Adaptive Multiparty NIKE

Venkata Koppula Brent Waters Mark Zhandry

TCC 2022

Non-interactive Key Exchange (NIKE)

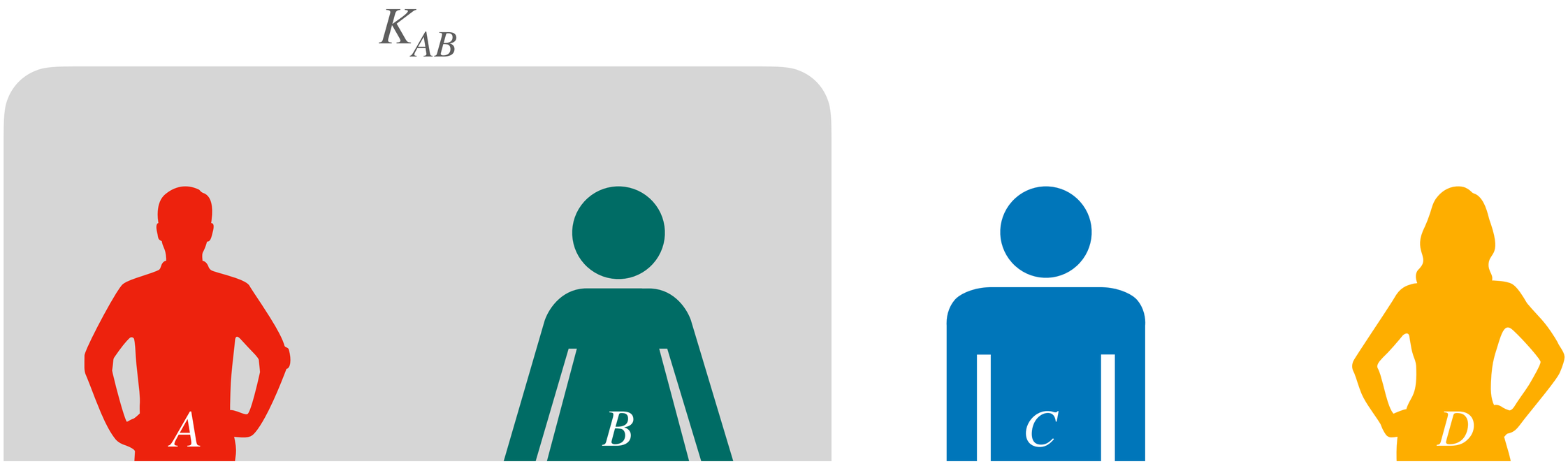
Goal: Any subset of users can derive a shared secret key

Non-interactive Key Exchange (NIKE)

Goal: Any subset of users can derive a shared secret key

Non-interactive Key Exchange (NIKE)

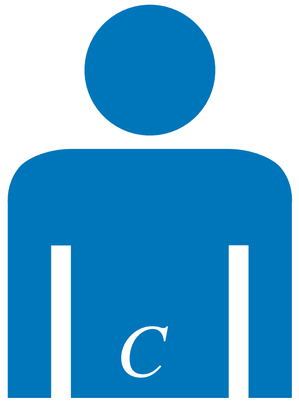
Goal: Any subset of users can derive a shared secret key



Non-interactive Key Exchange (NIKE)

Public Bulletin Board

pk_A pk_B pk_C pk_D

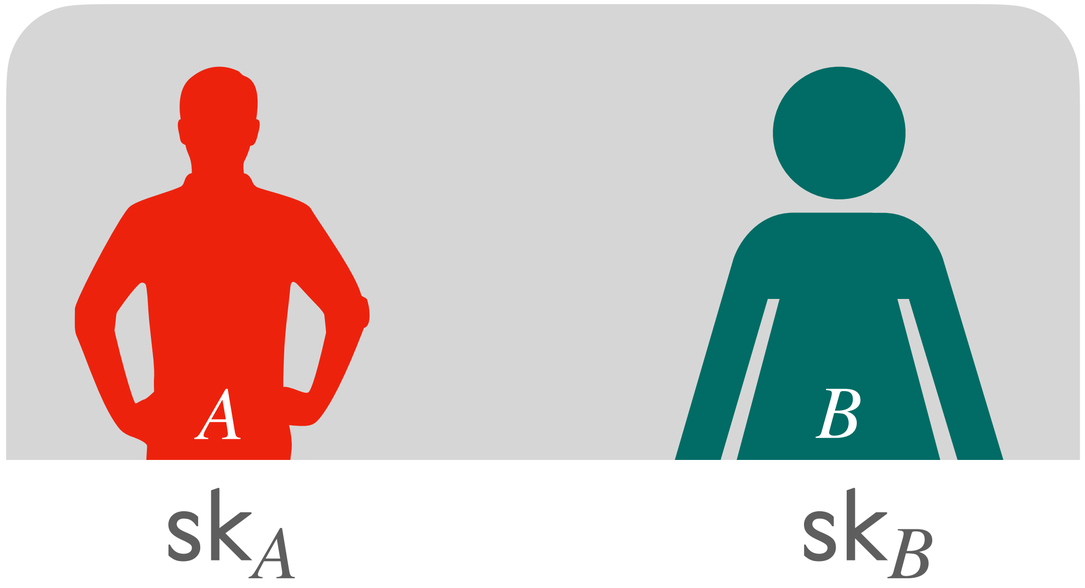


sk_A sk_B sk_C sk_D

Non-interactive Key Exchange (NIKE)

$\mathsf{KeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \rightarrow K_{AB}$

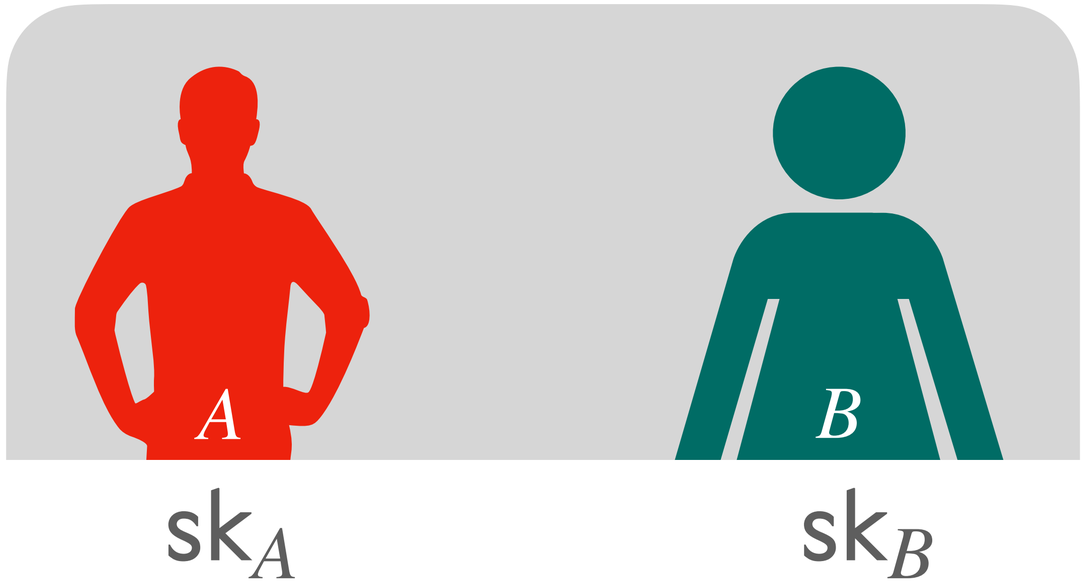
$\mathsf{KeyGen}(\mathsf{sk}_B, \mathsf{pk}_A) \rightarrow K_{AB}$



Non-interactive Key Exchange (NIKE)

$$\mathsf{KeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \rightarrow K_{AB}$$

