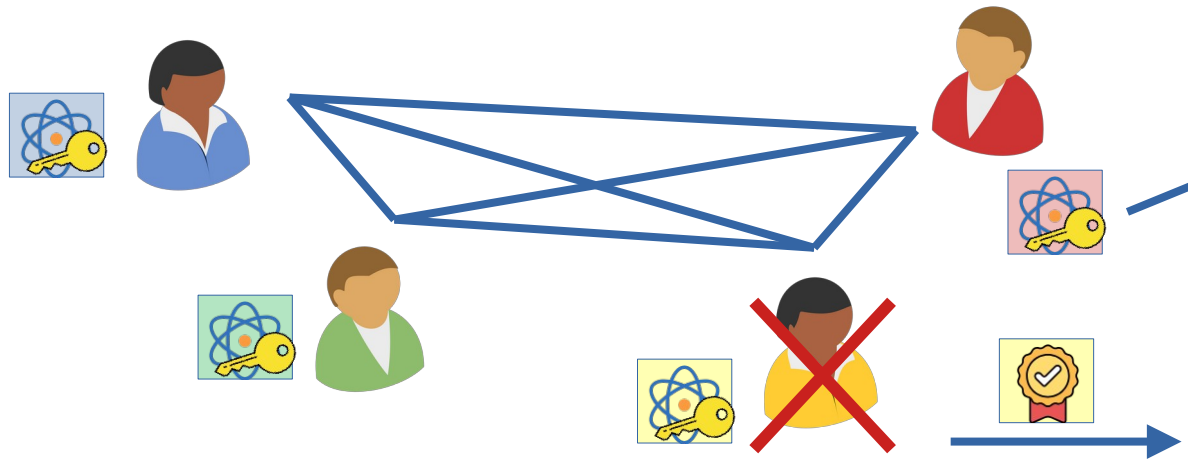


prfk

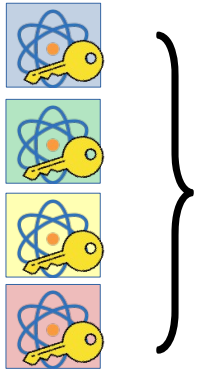
PRF-SKL Application

Dynamic Distributed Protocols

Eg: In **MPC**



Generate **correlated** randomness

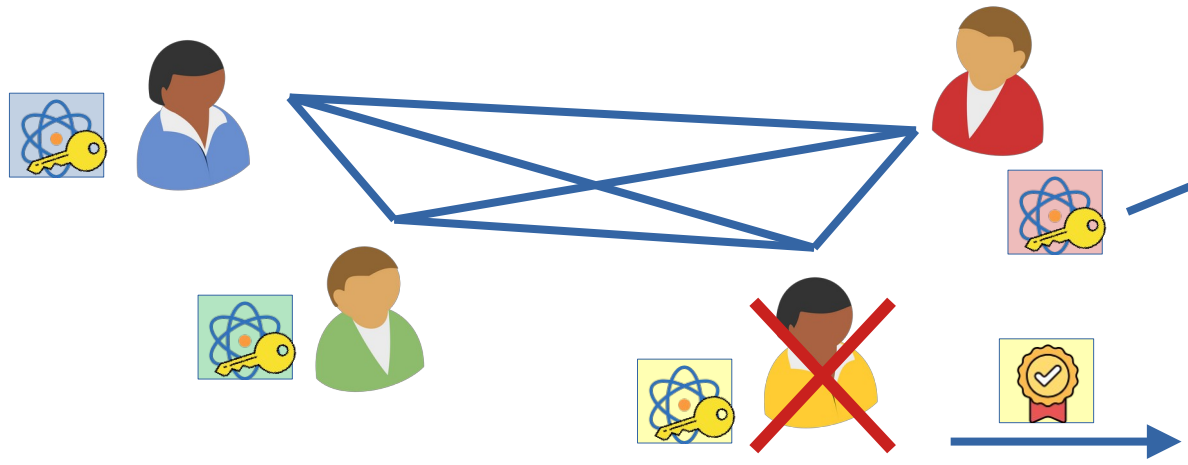


prfk

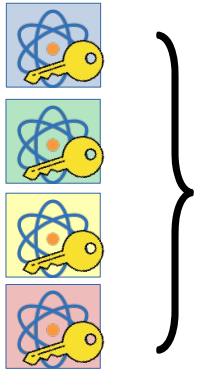
PRF-SKL Application

Dynamic Distributed Protocols

Eg: In **MPC**



Generate **correlated** randomness



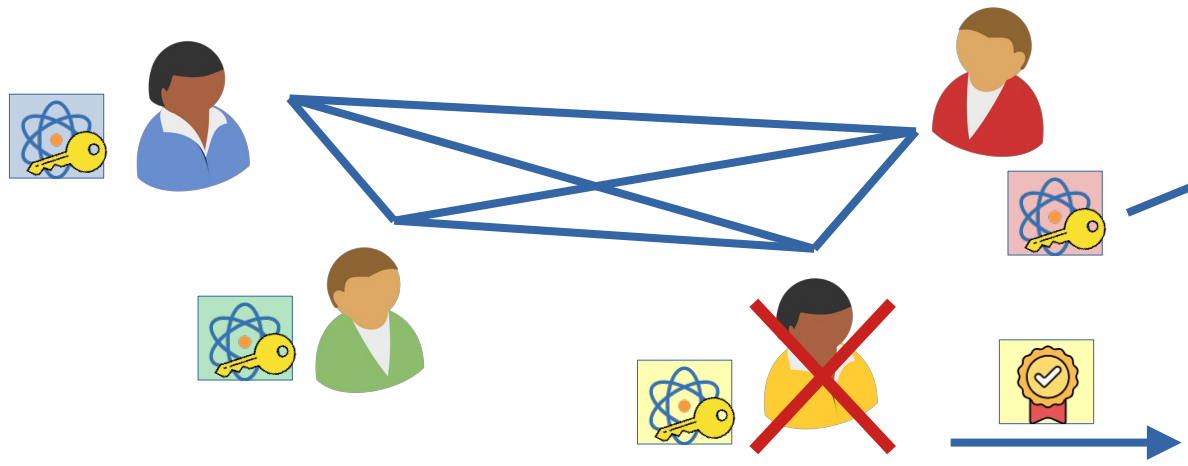
prfk

- **No need** to **update** setup for future iterations.

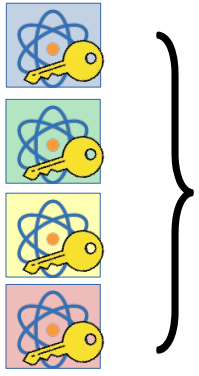
PRF-SKL Application

Dynamic Distributed Protocols

Eg: In **MPC**



Generate **correlated** randomness



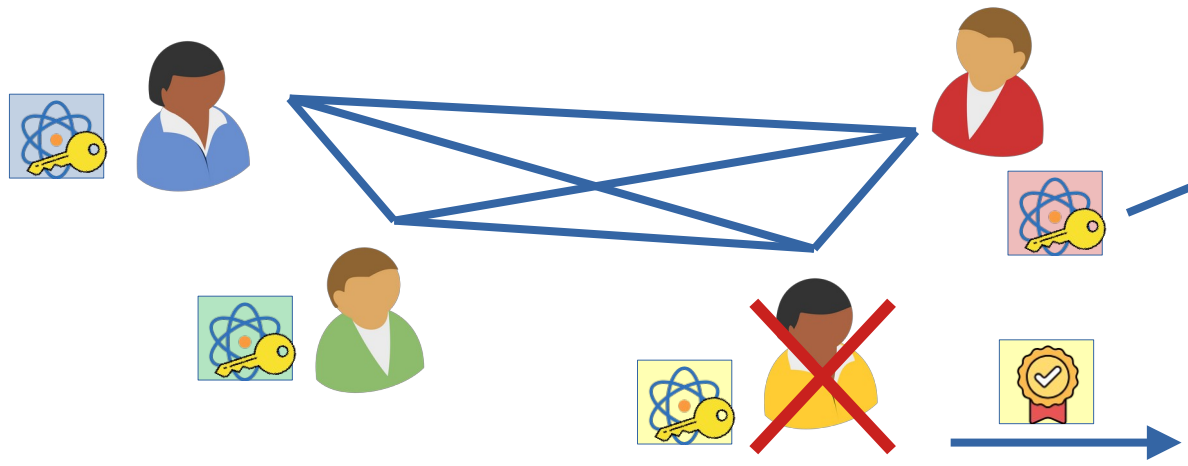
prfk

- **No need** to **update** setup for future iterations.
- Party can **delete** and leave.

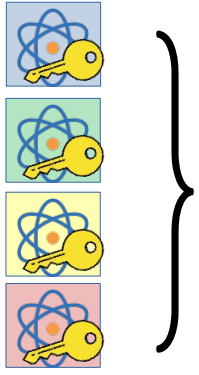
PRF-SKL Application

Dynamic Distributed Protocols

Eg: In **MPC**



Generate **correlated** randomness



prfk

- **No need** to **update** setup for future iterations.
- Party can **delete** and leave.
- ⚠ **Collusions** arise naturally!

Prior Work

Prior Work

Work	Primitives	Collusions	Channel	Assumption
[AKN+23]	PKE, ABE, PKFE	None, Bound, Unbound	Quantum	PKE, PKFE
[APV23,AHH24]	PKE, PRF	None	Quantum	LWE, LWE
[KMY25]	PKE, PRF, Sig	None	Quantum	PKE, OWF, SIS
[CGJL25]	PKE, FHE	None	Classical	LWE
[BGK+24]	Diff-Inp Func, PKFE	Unbound	Quantum	sub-exp iO
[KNP25]	PKE, ABE	Unbound	Quantum	LWE
[ÇG24]	PKE, PRF, Sig	Unbound	Classical [CHV23]	sub-exp iO

Prior Work

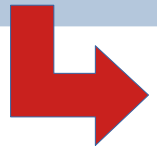
Work	Primitives	Collusions	Channel	Assumption
[AKN+23]	PKE, ABE, PKFE	None, Bound, Unbound	Quantum	PKE, PKFE
[APV23,AHH24]	PKE, PRF	None	Quantum	LWE, LWE
[KMY25]	PKE, PRF, Sig	None	Quantum	PKE, OWF, SIS
[CGJL25]	PKE, FHE	None	Classical	LWE
[BGK+24]	Diff-Inp Func, PKFE	Unbound	Quantum	sub-exp iO
[KNP25]	PKE, ABE	Unbound	Quantum	LWE
[ÇG24]	PKE, PRF, Sig	Unbound	Classical [CHV23]	sub-exp iO

Standard Assumption

Implies iO

Prior Work

Work	Primitives	Collusions	Channel	Assumption
[AKN+23]	PKE, ABE, PKFE	None, Bound, Unbound	Quantum	PKE, PKFE
[APV23,AHH24]	PKE, PRF	None	Quantum	LWE, LWE
[KMY25]	PKE, PRF, Sig	None	Quantum	PKE, OWF, SIS
[CGJL25]	PKE, FHE	None	Classical	LWE
[BGK+24]	Diff-Inp Func, PKFE	Unbound	Quantum	sub-exp iO
[KNP25]	PKE, ABE	Unbound	Quantum	LWE
[ÇG24]	PKE, PRF, Sig	Unbound	Classical [CHV23]	sub-exp iO



Known **Copy Protection** schemes imply SKL!

Standard Assumption

Implies iO

Motivation

Motivation

- **Collusion-Resistance** (Even Bounded) for **PRFs** and **Signatures**?

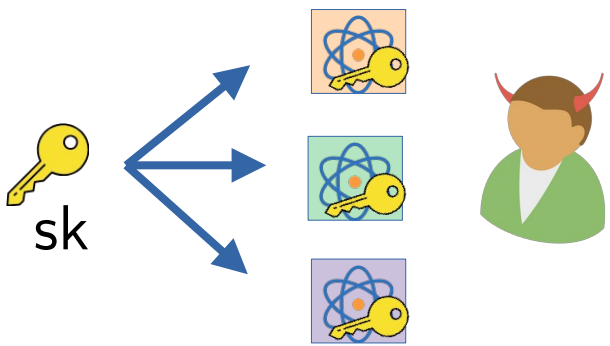
Motivation

- **Collusion-Resistance** (Even Bounded) for **PRFs** and **Signatures**?
- SKL **Definitions** and **Constructions** (w/o iO) are **case-by-case**.

Motivation

- **Collusion-Resistance** (Even Bounded) for **PRFs** and **Signatures**?
- SKL **Definitions** and **Constructions** (w/o iO) are **case-by-case**.

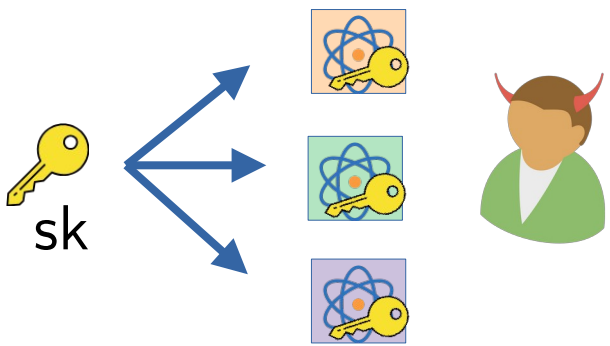
Challenging Setting



Motivation

- **Collusion-Resistance** (Even Bounded) for **PRFs** and **Signatures**?
- SKL **Definitions** and **Constructions** (w/o iO) are **case-by-case**.

Challenging Setting

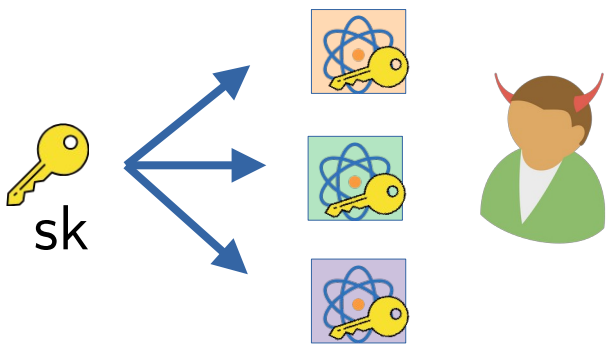


⚠ **Prone to Attacks. Eg:**

Motivation

- **Collusion-Resistance** (Even Bounded) for **PRFs** and **Signatures**?
- SKL **Definitions** and **Constructions** (w/o iO) are **case-by-case**.

