CCA-Secure Inner-Product Functional Encryption from Projective Hash Function

Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa

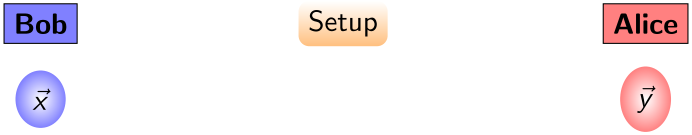
IBM Research, Yorktown Heights, NY, USA
École normale supérieure, CNRS, INRIA, PSL, Paris, France
Institute of Computer Science, University of Tartu, Estonia

PKC 2017 — Amsterdam, Netherlands
Thursday, March 30

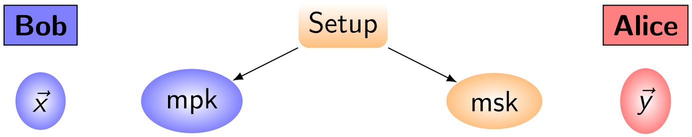
Inner-Product Functional Encryption[ABDP15]

Bob
Alice
$\vec{x}$
$\vec{y}$

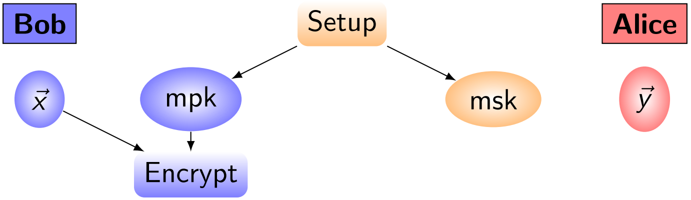
Inner-Product Functional Encryption[ABDP15]



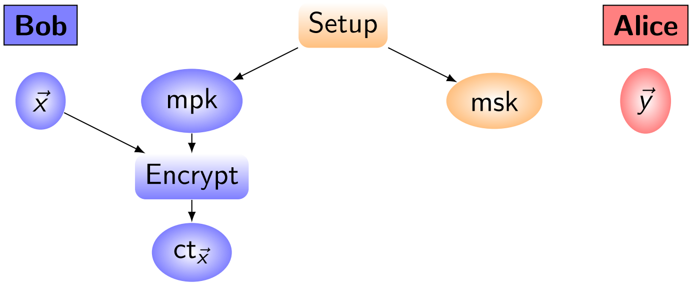
Inner-Product Functional Encryption[ABDP15]



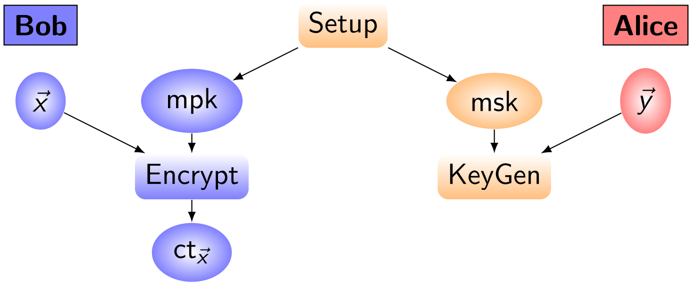
Inner-Product Functional Encryption[ABDP15]



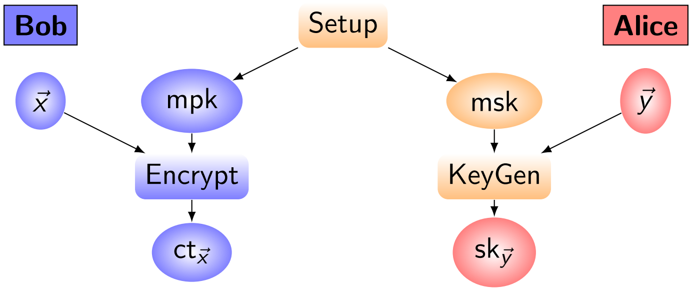
Inner-Product Functional Encryption[ABDP15]



