Functional Encryption for Inner-Product Evaluations

Florian Bourse

École normale supérieure, CNRS, INRIA, PSL, Paris, France

PhD Defense — Paris
Wednesday, December 13

Public Key Encryption

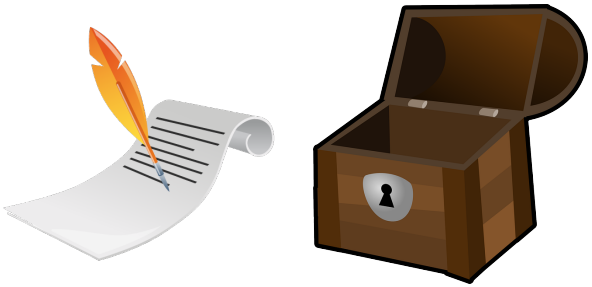
Bob

Alice

Public Key Encryption

Public Key Encryption

Bob



Alice

Public Key Encryption

Bob

Alice

Public Key Encryption

Public Key Encryption

Bob

Alice

Public Key Encryption

Bob

Alice

Inner-Product Functional Encryption

Inner-Product Functional Encryption

Applications:

Inner-Product Functional Encryption

Applications:

- evaluate statistics on encrypted data;

Inner-Product Functional Encryption

Applications:

- evaluate statistics on encrypted data;
- building block for more general Functional Encryption[AR17];

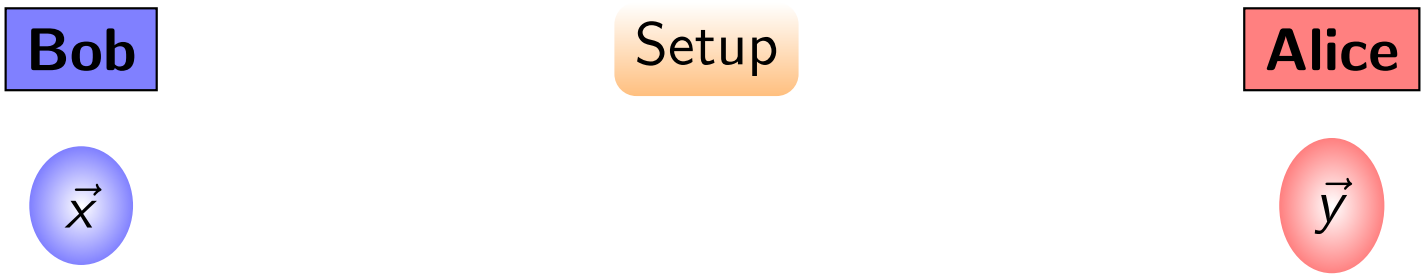
Inner-Product Functional Encryption

Applications:

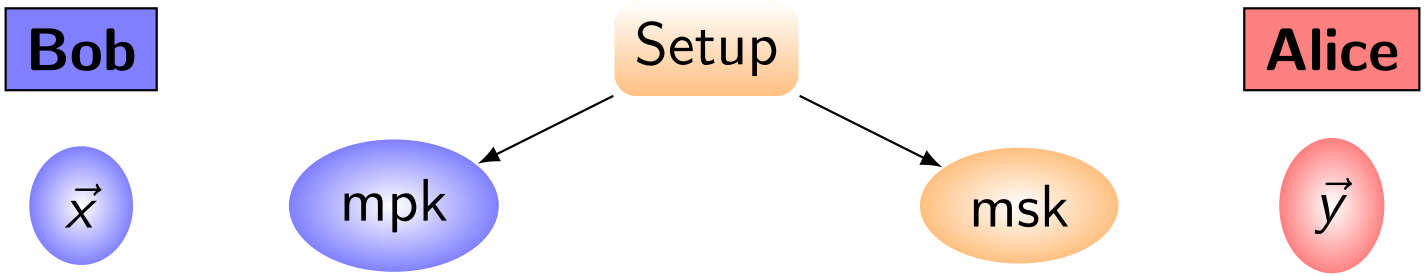
- evaluate statistics on encrypted data;
- building block for more general Functional Encryption[AR17];
- traitor tracing[ABP+17];
- ...

Inner-Product Functional Encryption[ABDP15]

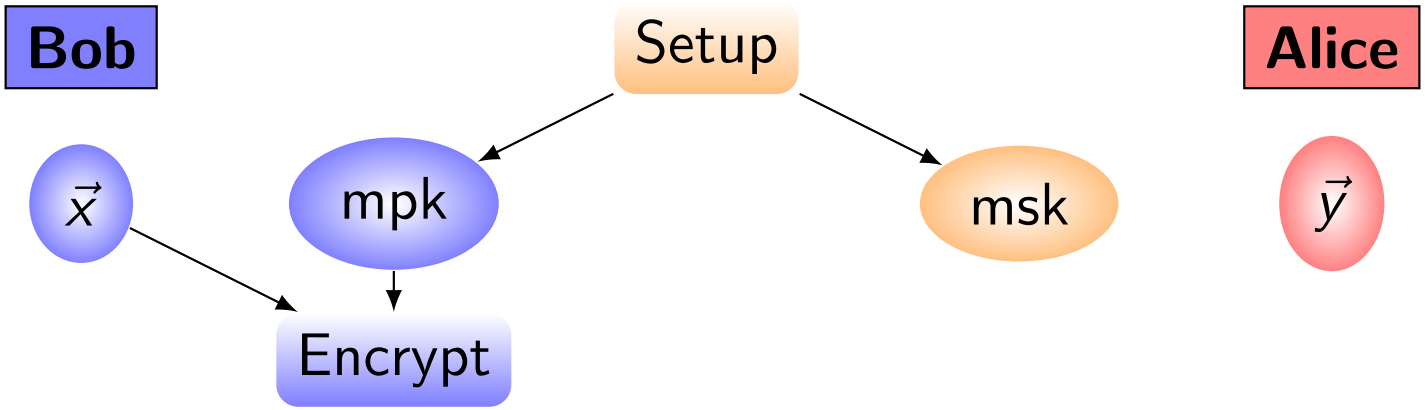
Inner-Product Functional Encryption[ABDP15]



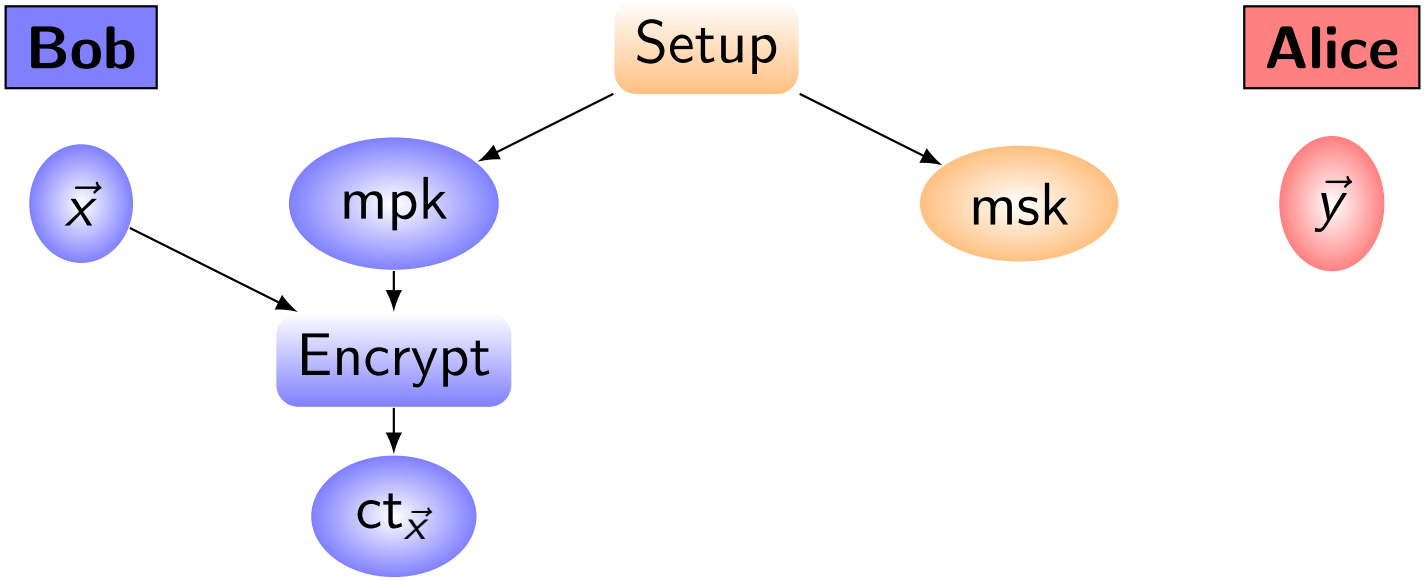
Inner-Product Functional Encryption[ABDP15]



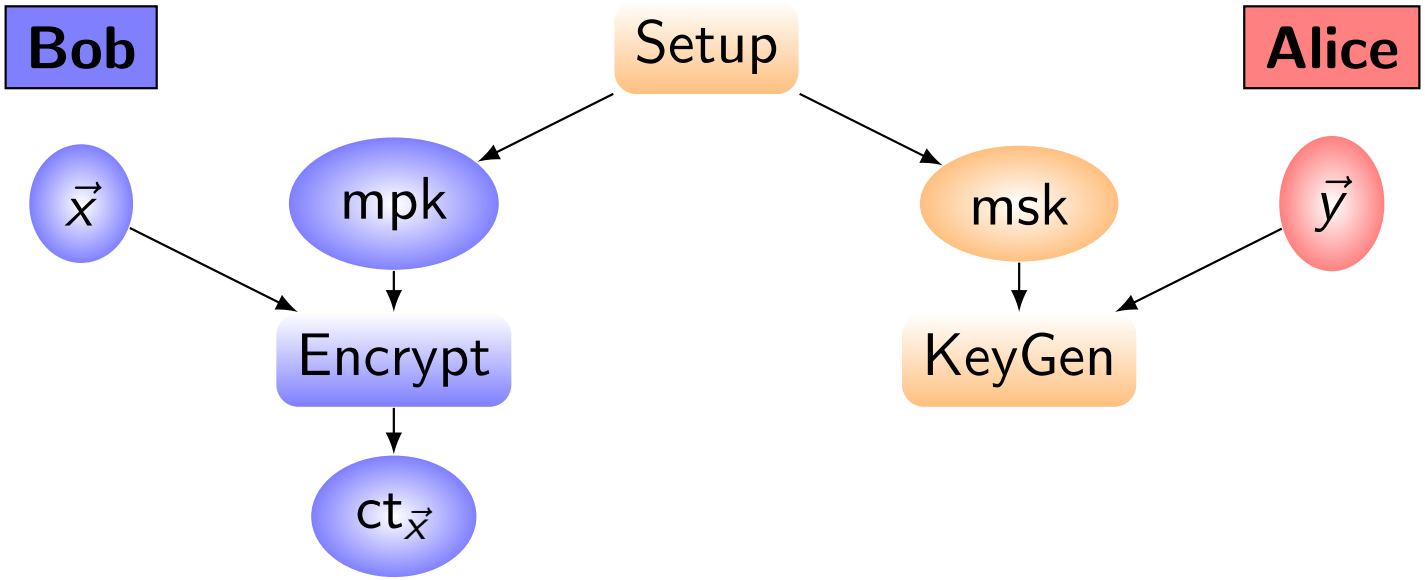
Inner-Product Functional Encryption[ABDP15]



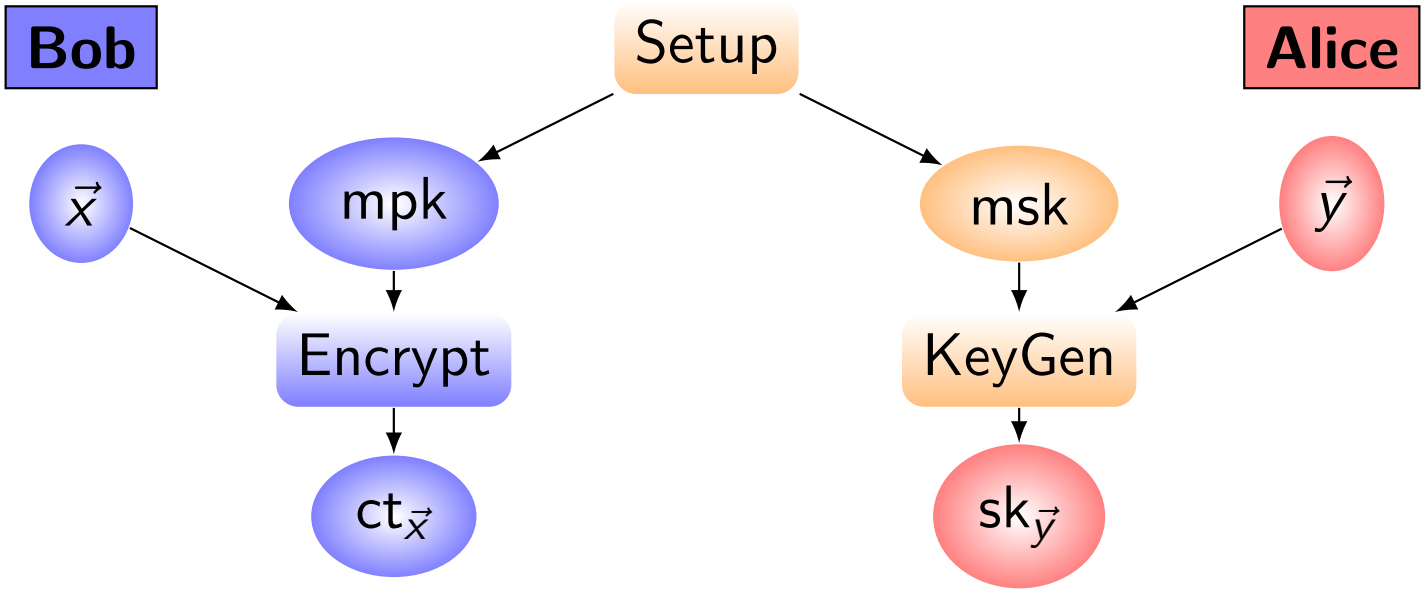
Inner-Product Functional Encryption[ABDP15]



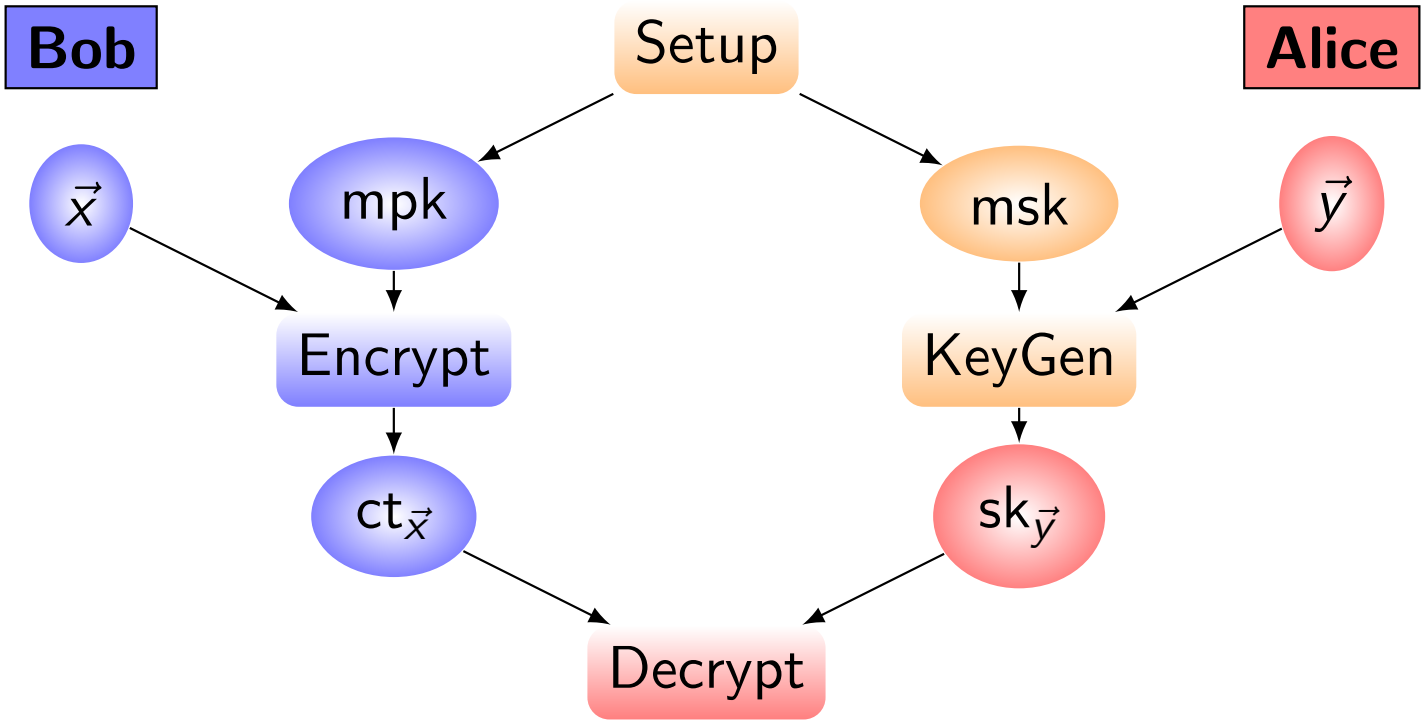
Inner-Product Functional Encryption[ABDP15]



