Memory-Tight Multi-Challenge Security of Public-Key Encryption

Joseph Jaeger and **Akshaya Kumar**

Georgia Tech

Asymptotic Security v/s Concrete Security

Asymptotic Security v/s Concrete Security

$\text{Adv}(A) \in [0,1]$

Asymptotic Security v/s Concrete Security

➔ **Asymptotic Security (parameterized by λ)** - Scheme S is secure if **all polynomially bounded** adversaries have negligible advantage

$Adv(A) \in [0,1]$

Asymptotic Security v/s Concrete Security

- **Asymptotic Security (parameterized by λ)** - Scheme S is secure if **all polynomially bounded** adversaries have negligible advantage

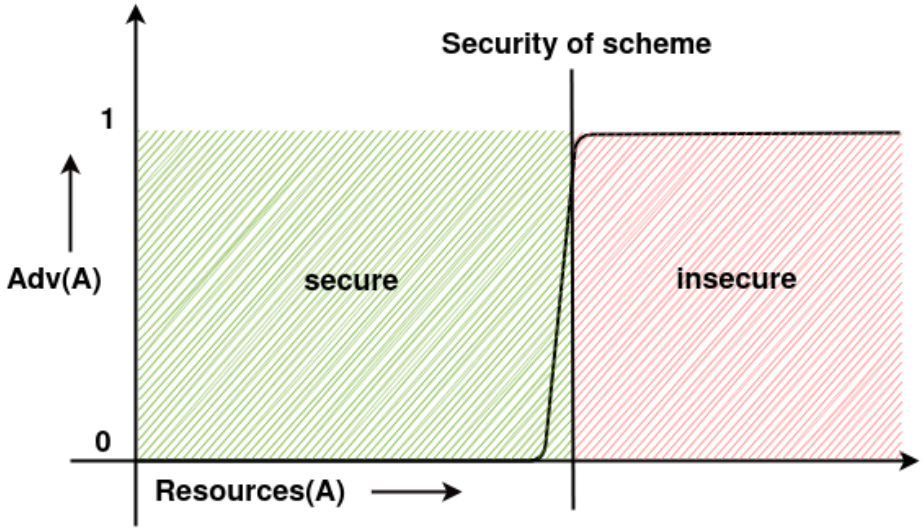
- **Concrete Security** - An instantiation of S[params] is secure if the advantage of **an adversary A** is bounded by a sufficiently small function of its **resources R**

$$Adv(A) \in [0,1]$$

Asymptotic Security v/s Concrete Security

- **Asymptotic Security (parameterized by λ)** - Scheme S is secure if **all polynomially bounded** adversaries have negligible advantage

- **Concrete Security** - An instantiation of S[params] is secure if the advantage of **an adversary A** is bounded by a sufficiently small function of its **resources R**

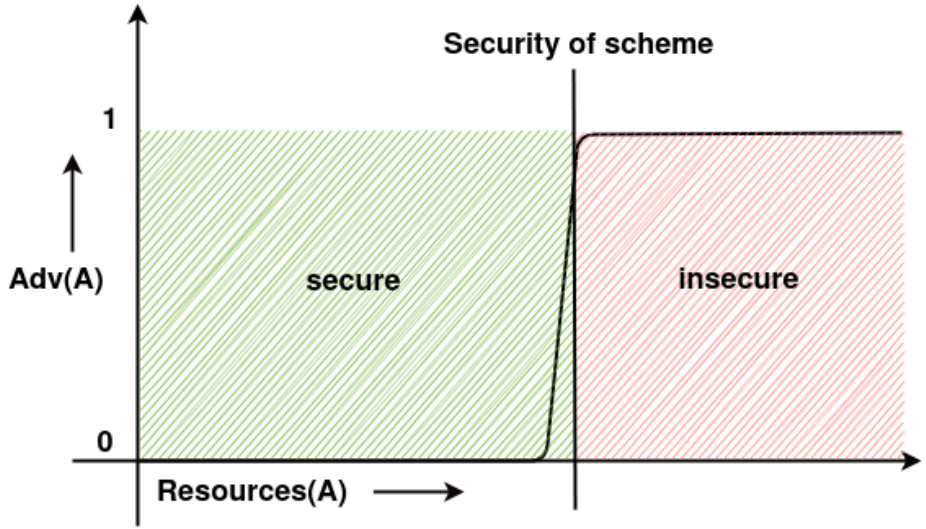


$Adv(A) \in [0,1]$

Asymptotic Security v/s Concrete Security

- **Asymptotic Security (parameterized by λ)** - Scheme S is secure if **all polynomially bounded** adversaries have negligible advantage

- **Concrete Security** - An instantiation of S[params] is secure if the advantage of **an adversary A** is bounded by a sufficiently small function of its **resources R**



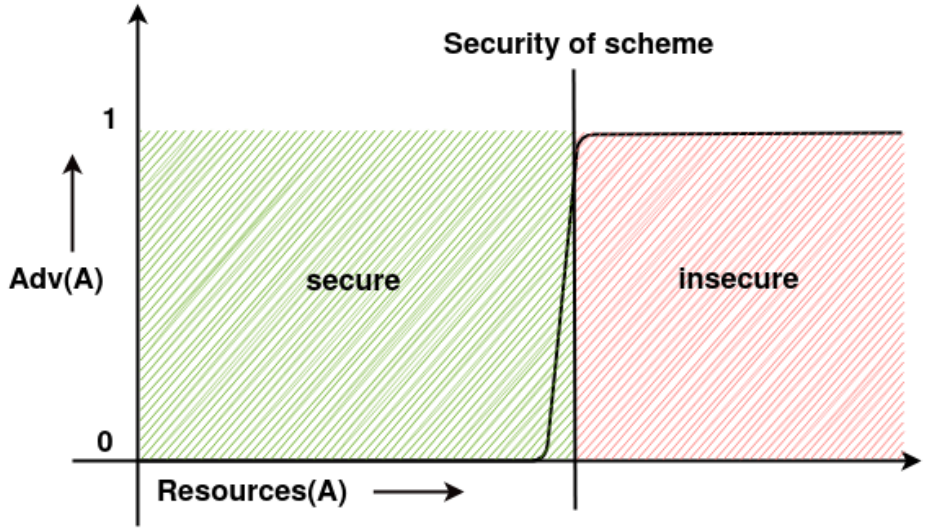
Adv(A) ∈ [0,1]

Usually **computational time**

Asymptotic Security v/s Concrete Security

- **Asymptotic Security (parameterized by λ)** - Scheme S is secure if **all polynomially bounded** adversaries have negligible advantage

- **Concrete Security** - An instantiation of S[params] is secure if the advantage of **an adversary A** is bounded by a sufficiently small function of its **resources R**

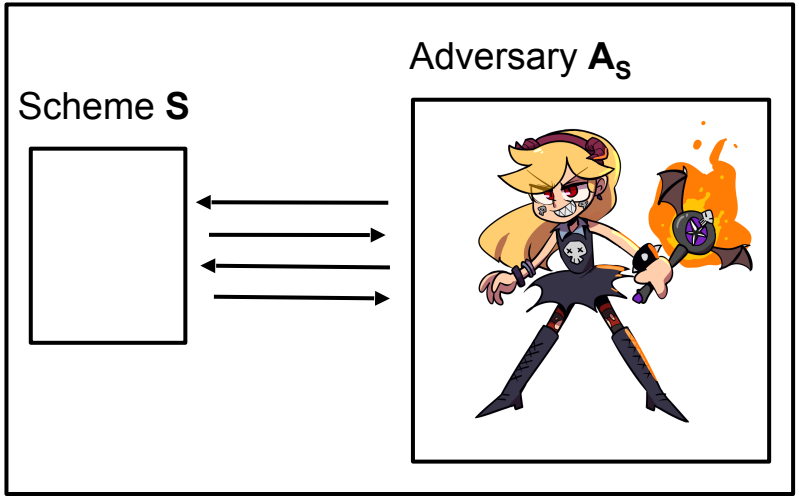


Adv(A) ∈ [0,1]

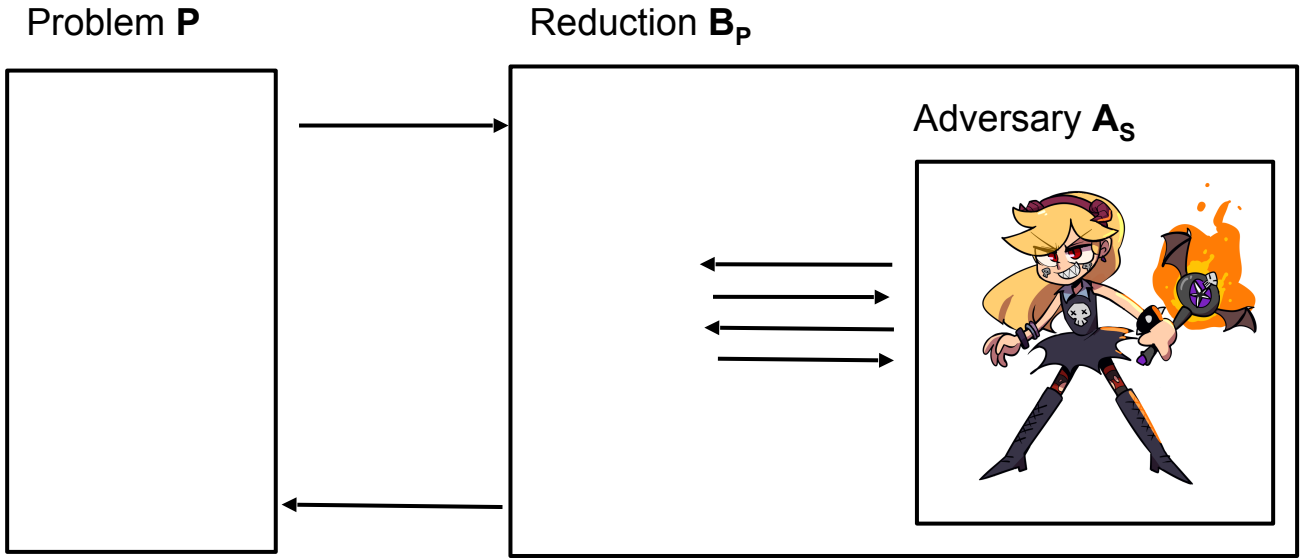
Memory?

Usually **computational time**

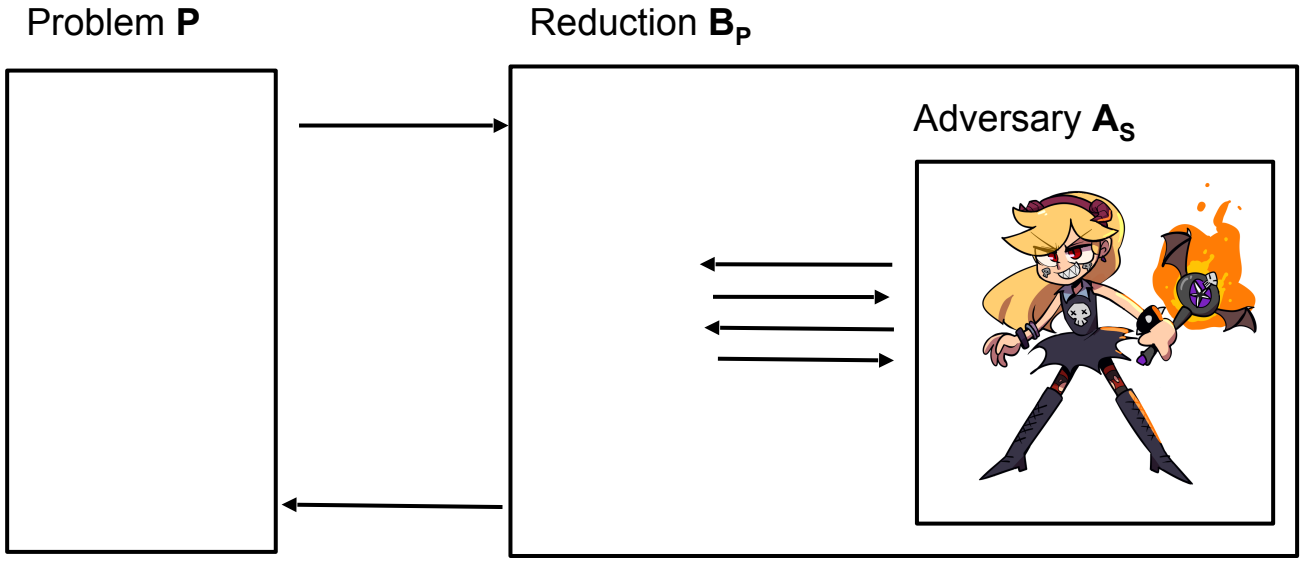
Cryptographic Reductions



Cryptographic Reductions

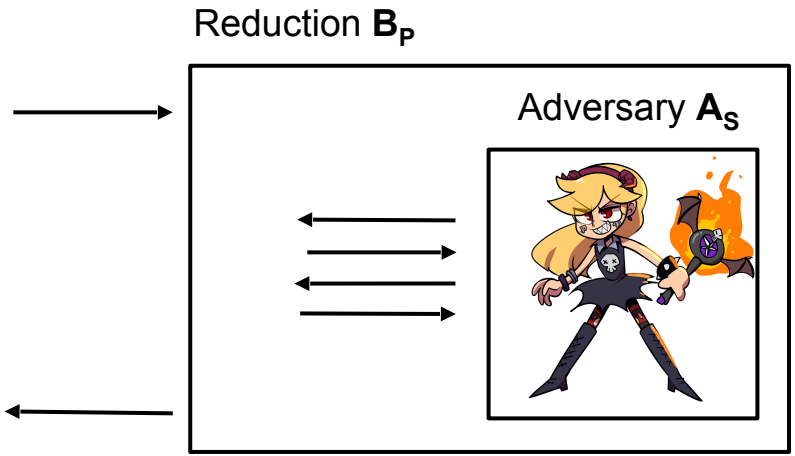


Cryptographic Reductions

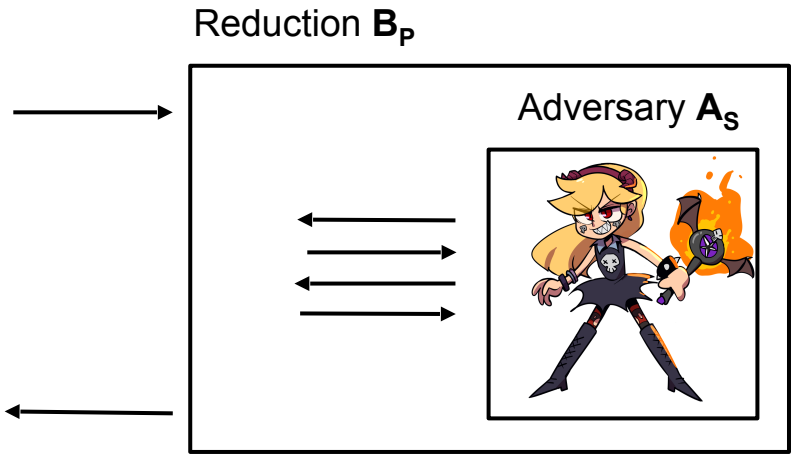


$$\mathbf{Adv}(\mathbf{A_S}) \leq \varepsilon(\mathbf{Adv}(\mathbf{B_P})),$$
$$\mathbf{Resources}(\mathbf{B_P}) \leq \Delta(\mathbf{Resources}(\mathbf{A_S}))$$