$$\mathsf{KeyGen}(\mathsf{sk}_B, \mathsf{pk}_A) \rightarrow K_{AB}$$

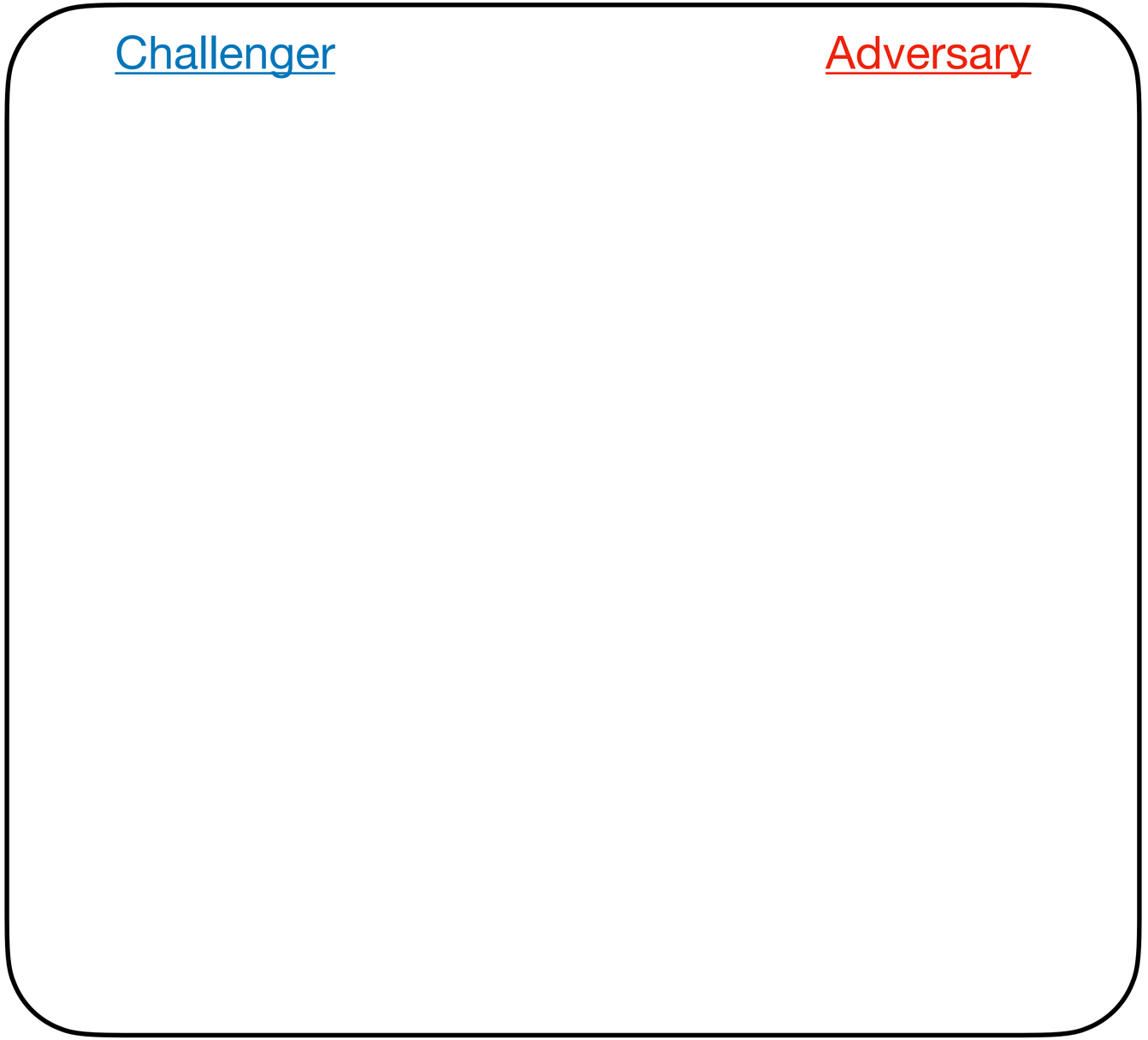


Security:

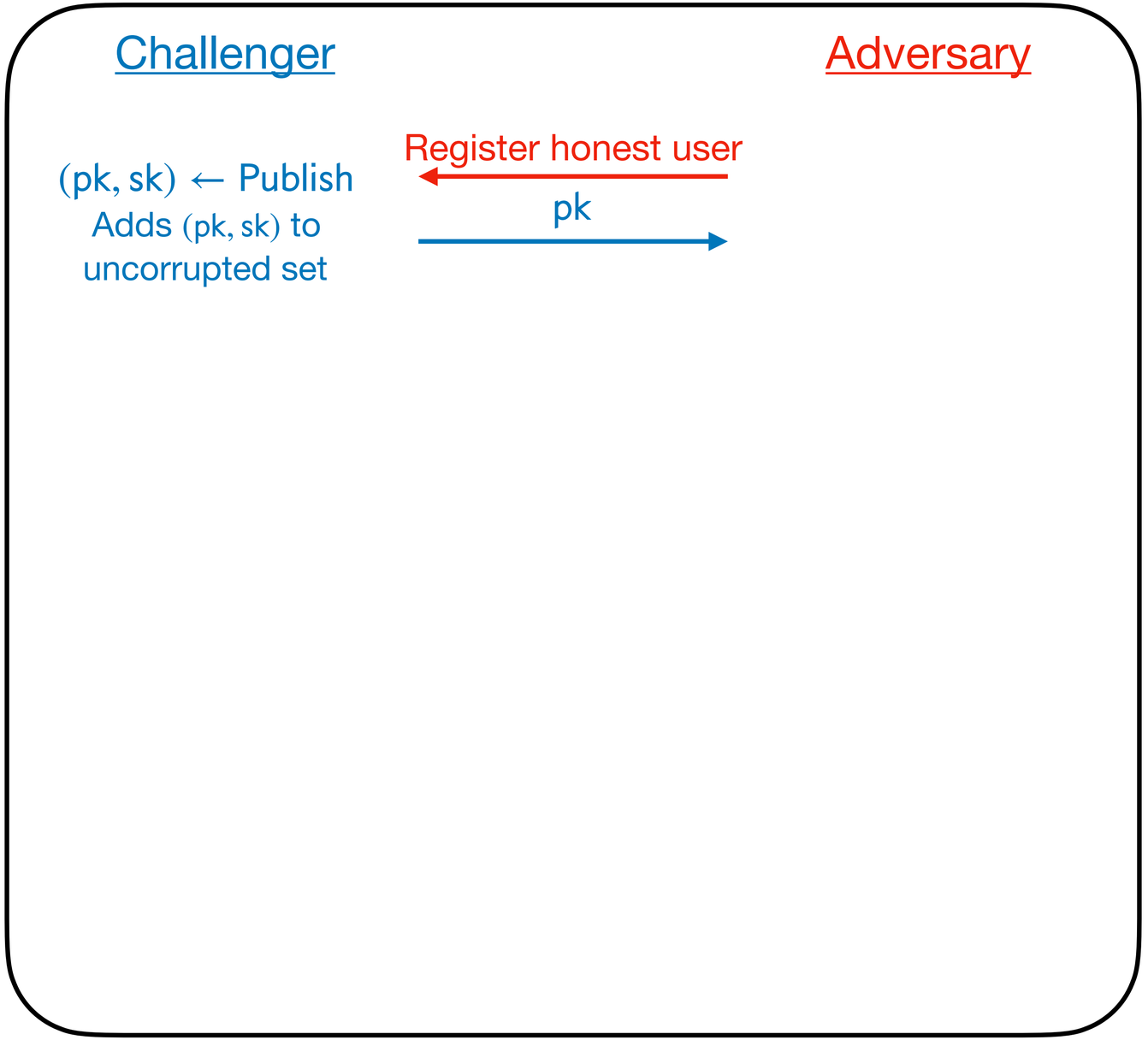
Adversary cannot guess the secret key for set S