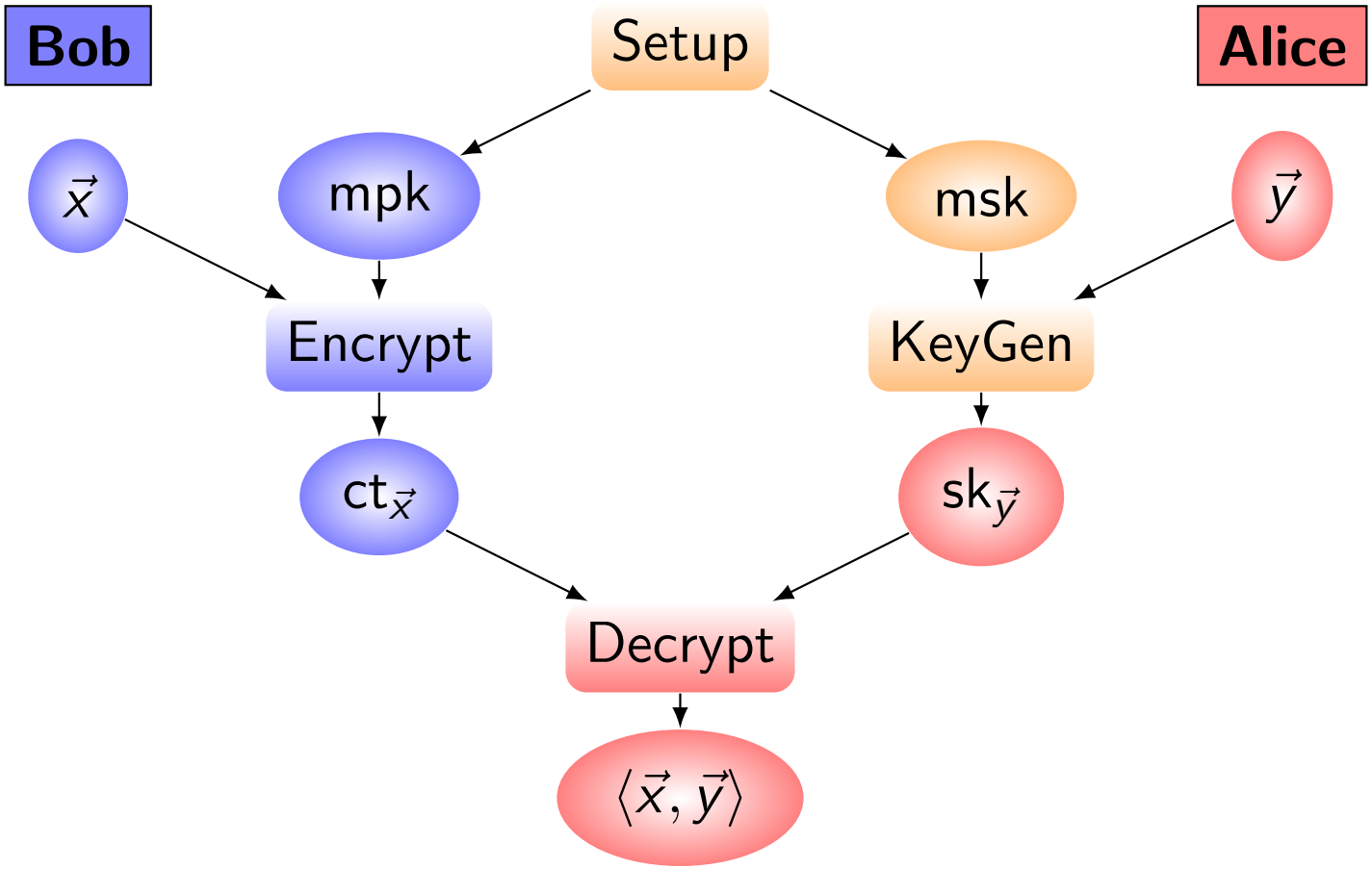
Inner-Product Functional Encryption[ABDP15]



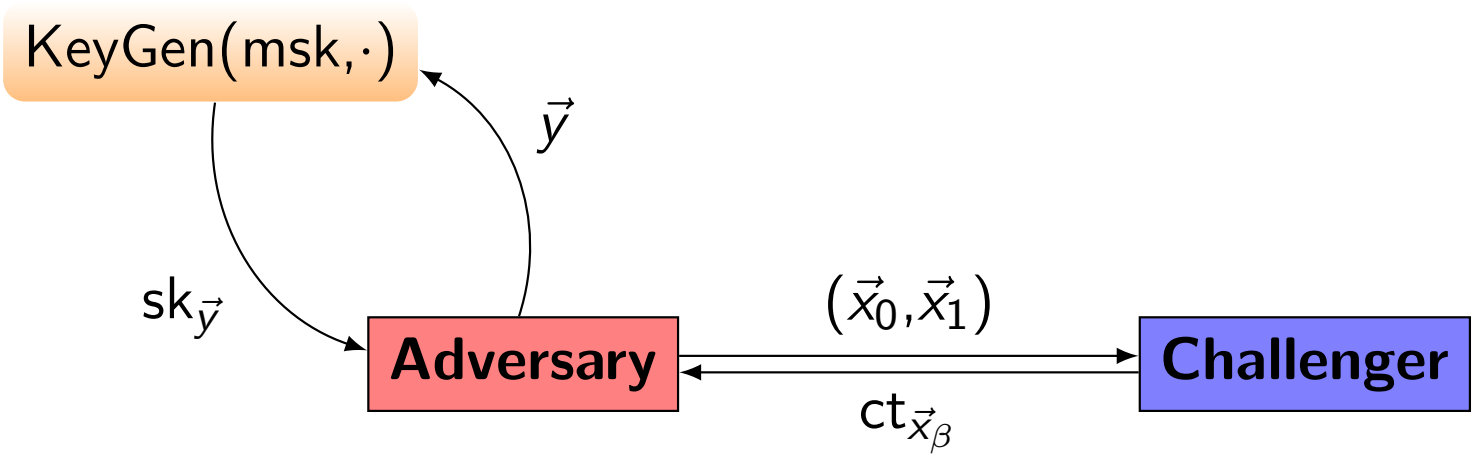
Inner-Product Functional Encryption[ABDP15]



Inner-Product Functional Encryption[ABDP15]

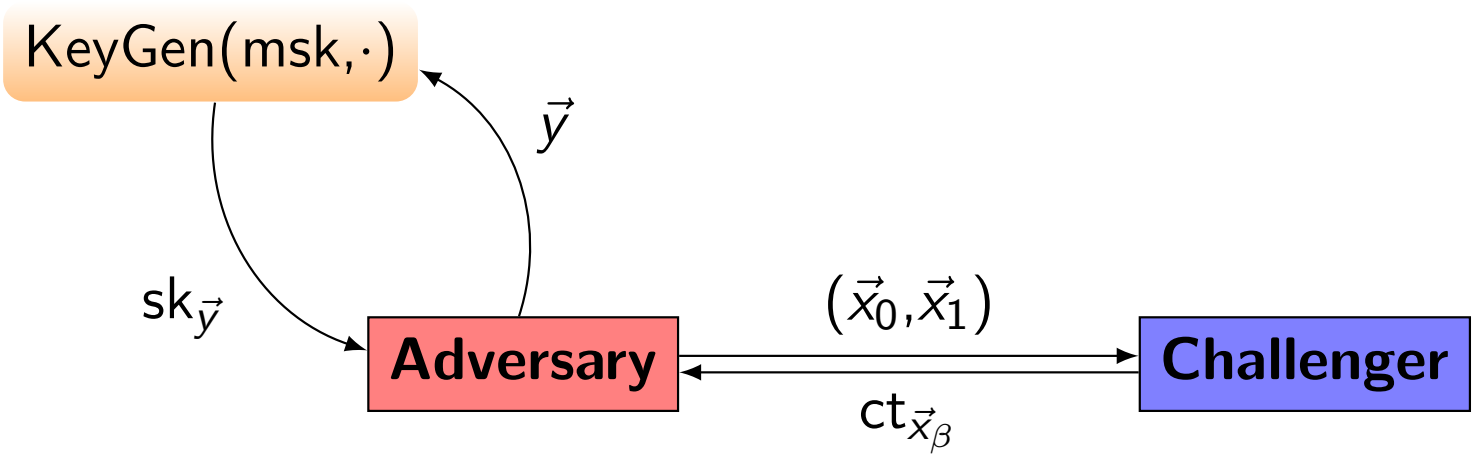


IND-CPA security



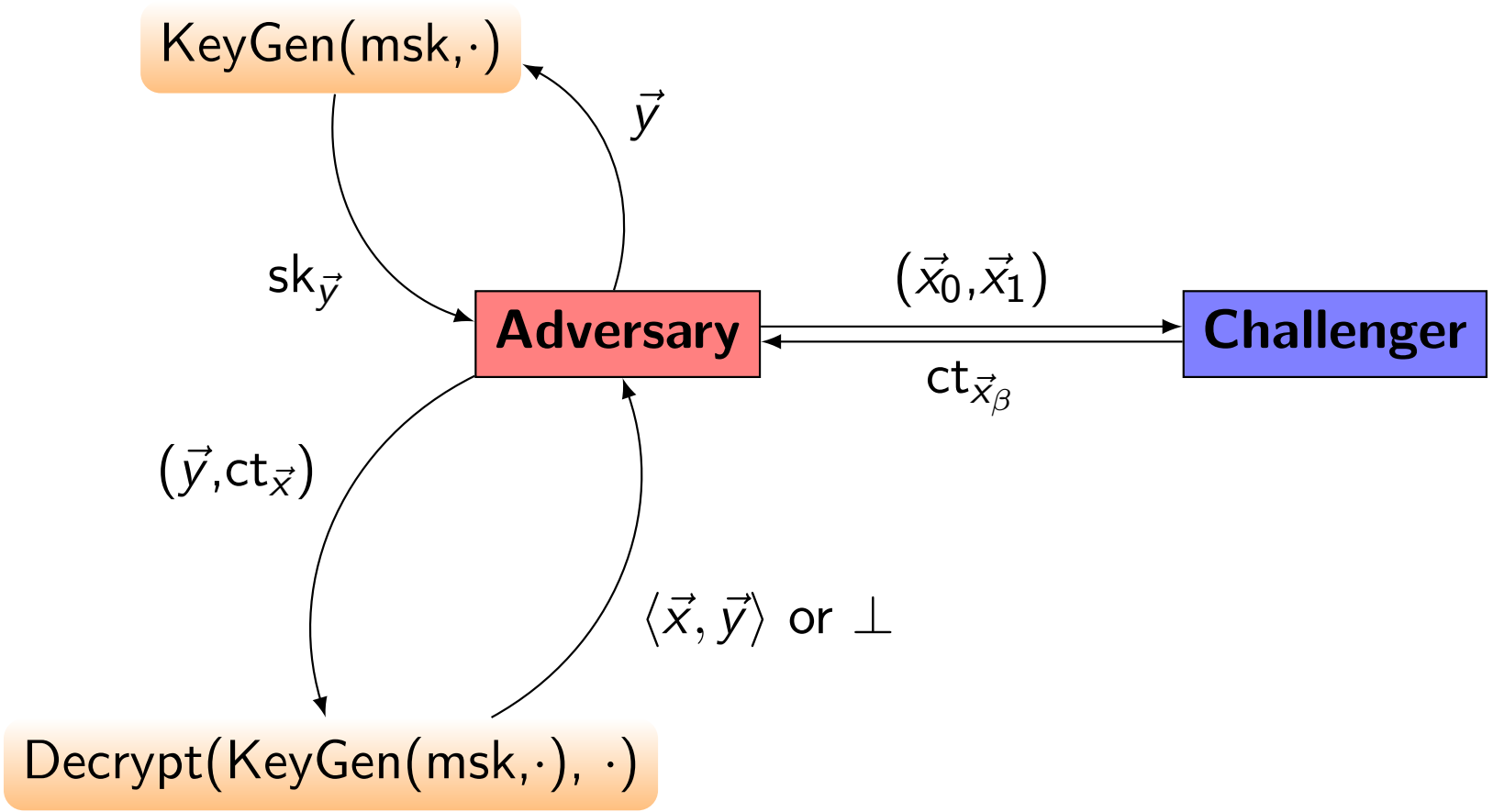
Adversary wins if $\beta' = \beta \wedge \forall \vec{y}, \langle \vec{x}_0, \vec{y} \rangle = \langle \vec{x}_1, \vec{y} \rangle$

IND-CPA security



Adversary wins if $\beta' = \beta \wedge \forall \vec{y}, \vec{y} \in (\vec{x}_1 - \vec{x}_0)^\perp$

IND-CCA security



Adversary wins if $\beta' = \beta \wedge \forall \vec{y}, \vec{y} \in (\vec{x}_1 - \vec{x}_0)^\perp$

NP language

Let $\mathcal{L} = \{\mathbf{b}/\exists w \text{ s.t. } \mathcal{R}(w, \mathbf{b}) = 1\}$ be an NP language.

NP language

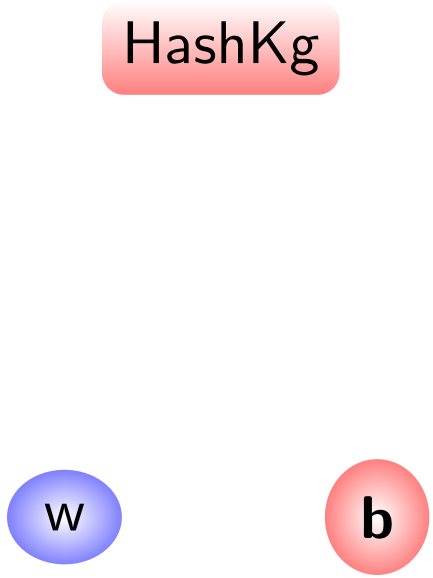
Let $\mathcal{L} = \{\mathbf{b}/\exists w \text{ s.t. } \mathcal{R}(w, \mathbf{b}) = 1\}$ be an NP language.

Example: DDH, fix $(g, h) \in \mathbb{G}^2$

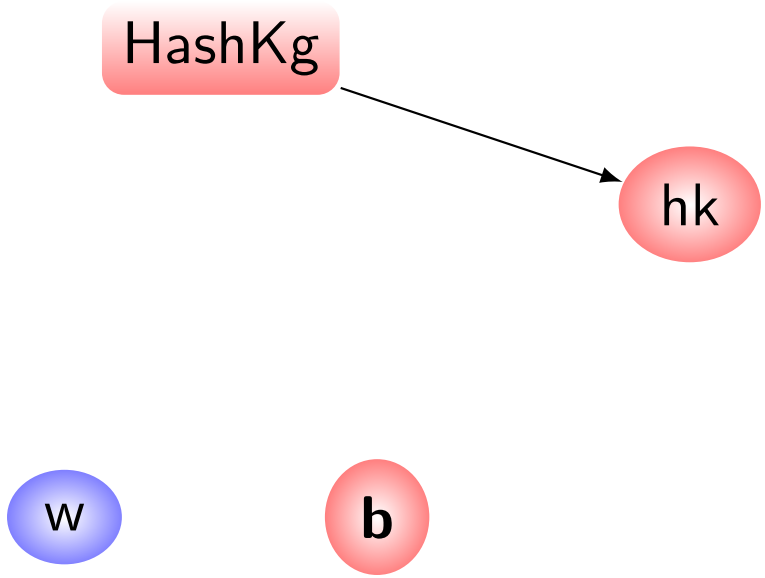
$$\mathcal{L} = \{(g^r, h^r)\}_{r \in \mathbb{Z}_p}$$

Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$

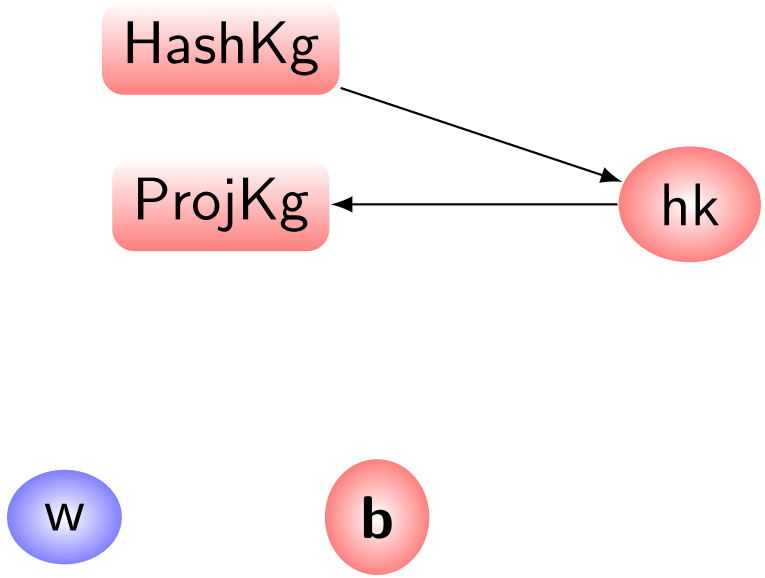
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



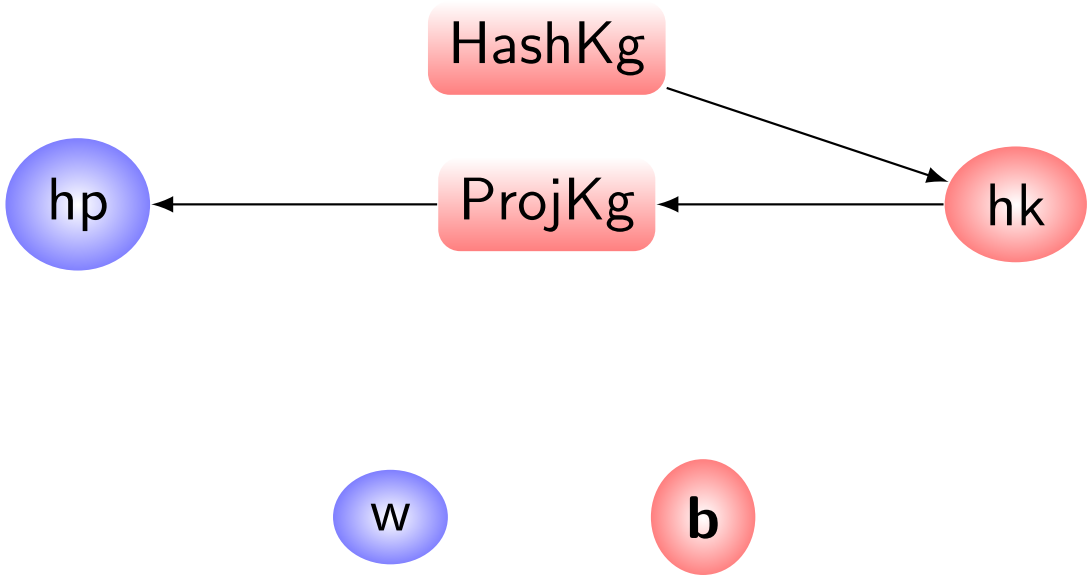
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



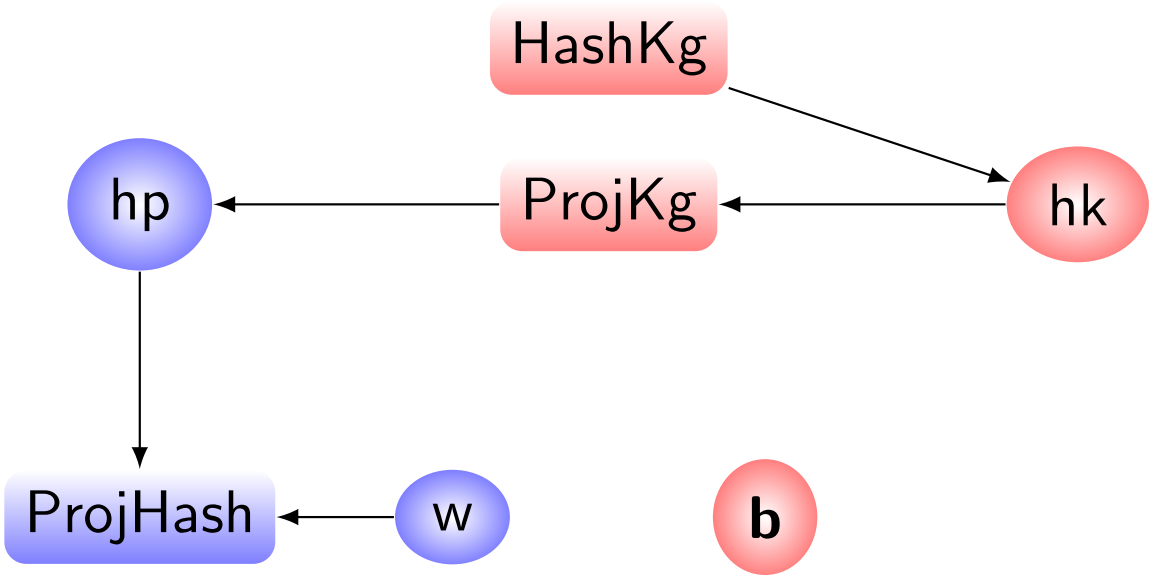
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