Tightness and Concrete Security



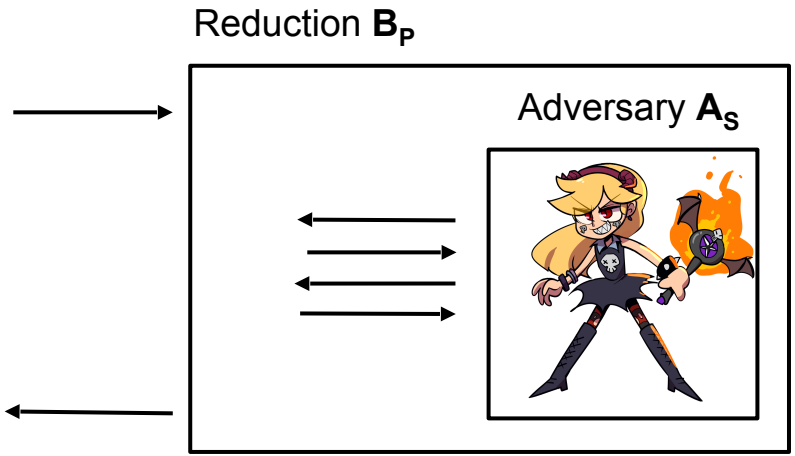
Tightness and Concrete Security

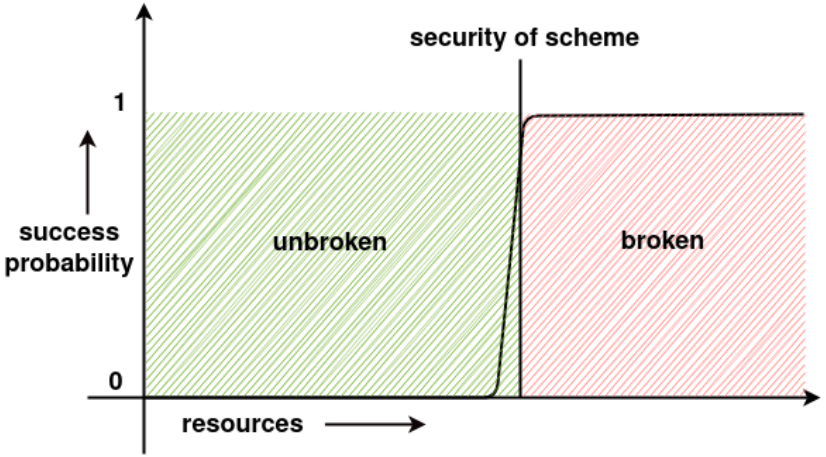


The reduction $\mathbf{B_P}$ is “tight” if

- ➔ $\mathbf{Adv}(\mathbf{B_P}) \approx \mathbf{Adv}(\mathbf{A_S})$
- ➔ $\mathbf{Resource}(\mathbf{B_P}) \approx \mathbf{Resource}(\mathbf{A_S})$

Tightness and Concrete Security

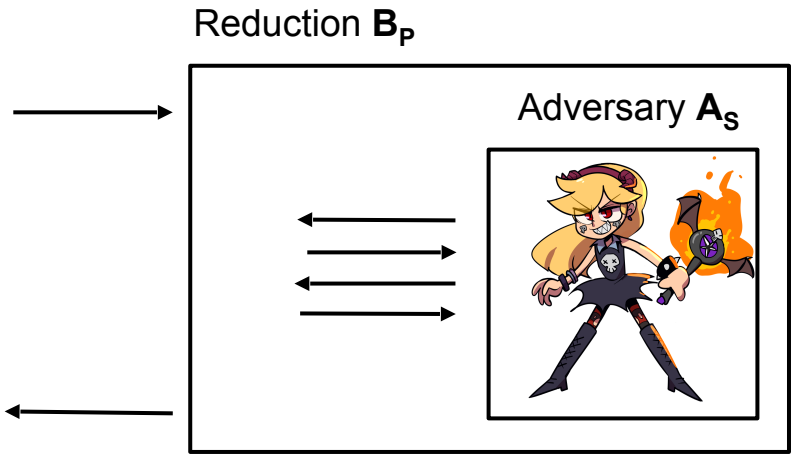


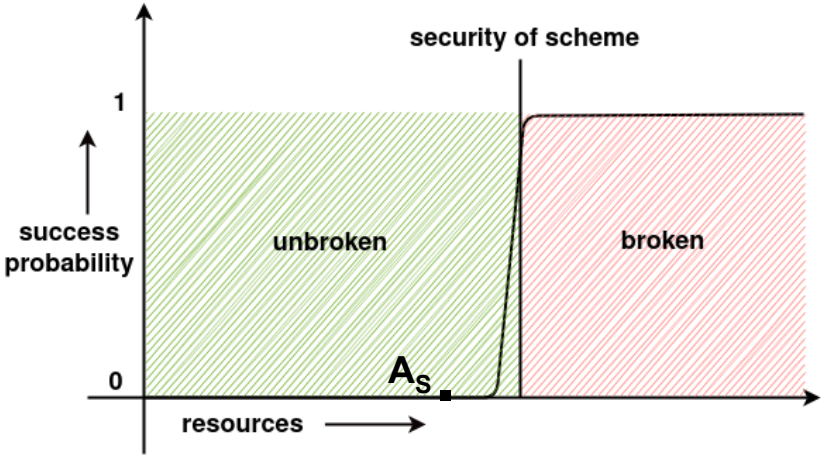


The reduction $\mathbf{B_P}$ is “tight” if

- ➔ $\mathbf{Adv(B_P) \approx Adv(A_S)}$
- ➔ $\mathbf{Resource(B_P) \approx Resource(A_S)}$

Tightness and Concrete Security

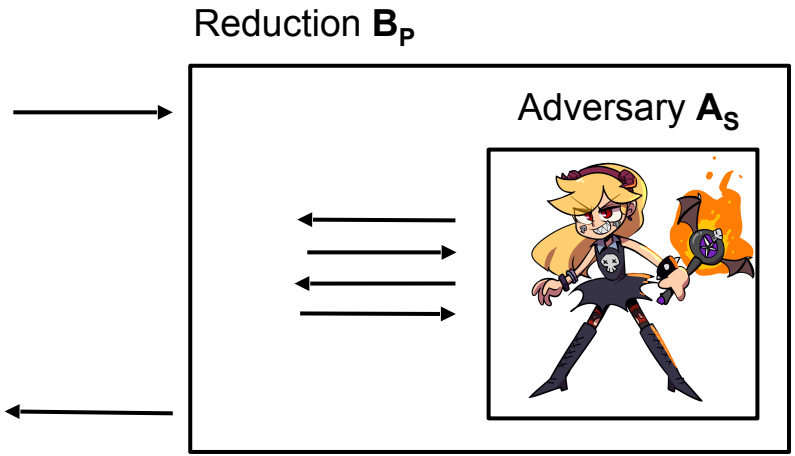


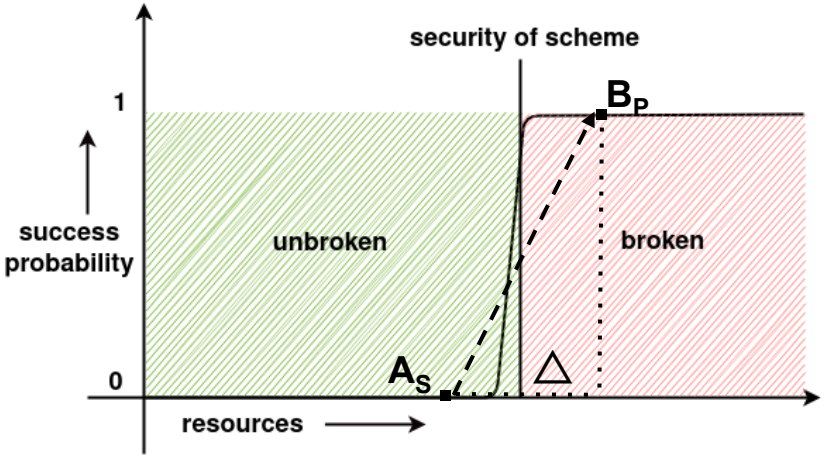


The reduction $\mathbf{B_P}$ is “tight” if

- ➔ $\mathbf{Adv(B_P) \approx Adv(A_S)}$
- ➔ $\mathbf{Resource(B_P) \approx Resource(A_S)}$

Tightness and Concrete Security

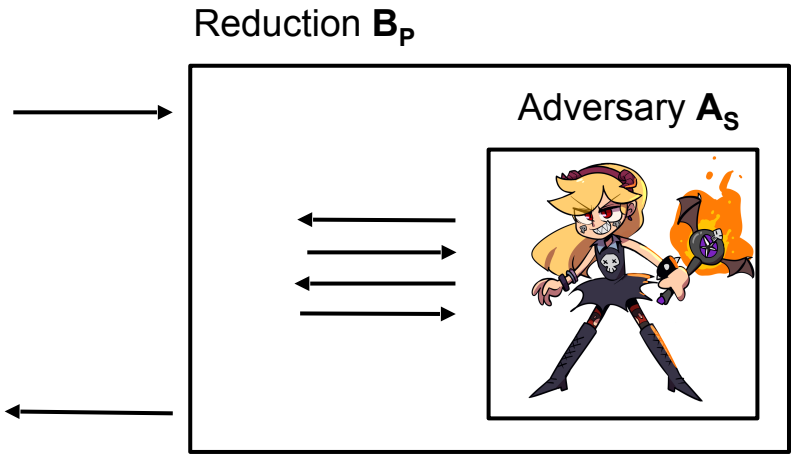




The reduction B_P is “tight” if

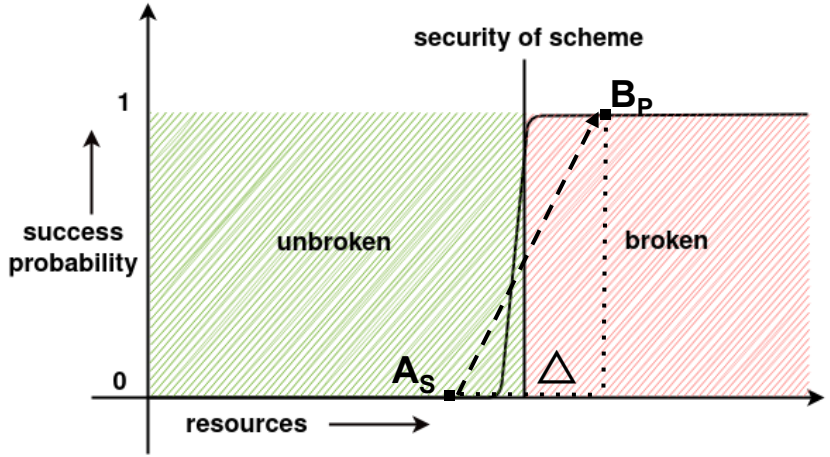
- ➔ $\mathbf{Adv}(B_P) \approx \mathbf{Adv}(A_S)$
- ➔ $\mathbf{Resource}(B_P) \approx \mathbf{Resource}(A_S)$

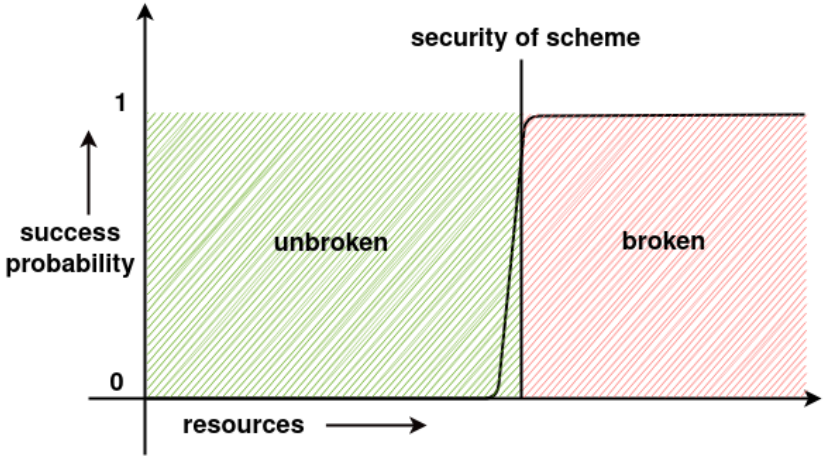
Tightness and Concrete Security



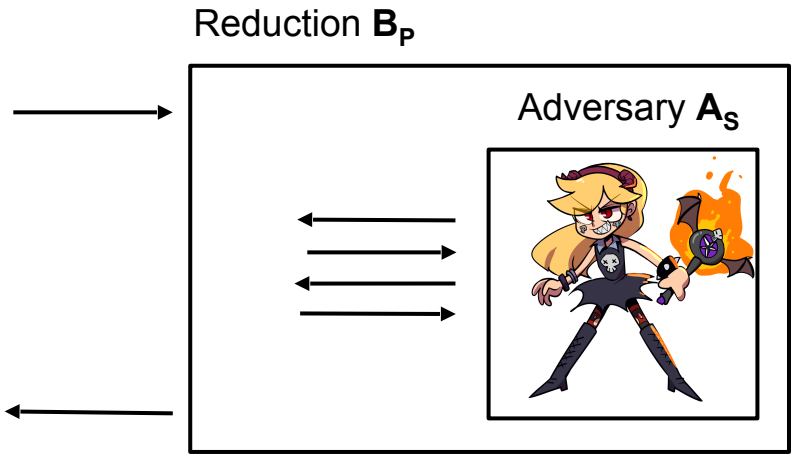
The reduction $\mathbf{B_P}$ is “tight” if