Challenging Setting



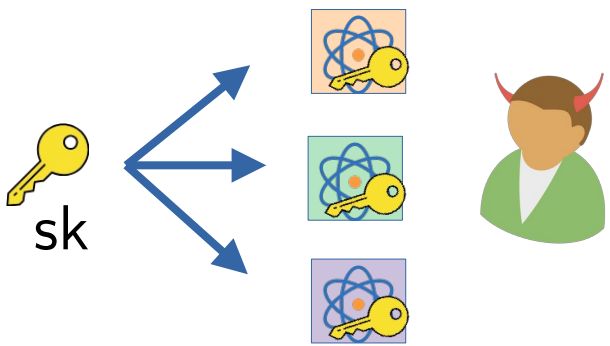
Prone to Attacks. Eg:

1) Correlate keys before deletion
2) Learn crucial classical info
3) Leave states undisturbed

Motivation

- **Collusion-Resistance** (Even Bounded) for **PRFs** and **Signatures**?
- SKL **Definitions** and **Constructions** (w/o iO) are **case-by-case**.

Challenging Setting



Prone to Attacks. Eg:

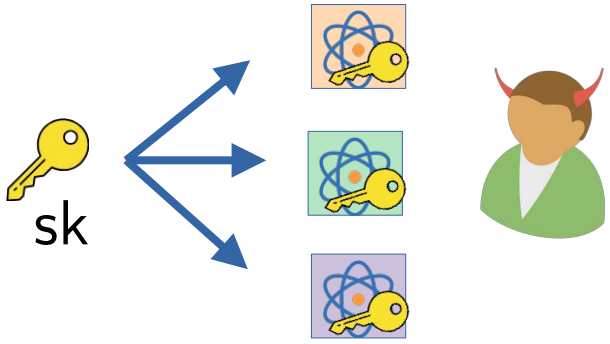
1) Correlate keys before deletion
2) Learn crucial classical info
3) Leave states undisturbed

- **Provable security** is even **harder** to argue.

Motivation

- **Collusion-Resistance** (Even Bounded) for **PRFs** and **Signatures**?
- SKL **Definitions** and **Constructions** (w/o iO) are **case-by-case**.

Challenging Setting



Prone to Attacks. Eg:

1) Correlate keys before deletion
2) Learn crucial classical info
3) Leave states undisturbed

- **Provable security** is even **harder** to argue.
- All **prior proofs** for PRFs and signatures **breakdown** even with **bounded** collusions.

Our Contributions I

Our Contributions I

New Abstraction: Multi-Level Traitor Tracing **MLTT**

Our Contributions I

New Abstraction: Multi-Level Traitor Tracing **MLTT**

X-MLTT → **X-SKL**

- **X** can be **PRF**, **PKE**, **Sig** …

Our Contributions I

New Abstraction: Multi-Level Traitor Tracing **MLTT**

X-MLTT

Y-Collusion Resistant

X-SKL

Y-Collusion Resistant

- **X** can be **PRF**, **PKE**, **Sig** …
- **Y** is **Bounded/Unbounded**

Our Contributions I

New Abstraction: Multi-Level Traitor Tracing **MLTT**

X-MLTT

Y-Collusion Resistant

X-SKL

Y-Collusion Resistant

- **X** can be **PRF**, **PKE**, **Sig** …
- **Y** is **Bounded/Unbounded**
- No Other Assumption!

Our Contributions I

New Abstraction: Multi-Level Traitor Tracing **MLTT**

X-MLTT

Y-Collusion Resistant

X-SKL

Y-Collusion Resistant

- **X** can be **PRF**, **PKE**, **Sig** …
- **Y** is **Bounded/Unbounded**
- No Other Assumption!

First **Generic** Approach **w/o iO**. Concurrent w/ **[KLYY25].**

Our Contributions I

New Abstraction: Multi-Level Traitor Tracing **MLTT**

X-MLTT → **X-SKL**

Y-Collusion Resistant → **Y-Collusion Resistant**

- **X** can be **PRF**, **PKE**, **Sig** …
- **Y** is **Bounded/Unbounded**
- No Other Assumption!

First **Generic** Approach **w/o iO**. Concurrent w/ **[KLYY25].**

LWE → **PRF-MLTT** → **PRF-SKL**

Bounded-CR **Bounded-CR**

Our Contributions II

Our Contributions II

Our Contributions II

Our Contributions II

- Allows to securely **lease** a secret **signing key**.

Our Contributions II

- Allows to securely **lease** a secret **signing key**.
- **Direct Approach** w/o MLTT Compiler

Our Contributions II

- Allows to securely **lease** a secret **signing key**.
- **Direct Approach** w/o MLTT Compiler
- Also works for **Copy Protection**.

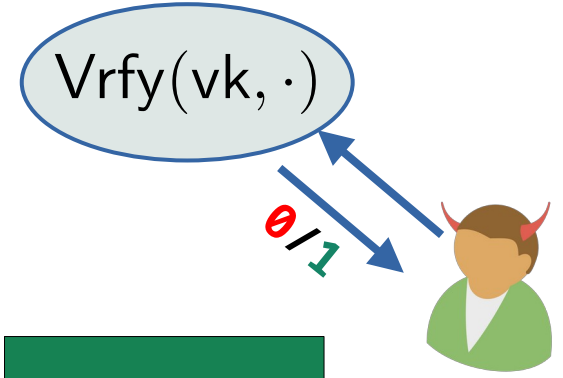
Our Contributions II

- Allows to securely **lease** a secret **signing key**.
- **Direct Approach** w/o MLTT Compiler
- Also works for **Copy Protection**.

Vrfy-Oracle Resilient

Our Contributions II

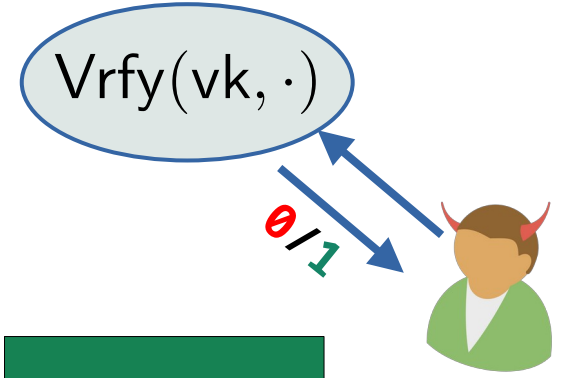
- Allows to securely **lease** a secret **signing key**.
- **Direct Approach** w/o MLTT Compiler
- Also works for **Copy Protection**.



Vrfy-Oracle Resilient

Our Contributions II

- Allows to securely **lease** a secret **signing key**.
- **Direct Approach** w/o MLTT Compiler
- Also works for **Copy Protection**.

