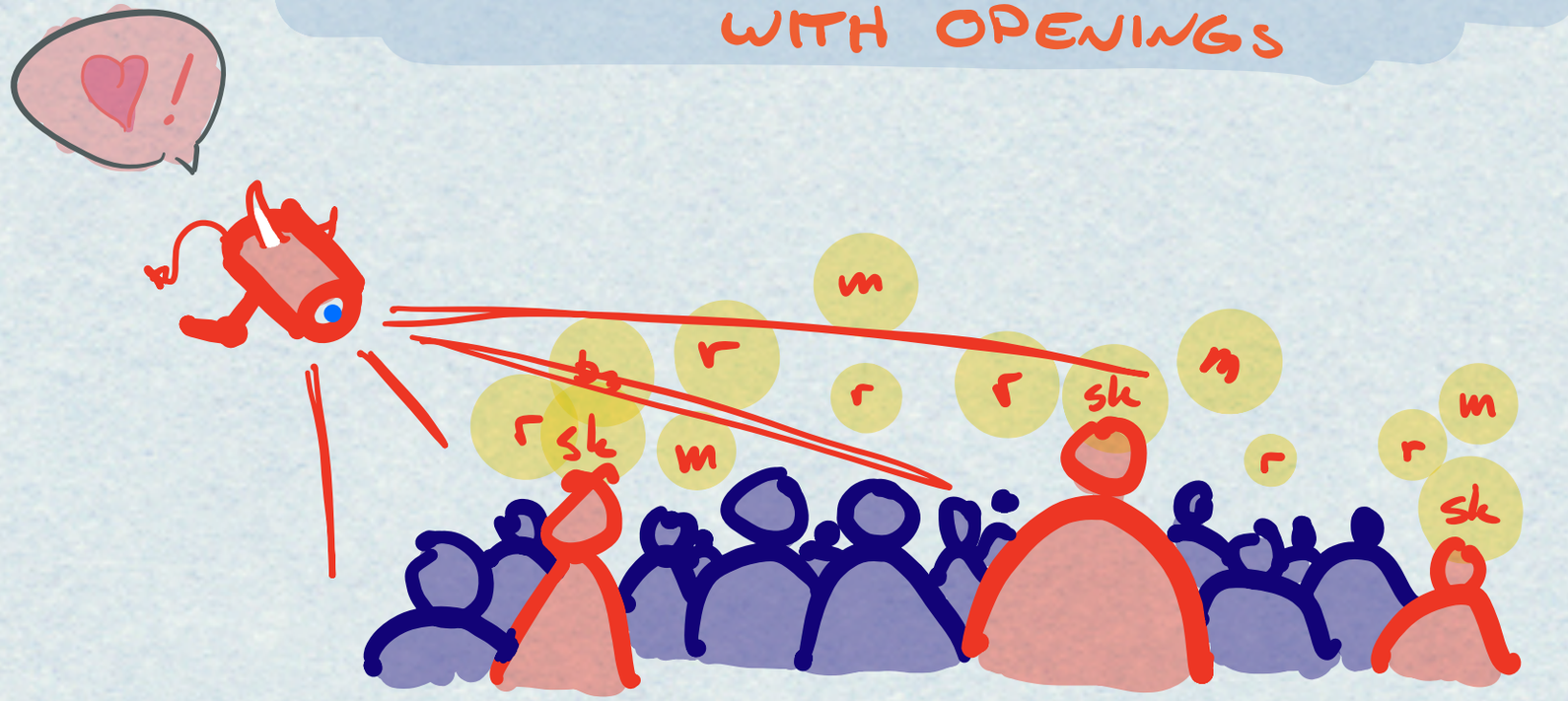
SoK: Public-Key Encryption with Openings