- ➔ $\mathbf{Adv(B_P) \approx Adv(A_S)}$
- ➔ $\mathbf{Resource(B_P) \approx Resource(A_S)}$



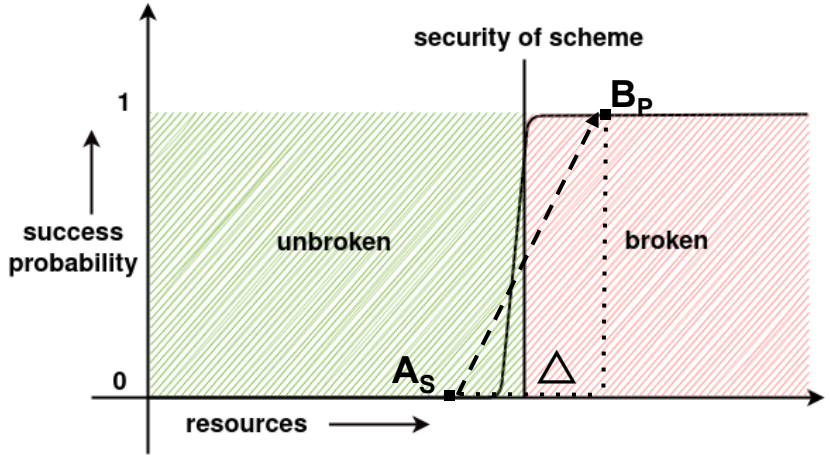


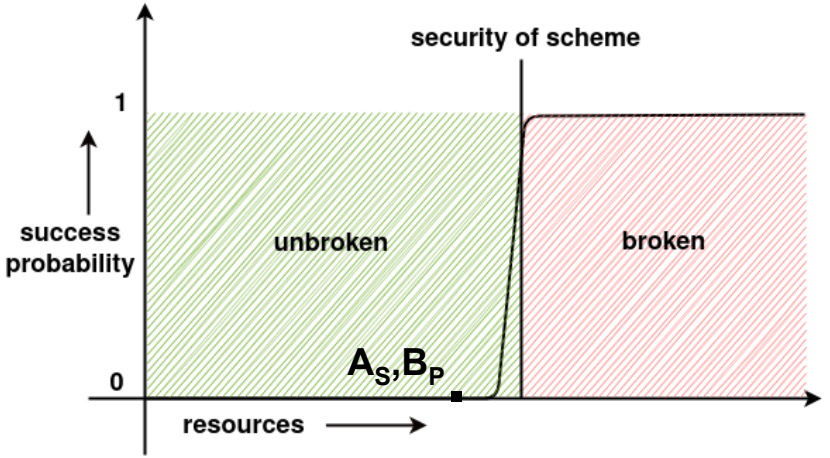
Tightness and Concrete Security



The reduction B_P is “tight” if

- ➔ $Adv(B_P) \approx Adv(A_S)$
- ➔ $Resource(B_P) \approx Resource(A_S)$





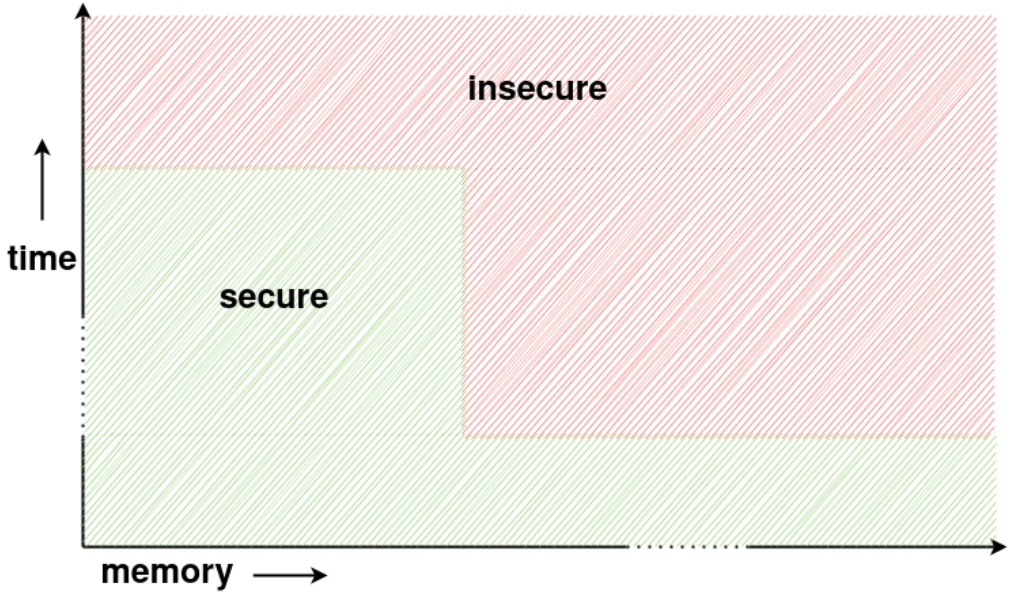
Time-Memory Trade Offs/Memory-Sensitive Problems

Time-Memory Trade Offs/Memory-Sensitive Problems

➔ DLP in prime fields - easier to solve with more memory/harder with less memory

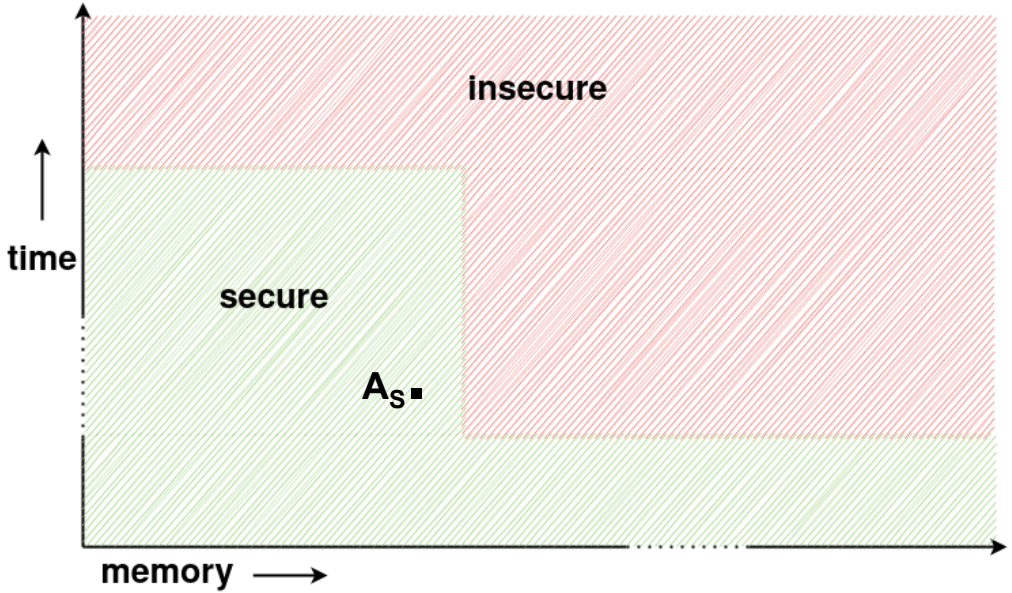
Time-Memory Trade Offs/Memory-Sensitive Problems

➔ DLP in prime fields - easier to solve with more memory/harder with less memory



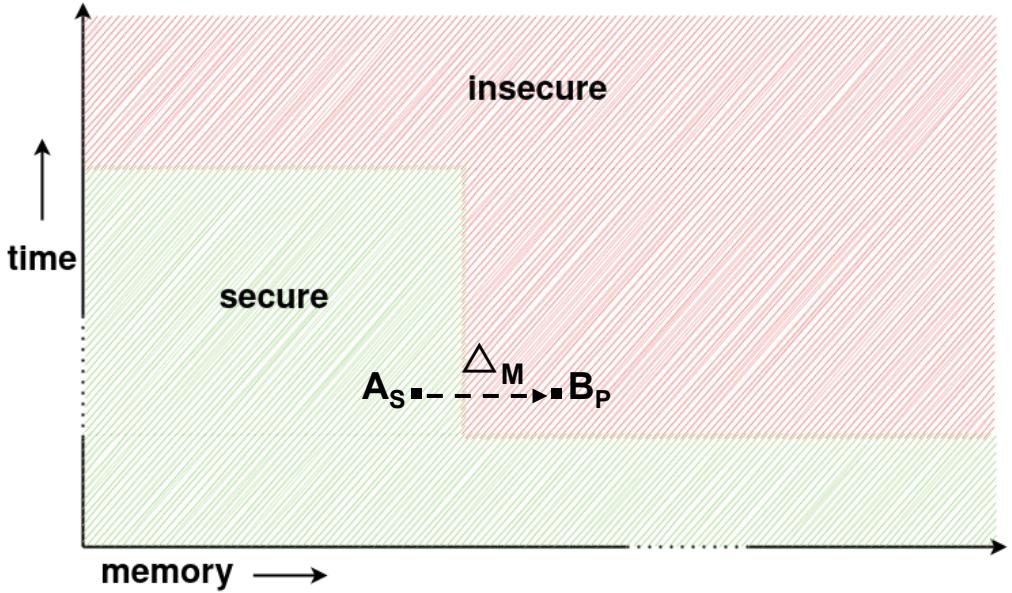
Time-Memory Trade Offs/Memory-Sensitive Problems

➔ DLP in prime fields - easier to solve with more memory/harder with less memory



Time-Memory Trade Offs/Memory-Sensitive Problems

➔ DLP in prime fields - easier to solve with more memory/harder with less memory



Memory-Tight Reductions [ACFK17]

Memory-Tight Reductions [ACFK17]

The reduction $\mathbf{B_P}$ is “memory-tight” if

- ➔ $\mathbf{Adv(B_P) \approx Adv(A_S)}$
- ➔ $\mathbf{Resource(B_P) \approx Resource(A_S)}$

$\mathbf{Time(B_P) \approx Time(A_S)}$

$\mathbf{Mem(B_P) \approx Mem(A_S)}$

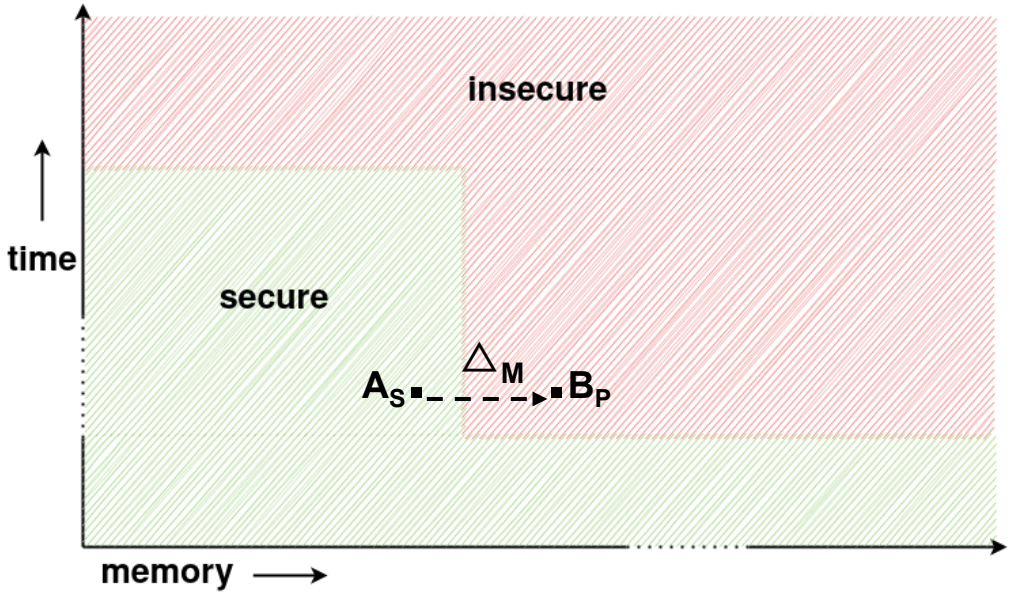
Memory-Tight Reductions [ACFK17]

The reduction $\mathbf{B_P}$ is "memory-tight" if

- ➔ $\mathbf{Adv(B_P) \approx Adv(A_S)}$
- ➔ $\mathbf{Resource(B_P) \approx Resource(A_S)}$

$\mathbf{Time(B_P) \approx Time(A_S)}$

$\mathbf{Mem(B_P) \approx Mem(A_S)}$



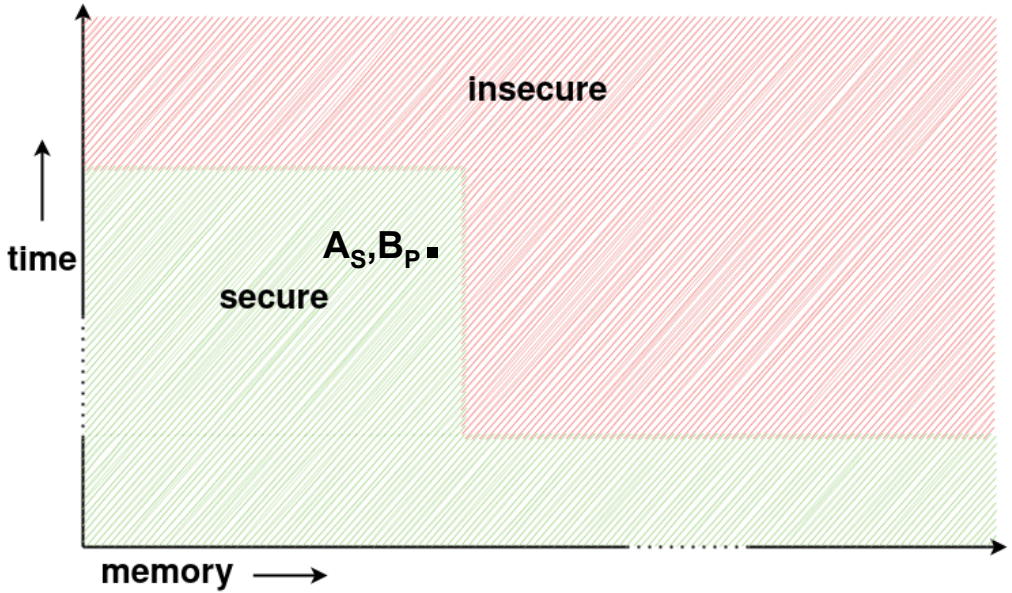
Memory-Tight Reductions [ACFK17]

