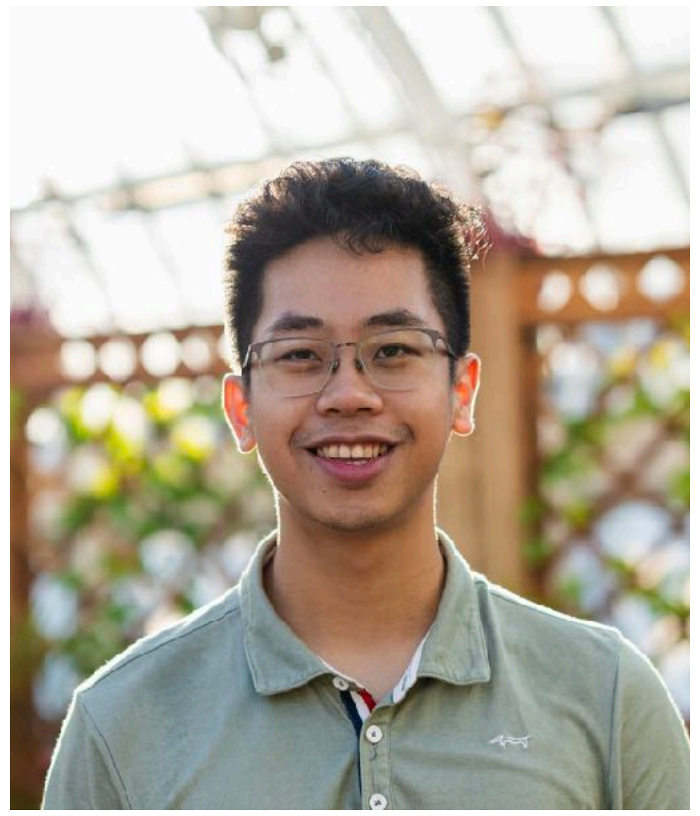
Non-Interactive Zero-Knowledge from LPN and MQ

Quang Dao

Aayush Jain

Zhengzhong Jin

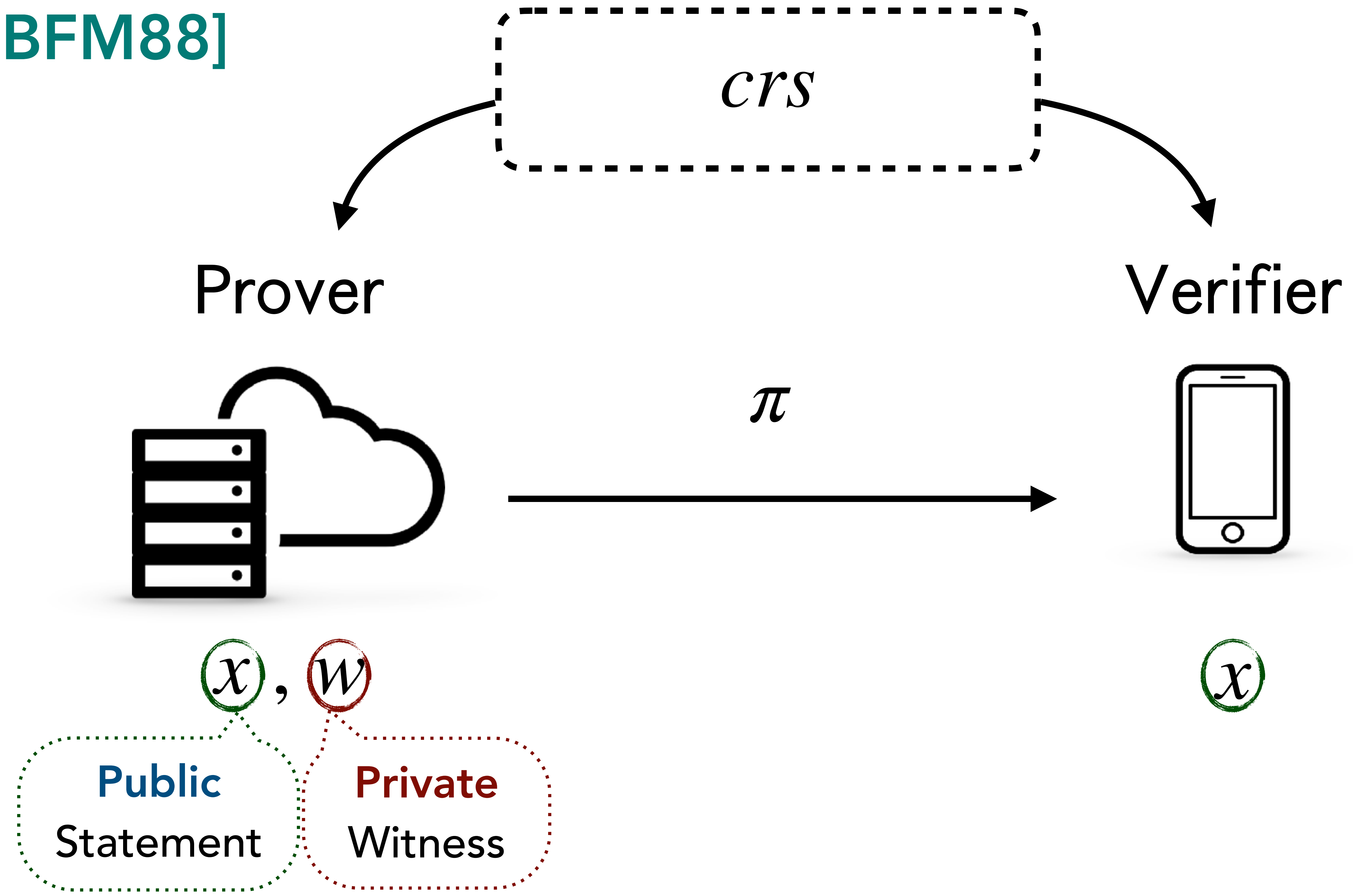


Crypto 2024

Non-Interactive Zero-Knowledge (NIZK)

Non-Interactive Zero-Knowledge (NIZK)

Fundamental notion in cryptography [GMR85, BFM88]



crs

Prover

Verifier

π

x, w

x

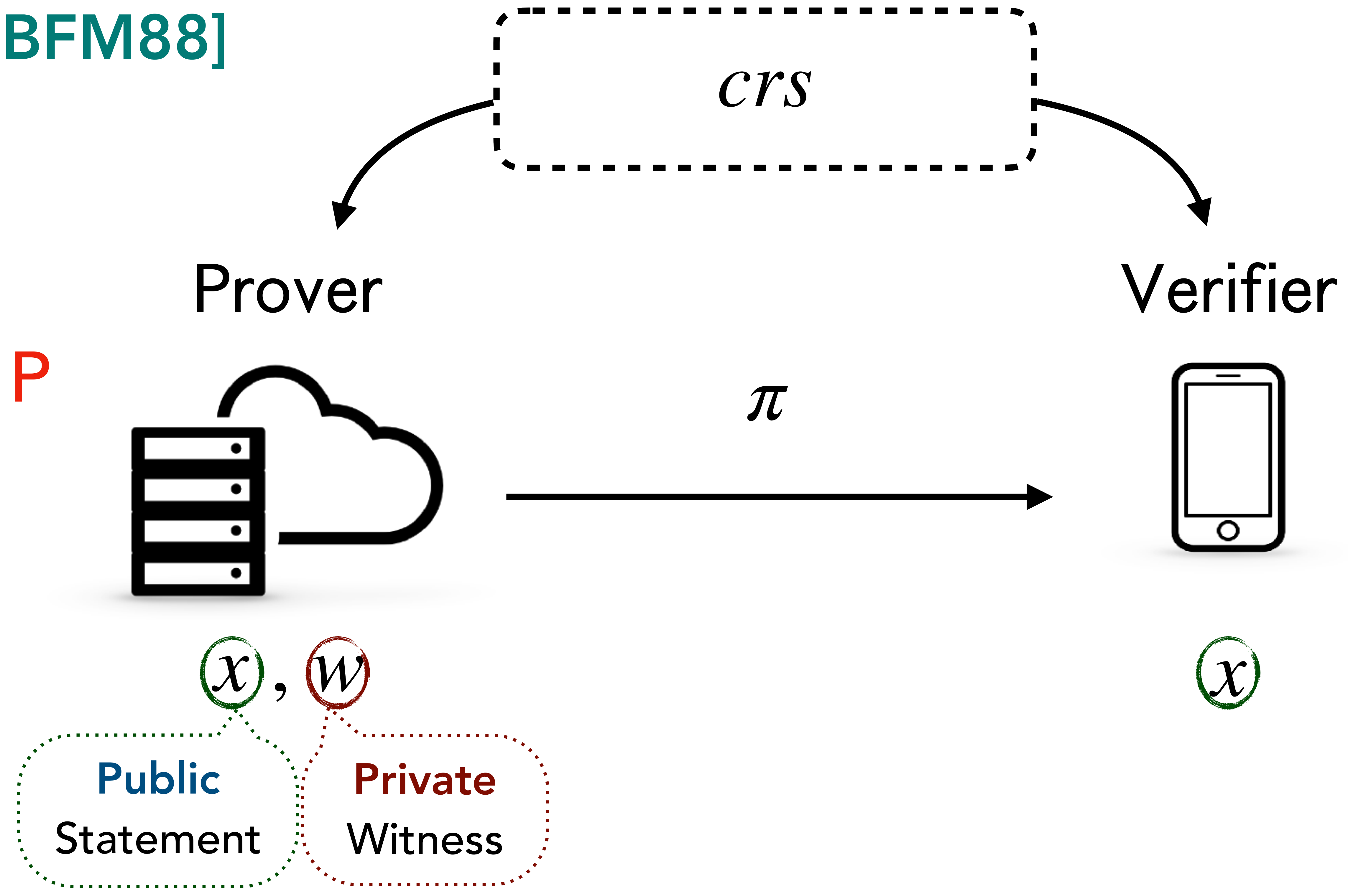
Public Statement

Private Witness

Non-Interactive Zero-Knowledge (NIZK)

Fundamental notion in cryptography [GMR85, BFM88]

- **Completeness:** honest P convinces V
- **Soundness:** V rejects $x \notin L$ for any malicious P
- **Zero-knowledge:** there exists a simulator S that can simulate (crs, π)



crs

Prover

Verifier

π

x, w

x

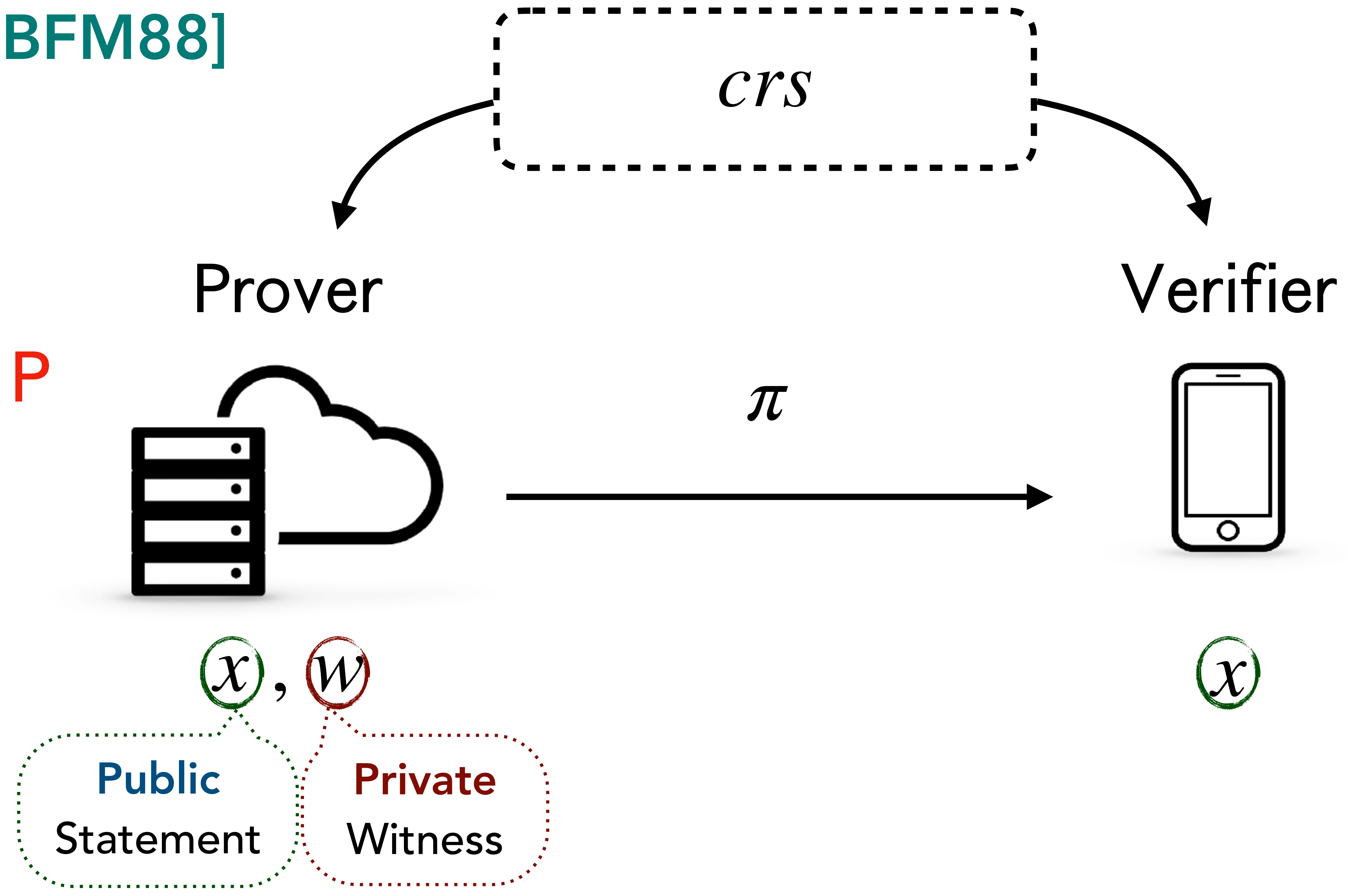
Public Statement

Private Witness

Non-Interactive Zero-Knowledge (NIZK)

Fundamental notion in cryptography [GMR85, BFM88]

- **Completeness:** honest P convinces V
- **Soundness:** V rejects $x \notin L$ for any malicious P
- **Zero-knowledge:** there exists a simulator S that can simulate (crs, π)



Many practical applications!

Private Cryptocurrencies

Voting Systems

Anonymous credentials

ZK-Rollups

Proving Image Transformations

...and more!

Achieving NIZK in Practice and Theory

Achieving NIZK in Practice and Theory

Practice: use random oracles [Kilian94, Micali00, BCS16, etc] , idealized group models, or non-falsifiable assumptions [GGPR13, Groth16, etc]

Achieving NIZK in Practice and Theory

Practice: use random oracles [Kilian94, Micali00, BCS16, etc] , idealized group models, or non-falsifiable assumptions [GGPR13, Groth16, etc]

Theory: surprisingly difficult to construct!

Achieving NIZK in Practice and Theory

Practice: use random oracles [Kilian94, Micali00, BCS16, etc] , idealized group models, or non-falsifiable assumptions [GGPR13, Groth16, etc]

Theory: surprisingly difficult to construct!

- From factoring / QR [BFM88, FLS90] or bilinear maps [CHK03, GOS06]

Achieving NIZK in Practice and Theory

Practice: use random oracles [Kilian94, Micali00, BCS16, etc] , idealized group models, or non-falsifiable assumptions [GGPR13, Groth16, etc]

Theory: surprisingly difficult to construct!

- From factoring / QR [BFM88, FLS90] or bilinear maps [CHK03, GOS06]
- Recent progress relies on ***correlation-intractable*** **(CI)** hash functions!

$\implies$ enables constructions from LWE [CCR+19, PS19], DDH/DCR + LPN [BKM20], or sub-exponential DDH [JJ21]

Achieving NIZK in Practice and Theory

Practice: use random oracles [Kilian94, Micali00, BCS16, etc] , idealized group models, or non-falsifiable assumptions [GGPR13, Groth16, etc]

Theory: surprisingly difficult to construct!

- From factoring / QR [BFM88, FLS90] or bilinear maps [CHK03, GOS06]
- Recent progress relies on ***correlation-intractable*** **(CI)** hash functions!

  $\Longrightarrow$ enables constructions from LWE [CCR+19, PS19], DDH/DCR + LPN [BKM20], or sub-exponential DDH [JJ21]
- The only post-quantum secure construction is from LWE!

Why Search for New Post-Quantum NIZK?

Why Search for New Post-Quantum NIZK?

1. Go beyond lattices & diversify constructions:

- Lack of post-quantum advanced cryptography from ***non-lattice-based*** assumptions

Why Search for New Post-Quantum NIZK?

1. Go beyond lattices & diversify constructions:

- Lack of post-quantum advanced cryptography from ***non-lattice-based*** assumptions

2. Achieve NIZK under “weaker” assumptions:

- Existing LWE-based constructions (w/ polynomial modulus) rely on FHE techniques*

* [Waters24] constructs NIZK from LWE with sub-exponential modulus

Why Search for New Post-Quantum NIZK?

1. **Go beyond lattices & diversify constructions:**
    - Lack of post-quantum advanced cryptography from *non-lattice-based* assumptions
2. **Achieve NIZK under "weaker" assumptions:**
    - Existing LWE-based constructions (w/ polynomial modulus) rely on FHE techniques*
3. **Stepping stone towards BARGs, SNARGs, etc.**

* [Waters24] constructs NIZK from LWE with sub-exponential modulus

Why Search for New Post-Quantum NIZK?

1. **Go beyond lattices & diversify constructions:**
    - Lack of post-quantum advanced cryptography from ***non-lattice-based*** assumptions
2. **Achieve NIZK under "weaker" assumptions:**
    - Existing LWE-based constructions (w/ polynomial modulus) rely on FHE techniques*
3. **Stepping stone towards BARGs, SNARGs, etc.**

Can we build NIZK from post-quantum assumptions other than lattices?