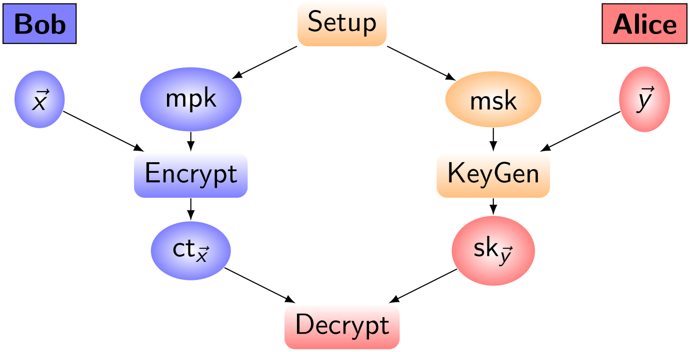
Inner-Product Functional Encryption[ABDP15]


$$\begin{tikzcd}
\text{Bob} & & \text{Setup} \arrow[dl] \arrow[dr] & & \text{Alice} \\
\vec{x} \arrow[dr] & \text{mpk} \arrow[d] & & \text{msk} \arrow[d] & \vec{y} \arrow[dl] \\
& \text{Encrypt} \arrow[d] & & \text{KeyGen} & \\
& \text{ct}_{\vec{x}} & & &
\end{tikzcd}$$

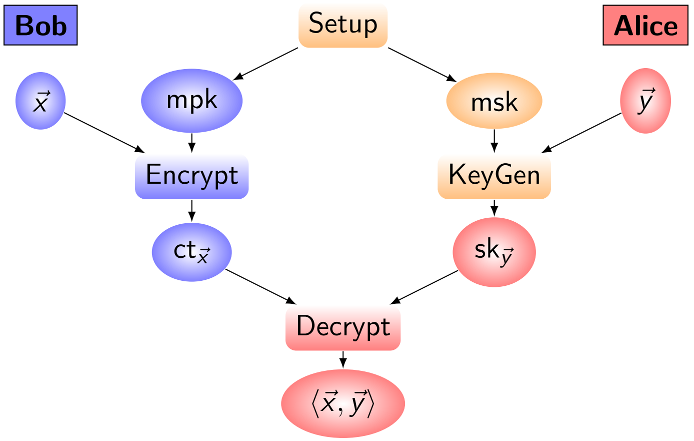
Inner-Product Functional Encryption[ABDP15]



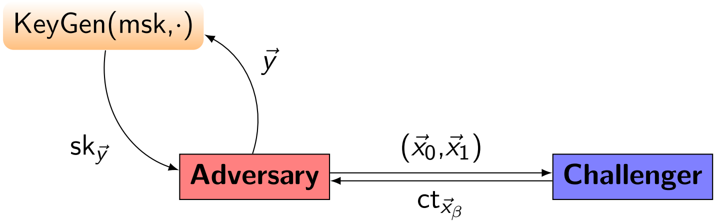
Inner-Product Functional Encryption[ABDP15]



Inner-Product Functional Encryption[ABDP15]

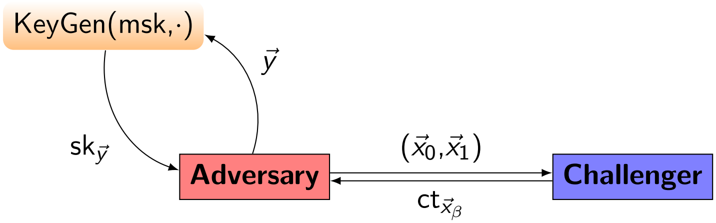


IND-CPA security



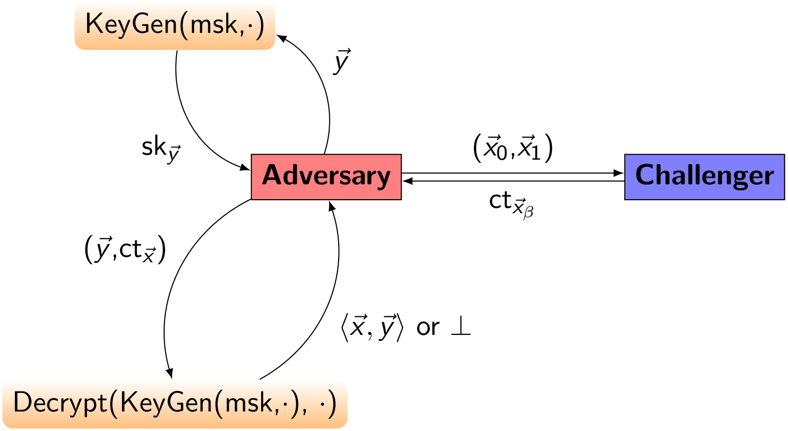
Adversary wins if $b' = b \land \forall \vec{y}, \langle \vec{x}_0, \vec{y} \rangle = \langle \vec{x}_1, \vec{y} \rangle$