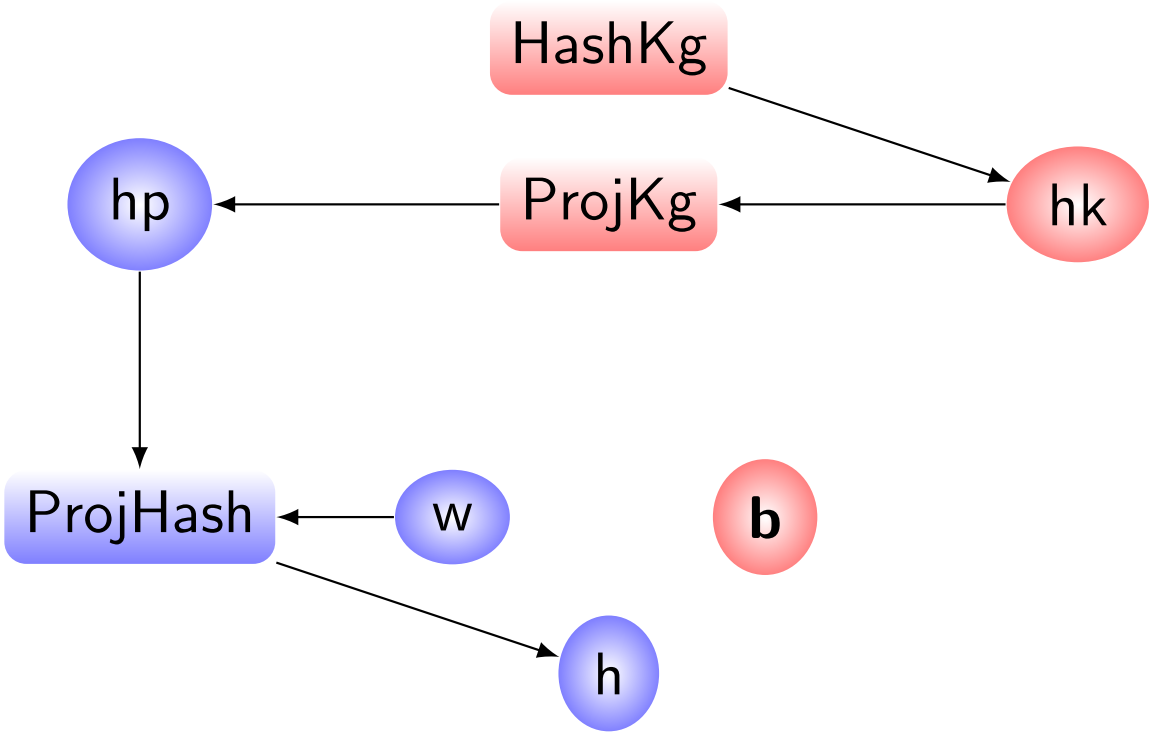
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



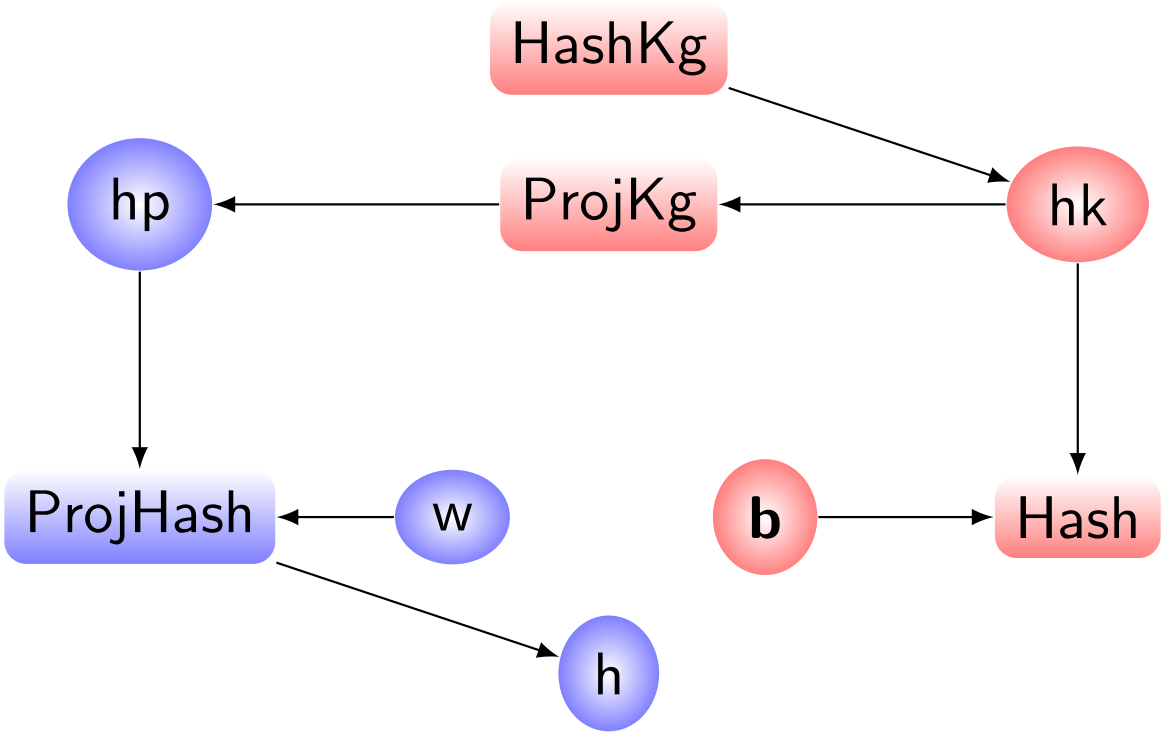
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



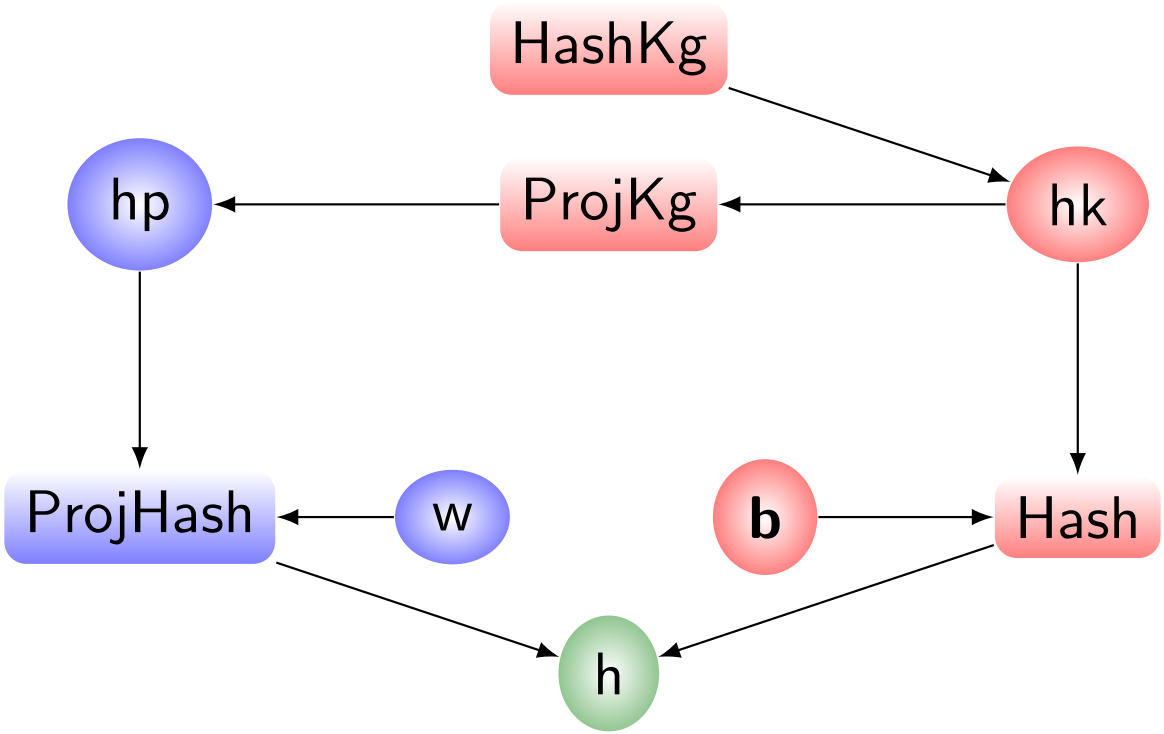
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



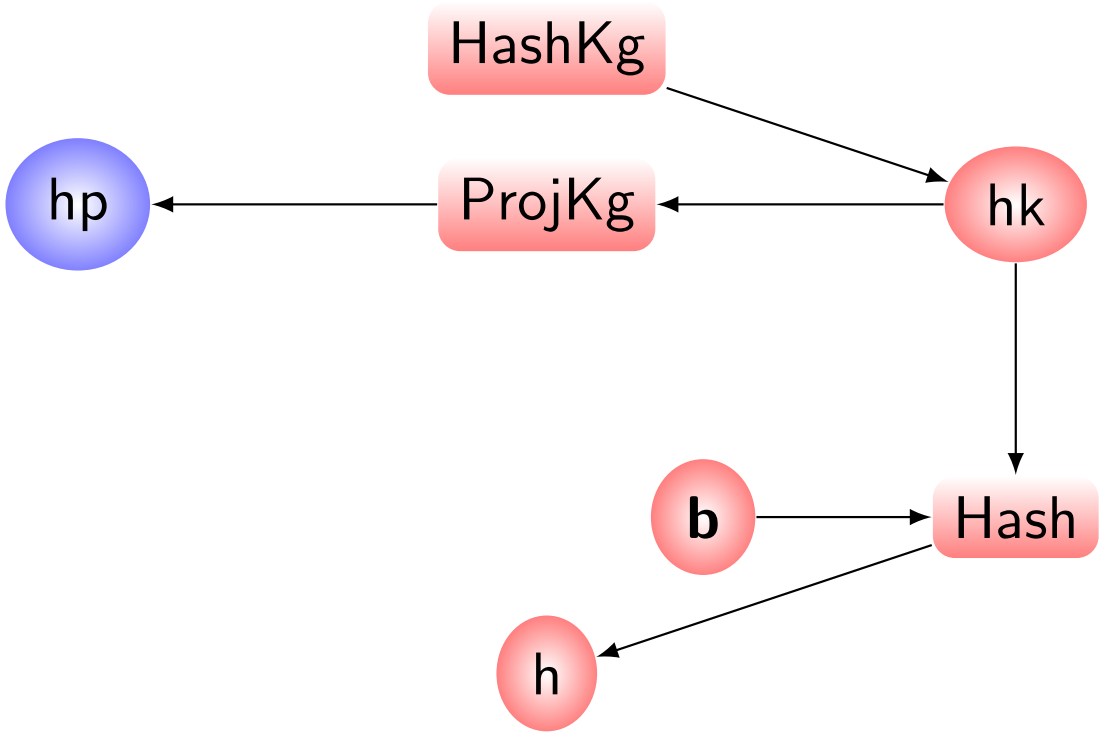
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



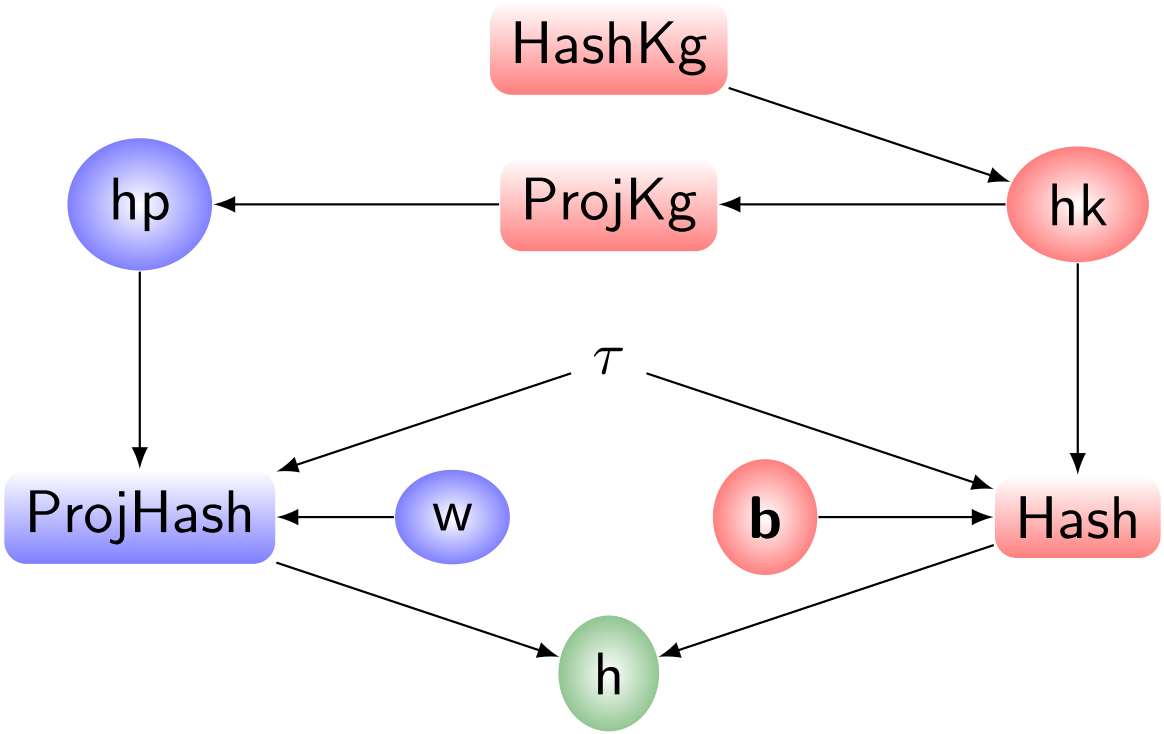
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



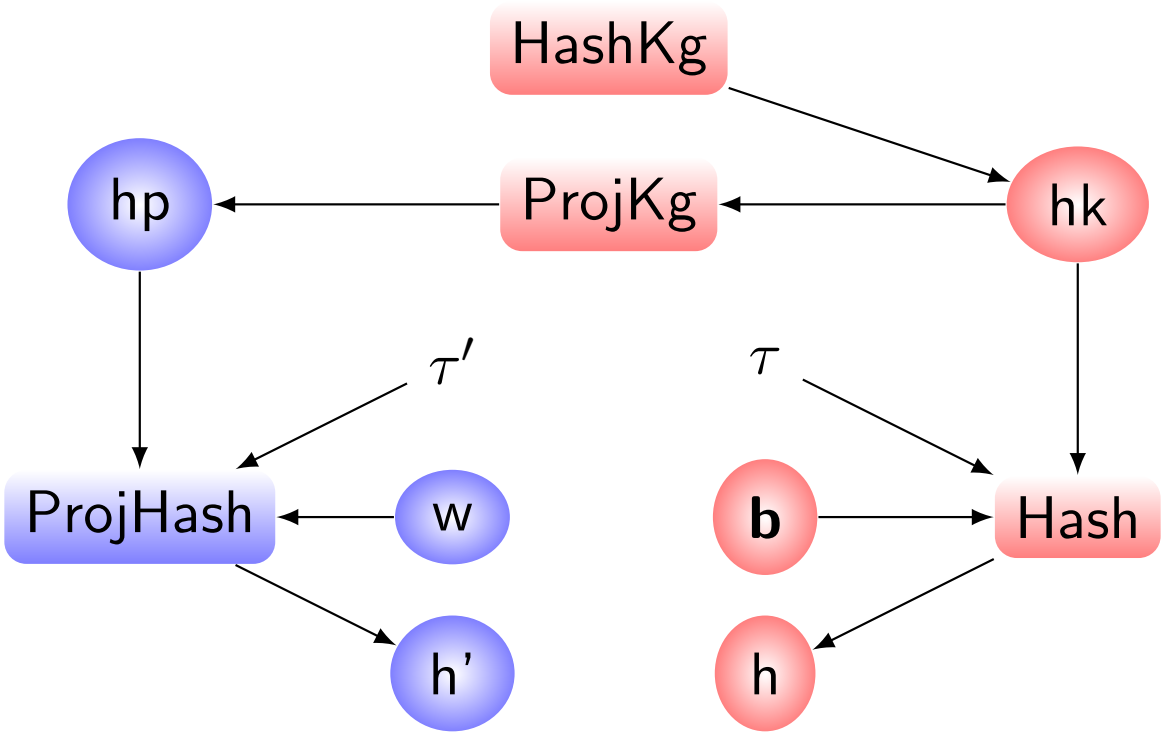
Projective Hash Function: $\mathbf{b} \notin \mathcal{L}$