The reduction $\mathbf{B_P}$ is "memory-tight" if

- ➔ $\mathbf{Adv(B_P) \approx Adv(A_S)}$
- ➔ $\mathbf{Resource(B_P) \approx Resource(A_S)}$

$\mathbf{Time(B_P) \approx Time(A_S)}$

$\mathbf{Mem(B_P) \approx Mem(A_S)}$



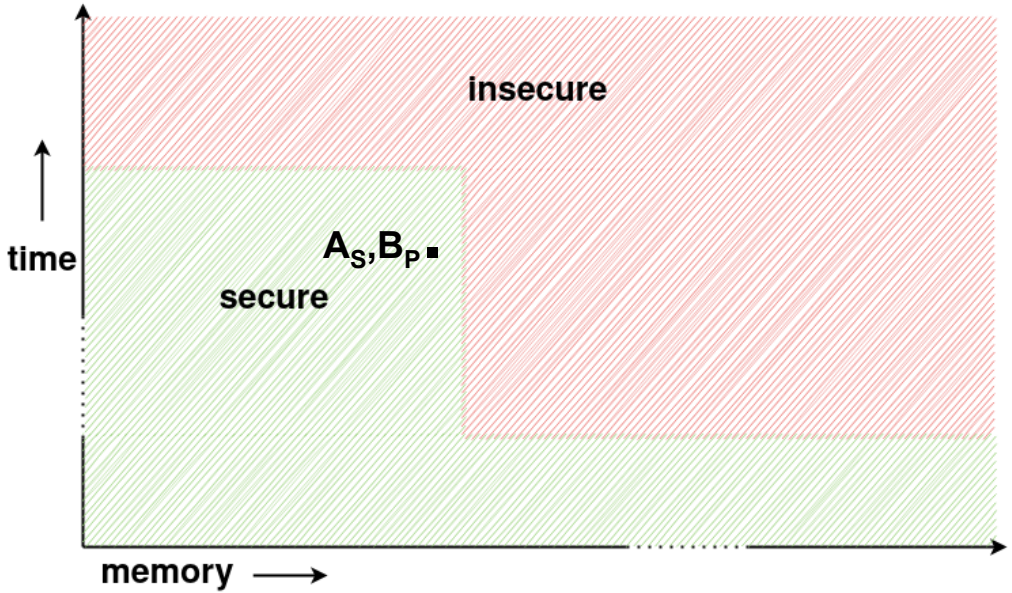
Memory-Tight Reductions [ACFK17]

The reduction $\mathbf{B_P}$ is "memory-tight" if

- ➔ $\mathbf{Adv(B_P) \approx Adv(A_S)}$
- ➔ $\mathbf{Resource(B_P) \approx Resource(A_S)}$

$\mathbf{Time(B_P) \approx Time(A_S)}$

$\mathbf{Mem(B_P) \approx Mem(A_S)}$



TAM-tight

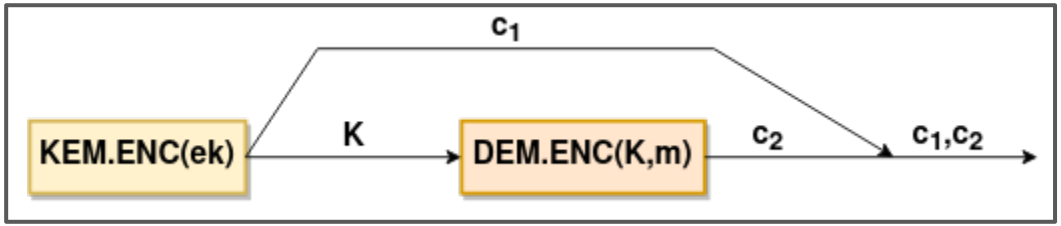
Our Focus - TAM tightness of PKE

Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm

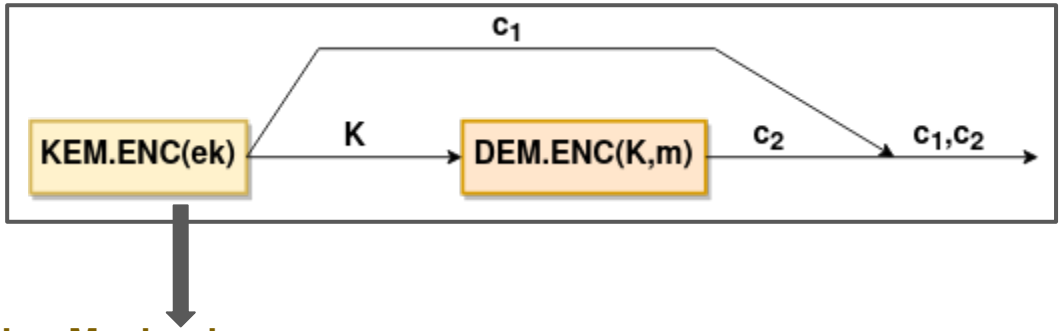
Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm



Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm

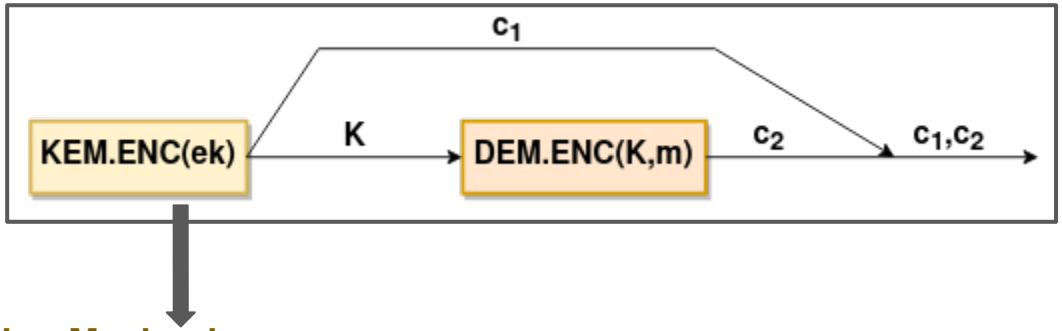


Key-Encapsulation Mechanism

[Bhattacharyya20]

Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm



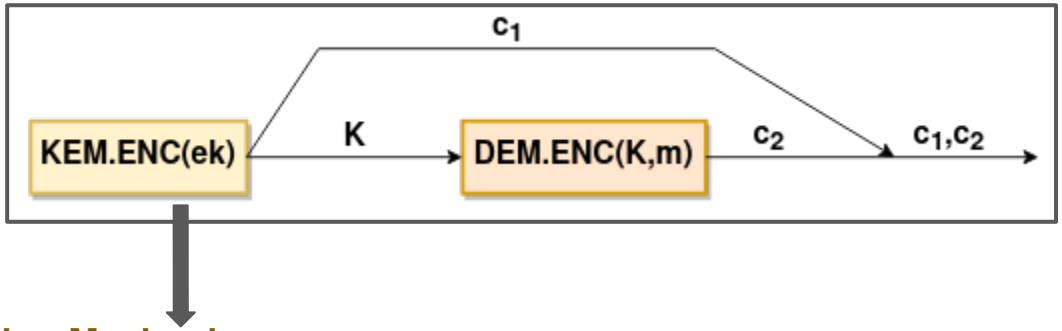
Key-Encapsulation Mechanism

Scheme S	Problem P
ECIES [AbdBelRog98]	CDH (Pairing-group)
Cramer-Shoup [CraSha01b]	Strong CDH
Fujisaki-Okamoto [FujOka13]	IND-CPA

[Bhattacharyya20]

Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm



Key-Encapsulation Mechanism

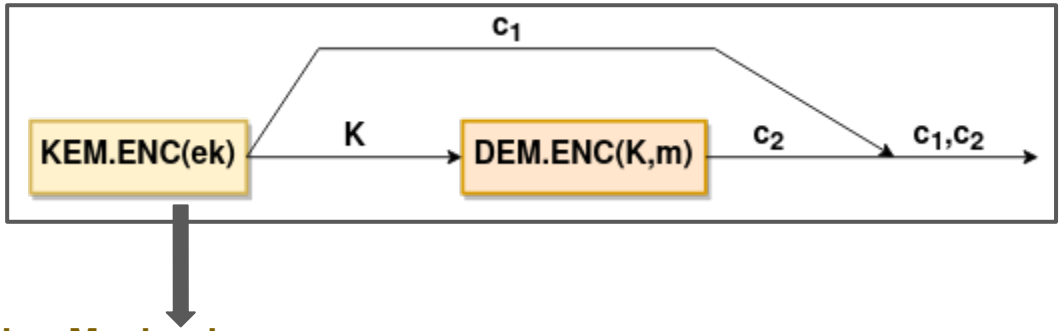
Scheme S	Problem P
ECIES [AbdBelRog98]	CDH (Pairing-group)
Cramer-Shoup [CraSha01b]	Strong CDH
Fujisaki-Okamoto [FujOka13]	IND-CPA

Hashed ElGamal

[Bhattacharyya20]

Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm



Key-Encapsulation Mechanism

Scheme S	Problem P
ECIES [AbdBelRog98]	CDH (Pairing-group)
Cramer-Shoup [CraSha01b]	Strong CDH
Fujisaki-Okamoto [FujOka13]	IND-CPA

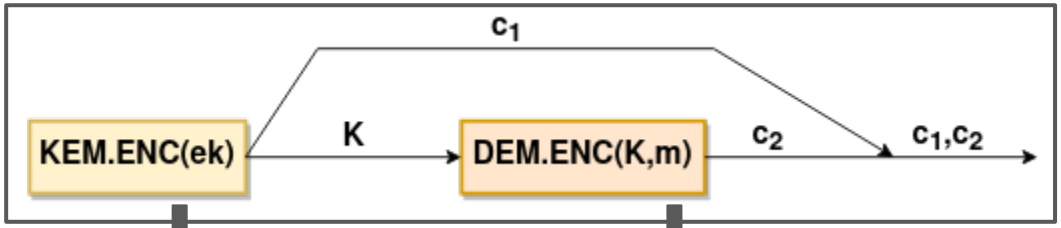
Hashed ElGamal

[Bhattacharyya20]

Multi-challenge/multi-user security
- ➔ schemes get deployed across multiple users
- ➔ users use scheme multiple times
- ➔ does security degrade with #users/queries?
- ➔ tight proofs imply not degrading

Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm



Key-Encapsulation Mechanism

Symmetric Encryption

Scheme S	Problem P
ECIES [AbdBelRog98]	CDH (Pairing-group)
Cramer-Shoup [CraSha01b]	Strong CDH
Fujisaki-Okamoto [FujOka13]	IND-CPA

Hashed ElGamal

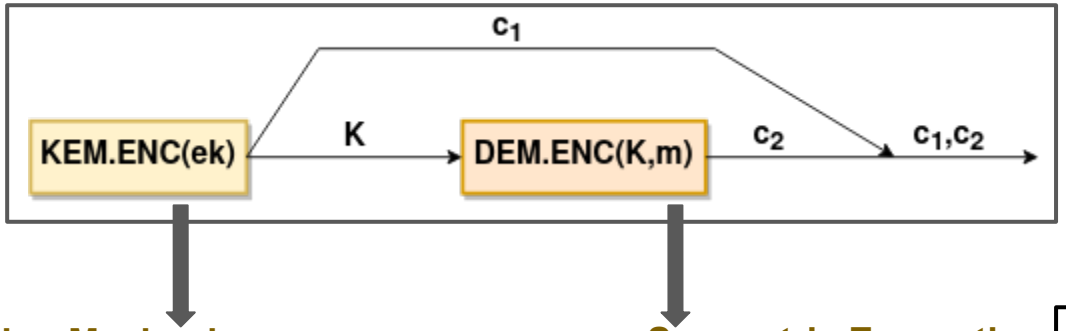
[Bhattacharyya20]

Multi-challenge/multi-user security

- ➔ schemes get deployed across multiple users
- ➔ users use scheme multiple times
- ➔ does security degrade with #users/queries?
- ➔ tight proofs imply not degrading

Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm



Key-Encapsulation Mechanism

Symmetric Encryption

Open question!

Scheme S	Problem P
ECIES [AbdBelRog98]	CDH (Pairing-group)
Cramer-Shoup [CraSha01b]	Strong CDH
Fujisaki-Okamoto [FujOka13]	IND-CPA

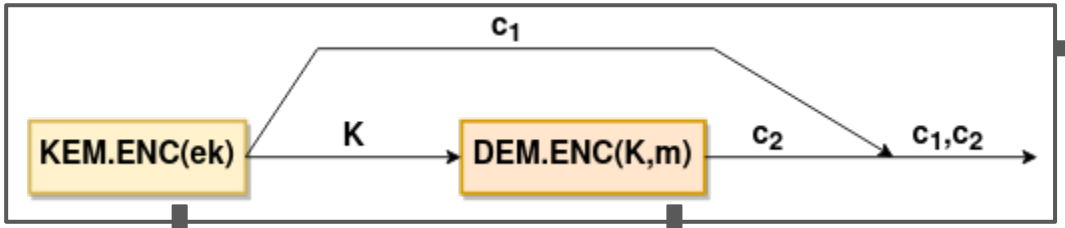
Hashed ElGamal

[Bhattacharyya20]

Multi-challenge/multi-user security
- ➔ schemes get deployed across multiple users
- ➔ users use scheme multiple times
- ➔ does security degrade with #users/queries?
- ➔ tight proofs imply not degrading

Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm



Composition

Key-Encapsulation Mechanism

Symmetric Encryption

Open question!