even after seeing many secret keys for users outside S

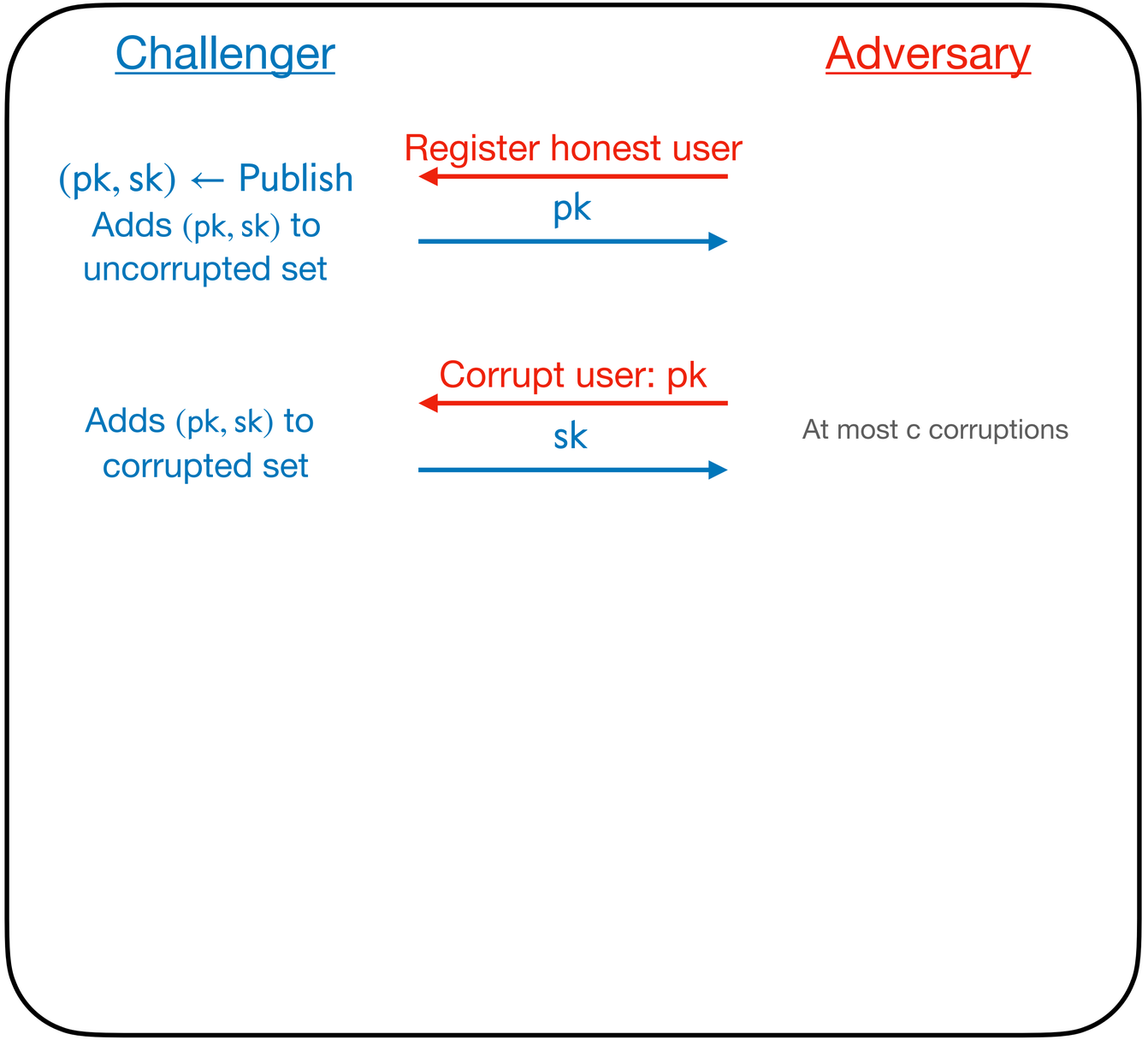
Adaptive Security for ‘c’ corruptions



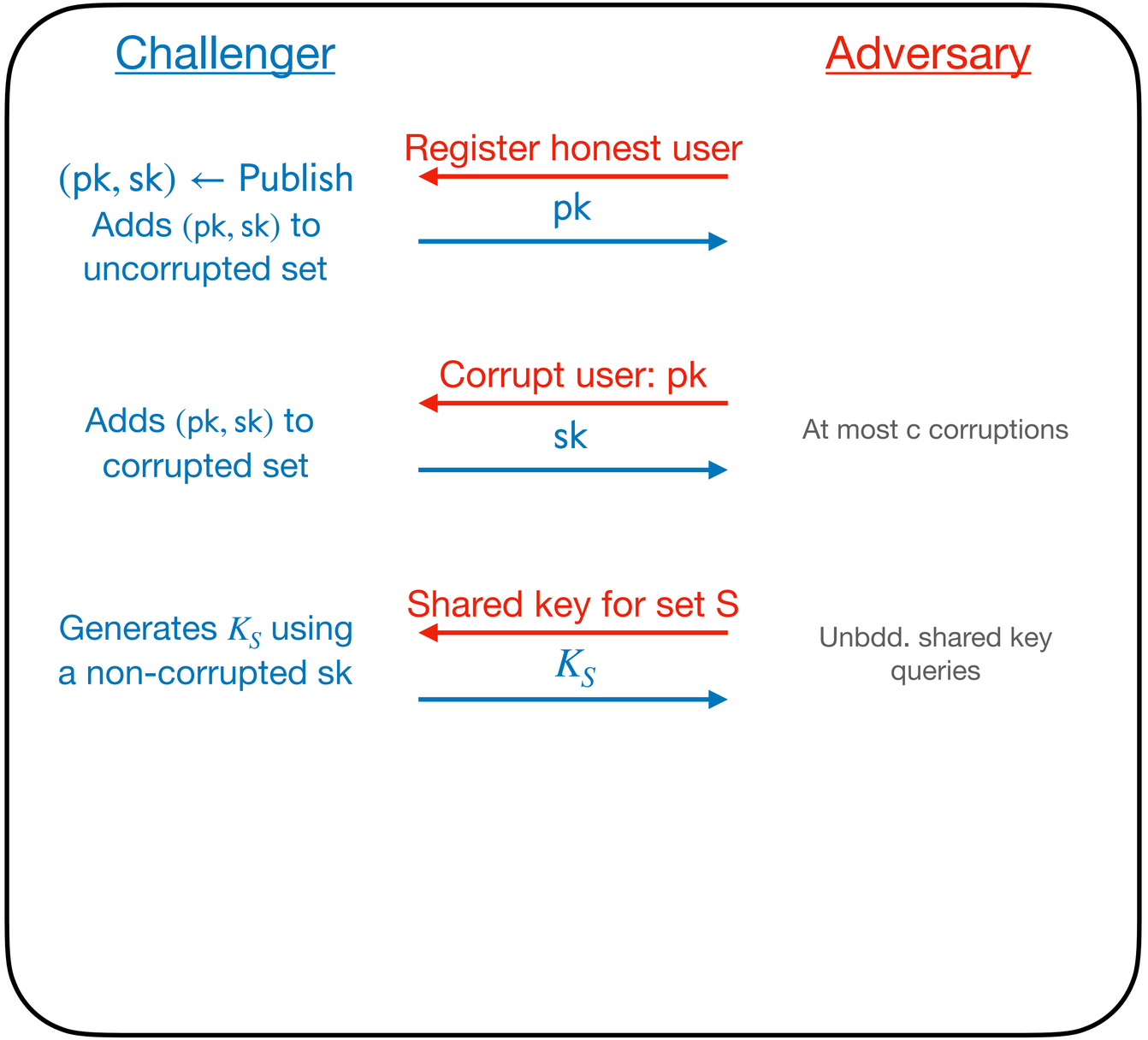
Adaptive Security for ‘c’ corruptions



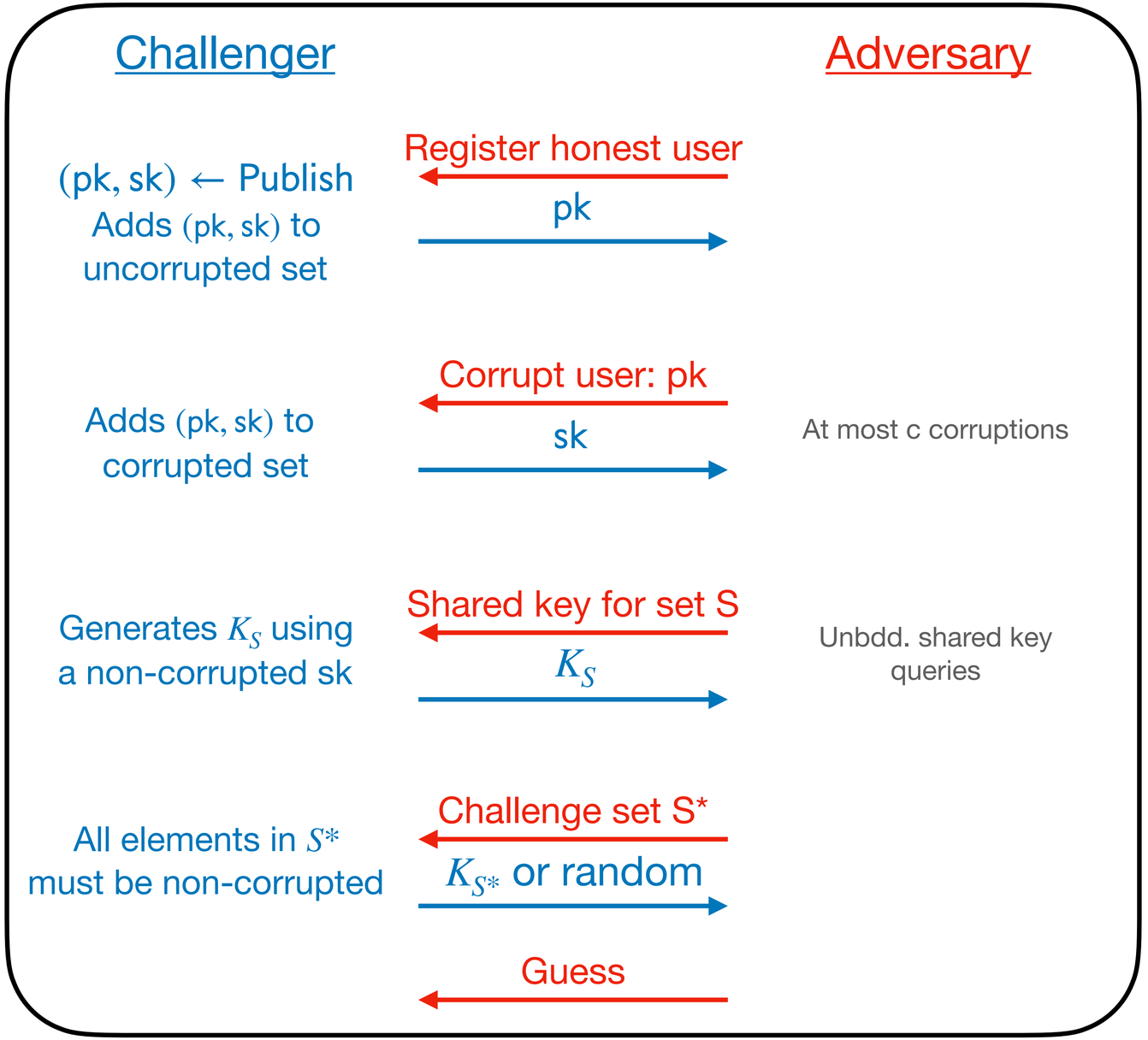
Adaptive Security for ‘c’ corruptions



Adaptive Security for ‘c’ corruptions



Adaptive Security for ‘c’ corruptions



Why Adaptive Security

Why Adaptive Security

A natural and important goal in crypto

- If shared key for one group is leaked, it should not compromise security of other groups
- Adversary may corrupt some users, and then decide which group's security is compromised

Why Adaptive Security

A natural and important goal in crypto

- If shared key for one group is leaked, it should not compromise security of other groups
- Adversary may corrupt some users, and then decide which group's security is compromised

2-party key exchange:

- can 'guess' the challenge group, N^2 factor loss in security
- Reducing this loss is an active area of research

Why Adaptive Security

A natural and important goal in crypto

- If shared key for one group is leaked, it should not compromise security of other groups
- Adversary may corrupt some users, and then decide which group's security is compromised

2-party key exchange:

- can 'guess' the challenge group, N^2 factor loss in security
- Reducing this loss is an active area of research

Multiparty key exchange:

- cannot guess the challenge group (N^ℓ factor loss, ℓ can be polynomial)

Prior Works

Prior Works

2-party :

- Diffie-Hellman (1976)

Prior Works

2-party :

- Diffie-Hellman (1976)

3-party :

- Joux (2000) using pairings

Prior Works

2-party :

- Diffie-Hellman (1976)

3-party :

- Joux (2000) using pairings

n-party :

- Garg-Gentry-Halevi (2013) using multilinear maps
  - Constructions for super-constant n are broken
- Boneh-Zhandry (2014) using iO
  - Achieves selective security
- Boneh-Zhandry (2014), Rao (2014)
  - Achieve adaptive security using interactive assumptions

Rao 2014 Impossibility

Barriers for achieving adaptive security from non-interactive assumptions

Public keys fix the shared key for every set S

`Admissible reductions' not possible

- Impossibility holds even if the number of corruptions is bounded

All prior constructions are admissible

Our Results and Contributions

Our Results and Contributions

Generic compilers to simplify design of multiparty NIKE :

- Common setup can be removed generically
- Shared key queries can be removed from the security game

Our Results and Contributions

Generic compilers to simplify design of multiparty NIKE :

- Common setup can be removed generically
- Shared key queries can be removed from the security game