IND-CPA security



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

IND-CCA security


KeyGen(msk,·)
$\vec{y}$
$\mathsf{sk}_{\vec{y}}$
Adversary
$(\vec{x}_0, \vec{x}_1)$
$\mathsf{ct}_{\vec{x}_\beta}$
Challenger
$(\vec{y}, \mathsf{ct}_{\vec{x}})$
$\langle \vec{x}, \vec{y} \rangle$ or $\perp$
Decrypt(KeyGen(msk,·), ·)

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

NP language

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

NP language

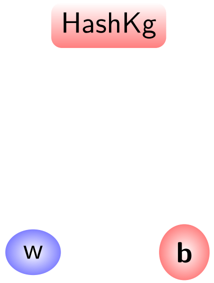
Let $\mathcal{L} = \{\mathbf{b} / \exists w \text{ s.t. } \mathcal{R}(w, \mathbf{b}) = 1\}$ be an NP language.

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

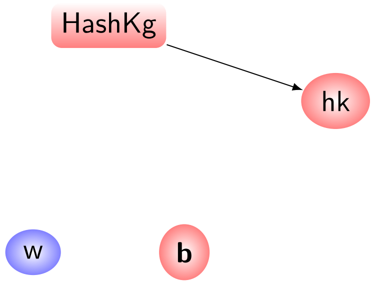
$$\mathcal{L} = \{(g^r, h^r)\}_{r \in \mathbb{Z}_p}$$

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

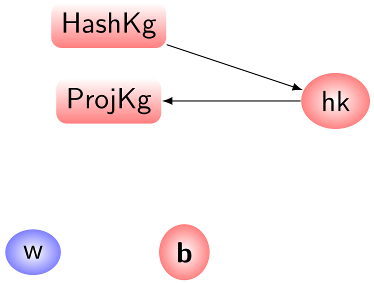
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



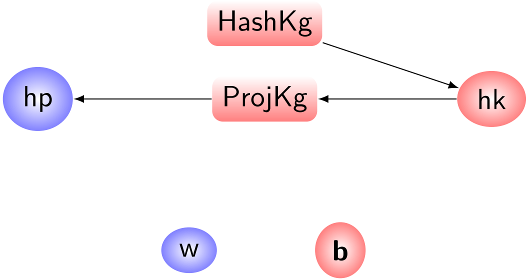
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



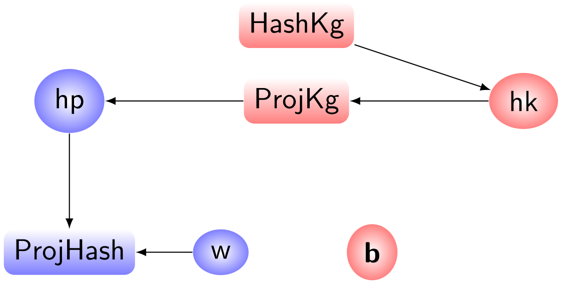
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



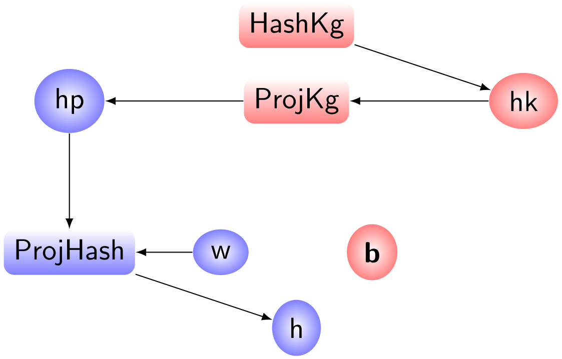
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



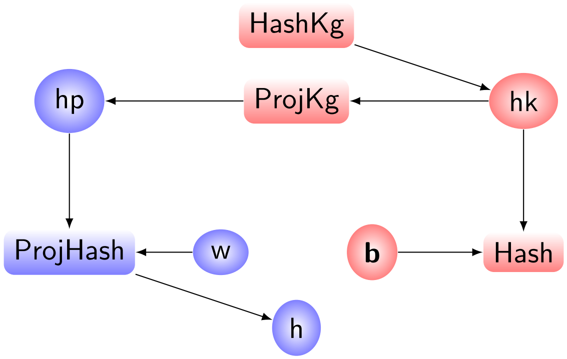
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