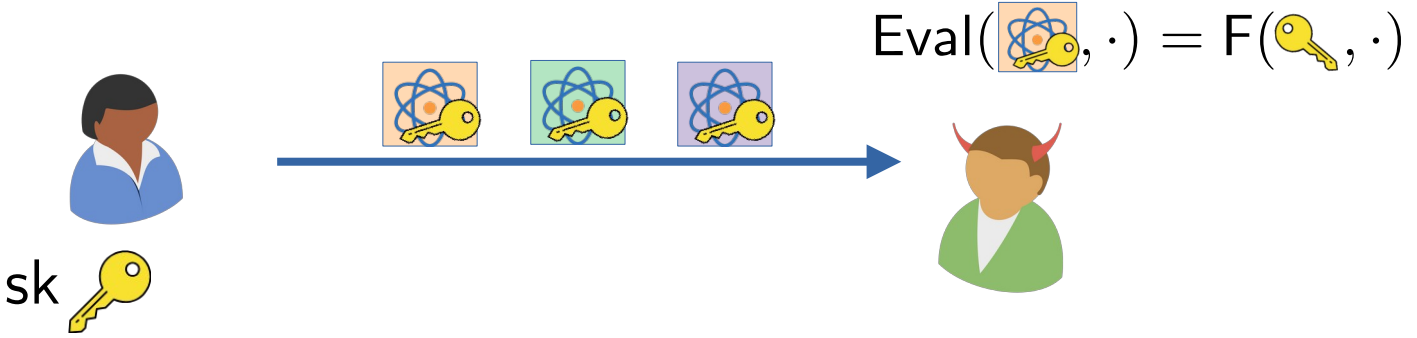


Vrfy-Oracle Resilient

Defining PRF-SKL

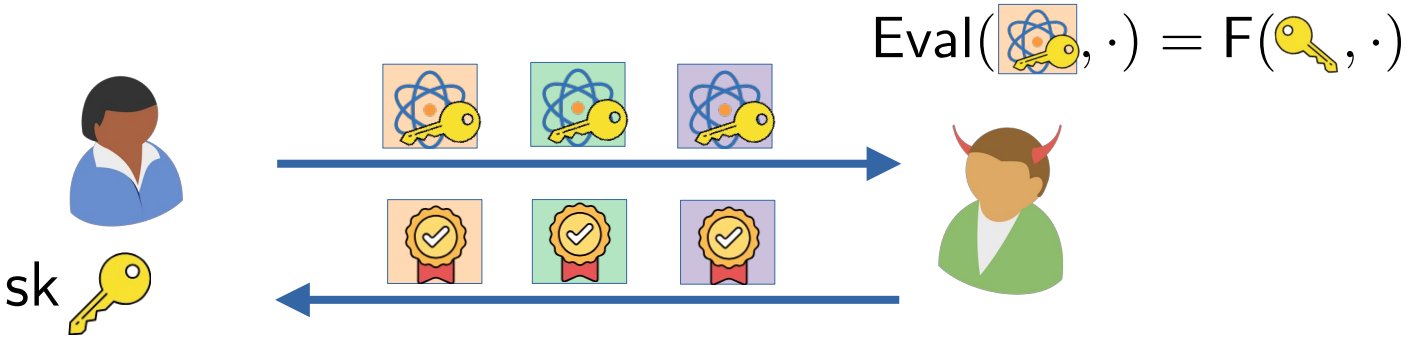
Defining PRF-SKL

PRF-SKL



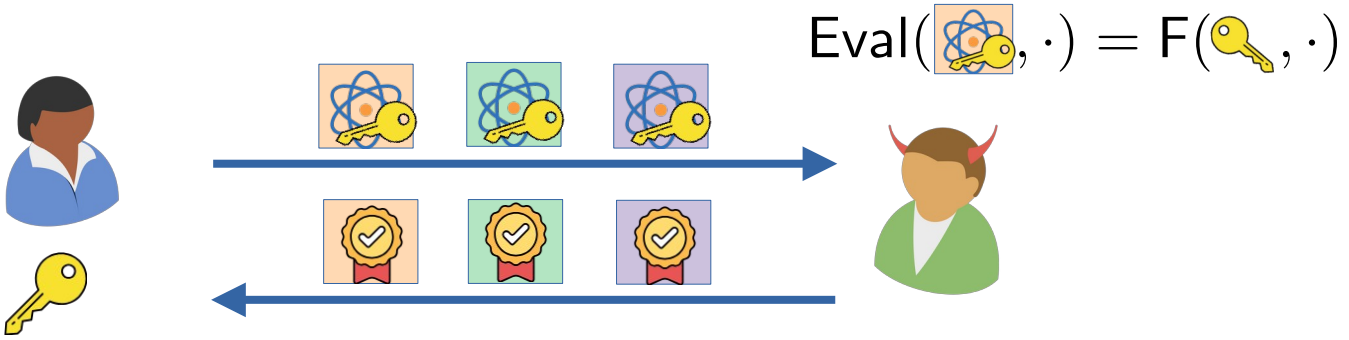
Defining PRF-SKL

PRF-SKL

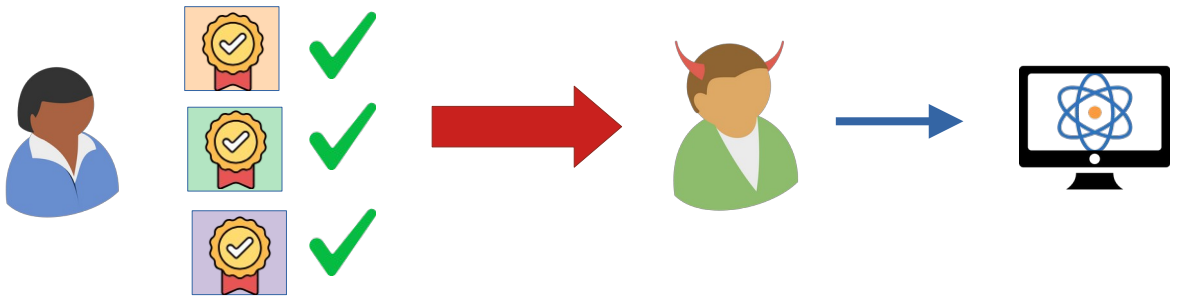


Defining PRF-SKL

PRF-SKL

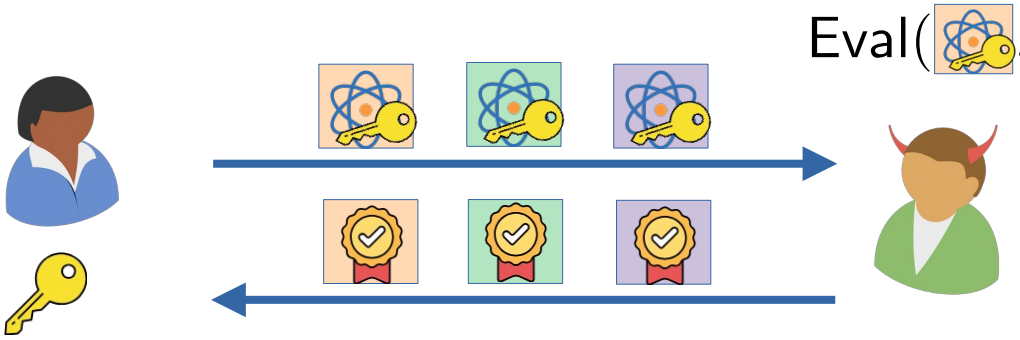


Event `VrfyPass`



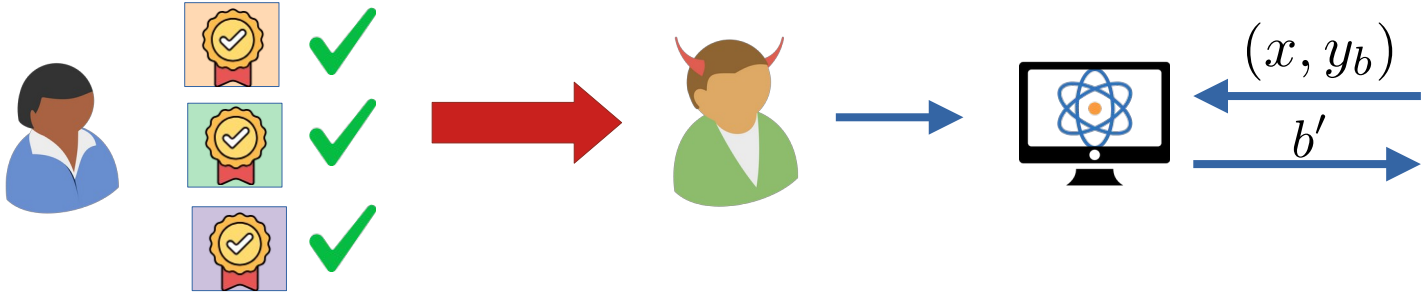
Defining PRF-SKL

PRF-SKL



$\mathsf{Eval}(\,\cdot\,) = \mathsf{F}(\,\cdot\,)$

Event VrfyPass



$(x, y_1, b) \leftarrow \mathcal{U}_n \times \mathcal{U}_m \times \mathcal{U}_1$

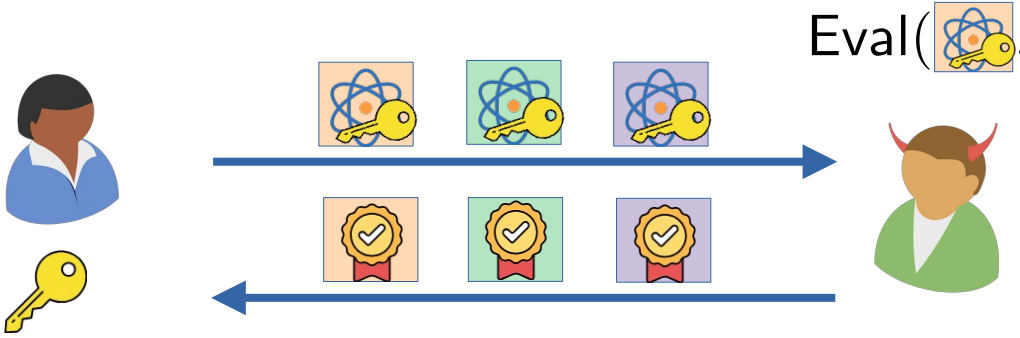
$y_0 = \mathsf{F}(\mathsf{sk}, x)$

(x, y_b)

b'

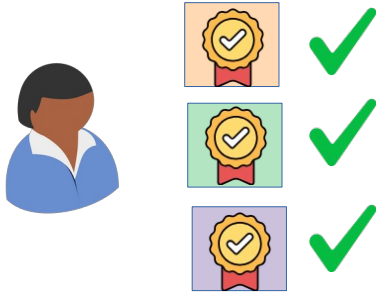
Defining PRF-SKL

PRF-SKL



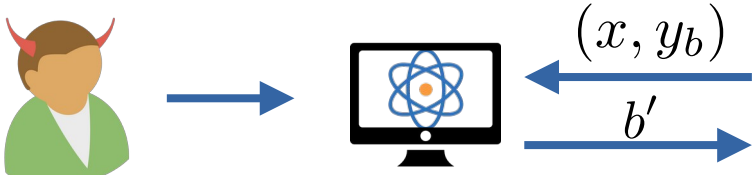
$\mathsf{Eval}(\,\cdot\,,\cdot) = \mathsf{F}(\,\cdot\,,\cdot)$

Event VrfyPass



$(x, y_1, b) \leftarrow \mathcal{U}_n \times \mathcal{U}_m \times \mathcal{U}_1$

$y_0 = \mathsf{F}(\mathsf{sk}, x)$

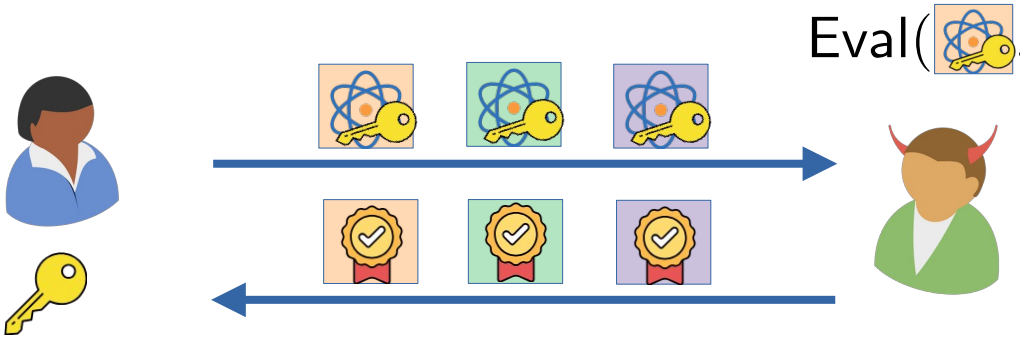


Event GoodProg

if $\Pr[b' = b] > 0.5 + 1/\mathsf{poly}(\lambda)$

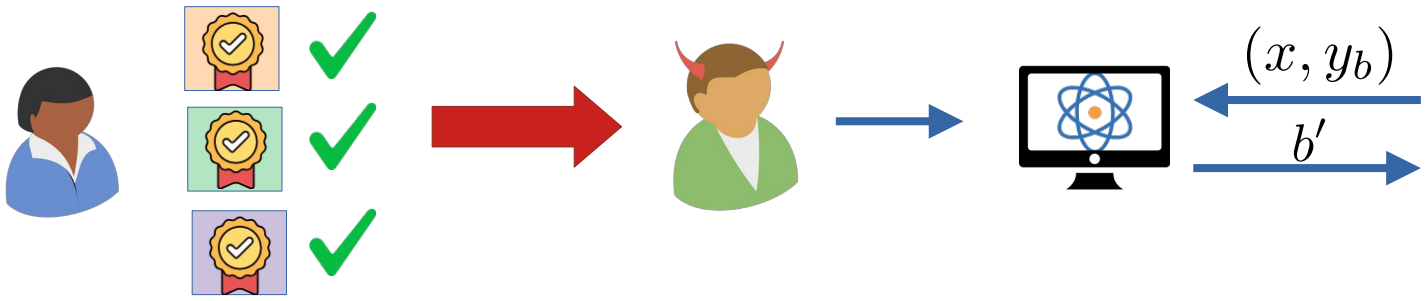
Defining PRF-SKL

PRF-SKL



$\mathsf{Eval}(\,\cdot\,,\cdot) = \mathsf{F}(\,\cdot\,,\cdot)$

Event VrfyPass



$(x, y_1, b) \leftarrow \mathcal{U}_n \times \mathcal{U}_m \times \mathcal{U}_1$

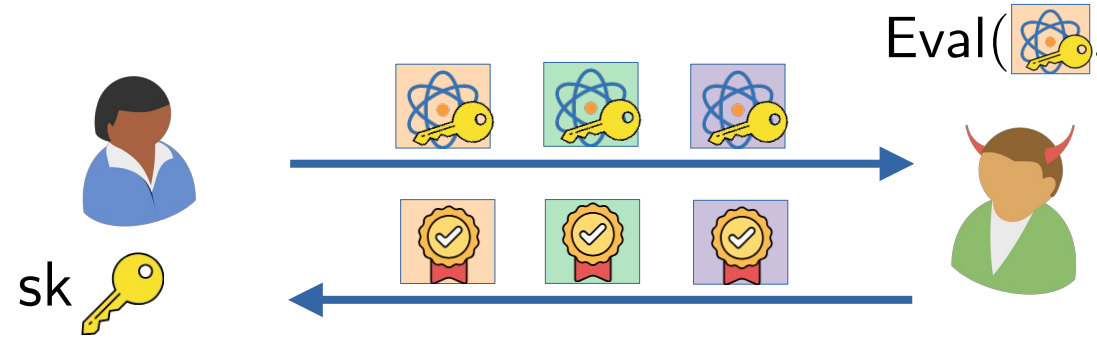
$y_0 = \mathsf{F}(\mathsf{sk}, x)$

Event GoodProg

$$\text{if } \underbrace{\Pr[b' = b]}_{\text{Estimation using [Zha20]}} > 0.5 + 1/\mathsf{poly}(\lambda)$$

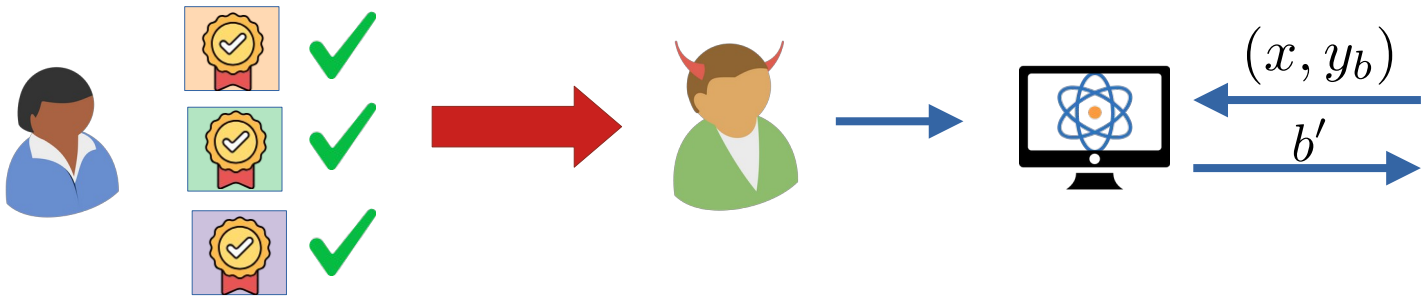
Defining PRF-SKL

PRF-SKL



$\mathsf{Eval}(\,\cdot\,,\cdot) = \mathsf{F}(\,\cdot\,,\cdot)$

Event VrfyPass



$(x, y_1, b) \leftarrow \mathcal{U}_n \times \mathcal{U}_m \times \mathcal{U}_1$

$y_0 = \mathsf{F}(\mathsf{sk}, x)$

(x, y_b)

b'

Event GoodProg

$$\text{if } \Pr[\underbrace{b' = b}] > 0.5 + 1/\mathsf{poly}(\lambda)$$

Estimation using [Zha20]