Hans Heum

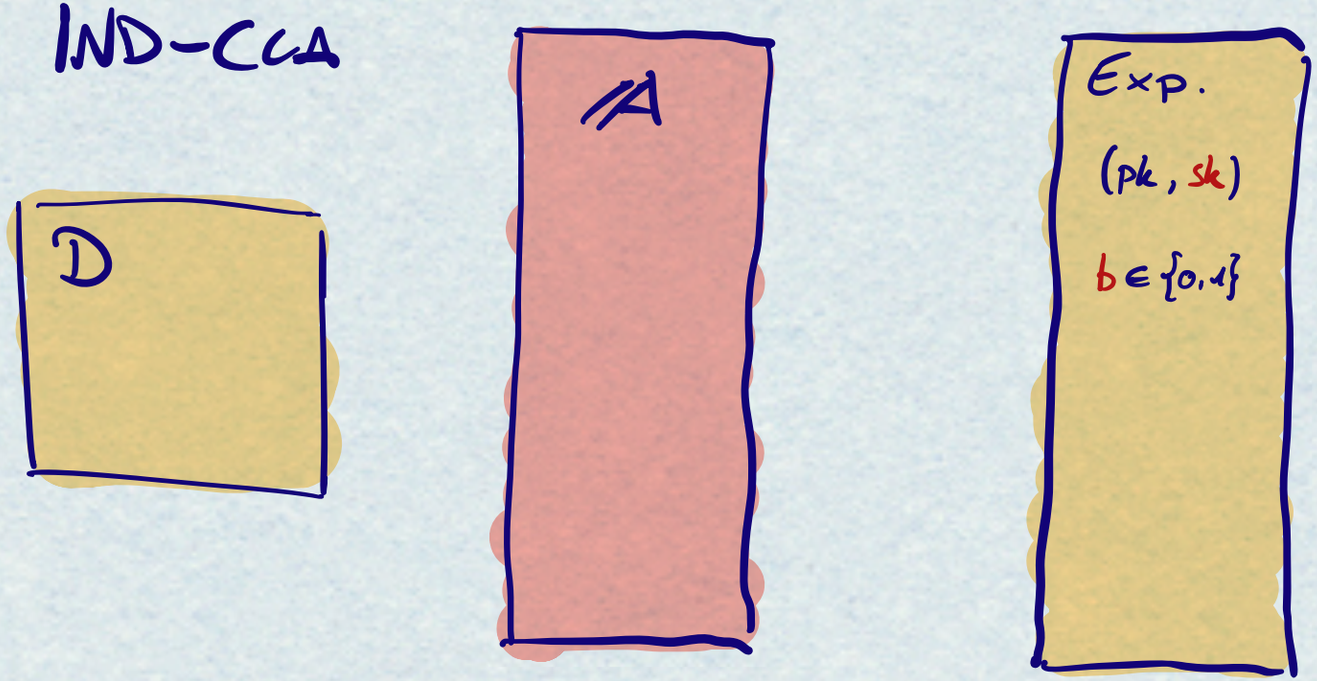
NTNU

with Carlo Brunetta and Martijn Stam

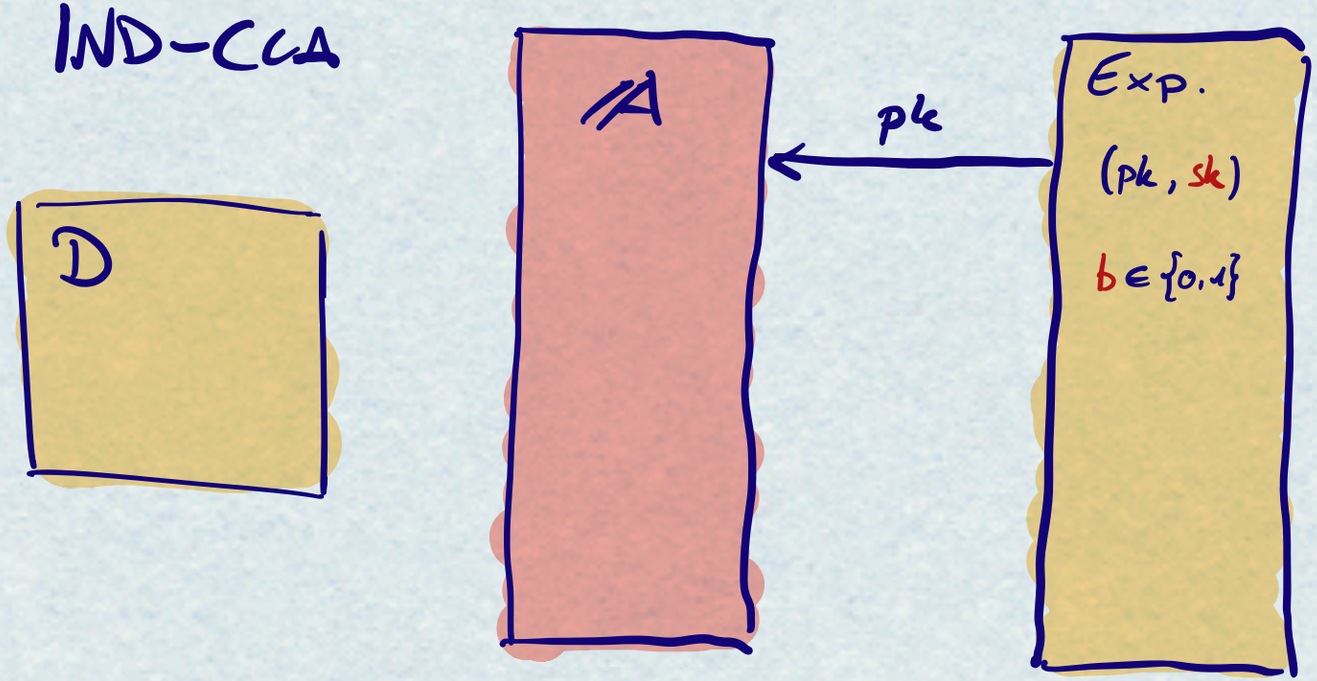
Simula UiB

Standard Security: Indistinguishability against Chosen-Ciphertext Attacks

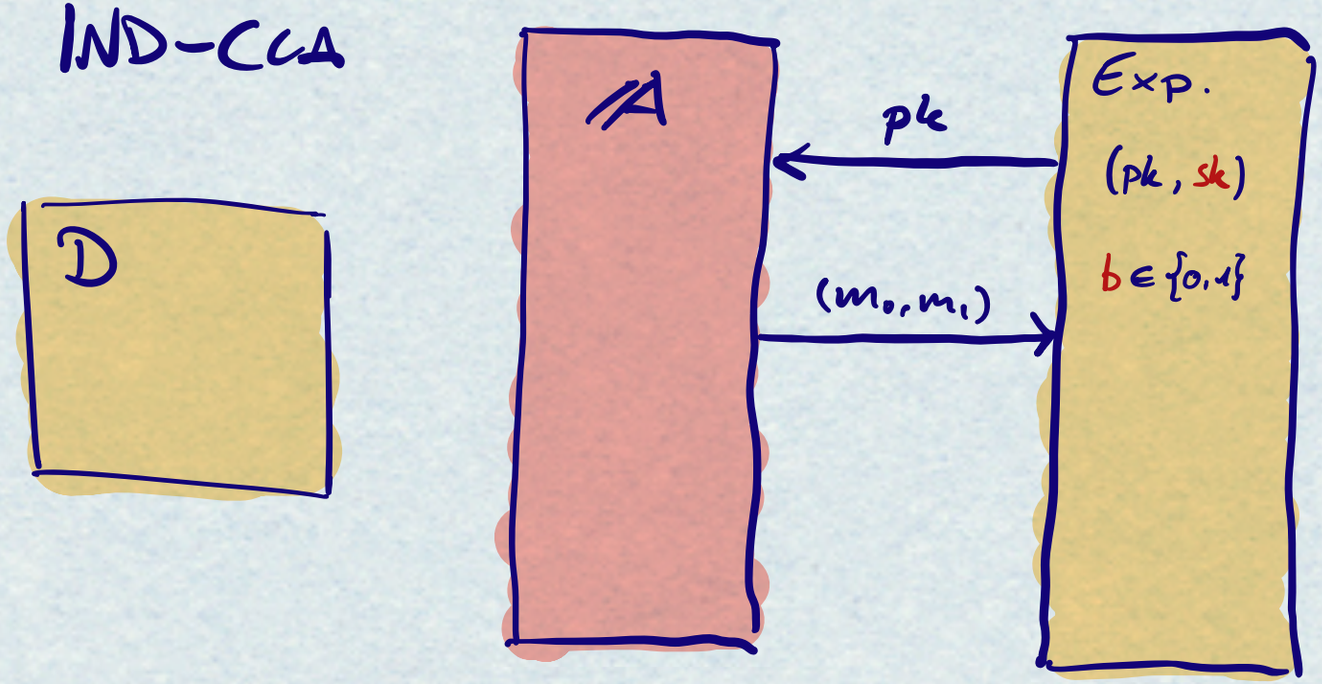
IND-CCA



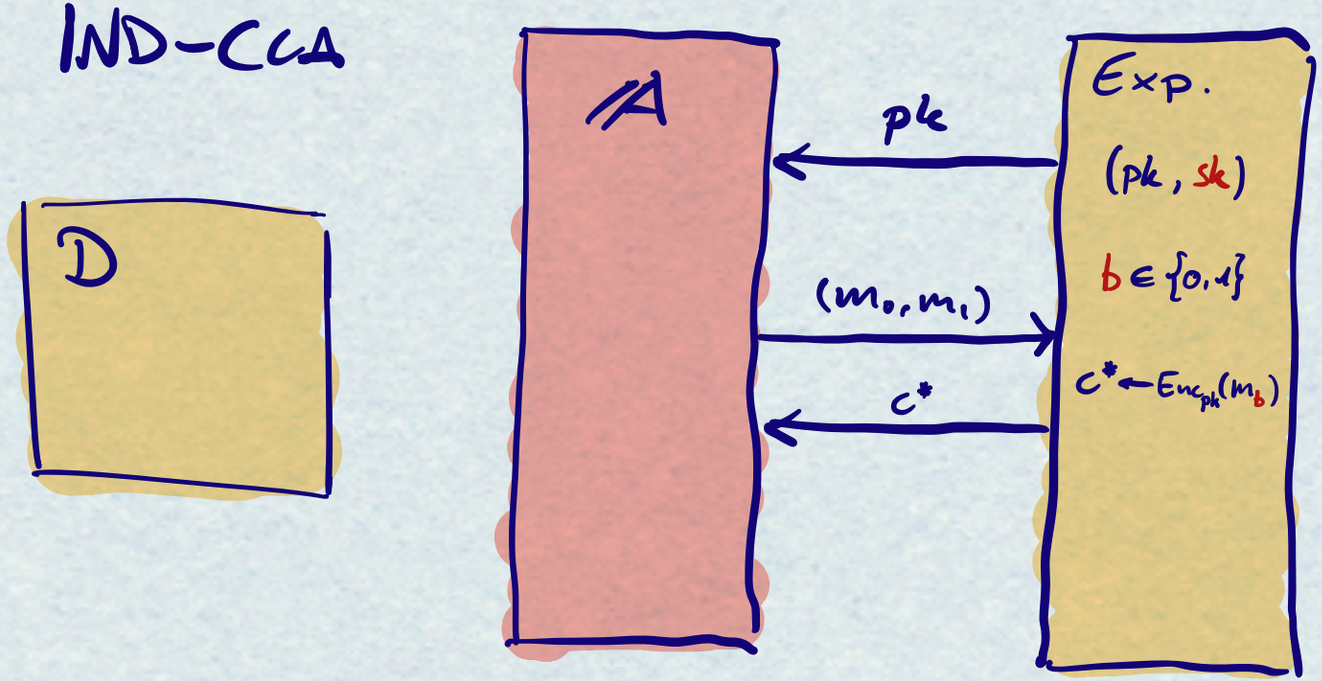
STANDARD SECURITY: INDISTINGUISHABILITY AGAINST CHOSEN-CIPHERTEXT ATTACKS



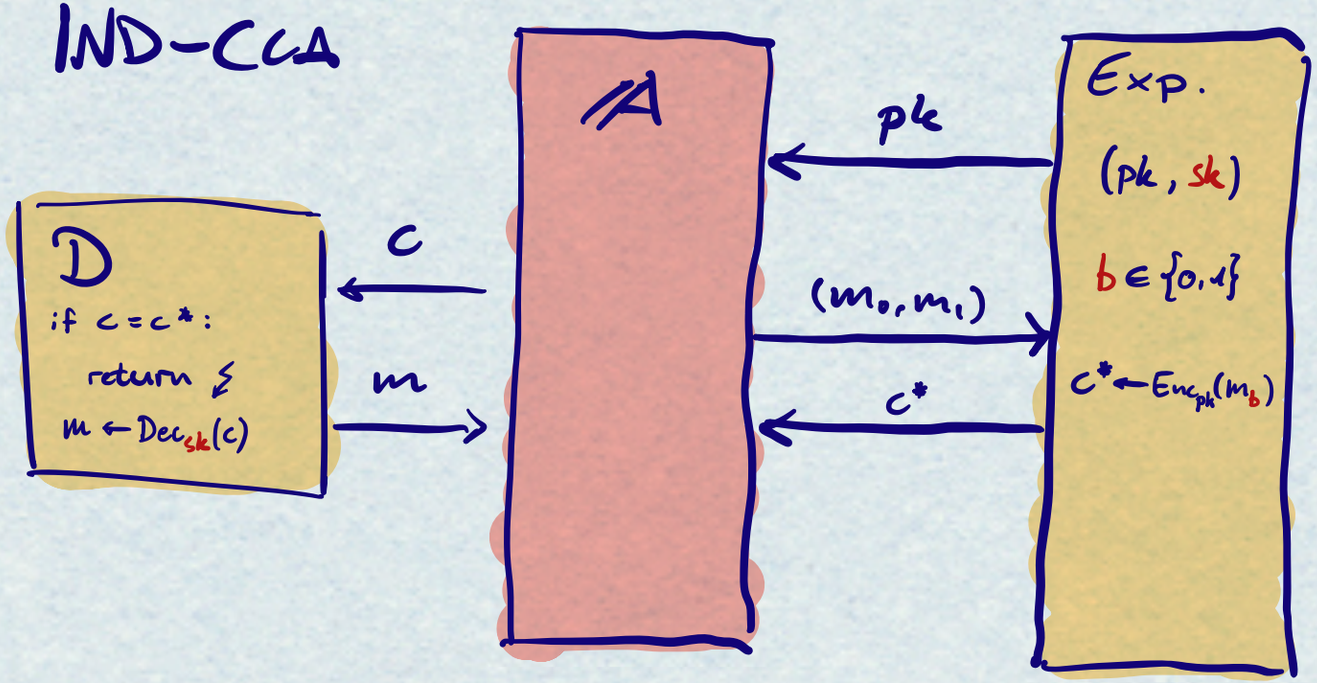
STANDARD SECURITY: INDISTINGUISHABILITY AGAINST CHOSEN-CIPHERTEXT ATTACKS



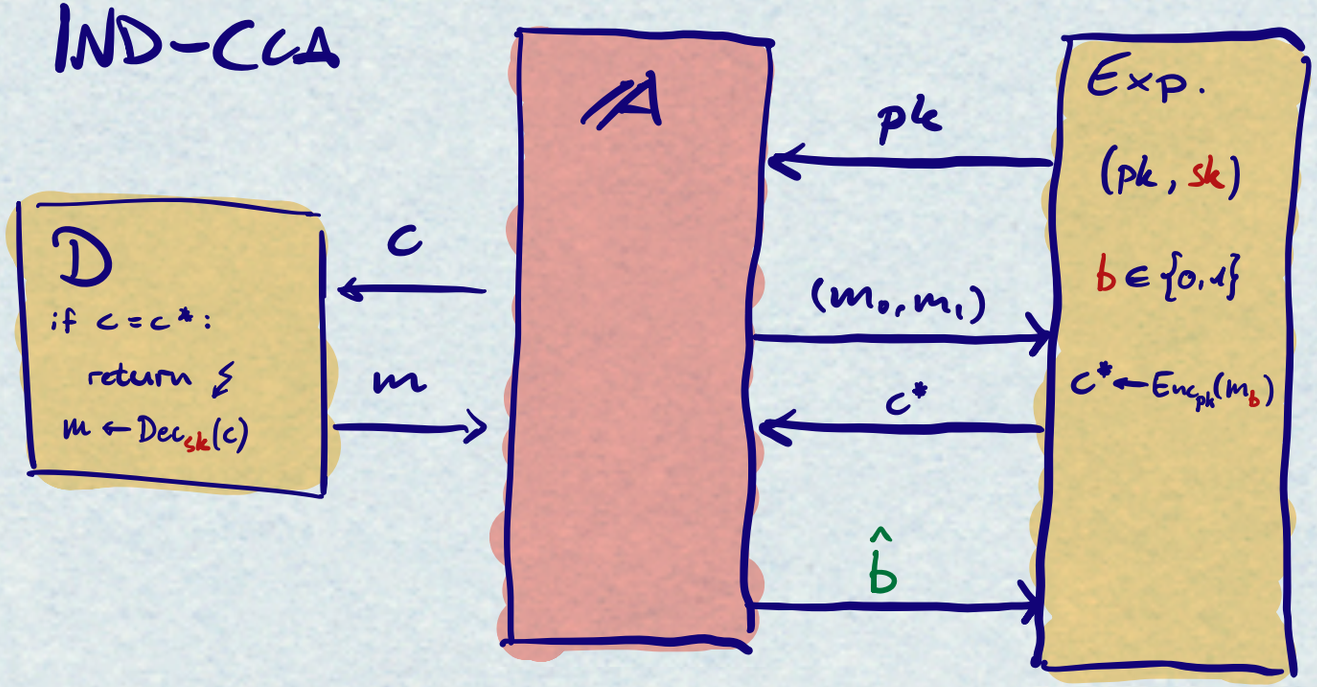
STANDARD SECURITY: INDISTINGUISHABILITY AGAINST CHOSEN-CIPHERTEXT ATTACKS



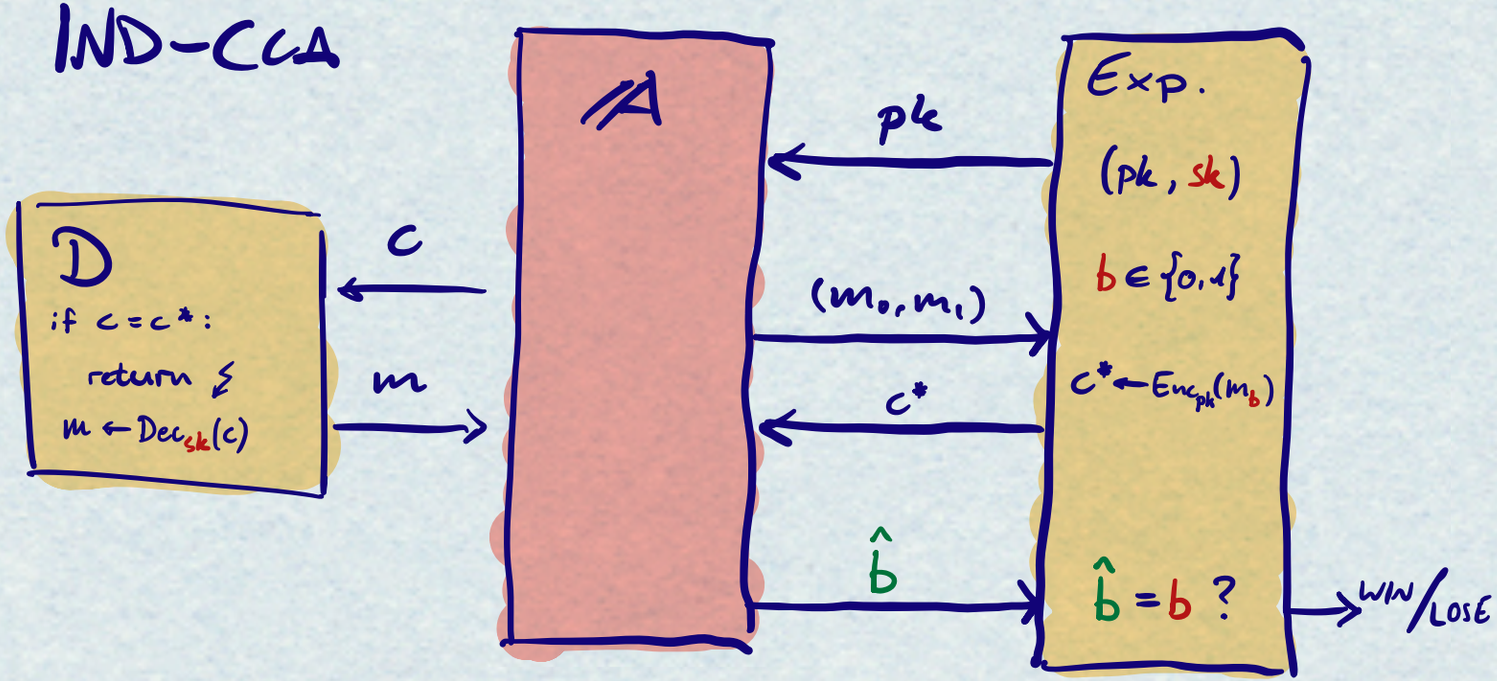
STANDARD SECURITY: INDISTINGUISHABILITY AGAINST CHOSEN-CIPHERTEXT ATTACKS