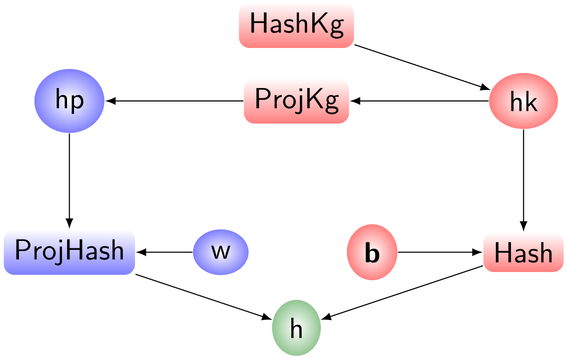
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



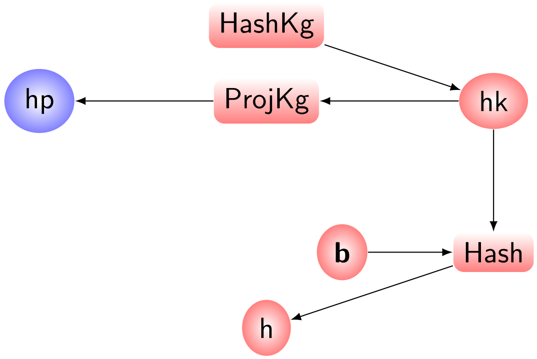
Projective Hash Function: $(w, \mathbf{b}) \in \mathcal{L}$



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



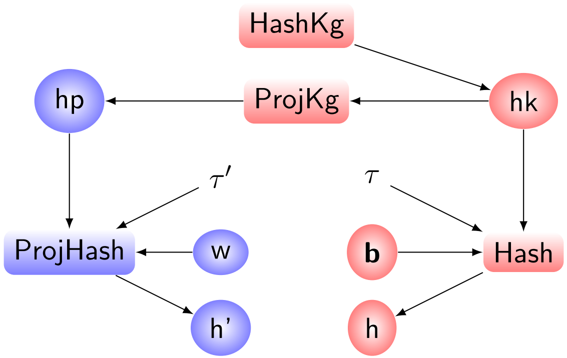
Projective Hash Function: $\mathbf{b} \notin \mathcal{L}$



Tag-Based PHF

$$\begin{tikzcd}
 & & \mathsf{HashKg} \arrow[drr] & & \\
\mathsf{hp} \arrow[dd] & & \mathsf{ProjKg} \arrow[ll] & & \mathsf{hk} \arrow[ll] \arrow[dd] \\
 & & \tau \arrow[dll] \arrow[drr] & & \\
\mathsf{ProjHash} \arrow[drr] & \mathsf{w} \arrow[l] & & \mathbf{b} \arrow[r] & \mathsf{Hash} \arrow[dll] \\
 & & \mathsf{h} & &
\end{tikzcd}$$

Tag-Based PHF

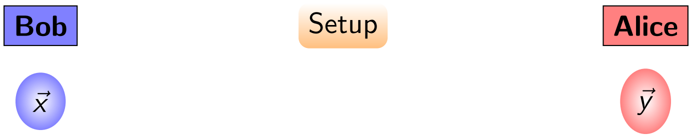


Hash is an Additive Homomorphism

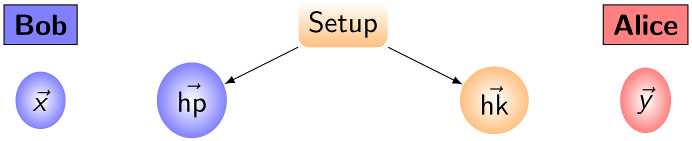
$$\begin{tikzcd}
 & \vec{\mathsf{hk}} \arrow[dl, "{\langle \cdot, \vec{y} \rangle}"'] \arrow[dr, "{\mathsf{Hash}(\cdot, \mathbf{b})}"] & \\
\langle \vec{\mathsf{hk}}, \vec{y} \rangle \arrow[dr, "{\mathsf{Hash}(\cdot, \mathbf{b})}"'] & & \vec{\mathsf{h}} \arrow[dl, "{\langle \cdot, \vec{y} \rangle}"] \\
 & \langle \vec{\mathsf{h}}, \vec{y} \rangle &
\end{tikzcd}$$