* [Waters24] constructs NIZK from LWE with sub-exponential modulus

Our Result: NIZK from LPN and MQ

Our Result: NIZK from LPN and MQ

1. We construct NIZK† from: † with computational soundness + zero-knowledge
   - Learning Parity with Noise (LPN), * with slightly-stronger-than-PKE noise rate
   - Approximate Multivariate Quadratic (ApxMQ) * implied by MQ with exponential hardness

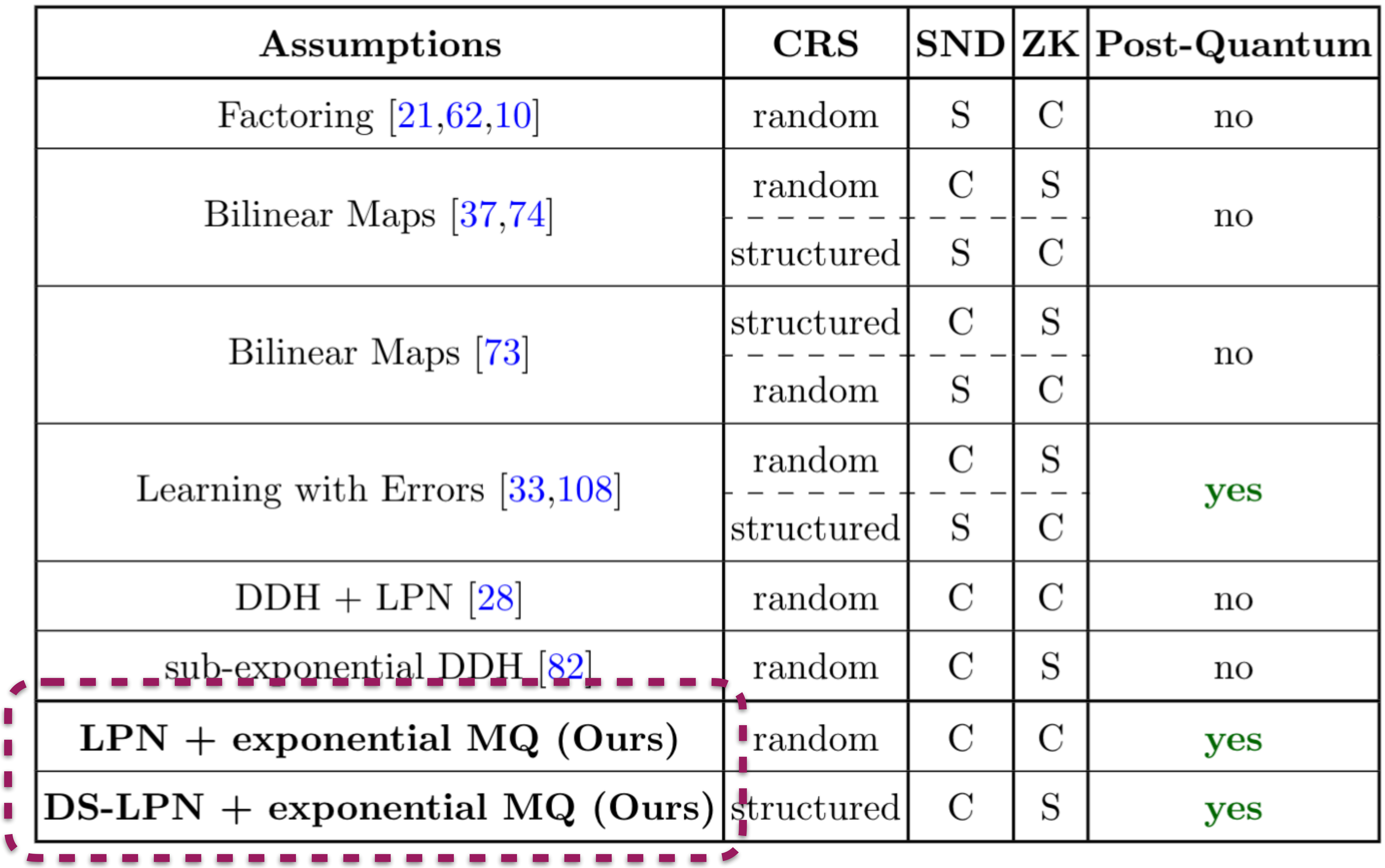
Our Result: NIZK from LPN and MQ

1. We construct NIZK[†] from: [†] with computational soundness + zero-knowledge
    - Learning Parity with Noise (LPN), * with slightly-stronger-than-PKE noise rate
    - Approximate Multivariate Quadratic (ApxMQ) * implied by MQ with exponential hardness
2. Our NIZK is achieved via an ***extremely simple*** construction of CI hashing:
    - For functions that can be approximated by *concatenated constant-degree polynomials*
    - Proof of CI reduces to hardness of Approximate MQ, or its higher-degree analogue

Our Result: NIZK from LPN and MQ

1. We construct NIZK[†] from: [†] with computational soundness + zero-knowledge
   - Learning Parity with Noise (**LPN**), * with slightly-stronger-than-PKE noise rate
   - Approximate Multivariate Quadratic (**ApxMQ**) * implied by MQ with exponential hardness
2. Our **NIZK** is achieved via an ***extremely simple*** construction of **CI hashing**:
   - For functions that can be approximated by *concatenated constant-degree polynomials*
   - Proof of **CI** reduces to hardness of **Approximate MQ**, or its higher-degree analogue
3. We can upgrade our **NIZK** to statistical zero-knowledge, assuming:
   - **Dense-Sparse LPN** [DJ24] * implies Lossy PKE with ≈ linear decryption & low correct. error

Our Result: NIZK from LPN and MQ



Assumptions	CRS	SND	ZK	Post-Quantum
Factoring [21,62,10]	random	S	C	no
Bilinear Maps [37,74]	random	C	S	no
	structured	S	C	
Bilinear Maps [73]	structured	C	S	no
	random	S	C	
Learning with Errors [33,108]	random	C	S	yes
	structured	S	C	
DDH + LPN [28]	random	C	C	no
sub-exponential DDH [82]	random	C	S	no
LPN + exponential MQ (Ours)	random	C	C	yes
DS-LPN + exponential MQ (Ours)	structured	C	S	yes

Talk Outline

1. Recap: NIZK from Correlation Intractability

2. CI Hashing from (Approximate) MQ

3. Putting Things Together

Talk Outline

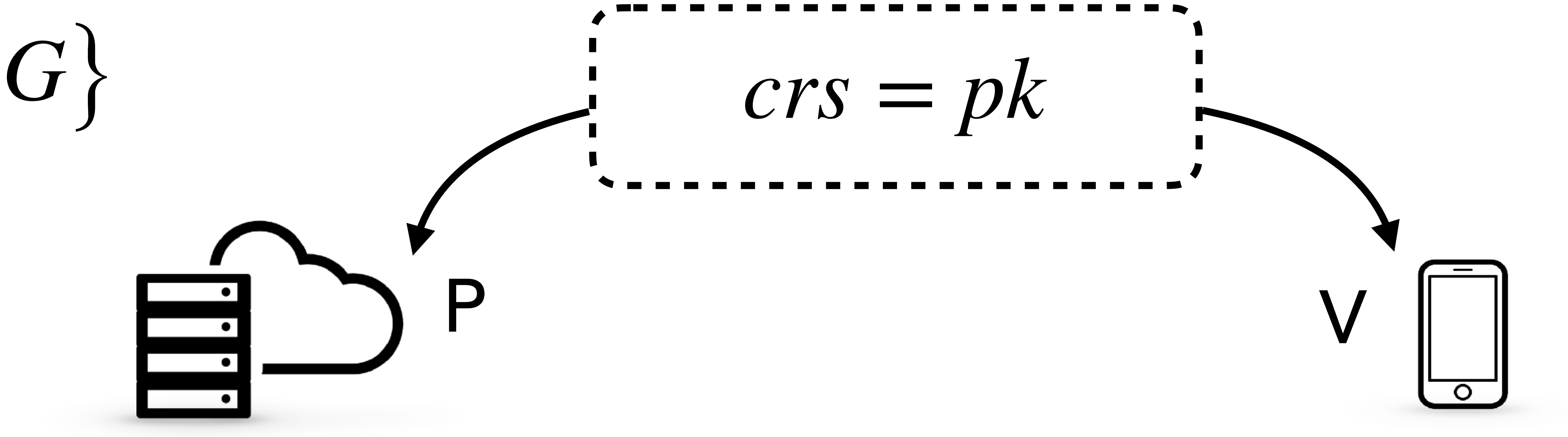
1. Recap: NIZK from Correlation Intractability
2. CI Hashing from (Approximate) MQ
3. Putting Things Together

Recap: Blum’s Hamiltonicity protocol

Recap: Blum's Hamiltonicity protocol

Relation: $\left\{(G, H) \,|\, H \text{ is a Hamiltonian cycle of } G\right\}$

contains all vertices of G



$crs = pk$

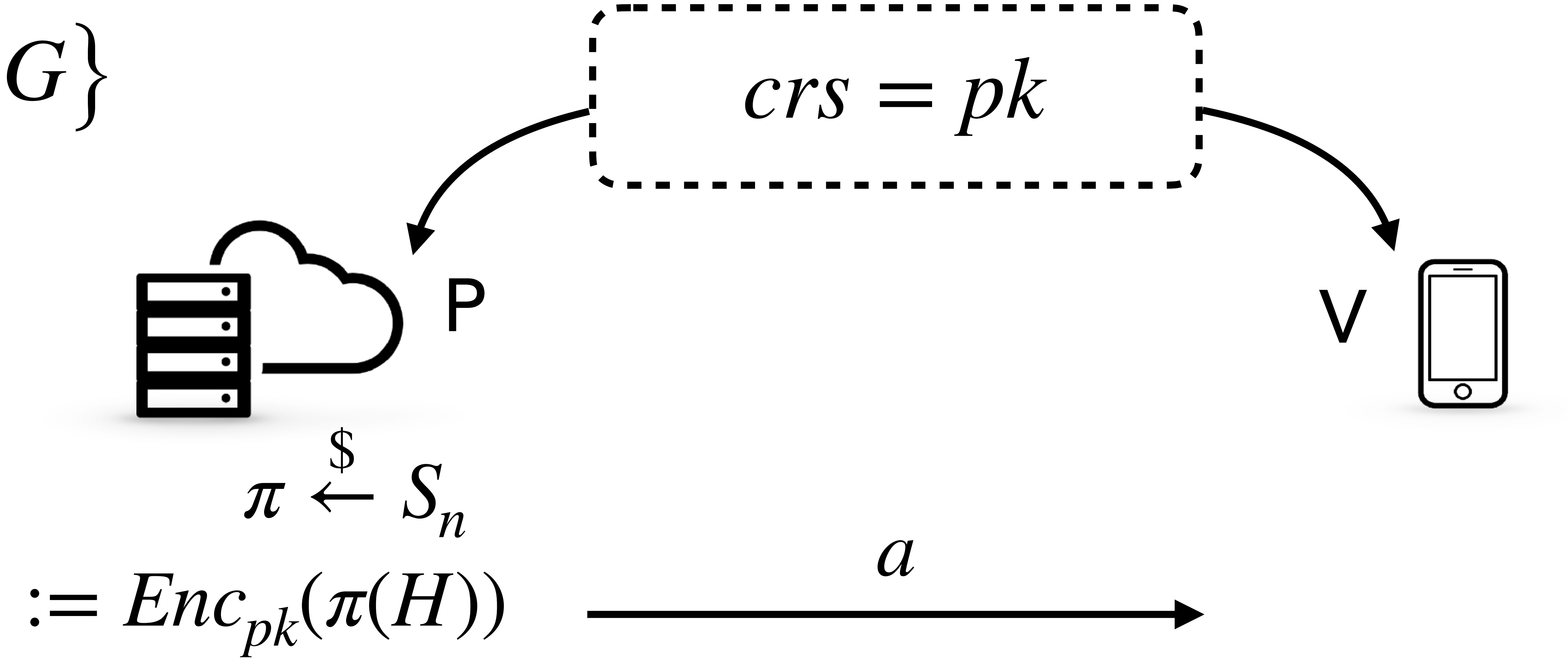
P

V

Recap: Blum's Hamiltonicity protocol

Relation: $\{(G, H) \,|\, H$ is a Hamiltonian cycle of $G\}$

contains all vertices of G



$crs = pk$

P

V

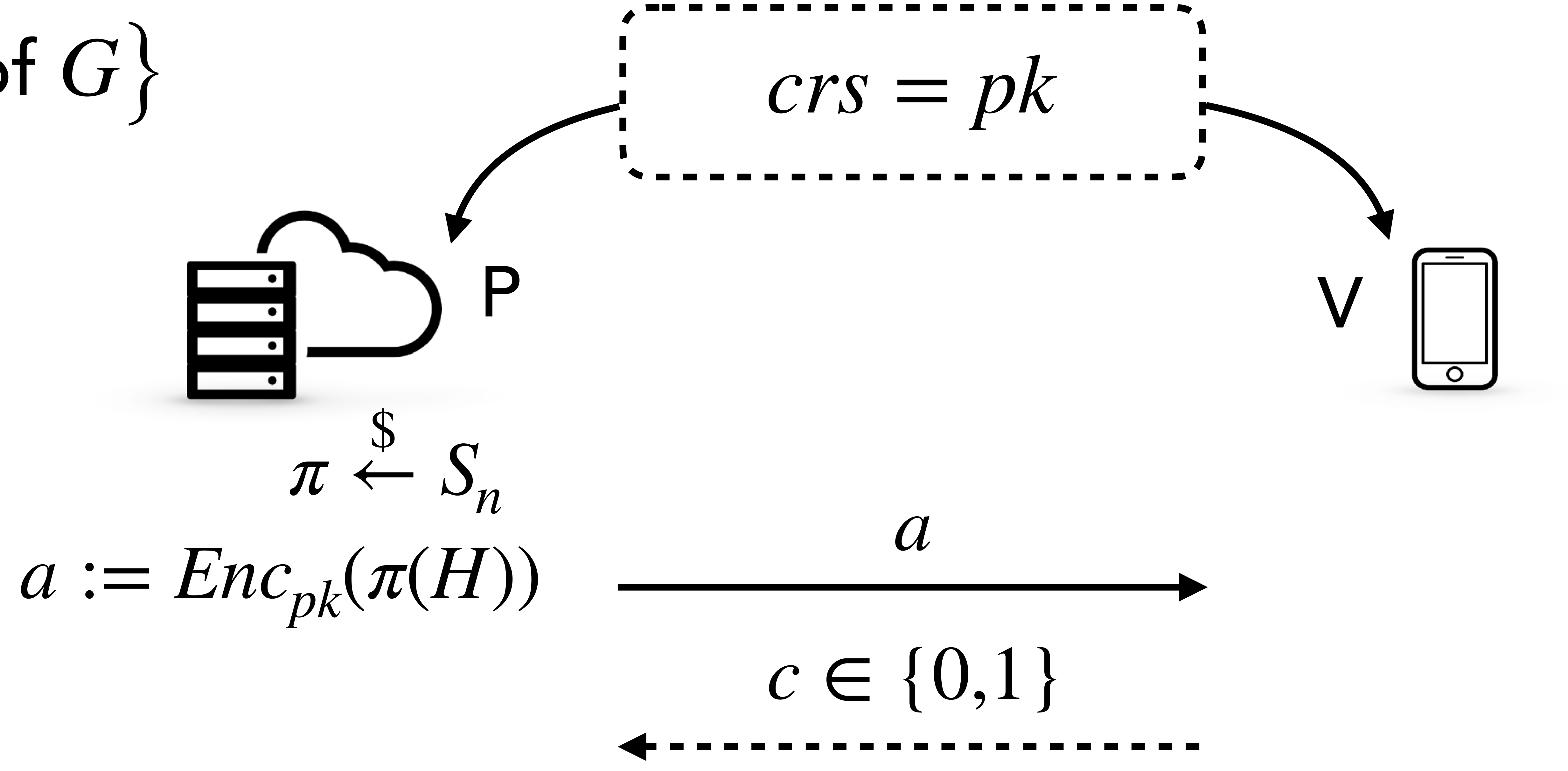
$\pi \overset{\$}{\leftarrow} S_n$

$a := Enc_{pk}(\pi(H)) \xrightarrow{a}$

Recap: Blum's Hamiltonicity protocol

Relation: $\{(G, H) \mid H$ is a Hamiltonian cycle of $G\}$

contains all vertices of G



$crs = pk$

P

V

$\pi \overset{\$}{\leftarrow} S_n$

$a := Enc_{pk}(\pi(H))$

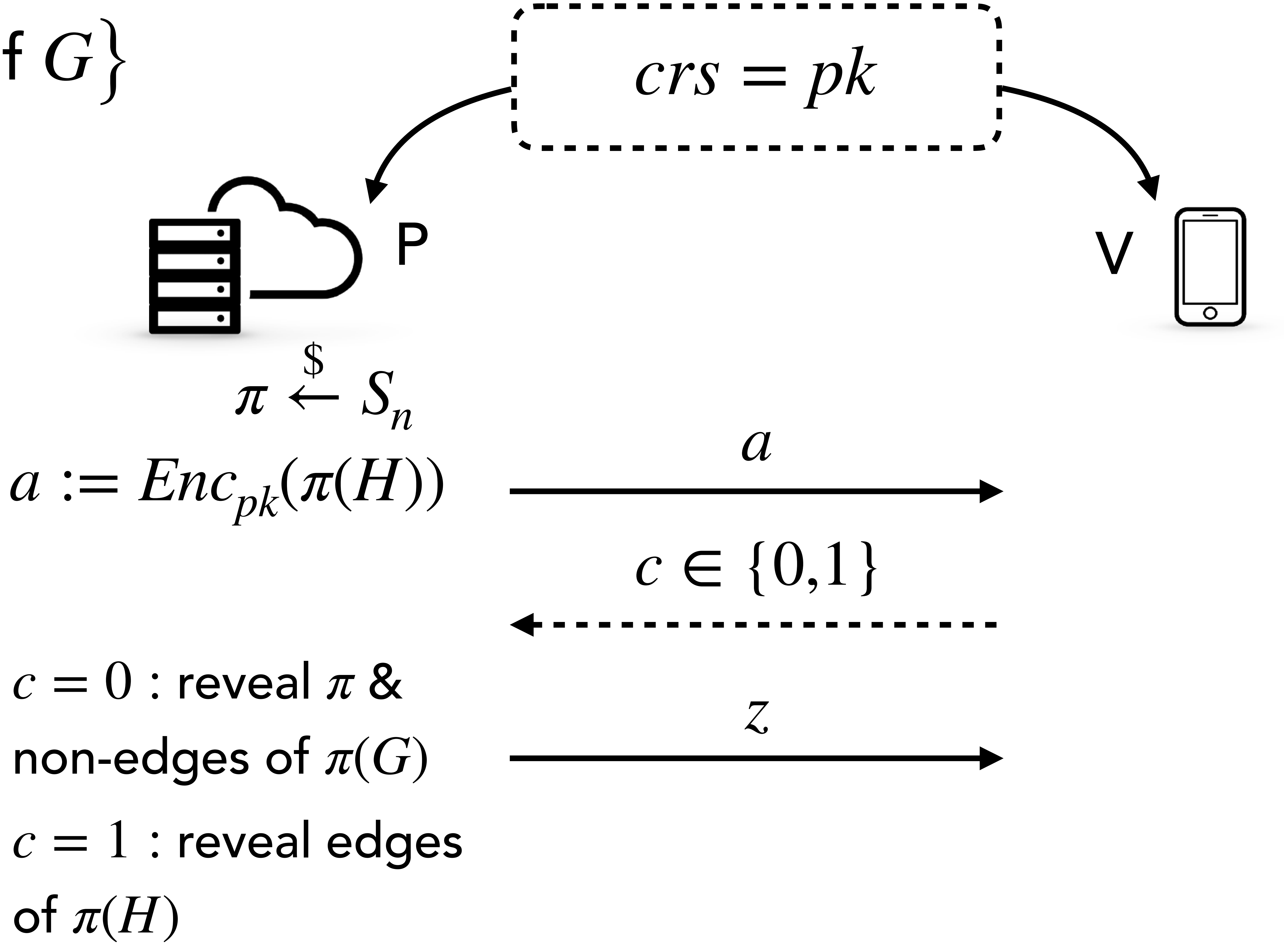
a

$c \in \{0,1\}$

Recap: Blum's Hamiltonicity protocol

Relation: $\{(G, H) \mid H$ is a Hamiltonian cycle of $G\}$

contains all vertices of G



$crs = pk$

P

V

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$ $\xrightarrow{\quad a \quad}$

$c \in \{0,1\}$

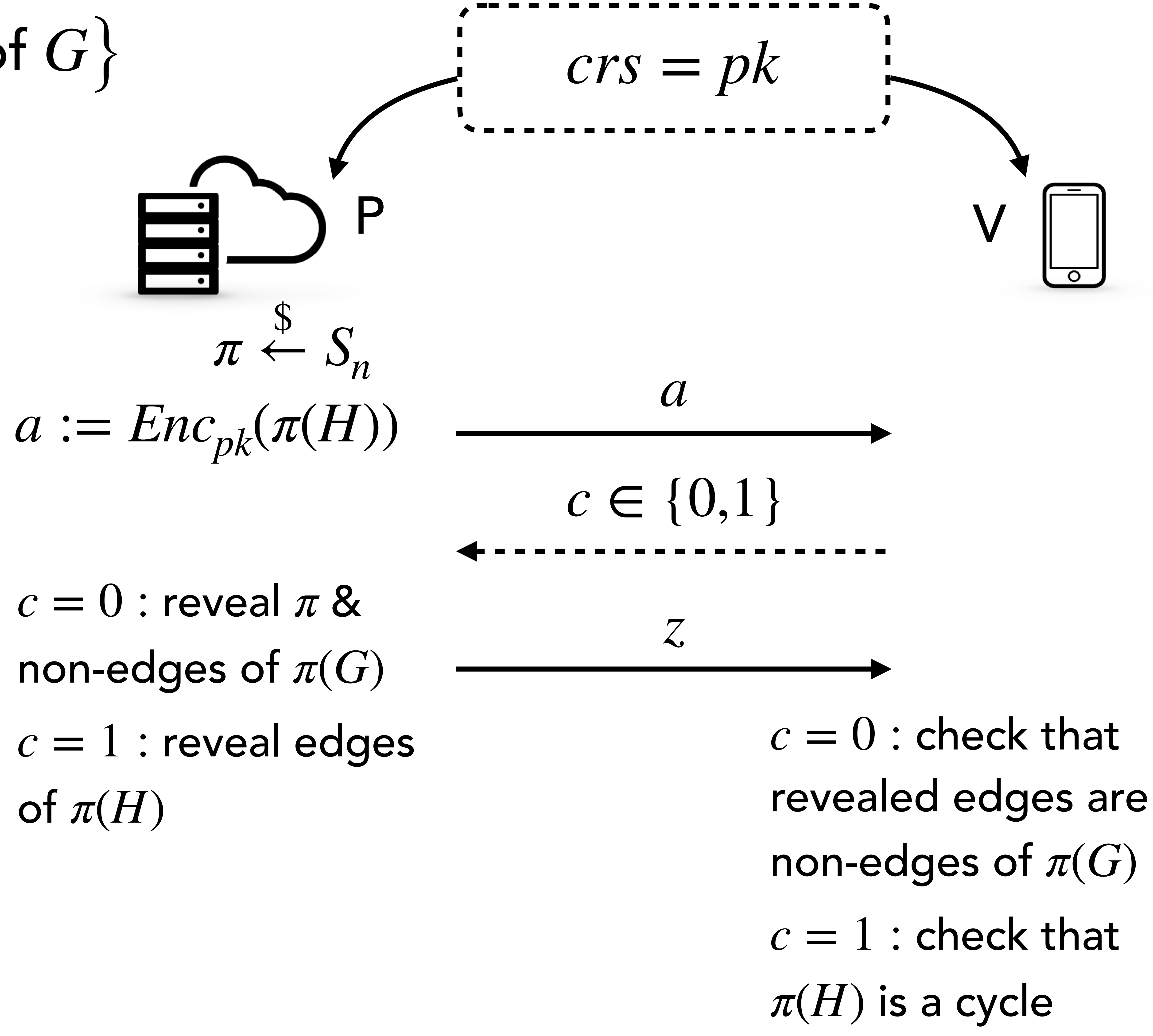
$c = 0$: reveal π & non-edges of $\pi(G)$ $\xrightarrow{\quad z \quad}$