$$\Pr[\textbf{VrfyPass} \wedge \textbf{GoodProg}] = \mathsf{negl}(\lambda)$$

Traitor Tracing

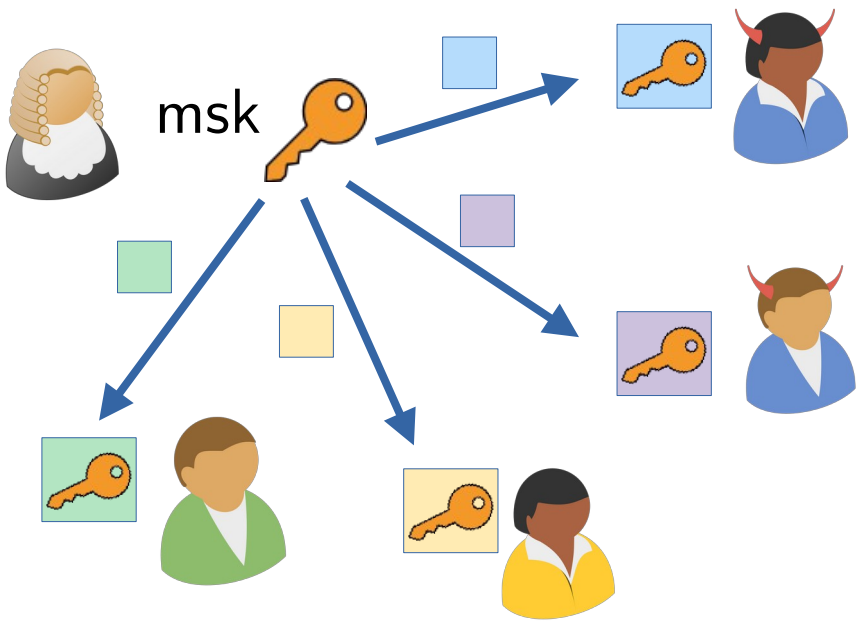
Traitor Tracing

Traitor Tracing

msk

Traitor Tracing

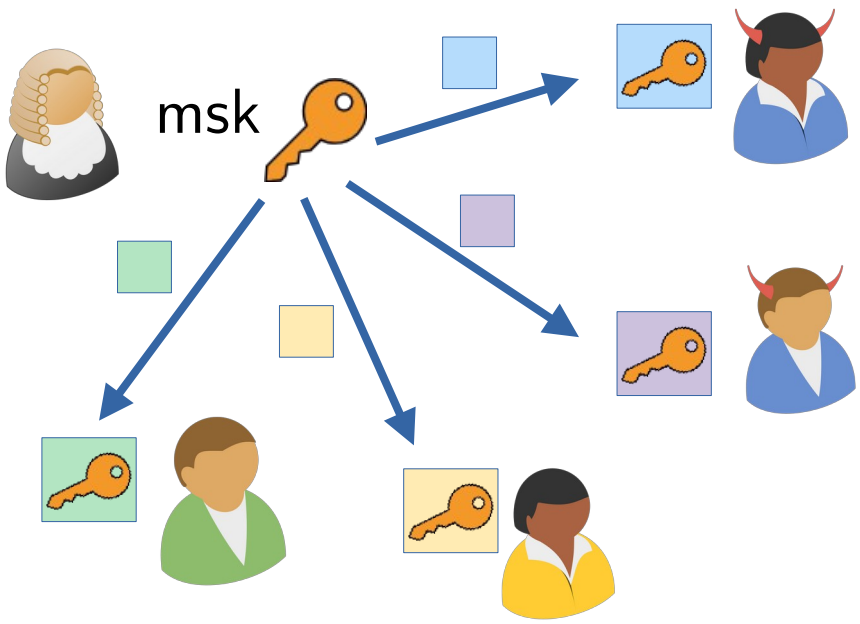
Traitor Tracing



Traitor Tracing

Traitor Tracing

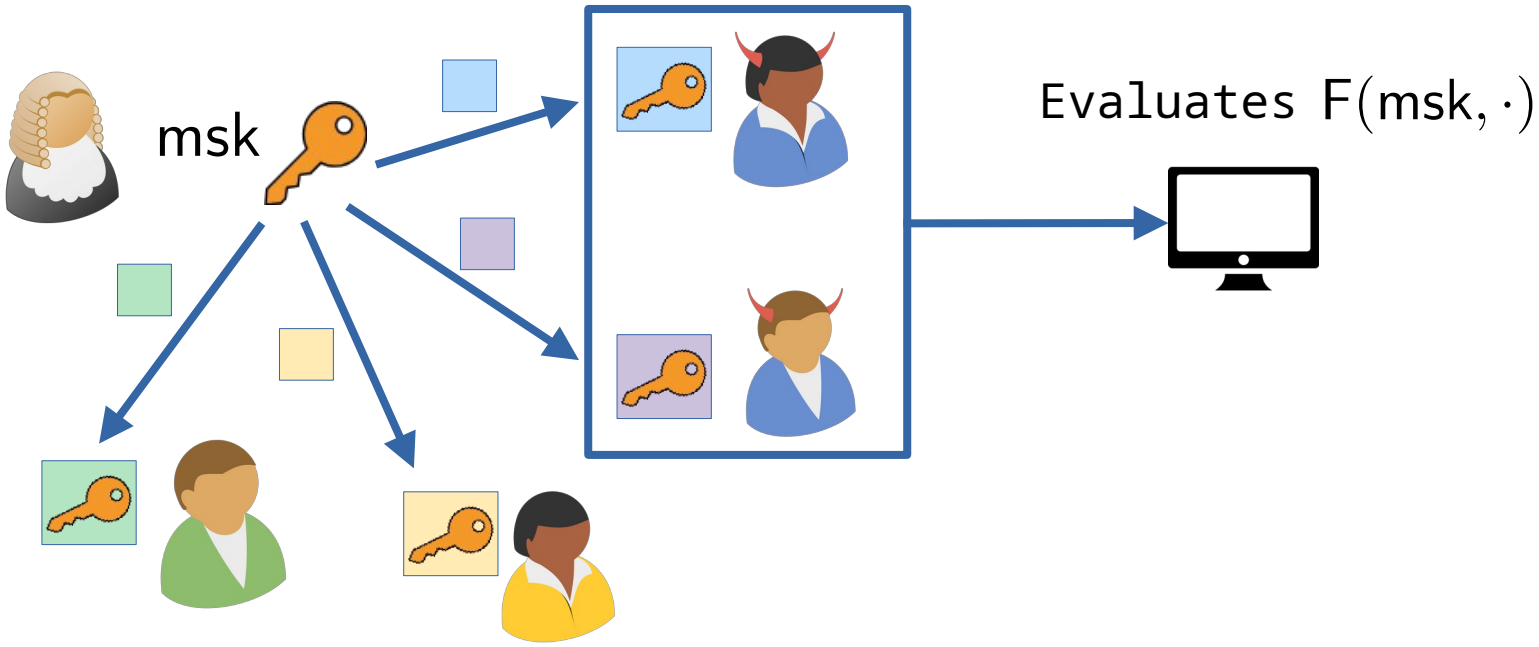
$$\mathsf{Eval}(\boxed{\text{🔑}}, x) = \ldots = \mathsf{Eval}(\boxed{\text{🔑}}, x) = \mathsf{F}(\text{🔑}, x)$$



Traitor Tracing

Traitor Tracing

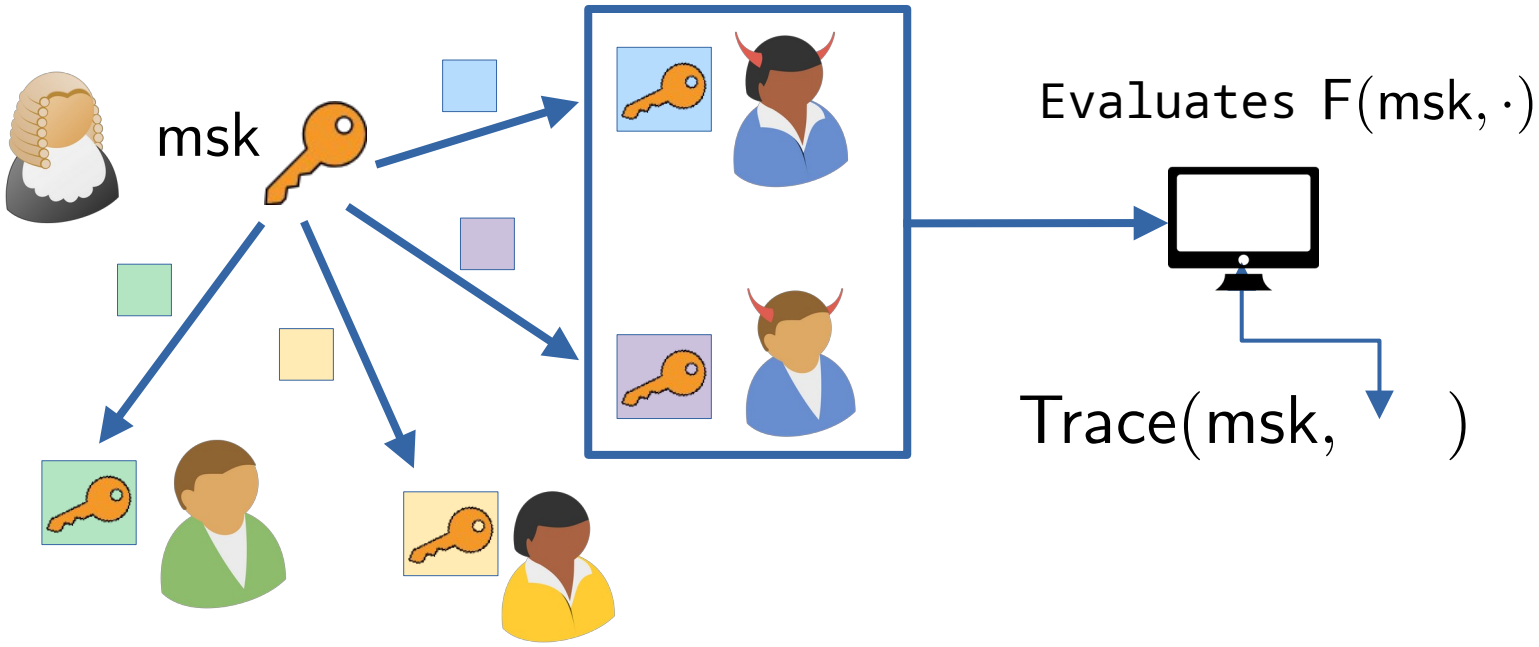
$$\mathsf{Eval}(\boxed{\text{🔑}}, x) = \ldots = \mathsf{Eval}(\boxed{\text{🔑}}, x) = \mathsf{F}(\text{🔑}, x)$$



Traitor Tracing

Traitor Tracing

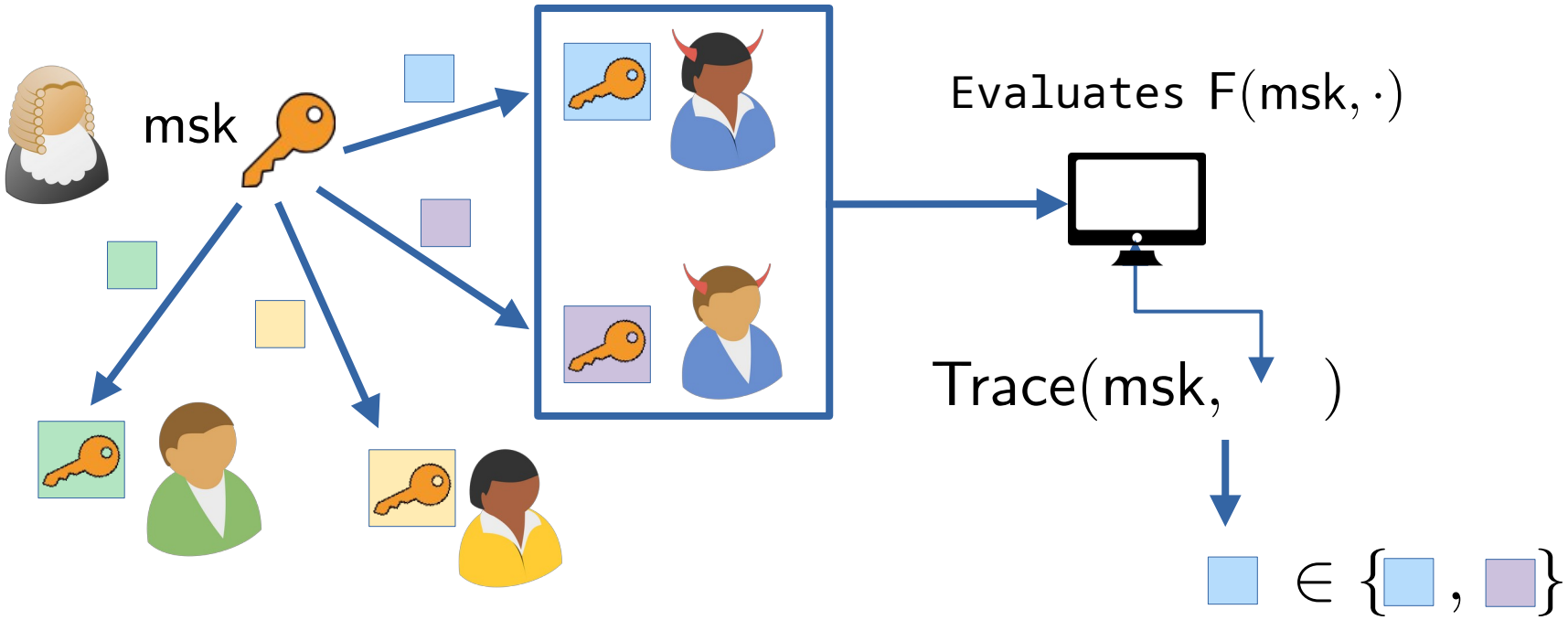
$$\mathsf{Eval}(\boxed{\text{🔑}}, x) = \ldots = \mathsf{Eval}(\boxed{\text{🔑}}, x) = \mathsf{F}(\text{🔑}, x)$$



Traitor Tracing

Traitor Tracing

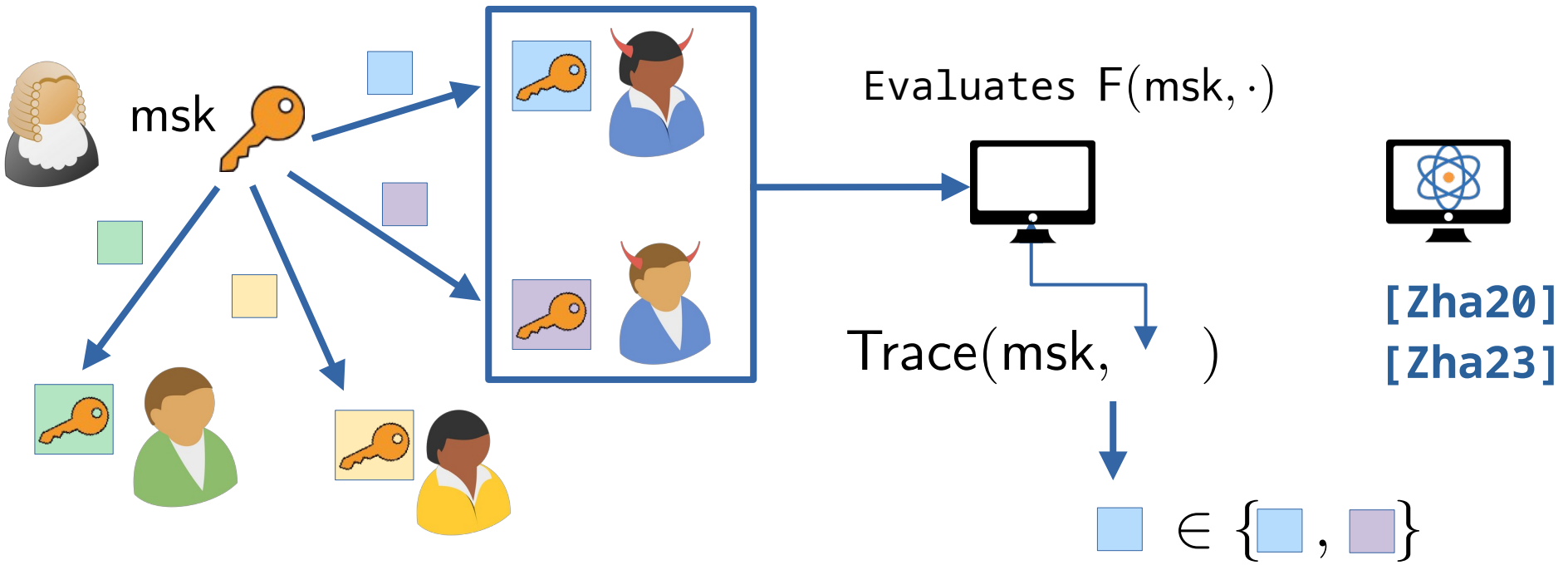
$$\mathsf{Eval}(\,\square\,, x) = \ldots = \mathsf{Eval}(\,\square\,, x) = \mathsf{F}(\,\square\,, x)$$



Traitor Tracing

Traitor Tracing

$\mathsf{Eval}(\blacksquare, x) = \ldots = \mathsf{Eval}(\blacksquare, x) = \mathsf{F}(\blacksquare, x)$