IPFE from PHF

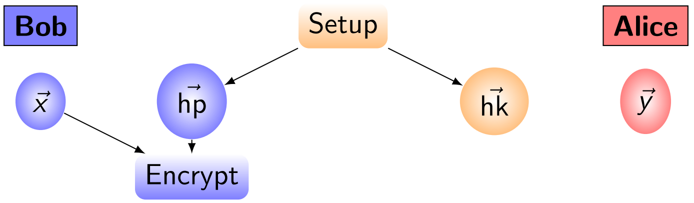
IPFE from PHF



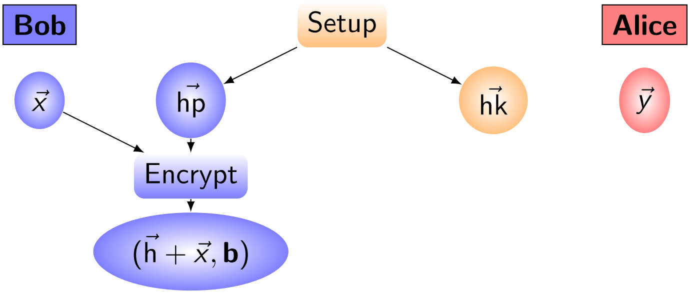
IPFE from PHF



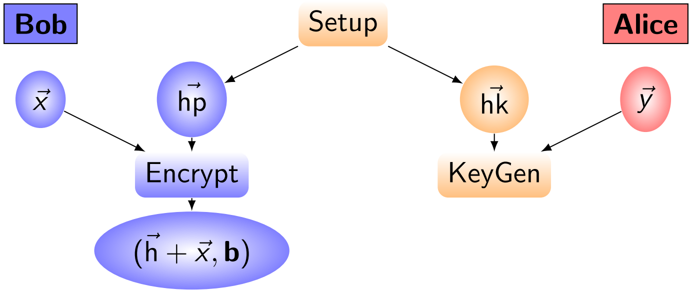
IPFE from PHF



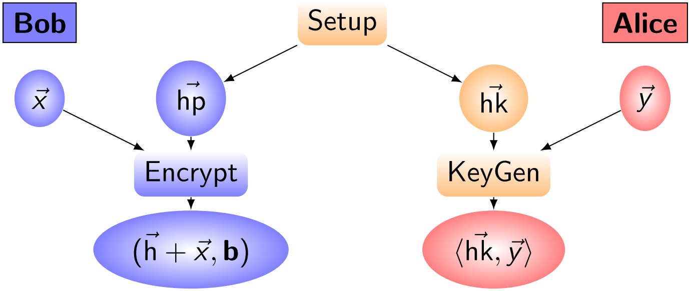
IPFE from PHF



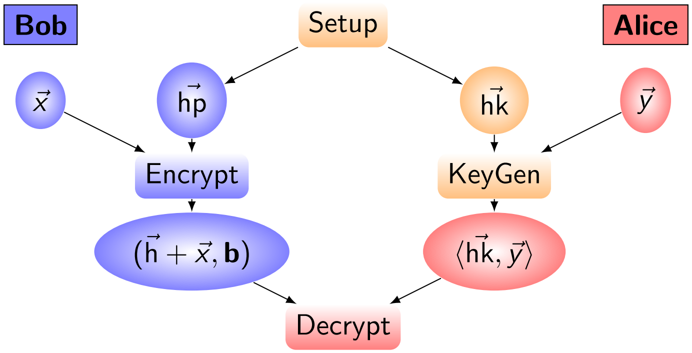
IPFE from PHF



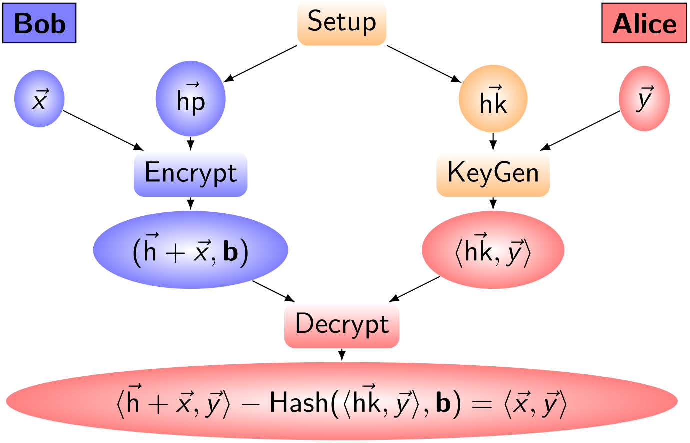
IPFE from PHF


Bob
Setup
Alice
$\vec{x}$
$\vec{\mathsf{hp}}$
$\vec{\mathsf{hk}}$
$\vec{y}$
Encrypt
KeyGen
$(\vec{\mathsf{h}} + \vec{x}, \mathbf{b})$
$\langle \vec{\mathsf{hk}}, \vec{y} \rangle$

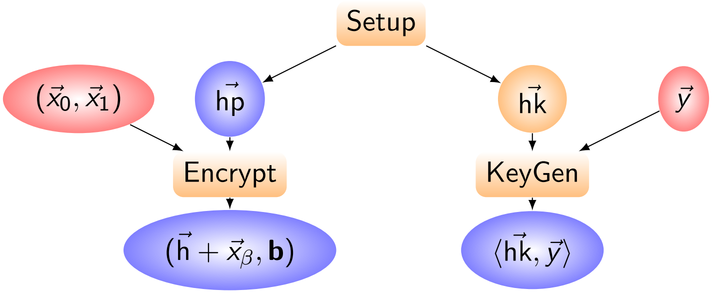
IPFE from PHF



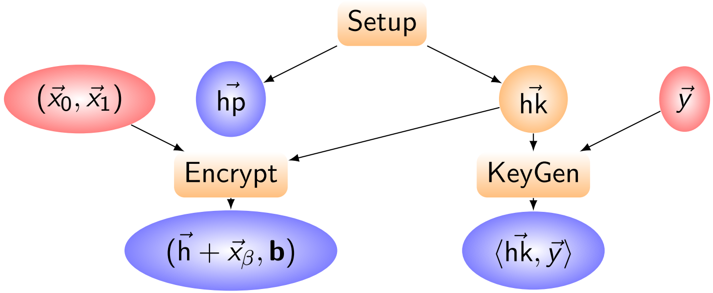