$c = 1$: reveal edges of $\pi(H)$

Recap: Blum's Hamiltonicity protocol

Relation: $\{(G, H) \,|\, H$ is a Hamiltonian cycle of $G\}$

contains all vertices of G



$crs = pk$

P

V

$\pi \overset{\$}{\leftarrow} S_n$

$a := Enc_{pk}(\pi(H))$ $\xrightarrow{a}$

$\xleftarrow{c \in \{0,1\}}$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

$\xrightarrow{z}$

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

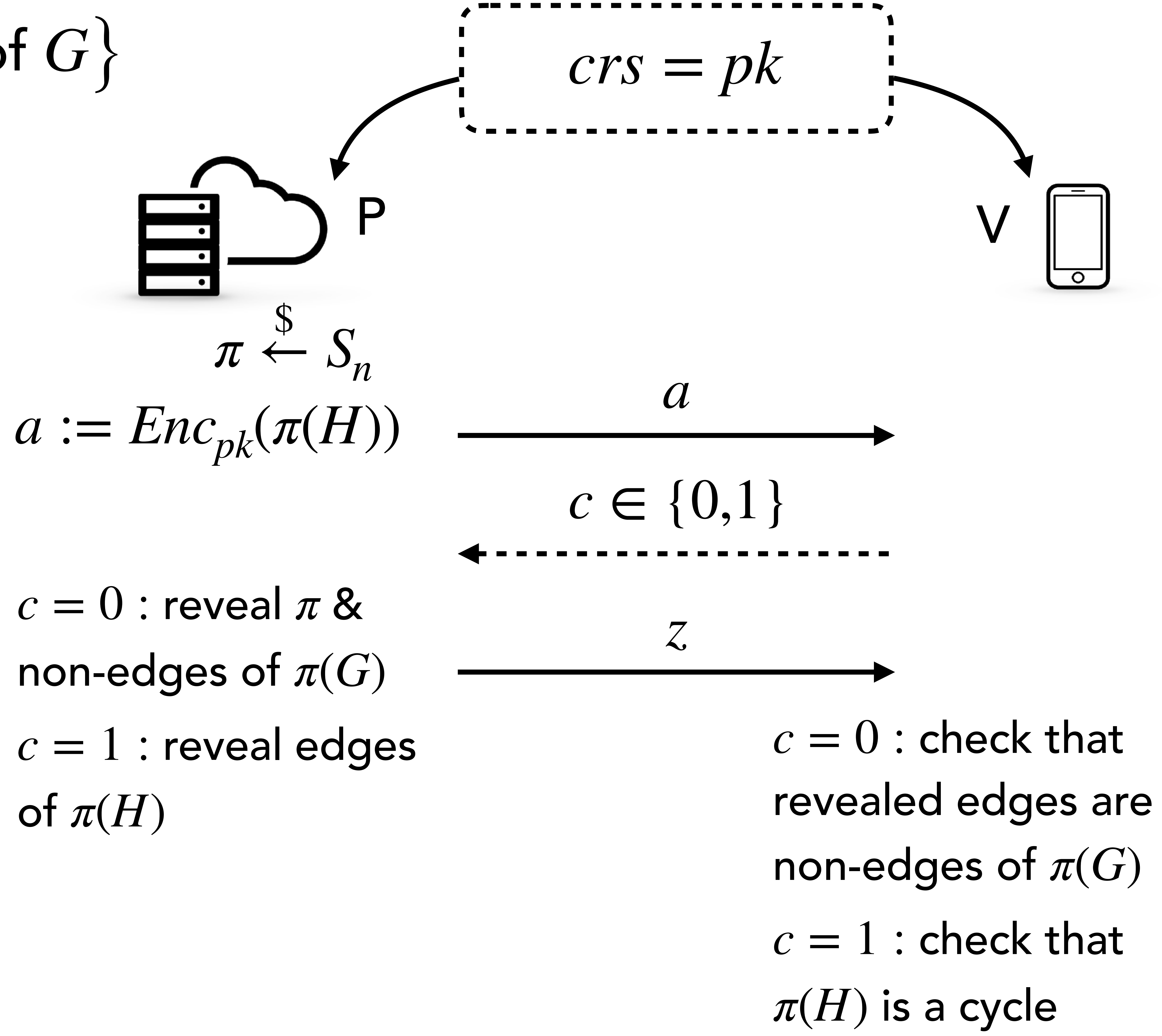
$c = 1$: check that $\pi(H)$ is a cycle

Recap: Blum's Hamiltonicity protocol

Relation: $\{(G,H)\,|\,H$ is a Hamiltonian cycle of $G\}$

contains all vertices of G

- **Perfect completeness**
- **Soundness error:** $1/2$
- **Honest-verifier zero-knowledge**



$crs = pk$

P

V

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$ $\xrightarrow{a}$

$\xleftarrow{c \in \{0,1\}}$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

$\xrightarrow{z}$

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

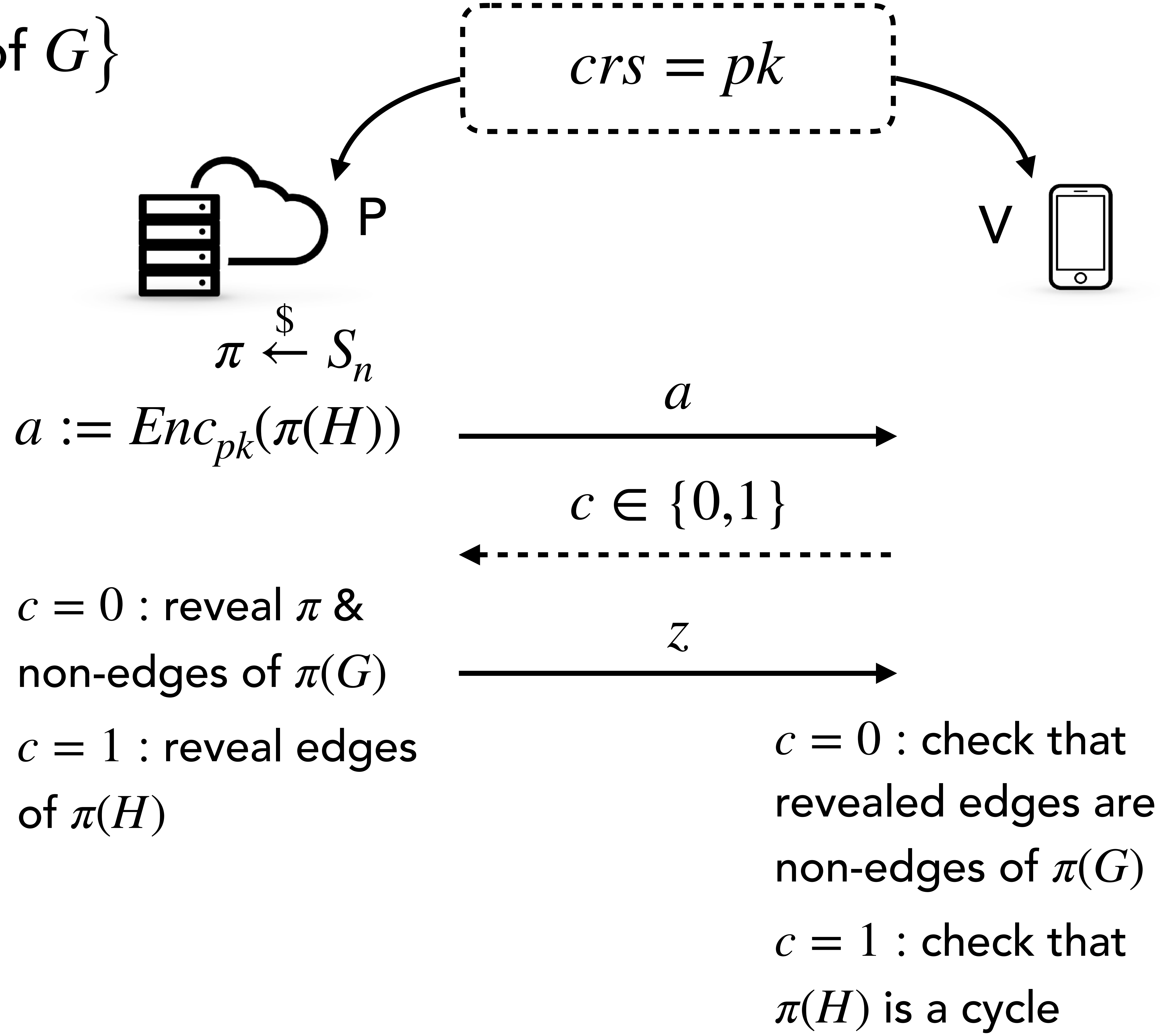
Recap: Blum's Hamiltonicity protocol

Relation: $\{(G, H) \mid H$ is a Hamiltonian cycle of $G\}$

contains all vertices of G

- Perfect completeness
- Soundness error: $1/2$
- Honest-verifier zero-knowledge

NIZK from Fiat-Shamir [FS86]?



$crs = pk$

P

V

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c \in \{0,1\}$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

Recap: Blum's Hamiltonicity protocol

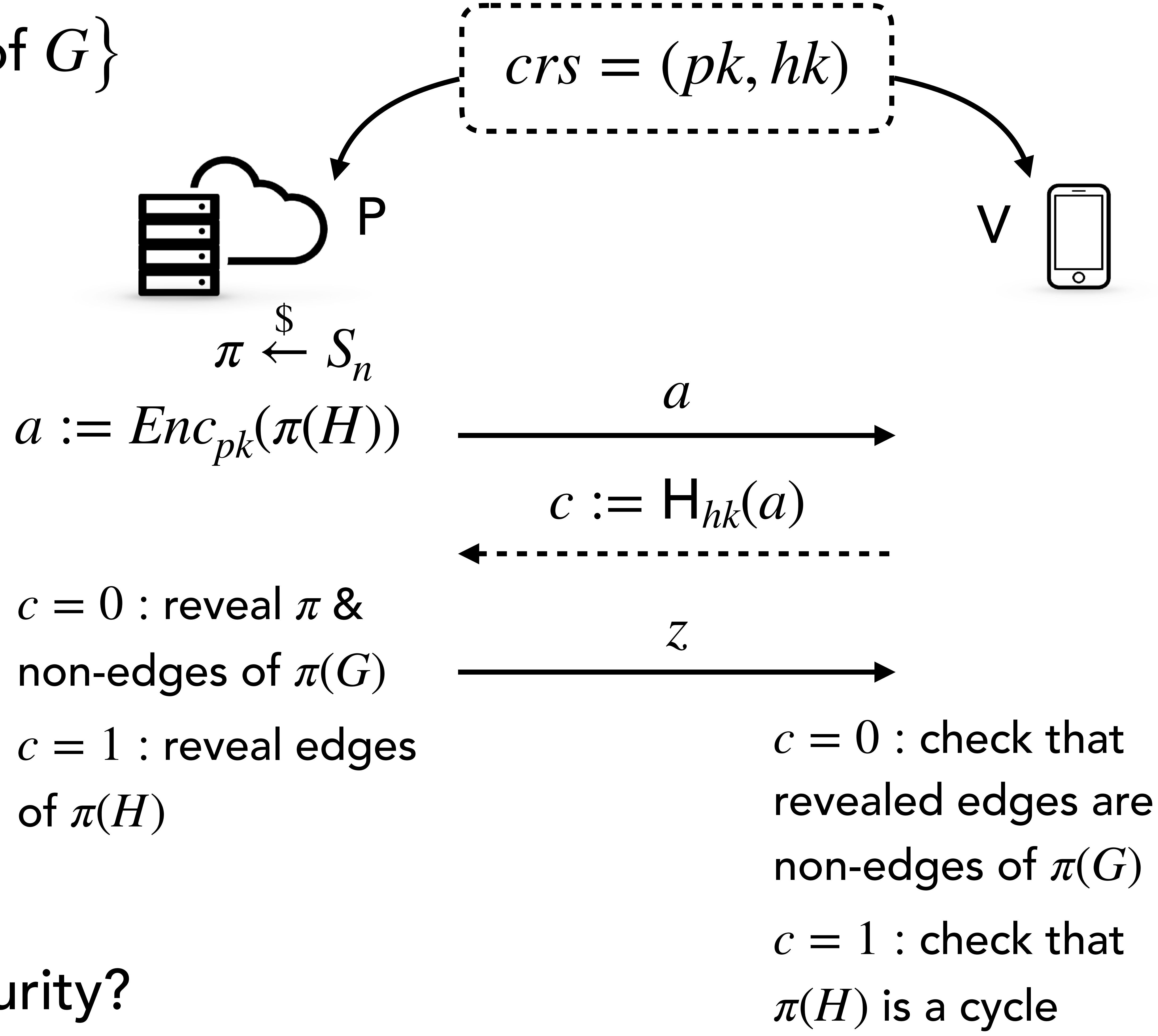
Relation: $\{(G, H) \mid H$ is a Hamiltonian cycle of $G\}$

contains all vertices of G

- Perfect completeness
- Soundness error: $1/2$
- Honest-verifier zero-knowledge

NIZK from Fiat-Shamir [FS86]?

- Derive $c := \mathsf{Hash}_{hk}(a)$
- Which hash function would preserve security?



$crs = (pk, hk)$

P

V

$\pi \overset{\$}{\leftarrow} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

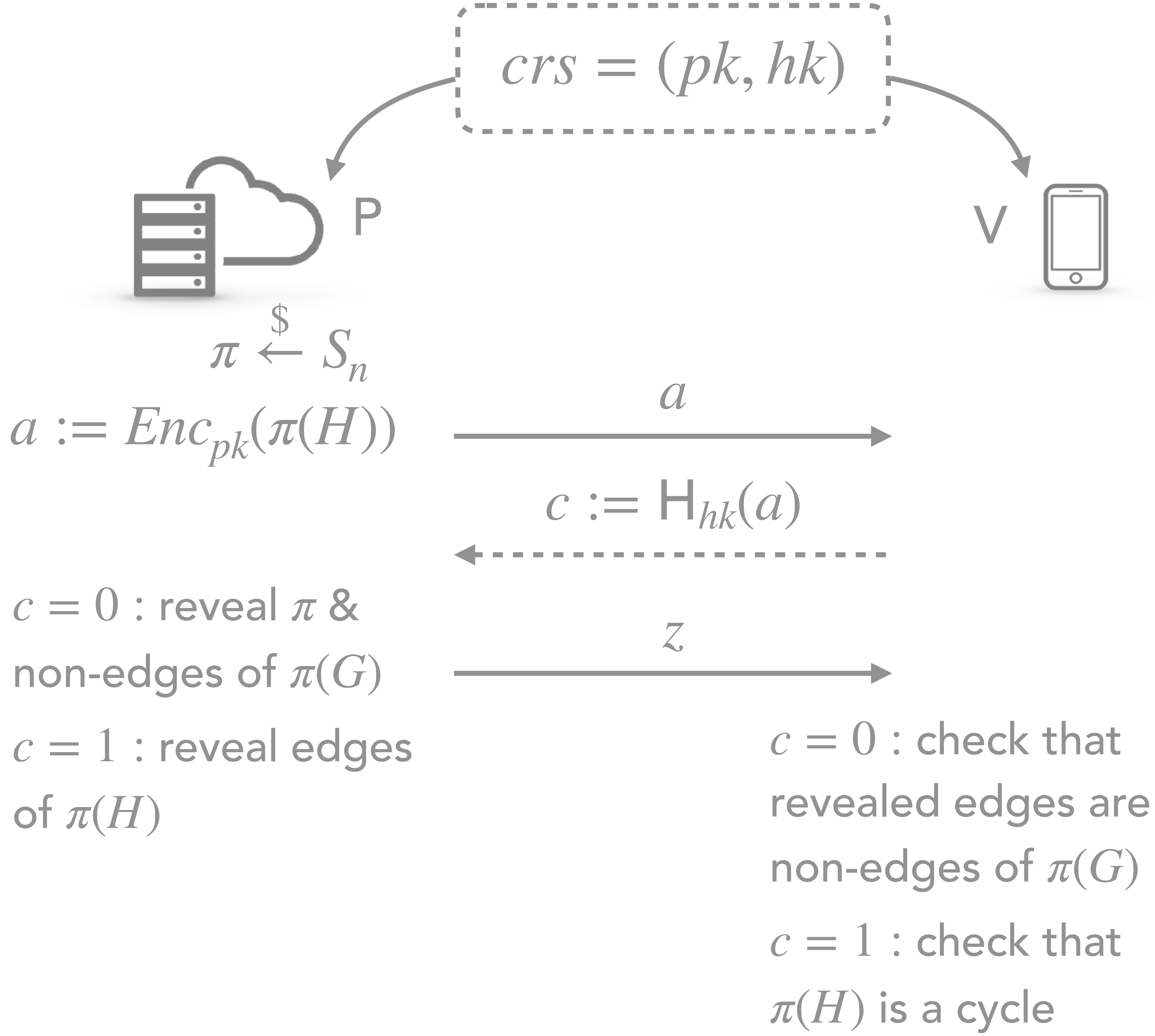
$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

NIZK from Correlation Intractability



$crs = (pk, hk)$

P

V

$\pi \overset{\$}{\leftarrow} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c :=$ $\mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

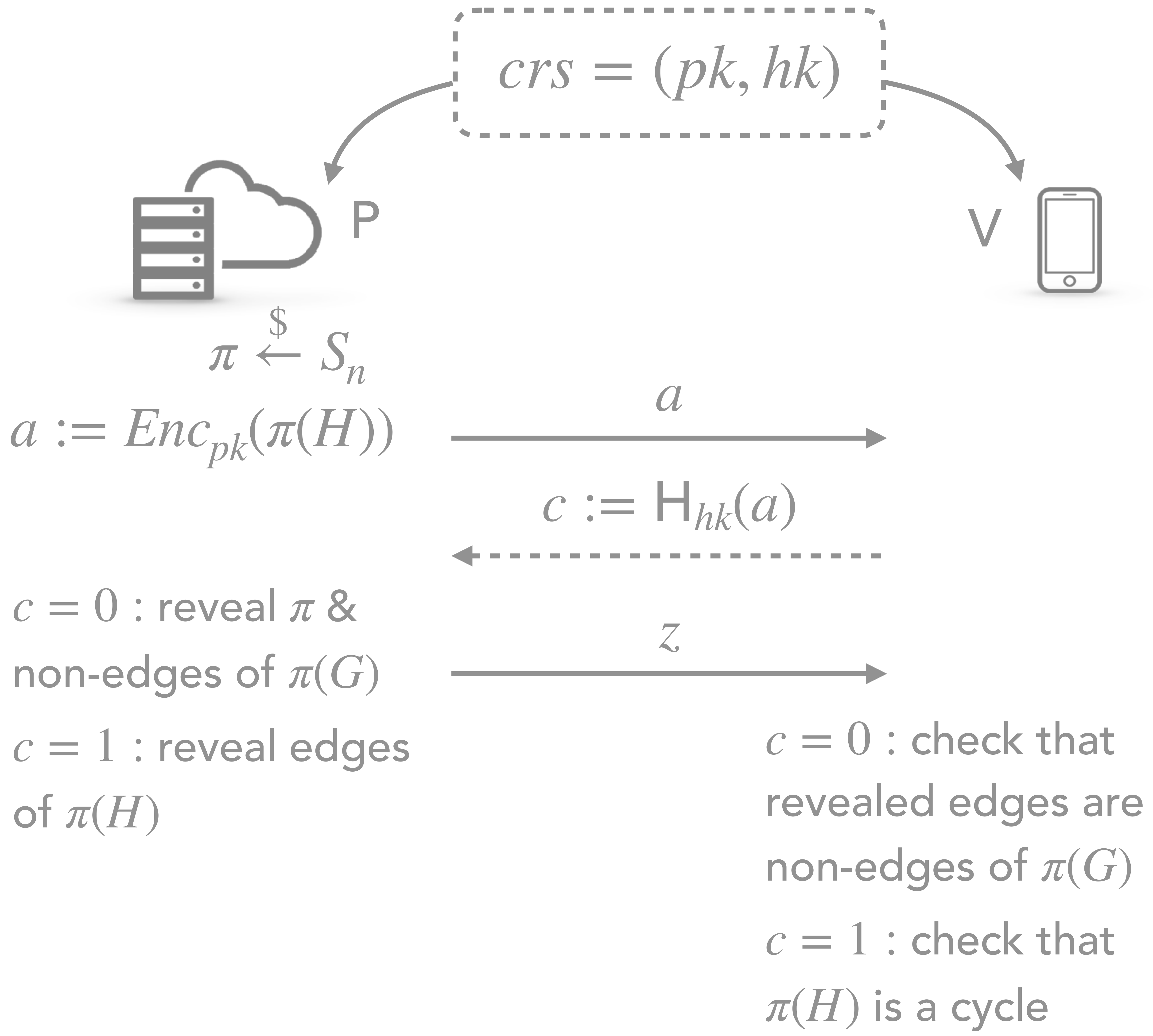
$c = 1$: check that $\pi(H)$ is a cycle

NIZK from Correlation Intractability

Correlation Intractability: [CGH04]