A close connection between adaptive NIKE and adaptive constrained PRFs :

- Constrained PRFs (for a special class) + iO => Multiparty NIKE with bounded honest users

Our Results and Contributions

Generic compilers to simplify design of multiparty NIKE :

- Common setup can be removed generically
- Shared key queries can be removed from the security game

A close connection between adaptive NIKE and adaptive constrained PRFs :

- Constrained PRFs (for a special class) + iO => Multiparty NIKE with bounded honest users

Construction of constrained PRFs for special class, using iO + DDH

Our Results and Contributions

Generic compilers to simplify design of multiparty NIKE :

- Common setup can be removed generically
- Shared key queries can be removed from the security game

A close connection between adaptive NIKE and adaptive constrained PRFs :

- Constrained PRFs (for a special class) + iO => Multiparty NIKE with bounded honest users

Construction of constrained PRFs for special class, using iO + DDH

Direct construction of multiparty NIKE with unbounded honest users, bounded corruptions, using iO + standard assumptions on groups

Our Results and Contributions

Generic compilers to simplify design of multiparty NIKE :

- Common setup can be removed generically
- Shared key queries can be removed from the security game

This talk

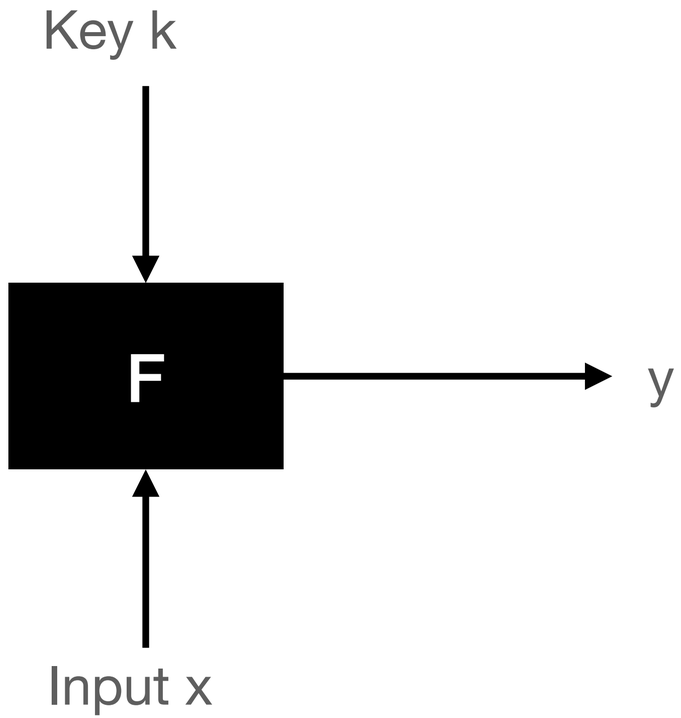
A close connection between adaptive NIKE and adaptive constrained PRFs :

- Constrained PRFs (for a special class) + iO => Multiparty NIKE with bounded honest users

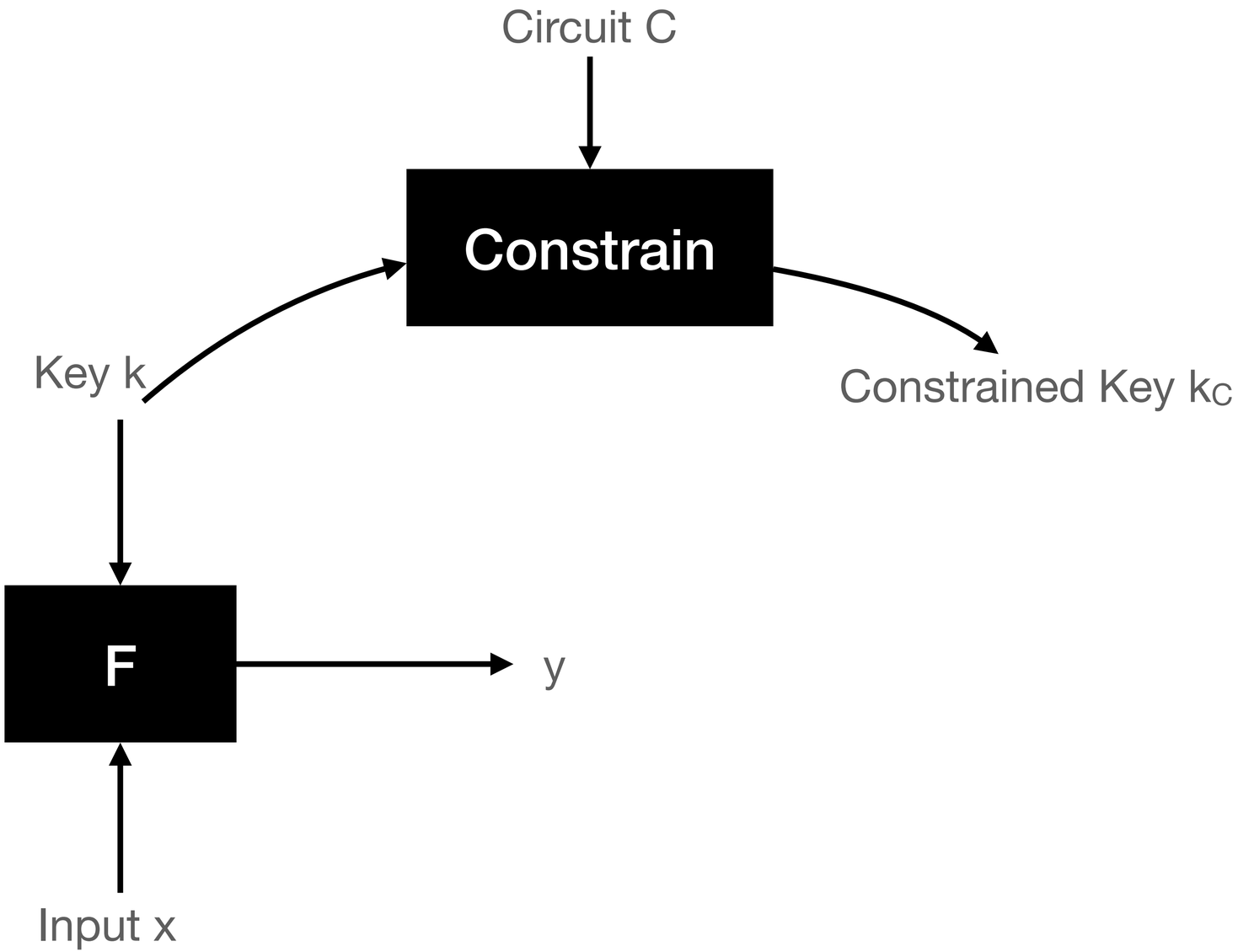
Construction of constrained PRFs for special class, using iO + DDH

Direct construction of multiparty NIKE with unbounded honest users, bounded corruptions, using iO + standard assumptions on groups