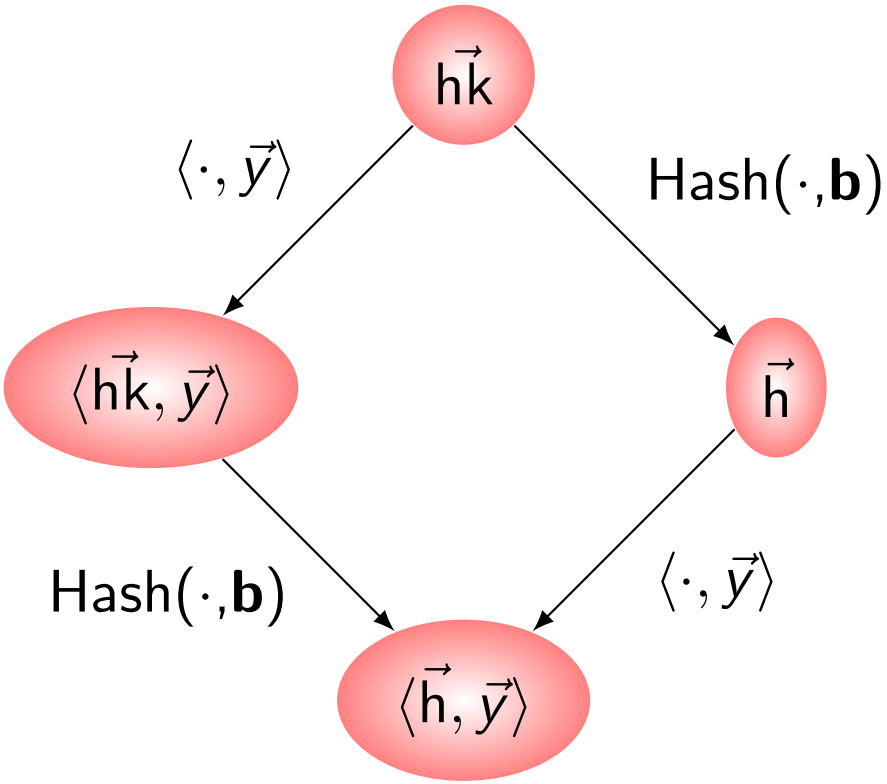
Tag-Based PHF



Tag-Based PHF



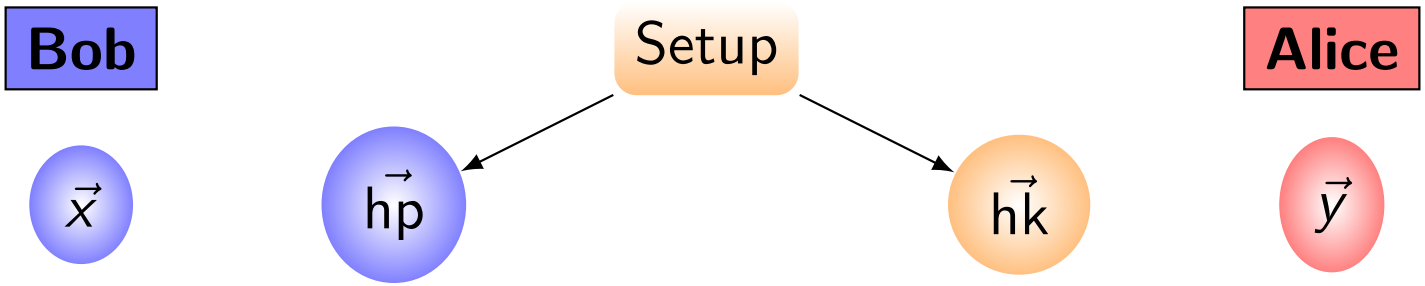
Hash is an Additive Homomorphism



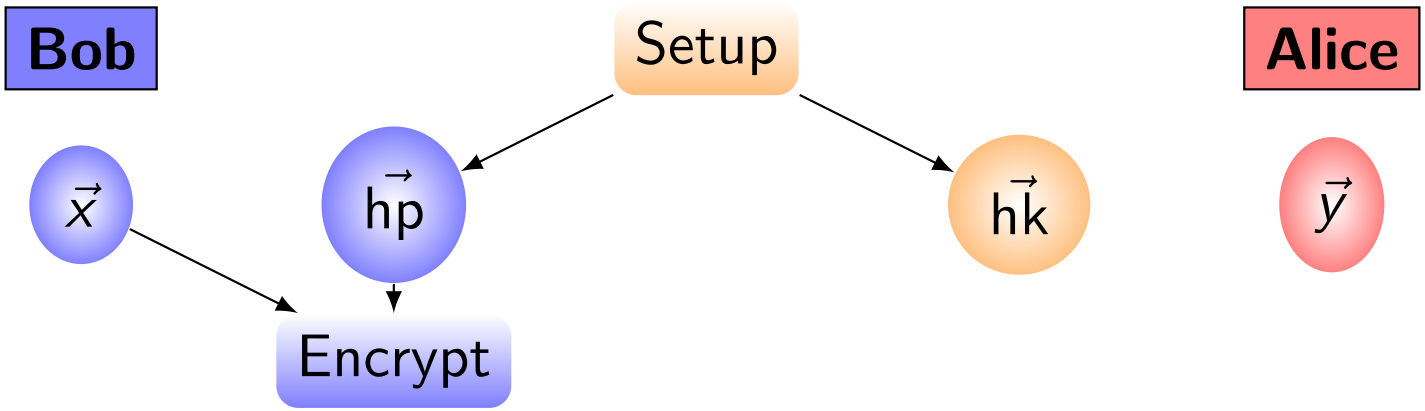
IPFE from PHF

IPFE from PHF

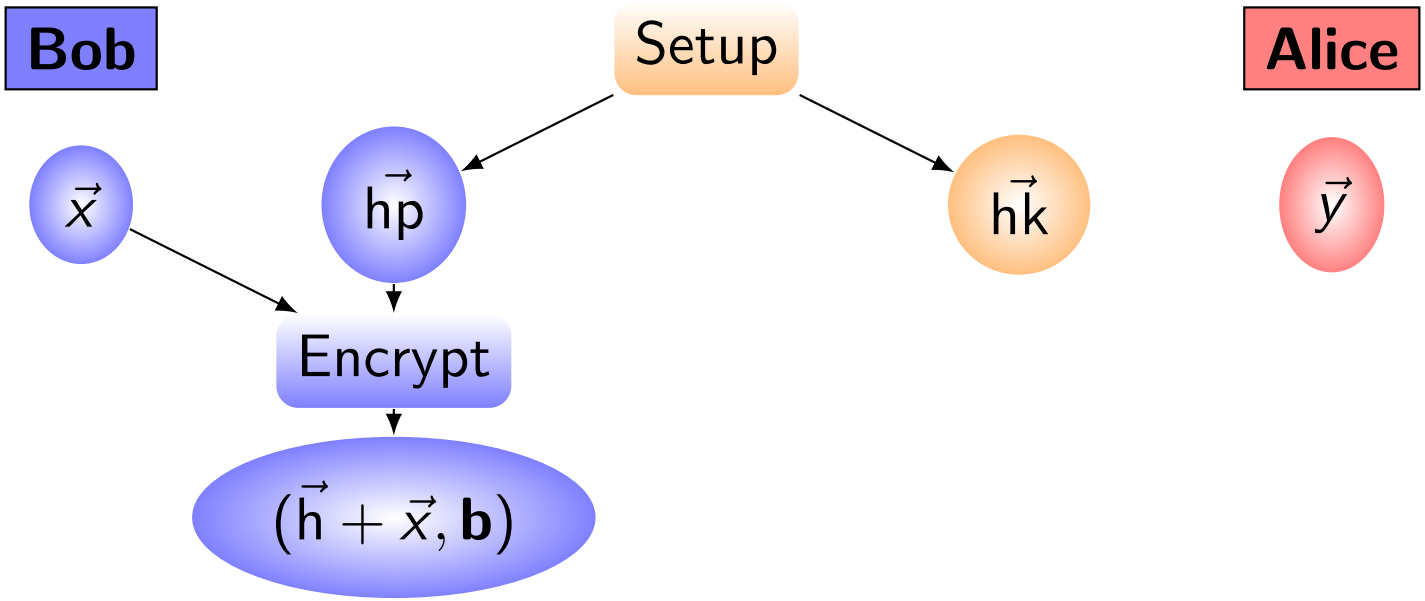
IPFE from PHF



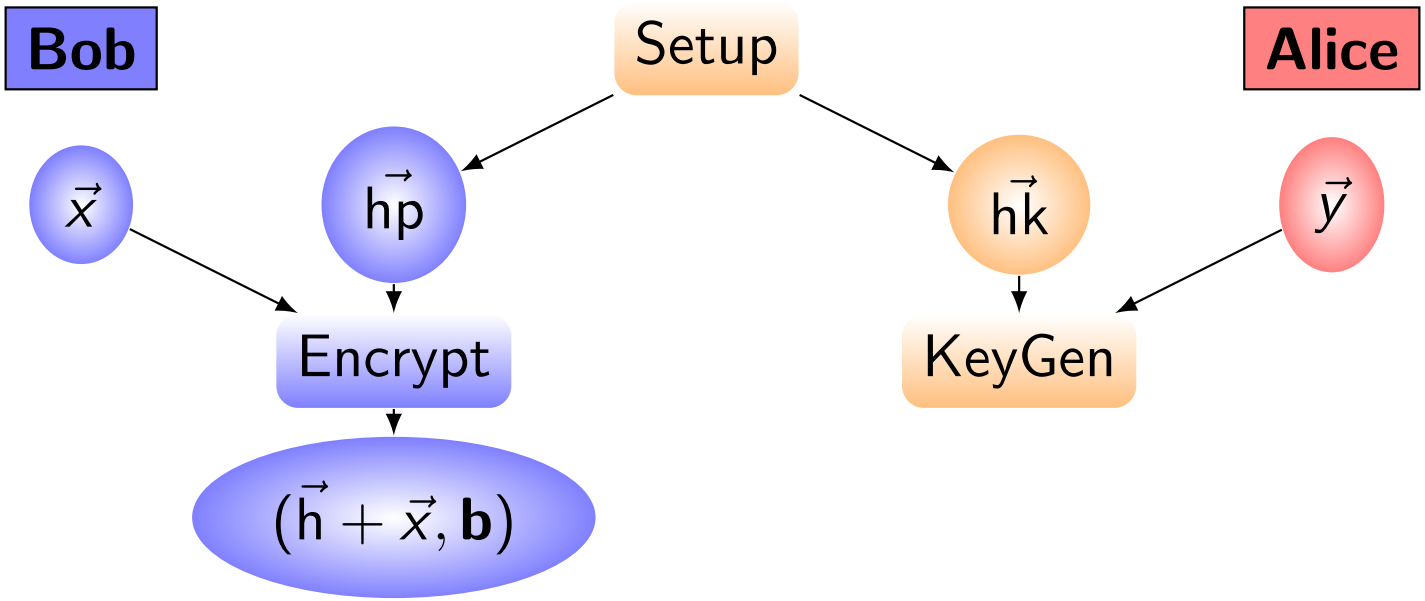
IPFE from PHF



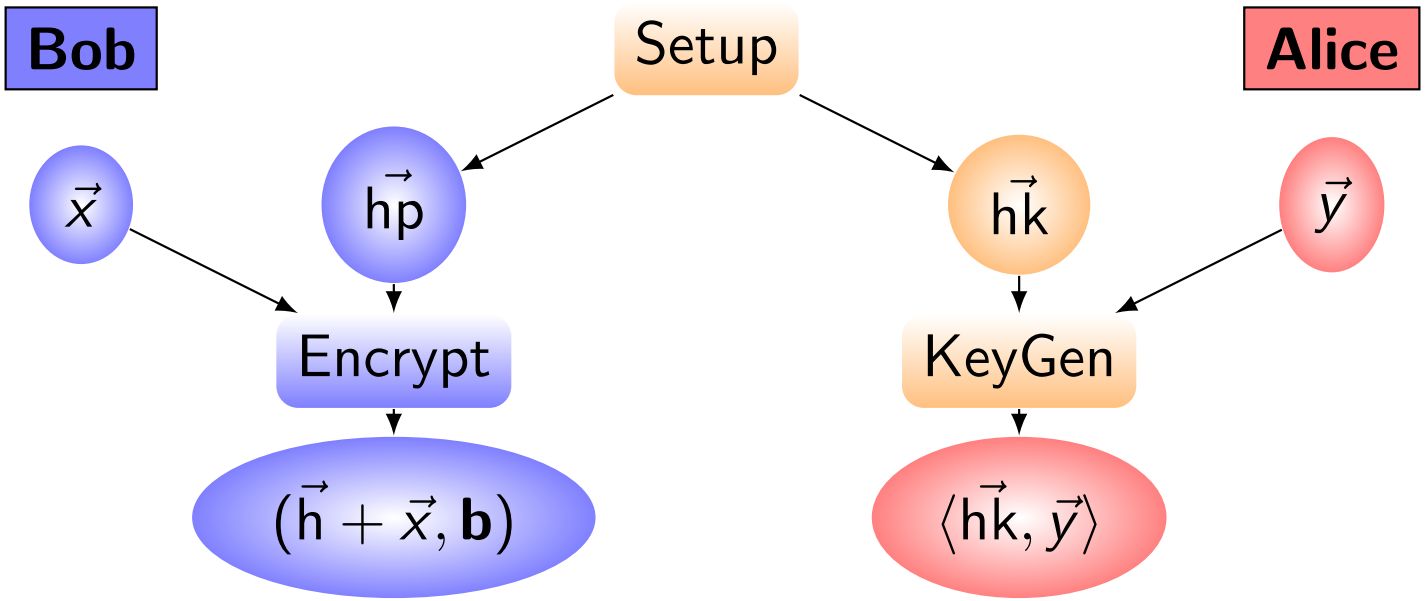
IPFE from PHF



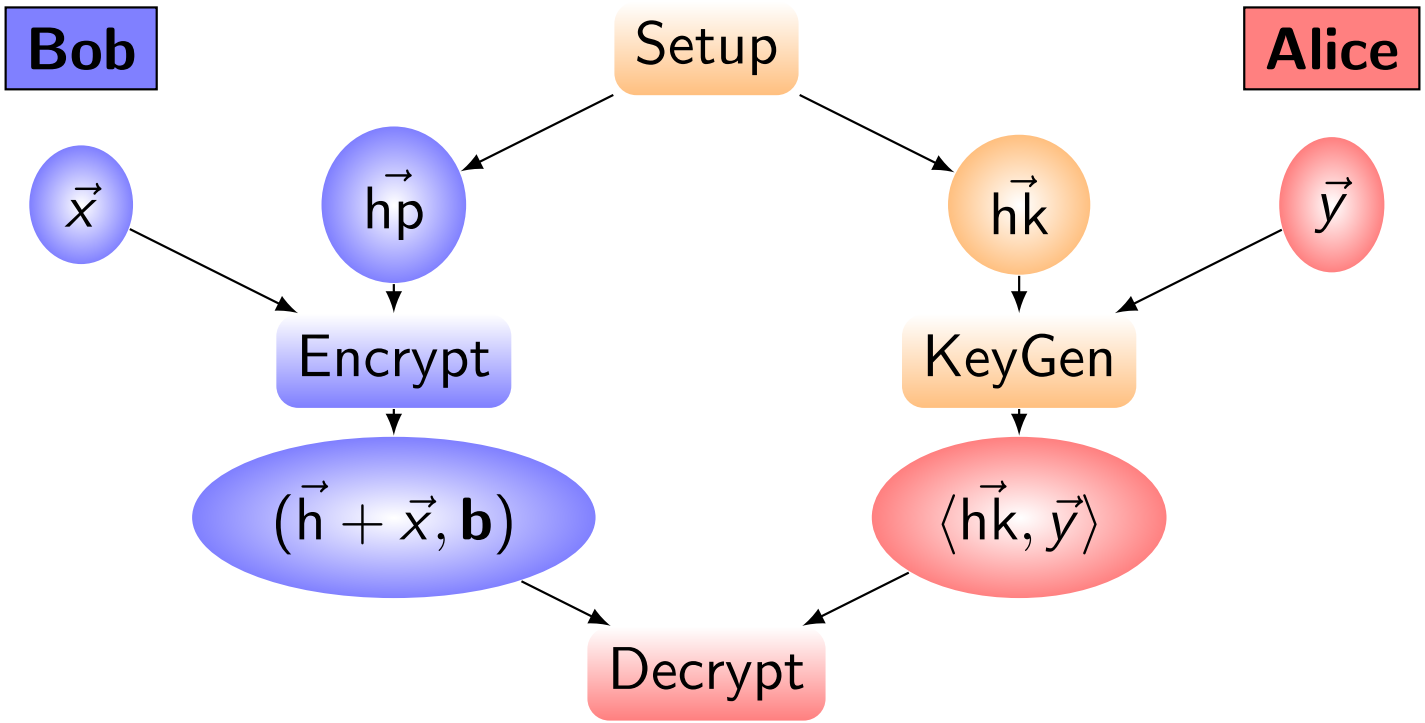
IPFE from PHF



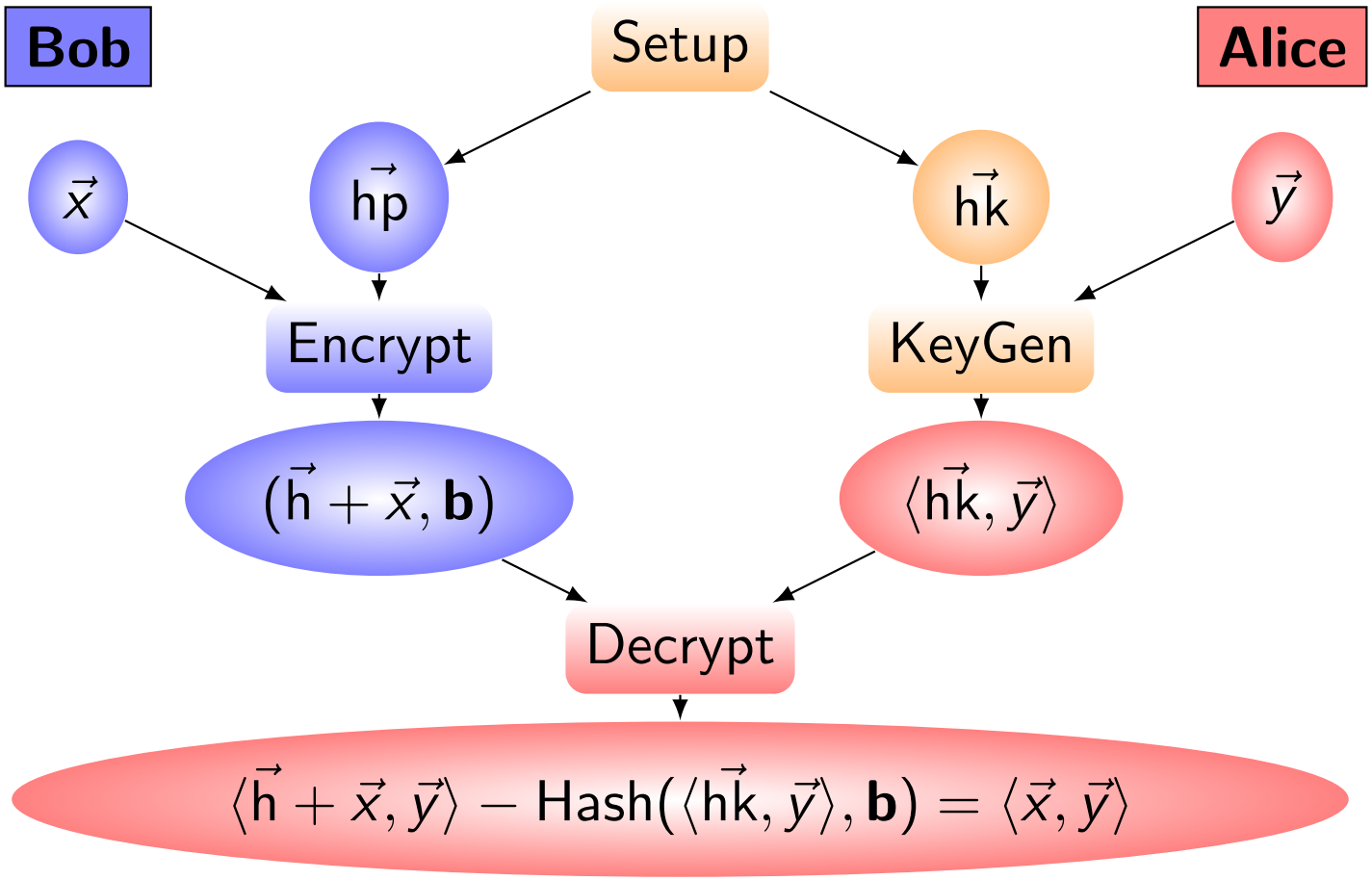
IPFE from PHF



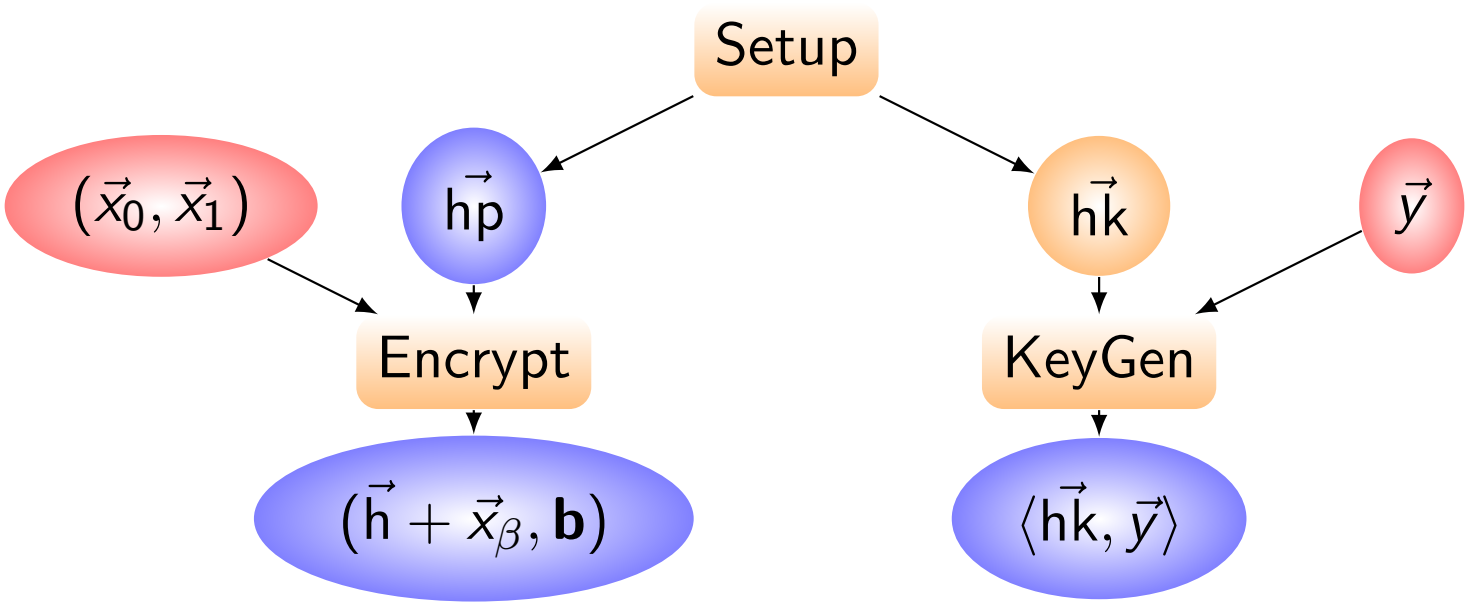
IPFE from PHF



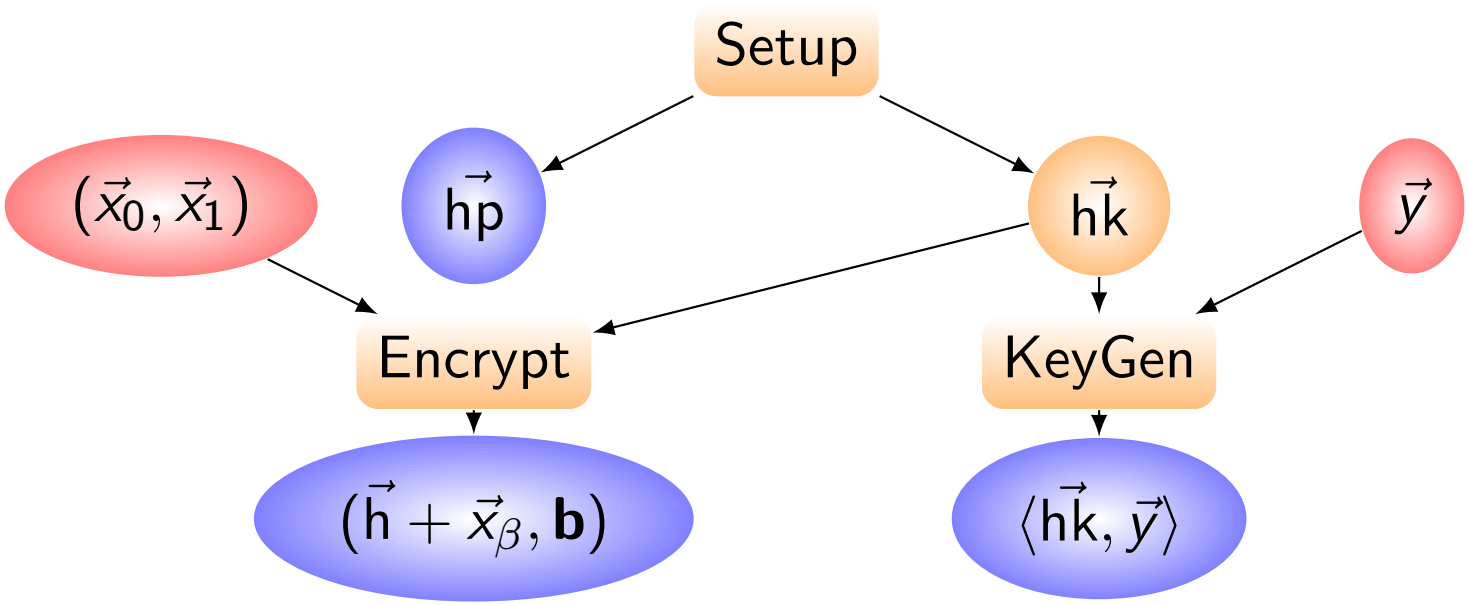
IPFE from PHF



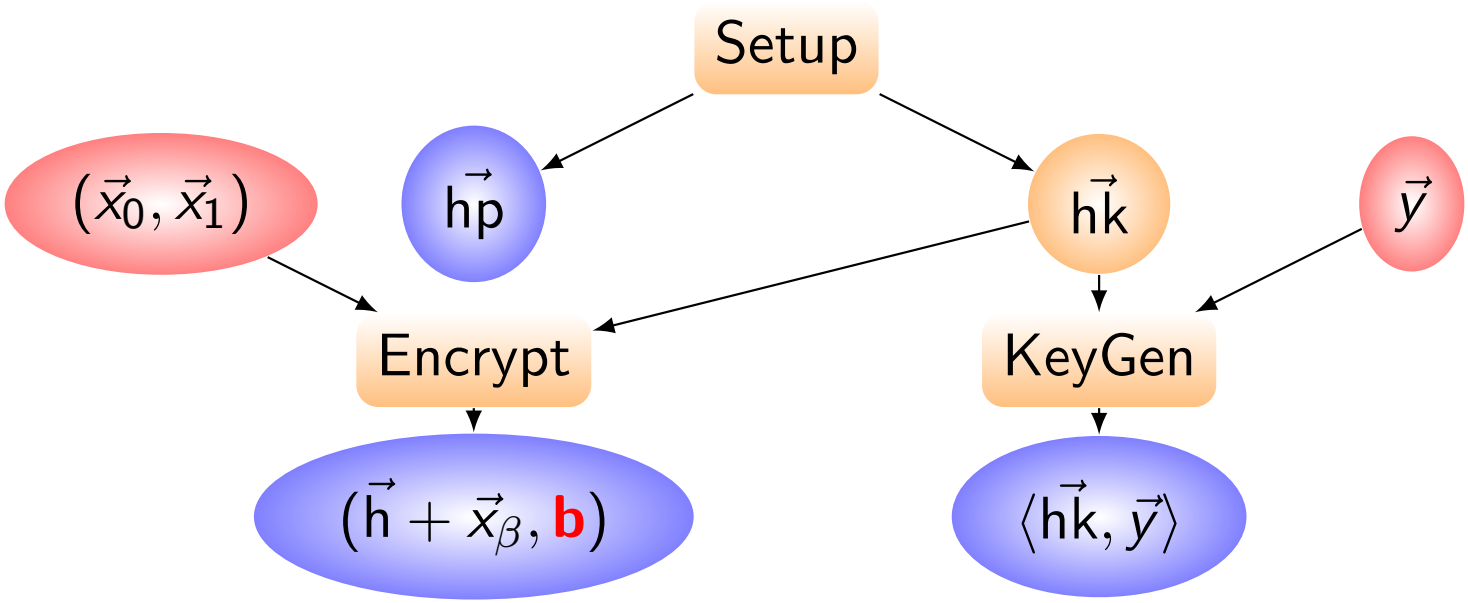
CPA-Security[ALS16]



CPA-Security[ALS16]



CPA-Security[ALS16]



CCA-Security: Issues to fix

issue: Decryption oracle might leak information about $\vec{hk}$

CCA-Security: Issues to fix

issue: Decryption oracle might leak information about $\vec{hk}$

solution: Add 2nd Hash to prove that $\mathbf{b} \in \mathcal{L}$

CCA-Security: Issues to fix

issue: Decryption oracle might leak information about $\vec{hk}$

solution: Add 2nd Hash to prove that $\mathbf{b} \in \mathcal{L}$

why: If $\mathbf{b} \in \mathcal{L}$, $\vec{h}$ can be computed using $\vec{hp}$ so it contains no more information about $\vec{hk}$ than $\vec{hp}$