Scheme S	Problem P
ECIES [AbdBelRog98]	CDH (Pairing-group)
Cramer-Shoup [CraSha01b]	Strong CDH
Fujisaki-Okamoto [FujOka13]	IND-CPA

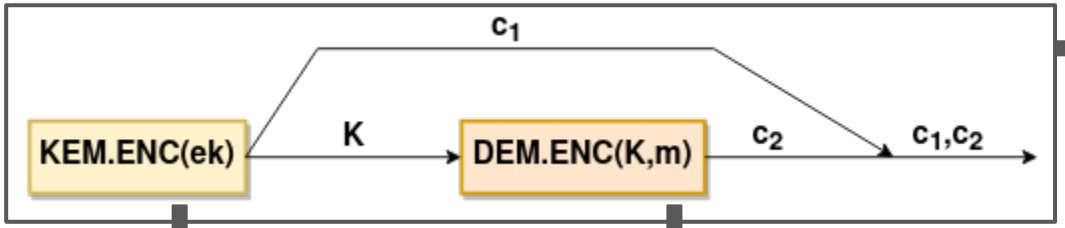
Hashed ElGamal

[Bhattacharyya20]

Multi-challenge/multi-user security
- ➔ schemes get deployed across multiple users
- ➔ users use scheme multiple times
- ➔ does security degrade with #users/queries?
- ➔ tight proofs imply not degrading

Our Focus - TAM tightness of PKE

➔ Approach: KEM/DEM paradigm



Composition [GGJT22]

Message Encoding

Key-Encapsulation Mechanism

Symmetric Encryption Open question!

Scheme S	Problem P
ECIES [AbdBelRog98]	CDH (Pairing-group)
Cramer-Shoup [CraSha01b]	Strong CDH
Fujisaki-Okamoto [FujOka13]	IND-CPA

Hashed ElGamal

[Bhattacharyya20]

Multi-challenge/multi-user security
- ➔ schemes get deployed across multiple users
- ➔ users use scheme multiple times
- ➔ does security degrade with #users/queries?
- ➔ tight proofs imply not degrading

Challenges with Multi-Challenge Security

Challenges with Multi-Challenge Security

Standard - CCA security

Challenges with Multi-Challenge Security

Standard - CCA security

ENC

O

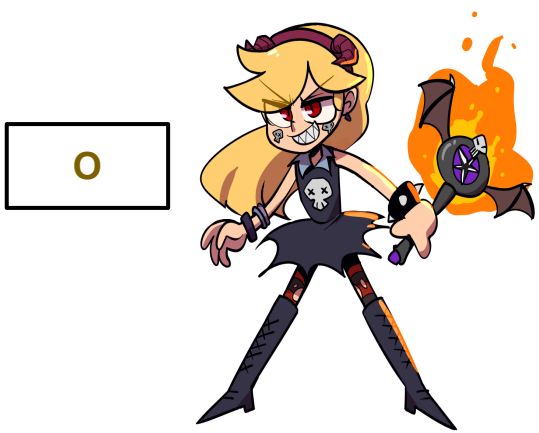
DEC

Challenges with Multi-Challenge Security

Standard - CCA security

Challenges with Multi-Challenge Security

Standard - CCA security



ENC	
Real	Ideal
c, k* ⟵ KEM.E(ek)	**c*, k*** ⟵ $

Challenges with Multi-Challenge Security

Standard - CCA security

O

ENC

c*, k*

DEC

ENC	
Real	Ideal
c, k⟵ KEM.E(ek)	**c*, k***⟵ $

Challenges with Multi-Challenge Security

Standard - CCA security

ENC	
Real	Ideal
c, k* ⟵ KEM.E(ek)	**c*, k**** ⟵ $

Challenges with Multi-Challenge Security

Standard - CCA security

ENC	
Real	Ideal
c, k* ⟵ KEM.E(ek)	**c*, k*** ⟵ $

DEC
KEM.D(dk, c) if c ≠ **c*** ; **k*** otherwise

Challenges with Multi-Challenge Security

Standard - CCA security

ENC	
Real	Ideal
c, k* ⟵ KEM.E(ek)	**c*, k*** ⟵ $

DEC
KEM.D(dk, c) if c ≠ **c*** ; **k*** otherwise

Challenges with Multi-Challenge Security

Standard - CCA security

ENC	
Real	Ideal
c, k* ⟵ KEM.E(ek)	**c*, k*** ⟵ $

DEC
KEM.D(dk, c) if c ≠ **c*** ; **k*** otherwise

Multiple **c*, k*** pairs
- ➔ Identify challenge ciphertexts in **SIMDEC**
- ➔ Identify challenge secrets in **SIMO**
- ➔ Consistent outputs from **SIMDEC**, **SIMO**

Our Contributions

Our Contributions

1. Multi-challenge/user TAM-tightness of “augmented” KEMs

Our Contributions

1. Multi-challenge/user TAM-tightness of “augmented” KEMs

Scheme S	**Problem P**
aECIES [AbdBelRog98]	Pair CDH
aCramer-Shoup [CraSha01b]	Strong CDH
aTwin ElGamal [CasKilSho09]	CDH
aFujisaki-Okamoto [FujOka13]	IND-CPA

Our Contributions

1. Multi-challenge/user TAM-tightness of “augmented” KEMs

Hashed ElGamal

Scheme S	Problem P
aECIES [AbdBelRog98]	Pair CDH
aCramer-Shoup [CraSha01b]	Strong CDH
aTwin ElGamal [CasKilSho09]	CDH
aFujisaki-Okamoto [FujOka13]	IND-CPA

Our Contributions

1. Multi-challenge/user TAM-tightness of “augmented” KEMs

Hashed ElGamal

Fujisaki-Okamoto

Scheme S	Problem P
aECIES [AbdBelRog98]	Pair CDH
aCramer-Shoup [CraSha01b]	Strong CDH
aTwin ElGamal [CasKilSho09]	CDH
aFujisaki-Okamoto [FujOka13]	IND-CPA

Our Contributions

1. Multi-challenge/user TAM-tightness of “augmented” KEMs

Hashed ElGamal

Fujisaki-Okamoto

Scheme S	Problem P
aECIES [AbdBelRog98]	Pair CDH
aCramer-Shoup [CraSha01b]	Strong CDH
aTwin ElGamal [CasKilSho09]	CDH
aFujisaki-Okamoto [FujOka13]	IND-CPA

NEW!

Our Contributions

1. Multi-challenge/user TAM-tightness of “augmented” KEMs

Hashed ElGamal

Fujisaki-Okamoto

Scheme S	Problem P
aECIES [AbdBelRog98]	Pair CDH
aCramer-Shoup [CraSha01b]	Strong CDH
aTwin ElGamal [CasKilSho09]	CDH
aFujisaki-Okamoto [FujOka13]	IND-CPA

NEW!

2. Multi-challenge/user TAM-tightness of KEM/DEM

Hashed ElGamal KEMs

Hashed ElGamal KEMs

$\underline{\mathsf{KEM.K}}$
$x \leftarrow_{\$} \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return (ek, dk)

ECIES	Cramer-Shoup
$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$ $y \leftarrow_{\$} \mathbb{Z}_p^*$ $Y \leftarrow \mathbf{g}^y$ $Z \leftarrow ek^y$ $K \leftarrow \mathcal{H}(Z)$ Return (Y, K)	$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$ $y \leftarrow_{\$} \mathbb{Z}_p^*$ $Y \leftarrow \mathbf{g}^y$ $Z \leftarrow ek^y$ $K \leftarrow \mathcal{H}(Y, Z)$ Return (Y, K)
$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$ $Z \leftarrow Y^{dk}$ $K \leftarrow \mathcal{H}(Z)$ Return K	$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$ $Z \leftarrow Y^{dk}$ $K \leftarrow \mathcal{H}(Y, Z)$ Return K

Hashed ElGamal KEMs

$\underline{\mathsf{KEM.K}}$
$x \leftarrow_\$ \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return (ek, dk)

ECIES	Cramer-Shoup
$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$ $y \leftarrow_\$ \mathbb{Z}_p^*$ $Y \leftarrow \mathbf{g}^y$ $Z \leftarrow ek^y$ $K \leftarrow \mathcal{H}(Z)$ Return (Y, K)	$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$ $y \leftarrow_\$ \mathbb{Z}_p^*$ $Y \leftarrow \mathbf{g}^y$ $Z \leftarrow ek^y$ $K \leftarrow \mathcal{H}(Y, Z)$ Return (Y, K)
$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$ $Z \leftarrow Y^{dk}$ $K \leftarrow \mathcal{H}(Z)$ Return K	$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$ $Z \leftarrow Y^{dk}$ $K \leftarrow \mathcal{H}(Y, Z)$ Return K

The CDH game

Single Challenge

Given X, Y, find $\mathbf{g}^{xy}$ where $x = \mathrm{dlog}(X), y = \mathrm{dlog}(Y)$.

Hashed ElGamal KEMs

$\underline{\mathsf{KEM.K}}$
$x \leftarrow_{\$} \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return (ek, dk)

ECIES	Cramer-Shoup
$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$ $y \leftarrow_{\$} \mathbb{Z}_p^*$ $Y \leftarrow \mathbf{g}^y$ $Z \leftarrow ek^y$ $K \leftarrow \mathcal{H}(Z)$ Return (Y, K)	$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$ $y \leftarrow_{\$} \mathbb{Z}_p^*$ $Y \leftarrow \mathbf{g}^y$ $Z \leftarrow ek^y$ $K \leftarrow \mathcal{H}(Y, Z)$ Return (Y, K)
$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$ $Z \leftarrow Y^{dk}$ $K \leftarrow \mathcal{H}(Z)$ Return K	$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$ $Z \leftarrow Y^{dk}$ $K \leftarrow \mathcal{H}(Y, Z)$ Return K

The CDH game

Single Challenge

Given X, Y, find $\mathbf{g}^{xy}$ where $x = \mathrm{dlog}(X), y = \mathrm{dlog}(Y)$.

Given $X, Y_1, Y_2, Y_3, ..., Y_n$, find $\mathbf{g}^{xy_i}$ for some $i \in [1, n]$ where $x = \mathrm{dlog}(X), y_i = \mathrm{dlog}(Y_i)$.

Multi-Challenge

Hashed ElGamal KEMs

KEM.K

$x \leftarrow_{\$} \mathbb{Z}_p^*$

$ek \leftarrow \mathbf{g}^x$

$dk \leftarrow x$

Return (ek, dk)

ECIES	Cramer-Shoup
KEM.E$^{\mathcal{H}}(ek)$ $y \leftarrow_{\$} \mathbb{Z}_p^*$ $Y \leftarrow \mathbf{g}^y$ $Z \leftarrow ek^y$ $K \leftarrow \mathcal{H}(Z)$ Return (Y, K)	KEM.E$^{\mathcal{H}}(ek)$ $y \leftarrow_{\$} \mathbb{Z}_p^*$ $Y \leftarrow \mathbf{g}^y$ $Z \leftarrow ek^y$ $K \leftarrow \mathcal{H}(Y, Z)$ Return (Y, K)
KEM.D$^{\mathcal{H}}(dk, Y)$ $Z \leftarrow Y^{dk}$ $K \leftarrow \mathcal{H}(Z)$ Return K	KEM.D$^{\mathcal{H}}(dk, Y)$ $Z \leftarrow Y^{dk}$ $K \leftarrow \mathcal{H}(Y, Z)$ Return K

The CDH game

Single Challenge

Given X, Y, find $\mathbf{g}^{xy}$ where $x = \text{dlog}(X), y = \text{dlog}(Y)$.

Given $X, Y_1, Y_2, Y_3, ..., Y_n$, find $\mathbf{g}^{xy_i}$ for some $i \in [1, n]$ where $x = \text{dlog}(X), y_i = \text{dlog}(Y_i)$.

Multi-Challenge

Single Challenge ⇒ Multi-Challenge (TAM tight from DH rerandomization!)

ECIES

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

ENC

DEC

H

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

ENC

DEC

H

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

Y_i, $H(Y_i^x)$

ENC

DEC

H

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

Y_i, $H(Y_i^x)$

ENC

Y

DEC

H

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

Y_i, $H(Y_i^x)$

ENC

Y

$H(Y^x)$

DEC

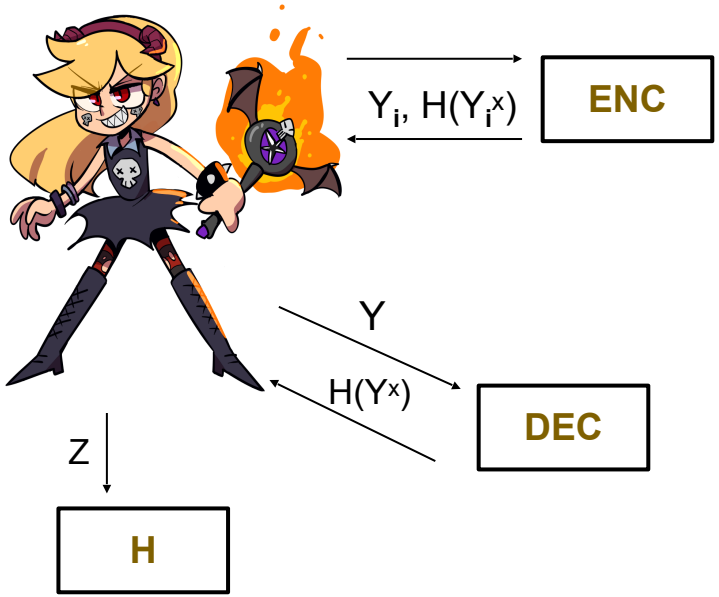
H

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K



ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

Y_i, $H(Y_i^x)$

ENC

Y

$H(Y^x)$

DEC

Z

H(Z)

H

$X, Y_1, Y_2, \ldots, Y_n$

REDUCTION R

Z

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

Y_i, $H(Y_i^x)$

ENC

SIMENC

Return Y_i, $H'(Y_i)$

Y

$H(Y^x)$

DEC

Z

H(Z)

H

$X, Y_1, Y_2, \ldots, Y_n$

REDUCTION R

Z

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

Y_i, $H(Y_i^x)$

ENC

SIMENC

Return Y_i, $H'(Y_i)$

Y

$H(Y^x)$

DEC

Z

H(Z)

H

SIMDEC(Y)

Return H'(Y)

$X, Y_1, Y_2, \ldots, Y_n$

REDUCTION R

Z

ECIES

ECIES

$\underline{\mathrm{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathrm{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