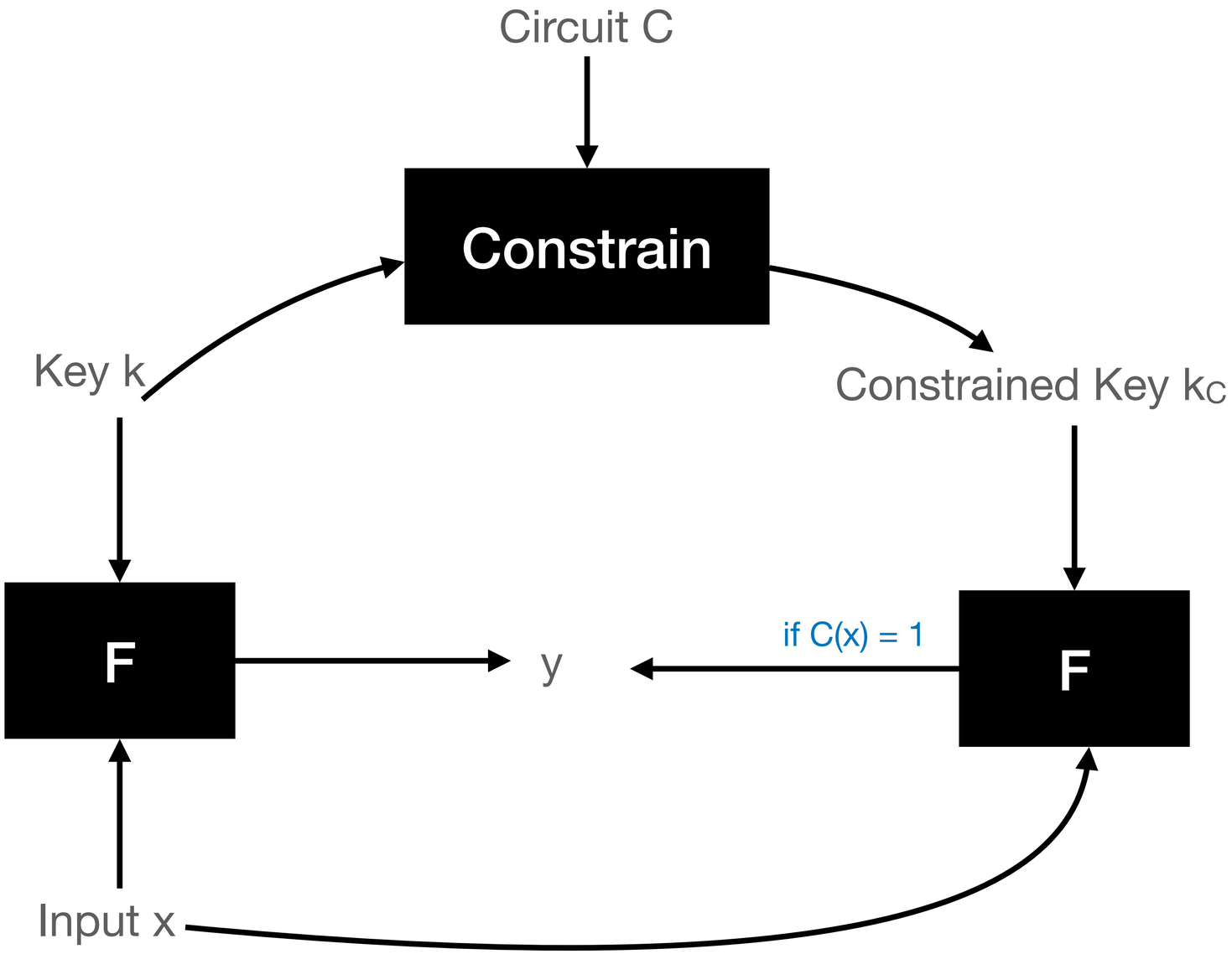
Constrained PRFs



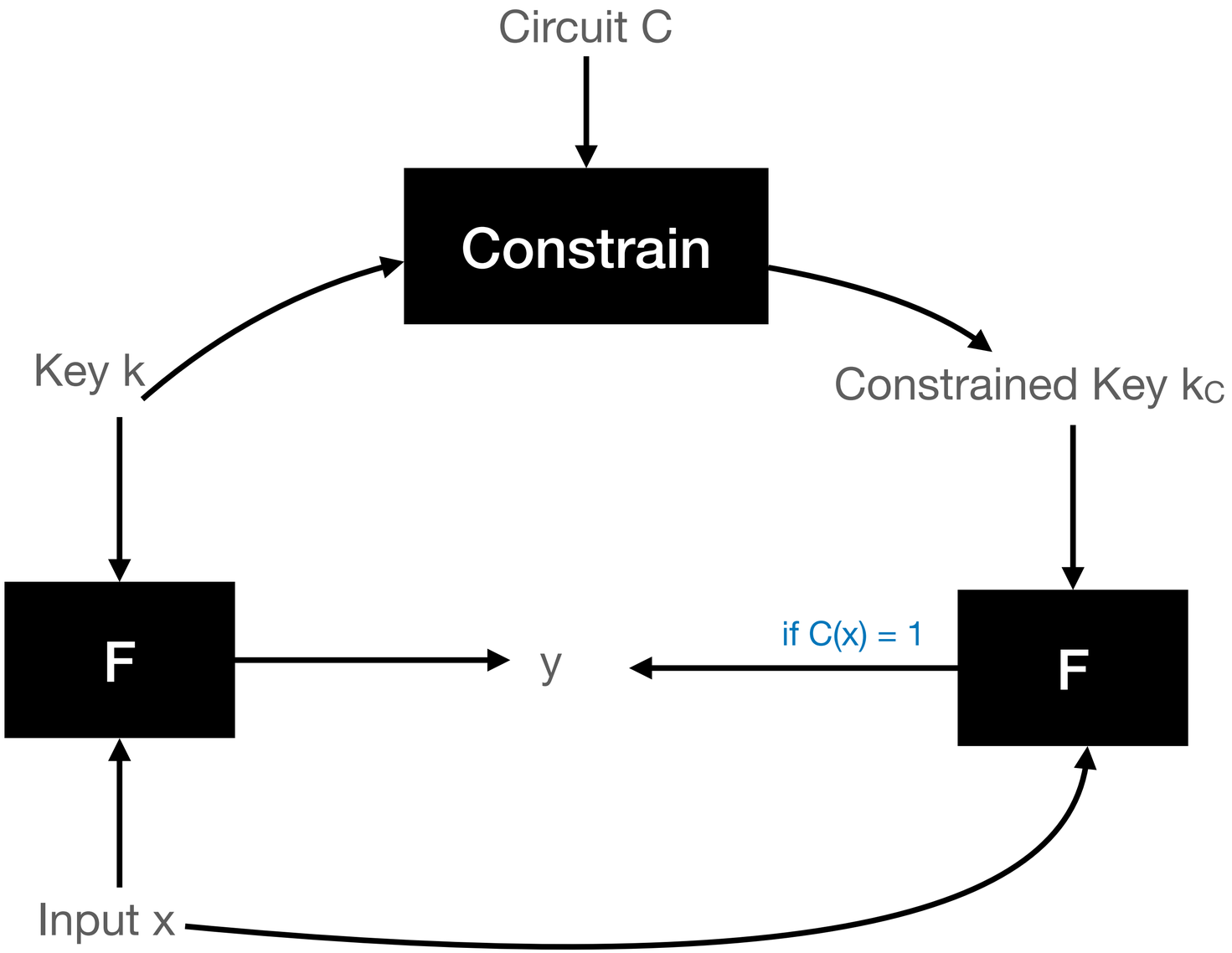
Constrained PRFs



Constrained PRFs



Constrained PRFs



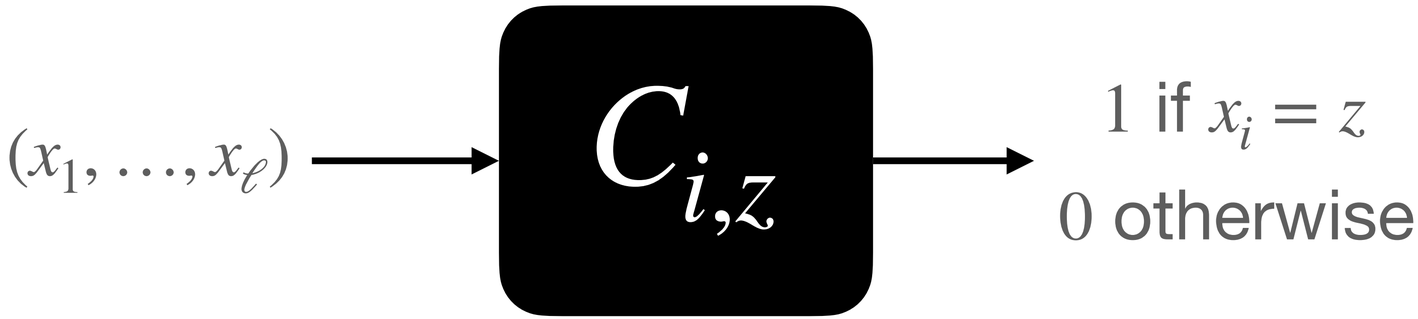
Security

If C(x) = 0, then F(k, x) should look random even if adv. has k_C

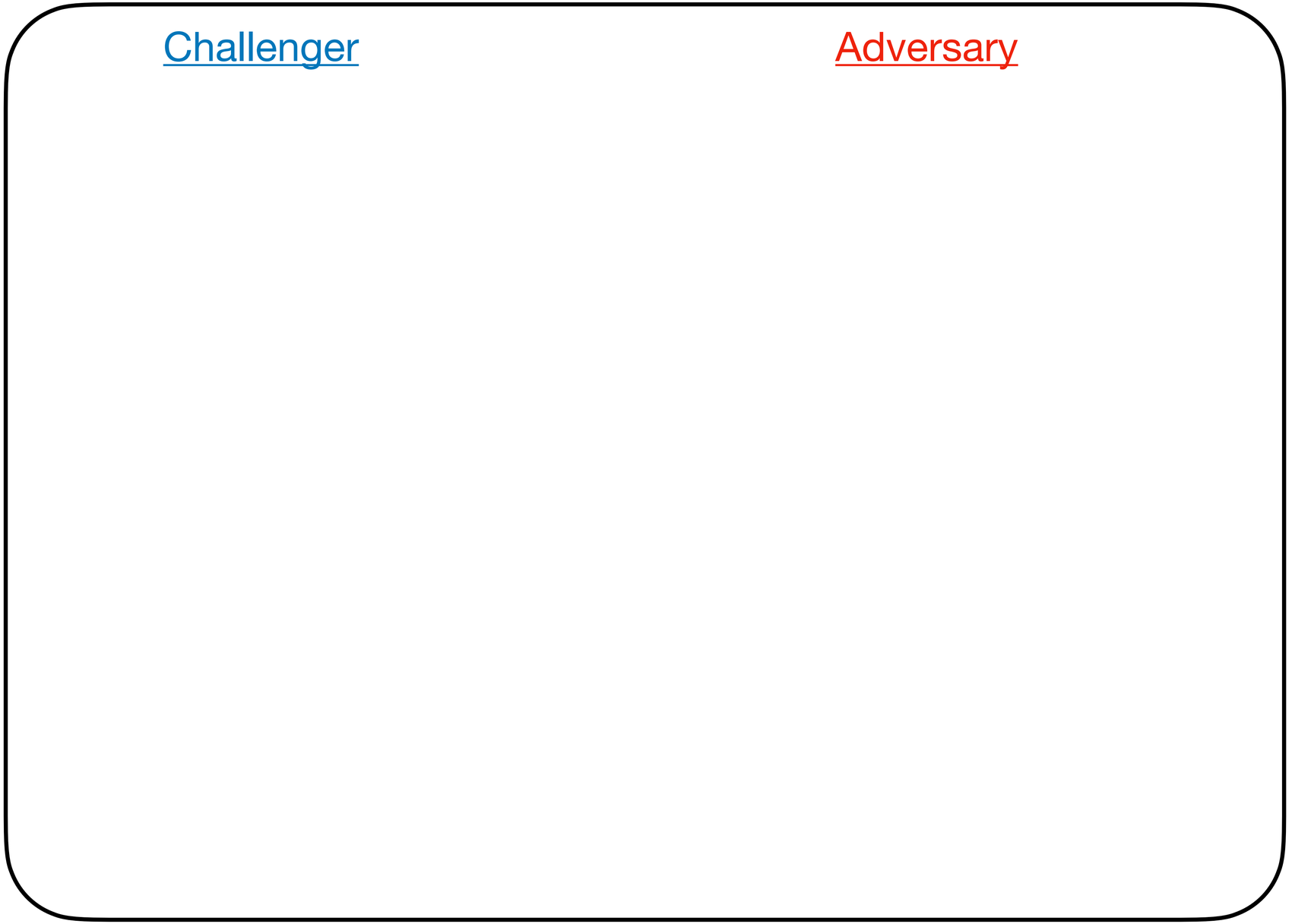
Our Family of Constraints: 1-Symbol Fixing (1-SF)

Defined by alphabet set Σ and length ℓ
$|\Sigma|, \ell$: poly(sec. par.)