CCA-Security: Issues to fix

issue: Decryption oracle might leak information about $\vec{hk}$

solution: Add 2nd Hash to prove that $\mathbf{b} \in \mathcal{L}$

why: If $\mathbf{b} \in \mathcal{L}$, $\vec{h}$ can be computed using $\vec{hp}$ so it contains no more information about $\vec{hk}$ than $\vec{hp}$

issue: Scheme is Malleable

CCA-Security: Issues to fix

issue: Decryption oracle might leak information about $\vec{hk}$

solution: Add 2nd Hash to prove that $\mathbf{b} \in \mathcal{L}$

why: If $\mathbf{b} \in \mathcal{L}$, $\vec{h}$ can be computed using $\vec{hp}$ so it contains no more information about $\vec{hk}$ than $\vec{hp}$

issue: Scheme is Malleable

solution: Sign ct using OTS. Use vk as tag for PHF

CCA-Security: Issues to fix

issue: Decryption oracle might leak information about $\vec{hk}$

solution: Add 2nd Hash to prove that $\mathbf{b} \in \mathcal{L}$

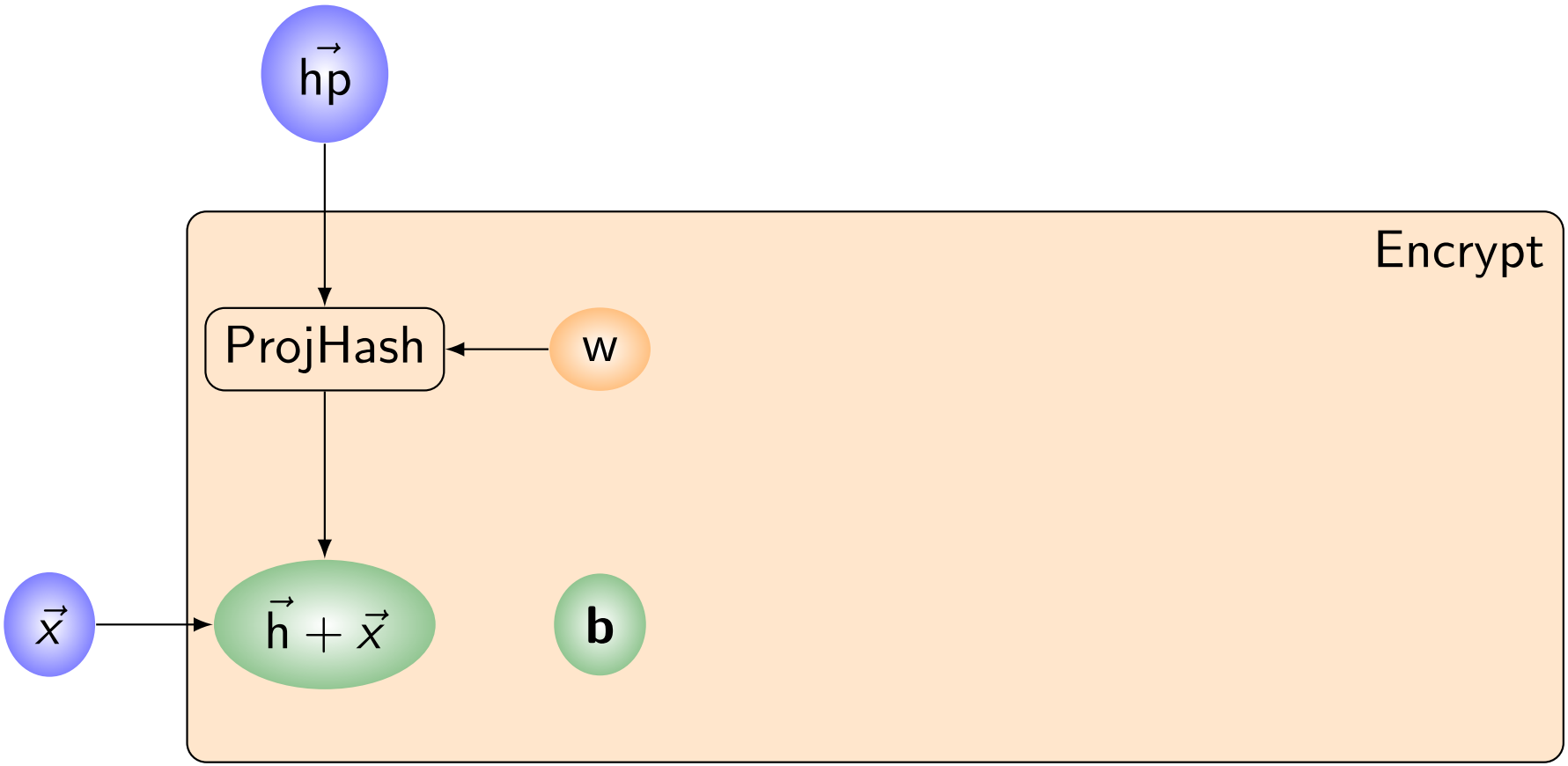
why: If $\mathbf{b} \in \mathcal{L}$, $\vec{h}$ can be computed using $\vec{hp}$ so it contains no more information about $\vec{hk}$ than $\vec{hp}$

issue: Scheme is Malleable

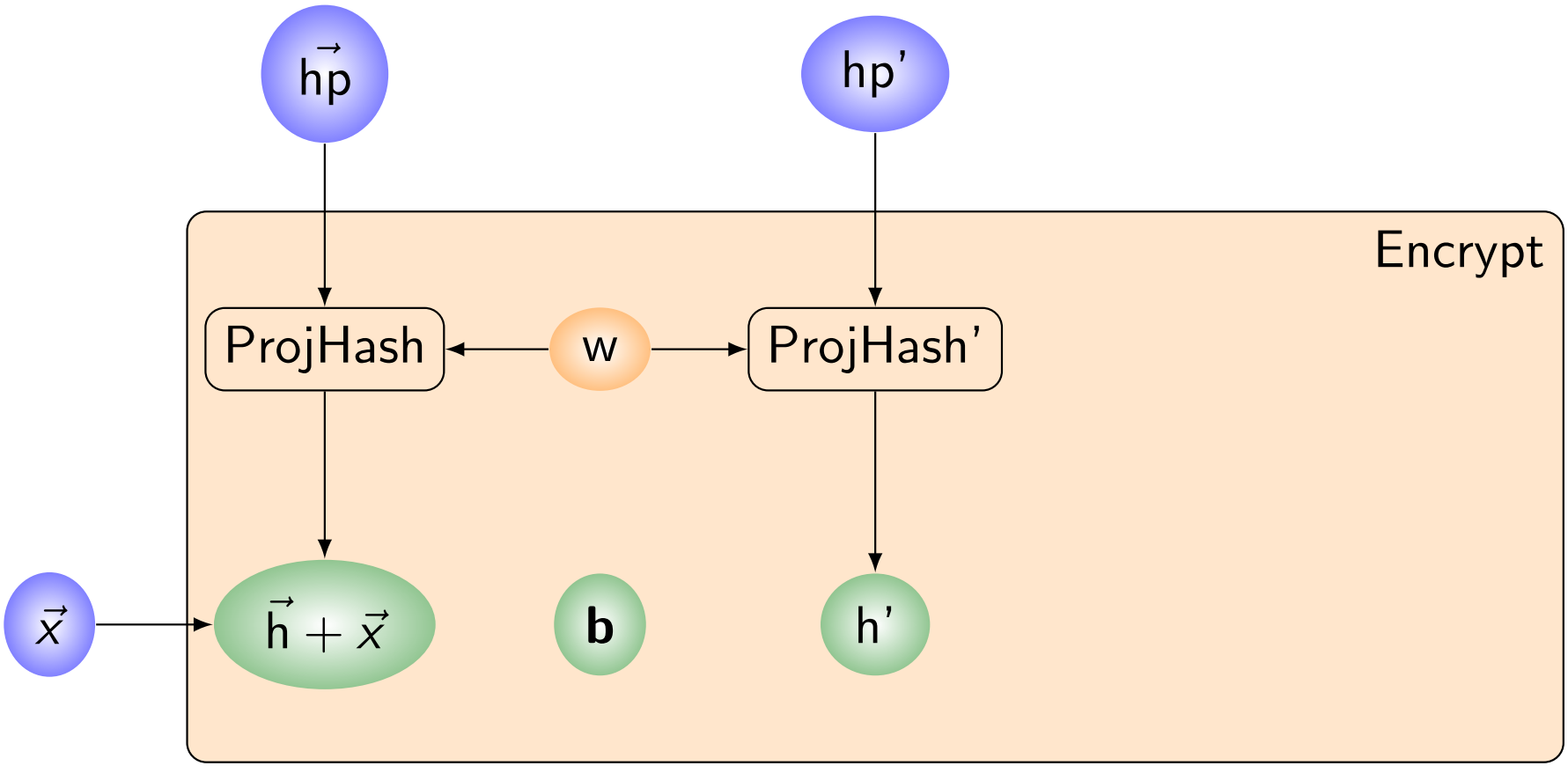
solution: Sign ct using OTS. Use vk as tag for PHF

why: Same tag $\Longrightarrow$ cannot sign
Signature $\Longrightarrow$ no parts from other ciphertexts