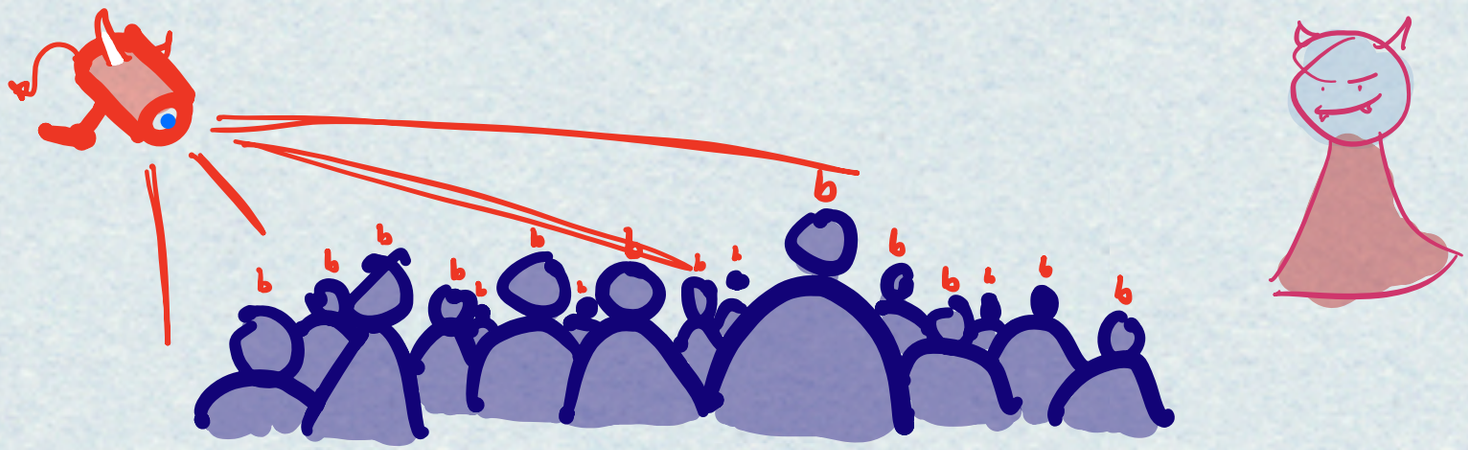
Standard security: Indistinguishability against Chosen-Ciphertext Attacks



STANDARD SECURITY: INDISTINGUISHABILITY AGAINST CHOSEN-CIPHERTEXT ATTACKS

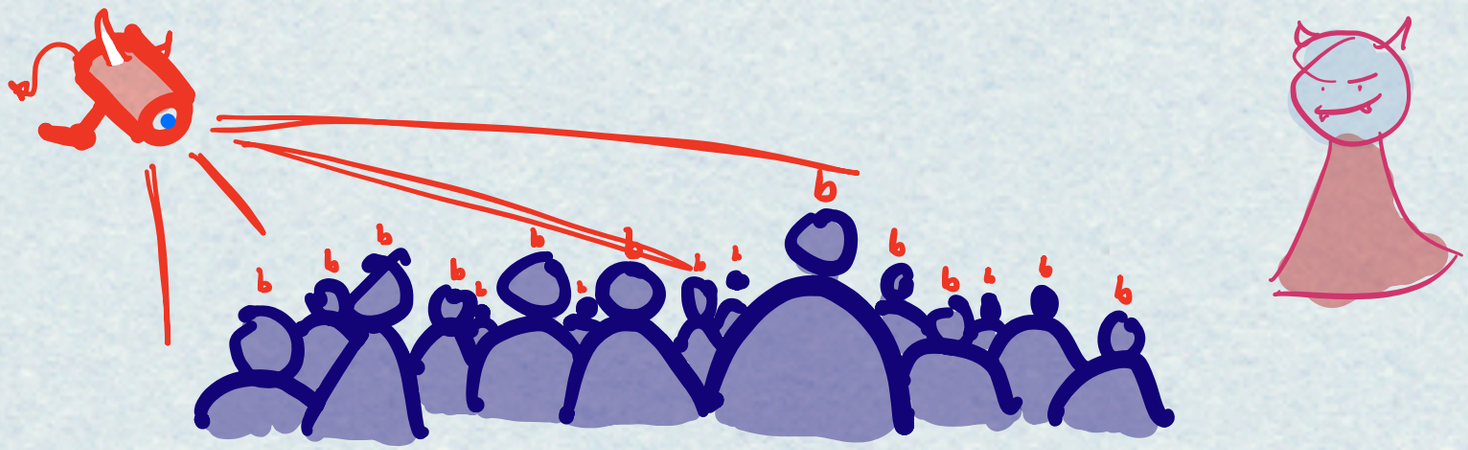


MULTI-USER SECURITY :



A CAN NOW ASK FOR SEVERAL CHALLENGES TO ANY OF n USERS. MORE REALISTIC!

MULTI-USER SECURITY :

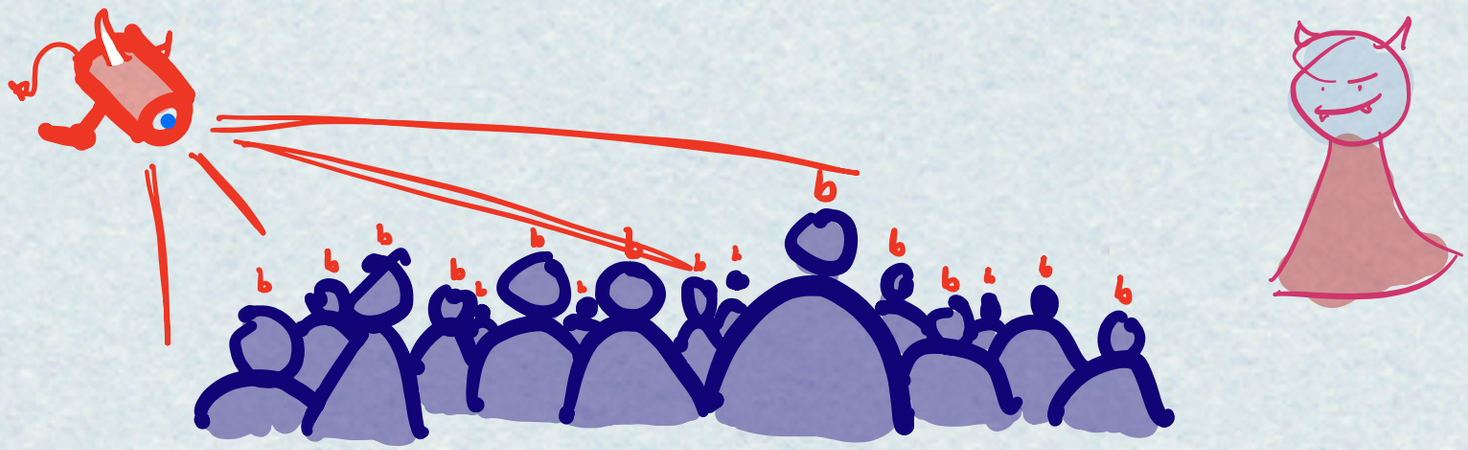


A CAN NOW ASK FOR SEVERAL CHALLENGES TO ANY OF n USERS. MORE REALISTIC!

THE GAME IS EASIER FOR A TO WIN

= STRONGER SECURITY MODEL.

MULTI-USER SECURITY :



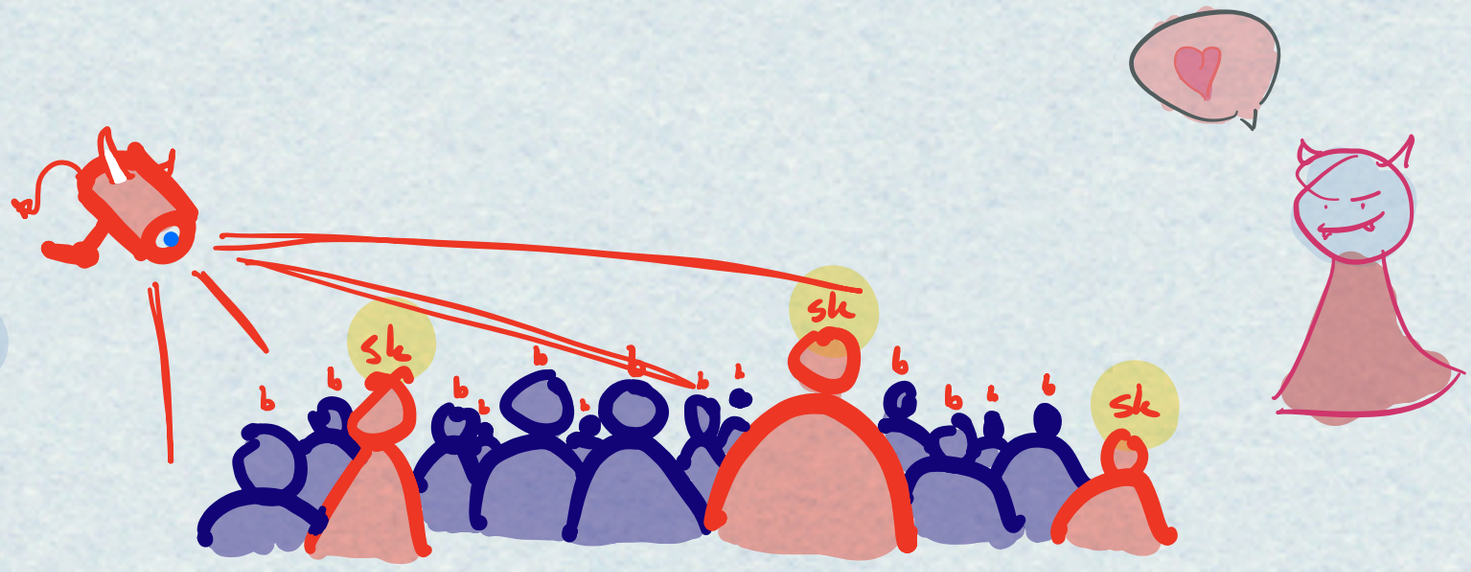
$\mathcal{A}$ CAN NOW ASK FOR SEVERAL CHALLENGES TO ANY OF n USERS. MORE REALISTIC!