Multi-Level Traitor Tracing

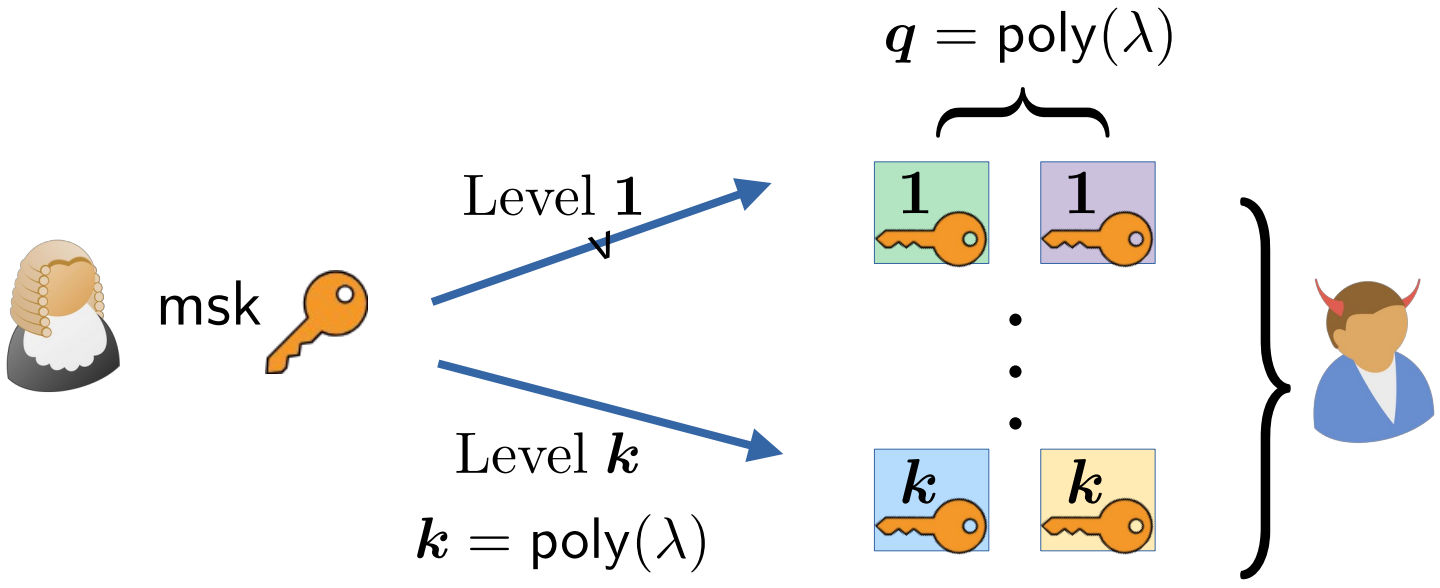
Multi-Level Traitor Tracing

MLTT

msk 

Multi-Level Traitor Tracing

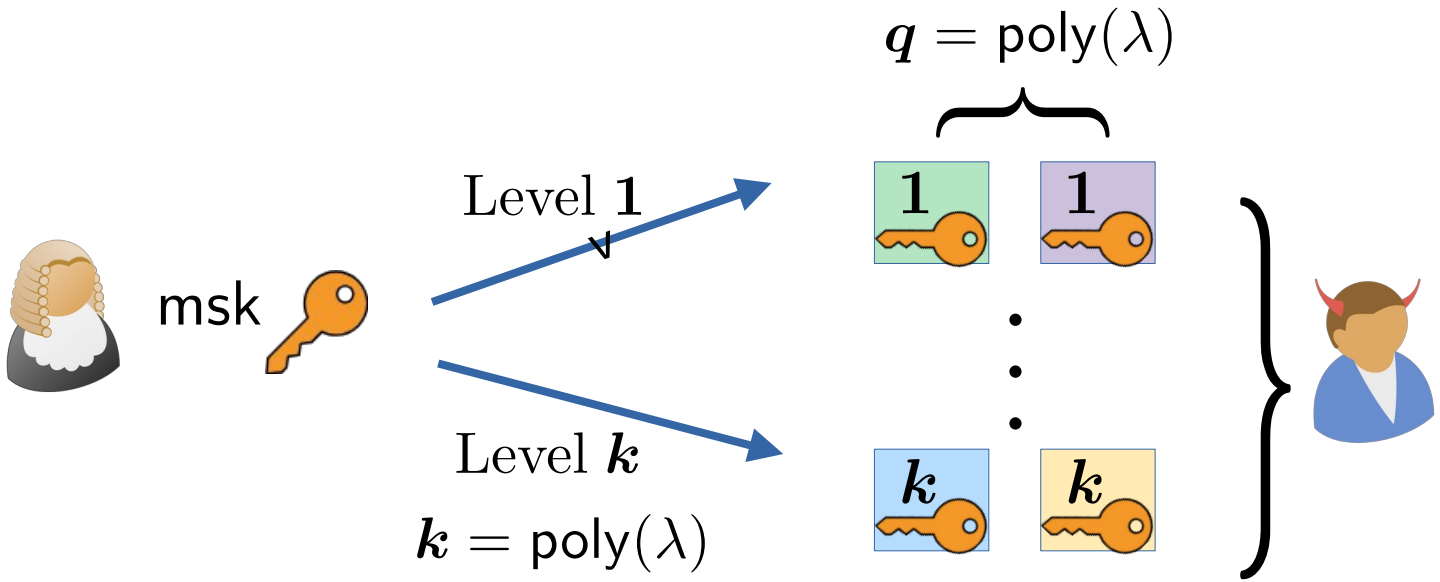
MLTT



Multi-Level Traitor Tracing

MLTT

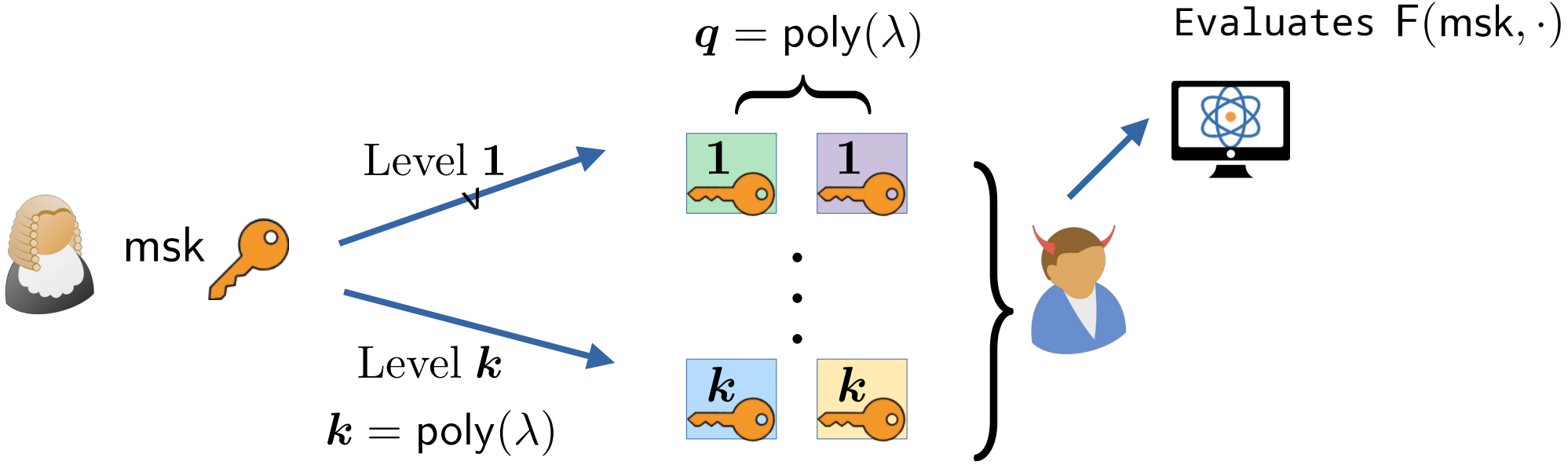
$$\mathsf{Eval}(\boxed{1}, \ldots, \boxed{k}, x) = \mathsf{F}(\,\cdot\,, x)$$



Multi-Level Traitor Tracing

MLTT

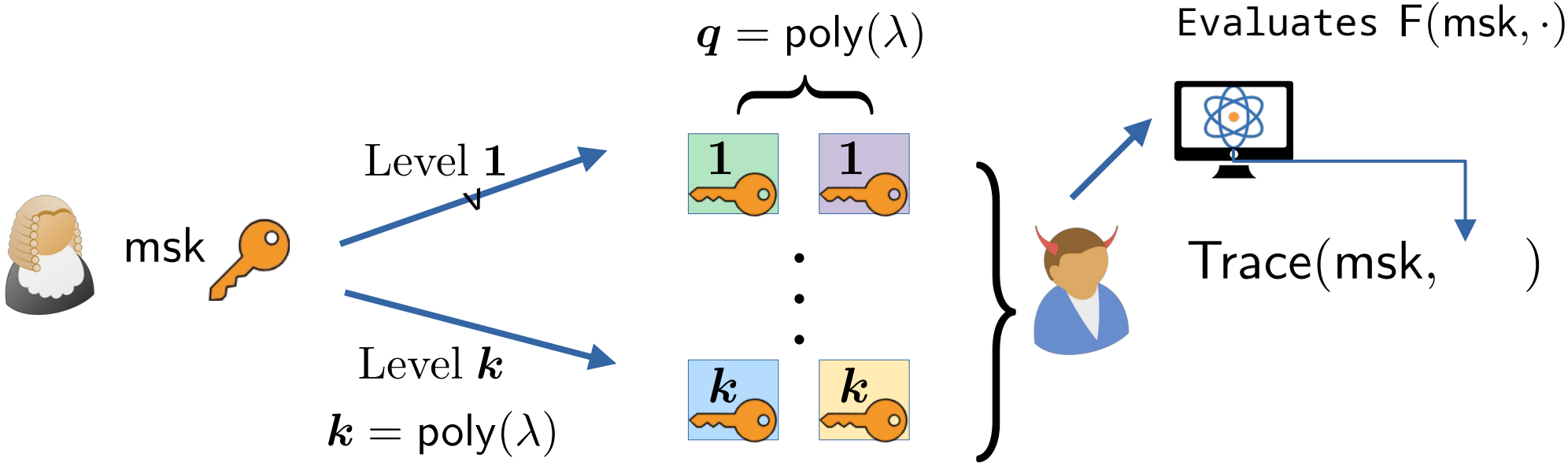
$\mathsf{Eval}(\boxed{1}, \ldots, \boxed{k}, x) = \mathsf{F}(\,\cdot\,, x)$



Multi-Level Traitor Tracing

MLTT

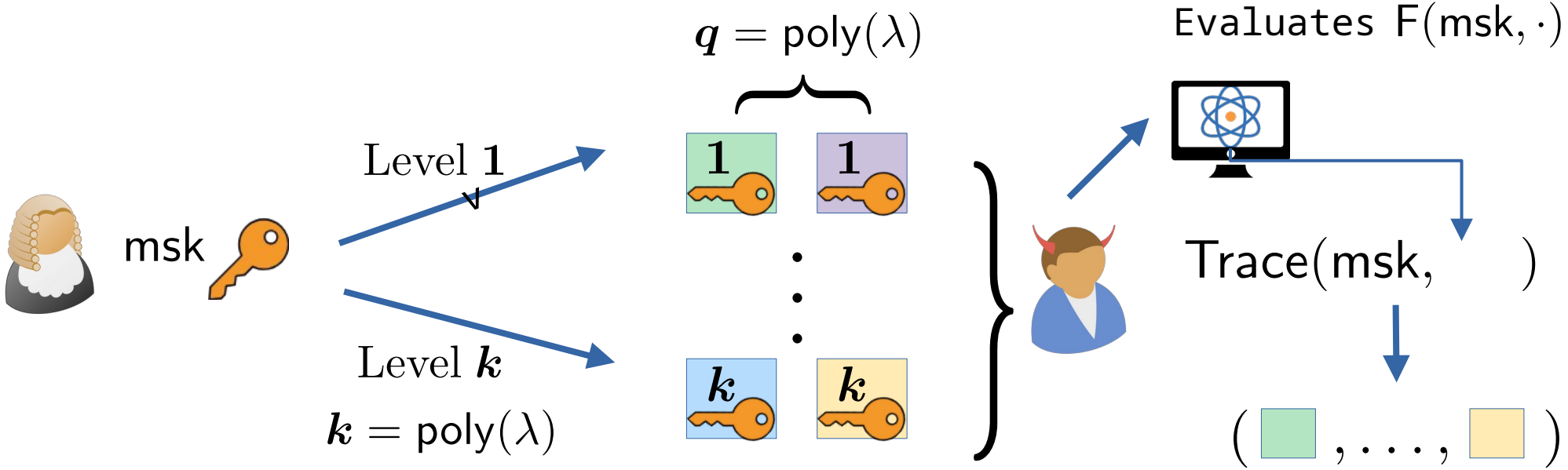
$\mathsf{Eval}(\boxed{1}, \ldots, \boxed{k}, x) = \mathsf{F}(\,\cdot\,, x)$



Multi-Level Traitor Tracing

MLTT

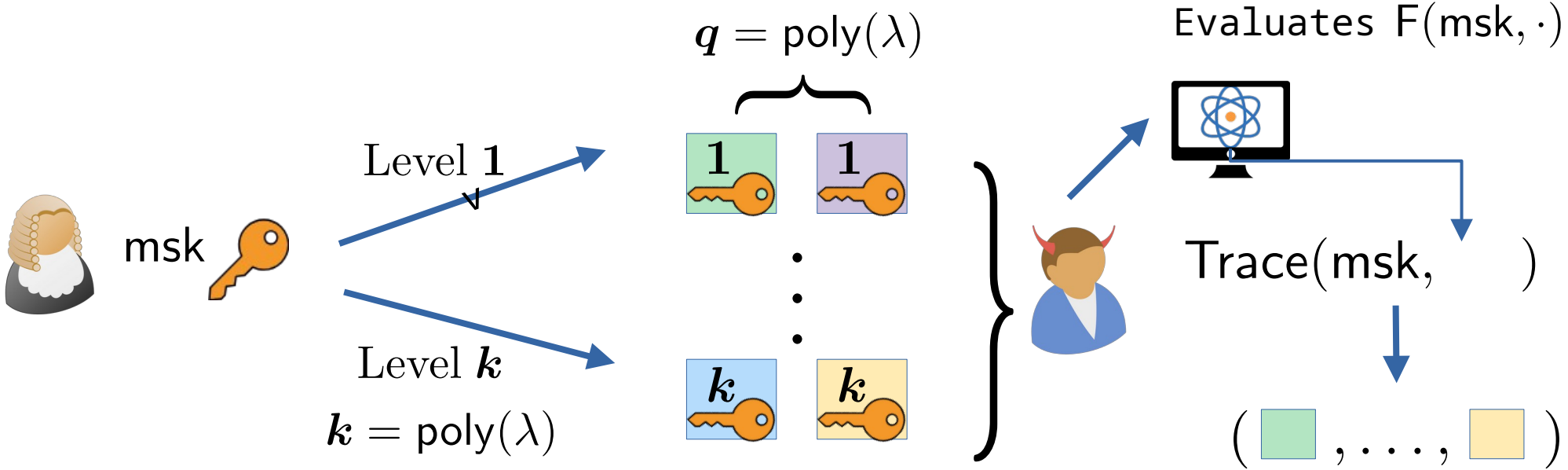
$\mathsf{Eval}(\boxed{1}, \ldots, \boxed{k}, x) = \mathsf{F}(\,\cdot\,, x)$