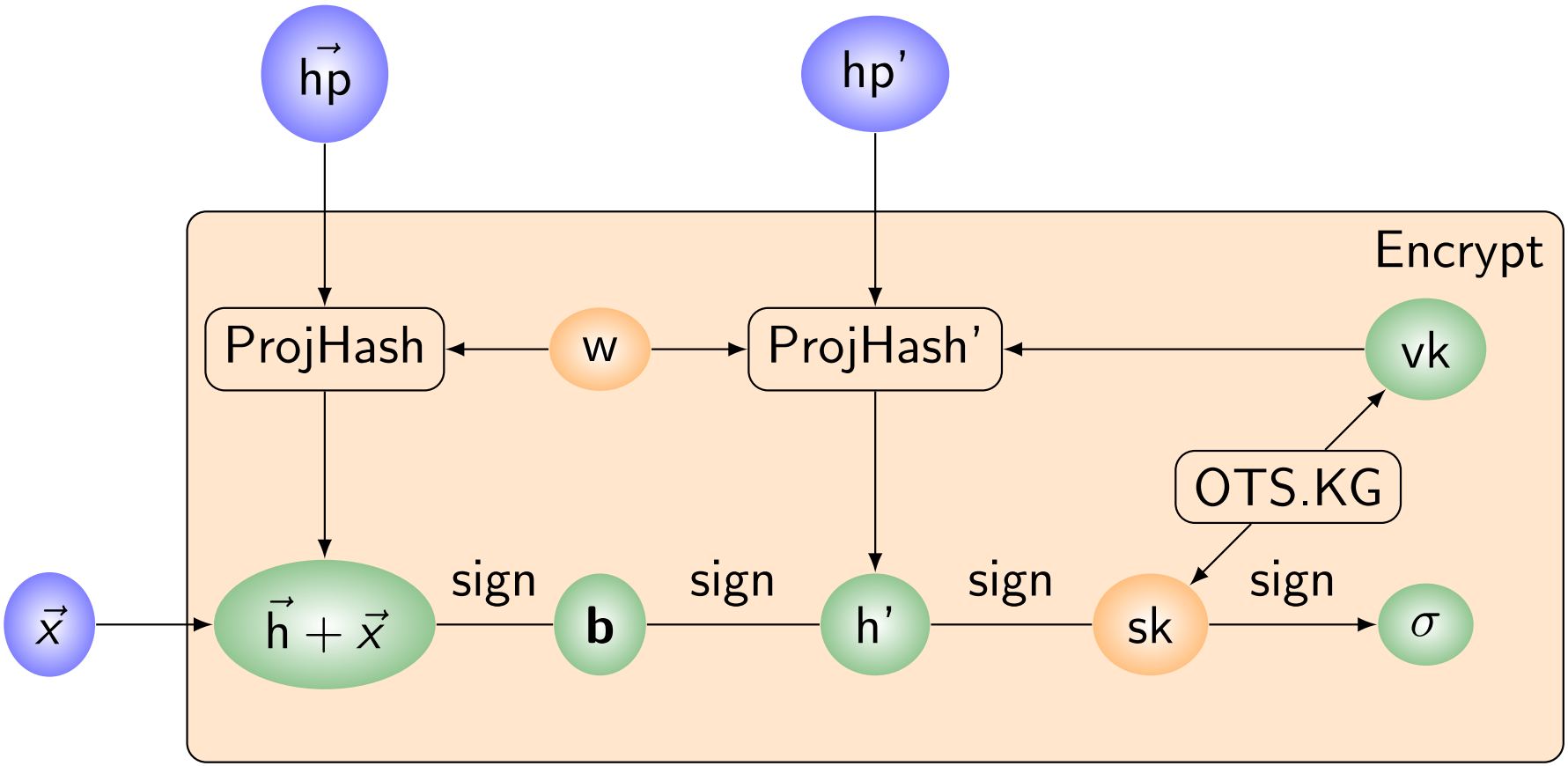
CCA-Security: Proving $\mathbf{b} \in \mathcal{L}$[CS98]



CCA-Security: Proving $\mathbf{b} \in \mathcal{L}$[CS98]



CCA-Security: Removing Malleability



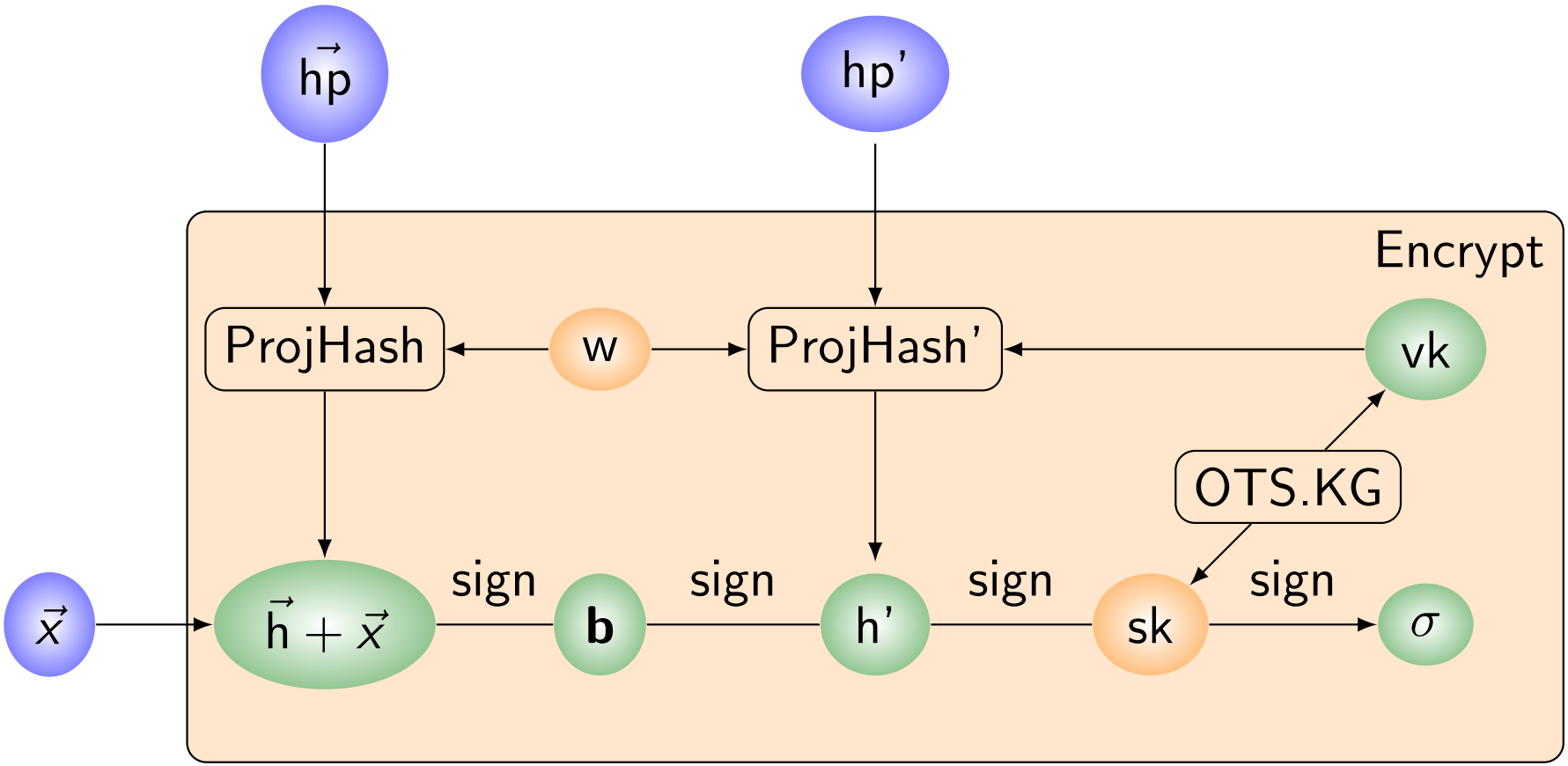
Nope!

This is not enough!

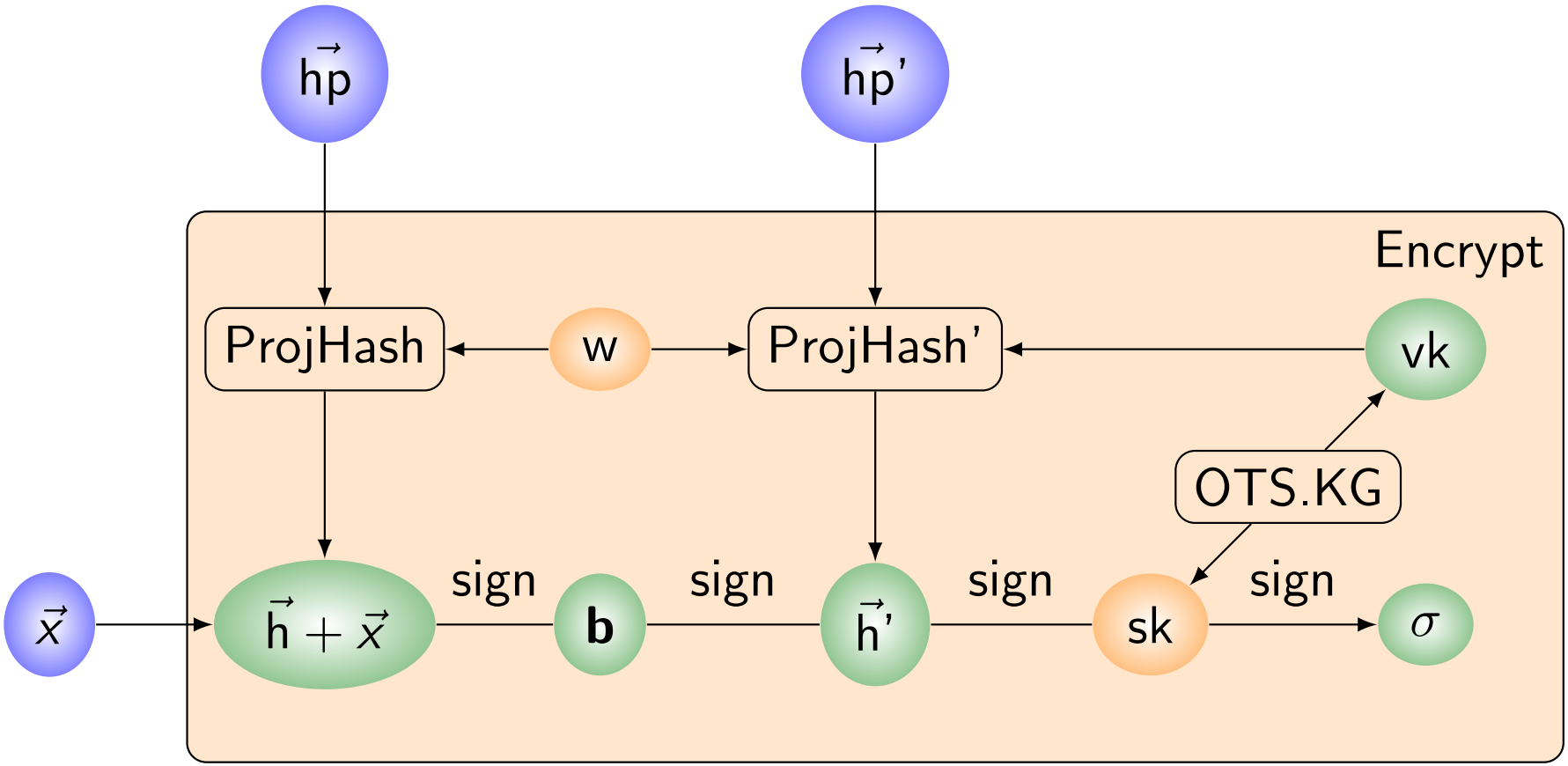
Nope!

This is not enough!
He who can check the proof, can fake it!

CCA-Security



CCA-Security



Why?

Previous construction can be based on DDH and DCR.

Why?

Previous construction can be based on DDH and DCR.
Both assumptions can be broken by quantum computers.

Learning With Errors

For $\mathbf{s} \leftarrow \mathbb{Z}_q^n$,

Learning With Errors

For $\mathbf{s} \leftarrow \mathbb{Z}_q^n$,

$$\mathbf{a}, \mathbf{a}^t\mathbf{s} + e \text{ looks uniform.}$$

Learning With Errors

For $\mathbf{s} \leftarrow \mathbb{Z}_q^n$,

$$\mathbf{a}, \mathbf{a}^t\mathbf{s} + e \text{ looks uniform.}$$

$$\mathbf{a} \leftarrow \mathbb{Z}_q^n;$$

Learning With Errors

For $\mathbf{s} \leftarrow \mathbb{Z}_q^n$,

$$\mathbf{a}, \mathbf{a}^t\mathbf{s} + e \text{ looks uniform.}$$

$\mathbf{a} \leftarrow \mathbb{Z}_q^n$;

$e \in \mathbb{Z}$ small.