THE GAME IS EASIER FOR $\mathcal{A}$ TO WIN

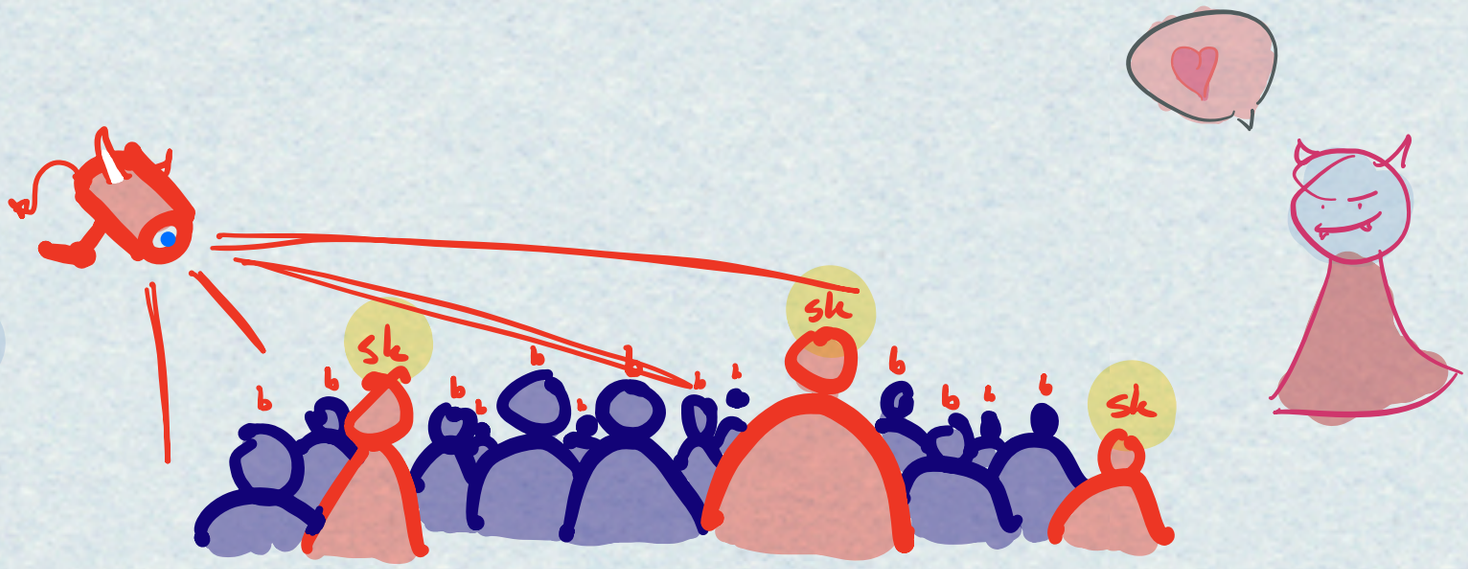
= STRONGER SECURITY MODEL.

$\mathcal{A}$ STILL ONLY NEEDS TO GUESS A SINGLE BIT!

Multi-User Security : With corruptions

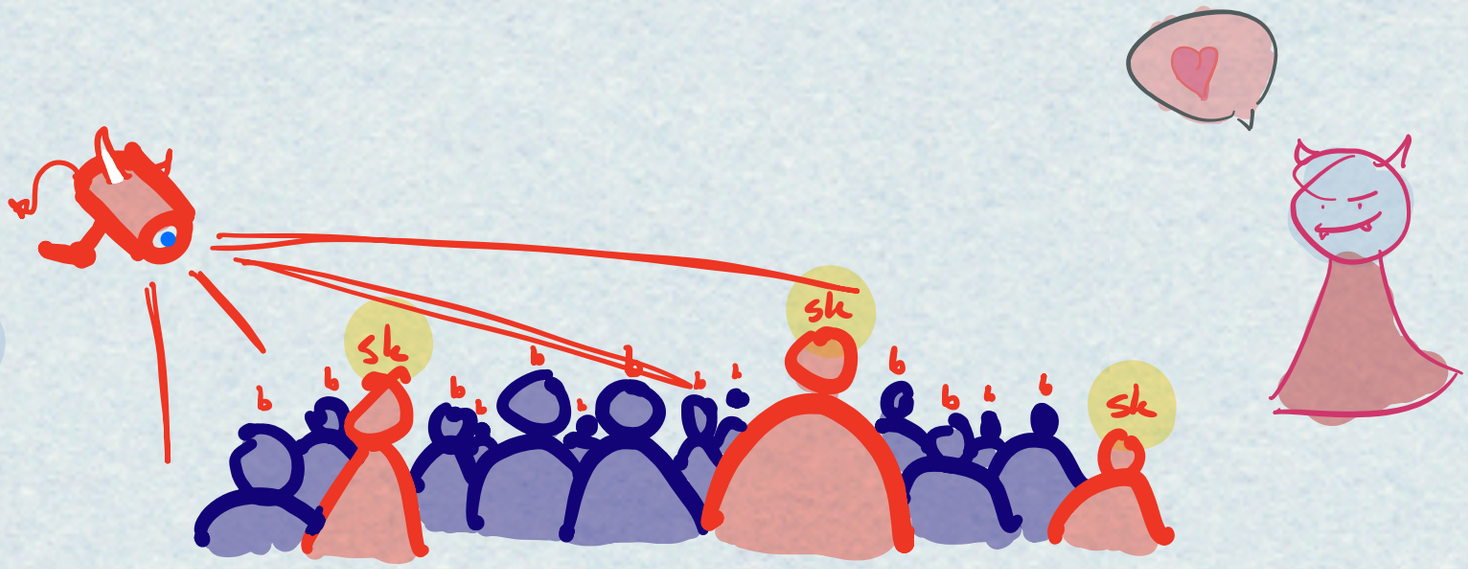


MULTI-USER SECURITY WITH CORRUPTIONS:



A CAN NOW ADDITIONALLY REVEAL PRIVATE KEYS OF USERS. EVEN MORE REALISTIC!

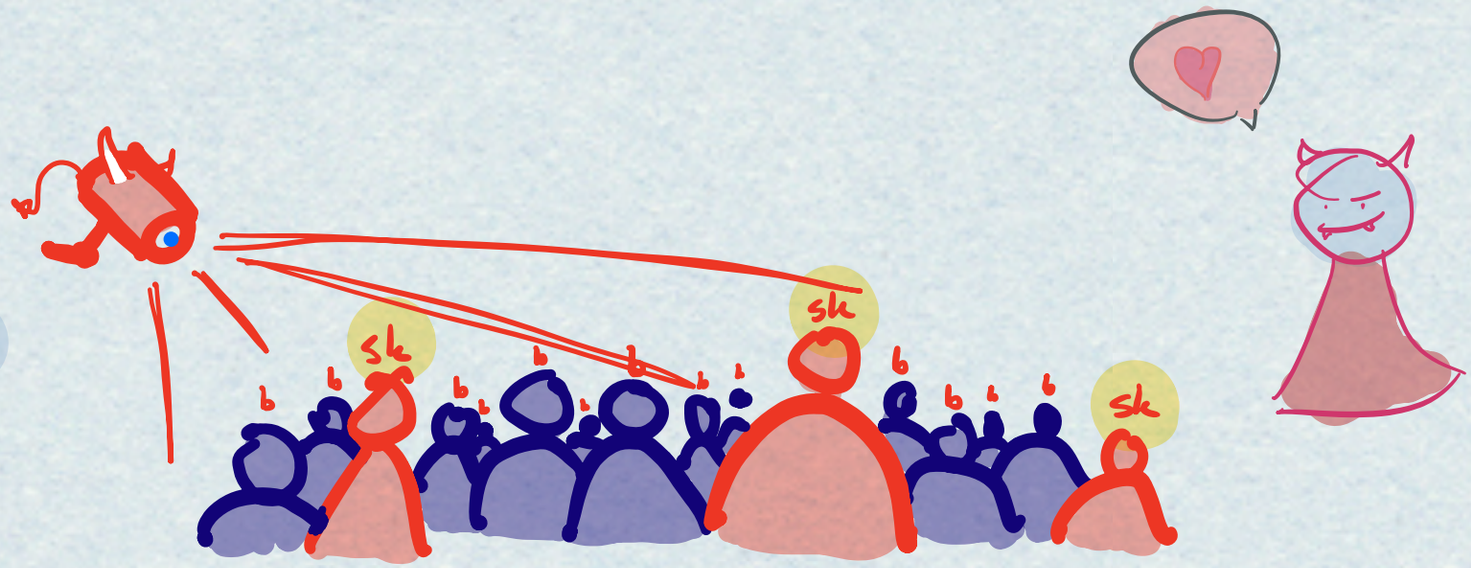
MULTI-USER SECURITY WITH CORRUPTIONS:



$\mathcal{A}$ CAN NOW ADDITIONALLY REVEAL PRIVATE KEYS OF USERS. EVEN MORE REALISTIC!

BUT: $\mathcal{A}$ CAN NOT BOTH CHALLENGE AND CORRUPT A USER:
THIS WOULD REVEAL b!

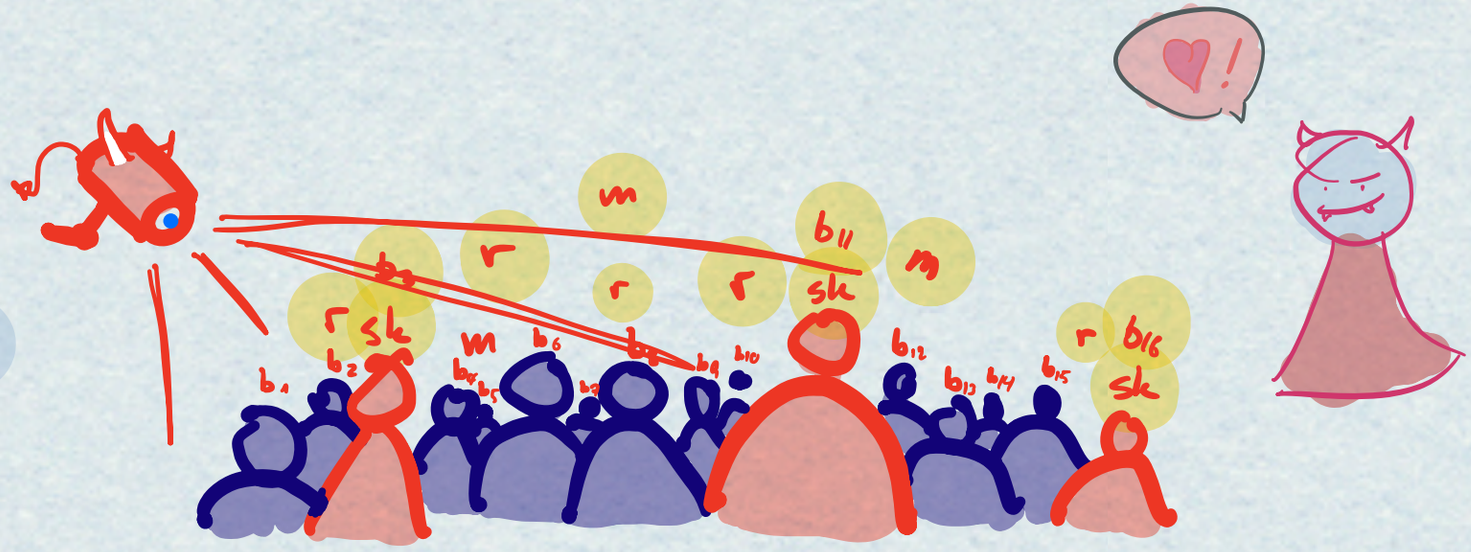
MULTI-USER SECURITY WITH CORRUPTIONS:



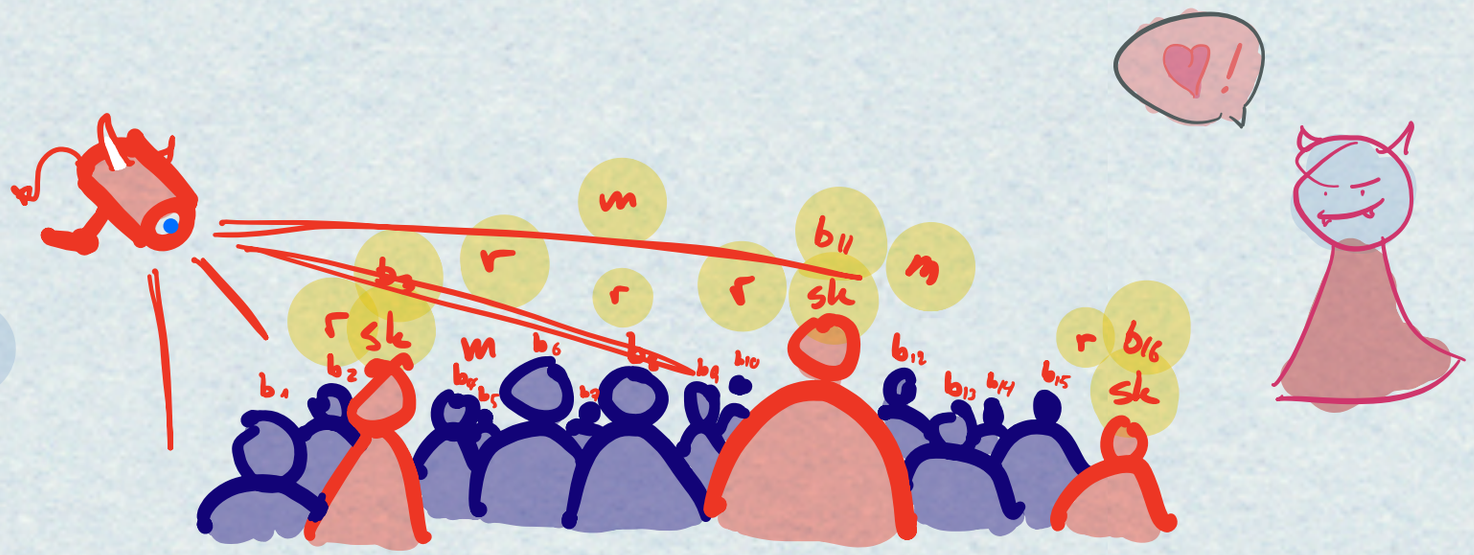
$\mathcal{A}$ CAN NOW ADDITIONALLY REVEAL PRIVATE KEYS OF USERS. EVEN MORE REALISTIC!

BUT: $\mathcal{A}$ CAN NOT BOTH CHALLENGE AND CORRUPT A USER:
THIS WOULD REVEAL b!
UNREALISTIC REQUIREMENT?

MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS

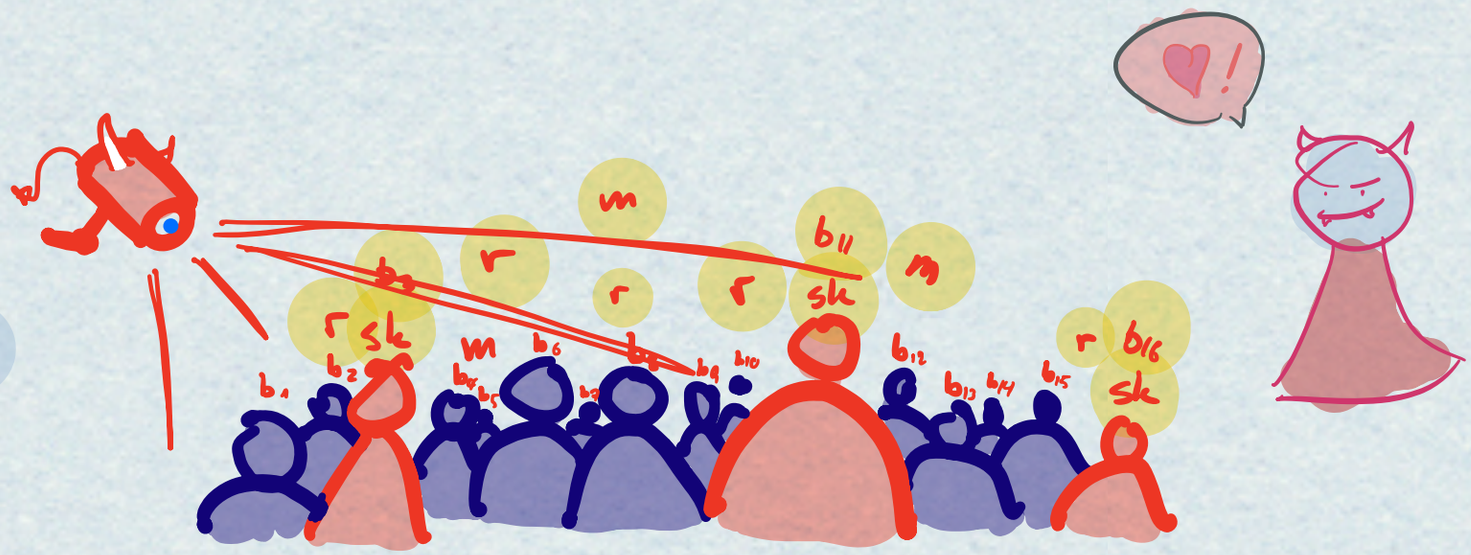


MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS:



WITH MULTIPLE BITS, $\mathcal{A}$ CAN ALSO COMPROMISE CHALLENGES TO LEARN THE RANDOMNESS USED, AND THE MESSAGE.

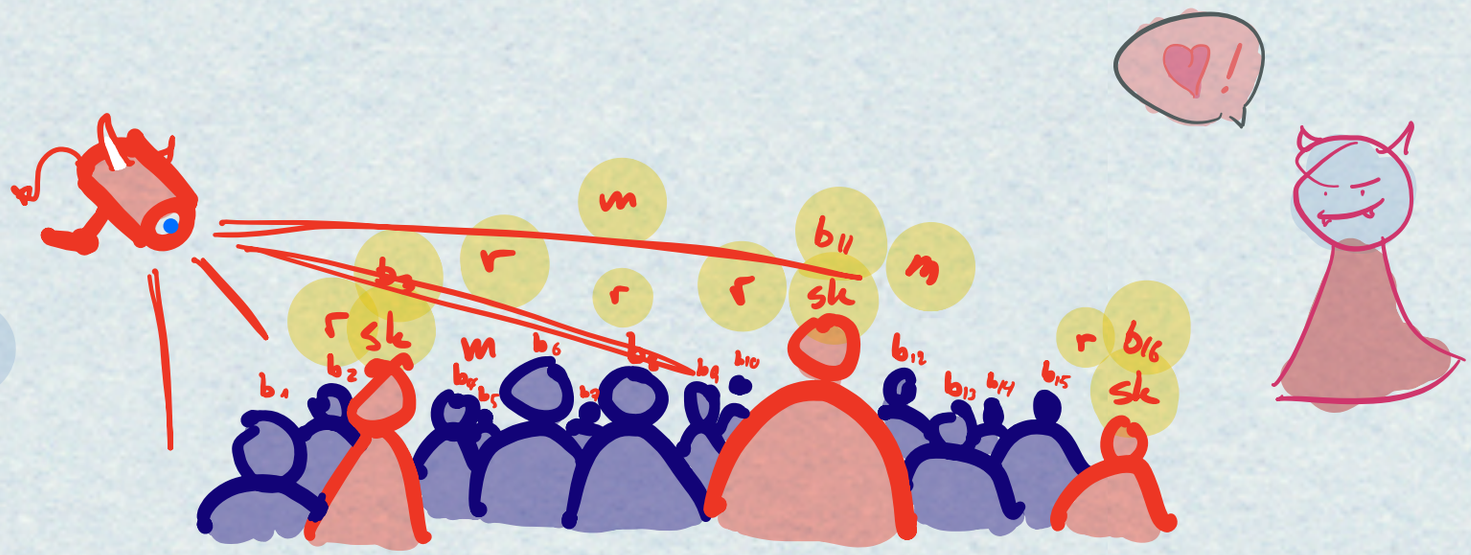
MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS :



WITH MULTIPLE BITS, $\mathcal{A}$ CAN ALSO COMPROMISE CHALLENGES TO LEARN THE RANDOMNESS USED, AND THE MESSAGE.

EVEN MORE REALISTIC ! AND STILL (NON-TIGHTLY) EQUIVALENT TO IND-CCA (DHIES)

MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS :



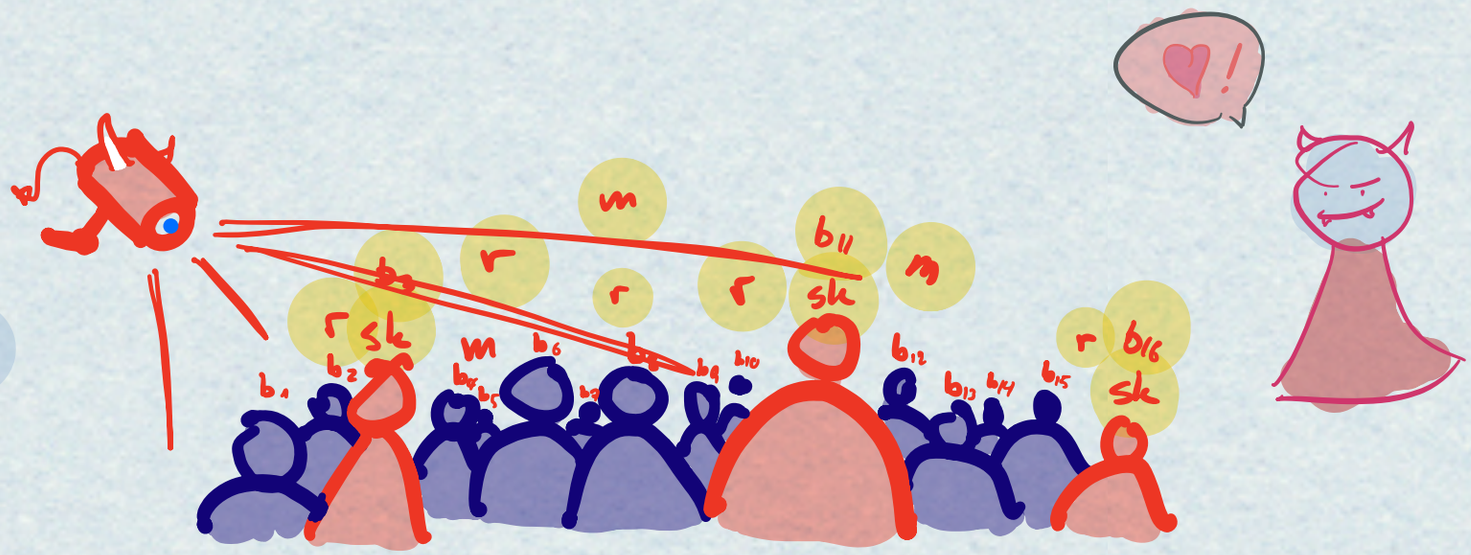
WITH MULTIPLE BITS, $\mathcal{A}$ CAN ALSO COMPROMISE CHALLENGES TO LEARN THE RANDOMNESS USED, AND THE MESSAGE.