Y_i, $H(Y_i^x)$

ENC

SIMENC

Return Y_i, $H'(Y_i)$

Y

$H(Y^x)$

DEC

Z

H(Z)

H

SIMH(Z)

Return H(Z)

SIMDEC(Y)

Return H'(Y)

$X, Y_1, Y_2, \ldots, Y_n$

REDUCTION R

Z

ECIES

ECIES

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Z)$
Return K

Y_i, $H(Y_i^x)$

ENC

SIMENC

Return Y_i, $H'(Y_i)$

Y

$H(Y^x)$

DEC

Z

H(Z)

H

SIMH(Z)

Return H(Z)

SIMDEC(Y)

Return H'(Y)

$X, Y_1, Y_2, \ldots, Y_n$

REDUCTION R

Z

Reduction Goals

- ➔ ~~Identify challenge ciphertexts in **SIMDEC**~~
- ➔ Identify challenge secrets in **SIMH**
- ➔ Consistent outputs from **SIMDEC**, **SIMH**

ECIES - Standard Proof - Not memory/time-tight

ECIES - Standard Proof - Not memory/time-tight

SIMENC
Return Y_i, $H'(Y_i)$

SIMDEC(Y)
Return $H'(Y)$

SIMH(Z)
Return $H(Z)$

ECIES - Standard Proof - Not memory/time-tight

SIMENC
Return Y_i, $H'(Y_i)$

SIMDEC(Y)
Return H'(Y)

SIMH(Z)
Return H(Z)

$\underline{\textsc{Strong}(Y, Z)}$
$y \leftarrow \mathrm{dlog}(A)$
Return $(Z = X^y)$

ECIES - Standard Proof - Not memory/time-tight

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

Return $H'(Y)$

SIMH(Z)

Return $H(Z)$

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return $H'(Y)$
Return $H(Z)$

$\underline{\textsc{Strong}(Y, Z)}$
$y \leftarrow \text{dlog}(A)$
Return $(Z = X^y)$

ECIES - Standard Proof - Not memory/time-tight

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

Return $H'(Y)$

SIMH(Z)

Return $H(Z)$

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

$\underline{\text{STRONG}(Y, Z)}$
$y \leftarrow \text{dlog}(A)$
Return $(Z = X^y)$

ECIES - Standard Proof - Not memory/time-tight

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

Return $H'(Y)$

SIMH(Z)

Return $H(Z)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return $H'(Y)$

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return $H'(Y)$
Return H(Z)

$\underline{\text{STRONG}(Y, Z)}$
$y \leftarrow \text{dlog}(A)$
Return $(Z = X^y)$

ECIES - Standard Proof - Not memory/time-tight

SIMENC

Return Y_i, H'(Y_i)

SIMDEC(Y)

Return H'(Y)

SIMH(Z)

Return H(Z)

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

$\underline{\textsc{Strong}(Y, Z)}$
$y \leftarrow \mathrm{dlog}(A)$
Return $(Z = X^y)$

ECIES - Standard Proof - Not memory/time-tight

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

Return $H'(Y)$

SIMH(Z)

Return $H(Z)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

$\underline{\text{STRONG}(Y, Z)}$
$y \leftarrow \text{dlog}(A)$
Return $(Z = X^y)$

ECIES - Standard Proof - Not memory/time-tight

SIMENC

Return Y_i, H'(Y_i)

SIMDEC(Y)

Return H'(Y)

SIMH(Z)

Return H(Z)

SIMENC

Return Y_i, H'(Y_i)

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

$\underline{\textsc{Strong}(Y, Z)}$
$y \leftarrow \text{dlog}(A)$
Return $(Z = X^y)$

ECIES - Standard Proof - Not memory/time-tight

SIMENC

Return Y_i, H'(Y_i)

SIMDEC(Y)

Return H'(Y)

SIMH(Z)

Return H(Z)

SIMENC

Return Y_i, H'(Y_i)

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

$\underline{\text{STRONG}(Y, Z)}$
$y \leftarrow \text{dlog}(A)$
Return $(Z = X^y)$

Reduction Goals

- ➔ ~~Identify challenge ciphertexts in **SIMDEC**~~
- ➔ ~~Identify challenge secrets in **SIMH**~~
- ➔ ~~Consistent outputs from **SIMDEC**, **SIMH**~~

ECIES - Standard Proof - Not memory/time-tight

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

Return $H'(Y)$

SIMH(Z)

Return $H(Z)$

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

Not time/memory tight 😕

$\underline{\text{STRONG}(Y, Z)}$
$y \leftarrow \text{dlog}(A)$
Return $(Z = X^y)$

Reduction Goals

- ➔ ~~Identify challenge ciphertexts in **SIMDEC**~~
- ➔ ~~Identify challenge secrets in **SIMH**~~
- ➔ ~~Consistent outputs from **SIMDEC**, **SIMH**~~

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

Assumes a pairing friendly group

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMH(Z)

If ∃ Y_i s.t. pair(X,Y_i) = pair(g, Z):
    Win with Z
Return H(pair(g,Z))

Assumes a pairing friendly group

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMH(Z)

If ∃ Y_i s.t. pair(X,Y_i) = pair(g, Z):
    Win with Z
Return H(pair(g,Z))

Assumes a pairing friendly group

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

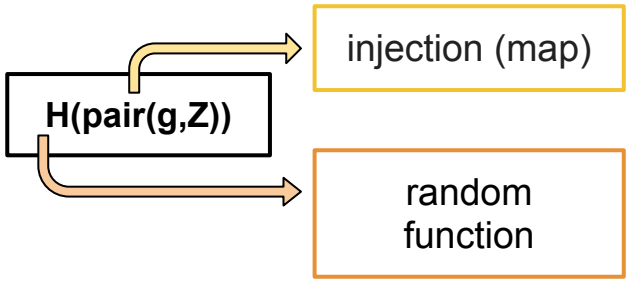
SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMH(Z)

If ∃ Y_i s.t. pair(X,Y_i) = pair(g, Z):
    Win with Z
Return H(pair(g,Z))

Assumes a pairing friendly group



Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMENC

Return Y_i, $H'(Y_i)=H(pair(X,Y_i))$

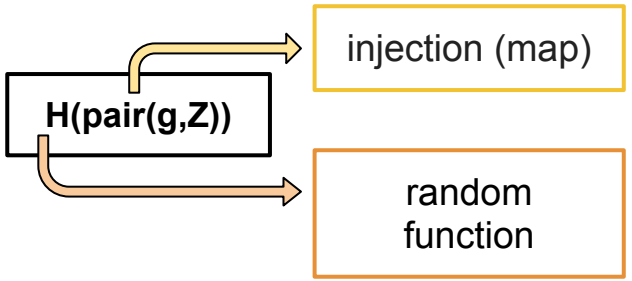
SIMDEC(Y)

Return $H'(Y)=H(pair(X,Y))$

SIMH(Z)

If ∃ Y_i s.t. $pair(X,Y_i) = pair(g, Z)$:
    Win with Z
Return H(pair(g,Z))

Assumes a pairing friendly group



Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMENC

Return Y_i, $H'(Y_i)=H(pair(X,Y_i))$

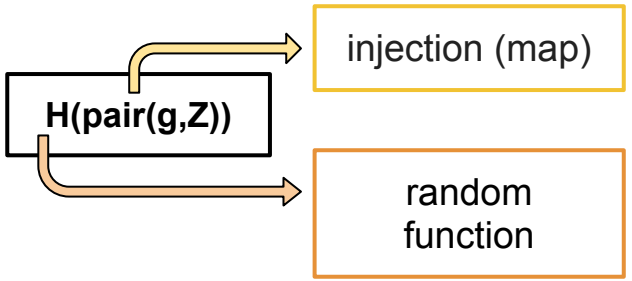
SIMDEC(Y)

Return $H'(Y)=H(pair(X,Y))$

SIMH(Z)

If ∃ Y_i s.t. $pair(X,Y_i) = pair(g, Z)$:
    Win with Z
Return H(pair(g,Z))

Assumes a pairing friendly group



Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMENC

Return Y_i, $H'(Y_i)=H(pair(X,Y_i))$

SIMDEC(Y)

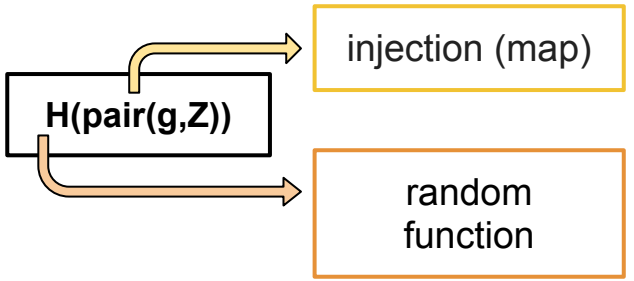
Return $H'(Y)=H(pair(X,Y))$

SIMH(Z)

If ∃ Y_i s.t. $pair(X,Y_i) = pair(g, Z)$:
    Win with Z
Return H(pair(g,Z))

pair(X,Y) = pair(g, Z) iff STRONG(Y, Z)

Assumes a pairing friendly group



Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMENC

Return Y_i, $H'(Y_i)=H(pair(X,Y_i))$

SIMDEC(Y)

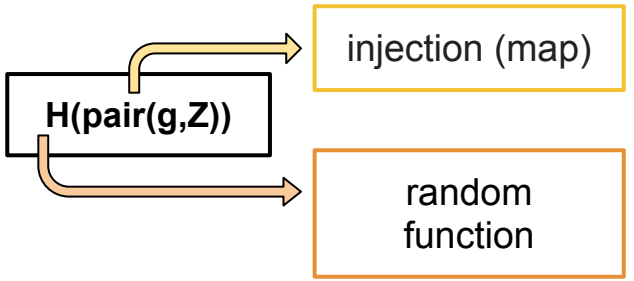
Return $H'(Y)=H(pair(X,Y))$

SIMH(Z)

If ∃ Y_i s.t. $pair(X,Y_i) = pair(g, Z)$:
    Win with Z
Return H(pair(g,Z))

pair(X,Y) = pair(g, Z) iff STRONG(Y, Z)

Assumes a pairing friendly group



Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMENC

Return Y_i, $H'(Y_i)=H(pair(X,Y_i))$

SIMDEC(Y)

Return $H'(Y)=H(pair(X,Y))$

SIMH(Z)

If ∃ Y_i s.t. $pair(X,Y_i) = pair(g, Z)$:
    Win with Z
Return H(pair(g,Z))

pair(X,Y) = pair(g, Z) iff STRONG(Y, Z)

Assumes a pairing friendly group

Reduction checks ALL challenge ciphertexts for SIMH 😕

H(pair(g,Z))

injection (map)

random function

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Y_i, $H'(Y_i)$

SIMDEC(Y)

If ∃ HASH(Z) s.t. STRONG(Y, Z):
    Return H(Z)
Return H'(Y)

SIMH(Z)

If ∃ Y_i s.t. STRONG(Y_i, Z):
    Win with Z
If ∃ SIMDEC(Y) s.t. STRONG(Y, Z):
    Return H'(Y)
Return H(Z)

SIMENC

Return Y_i, $H'(Y_i)=H(pair(X,Y_i))$

SIMDEC(Y)

Return $H'(Y)=H(pair(X,Y))$

SIMH(Z)

If ∃ Y_i s.t. $pair(X,Y_i) = pair(g, Z)$:
    Win with Z
Return H(pair(g,Z))

pair(X,Y) = pair(g, Z) iff STRONG(Y, Z)

Assumes a pairing friendly group 😕

Reduction checks ALL challenge ciphertexts for SIMH 😕

H(pair(g,Z))

injection (map)

random function

Augmented Construction

Augmented Construction

aECIES

aECIES.K

$x \leftarrow_{\$} \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return (ek, dk)

aECIES.E$^{\mathcal{H}}(ek)$

$a \leftarrow_{\$} \{0,1\}^l$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(a, Z)$
Return $((a, Y), K)$

aECIES.D$^{\mathcal{H}}(dk, (a, Y))$

$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(a, Z)$
Return K

Augmented Construction

aECIES

aECIES.K

$x \leftarrow_{\$} \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return (ek, dk)

aECIES.E$^{\mathcal{H}}(ek)$

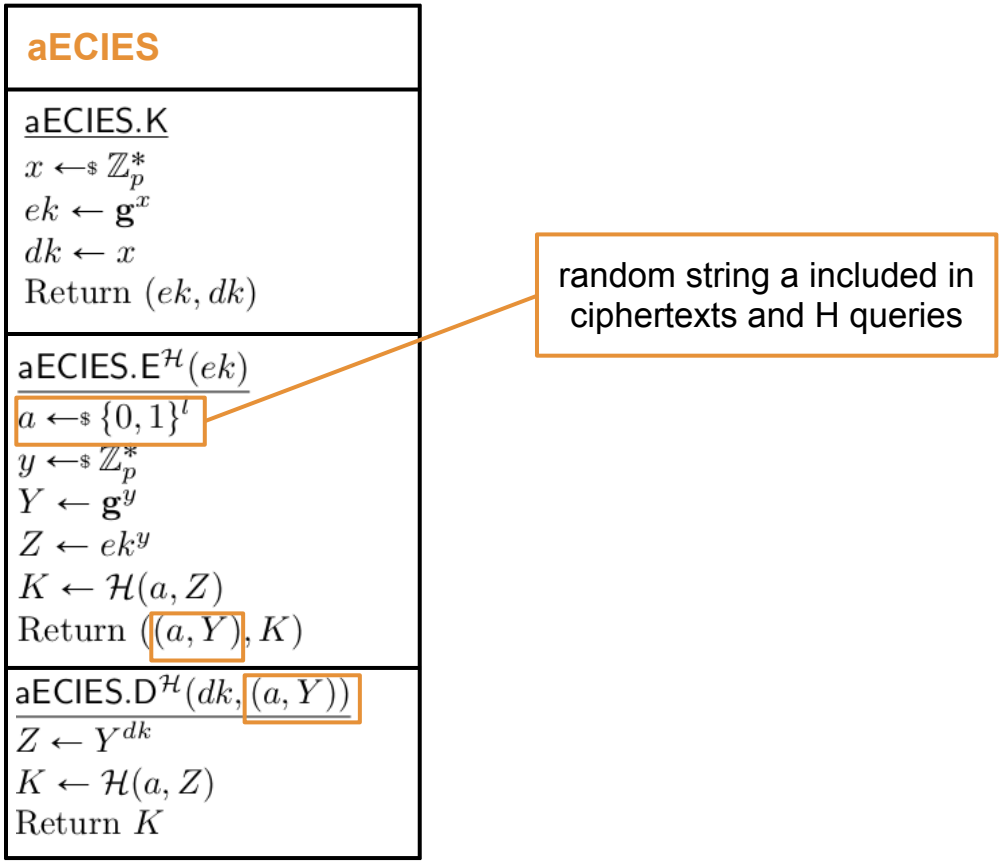
$a \leftarrow_{\$} \{0,1\}^l$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(a, Z)$
Return $((a, Y), K)$