Regev's Encryption Scheme

Bob **Alice**

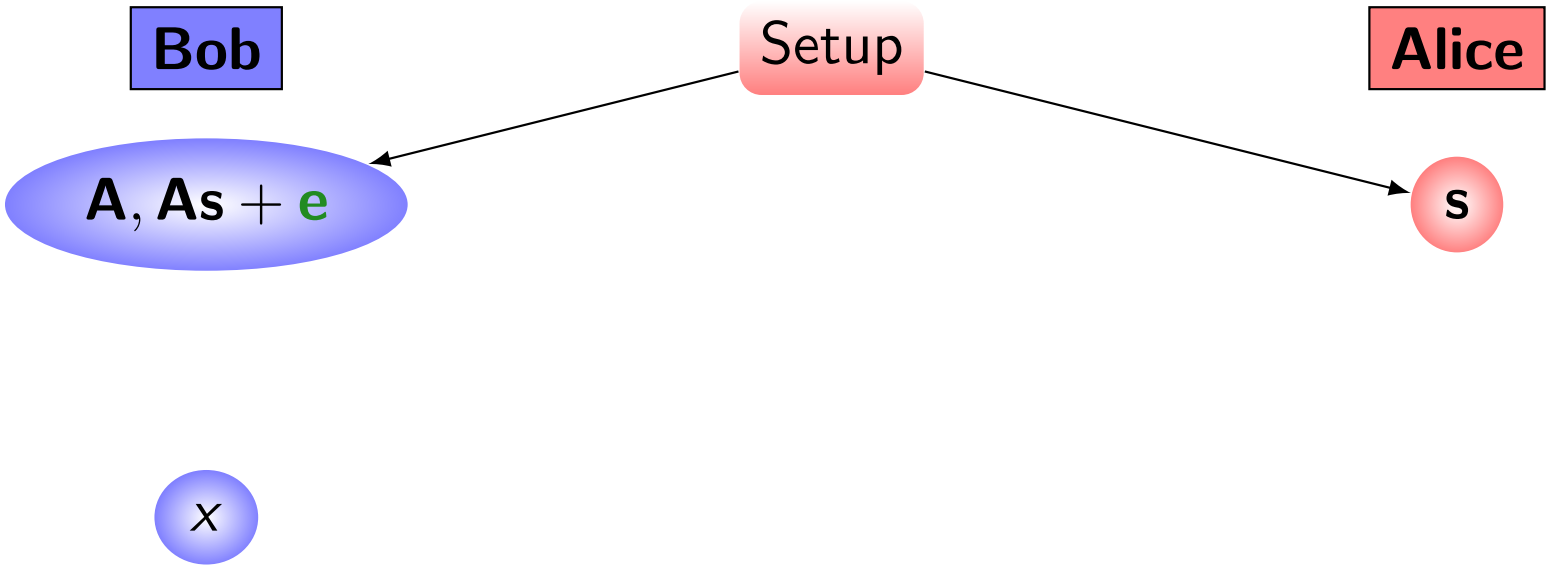
x

Regev's Encryption Scheme

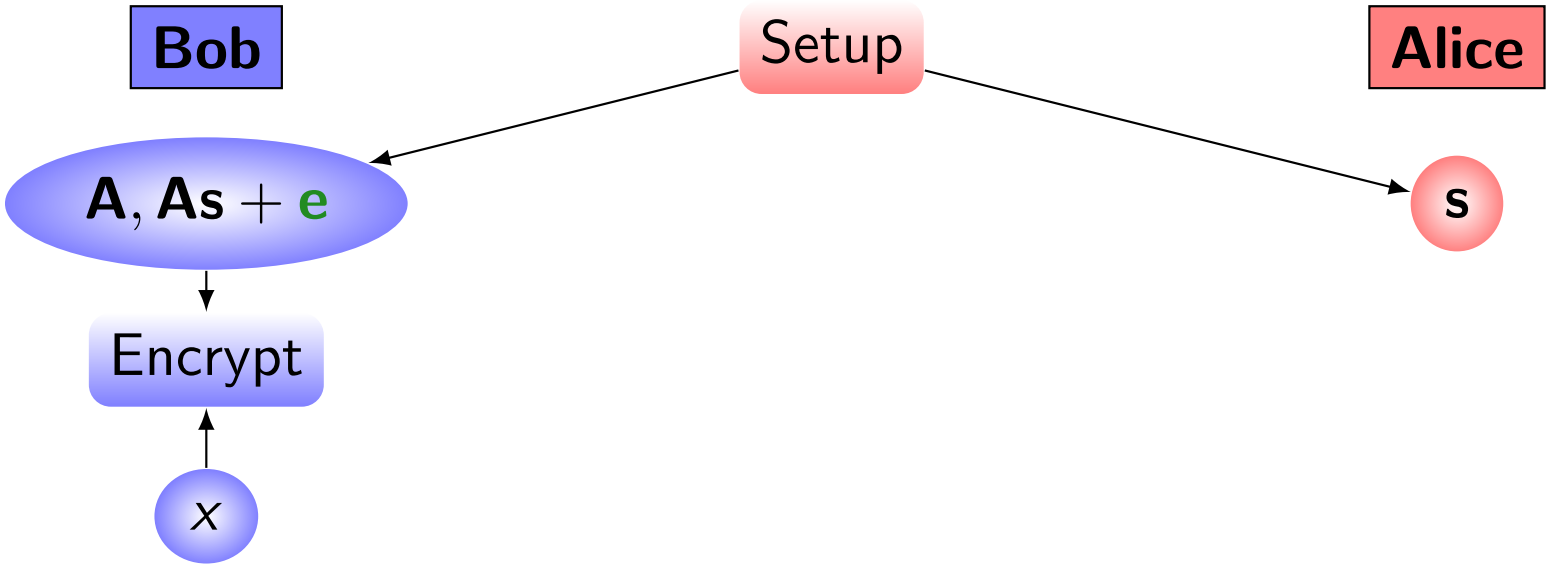
Bob Setup **Alice**

x

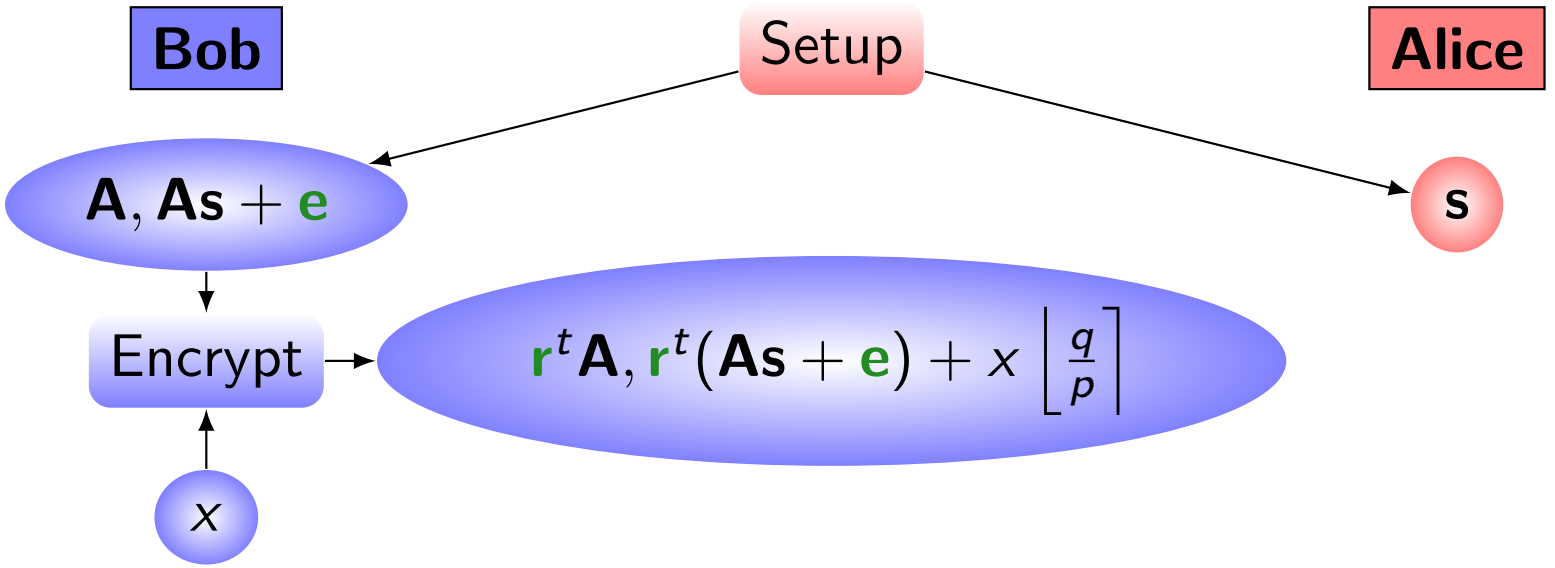
Regev's Encryption Scheme



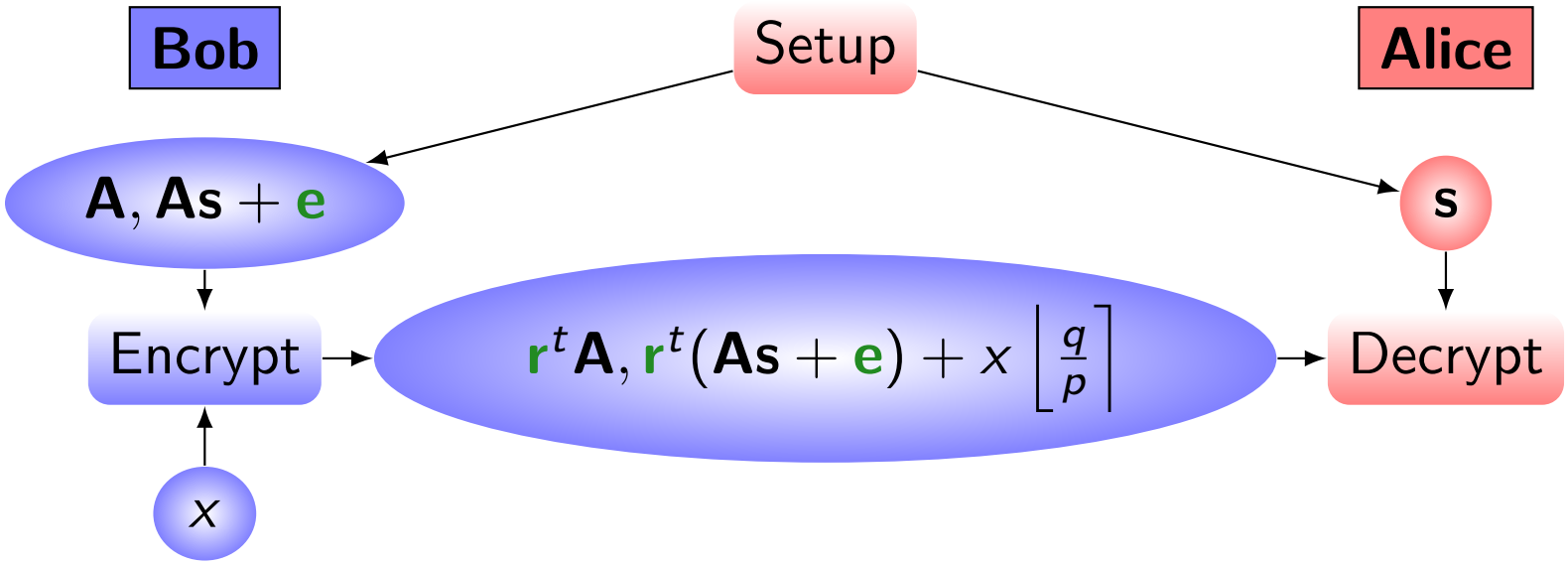
Regev's Encryption Scheme



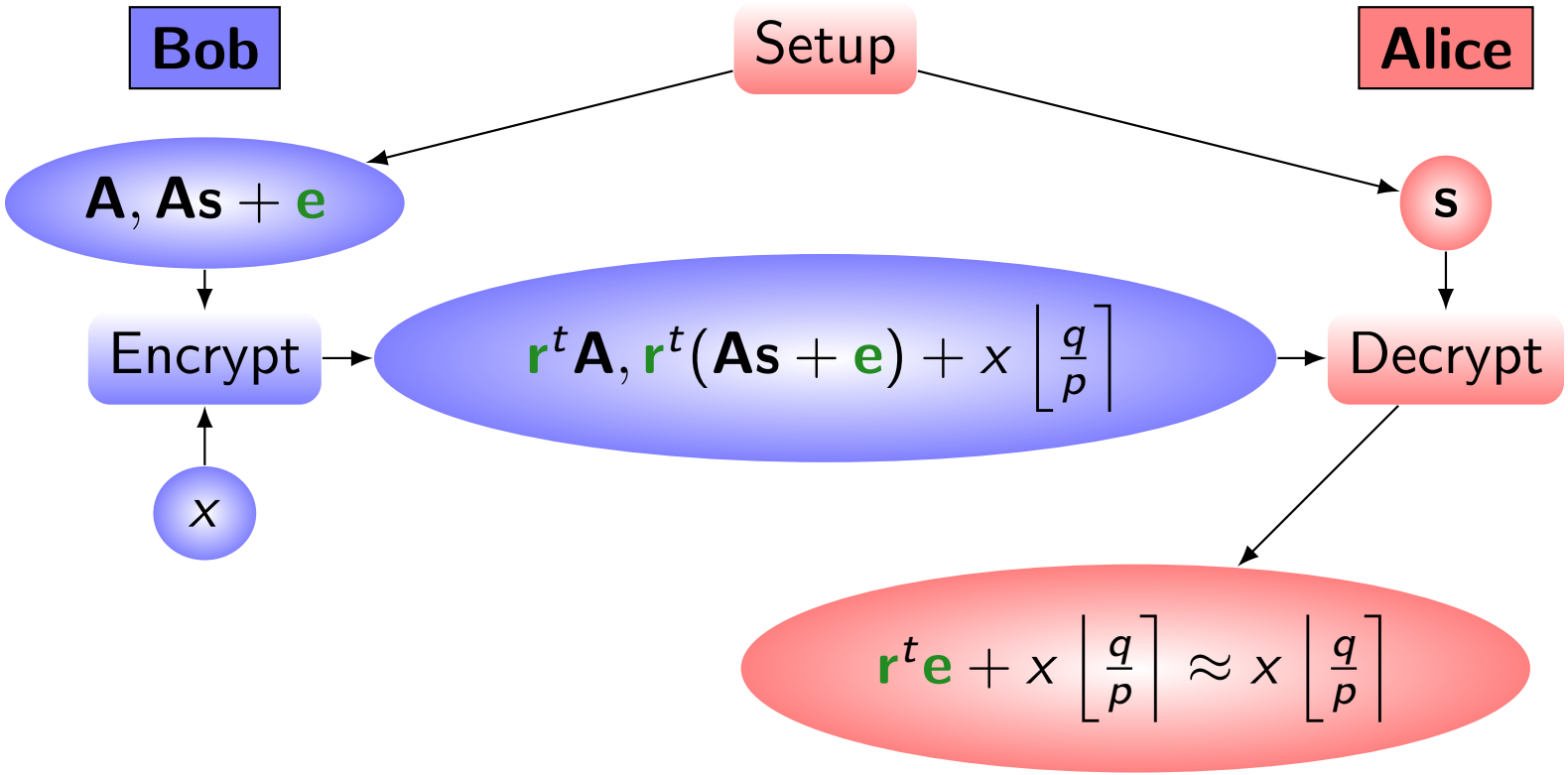
Regev's Encryption Scheme



Regev's Encryption Scheme



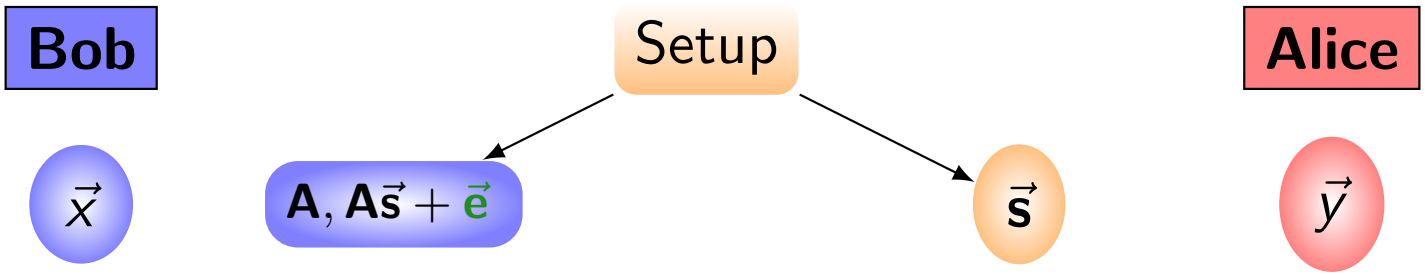
Regev's Encryption Scheme



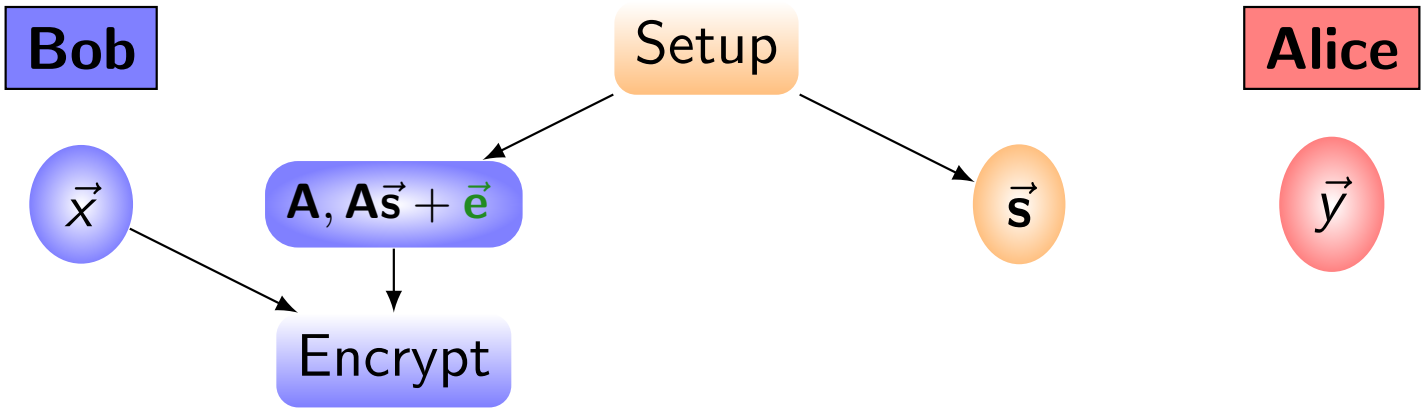
Selectively Secure Scheme

Selectively Secure Scheme

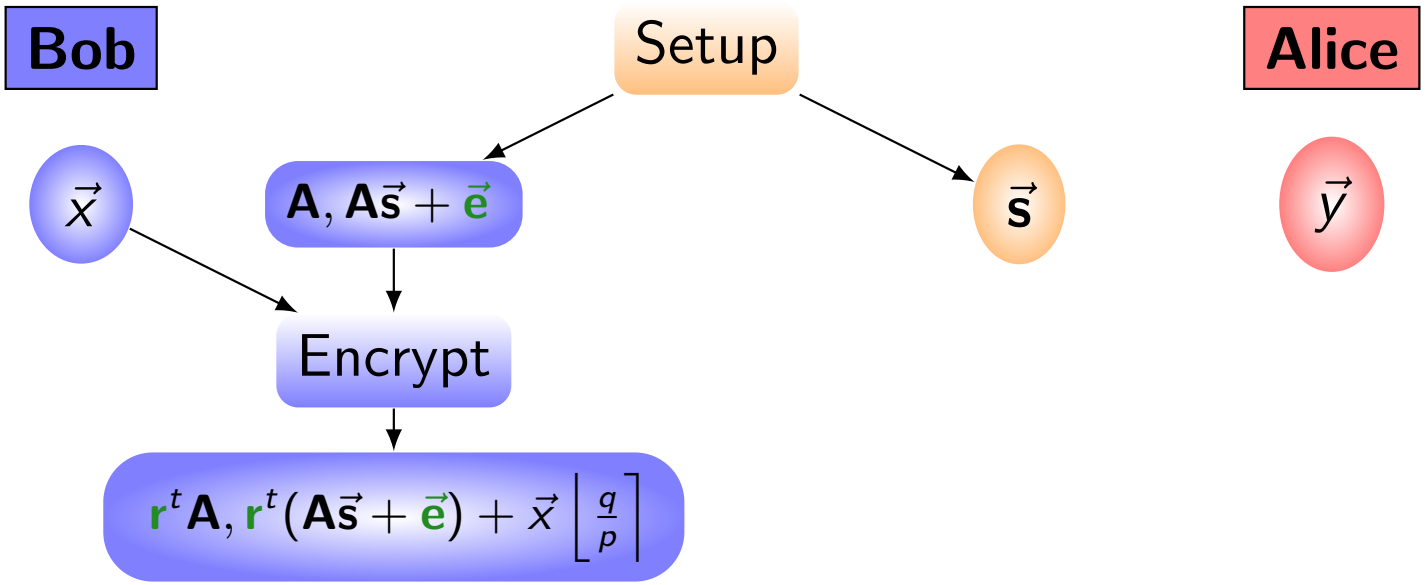
Selectively Secure Scheme



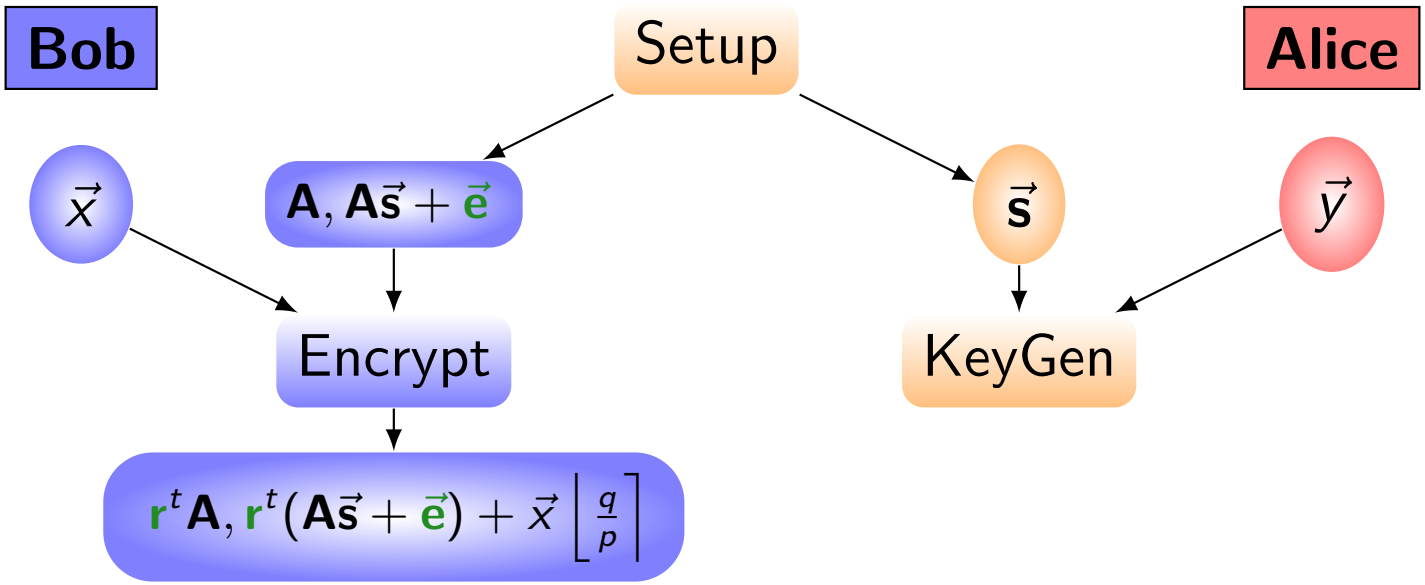
Selectively Secure Scheme



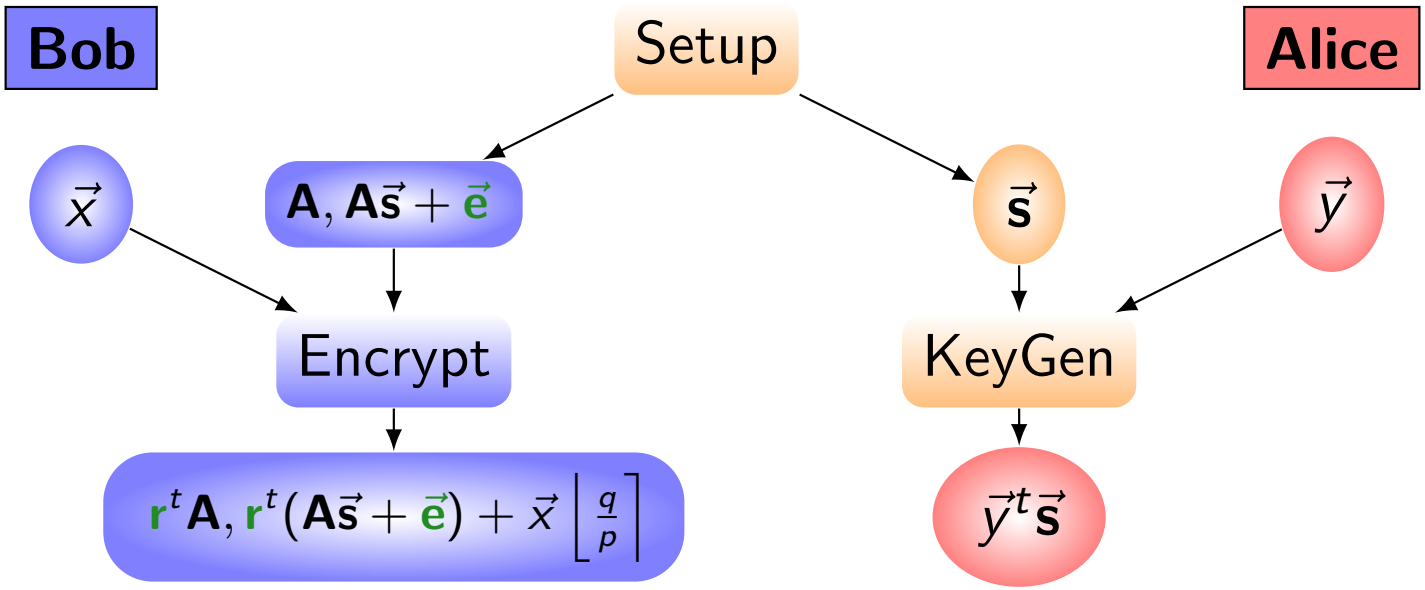
Selectively Secure Scheme



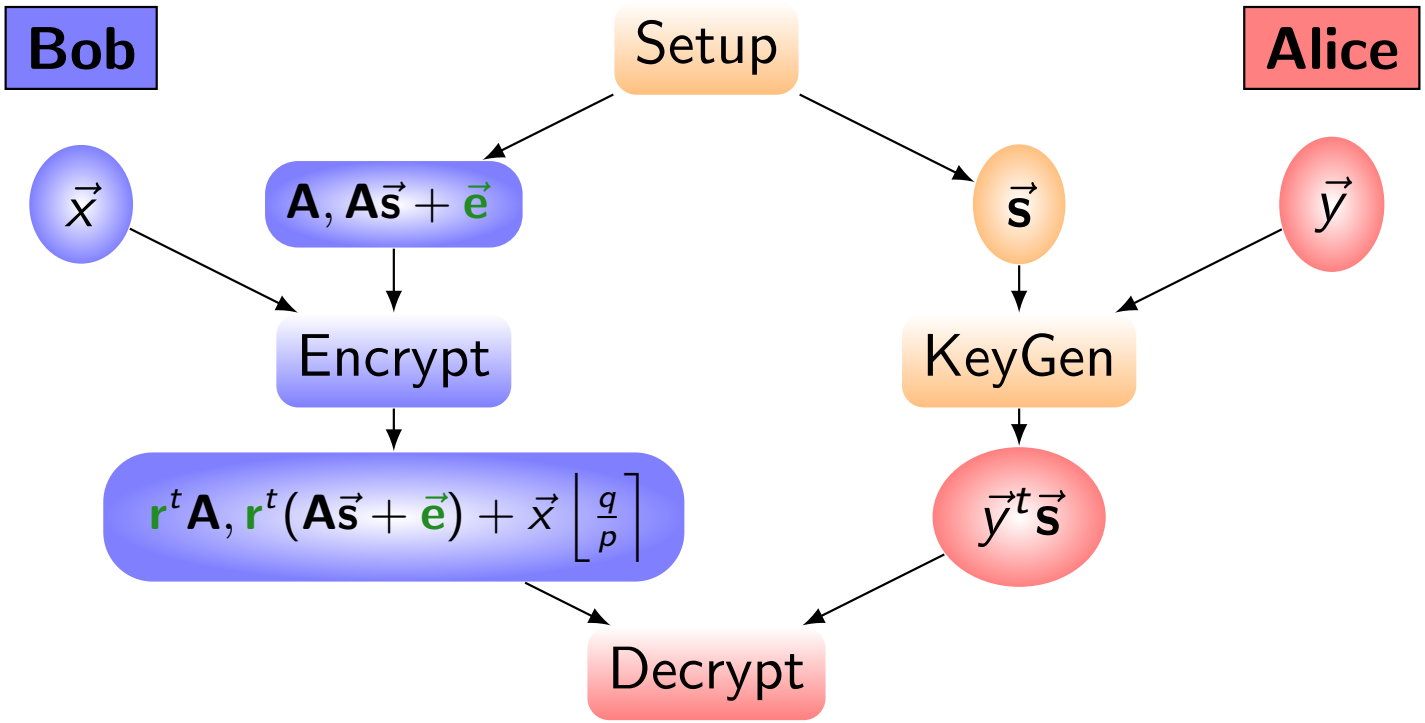
Selectively Secure Scheme



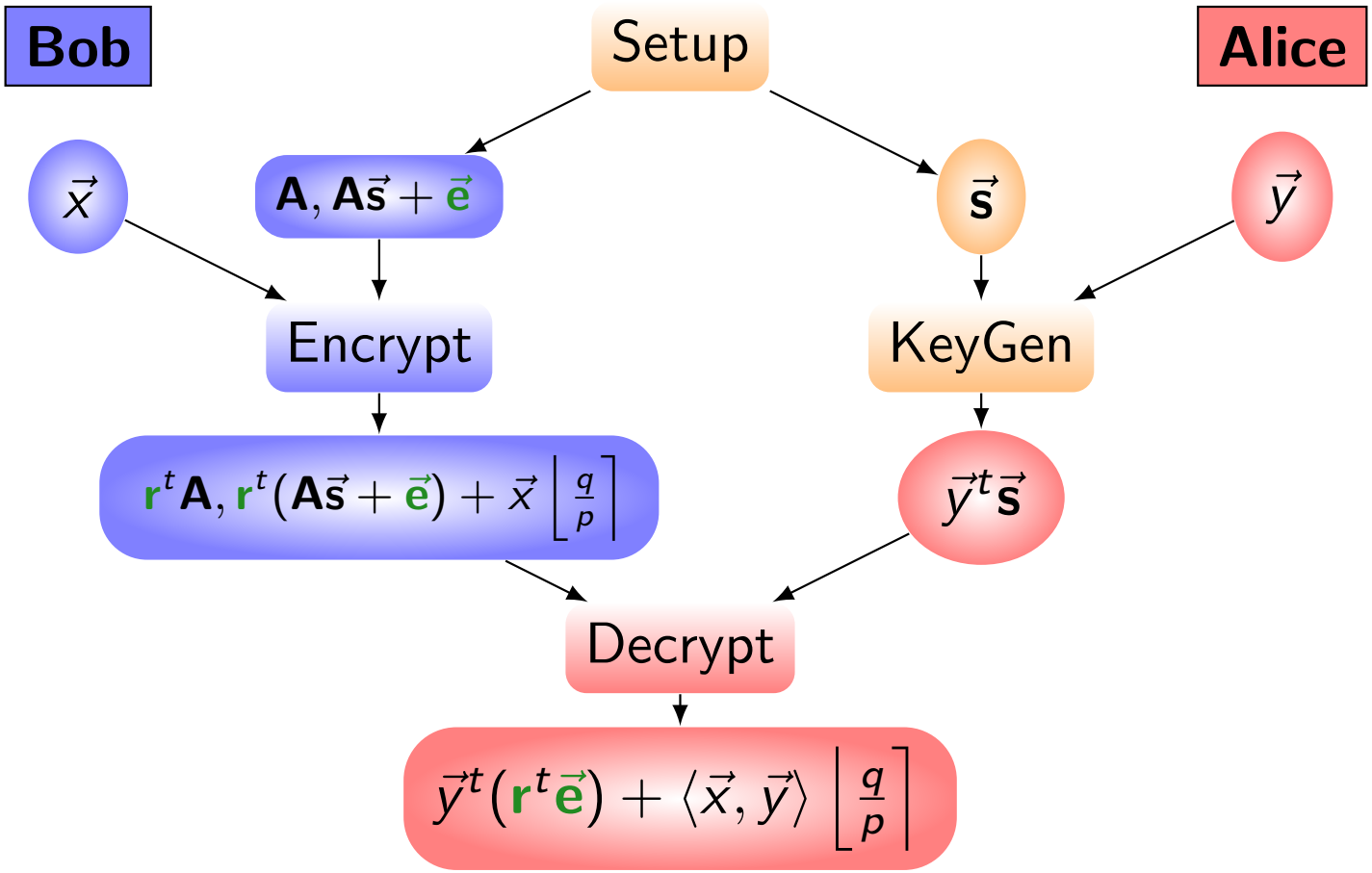
Selectively Secure Scheme



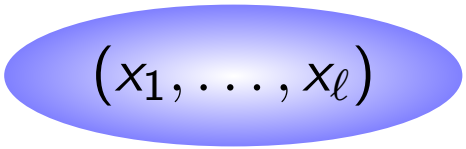
Selectively Secure Scheme

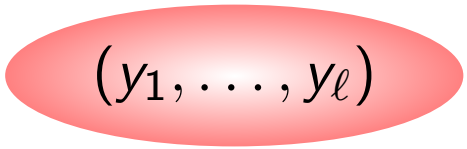


Selectively Secure Scheme

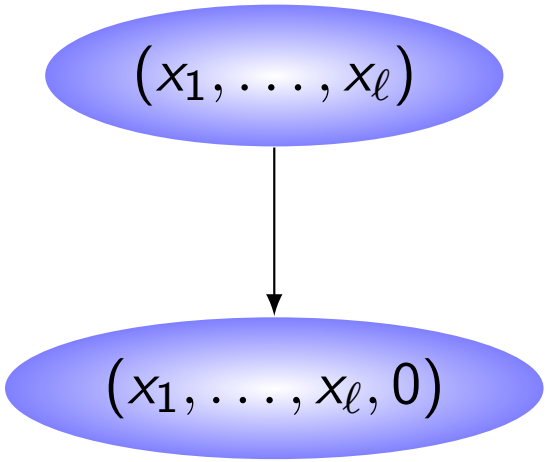