Multi-Level Traitor Tracing

MLTT

$\mathsf{Eval}(\boxed{1}, \ldots, \boxed{k}, x) = \mathsf{F}(\,\cdot\,, x)$



$(\quad , \ldots , \quad) \in \{ \quad , \quad \} \times \ldots \times \{ \quad , \quad \}$

Two-Superposition States I

Two-Superposition States I

$$(\square, \square, b) \leftarrow \mathcal{U}_\lambda \times \mathcal{U}_\lambda \times \mathcal{U}_2$$

$$2^{-0.5}\left(|\square\rangle + (-1)^b |\square\rangle\right) \longrightarrow$$

[BKM+23] [MPY24]

Two-Superposition States I

$(\square, \square, b) \leftarrow \mathcal{U}_\lambda \times \mathcal{U}_\lambda \times \mathcal{U}_2$

[BKM+23] [MPY24]

$2^{-0.5}\left(|\square\rangle + (-1)^b |\square\rangle\right)$ →  → $(\mathbf{m}, g(\cdot,\cdot))$

Two-Superposition States I

$(\square, \square, b) \leftarrow \mathcal{U}_\lambda \times \mathcal{U}_\lambda \times \mathcal{U}_2$

[BKM+23] [MPY24]

$2^{-0.5}\left(|\square\rangle + (-1)^b |\square\rangle\right)$ →  → $(\mathbf{m}, g(\cdot,\cdot))$

Event Win: $\mathbf{m} \in \{\square, \square\} \wedge g(\square, \square) = b$

Two-Superposition States I

$(\square, \square, b) \leftarrow \mathcal{U}_\lambda \times \mathcal{U}_\lambda \times \mathcal{U}_2$

[BKM+23] [MPY24]

$2^{-0.5}\left(|\square\rangle + (-1)^b |\square\rangle\right)$ →  → $(\mathbf{m}, g(\cdot,\cdot))$

Event Win: $\mathbf{m} \in \{\square, \square\} \wedge g(\square, \square) = b$

$$\Pr[\ \text{Win}\ ] \leq 0.5 + \mathsf{negl}(\lambda)$$

Two-Superposition States I

$(\square, \square, b) \leftarrow \mathcal{U}_\lambda \times \mathcal{U}_\lambda \times \mathcal{U}_2$

[BKM+23] [MPY24]

$2^{-0.5}\left(|\square\rangle + (-1)^b |\square\rangle\right)$ →  → $(\mathbf{m}, g(\cdot,\cdot))$

Event Win: $\mathbf{m} \in \{\square, \square\}$ $\wedge$ $g(\square, \square) = b$

$$\Pr[\ \text{Win}\ ] \leq 0.5 + \mathsf{negl}(\lambda)$$

Parallel Repetition Success Prob: $\mathsf{negl}(\lambda)$

Two-Superposition States II

Two-Superposition States II

$(\square, \square, b_i) \leftarrow [N] \times [N] \times \{0,1\}$

Our Work

$$\left\{ 2^{-0.5}\left(|\square\rangle + (-1)^{b_i} |\square\rangle \right) \right\}_{i \in [q]}$$

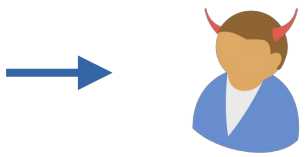
Poly many **iid** states

Two-Superposition States II

$(\square, \square, b_i) \leftarrow [N] \times [N] \times \{0,1\}$

Our Work

$$\left\{ 2^{-0.5}\left(|\square\rangle + (-1)^{b_i} |\square\rangle \right) \right\}_{i \in [q]}$$



$$(\mathbf{m}, g_1, \ldots, g_q)$$

Poly many **iid** states

Two-Superposition States II

$(\square, \square, b_i) \leftarrow [N] \times [N] \times \{0,1\}$

Our Work

$$\left\{ 2^{-0.5}\left(|\square\rangle + (-1)^{b_i} |\square\rangle \right) \right\}_{i \in [q]}$$

$(\mathbf{m}, g_1, \ldots, g_q)$

Poly many **iid** states

$Q_i := \{\square, \square\}$

$\exists N = O(q^2) :$

Two-Superposition States II

$(\square, \square, b_i) \leftarrow [N] \times [N] \times \{0,1\}$

Our Work

$$\left\{ 2^{-0.5}\left(|\square\rangle + (-1)^{b_i} |\square\rangle \right) \right\}_{i \in [q]}$$

$$(\mathbf{m}, g_1, \ldots, g_q)$$

Poly many **iid** states

$Q_i := \{\square, \square\}$

$$\exists N = O(q^2): \quad \Pr\left[\exists i \in [q] : \mathbf{m} \in Q_i \ \wedge \ g_i(\square, \square) = b_i\right] \leq 0.75$$

Two-Superposition States II

$(\square, \square, b_i) \leftarrow [N] \times [N] \times \{0,1\}$

Our Work

$$\left\{ 2^{-0.5}\left(|\square\rangle + (-1)^{b_i} |\square\rangle \right) \right\}_{i \in [q]}$$

$$(\mathbf{m}, g_1, \ldots, g_q)$$

Poly many **iid** states

$Q_i := \{\square, \square\}$

$\exists N = O(q^2)$:

$$\Pr\left[\exists i \in [q] : \mathbf{m} \in Q_i \ \wedge \ g_i(\square, \square) = b_i\right] \leq 0.75$$

Parallel Repetition Success Prob: $\mathsf{negl}(\lambda)$

Our Compiler

Our Compiler

msk

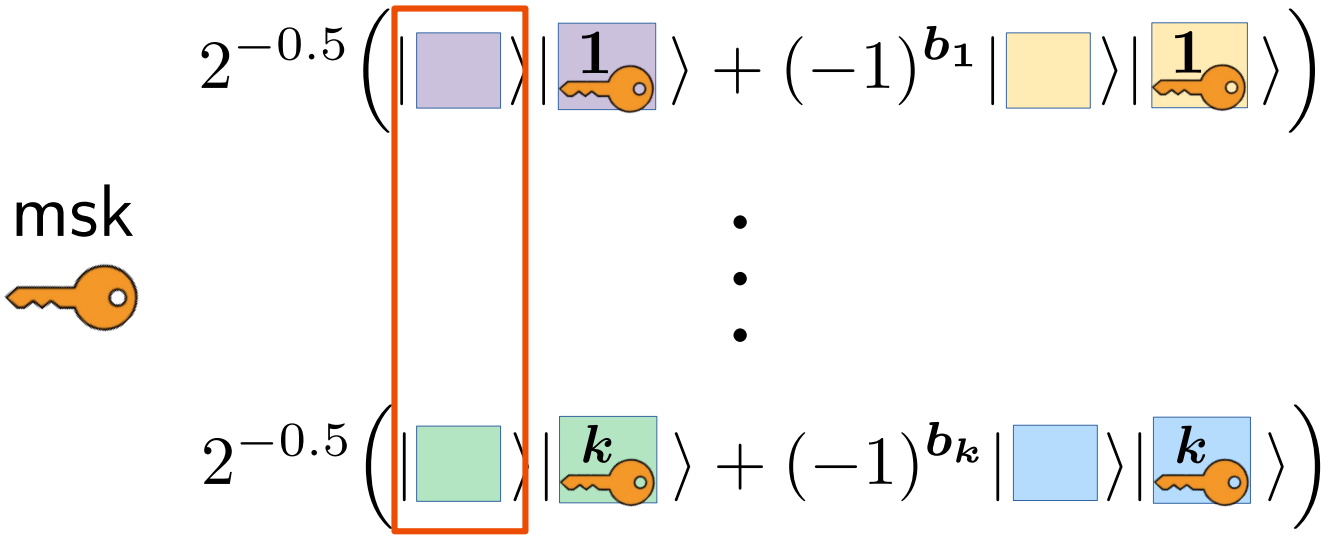
$$2^{-0.5}\left(|\square\rangle|\boxed{1}\rangle + (-1)^{b_1}|\square\rangle|\boxed{1}\rangle\right)$$

$$\vdots$$

$$2^{-0.5}\left(|\square\rangle|\boxed{k}\rangle + (-1)^{b_k}|\square\rangle|\boxed{k}\rangle\right)$$