- H_{hk} is **CI** against a relation R if

$$\Pr_{hk}\left[(x, \mathsf{H}_{hk}(x)) \in R \mid x \leftarrow \mathscr{A}(hk)\right] \le \mathsf{negl}(\lambda)$$



$crs = (pk, hk)$

P

V

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

NIZK from Correlation Intractability

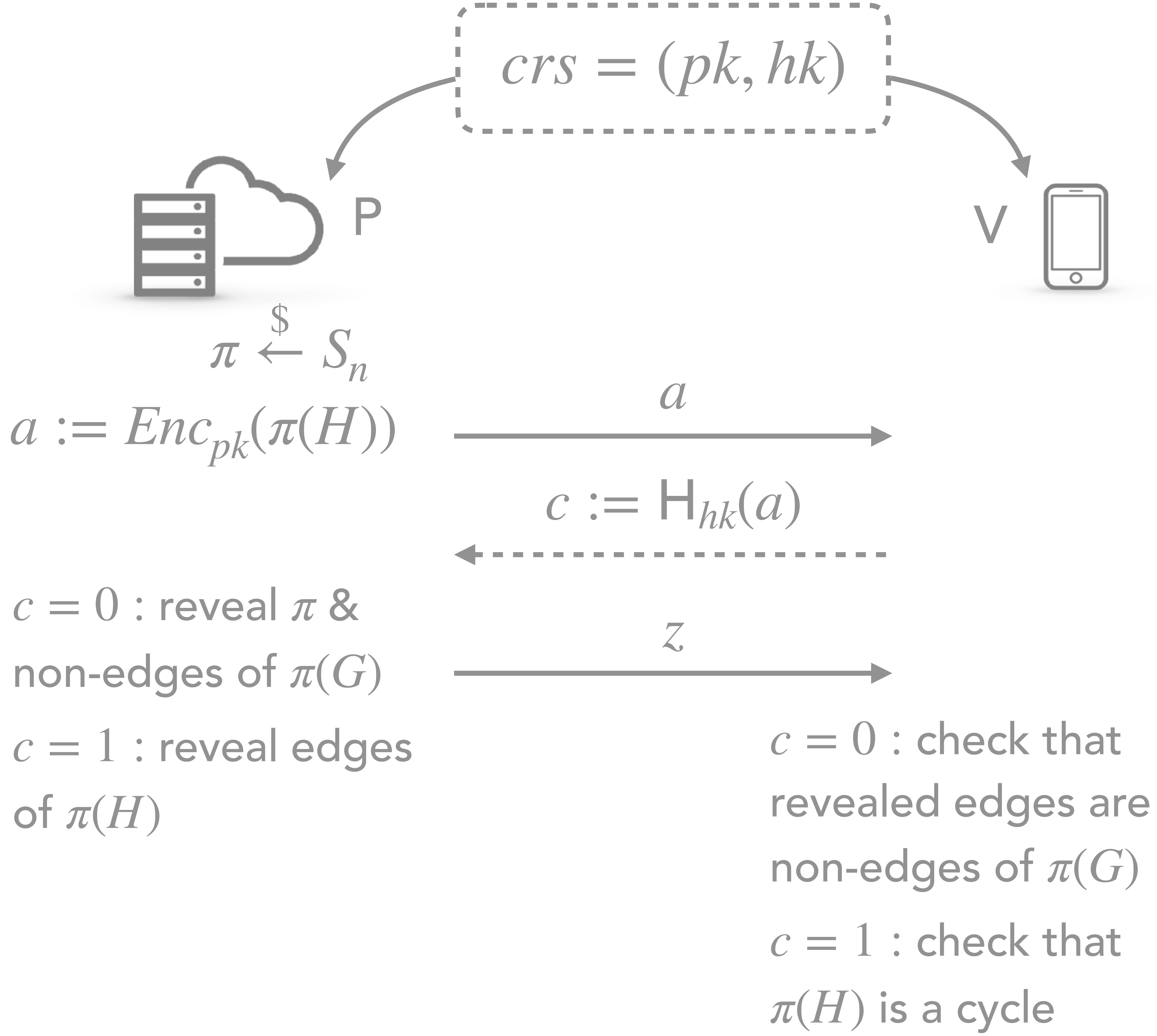
Correlation Intractability: [CGH04]

- H_{hk} is CI against a relation R if

  $$\Pr_{hk}\left[(x, \mathsf{H}_{hk}(x)) \in R \mid x \leftarrow \mathscr{A}(hk)\right] \leq \mathsf{negl}(\lambda)$$

- Fiat-Shamir is secure if H is CI against

  $$R_{bad}(x) := \left\{(a, c) \mid \exists z \text{ s.t. V accepts } (x, a, c, z)\right\}$$



$crs = (pk, hk)$

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

NIZK from Correlation Intractability

Correlation Intractability: [CGH04]

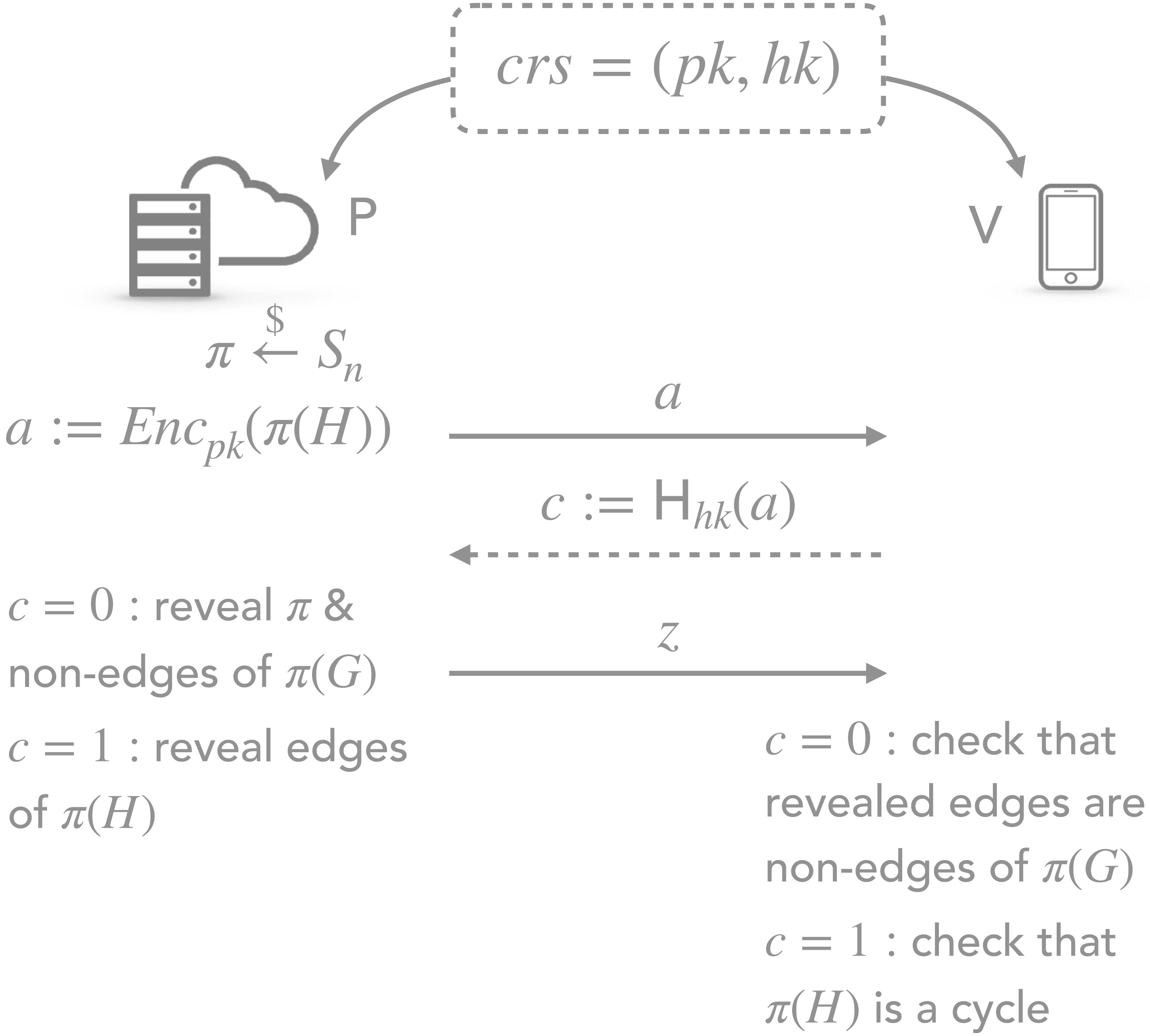
- H_{hk} is **CI** against a relation R if

$$\Pr_{hk}\left[(x, \mathsf{H}_{hk}(x)) \in R \mid x \leftarrow \mathscr{A}(hk)\right] \le \mathsf{negl}(\lambda)$$

- Fiat-Shamir is secure if H is **CI** against

$$R_{bad}(x) := \left\{(a, c) \mid \exists z \text{ s.t. V accepts } (x, a, c, z)\right\}$$

- For Blum's protocol, bad c is unique & efficiently-computable via **BadChal**$_{sk}$:
  - Decrypt $a \implies$ get $\pi(H)$
  - Output $c = 0$ if $\pi(H)$ is a cycle, else output $c = 1$.



$crs = (pk, hk)$

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

NIZK from Correlation Intractability

Correlation Intractability: [CGH04]

- H_{hk} is **CI** against a relation R if

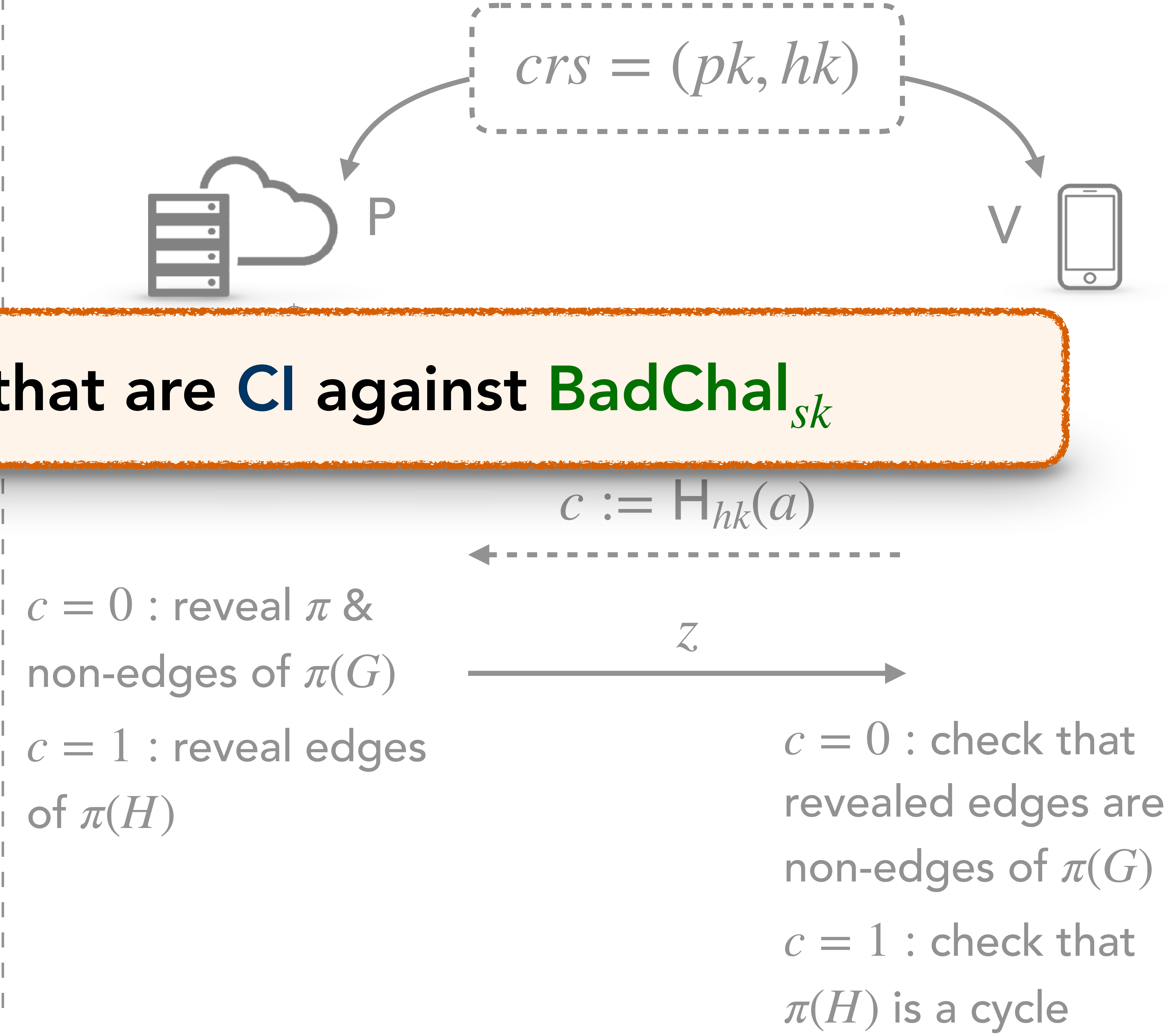
  $$\Pr_{hk}\left[(x, \mathsf{H}_{hk}(x)) \in R \mid x \leftarrow \mathscr{A}(hk)\right] \le \mathsf{negl}(\lambda)$$

- F

  $$R_{bad}(x) := \left\{(a, c) \mid \exists z \text{ s.t. V accepts } (x, a, c, z)\right\}$$

- For Blum's protocol, bad c is unique & efficiently-computable via **BadChal**$_{sk}$:
  - Decrypt $a \implies$ get $\pi(H)$
  - Output $c = 0$ if $\pi(H)$ is a cycle, else output $c = 1$.

Goal: build hash functions that are **CI** against **BadChal**$_{sk}$



$crs = (pk, hk)$

P

V

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

NIZK from Correlation Intractability

Correlation Intractability: [CGH04]

- H_{hk} is **CI** against a relation R if

$$\Pr_{hk}\left[(x, \mathsf{H}_{hk}(x)) \in R \mid x \leftarrow \mathscr{A}(hk)\right] \le \mathsf{negl}(\lambda)$$

- F

$R_{bad}(x) := \{(a, c) \mid$

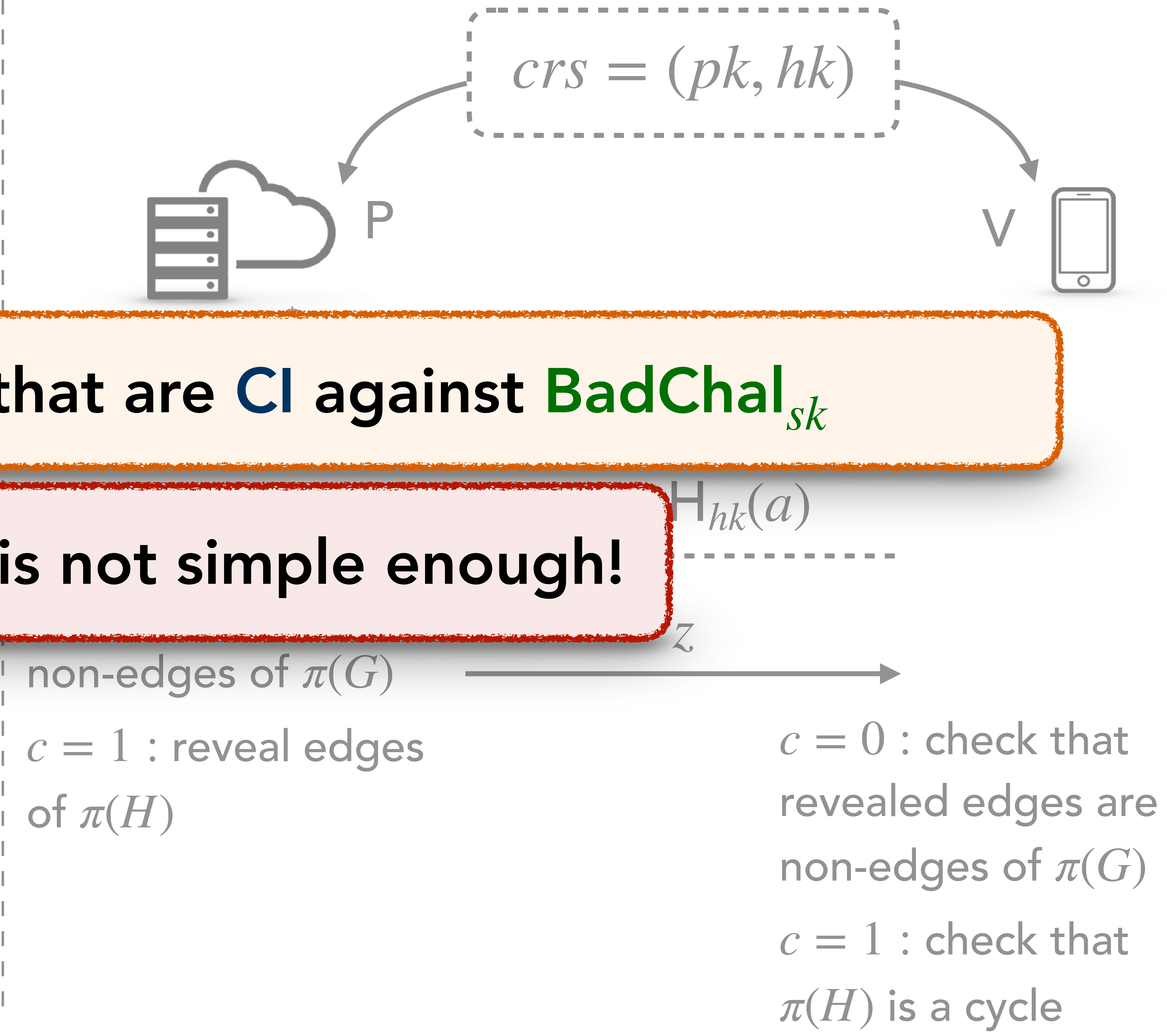
- For Blum's protoc

efficiently-computable via **BadChal**$_{sk}$:

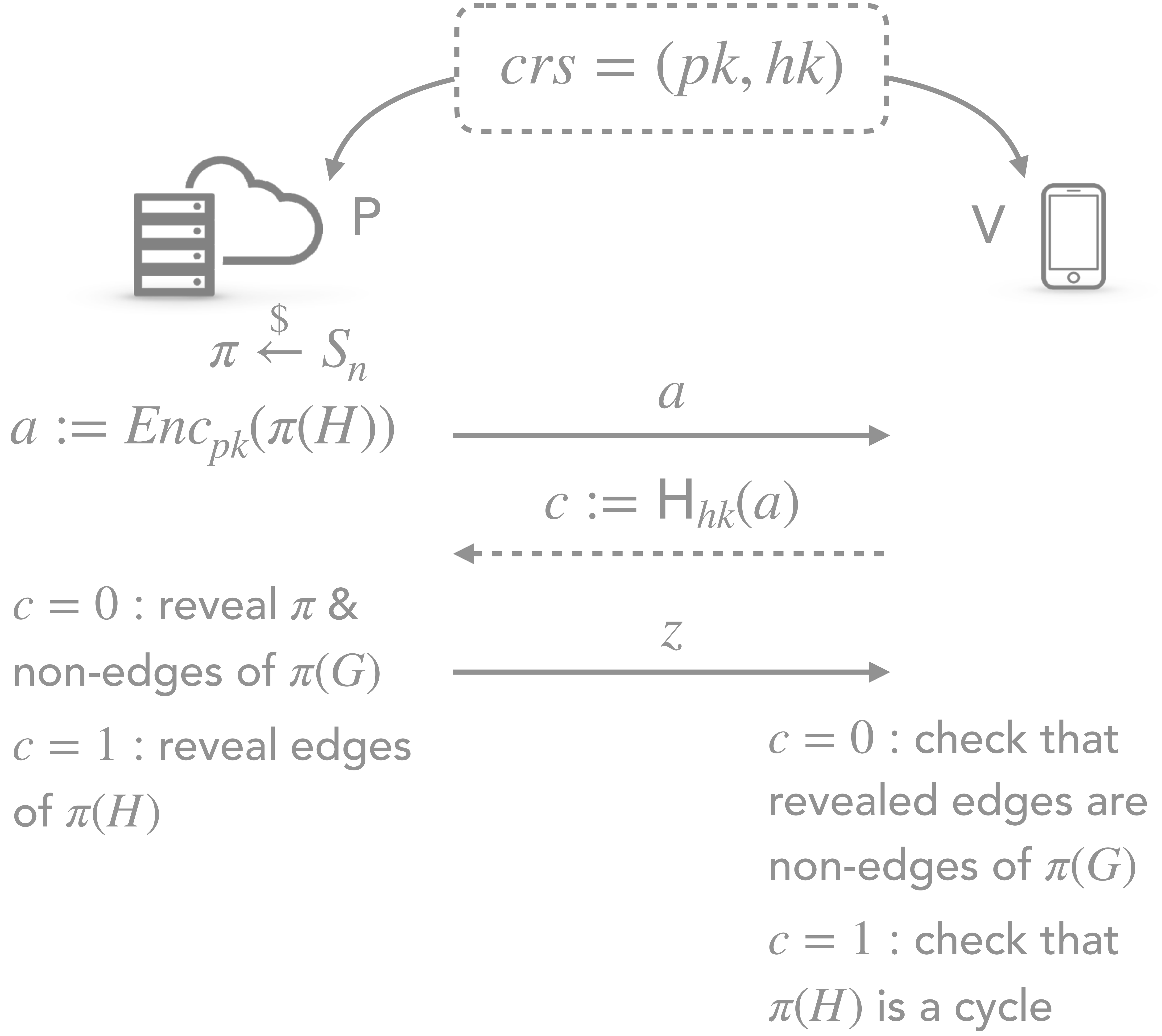
- Decrypt $a \implies$ get $\pi(H)$
- Output $c = 0$ if $\pi(H)$ is a cycle, else output $c = 1$.