aECIES.D$^{\mathcal{H}}(dk, (a, Y))$

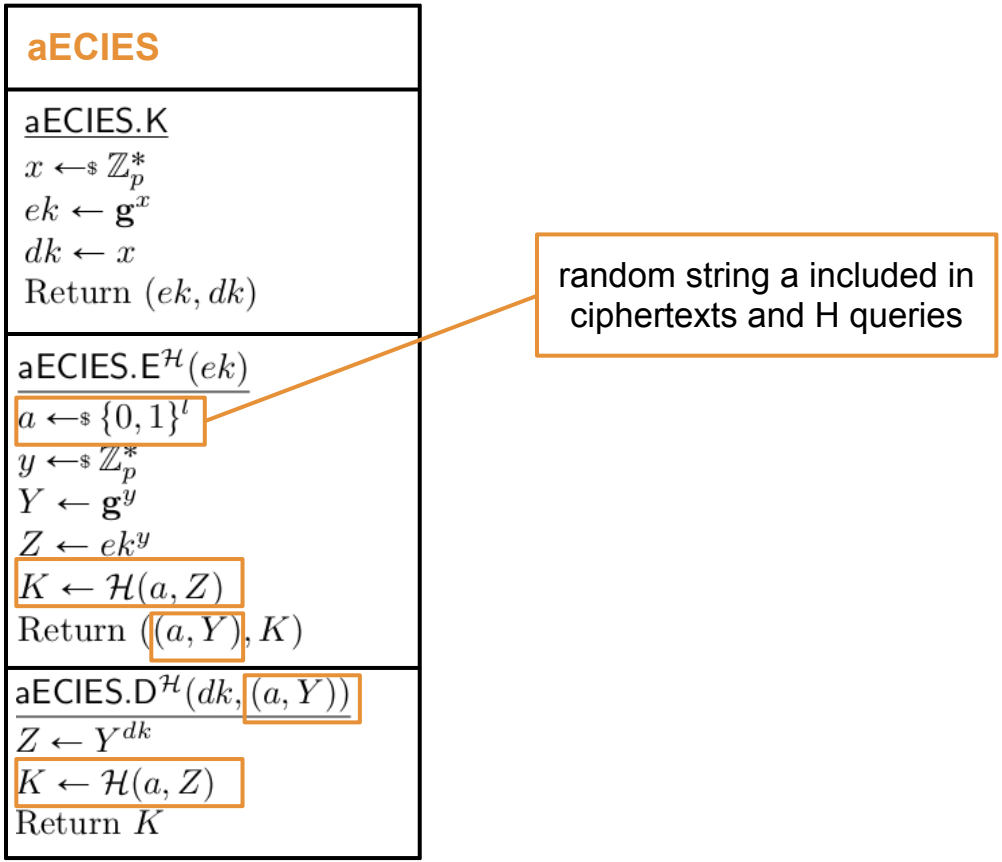
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(a, Z)$
Return K

random string a included in ciphertexts and H queries

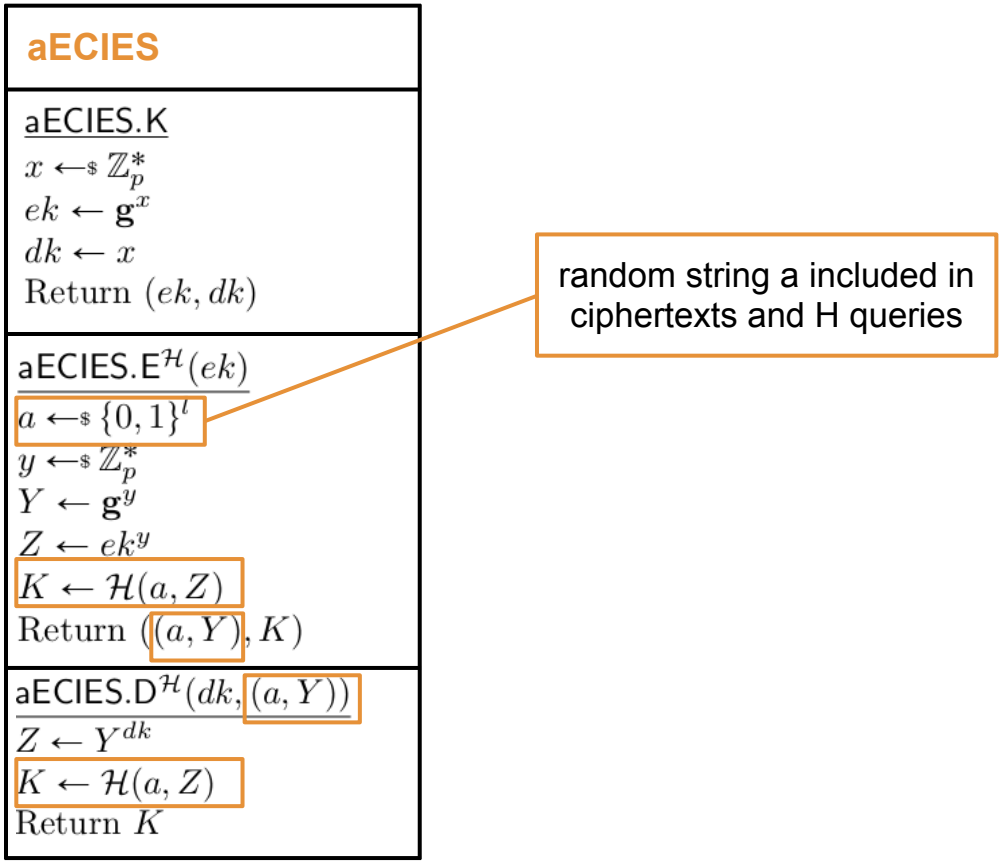
Augmented Construction



Augmented Construction



Augmented Construction



random string a included in ciphertexts and H queries

In proof **a_i = encode(i)** [message-encoding GGJT22]

Augmented Construction

aECIES

aECIES.K

$x \leftarrow_\$ \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return (ek, dk)

aECIES.E$^{\mathcal{H}}(ek)$

$a \leftarrow_\$ \{0,1\}^l$
$y \leftarrow_\$ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(a, Z)$
Return $((a, Y), K)$

aECIES.D$^{\mathcal{H}}(dk, (a, Y))$

$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(a, Z)$
Return K

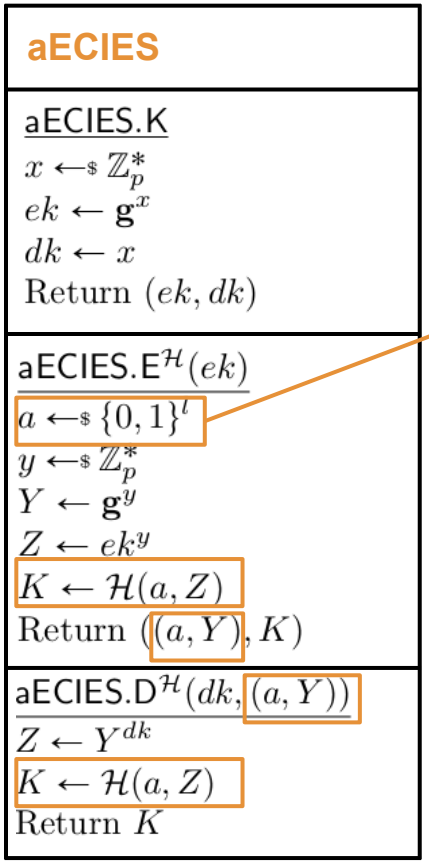
random string a included in ciphertexts and H queries

In proof **a_i = encode(i)** [message-encoding GGJT22]

SIMENC

a_i ←encode(i)
Return (a_i, Y_i), H'(a_i,Y_i)=H(a_i, pair(X,Y_i))

Augmented Construction



random string a included in ciphertexts and H queries

In proof **a_i = encode(i)** [message-encoding GGJT22]

SIMENC

a_i ←encode(i)
Return (**a_i**, Y_i), H'(**a_i**,Y_i)=H(**a_i**, pair(X,Y_i))

SIMDEC(a, Y)

Return H'(**a**, Y)=H(**a**, pair(X,Y))

Augmented Construction

aECIES

aECIES.K

$x \leftarrow\$ \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return (ek, dk)

aECIES.E$^{\mathcal{H}}(ek)$

$a \leftarrow\$ \{0,1\}^l$
$y \leftarrow\$ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(a, Z)$
Return $((a, Y), K)$

aECIES.D$^{\mathcal{H}}(dk, (a, Y))$

$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(a, Z)$
Return K

random string a included in ciphertexts and H queries

In proof **a_i = encode(i)** [message-encoding GGJT22]

SIMENC

a_i ←encode(i)
Return (**a_i**, Y_i), H'(**a_i**,Y_i)=H(**a_i**, pair(X,Y_i))

SIMDEC(a, Y)

Return H'(**a**, Y)=H(**a**, pair(X,Y))

SIMH(a, Z)

i ←decode(a)
If pair(X,Y_i) = pair(g, Z):
    Win with Z
Return H(**a**, pair(g,Z))

Augmented Construction

aECIES

aECIES.K

$x \leftarrow\$ \mathbb{Z}_p^*$

$ek \leftarrow \mathbf{g}^x$

$dk \leftarrow x$

Return (ek, dk)

aECIES.$\mathsf{E}^{\mathcal{H}}(ek)$

$a \leftarrow\$ \{0,1\}^l$

$y \leftarrow\$ \mathbb{Z}_p^*$

$Y \leftarrow \mathbf{g}^y$

$Z \leftarrow ek^y$

$K \leftarrow \mathcal{H}(a, Z)$

Return $((a, Y), K)$

aECIES.$\mathsf{D}^{\mathcal{H}}(dk, (a, Y))$

$Z \leftarrow Y^{dk}$

$K \leftarrow \mathcal{H}(a, Z)$

Return K

random string a included in ciphertexts and H queries

In proof **a_i = encode(i)** [message-encoding GGJT22]

SIMENC

a_i ←encode(i)

Return (**a_i**, Y_i), H'(**a_i**,Y_i)=H(**a_i**, pair(X,Y_i))

SIMDEC(a, Y)

Return H'(**a**, Y)=H(**a**, pair(X,Y))

SIMH(a, Z)

i ←decode(a)

If pair(X,Y_i) = pair(g, Z):

Win with Z

Return H(**a**, pair(g,Z))

Augmented Construction

aECIES

aECIES.K

$x \leftarrow\$ \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return (ek, dk)

aECIES.$\mathsf{E}^{\mathcal{H}}(ek)$

$a \leftarrow\$ \{0,1\}^l$
$y \leftarrow\$ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(a, Z)$
Return $((a, Y), K)$

aECIES.$\mathsf{D}^{\mathcal{H}}(dk, (a, Y))$

$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(a, Z)$
Return K

random string a included in ciphertexts and H queries

Check one ciphertext in SIMH 🙂

TAM-Tight ✅

In proof **a_i = encode(i)** [message-encoding GGJT22]

SIMENC

a_i ←encode(i)
Return (a_i, Y_i), H'(a_i,Y_i)=H(a_i, pair(X,Y_i))

SIMDEC(a, Y)

Return H'(**a**, Y)=H(**a**, pair(X,Y))

SIMH(a, Z)

i ←decode(a)
If pair(X,Y_i) = pair(g, Z):
Win with Z
Return H(**a**, pair(g,Z))

Extensions

Extensions

➔ **Groups without pairings - The Pair CDH Assumption**

Extensions

➔ **Groups without pairings - The Pair CDH Assumption**

Given $X, Y_1, Y_2, Y_3, ..., Y_n$, find $\mathbf{g}^{xy_i}$ for some $i \in [1, n]$
where $x = \text{dlog}(X), y_i = \text{dlog}(Y_i)$.

Extensions

➔ Groups without pairings - The Pair CDH Assumption

Given $X, Y_1, Y_2, Y_3, ..., Y_n$, find $\mathbf{g}^{xy_i}$ for some $i \in [1, n]$ where $x = \text{dlog}(X), y_i = \text{dlog}(Y_i)$.

$\underline{\text{PAIR}(A, B)}$
$a \leftarrow \text{dlog}(A);\ b \leftarrow \text{dlog}(B)$
Return $f(ab)$

Extensions

➔ Groups without pairings - The Pair CDH Assumption

Given $X, Y_1, Y_2, Y_3, ..., Y_n$, find $\mathbf{g}^{xy_i}$ for some $i \in [1, n]$ where $x = \text{dlog}(X), y_i = \text{dlog}(Y_i)$.

$\underline{\text{PAIR}(A, B)}$
$a \leftarrow \text{dlog}(A);\ b \leftarrow \text{dlog}(B)$
Return $f(ab)$

- ★ New CDH variant
- ★ Attacker is given oracle access to a pairing to a “random group”
- ★ Heuristic justification: Showed hard in Algebraic / Generic Group model

Extensions

➔ Groups without pairings - The Pair CDH Assumption

Given $X, Y_1, Y_2, Y_3, ..., Y_n$, find $\mathbf{g}^{xy_i}$ for some $i \in [1, n]$
where $x = \text{dlog}(X), y_i = \text{dlog}(Y_i)$.

$\underline{\text{PAIR}(A, B)}$
$a \leftarrow \text{dlog}(A);\ b \leftarrow \text{dlog}(B)$
Return $f(ab)$

- ★ New CDH variant
- ★ Attacker is given oracle access to a pairing to a “random group”
- ★ Heuristic justification: Showed hard in Algebraic / Generic Group model

➔ Multi-user

Extensions

➔ Groups without pairings - The Pair CDH Assumption

Given $X, Y_1, Y_2, Y_3, ..., Y_n$, find $\mathbf{g}^{xy_i}$ for some $i \in [1, n]$
where $x = \text{dlog}(X), y_i = \text{dlog}(Y_i)$.