EVEN MORE REALISTIC ! AND STILL (NON-TIGHTLY) EQUIVALENT TO IND-CCA (DHIES)

learn the messages and randomness underlying some of the ciphertexts). The concern is that the messages sent by uncorrupted senders stay secret. The second scenario deals with *receiver security*. Here we consider one sender and n receivers who hold independently generated public and secret keys. The attacker is allowed to corrupt some of the receivers (and learn the secret keys that decrypt some of the observed ciphertexts). Security in this setting is concerned with the messages received by uncorrupted receivers. For each of these

Hazay, Patra, Warinschi, AC'15

MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS :



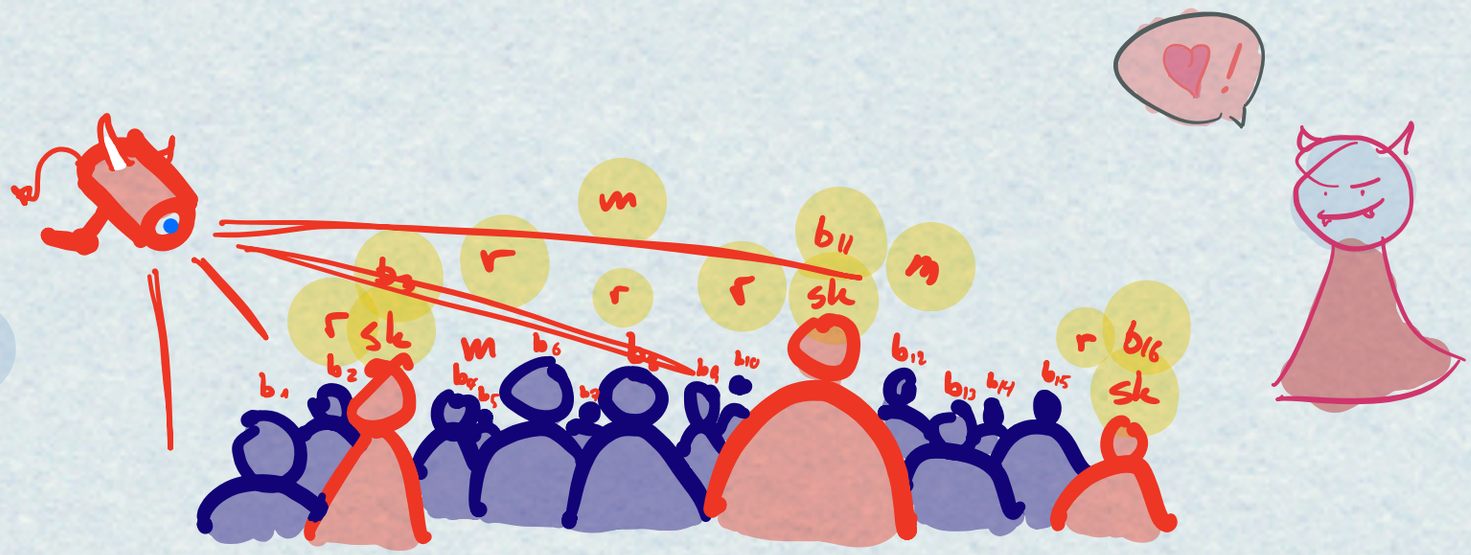
WITH MULTIPLE BITS, $\mathcal{A}$ CAN ALSO COMPROMISE CHALLENGES TO LEARN THE RANDOMNESS USED, AND THE MESSAGE.

EVEN MORE REALISTIC! AND STILL (NON-TIGHTLY) EQUIVALENT TO IND-CCA (DHIES)

learn the messages and randomness underlying some of the ciphertexts). The concern is that the messages sent by uncorrupted senders stay secret. The second scenario deals with *receiver security*. Here we consider one sender and n receivers who hold independently generated public and secret keys. The attacker is allowed to corrupt some of the receivers (and learn the secret keys that decrypt some of the observed ciphertexts). Security in this setting is concerned with the messages received by uncorrupted receivers. For each of these

"SELECTIVE OPENING ATTACKS" Hazay, Patra, Warinschi, AC'15

MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS :



WITH MULTIPLE BITS, $\mathcal{A}$ CAN ALSO COMPROMISE CHALLENGES TO LEARN THE RANDOMNESS USED, AND THE MESSAGE.

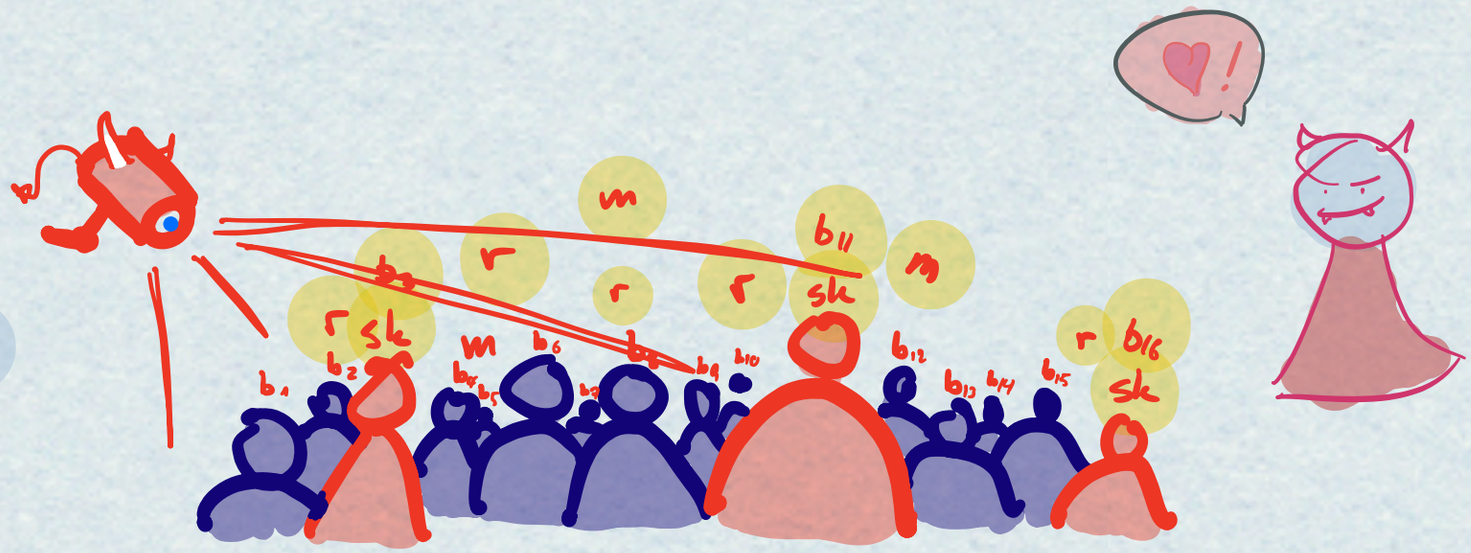
EVEN MORE REALISTIC ! AND STILL (NON-TIGHTLY) EQUIVALENT TO IND-CCA (DHIES)

learn the messages and randomness underlying some of the ciphertexts). The concern is that the messages sent by uncorrupted senders stay secret. The second scenario deals with *receiver security*. Here we consider one sender and n receivers who hold independently generated public and secret keys. The attacker is allowed to corrupt some of the receivers (and learn the secret keys that decrypt some of the observed ciphertexts). Security in this setting is concerned with the messages received by uncorrupted receivers. For each of these

"SELECTIVE OPENING ATTACKS" Hazay, Patra, Warinschi, AC'15

SOA ⇕ IND-CCA ?

MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS



WITH MULTIPLE BITS, $\mathcal{A}$ CAN ALSO COMPROMISE CHALLENGES TO LEARN THE RANDOMNESS USED, AND THE MESSAGE.

EVEN MORE REALISTIC! AND STILL (NON-TIGHTLY) EQUIVALENT TO IND-CCA (DHIES)

learn the messages and randomness underlying some of the ciphertexts). The concern is that the messages sent by uncorrupted senders stay secret. The second scenario deals with *receiver security*. Here we consider one sender and n receivers who hold independently generated public and secret keys. The attacker is allowed to corrupt some of the receivers (and learn the secret keys that decrypt some of the observed ciphertexts). Security in this setting is concerned with the messages received by uncorrupted receivers. For each of these

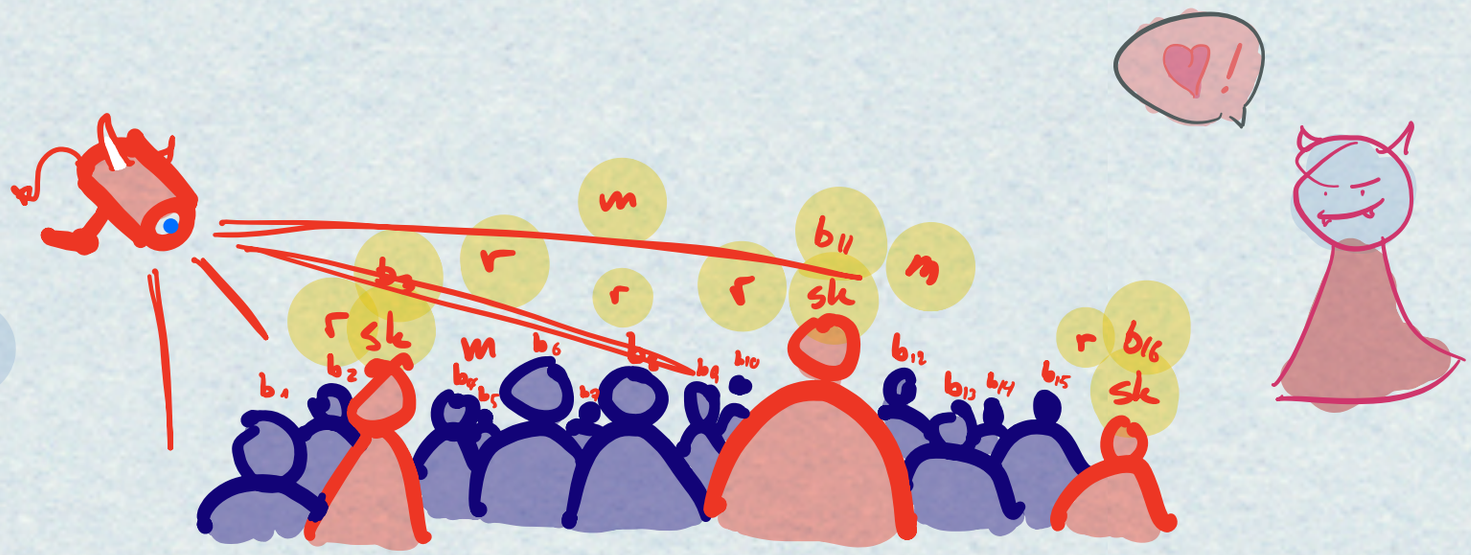
"SELECTIVE OPENING ATTACKS" Hazay, Patra, Warinschi, AC'15

a) Non-Committing Encryption: Security against adaptive corruptions is obviously the more realistic notion for practical applications, but is notoriously difficult to achieve for public-key encryption because of the so-called *selective decommitment problem* [22], [34]. In a nutshell, the problem is the

Camenisch, Lehmann, Neven, Sametin, CSF'17

SOA ⇕ IND-CCA ?

MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS :



WITH MULTIPLE BITS, $\mathcal{A}$ CAN ALSO COMPROMISE CHALLENGES TO LEARN THE RANDOMNESS USED, AND THE MESSAGE.

EVEN MORE REALISTIC! AND STILL (NON-TIGHTLY) EQUIVALENT TO IND-CCA (DHIES)

learn the messages and randomness underlying some of the ciphertexts). The concern is that the messages sent by uncorrupted senders stay secret. The second scenario deals with *receiver security*. Here we consider one sender and n receivers who hold independently generated public and secret keys. The attacker is allowed to corrupt some of the receivers (and learn the secret keys that decrypt some of the observed ciphertexts). Security in this setting is concerned with the messages received by uncorrupted receivers. For each of these

"SELECTIVE OPENING ATTACKS" Hazay, Patra, Warinschi, AC'15

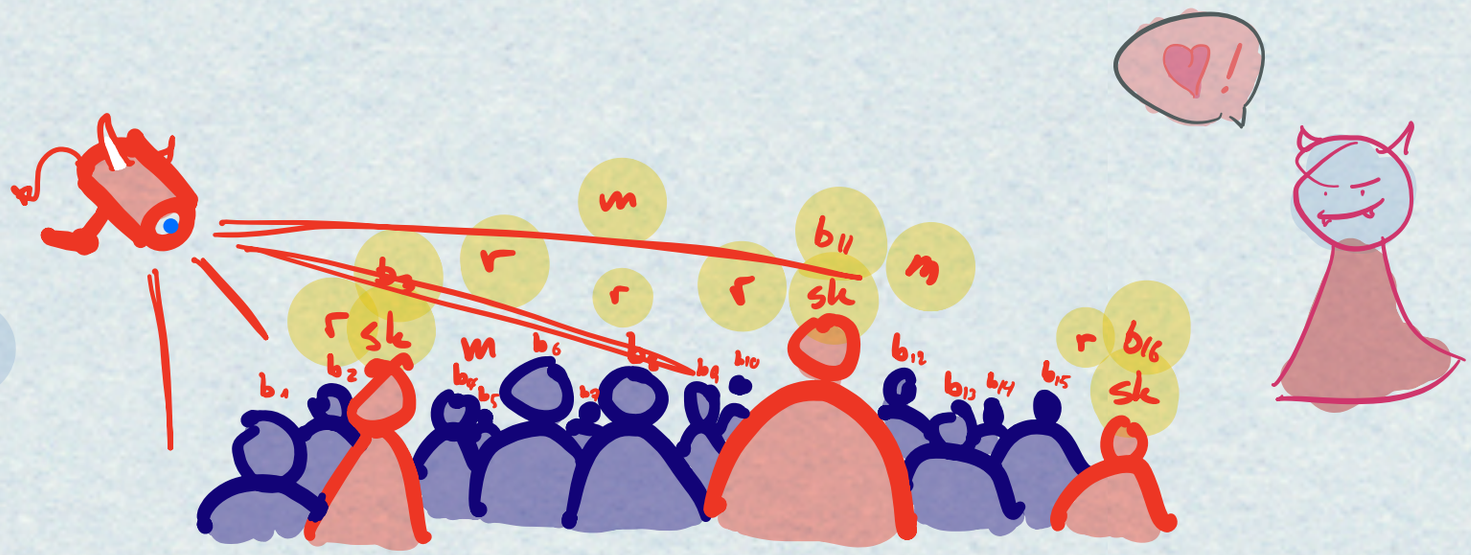
a) Non-Committing Encryption: Security against adaptive corruptions is obviously the more realistic notion for practical applications, but is notoriously difficult to achieve for public-key encryption because of the so-called *selective decommitment problem* [22], [34]. In a nutshell, the problem is the

Camenisch, Lehmann, Neven, Sametin, CSF'17

"NON-COMMITTING ENCRYPTION"

SOA ⇕ IND-CCA ?

MULTI-USER SECURITY WITH MULTIPLE CHALLENGE BITS :



WITH MULTIPLE BITS, $\mathcal{A}$ CAN ALSO COMPROMISE CHALLENGES TO LEARN THE RANDOMNESS USED, AND THE MESSAGE.

EVEN MORE REALISTIC! AND STILL (NON-TIGHTLY) EQUIVALENT TO IND-CCA (DNIEU)

learn the messages and randomness underlying some of the ciphertexts). The concern is that the messages sent by uncorrupted senders stay secret. The second scenario deals with *receiver security*. Here we consider one sender and n receivers who hold independently generated public and secret keys. The attacker is allowed to corrupt some of the receivers (and learn the secret keys that decrypt some of the observed ciphertexts). Security in this setting is concerned with the messages received by uncorrupted receivers. For each of these

"SELECTIVE OPENING ATTACKS" Hazay, Patra, Warinschi, AC'15

SOA ⇕ IND-CCA ?

a) Non-Committing Encryption: Security against adaptive corruptions is obviously the more realistic notion for practical applications, but is notoriously difficult to achieve for public-key encryption because of the so-called *selective decommitment problem* [22], [34]. In a nutshell, the problem is the

Camenisch, Lehmann, Neven, Sanchez, CSF'17

"NON-COMMITTING ENCRYPTION"

UNACHIEVABLE IN THE STANDARD MODEL ??

4 PHILOSOPHIES OF SECURITY

WITHOUT COMPROMISE

WITH COMPROMISE

A PRIORI INDISTINGUISHABILITY:

A POSTERIORI INDISTINGUISHABILITY:

A POSTERIORI SIMULATABILITY:

A PRIORI SIMULATABILITY:

4 PHILOSOPHIES OF SECURITY

WITHOUT COMPROMISE | WITH COMPROMISE

A PRIORI INDISTINGUISHABILITY:

$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$

A POSTERIORI INDISTINGUISHABILITY:

A POSTERIORI SIMULATABILITY:

A PRIORI SIMULATABILITY:

4 PHILOSOPHIES OF SECURITY

	WITHOUT COMPROMISE	WITH COMPROMISE
A PRIORI INDISTINGUISHABILITY:	IND-CCA	

$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$

A POSTERIORI INDISTINGUISHABILITY:

A POSTERIORI SIMULATABILITY:

A PRIORI SIMULATABILITY:

4 PHILOSOPHIES OF SECURITY

WITHOUT COMPROMISE — WITH COMPROMISE

A PRIORI INDISTINGUISHABILITY: — IND-CCA

$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$

A POSTERIORI INDISTINGUISHABILITY:

$\rightsquigarrow \forall m_0, m_1, c: \Pr[M = m_0 \mid C = c] = \Pr[M = m_1 \mid C = c]$

A POSTERIORI SIMULATABILITY:

A PRIORI SIMULATABILITY:

4 PHILOSOPHIES OF SECURITY

WITHOUT COMPROMISE | WITH COMPROMISE

A PRIORI INDISTINGUISHABILITY: IND-CCA

$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$

A POSTERIORI INDISTINGUISHABILITY: (KEM-IND LIKE)

$\rightsquigarrow \forall m_0, m_1, c: \Pr[M = m_0 \mid C = c] = \Pr[M = m_1 \mid C = c]$

A POSTERIORI SIMULATABILITY:

A PRIORI SIMULATABILITY:

4 PHILOSOPHIES OF SECURITY

WITHOUT COMPROMISE — WITH COMPROMISE

A PRIORI INDISTINGUISHABILITY: IND-CCA

$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$

A POSTERIORI INDISTINGUISHABILITY: (KEM-IND LIKE)

$\rightsquigarrow \forall m_0, m_1, c: \Pr[M = m_0 \mid C = c] = \Pr[M = m_1 \mid C = c]$

A POSTERIORI SIMULATABILITY:

$\rightsquigarrow \forall m, c: \Pr[M = m \mid C = c] = \Pr[M = m]$

A PRIORI SIMULATABILITY:

4 PHILOSOPHIES OF SECURITY

	WITHOUT COMPROMISE	WITH COMPROMISE
A PRIORI INDISTINGUISHABILITY:	IND-CCA	
A POSTERIORI INDISTINGUISHABILITY:	(KEM-IND LIKE)	
A POSTERIORI SIMULATABILITY:	SEMANTIC SECURITY	
A PRIORI SIMULATABILITY:		

$$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$$

$$\rightsquigarrow \forall m_0, m_1, c: \Pr[M = m_0 \mid C = c] = \Pr[M = m_1 \mid C = c]$$

$$\rightsquigarrow \forall m, c: \Pr[M = m \mid C = c] = \Pr[M = m]$$

4 PHILOSOPHIES OF SECURITY

	WITHOUT COMPROMISE	WITH COMPROMISE
A PRIORI INDISTINGUISHABILITY:	IND-CCA	
⇝ $\forall m_0, m_1, c: \Pr[C=c \mid M=m_0] = \Pr[C=c \mid M=m_1]$		
A POSTERIORI INDISTINGUISHABILITY:	(KEM-IND LIKE)	
⇝ $\forall m_0, m_1, c: \Pr[M=m_0 \mid C=c] = \Pr[M=m_1 \mid C=c]$		
A POSTERIORI SIMULATABILITY:	SEMANTIC SECURITY	
⇝ $\forall m, c: \Pr[M=m \mid C=c] = \Pr[M=m]$		
A PRIORI SIMULATABILITY:		
⇝ $\forall m, c: \Pr[C=c \mid M=m] = \Pr[C=c]$		

4 PHILOSOPHIES OF SECURITY

	WITHOUT COMPROMISE	WITH COMPROMISE
A PRIORI INDISTINGUISHABILITY:	IND-CCA	
A POSTERIORI INDISTINGUISHABILITY:	(KEM-IND LIKE)	
A POSTERIORI SIMULATABILITY:	SEMANTIC SECURITY	
A PRIORI SIMULATABILITY:	UNIVERSAL COMPOSABILITY	

$$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$$

$$\rightsquigarrow \forall m_0, m_1, c: \Pr[M = m_0 \mid C = c] = \Pr[M = m_1 \mid C = c]$$

$$\rightsquigarrow \forall m, c: \Pr[M = m \mid C = c] = \Pr[M = m]$$

$$\rightsquigarrow \forall m, c: \Pr[C = c \mid M = m] = \Pr[C = c]$$

4 PHILOSOPHIES OF SECURITY

	WITHOUT COMPROMISE	WITH COMPROMISE
A PRIORI INDISTINGUISHABILITY:	IND-CCA	
A POSTERIORI INDISTINGUISHABILITY:	(KEM-IND LIKE)	
A POSTERIORI SIMULATABILITY:	SEMANTIC SECURITY	
A PRIORI SIMULATABILITY:	UNIVERSAL COMPOSABILITY	

$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$

$\rightsquigarrow \forall m_0, m_1, c: \Pr[M = m_0 \mid C = c] = \Pr[M = m_1 \mid C = c]$

$\rightsquigarrow \forall m, c: \Pr[M = m \mid C = c] = \Pr[M = m]$

$\rightsquigarrow \forall m, c: \Pr[C = c \mid M = m] = \Pr[C = c]$

WITHOUT CORRUPTIONS, ALL* EQUIVALENT $\rightsquigarrow$ JUST USE IND-CCA!