Goal: build hash functions that are **CI** against **BadChal**$_{sk}$

Problem: **BadChal**$_{sk}$ **is not simple enough!**



Simplifying Bad Challenge Function



$crs = (pk, hk)$

P

V

$\pi \overset{\$}{\leftarrow} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

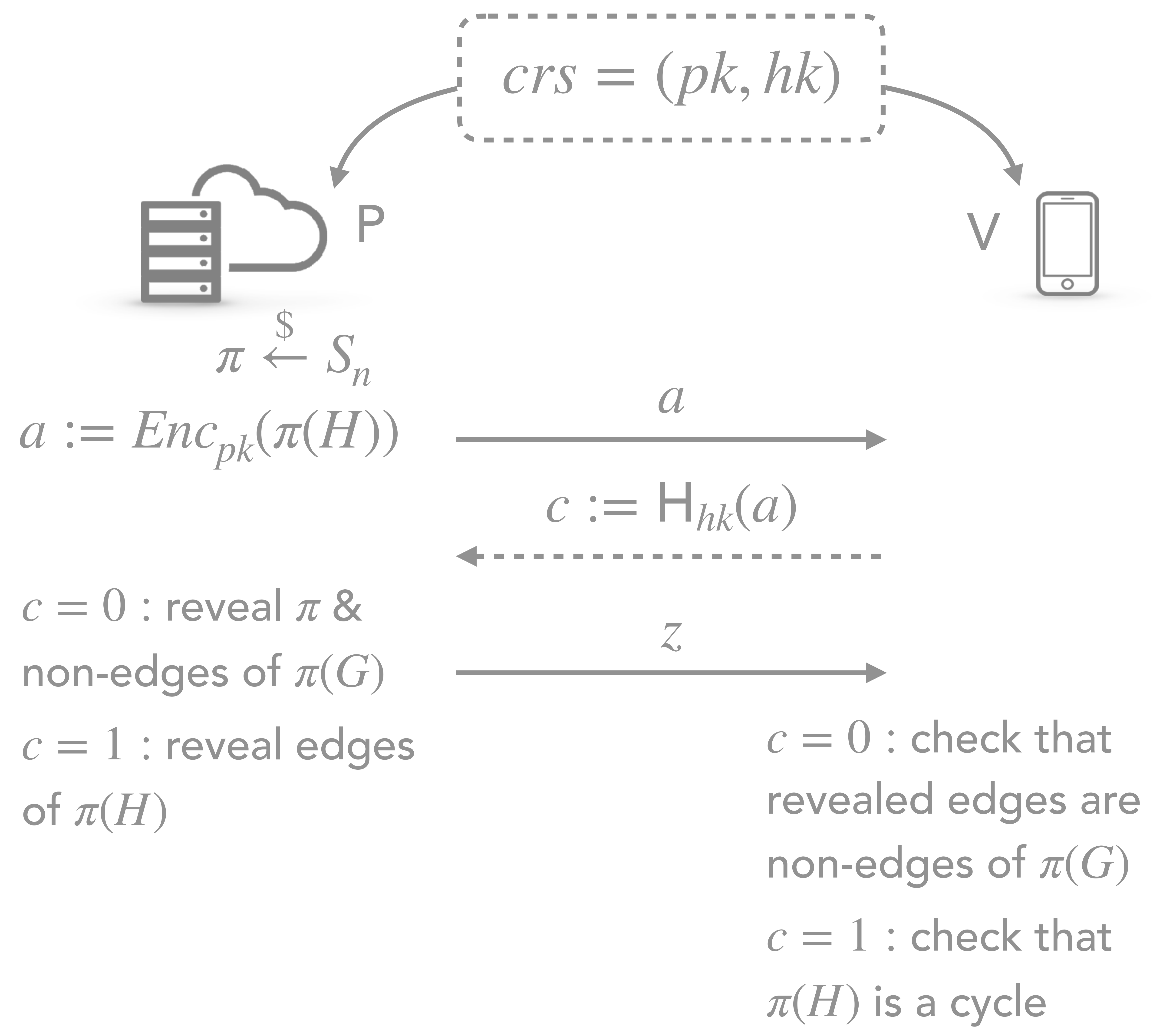
$c = 1$: check that $\pi(H)$ is a cycle

Simplifying Bad Challenge Function

[BKM20] **CI** against functions f approximable by constant-degree polynomials!

$\underset{g\leftarrow\mathscr{G}}{Pr}\,[f(x) = g(x)] \geq 0.99$ for some distribution

$\mathscr{G}$ over constant-degree polynomials



$crs = (pk, hk)$

P

V

$\pi \overset{\$}{\leftarrow} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

Simplifying Bad Challenge Function

[BKM20] **CI** against functions f approximable by constant-degree polynomials!

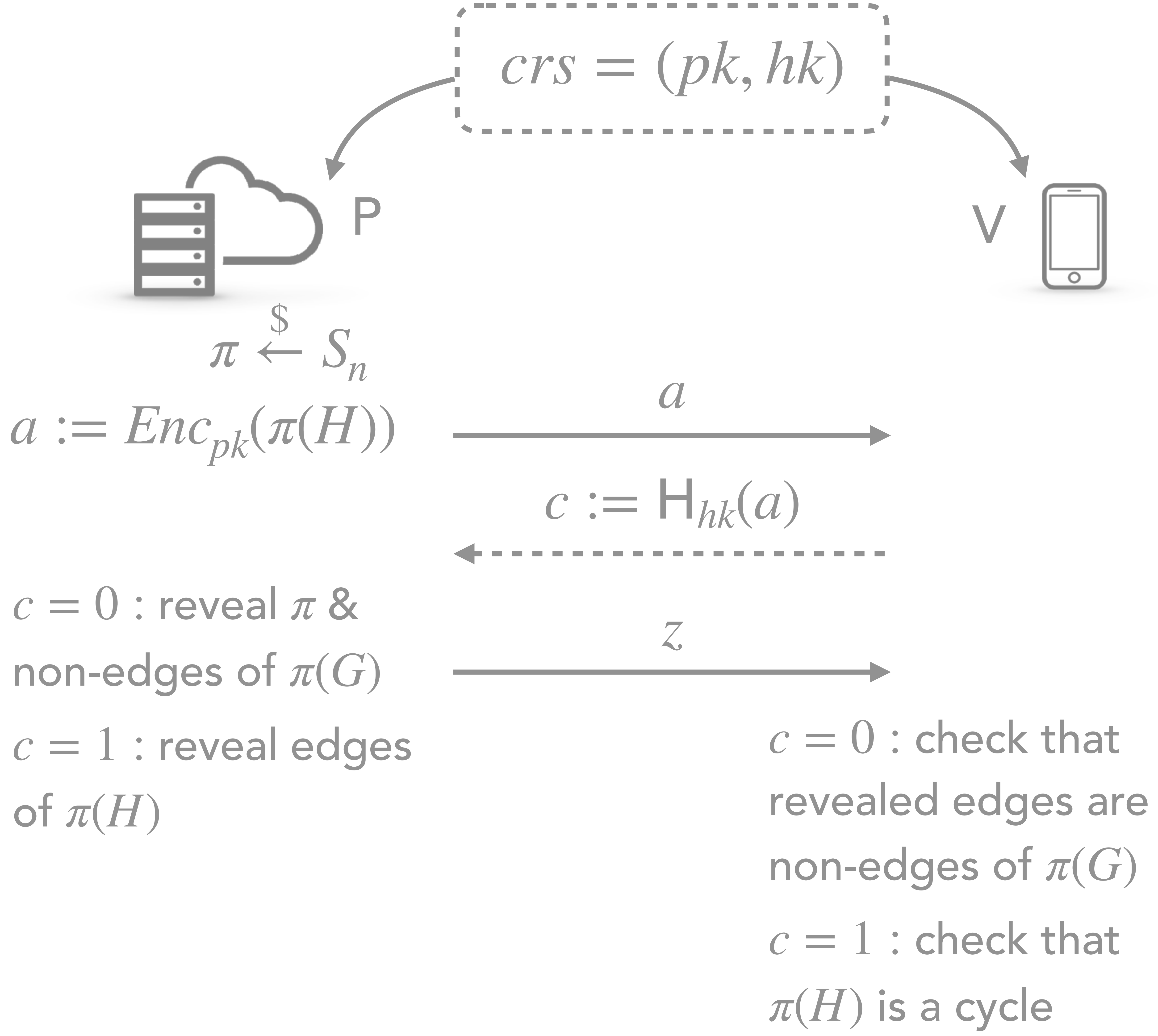
$\underset{g \leftarrow \mathscr{G}}{Pr}\,[f(x) = g(x)] \geq 0.99$ for some distribution

$\mathscr{G}$ over constant-degree polynomials

$f =$ **BadChal**$_{sk}(a)$**:**

- Decrypt $a \implies$ get $\pi(H)$
- Output $c = 0$ if $\pi(H)$ is a cycle, else output $c = 1$.

Can we modify BadChal$_{sk}$ to fall into this function class?



$crs = (pk, hk)$

P

V

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

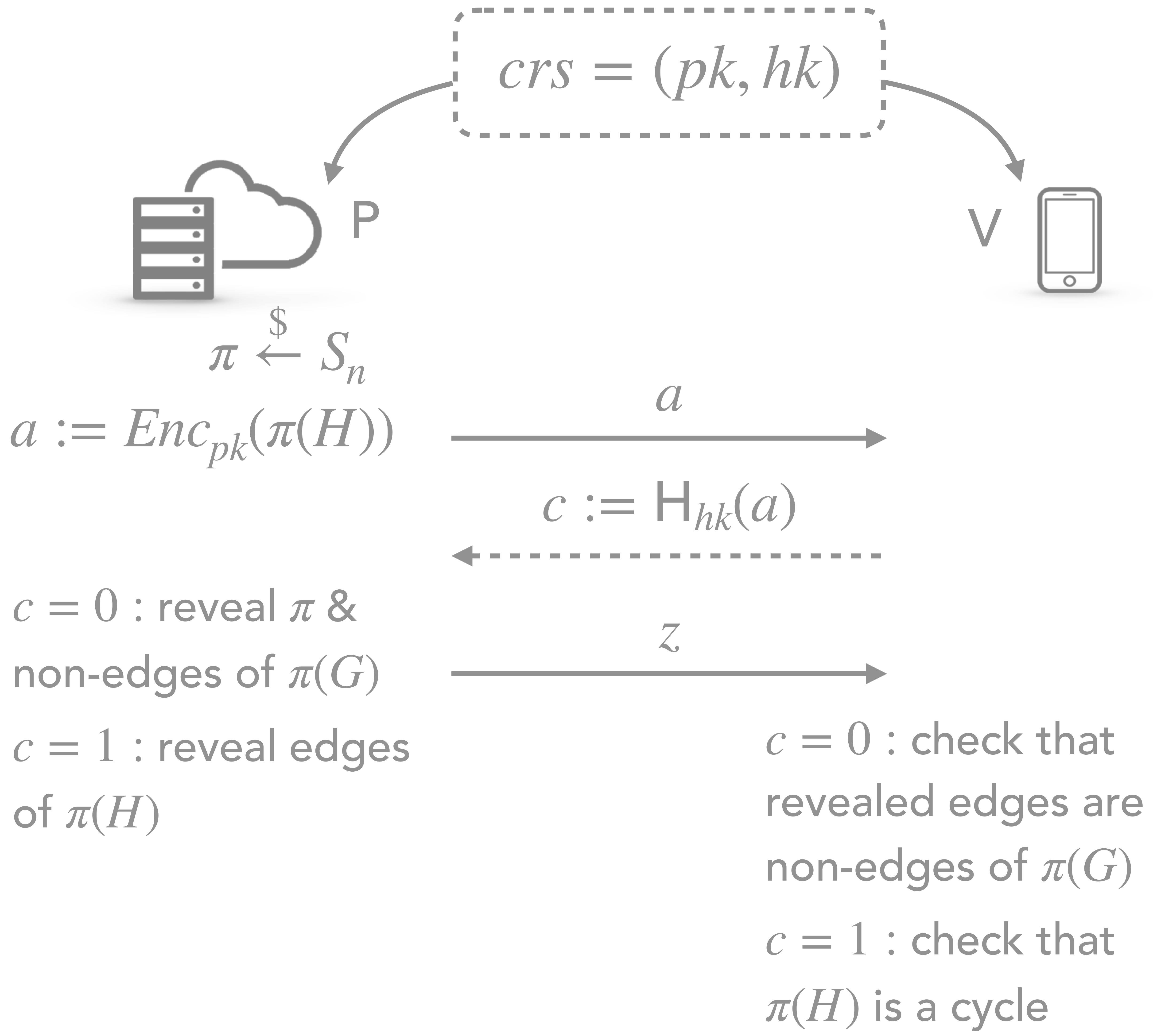
$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

Simplifying Bad Challenge Function

$f = \textsf{BadChal}_{sk}(a)$:

- Decrypt $a \implies$ get $\pi(H)$
- Output $c = 0$ if $\pi(H)$ is a cycle, else output $c = 1$.



$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

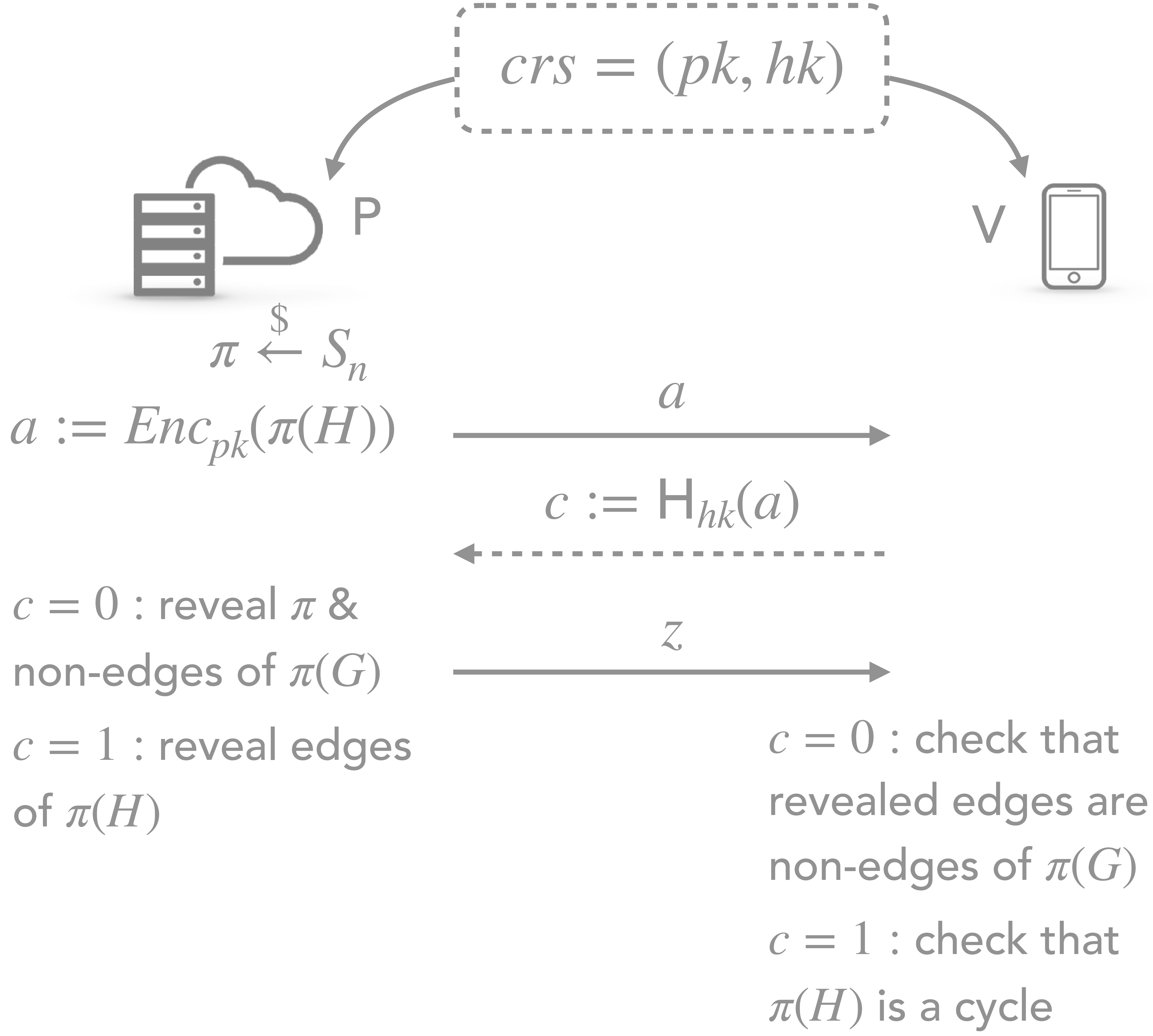
Simplifying Bad Challenge Function

$f = \mathsf{BadChal}_{sk}(a)$:

- Decrypt $a \implies$ get $\pi(H)$
- Output $c = 0$ if $\pi(H)$ is a cycle, else output $c = 1$.

1. Have Dec_{sk} be approximately linear

$\implies$ achieved via **LPN**-based PKE



$crs = (pk, hk)$

P

V

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

Simplifying Bad Challenge Function

$f = \textsf{BadChal}_{sk}(a)$:

- Decrypt $a \implies$ get $\pi(H)$
- Output $c = 0$ if $\pi(H)$ is a cycle, else output $c = 1$.