Our Compiler

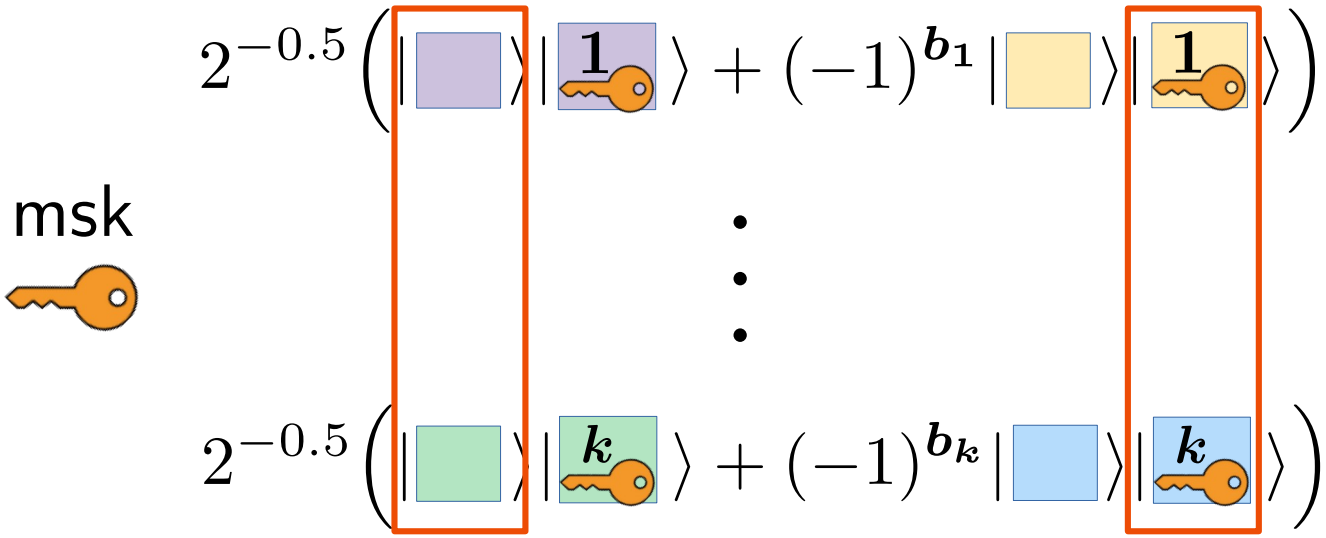
msk



Random **Identities**

Our Compiler

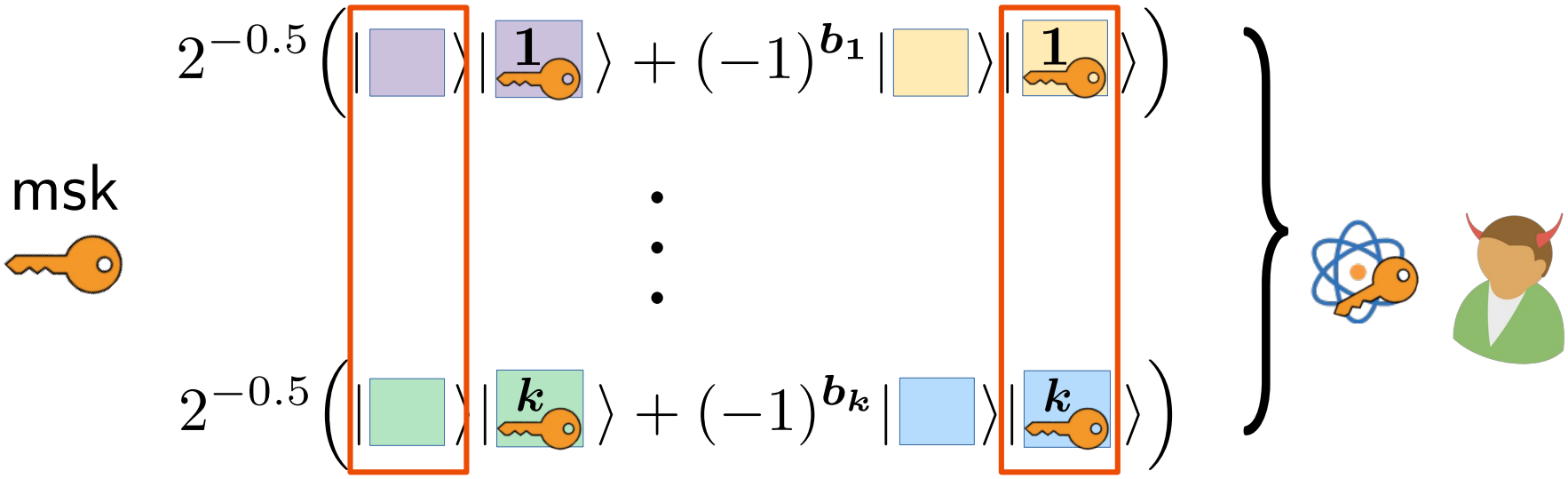
msk



Random **Identities** Corresponding **MLTT Secret Keys**

Our Compiler

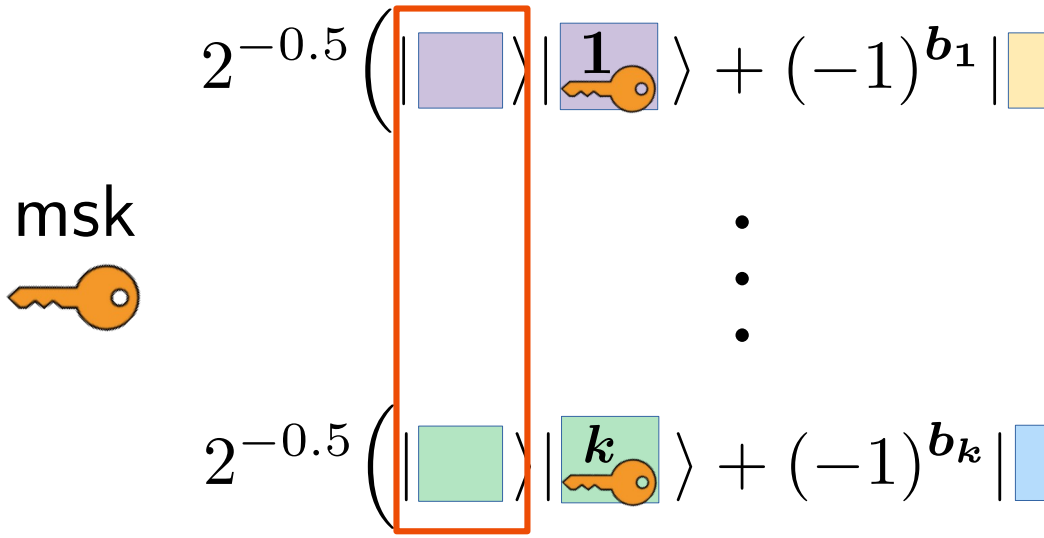
msk



Random **Identities** Corresponding **MLTT Secret Keys**

Our Compiler

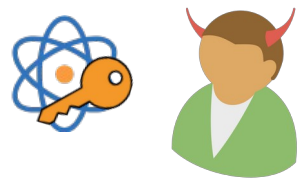
msk



Random **Identities**

Corresponding **MLTT Secret Keys**

Gets $q = \mathsf{poly}(\lambda)$ **such iid states**



Our Compiler

msk

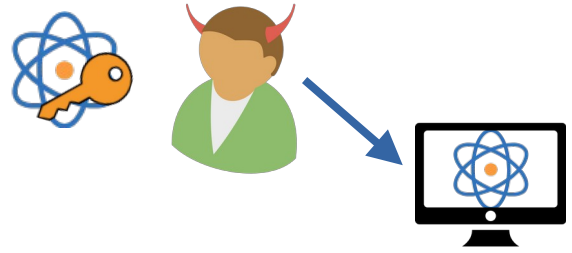
$$2^{-0.5}\left(|\blacksquare\rangle|1\rangle + (-1)^{b_1}|\blacksquare\rangle|1\rangle\right)$$

$$\vdots$$

$$2^{-0.5}\left(|\blacksquare\rangle|k\rangle + (-1)^{b_k}|\blacksquare\rangle|k\rangle\right)$$

Random **Identities** Corresponding **MLTT Secret Keys**

Gets $q = \mathsf{poly}(\lambda)$ **such iid states**



Our Compiler

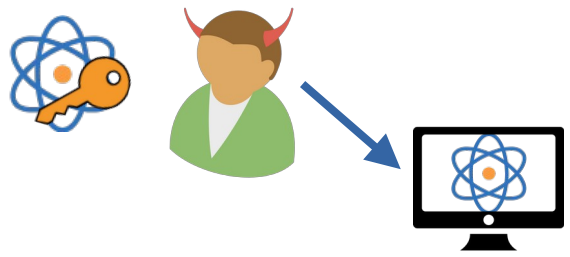
msk

$$2^{-0.5}\left(|\blacksquare\rangle|1\rangle + (-1)^{b_1}|\blacksquare\rangle|1\rangle\right)$$

$$\vdots$$

$$2^{-0.5}\left(|\blacksquare\rangle|k\rangle + (-1)^{b_k}|\blacksquare\rangle|k\rangle\right)$$

Gets $q = \mathsf{poly}(\lambda)$ **such iid states**



Random **Identities** Corresponding **MLTT Secret Keys**

VrfyPass → **Extract**

$$\{g_i^j(\cdot,\cdot)\}_{(i,j)\in[k]\times[q]}$$

Our Compiler

msk

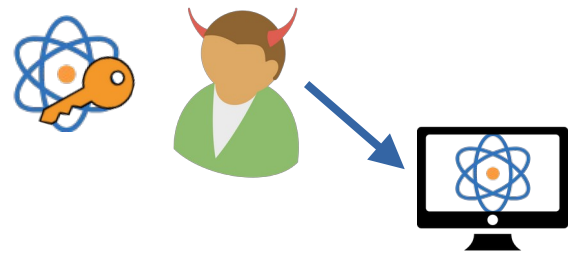
$$2^{-0.5}\left(|\blacksquare\rangle|1\rangle + (-1)^{b_1}|\blacksquare\rangle|1\rangle\right)$$

$$\vdots$$

$$2^{-0.5}\left(|\blacksquare\rangle|k\rangle + (-1)^{b_k}|\blacksquare\rangle|k\rangle\right)$$

Random **Identities** Corresponding **MLTT Secret Keys**

Gets $q = \mathrm{poly}(\lambda)$ **such iid states**



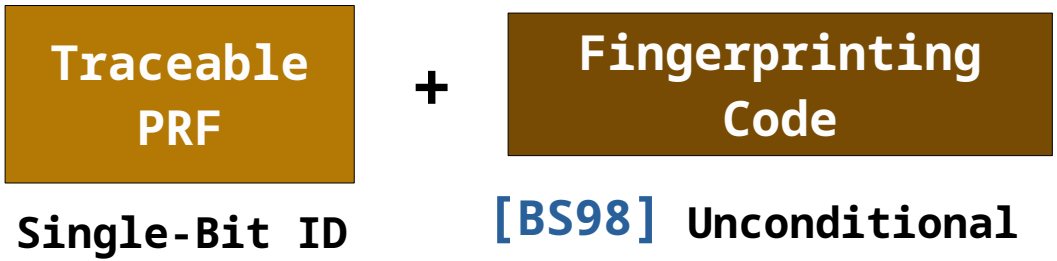
VrfyPass → **Extract** $\{g_i^j(\cdot,\cdot)\}_{(i,j)\in[k]\times[q]}$