Constraints parameterized by $i \in [\ell]$ and $z \in \Sigma$



Adaptive Security for 1-SF-PRF



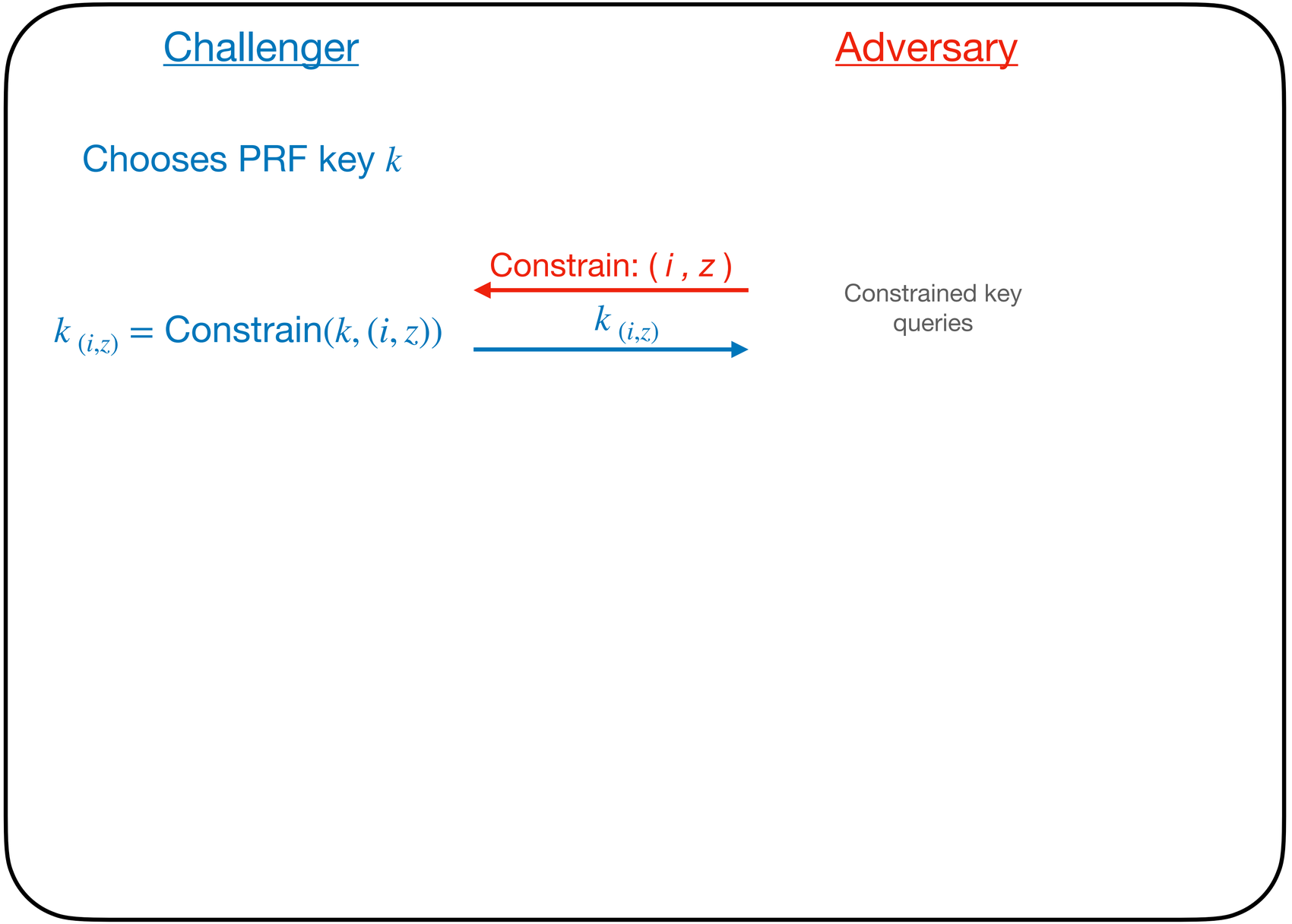
Adaptive Security for 1-SF-PRF

Challenger

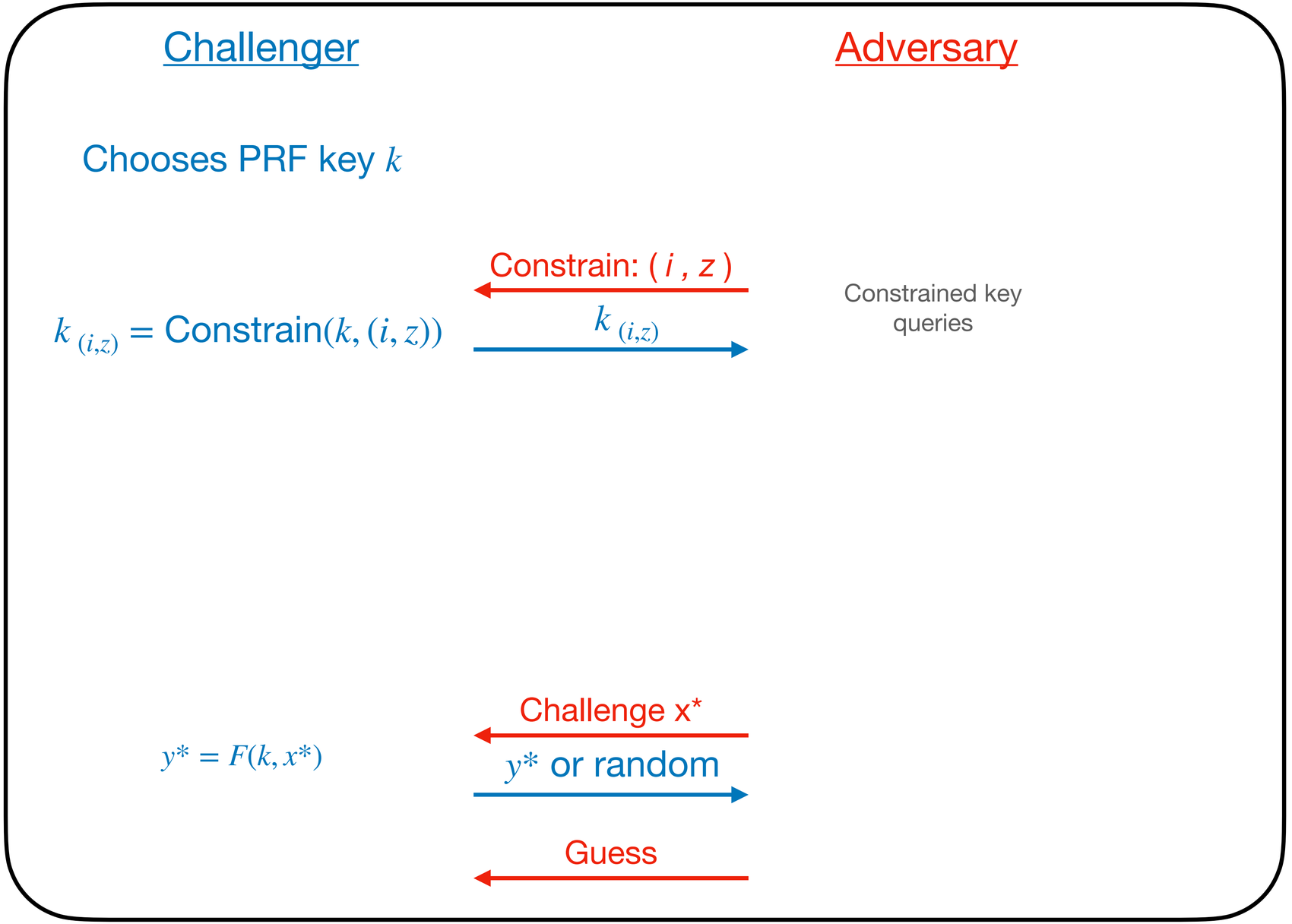
Adversary

Chooses PRF key k

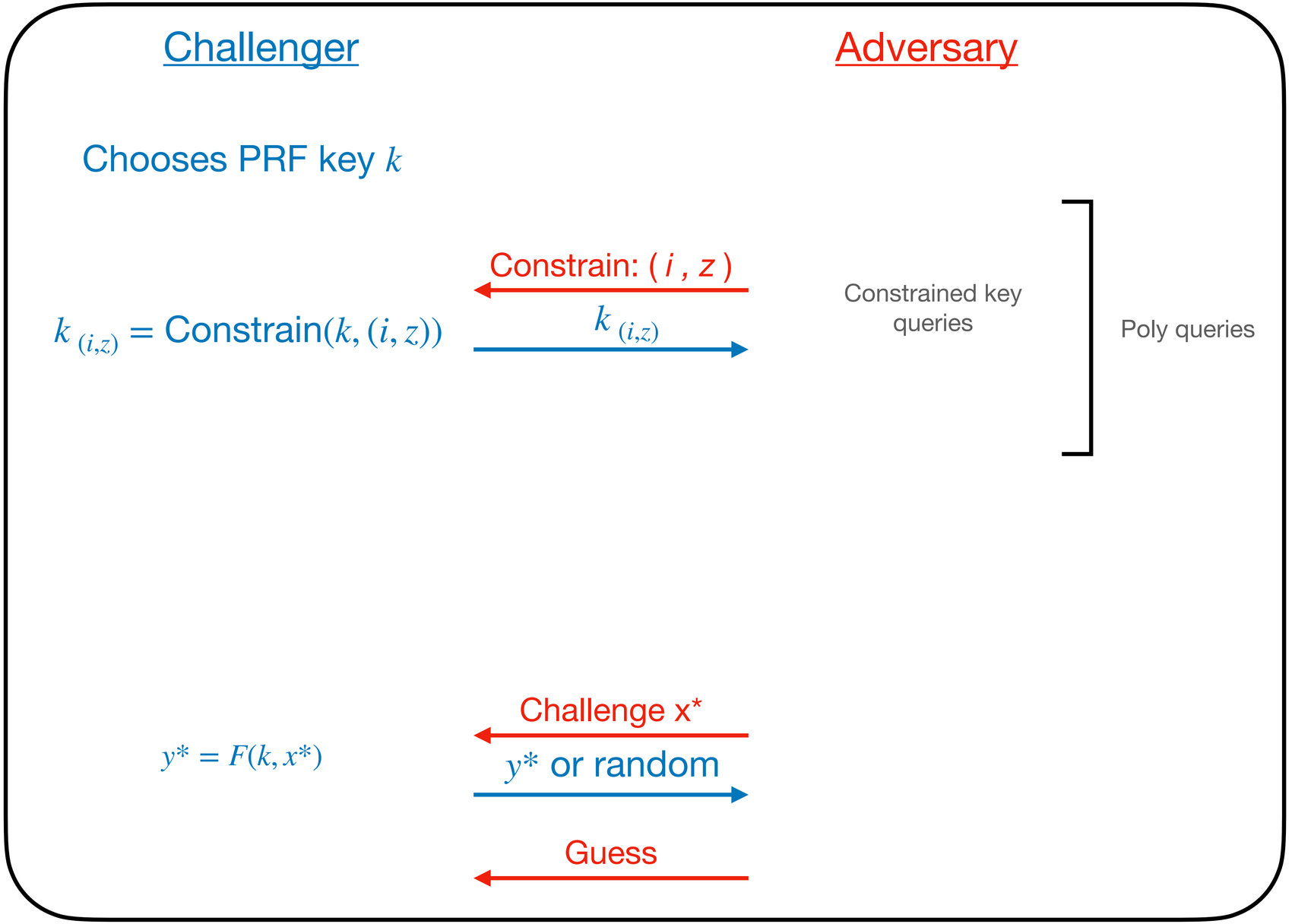
Adaptive Security for 1-SF-PRF



Adaptive Security for 1-SF-PRF



Adaptive Security for 1-SF-PRF



Constructing Adaptively Secure 1-SF-PRF

Our approach: iO + algebraic PRFs

Naor-Reingold PRF

Our Construction