$\underline{\text{PAIR}(A, B)}$
$a \leftarrow \text{dlog}(A);\ b \leftarrow \text{dlog}(B)$
Return $f(ab)$

★ New CDH variant
★ Attacker is given oracle access to a pairing to a “random group”
★ Heuristic justification: Showed hard in Algebraic / Generic Group model

➔ Multi-user

a_i = encode(i) → a_i = encode(u,i)

Cramer-Shoup

Cramer-Shoup

$\underline{\mathsf{KEM.E}^{\mathcal{H}}(ek)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(Y, Z)$
Return (Y, K)

$\underline{\mathsf{KEM.D}^{\mathcal{H}}(dk, Y)}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(Y, Z)$
Return K

Y_i, $H(Y_i, Y_i^x)$

ENC

SIMENC

Return Y_i, $H'(Y_i)$

Y

$H(Y,Y^x)$

DEC

Y,Z

H(Y,Z)

H

SIMH(Y,Z)

Return H(Y,Z)

SIMDEC(Y)

Return H'(Y)

$X, Y_1, Y_2, \ldots, Y_n$

REDUCTION R

Z

Reduction Goals

- ➔ ~~Identify challenge ciphertexts in **SIMDEC**~~
- ➔ Identify challenge secrets in **SIMH**
- ➔ Consistent outputs from **SIMDEC**, **SIMH**

CS Standard Proof - Not memory/time-tight

SIMENC
Return Yi, H'(Y_i)

SIMDEC(Y)
Return H'(Y)

SIMH(Y,Z)
Return H(Y,Z)

SIMENC
Return Yi, H'(Yi)

SIMDEC(Y)
If ∃ HASH(Y, Z) s.t. STRONG(Y, Z): Return H(Y, Z) Return H'(Y)

SIMH(Y, Z)
If ∃ Y_i=Y STRONG(Y, Z): Win with Z If ∃ DEC(Y) Λ STRONG(Y, Z): Return H'(Y) Return H(Y, Z)

$$\frac{\textsc{Strong}(Y, Z)}{y \leftarrow \text{dlog}(A)}$$
$$\text{Return } (Z = X^y)$$

Reduction Goals

- ➔ ~~Identify challenge ciphertexts in **SIMDEC**~~
- ➔ ~~Identify challenge secrets in **SIMH**~~
- ➔ ~~Consistent outputs from **SIMDEC**, **SIMH**~~

CS Standard Proof - Not memory/time-tight

SIMENC

Return Yi, H'(Y_i)

SIMDEC(Y)

Return H'(Y)

SIMH(Y,Z)

Return H(Y,Z)

SIMENC

Return Yi, H'(Yi)

SIMDEC(Y)

If ∃ HASH(Y, Z) s.t. STRONG(Y, Z):
    Return H(Y, Z)
Return H'(Y)

SIMH(Y, Z)

If ∃ Y_i=Y STRONG(Y, Z):
    Win with Z
If ∃ DEC(Y) ∧ STRONG(Y, Z):
    Return H'(Y)
Return H(Y, Z)

$$\frac{\textsc{Strong}(Y, Z)}{y \leftarrow \text{dlog}(A)}$$
$$\text{Return } (Z = X^y)$$

Reduction Goals

- ➔ ~~Identify challenge ciphertexts in **SIMDEC**~~
- ➔ ~~Identify challenge secrets in **SIMH**~~
- ➔ ~~Consistent outputs from **SIMDEC**, **SIMH**~~

CS Standard Proof - Not memory/time-tight

SIMENC

Return Yi, H'(Y_i)

SIMDEC(Y)

Return H'(Y)

SIMH(Y,Z)

Return H(Y,Z)

SIMENC

Return Yi, H'(Yi)

SIMDEC(Y)

If ∃ HASH(Y, Z) s.t. STRONG(Y, Z):
    Return H(Y, Z)
Return H'(Y)

SIMH(Y, Z)

If ∃ Y_i=Y STRONG(Y, Z):
    Win with Z
If ∃ DEC(Y) ∧ STRONG(Y, Z):
    Return H'(Y)
Return H(Y, Z)

Not time/memory tight 🙁

$$\frac{\textsc{Strong}(Y, Z)}{y \leftarrow \text{dlog}(A)}$$
$$\text{Return } (Z = X^y)$$

Reduction Goals

- ➔ ~~Identify challenge ciphertexts in **SIMDEC**~~
- ➔ ~~Identify challenge secrets in **SIMH**~~
- ➔ ~~Consistent outputs from **SIMDEC**, **SIMH**~~

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Yi, H'(Yi)

SIMDEC(Y)

If ∃ HASH(Y, Z) s.t. STRONG(Y, Z):
    Return H(Y, Z)
Return H'(Y)

SIMH(Y, Z)

If ∃ Y_i=Y STRONG(Y, Z):
    Win with Z
If ∃ DEC(Y) Λ STRONG(Y, Z):
    Return H'(Y)
Return H(Y, Z)

SIMENC

Return Y_i, H'(Y_i)=H(Y_i,★)

SIMDEC(Y)

Return H'(Y)=H(Y, ★)

SIMH(Y,Z)

If ∃ Y_i=Y AND STRONG(Y, Z):
    Win with Z
If STRONG(Y,Z):
    Return H(Y, ★)
Return H(Y,Z)

H(Y,Z) = H(Y, ★) if STRONG(Y,Z);
H(Y, Z) otherwise

injection (map)

hash function

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Yi, H'(Yi)

SIMDEC(Y)

If ∃ HASH(Y, Z) s.t. STRONG(Y, Z):
    Return H(Y, Z)
Return H'(Y)

SIMH(Y, Z)

If ∃ Y_i=Y STRONG(Y, Z):
    Win with Z
If ∃ DEC(Y) Λ STRONG(Y, Z):
    Return H'(Y)
Return H(Y, Z)

SIMENC

Return Y_i, H'(Y_i)=H(Y_i,★)

SIMDEC(Y)

Return H'(Y)=H(Y, ★)

SIMH(Y,Z)

If ∃ Y_i=Y AND STRONG(Y, Z):
    Win with Z
If STRONG(Y,Z):
    Return H(Y, ★)
Return H(Y,Z)

H(Y,Z) = H(Y, ★) if STRONG(Y,Z);
H(Y, Z) otherwise

injection (map)

hash function

Reduction checks ALL challenge ciphertexts for SIMH 🙁

Improvement - map-then-hash - [Bhattacharya20]

SIMENC

Return Yi, H'(Yi)

SIMDEC(Y)

If ∃ HASH(Y, Z) s.t. STRONG(Y, Z):
    Return H(Y, Z)
Return H'(Y)

SIMH(Y, Z)

If ∃ Y_i=Y STRONG(Y, Z):
    Win with Z
If ∃ DEC(Y) Λ STRONG(Y, Z):
    Return H'(Y)
Return H(Y, Z)

SIMENC

Return Y_i, H'(Y_i)=H(Y_i,★)

SIMDEC(Y)

Return H'(Y)=H(Y, ★)

SIMH(Y,Z)

If ∃ Y_i=Y AND STRONG(Y, Z):
    Win with Z
If STRONG(Y,Z):
    Return H(Y, ★)
Return H(Y,Z)

H(Y,Z) = H(Y, ★) if STRONG(Y,Z);
H(Y, Z) otherwise

injection (map)

hash function

Reduction checks ALL challenge ciphertexts for SIMH 😕

Solution - Augment the scheme!
Check one ciphertext in SIMH 🙂
TAM-Tight ✅

Extensions

Extensions

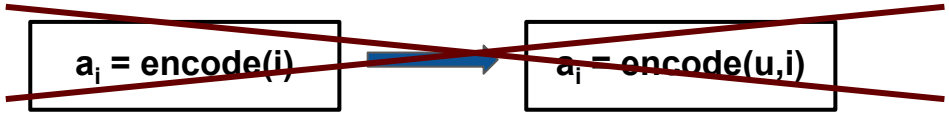
➔ **Multi-user**

Extensions

➔ **Multi-user**

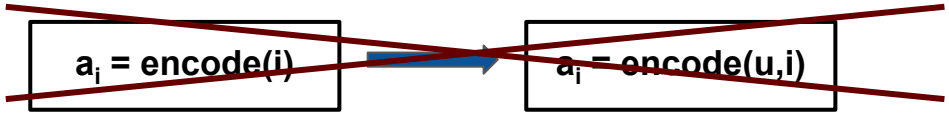
Extensions

➔ **Multi-user**



Extensions

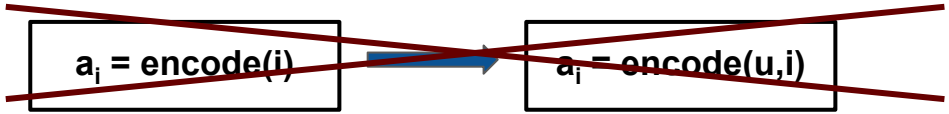
➔ **Multi-user**



★ Also augment encapsulation key with a random string α!

Extensions

➔ **Multi-user**

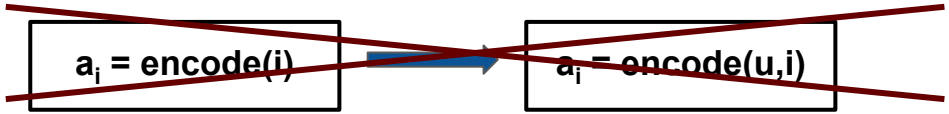


★ Also augment encapsulation key with a random string α!

$\underline{\mathsf{aCS.K}}$	$\underline{\mathsf{aCS.E}^{\mathcal{H}}((\alpha, X))}$	$\underline{\mathsf{aCS.D}^{\mathcal{H}}((\alpha, x), (a, Y))}$
$\alpha \leftarrow\!\!\$\ \{0,1\}^{l_1}$	$a \leftarrow\!\!\$\ \{0,1\}^{l_2}$	$Z \leftarrow Y^x$
$x \leftarrow\!\!\$\ \mathbb{Z}_p^*$	$y \leftarrow\!\!\$\ \mathbb{Z}_p^*$	$K \leftarrow \mathcal{H}(\alpha, a, Y, Z)$
$ek \leftarrow (\alpha, \mathbf{g}^x)$	$Y \leftarrow \mathbf{g}^y$	Return K
$dk \leftarrow (\alpha, x)$	$Z \leftarrow X^y$	
Return (ek, dk)	$K \leftarrow \mathcal{H}(\alpha, a, Y, Z)$	
	Return $((a, Y), K)$	

Extensions

➔ **Multi-user**



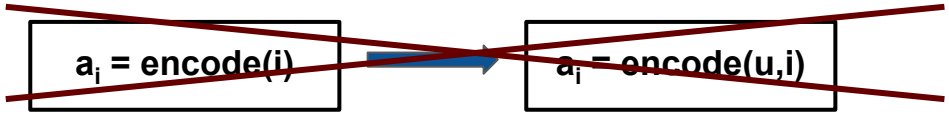
★ Also augment encapsulation key with a random string α!

$\underline{\mathsf{aCS.K}}$	$\underline{\mathsf{aCS.E}^{\mathcal{H}}((\alpha, X))}$	$\underline{\mathsf{aCS.D}^{\mathcal{H}}((\alpha, x), (a, Y))}$
$\alpha \leftarrow\!\!\$\ \{0,1\}^{l_1}$	$a \leftarrow\!\!\$\ \{0,1\}^{l_2}$	$Z \leftarrow Y^x$
$x \leftarrow\!\!\$\ \mathbb{Z}_p^*$	$y \leftarrow\!\!\$\ \mathbb{Z}_p^*$	$K \leftarrow \mathcal{H}(\alpha, a, Y, Z)$
$ek \leftarrow (\alpha, \mathbf{g}^x)$	$Y \leftarrow \mathbf{g}^y$	Return K
$dk \leftarrow (\alpha, x)$	$Z \leftarrow X^y$	
Return (ek, dk)	$K \leftarrow \mathcal{H}(\alpha, a, Y, Z)$	
	Return $((a, Y), K)$	

random string **α** included in ciphertexts and H queries

Extensions

➔ **Multi-user**



★ Also augment encapsulation key with a random string α!

$\underline{\mathsf{aCS.K}}$	$\underline{\mathsf{aCS.E}^{\mathcal{H}}((\alpha, X))}$	$\underline{\mathsf{aCS.D}^{\mathcal{H}}((\alpha, x), (a, Y))}$
$\alpha \leftarrow\!\!\$\ \{0,1\}^{l_1}$	$a \leftarrow\!\!\$\ \{0,1\}^{l_2}$	$Z \leftarrow Y^x$
$x \leftarrow\!\!\$\ \mathbb{Z}_p^*$	$y \leftarrow\!\!\$\ \mathbb{Z}_p^*$	$K \leftarrow \mathcal{H}(\alpha, a, Y, Z)$
$ek \leftarrow (\alpha, \mathbf{g}^x)$	$Y \leftarrow \mathbf{g}^y$	Return K
$dk \leftarrow (\alpha, x)$	$Z \leftarrow X^y$	
Return (ek, dk)	$K \leftarrow \mathcal{H}(\alpha, a, Y, Z)$	
	Return $((a, Y), K)$	

random string **α** included in ciphertexts and H queries

In proof **α = encode'(u), $a_{u,i}$ = encode_u(i)**

Summary

Hashed ElGamal

Fujisaki-Okamoto

Scheme S	Problem P
aECIES [AbdBelRog98] ✅	Pair CDH
aCramer-Shoup [CraSha01b] ✅	Strong CDH
aTwin ElGamal [CasKilSho09]	CDH
aFujisaki-Okamoto [FujOka13]	IND-CPA
PKE = KD[KEM,SKE]	IND-CCA(KEM), IND-CCA(SKE)

Summary

Hashed ElGamal

Fujisaki-Okamoto

Scheme S	Problem P
aECIES [AbdBelRog98] ✅	Pair CDH
aCramer-Shoup [CraSha01b] ✅	Strong CDH
aTwin ElGamal [CasKilSho09]	CDH
aFujisaki-Okamoto [FujOka13]	IND-CPA
PKE = KD[KEM,SKE]	IND-CCA(KEM), IND-CCA(SKE)

Thank You!
Questions?