GoodProg → **Extract** $\{\mathsf{m}_i\}_{i\in[k]}$

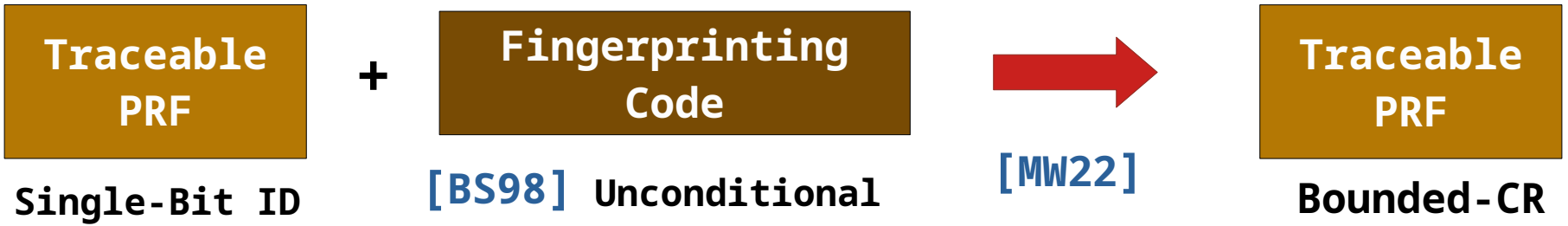
MLTT

Constructing PRF-MLTT

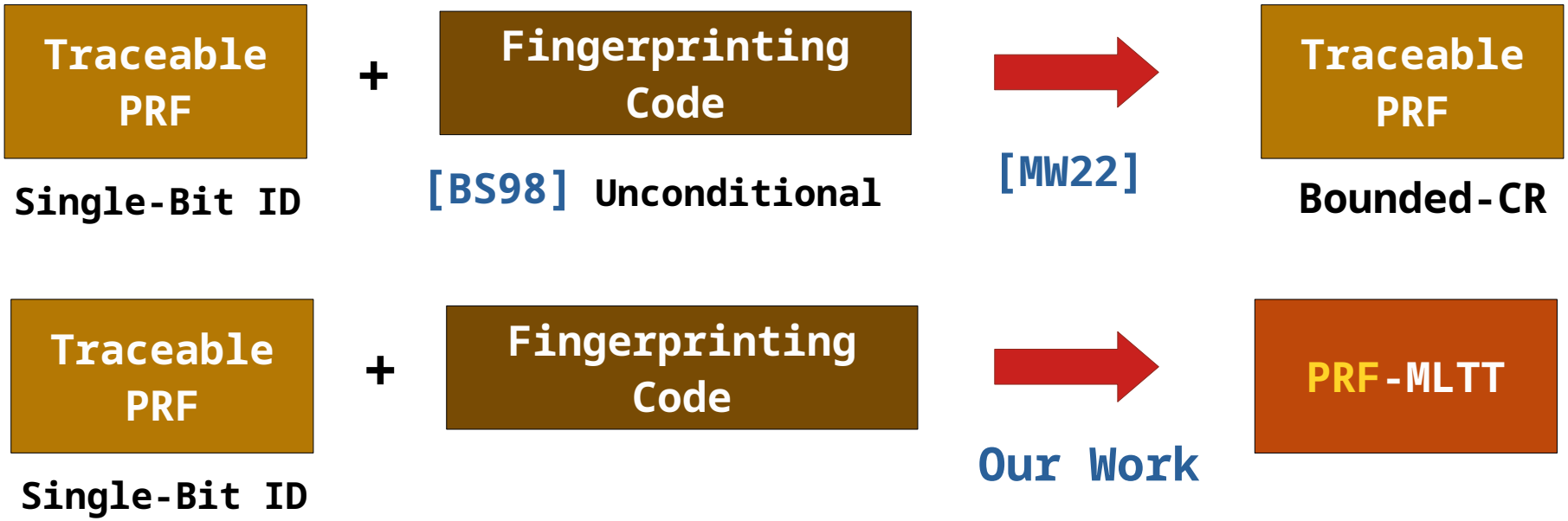
Constructing PRF-MLTT



Constructing PRF-MLTT

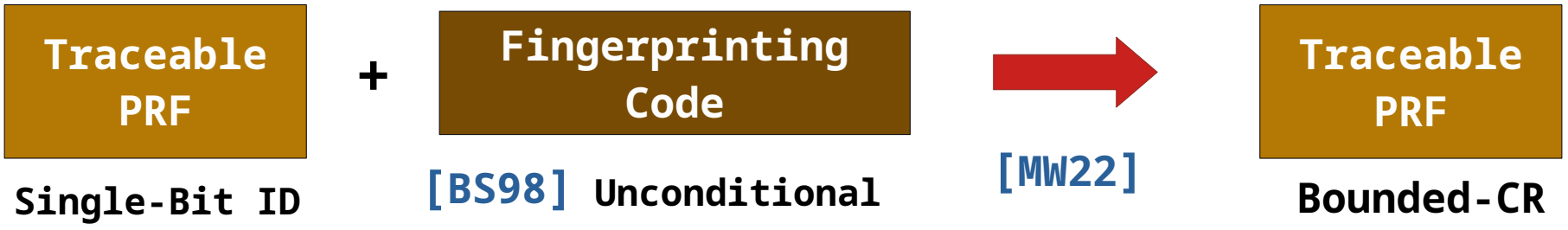


Constructing PRF-MLTT



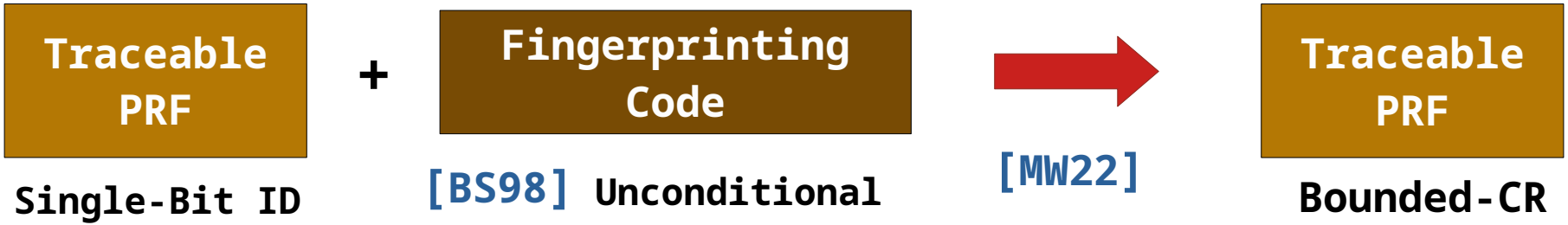
- Single-Bit ID

Constructing PRF-MLTT



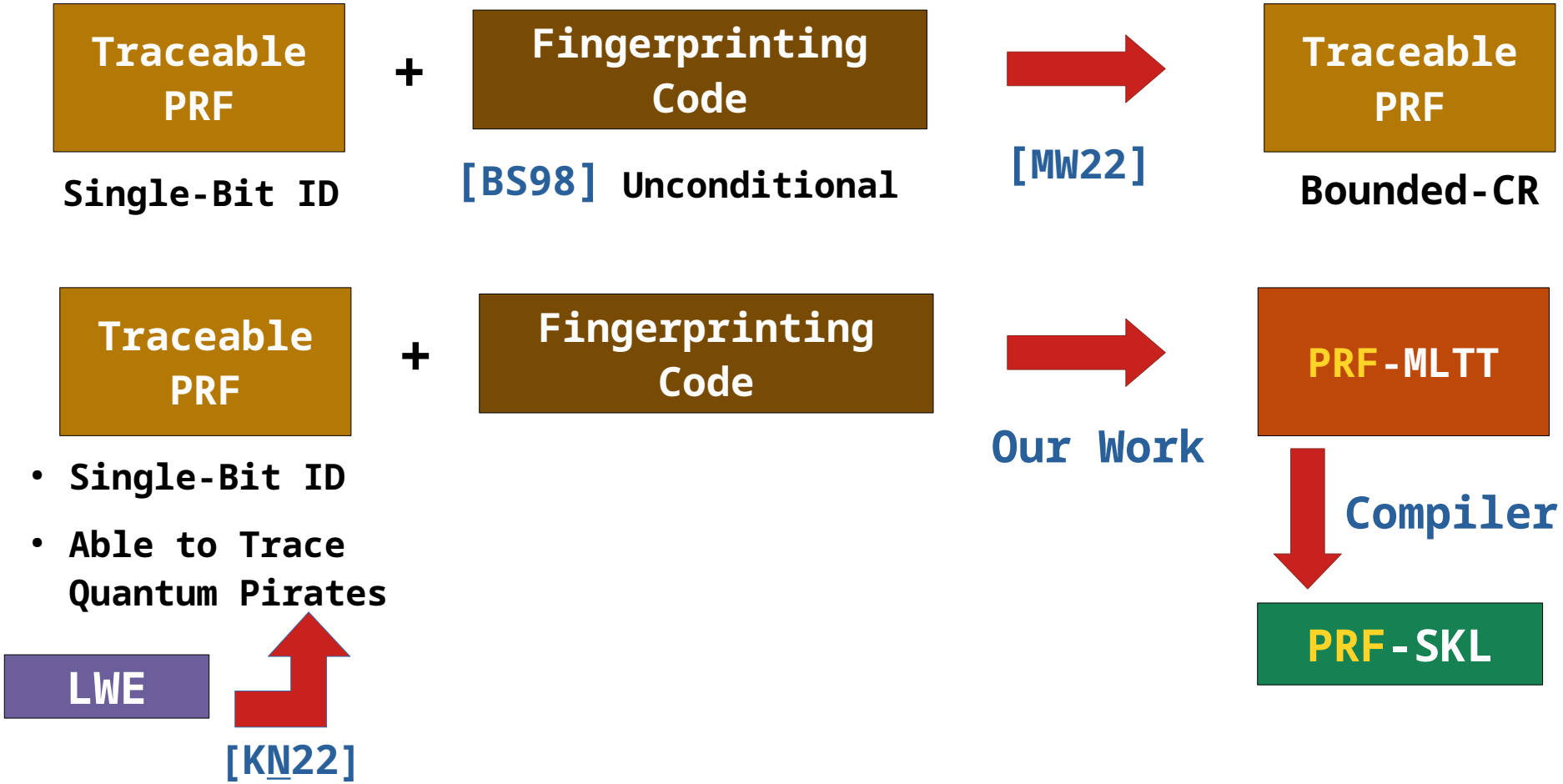
- Single-Bit ID
- Able to Trace Quantum Pirates

Constructing PRF-MLTT

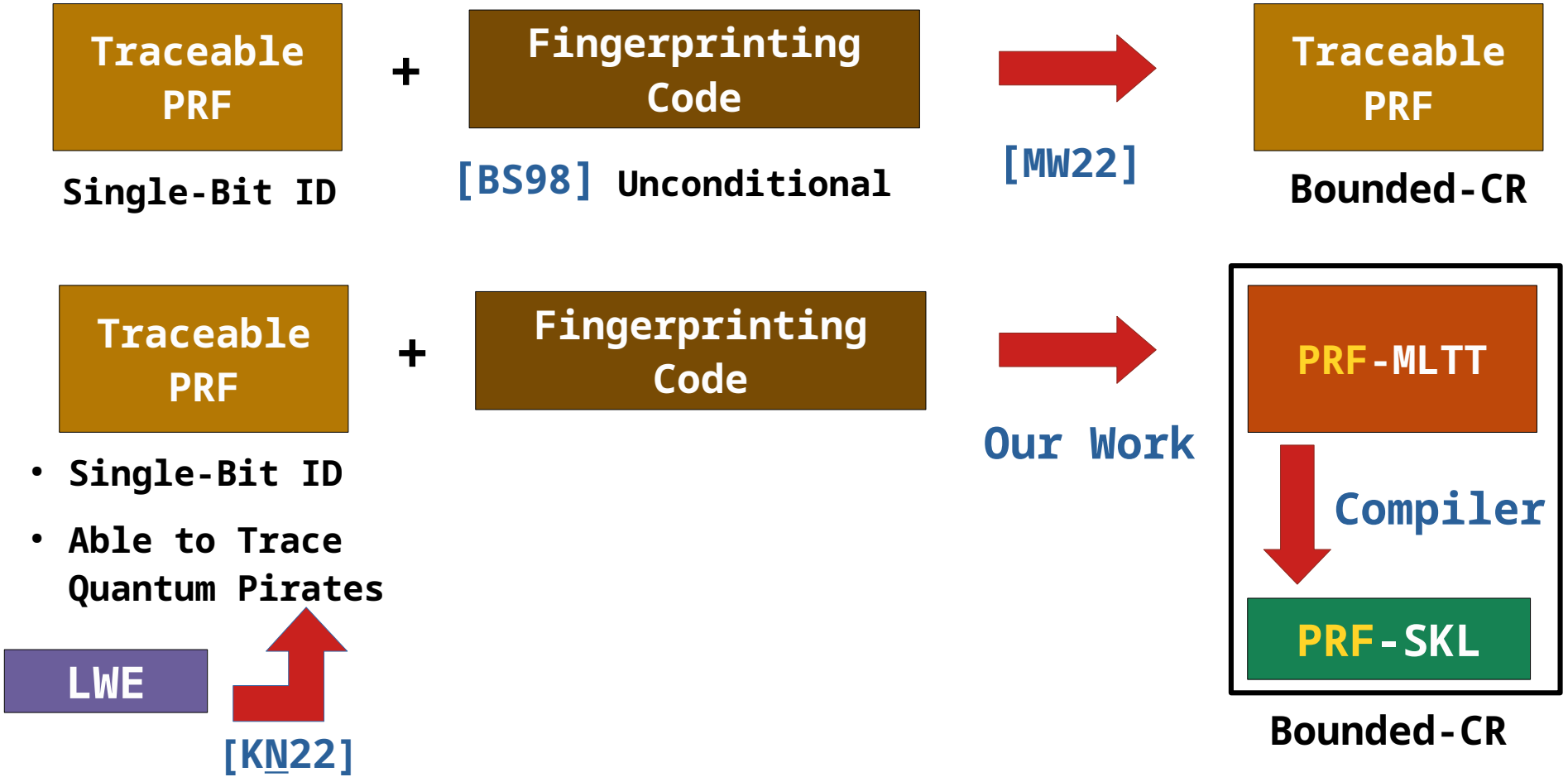


- Single-Bit ID
- Able to Trace Quantum Pirates

Constructing PRF-MLTT



Constructing PRF-MLTT



Constructing PRF-MLTT

Traceable PRF + Fingerprinting Code → Traceable PRF

Single-Bit ID

[BS98] Unconditional

[MW22]

Bounded-CR

Traceable PRF + Fingerprinting Code → Our Work

- Single-Bit ID
- Able to Trace Quantum Pirates

LWE

[KN22]

⚠ **Challenge:** Tracing **Quantum** Pirates!

PRF-MLTT

Compiler

PRF-SKL

Bounded-CR

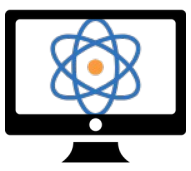
Tracing Quantum Pirates

Tracing Quantum Pirates

$$\begin{cases} |\square\rangle|\,1\text{-key}\,\rangle + (-1)^{b_1}|\square\rangle|\,1\text{-key}\,\rangle \\ \vdots \\ |\square\rangle|\,k\text{-key}\,\rangle + (-1)^{b_k}|\square\rangle|\,k\text{-key}\,\rangle \end{cases}$$

Gets $q = \mathrm{poly}(\lambda)$ **such iid states**

Tracing Quantum Pirates



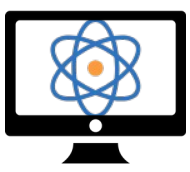
$$\begin{cases} |\square\rangle|\,\mathbf{1}_{\text{key}}\rangle + (-1)^{b_1}|\square\rangle|\,\mathbf{1}_{\text{key}}\rangle \\ \vdots \\ |\square\rangle|\,k_{\text{key}}\rangle + (-1)^{b_k}|\square\rangle|\,k_{\text{key}}\rangle \end{cases}$$

Gets $q = \text{poly}(\lambda)$ **such iid states**

Tracing Quantum Pirates

$$\begin{cases} |\square\rangle|\,1\,\rangle + (-1)^{b_1}|\square\rangle|\,1\,\rangle \\ \vdots \\ |\square\rangle|\,k\,\rangle + (-1)^{b_k}|\square\rangle|\,k\,\rangle \end{cases}$$

Gets $q = \mathrm{poly}(\lambda)$ **such iid states**

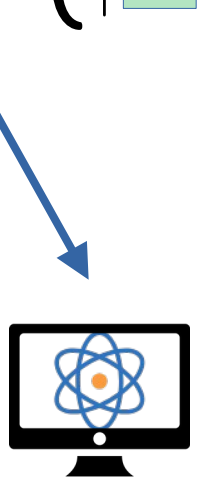


Trace(msk, )

Tracing Quantum Pirates

$$\begin{cases} |\ \rangle|\ 1\ \rangle + (-1)^{b_1}|\ \rangle|\ 1\ \rangle \\ \quad\vdots \\ |\ \rangle|\ k\ \rangle + (-1)^{b_k}|\ \rangle|\ k\ \rangle \end{cases}$$

Gets $q = \text{poly}(\lambda)$ **such iid states**



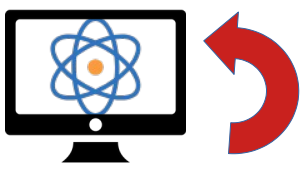
Trace(msk, )

- **Estimate** wrt $(b, x, y) \leftarrow \mathcal{D}_1 \approx \mathcal{D}$ to get m_1.

Tracing Quantum Pirates

$$\begin{cases} |\square\rangle|\boxed{1}\rangle + (-1)^{b_1}|\square\rangle|\boxed{1}\rangle \\ \vdots \\ |\square\rangle|\boxed{k}\rangle + (-1)^{b_k}|\square\rangle|\boxed{k}\rangle \end{cases}$$

Gets $q = \mathrm{poly}(\lambda)$ **such iid states**



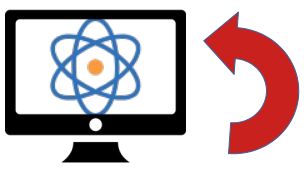
Trace(msk, )

- **Estimate** wrt $(b, x, y) \leftarrow \mathcal{D}_1 \approx \mathcal{D}$ to get $\boxed{\mathsf{m}_1}$.
- Apply **Quantum Rewinding** [CMSZ21,Zha23].

Tracing Quantum Pirates

$$\begin{cases} |\square\rangle|1\text{-key}\rangle + (-1)^{b_1}|\square\rangle|1\text{-key}\rangle \\ \vdots \\ |\square\rangle|k\text{-key}\rangle + (-1)^{b_k}|\square\rangle|k\text{-key}\rangle \end{cases}$$

Gets $q = \mathrm{poly}(\lambda)$ **such iid states**

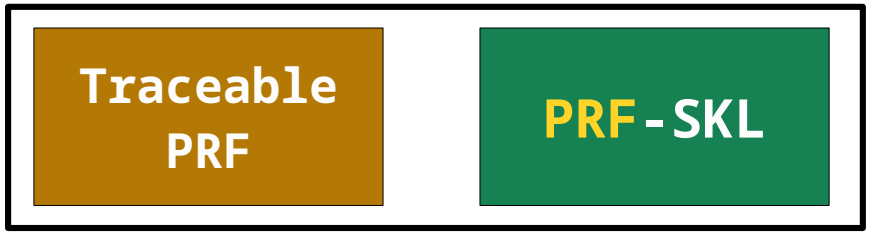


Trace(msk, )

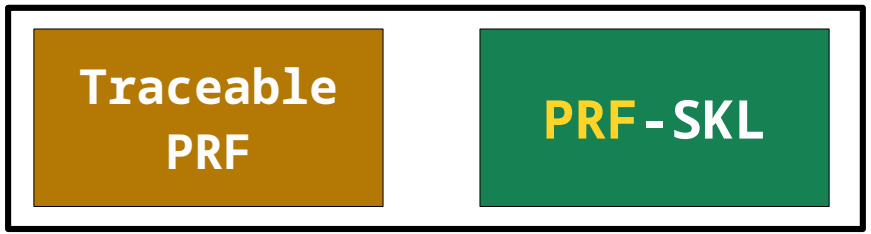
- **Estimate** wrt $(b, x, y) \leftarrow \mathcal{D}_1 \approx \mathcal{D}$ to get m_1.
- Apply **Quantum Rewinding** [CMSZ21,Zha23].
- **Repeat** w/ $\mathcal{D}_2 \approx \ldots \approx \mathcal{D}_k \approx \mathcal{D}$ until getting m_k.

Open Questions

Open Questions



Open Questions



Unbounded Collusion Resistance??

Open Questions

Unbounded Collusion Resistance??

PRF-SKL may not need to go through **Traceable PRFs** ...

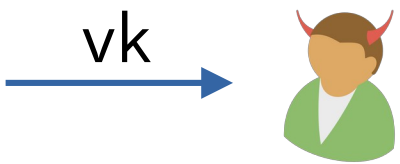
Open Questions

Unbounded Collusion Resistance??

PRF-SKL may not need to go through **Traceable PRFs** ...

Publicly Verifiable Deletion (PVD)

$\mathsf{Vrfy}(\mathsf{vk}, \cdot)$



Open Questions

Traceable PRF

PRF-SKL

Unbounded Collusion Resistance??

PRF-SKL may not need to go through **Traceable PRFs** ...

Publicly Verifiable Deletion (PVD)

PRF-PVD-SKL

$\mathsf{Vrfy}(\mathsf{vk}, \cdot)$

vk

PRF-Copy Protection

PK-Q-Money

Open Questions

Traceable PRF

PRF-SKL

Unbounded Collusion Resistance??

PRF-SKL may not need to go through **Traceable PRFs** ...

Publicly Verifiable Deletion (PVD)

$\mathsf{Vrfy}(\mathsf{vk}, \cdot)$

vk

PRF-PVD-SKL

??

??

PRF-Copy Protection

??

PK-Q-Money

Open Questions

Traceable PRF

PRF-SKL

Unbounded Collusion Resistance??

PRF-SKL may not need to go through **Traceable PRFs** ...

Publicly Verifiable Deletion (PVD)

$\mathsf{Vrfy}(\mathsf{vk}, \cdot)$

vk

PRF-PVD-SKL

??

??

Even w/o collusion!

PRF-Copy Protection

??

PK-Q-Money

Thank You!