PRF key k : exponents $\{e_{i,w}$ for each $i \in [\ell], w \in \Sigma\}$, group element $h \in \mathbb{G}$

$$F(k, (x_1, \ldots, x_\ell)) = h^{\prod_i e_{i,x_i}}$$

Our Construction

Constrained key for $(\ i^*\ ,\ z\ )$: $iO\big(\ \text{Prog-}(i^*, z)\ \big)$

$(x_1, x_2, \ldots, x_\ell)$ →

Prog-$(\ i^*, z\ )$

Constants: exponents $e_{i,w}$ for all $i \neq i^*, w \in \Sigma$

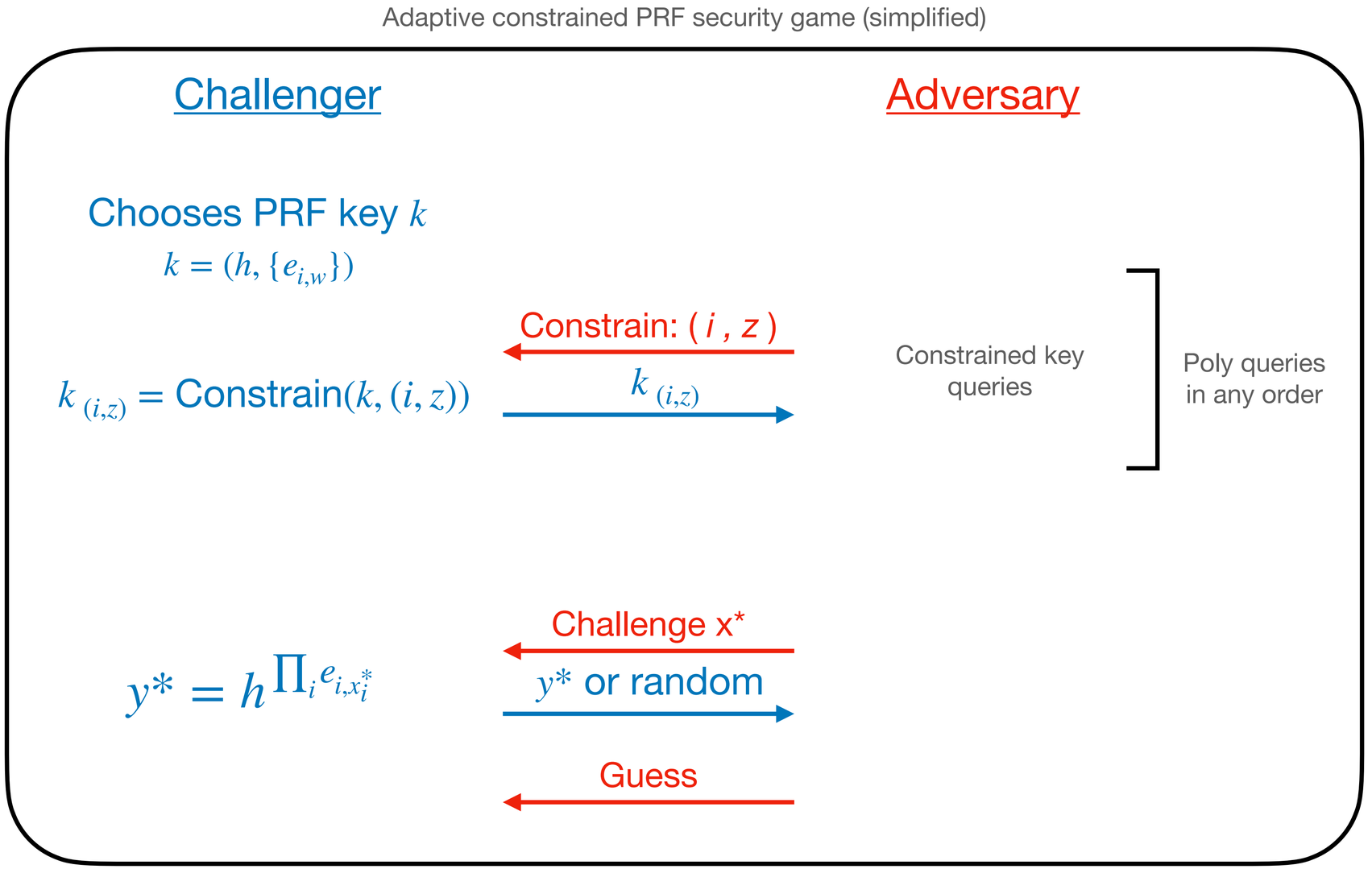
exponent $e_{\ i^*,z}$

group element $h \in \mathbb{G}$

If $x_{i^*} \neq z$, output $\perp$.

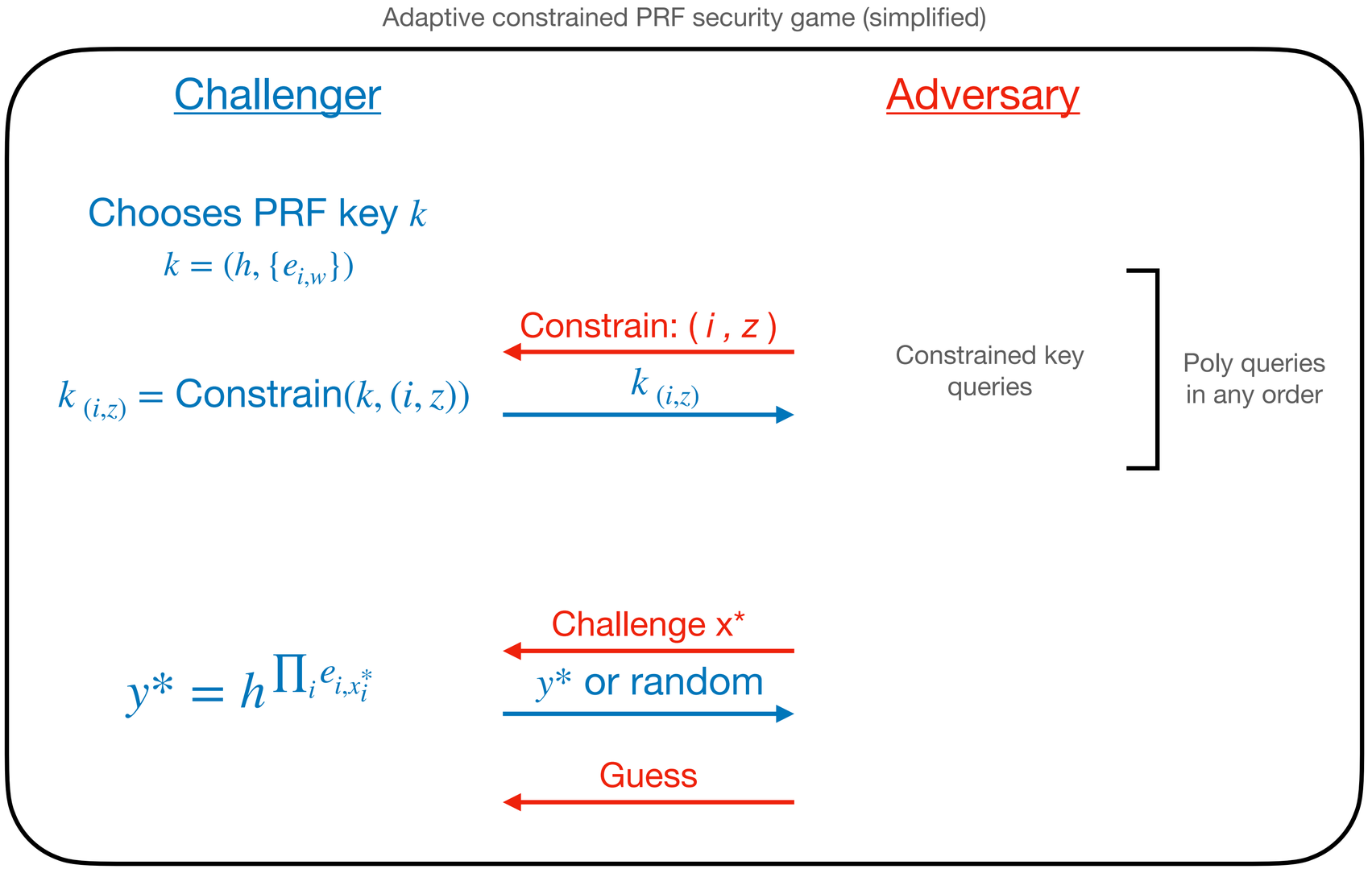
Else compute $v = \prod e_{\ i,x_i}$, output h^v .