IPFE from PHF


Bob
Setup
Alice
$\vec{x}$
$\vec{\mathsf{hp}}$
$\vec{\mathsf{hk}}$
$\vec{y}$
Encrypt
KeyGen
$(\vec{\mathsf{h}} + \vec{x}, \mathbf{b})$
$\langle \vec{\mathsf{hk}}, \vec{y} \rangle$
Decrypt
$\langle \vec{\mathsf{h}} + \vec{x}, \vec{y} \rangle - \mathsf{Hash}(\langle \vec{\mathsf{hk}}, \vec{y} \rangle, \mathbf{b}) = \langle \vec{x}, \vec{y} \rangle$

CPA-Security[ALS16]

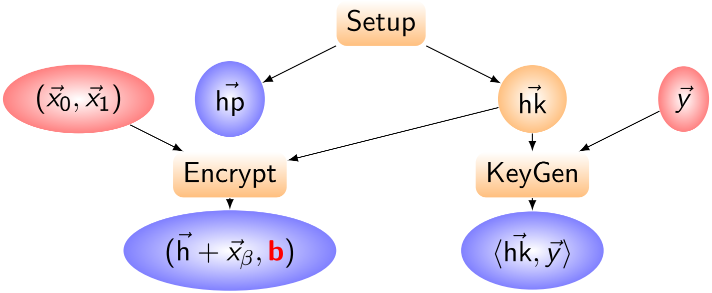

Setup
$(\vec{x}_0, \vec{x}_1)$
$\vec{\text{hp}}$
$\vec{\text{hk}}$
$\vec{y}$
Encrypt
KeyGen
$(\vec{\text{h}} + \vec{x}_\beta, \mathbf{b})$
$\langle \vec{\text{hk}}, \vec{y} \rangle$