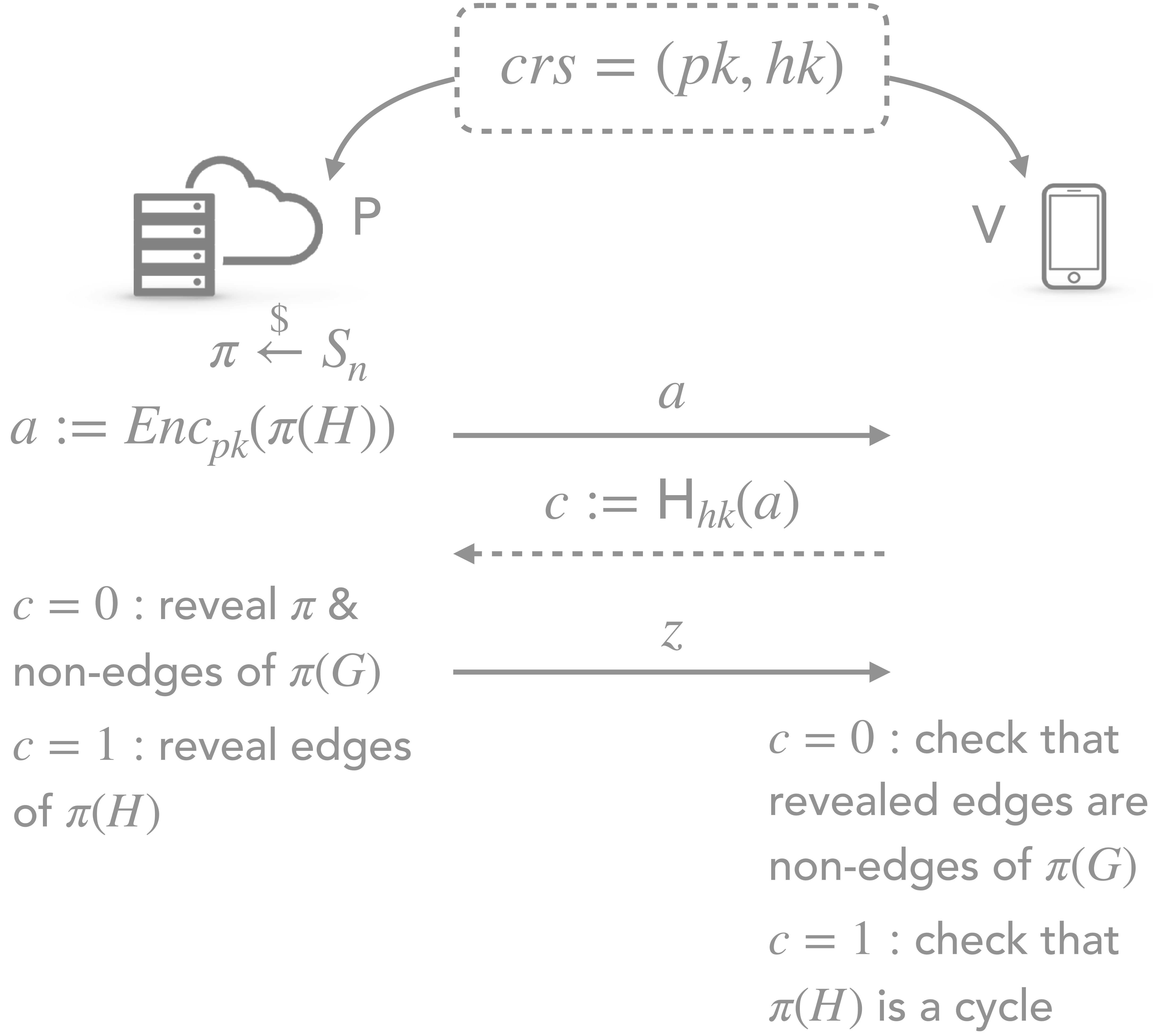
1. Have Dec_{sk} be approximately linear

   $\implies$ achieved via **LPN**-based PKE

2. Turn cycle check into 3CNF formula Φ :

   $\pi(H)$ is a cycle $\iff \exists w$ s.t. $\Phi(\pi(H), w) = 1$

   (Φ is approximable by $O(1)$-degree poly)



$crs = (pk, hk)$

P

V

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

Simplifying Bad Challenge Function

$f = \textbf{BadChal}_{sk}(a)$:

- Decrypt $a \implies$ get $\pi(H)$
- Output $c = 0$ if $\pi(H)$ is a cycle,
  else output $c = 1$.

1. Have Dec_{sk} be approximately linear

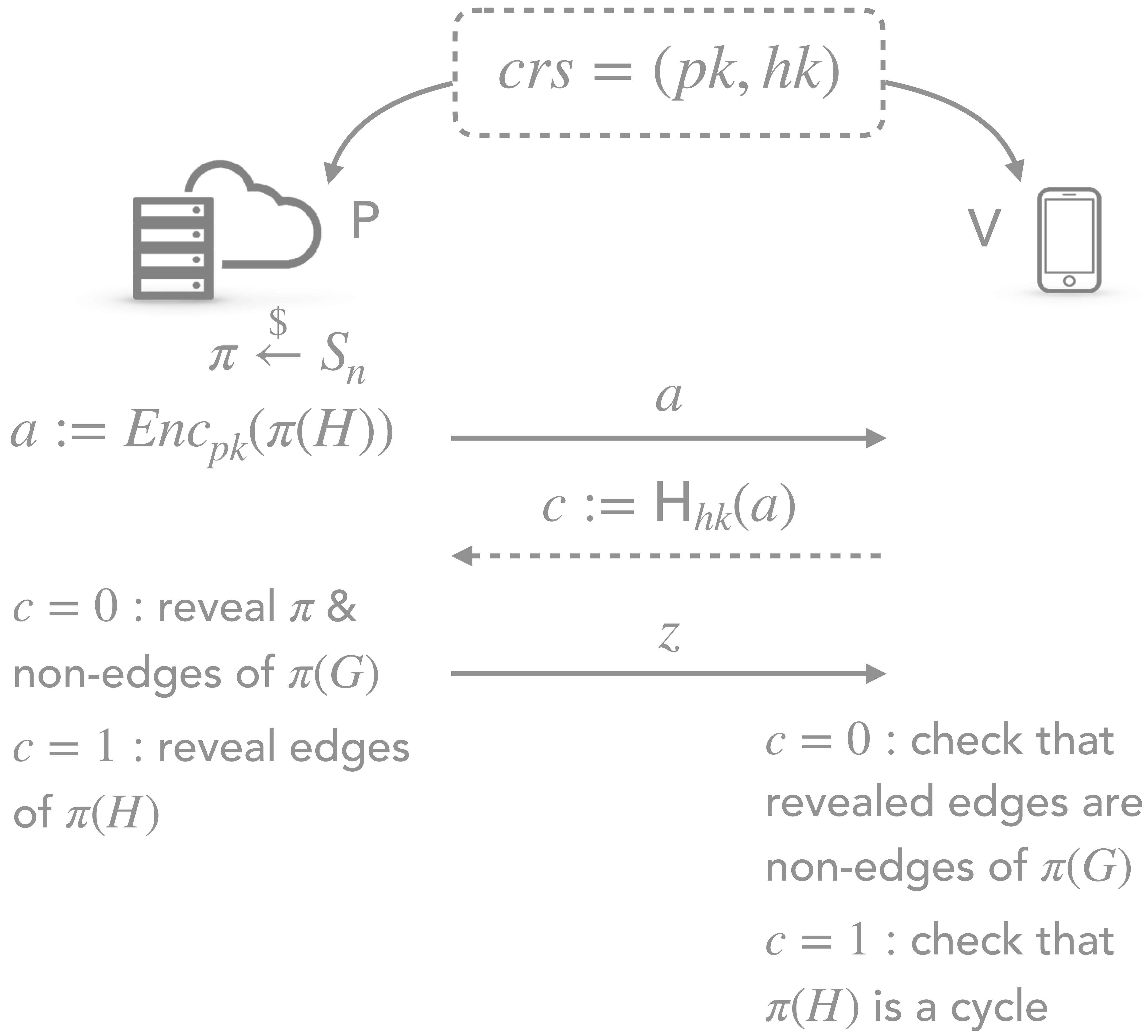
   $\implies$ achieved via **LPN**-based PKE

2. Turn cycle check into 3CNF formula Φ :

   $\pi(H)$ is a cycle $\iff \exists w$ s.t. $\Phi(\pi(H), w) = 1$

   (Φ is approximable by $O(1)$-degree poly)

3. Encrypt & send w in the first round



$crs = (pk, hk)$

P

V

$\pi \overset{\$}{\leftarrow} S_n$

$a := Enc_{pk}(\pi(H))$

a

$c := \mathsf{H}_{hk}(a)$

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

z

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

Simplifying Bad Challenge Function

$f = \textsf{BadChal}_{sk}(a)$:

- Decrypt $a \implies$ get $\pi(H), w$
- Output $c = 0$ if $\Phi(\pi(H), w) = 1$, else output $c = 1$.

1. Have Dec_{sk} be approximately linear

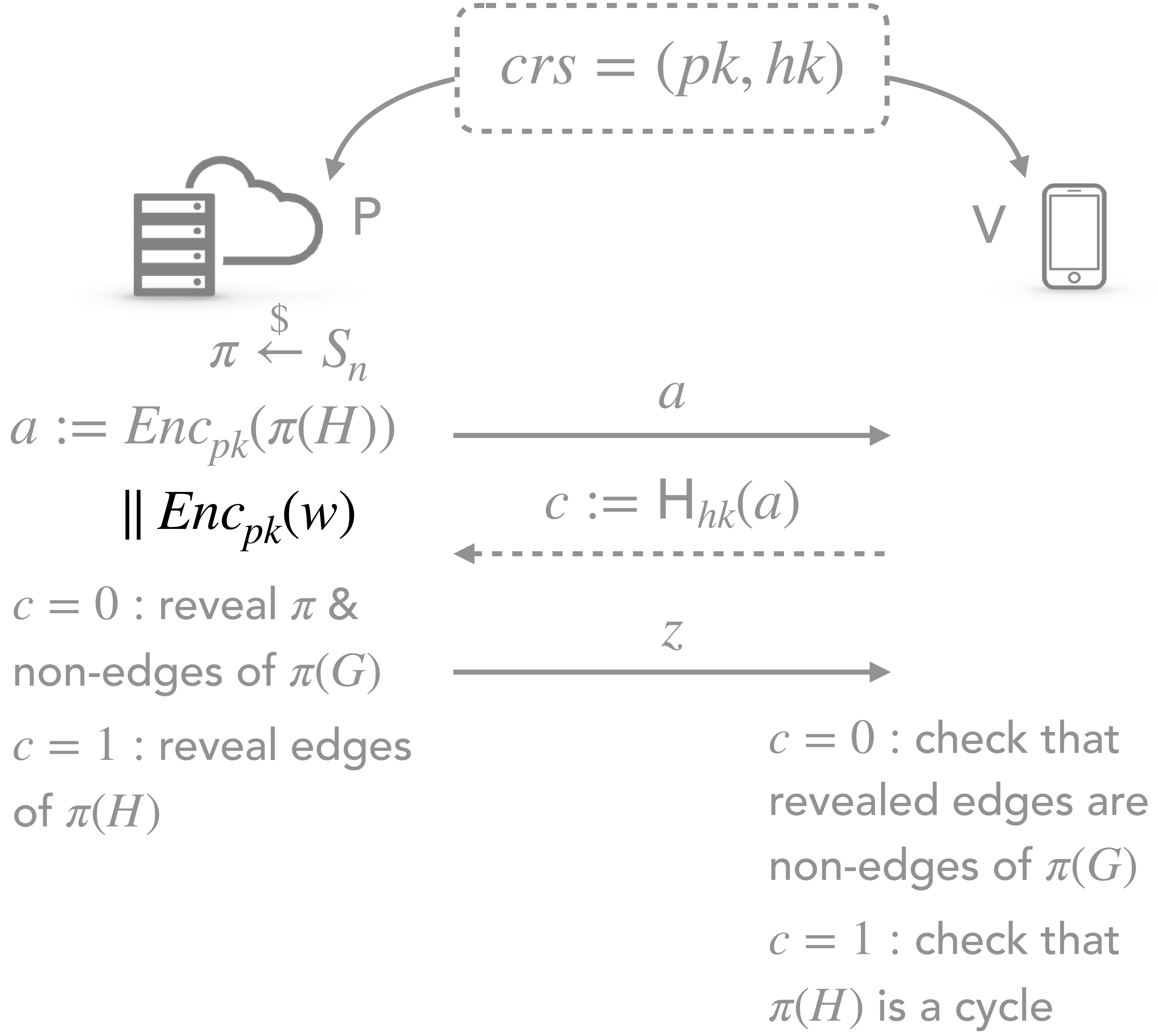
   $\implies$ achieved via **LPN**-based PKE

2. Turn cycle check into 3CNF formula Φ :

   $\pi(H)$ is a cycle $\iff \exists w$ s.t. $\Phi(\pi(H), w) = 1$

   (Φ is approximable by $O(1)$-degree poly)

3. Encrypt & send w in the first round



$crs = (pk, hk)$

$\pi \xleftarrow{\$} S_n$

$a := Enc_{pk}(\pi(H))$

$\| \, Enc_{pk}(w)$

a

$c := \mathsf{H}_{hk}(a)$

z

$c = 0$: reveal π & non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

Simplifying Bad Challenge Function

$f = \mathsf{BadChal}_{sk}(a)$:

- Decrypt $a \implies$ get $\pi(H), w$
- Output $c = 0$ if $\Phi(\pi(H), w) = 1$, else output $c = 1$.

<u>New Goal:</u> build hash functions that are <u>approximate CI</u> against constant-degree polynomials

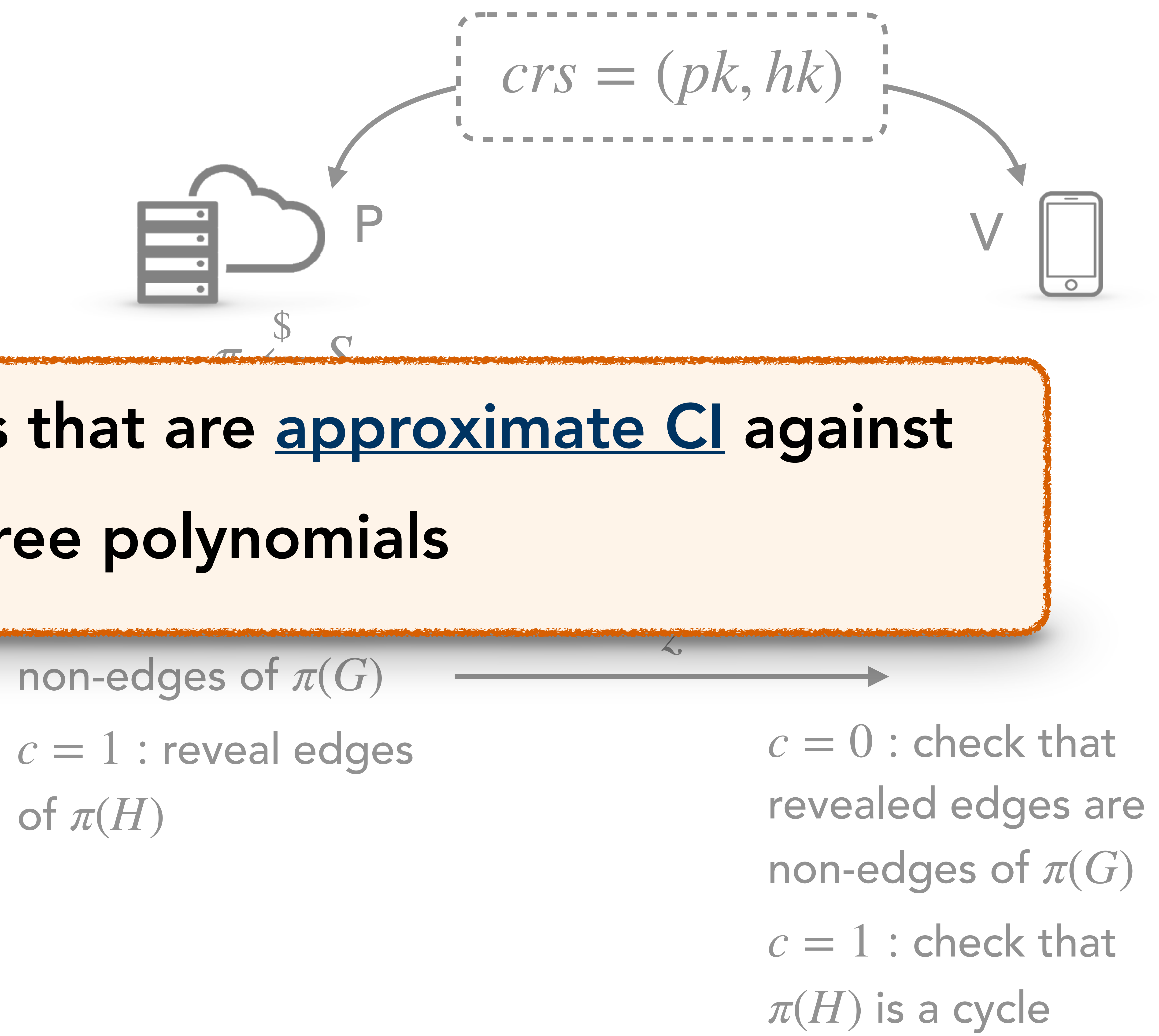
1.

2. Turn cycle check into 3CNF formula Φ :

$$\pi(H) \text{ is a cycle} \iff \exists w \text{ s.t. } \Phi(\pi(H), w) = 1$$

(Φ is approximable by $O(1)$-degree poly)

3. Encrypt & send w in the first round



non-edges of $\pi(G)$

$c = 1$: reveal edges of $\pi(H)$

$c = 0$: check that revealed edges are non-edges of $\pi(G)$

$c = 1$: check that $\pi(H)$ is a cycle

Talk Outline

1. Recap: NIZK from Correlation Intractability
2. CI Hashing from (Approximate) MQ
3. Putting Things Together

Multivariate Quadratic

Multivariate Quadratic

Solving a random system of quadratic polynomial equations (over finite $\mathbb{F}$) is hard!

$$\begin{cases} \sum_{i,j=1}^{n} a_{i,j}^{(1)} \cdot x_i \cdot x_j + \sum_{i=1}^{n} b_i^{(1)} \cdot x_i + c^{(1)} = 0 \\ \vdots \\ \sum_{i,j=1}^{n} a_{i,j}^{(m)} \cdot x_i \cdot x_j + \sum_{i=1}^{n} b_i^{(m)} \cdot x_i + c^{(m)} = 0 \end{cases}, \quad \text{where} \quad \begin{cases} n = \#\ \text{variables} \\ m = \#\ \text{equations} \\ \text{eqns. over a finite field } \mathbb{F} \end{cases}$$

Multivariate Quadratic

Solving a random system of quadratic polynomial equations (over finite $\mathbb{F}$) is hard!

$$\begin{cases} \sum_{i,j=1}^{n} a_{i,j}^{(1)} \cdot x_i \cdot x_j + \sum_{i=1}^{n} b_i^{(1)} \cdot x_i + c^{(1)} = 0 \\ \vdots \\ \sum_{i,j=1}^{n} a_{i,j}^{(m)} \cdot x_i \cdot x_j + \sum_{i=1}^{n} b_i^{(m)} \cdot x_i + c^{(m)} = 0 \end{cases}, \quad \text{where} \quad \begin{cases} n = \#\text{ variables} \\ m = \#\text{ equations} \\ \text{eqns. over a finite field } \mathbb{F} \end{cases}$$

- One of the main branches of assumptions in post-quantum cryptography
- Hard for $\sqrt{n} \ll m \ll n^2$. Usual parameter regime: $m = \Theta(n)$

Multivariate Quadratic

Solving a random system of quadratic polynomial equations (over finite $\mathbb{F}$) is hard!

$$\begin{cases} \sum_{i,j=1}^{n} a_{i,j}^{(1)} \cdot x_i \cdot x_j + \sum_{i=1}^{n} b_i^{(1)} \cdot x_i + c^{(1)} = 0 \\ \vdots \\ \sum_{i,j=1}^{n} a_{i,j}^{(m)} \cdot x_i \cdot x_j + \sum_{i=1}^{n} b_i^{(m)} \cdot x_i + c^{(m)} = 0 \end{cases}, \quad \text{where} \quad \begin{cases} n = \text{\# variables} \\ m = \text{\# equations} \\ \text{eqns. over a finite field } \mathbb{F} \end{cases}$$

- One of the main branches of assumptions in post-quantum cryptography
- Hard for $\sqrt{n} \ll m \ll n^2$. Usual parameter regime: $m = \Theta(n)$
- **<u>This work:</u>** under-determined setting, with $m = n^{1-\epsilon}$ for any $\epsilon > 0$

$\Longrightarrow$ best cryptanalysis [TW12, MHT13] suggests ***<u>exponential</u>*** security*

* poly-time attackers have ***<u>exponentially</u>*** small success probability

Hash Function from Multivariate Quadratic

Hash Function from Multivariate Quadratic

Hash key

$$Q := \left(a_{i,j}^{(k)},\, b_i^{(k)},\, c^{(k)}\right)_{i,j\in[n],\, k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

Hash Function from Multivariate Quadratic

Hash key

$$Q := \left(a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)}\right)_{i,j\in[n],\ k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

This is not collision-resistant! * Choose random Δ, solve for $Q(x + \Delta) = Q(x)$

Hash Function from Multivariate Quadratic

Hash key

$$Q := \left(a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)}\right)_{i,j\in[n],\, k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

This is not collision-resistant! * Choose random Δ, solve for $Q(x+\Delta) = Q(x)$

However, it is correlation-intractable against quadratic polynomials!

Hash Function from Multivariate Quadratic

Hash key

$$Q := \left(a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)}\right)_{i,j\in[n],\, k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

This is not collision-resistant! * Choose random Δ, solve for $Q(x+\Delta) = Q(x)$

However, it is **correlation-intractable** against quadratic polynomials!