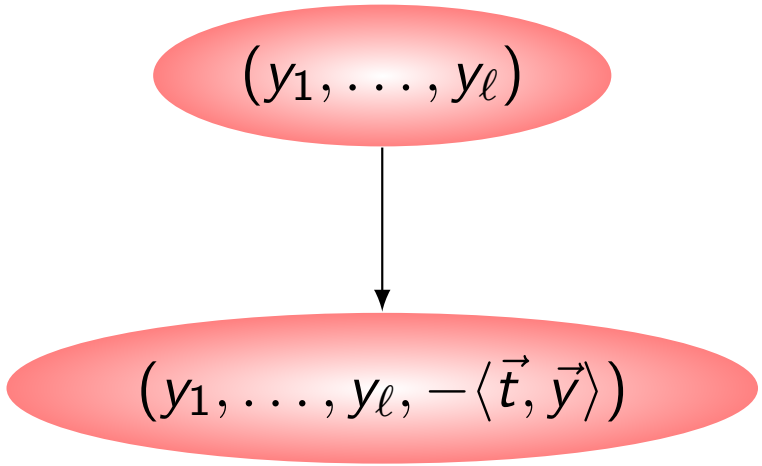
Adaptive Security



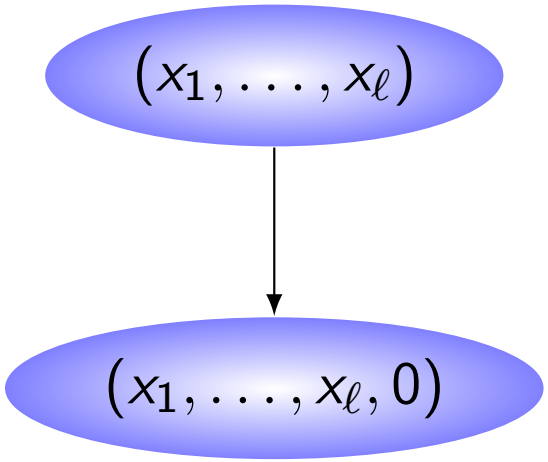


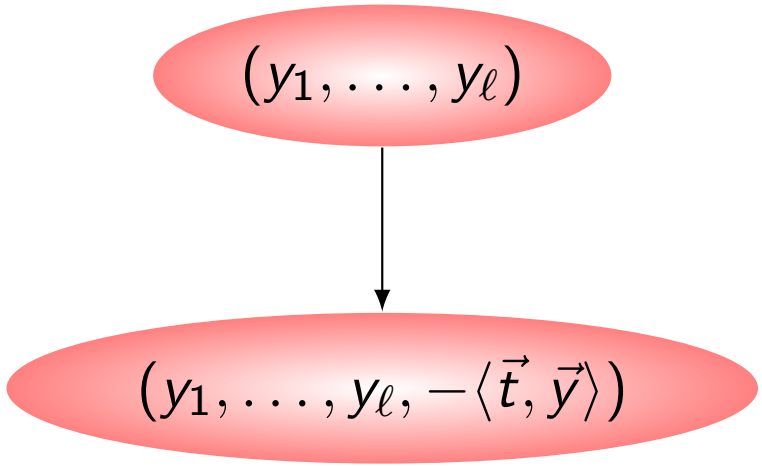
Adaptive Security





Adaptive Security





$$\mathsf{Enc}(\vec{0}, 0) \approx_c \mathsf{Enc}(\vec{t}, 1)$$

The Proof

$(\vec{x}_0, 0)$

$(\vec{x}_1, 0)$

The Proof

$(\vec{x}_0, 0)$

$\approx_c$

$(\vec{x}_0 + \vec{t}, 1)$

$(\vec{x}_1, 0)$

The Proof

$(\vec{x}_0, 0)$ $\approx_c$ $(\vec{x}_1, 0)$

$(\vec{x}_0 + \vec{t}, 1)$ $\approx_c$ $(\vec{x}_1 + \vec{t}, 1)$

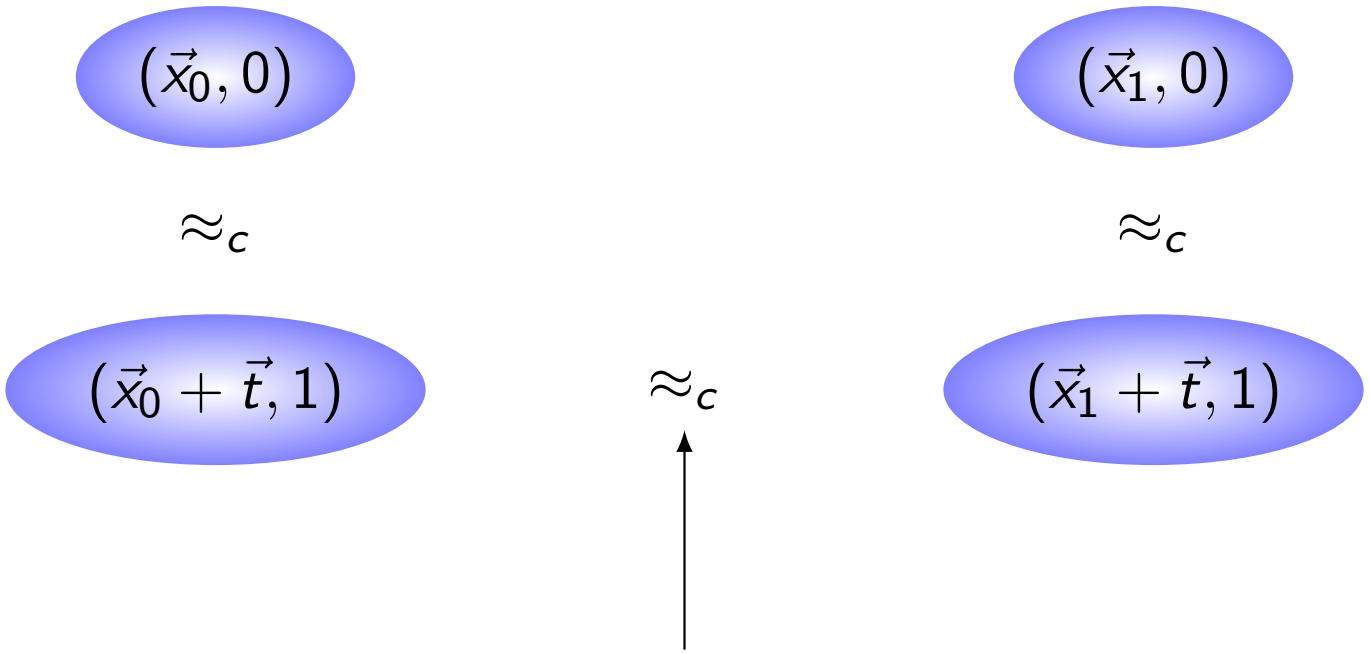
The Proof

$(\vec{x}_0, 0)$ $\approx_c$ $(\vec{x}_1, 0)$

$(\vec{x}_0 + \vec{t}, 1)$ $\approx_c$ $(\vec{x}_1 + \vec{t}, 1)$

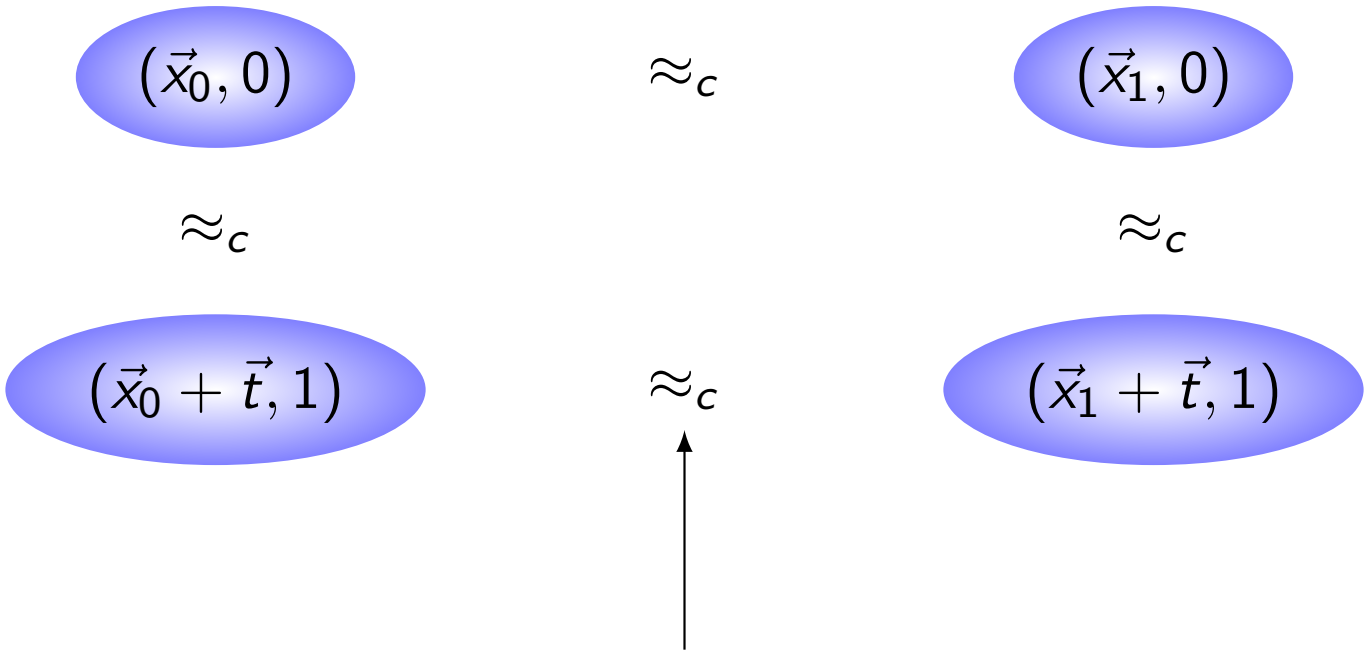
flooding, requires $\vec{t}$ bigger than $\vec{x}$ by a super-polynomial factor

The Proof



flooding, requires $\vec{t}$ bigger than $\vec{x}$ by a super-polynomial factor

The Proof



flooding, requires $\vec{t}$ bigger than $\vec{x}$ by a super-polynomial factor

Table of Comparison

Assumption	CCA-Security	Efficient Decryption	Modular	Post-Quantum
DDH	✓		✓	
DCR	✓	✓	✓ *	
LWE		✓	✓ *	✓

*Stateful KeyGen [ALS16]