CPA-Security[ALS16]



$$\begin{tikzcd}
 & & \mathsf{Setup} \arrow[dl] \arrow[dr] & & \\
(\vec{x}_0, \vec{x}_1) \arrow[dr] & \vec{\mathsf{hp}} & & \vec{\mathsf{hk}} \arrow[dll] \arrow[d] & \vec{y} \arrow[dl] \\
 & \mathsf{Encrypt} \arrow[d] & & \mathsf{KeyGen} \arrow[d] & \\
 & (\vec{\mathsf{h}} + \vec{x}_\beta, \mathbf{b}) & & \langle \vec{\mathsf{hk}}, \vec{y} \rangle &
\end{tikzcd}$$

CPA-Security[ALS16]


Setup
$\vec{\mathsf{hp}}$
$\vec{\mathsf{hk}}$
$(\vec{x}_0, \vec{x}_1)$
$\vec{y}$
Encrypt
KeyGen
$(\vec{\mathsf{h}} + \vec{x}_\beta, \mathbf{b})$
$\langle \vec{\mathsf{hk}}, \vec{y} \rangle$

CCA-Security: Issues to fix

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

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 $\vec{\mathrm{hk}}$

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

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

CCA-Security: Issues to fix

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

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

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

issue: Scheme is Malleable

CCA-Security: Issues to fix

issue:	Decryption oracle might leak information about $\vec{\mathsf{hk}}$
solution:	Add 2nd Hash to prove that $\mathbf{b} \in \mathcal{L}$
why:	If $\mathbf{b} \in \mathcal{L}$, $\vec{\mathsf{h}}$ can be computed using $\vec{\mathsf{hp}}$ so it contains no more information about $\vec{\mathsf{hk}}$ than $\vec{\mathsf{hp}}$

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 $\vec{\mathsf{hk}}$

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

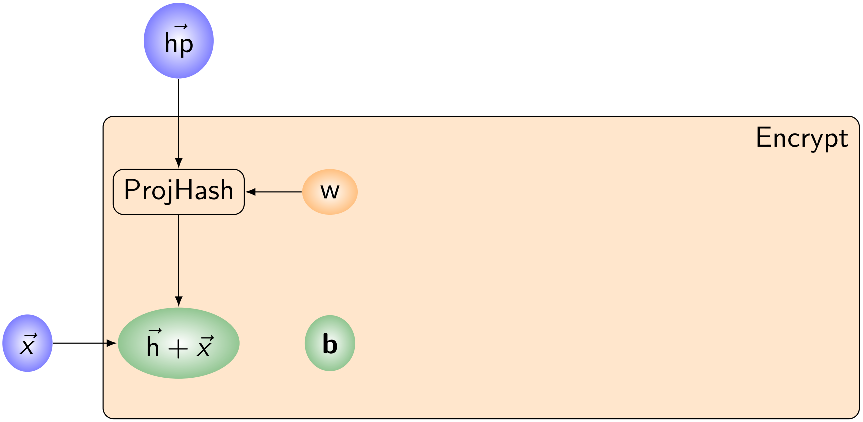
why: If $\mathbf{b} \in \mathcal{L}$, $\vec{\mathsf{h}}$ can be computed using $\vec{\mathsf{hp}}$ so it contains no more information about $\vec{\mathsf{hk}}$ than $\vec{\mathsf{hp}}$

issue: Scheme is Malleable

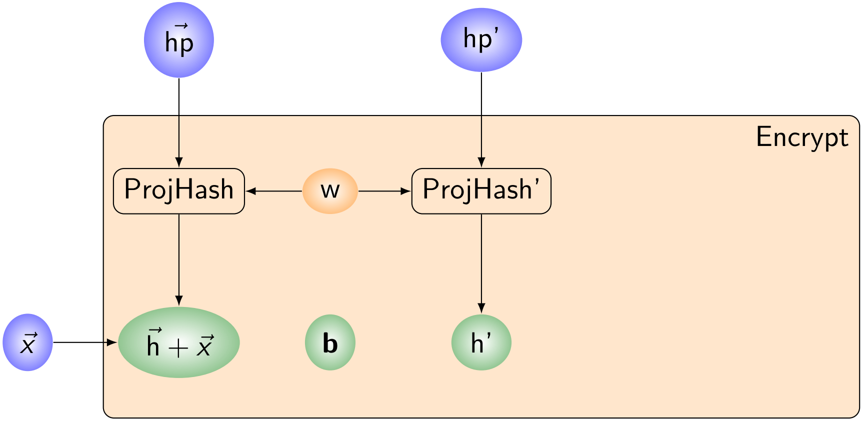
solution: Sign ct using OTS. Use vk as tag for PHF