- Assume $\mathsf{H}_Q(x) = f(x)$ for a quadratic function f.
- Switch to hybrid where $Q \mapsto Q + f \implies$ hash key is still random
- But we have: $\mathsf{H}_{Q+f}(x) = f(x) \iff Q(x) = 0$, which breaks **MQ**.

Approximate CI from Approximate MQ

Hash key

$$Q := \left(a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)}\right)_{i,j\in[n],\, k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

Approximate CI from Approximate MQ

Hash key

$$Q := \left(a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)}\right)_{i,j\in[n],\, k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

Does H_Q satisfy **approximate CI** against quadratic polynomials?

Approximate CI from Approximate MQ

Hash key

$$Q := \left(a_{i,j}^{(k)},\, b_i^{(k)},\, c^{(k)}\right)_{i,j\in[n],\, k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

Does H_Q satisfy **approximate CI** against quadratic polynomials?

- Assume $\mathsf{H}_Q(x)$ has $99\,\%$ agreement with $f(x)$, for some quadratic function f
- Use hybrid switch $Q \mapsto Q + f \implies$ we have $Q(x) = 0$ for $99\,\%$ of equations

Approximate CI from Approximate MQ

Hash key

$$Q := \left(a_{i,j}^{(k)},\, b_i^{(k)},\, c^{(k)}\right)_{i,j\in[n],\; k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

Does H_Q satisfy **approximate CI** against quadratic polynomials?

- Assume $\mathsf{H}_Q(x)$ has $99\,\%$ agreement with $f(x)$, for some quadratic function f
- Use hybrid switch $Q \mapsto Q + f \implies$ we have $Q(x) = 0$ for $99\,\%$ of equations
- **New Assumption:** Approximate MQ states that this is still hard

Approximate CI from Approximate MQ

Hash key

$$Q := \left(a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)}\right)_{i,j\in[n],\, k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

Does H_Q satisfy **approximate CI** against quadratic polynomials?

- Assume $\mathsf{H}_Q(x)$ has $99\,\%$ agreement with $f(x)$, for some quadratic function f
- Use hybrid switch $Q \mapsto Q + f \implies$ we have $Q(x) = 0$ for $99\,\%$ of equations
- <u>**New Assumption:**</u> **Approximate MQ** states that this is still hard
- **Approximate MQ** is implied by **MQ** with exponential security

* via guessing
error pattern

Approximate CI from Approximate MQ

Hash key

$$Q := \left(a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)}\right)_{i,j\in[n],\, k\in[m]}$$

Hash evaluation

$$\mathsf{H}_Q(x) := Q(x) = \left(\sum_{i,j} a_{i,j}^{(k)} x_i x_j + b_i^{(k)} x_i + c^{(k)}\right)_{k=1}^{m}$$

Does H_Q satisfy approximate CI against quadratic polynomials?

- Assume $\mathsf{H}_Q(x)$ has $99\,\%$ agreement with $f(x)$, for some quadratic function f
- Use
- Nev
- Approximate MQ is implied by MQ with exponential security

* via guessing error pattern

Question: How to achieve ApproxCI against degree$-d$ polynomials, for *any* constant d?

Approximate CI against Degree$-d$ Polys

Approximate CI against Degree$-d$ Polys

Two Solutions:

Approximate CI against Degree$-d$ Polys

Two Solutions:

1. Evaluate random degree$-d$ polynomials on input
    - Hardness follows from degree$-d$ analogue of **(Approximate) MQ**
    - **Downsides:** not as well-studied, blows up key size & evaluation time

Approximate CI against Degree$-d$ Polys

Two Solutions:

1. Evaluate random degree$-d$ polynomials on input
   - Hardness follows from degree$-d$ analogue of (Approximate) MQ
   - **Downsides:** not as well-studied, blows up key size & evaluation time
2. Achieve approximate CI against a ***sub-class*** of degree$-d$ polynomials
   - ***Concatenated*** degree$-d$ polys: $P\left(x_1 \| \ldots \| x_l\right) = P_1(x_1) \| \ldots \| P_l(x_l)$, $\deg(P_i) = d$
   - **Setting:** $|x_i| = s$ is fixed, $l = poly(\lambda)$ may grow
   - Hash evaluation: $\mathsf{H}_Q\left(x_1 \| \ldots \| x_l\right) := Q\left(x_1^{\otimes d/2}, \ldots, x_l^{\otimes d/2}\right)$

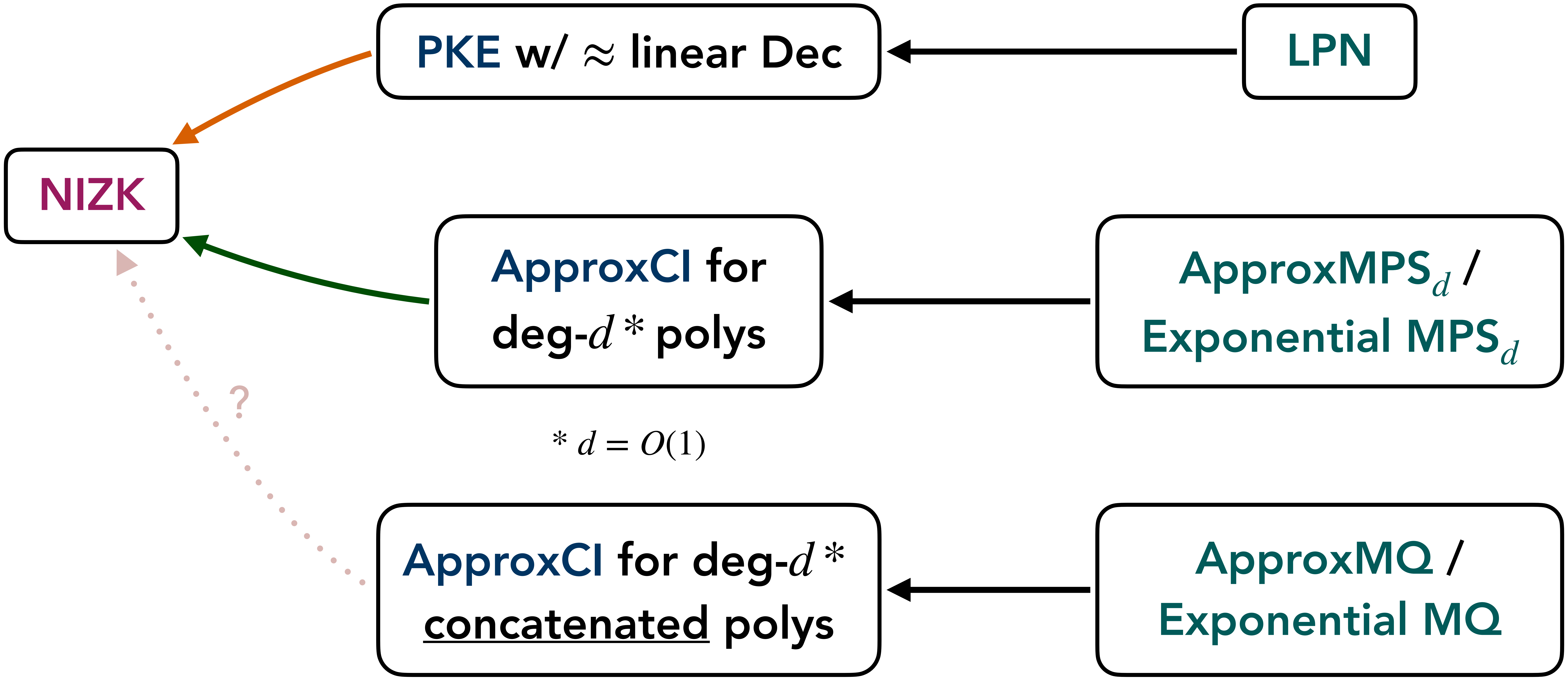
     $\Longrightarrow$ achieves compression for large enough $l = poly(\lambda)$

Talk Outline

1. Recap: NIZK from Correlation Intractability
2. CI Hashing from (Approximate) MQ
3. Putting Things Together

Road to Achieve NIZK

Road to Achieve **NIZK**



PKE w/ ≈ linear Dec

LPN

NIZK

ApproxCI for deg-d* polys

ApproxMPS$_d$ / Exponential MPS$_d$

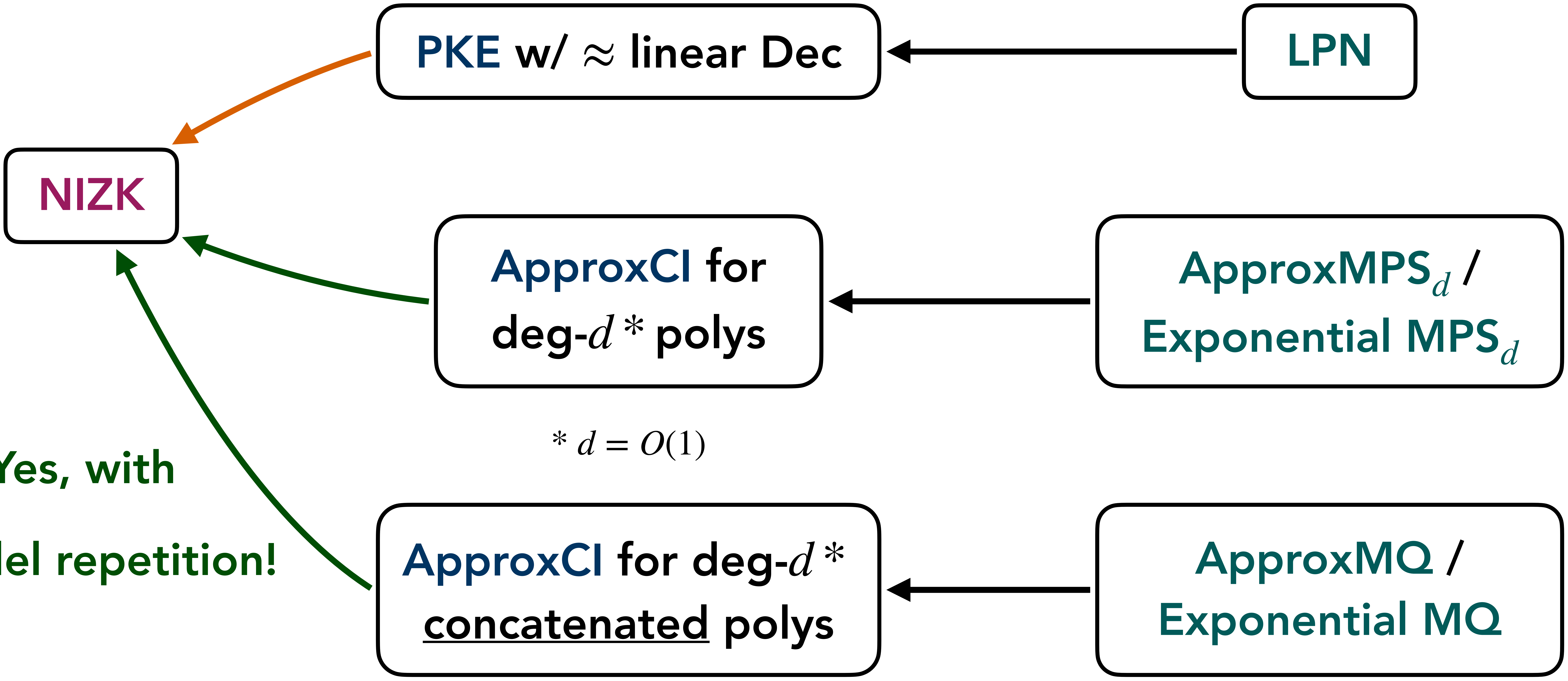
?

* $d = O(1)$

ApproxCI for deg-d* concatenated polys

ApproxMQ / Exponential MQ

Road to Achieve NIZK



NIZK

PKE w/ ≈ linear Dec

LPN

ApproxCI for deg-$d*$ polys

ApproxMPS$_d$ / Exponential MPS$_d$

$* \; d = O(1)$

Yes, with parallel repetition!

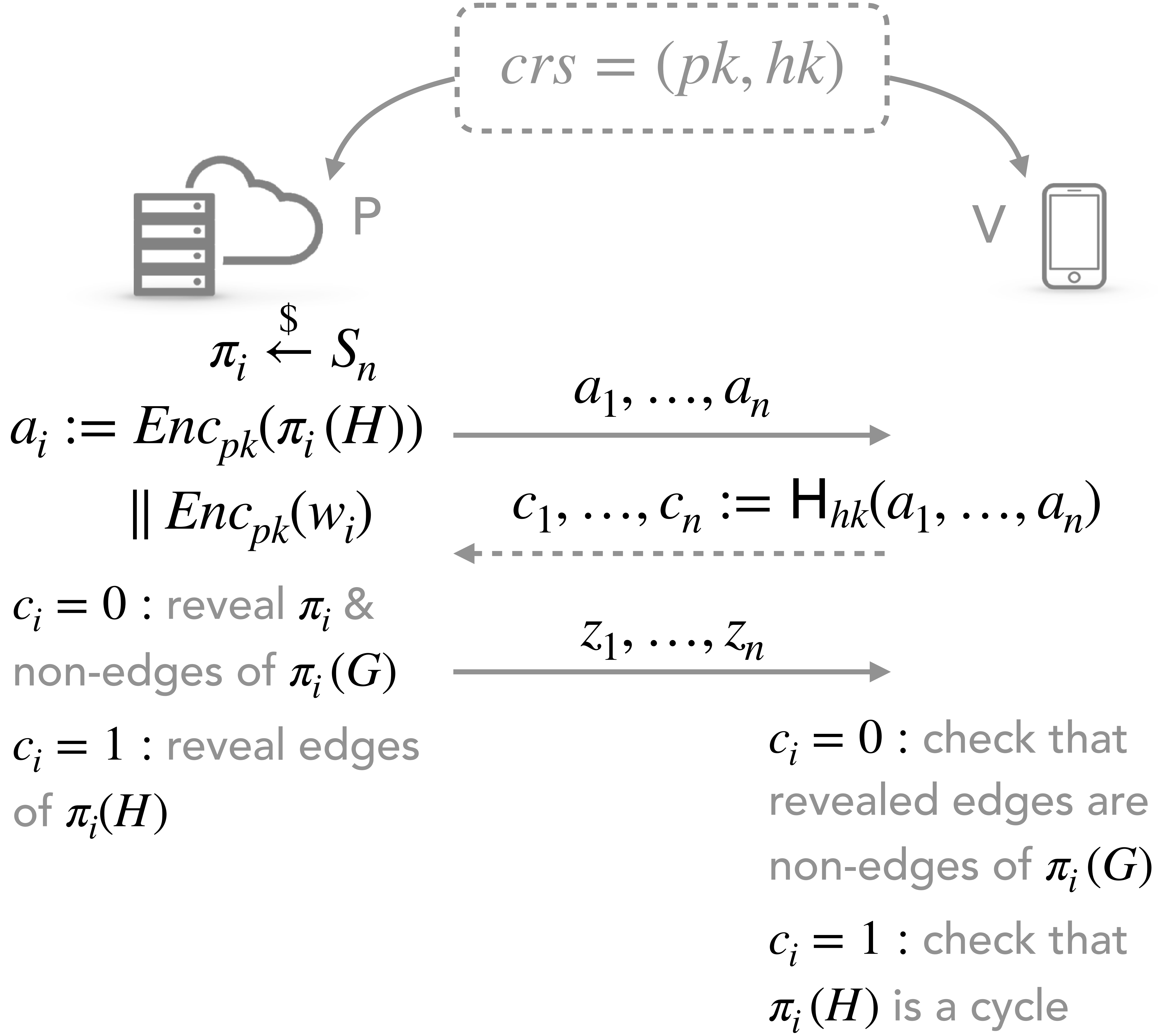
ApproxCI for deg-$d*$ concatenated polys

ApproxMQ / Exponential MQ

ApproxCI for Concatenated Poly's Suffices

ApproxCI for Concatenated Poly's Suffices

Bad challenge function of parallel-repeated protocol has ***concatenated*** format



$crs = (pk, hk)$

$\pi_i \overset{\$}{\leftarrow} S_n$

$a_i := Enc_{pk}(\pi_i(H)) \,\|\, Enc_{pk}(w_i)$

$a_1, \ldots, a_n$

$c_1, \ldots, c_n := \mathsf{H}_{hk}(a_1, \ldots, a_n)$

$c_i = 0$: reveal π_i & non-edges of $\pi_i(G)$

$c_i = 1$: reveal edges of $\pi_i(H)$

$z_1, \ldots, z_n$

$c_i = 0$: check that revealed edges are non-edges of $\pi_i(G)$

$c_i = 1$: check that $\pi_i(H)$ is a cycle

ApproxCI for Concatenated Poly's Suffices

Bad challenge function of parallel-repeated protocol has ***concatenated*** format

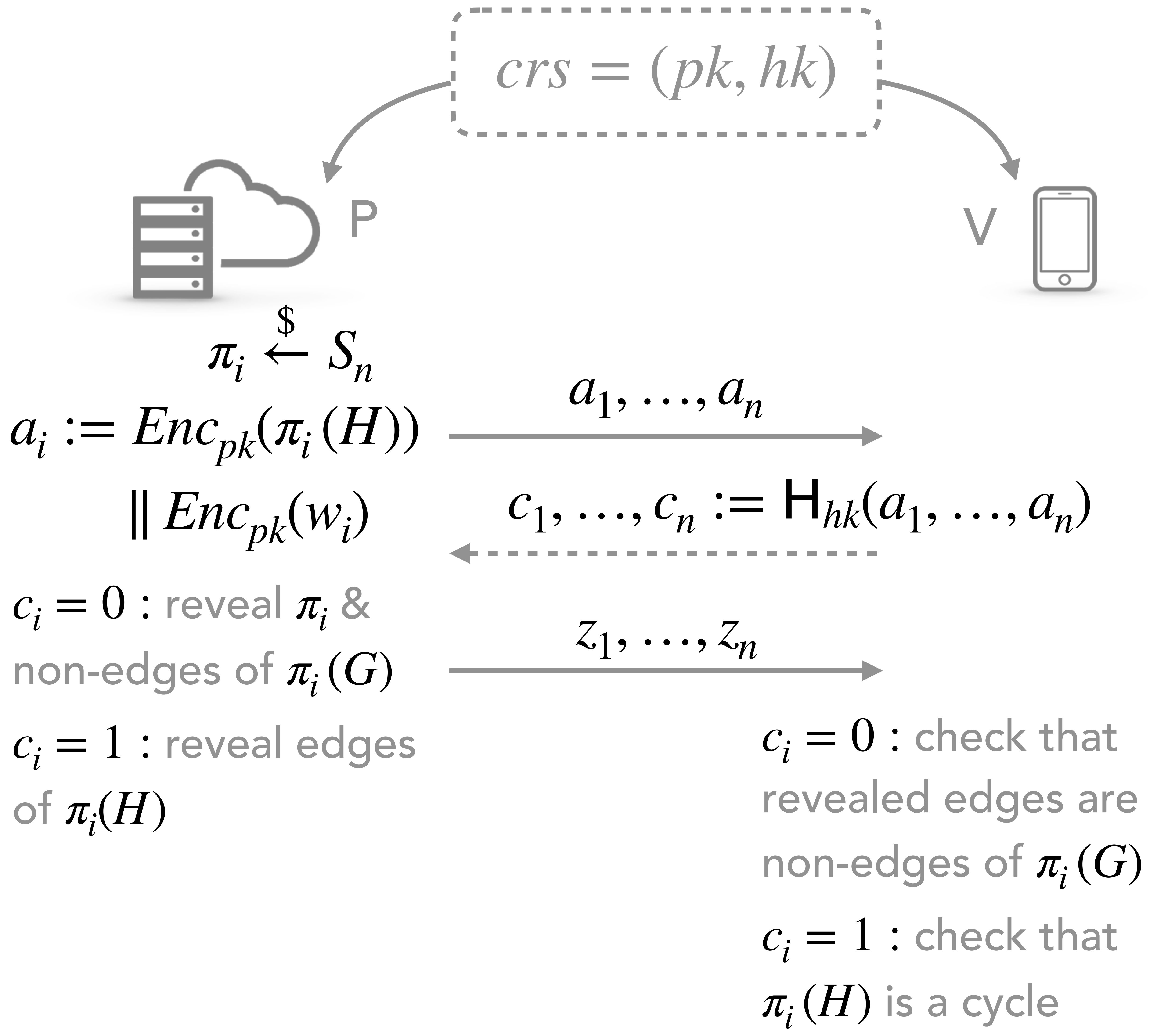
BadChal$_{sk}(a_1, \ldots, a_n)$:

For each $i = 1,\ldots,n$:

- Decrypt $a_i \implies$ get $\pi_i(H), w_i$
- Output $c_i = 0$ if $\Phi(\pi_i(H), w_i) = 1$. Else output $c_i = 1$.