Proof Idea:


Adaptive constrained PRF security game (simplified)
Challenger
Adversary
Chooses PRF key k
$k = (h, \{e_{i,w}\})$
$k_{(i,z)} = \text{Constrain}(k, (i, z))$
Constrain: (i , z)
$k_{(i,z)}$
Constrained key queries
Poly queries in any order
$y^* = h^{\prod_i e_{i,x_i^*}}$
Challenge x*
y^* or random
Guess

Proof Idea:

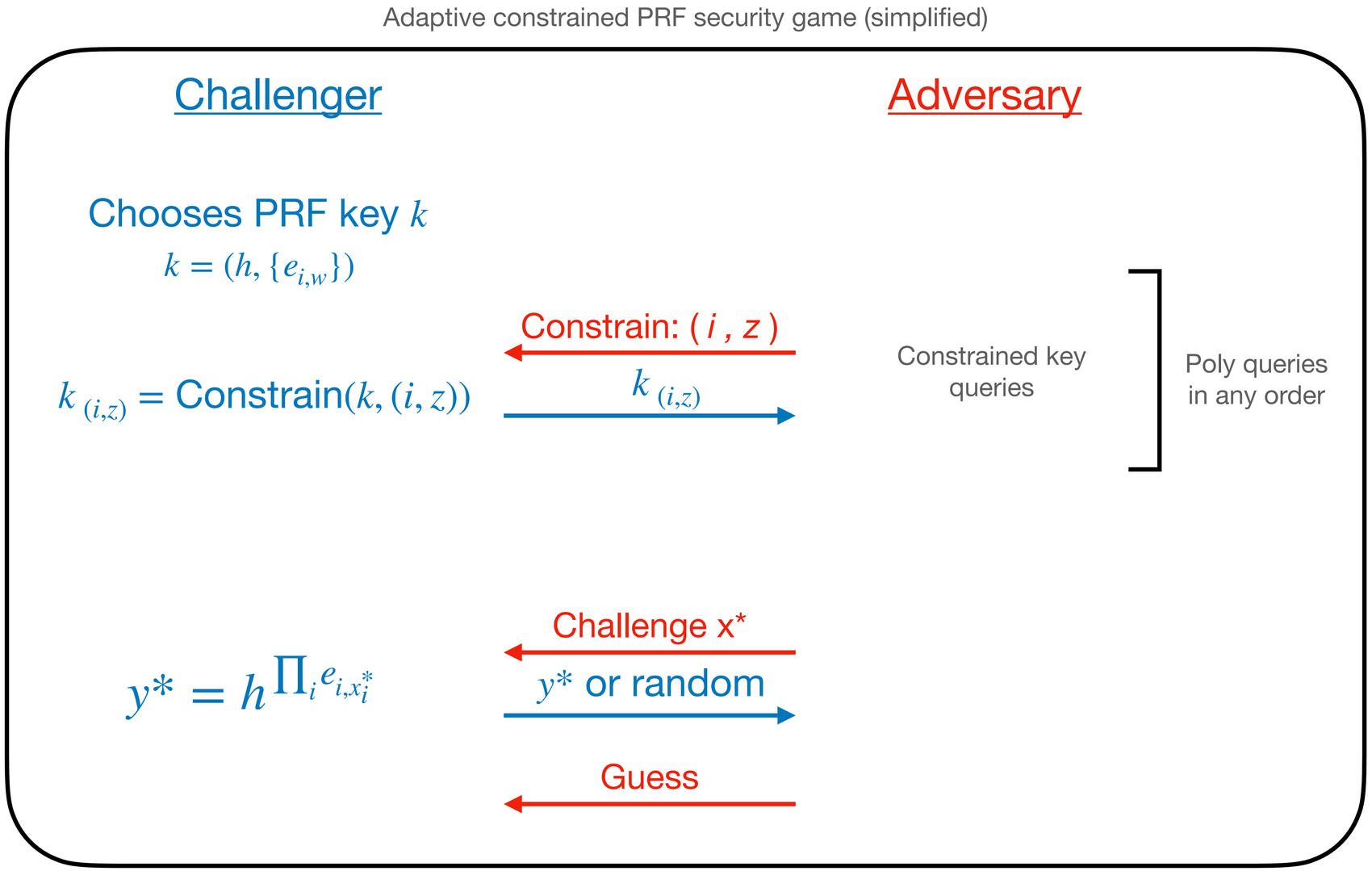
Group element h is contained in the constrained keys.



Proof Idea:

Group element h is contained in the constrained keys.

Can we remove h from the constrained keys?



Idea 1: A different h_i for each constrained key

i^{th} Constrained key for $(\ i^*\ ,\ z\ )$: $iO\big(\ \text{Prog}_i - (i^*, z)\ \big)$

$$\text{Prog}_i - (\ i^*, z\ )$$

Constants: exponents $e_{i,w}$ for all $i \neq i^*, w \in \Sigma$

exponent $e_{\ i^*,z}$

group element $h_i \in \mathbb{G}$

If $x_{i^*} \neq z$, output $\perp$.

Else compute $v = \prod e_{\ i,x_i}$, output h_i^v .

Idea 1: A different h_i for each constrained key

i^{th} Constrained key for $(\ i^*\ ,\ z\ )$: $iO\big(\ \text{Prog}_i - (i^*, z)\ \big)$

The constrained keys need to be consistent !

$$\text{Prog}_i - (\ i^*, z\ )$$

Constants: exponents $e_{i,w}$ for all $i \neq i^*, w \in \Sigma$

exponent $e_{\ i^*,z}$

group element $h_i \in \mathbb{G}$

If $x_{i^*} \neq z$, output $\perp$.

Else compute $v = \prod e_{\ i,x_i}$, output h_i^v .

Idea 1: A different h_i for each constrained key

$x = (z, z, \ldots)$

1^{st} key for constraint $(1\ ,\ z)$

2^{nd} key for constraint $(2\ ,\ z)$

$\text{Prog}_1\text{-}(1\ ,\ z)$

Constants: exponents $e_{i,w}$ for all $i \neq 1, w \in \Sigma$

exponent $e_{1,z}$ $\quad$ $h_1 \in \mathbb{G}$

If $x_2 \neq z$, output $\perp$.

Else compute $v = \prod e_{i,x_i}$, output h_1^v.

$\text{Prog}_1\text{-}(2\ ,\ z)$

Constants: exponents $e_{i,w}$ for all $i \neq 2, w \in \Sigma$

exponent $e_{2,z}$ $\quad$ $h_2 \in \mathbb{G}$

If $x_2 \neq z$, output $\perp$.

Else compute $v = \prod e_{i,x_i}$, output h_2^v.

h_1^v $\quad$ Inconsistent outputs ! $\quad$ h_2^v

Idea 2: In i^{th} key, hardwire all previous h_j and their constraints

Modify program logic to maintain consistency across keys

2^{nd} key : constraint (i^*, z)

Prog_2-(i^*, z)

Constants: exponents $e_{i,w}$ for all $i \neq i^*, w \in \Sigma$

exponent $e_{i^*,z}$ $\quad$ $h_2 \in \mathbb{G}$

$h_1 \in \mathbb{G}$ $\quad$ constraint (i_1, z_1) for 1st query

If $x_2 \neq z$, output $\perp$.

If $x_{i_1} = z_1$, compute $v = \prod e_{i,x_i}$, output h_1^v .

Else compute $v = \prod e_{i,x_i}$, output h_2^v .

second constrained key is consistent with the first key

Summary of Proof

Summary of Proof

Using iO + DDH, we can switch the constrained keys one by one to not use h (but the challenge PRF evaluation still uses h)

Summary of Proof

Using iO + DDH, we can switch the constrained keys one by one to not use h (but the challenge PRF evaluation still uses h)

PRF evaluation at all points not covered by the constrained keys is undefined
Therefore PRF not fixed even after the constrained key queries

Summary of Proof

Using iO + DDH, we can switch the constrained keys one by one to not use h (but the challenge PRF evaluation still uses h)

PRF evaluation at all points not covered by the constrained keys is undefined
Therefore PRF not fixed even after the constrained key queries

Bypasses Rao’s impossibility result

Open Questions:

Open Questions:

Remove the bound restriction on group size?

Open Questions:

Remove the bound restriction on group size?

Open Questions:

Remove the bound restriction on group size?

Remove bound on corruptions?

Open Questions:

Remove the bound restriction on group size?

Remove bound on corruptions?

Thank you!