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

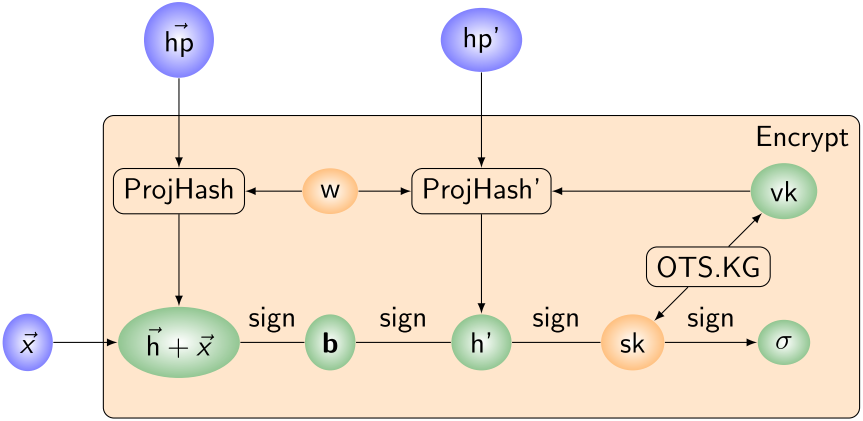
CCA-Security: Proving $\mathbf{b} \in \mathcal{L}$[CS98]



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



CCA-Security: Removing Malleability



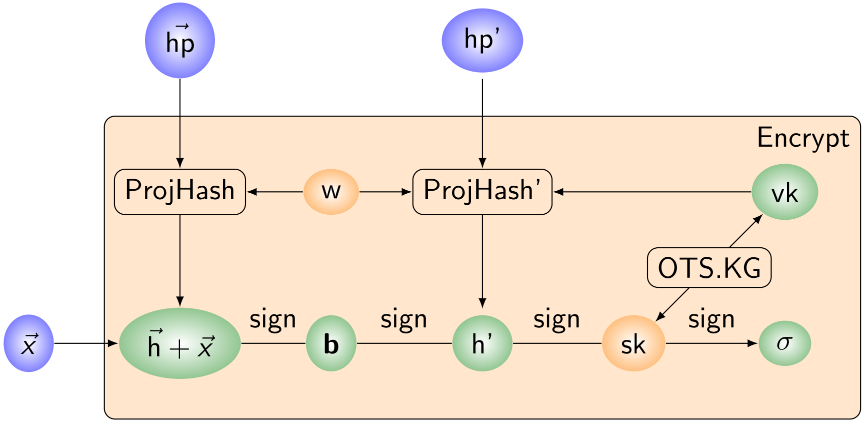
Nope!

This is not enough!

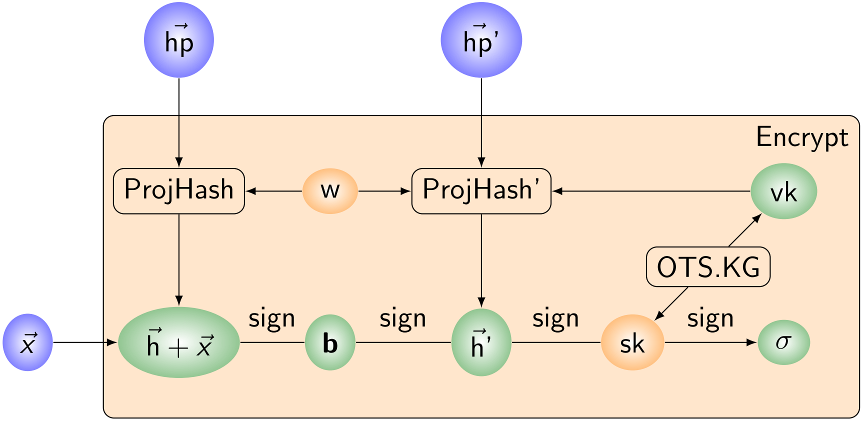
Nope!

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

CCA-Security



CCA-Security



Thank you!

Thanks for you attention! Question?