$= \textbf{BadChal}_{sk}(a_1)\|\ldots\|\textbf{BadChal}_{sk}(a_n)$

$\implies$ **BadChal**$_{sk}$ is approximable by ***concatenated*** constant-degree polynomials



$crs = (pk, hk)$

$\pi_i \xleftarrow{\$} S_n$

$a_i := Enc_{pk}(\pi_i(H)) \,\|\, Enc_{pk}(w_i)$

$a_1, \ldots, a_n$

$c_1, \ldots, c_n := \mathsf{H}_{hk}(a_1, \ldots, a_n)$

$c_i = 0$: reveal π_i & non-edges of $\pi_i(G)$

$c_i = 1$: reveal edges of $\pi_i(H)$

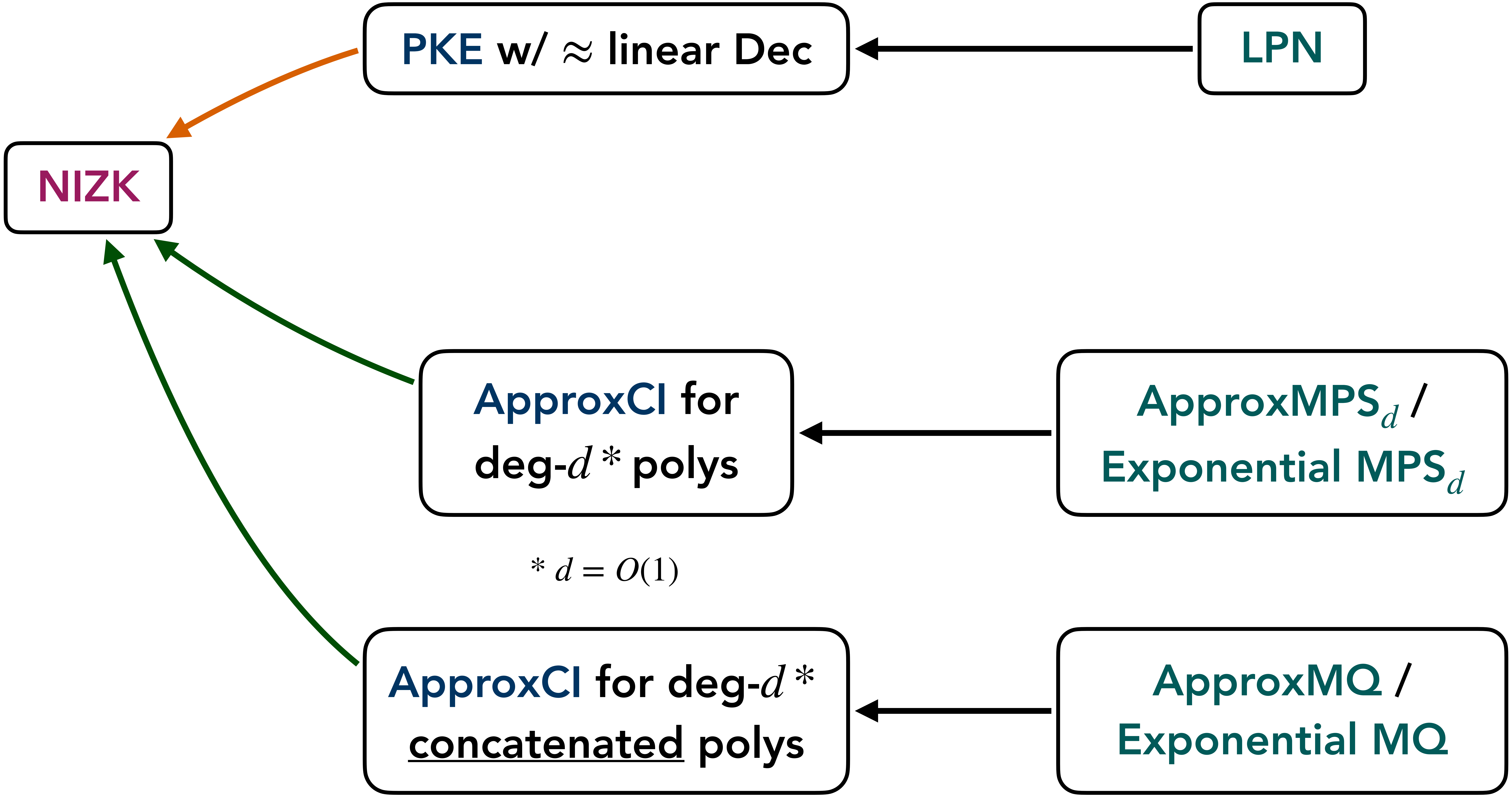
$z_1, \ldots, z_n$

$c_i = 0$: check that revealed edges are non-edges of $\pi_i(G)$

$c_i = 1$: check that $\pi_i(H)$ is a cycle

Statistical ZK via Dense-Sparse LPN

Statistical ZK via Dense-Sparse LPN



PKE w/ ≈ linear Dec

LPN

NIZK

ApproxCI for deg-d* polys

ApproxMPS$_d$ / Exponential MPS$_d$

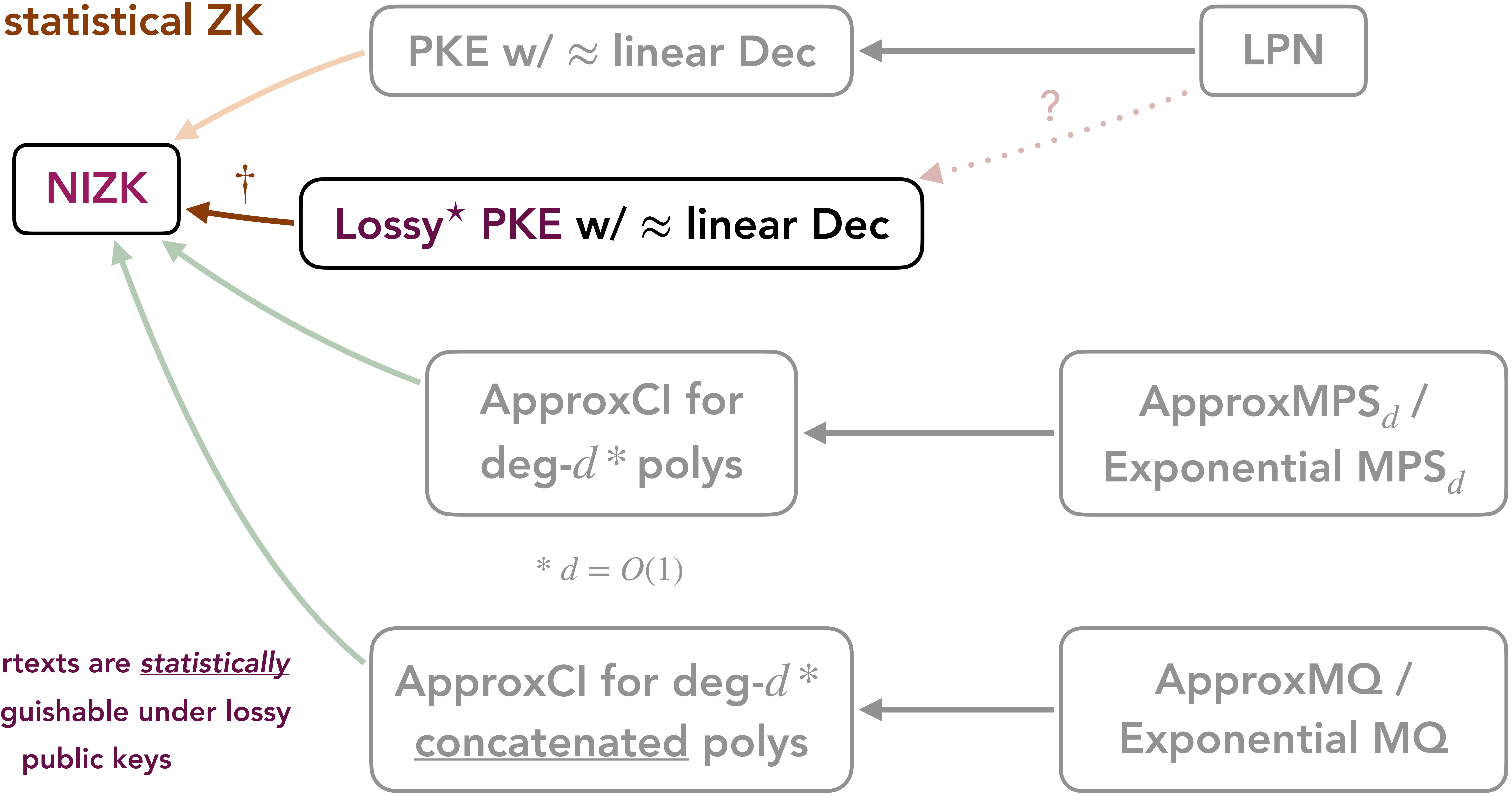
$* \; d = O(1)$

ApproxCI for deg-d* concatenated polys

ApproxMQ / Exponential MQ

Statistical ZK via Dense-Sparse LPN

† For statistical ZK



PKE w/ $\approx$ linear Dec

LPN

?

NIZK

†

Lossy* PKE w/ $\approx$ linear Dec

ApproxCI for deg-d* polys

ApproxMPS$_d$ / Exponential MPS$_d$

$* \ d = O(1)$

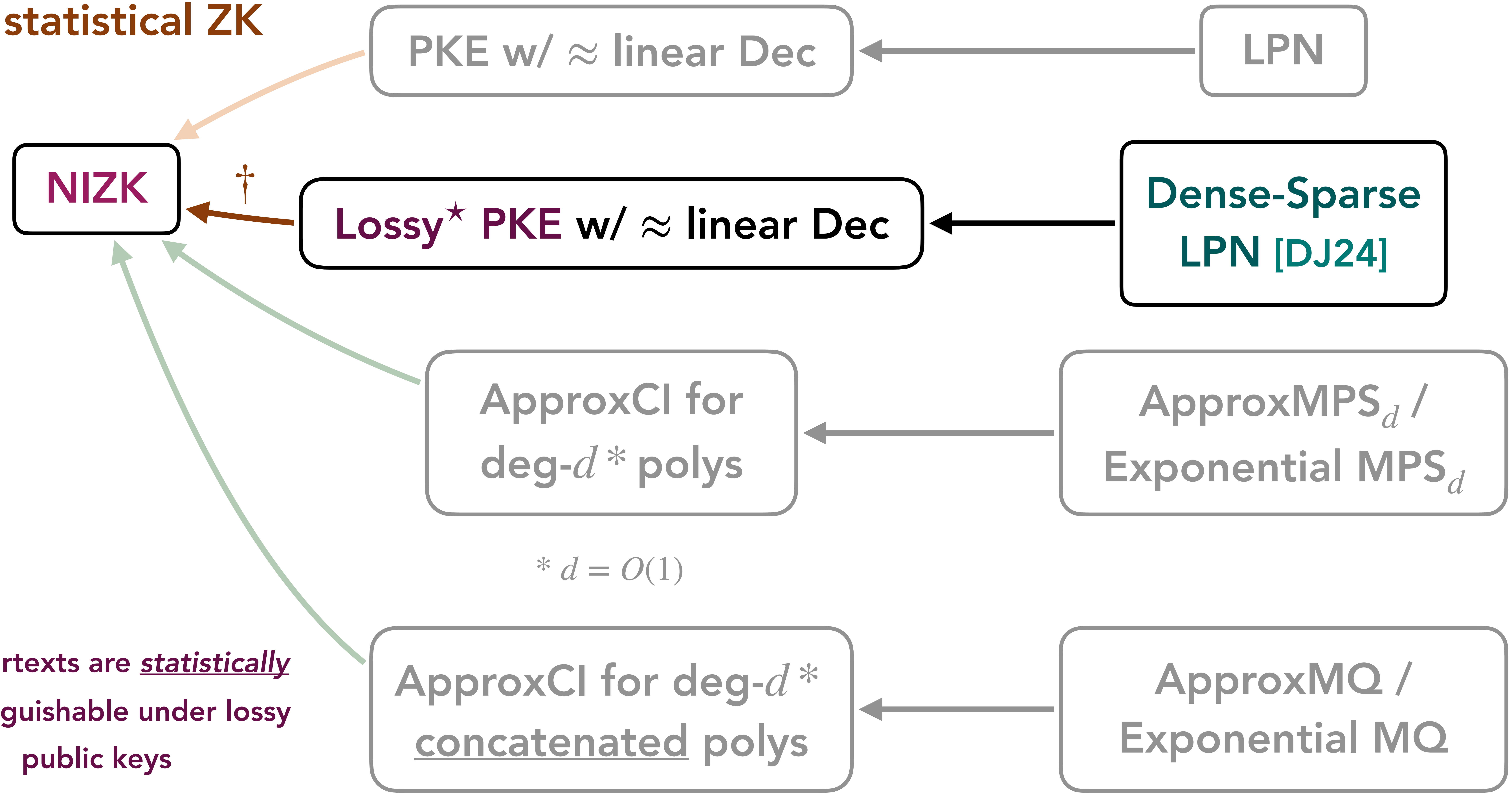
ApproxCI for deg-d* concatenated polys

ApproxMQ / Exponential MQ

* ciphertexts are *statistically* indistinguishable under lossy public keys

Statistical ZK via Dense-Sparse LPN

† For statistical ZK



PKE w/ ≈ linear Dec

LPN

NIZK

†

Lossy* PKE w/ ≈ linear Dec

Dense-Sparse LPN [DJ24]

ApproxCI for deg-d* polys

ApproxMPS$_d$ / Exponential MPS$_d$

$* \; d = O(1)$

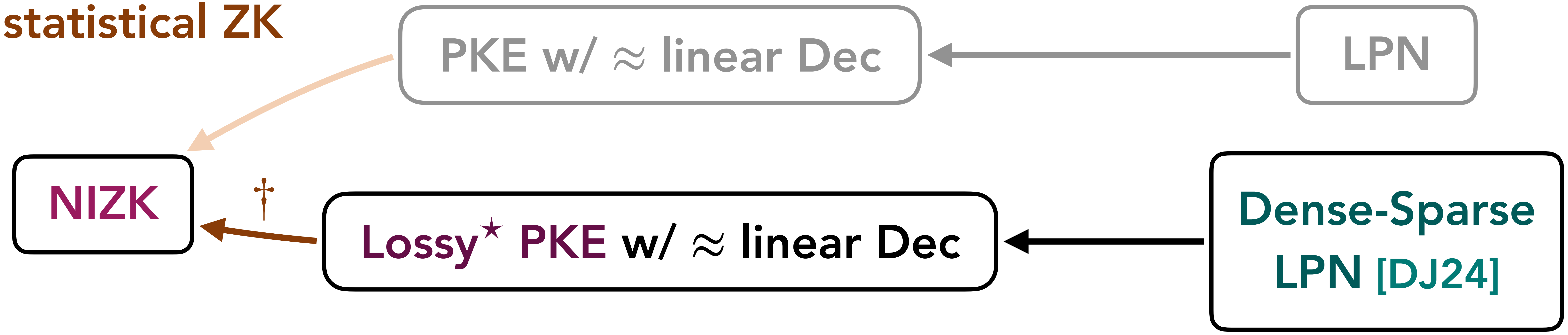
ApproxCI for deg-d* concatenated polys

ApproxMQ / Exponential MQ

* ciphertexts are *statistically* indistinguishable under lossy public keys

Statistical ZK via Dense-Sparse LPN

† For statistical ZK



PKE w/ ≈ linear Dec

LPN

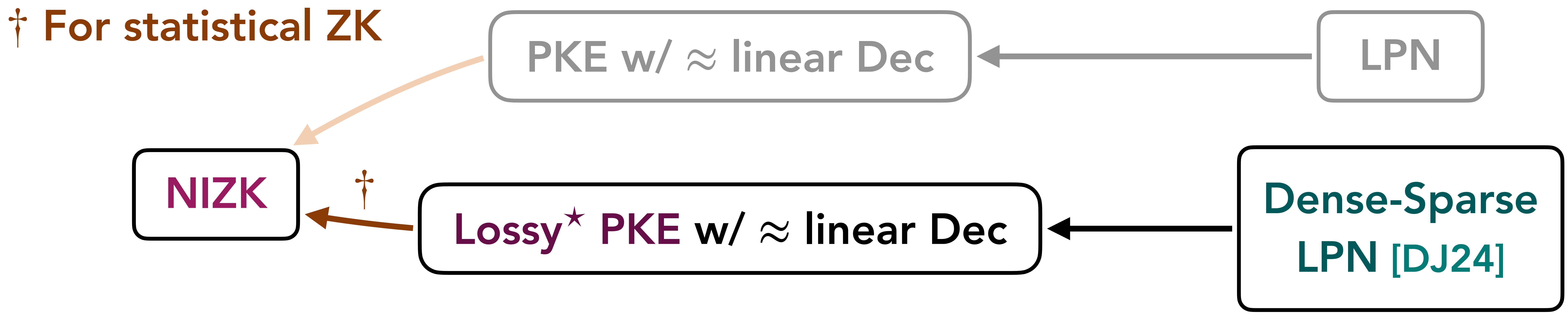
NIZK

†

Lossy* PKE w/ ≈ linear Dec

Dense-Sparse LPN [DJ24]

Statistical ZK via Dense-Sparse LPN



† For statistical ZK

PKE w/ ≈ linear Dec

LPN

NIZK

†

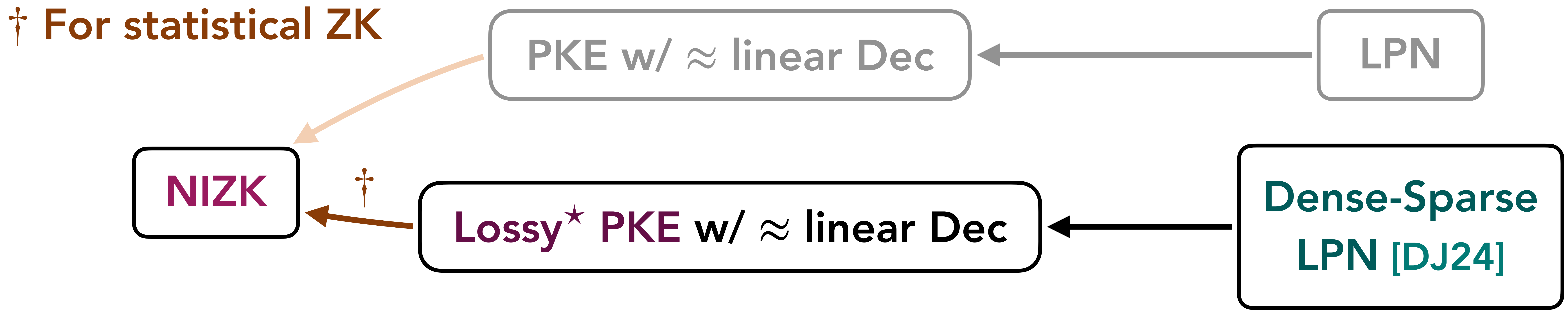
Lossy* PKE w/ ≈ linear Dec

Dense-Sparse LPN [DJ24]

Problem: lossy PKE from LPN has correctness error $\frac{1}{2} - \frac{1}{poly(\lambda)}$

$\Longrightarrow$ not strong enough to achieve NIZK

Statistical ZK via Dense-Sparse LPN



† For statistical ZK

PKE w/ ≈ linear Dec

LPN

NIZK

†

Lossy* PKE w/ ≈ linear Dec

Dense-Sparse LPN [DJ24]

Problem: lossy PKE from LPN has correctness error $\frac{1}{2} - \frac{1}{poly(\lambda)}$

$\implies$ not strong enough to achieve NIZK

Instead, use Dense-Sparse LPN (variant of LPN with structured public matrix)

$\implies$ lossy PKE from DS-LPN with correctness error $\frac{1}{poly(\lambda)}$

Summary & Open Problems

Summary & Open Problems

Our Result: We build NIZK from two well-studied post-quantum assumptions, Learning Parity with Noise (LPN) and Multivariate Quadratic (MQ).

Summary & Open Problems

Our Result: We build NIZK from two well-studied post-quantum assumptions, Learning Parity with Noise (LPN) and Multivariate Quadratic (MQ).

Future Directions:

- NIZK solely from code-based / multivariate assumptions?
- New post-quantum constructions of advanced proof systems?
  - ZAPs, BARGs, SNARGs, etc.
- Cryptanalysis on higher-degree analogue of MQ

Summary & Open Problems

Our Result: We build NIZK from two well-studied post-quantum assumptions, Learning Parity with Noise (LPN) and Multivariate Quadratic (MQ).

Future Directions:

- NIZK solely from code-based / multivariate assumptions?
- New post-quantum constructions of advanced proof systems?
  - ZAPs, BARGs, SNARGs, etc.
- Cryptanalysis on higher-degree analogue of MQ

Read our paper!
(ePrint 2024/1254)

Thank you! Questions?