4 PHILOSOPHIES OF SECURITY

	WITHOUT COMPROMISE	WITH COMPROMISE
A PRIORI INDISTINGUISHABILITY:	IND-CCA	MULTI-USER IND-CCA WITH CORRUPTIONS
A POSTERIORI INDISTINGUISHABILITY:	(KEM-IND LIKE)	INDISTINGUISHABILITY-BASED SOA (ISO)
A POSTERIORI SIMULATABILITY:	SEMANTIC SECURITY	SIMULATABILITY-BASED SOA (SSO)
A PRIORI SIMULATABILITY:	UNIVERSAL COMPOSABILITY	NON-COMMITTING ENCRYPTION (NCE)

$\rightsquigarrow \forall m_0, m_1, c: \Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$

$\rightsquigarrow \forall m_0, m_1, c: \Pr[M = m_0 \mid C = c] = \Pr[M = m_1 \mid C = c]$

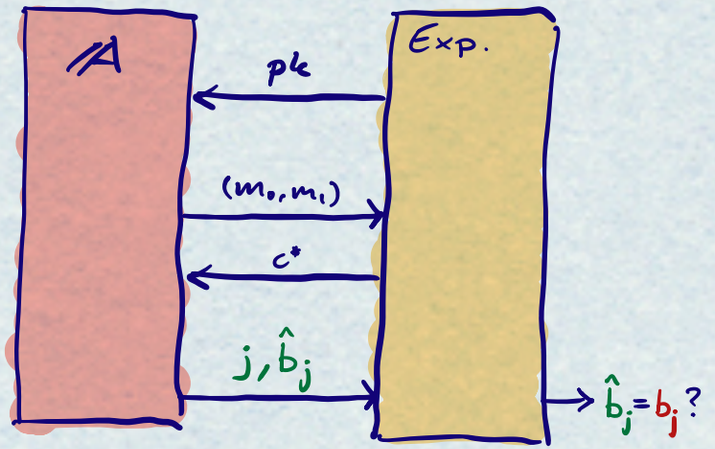
$\rightsquigarrow \forall m, c: \Pr[M = m \mid C = c] = \Pr[M = m]$

$\rightsquigarrow \forall m, c: \Pr[C = c \mid M = m] = \Pr[C = c]$

WITHOUT CORRUPTIONS, ALL* EQUIVALENT $\rightsquigarrow$ JUST USE IND-CCA!

4 PHILOSOPHIES OF SECURITY

A PRIORI INDISTINGUISHABILITY (IND)



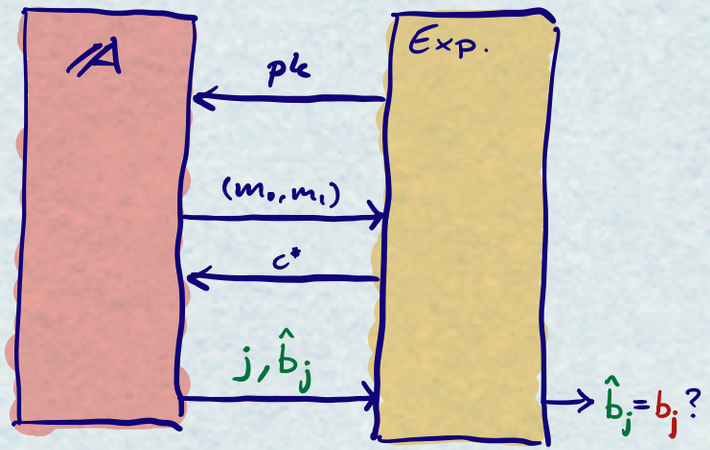
A POSTERIORI INDISTINGUISHABILITY (ISO)

A POSTERIORI SIMULATABILITY (SSO)

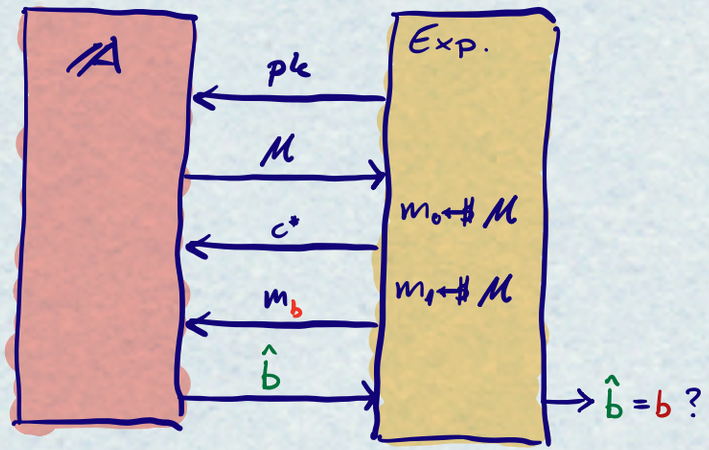
A PRIORI SIMULATABILITY (NCE)

4 PHILOSOPHIES OF SECURITY

A PRIORI INDISTINGUISHABILITY (IND)



A POSTERIORI INDISTINGUISHABILITY (ISO)

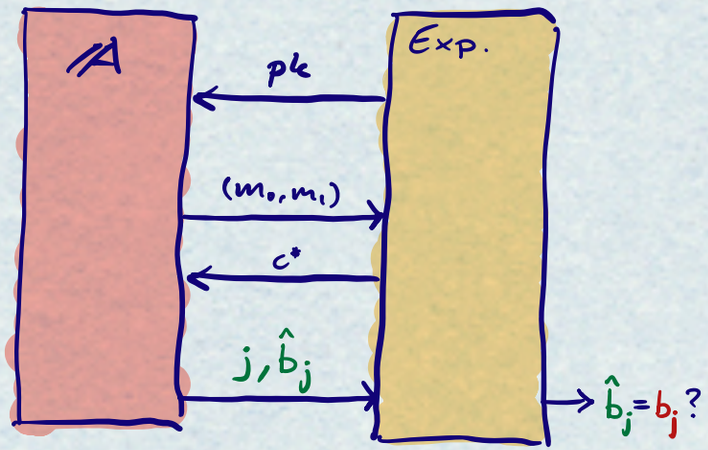


A POSTERIORI SIMULATABILITY (SSO)

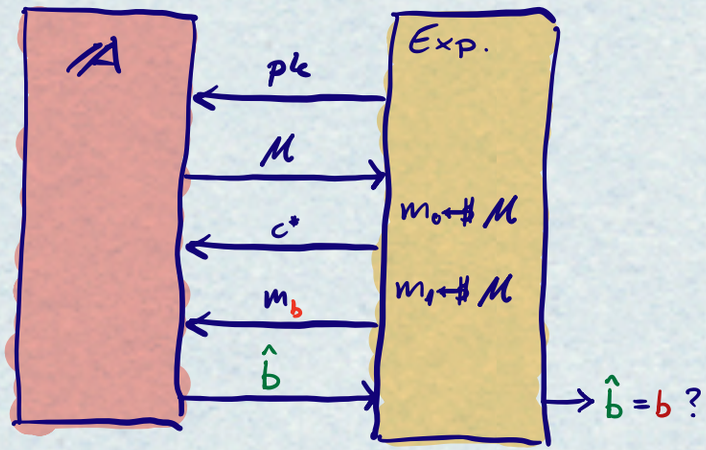
A PRIORI SIMULATABILITY (NCE)

4 PHILOSOPHIES OF SECURITY

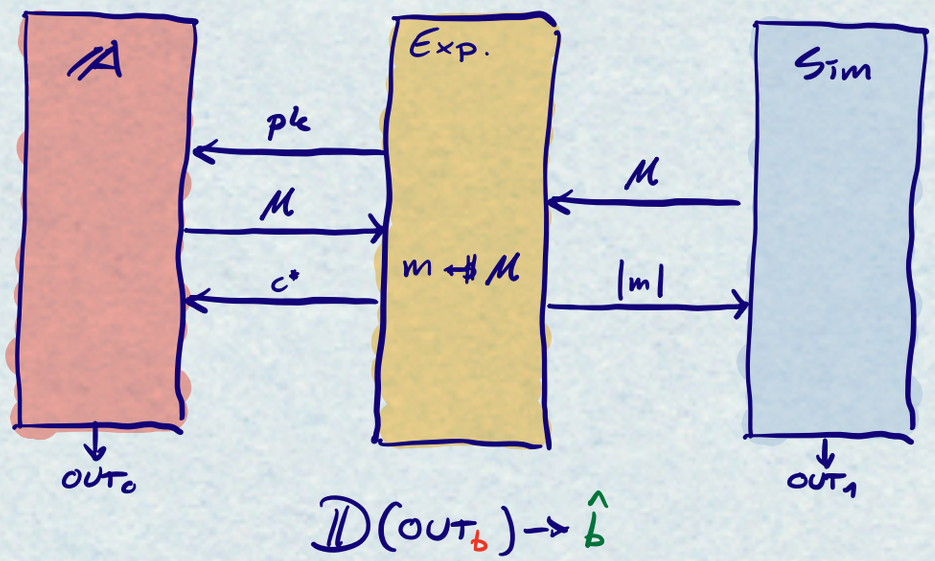
A PRIORI INDISTINGUISHABILITY (IND)



A POSTERIORI INDISTINGUISHABILITY (ISO)



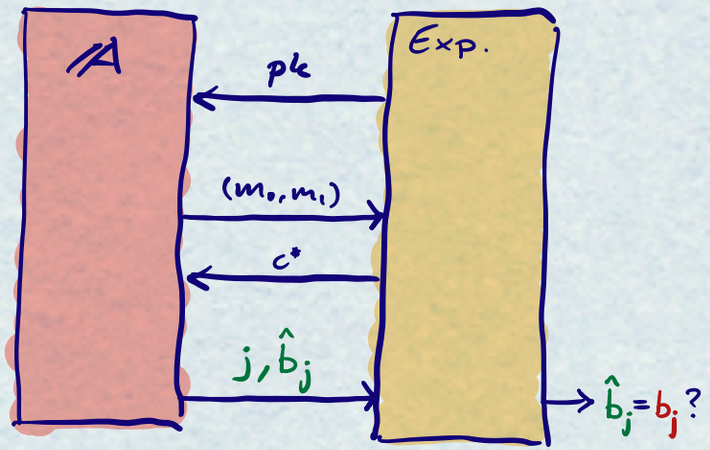
A POSTERIORI SIMULATABILITY (SSO)



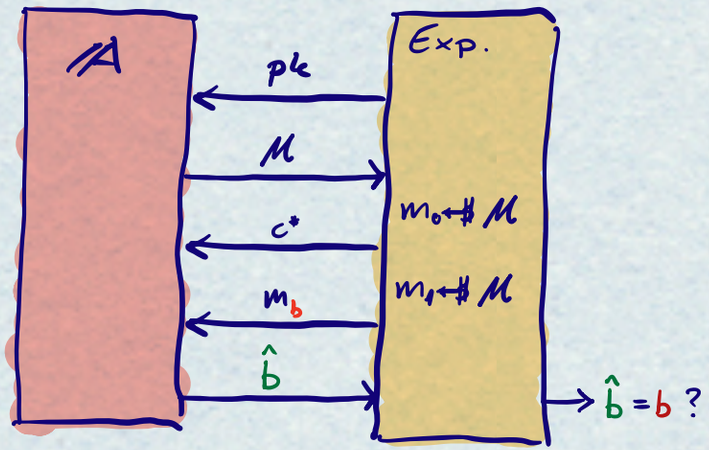
A PRIORI SIMULATABILITY (NCE)

4 PHILOSOPHIES OF SECURITY

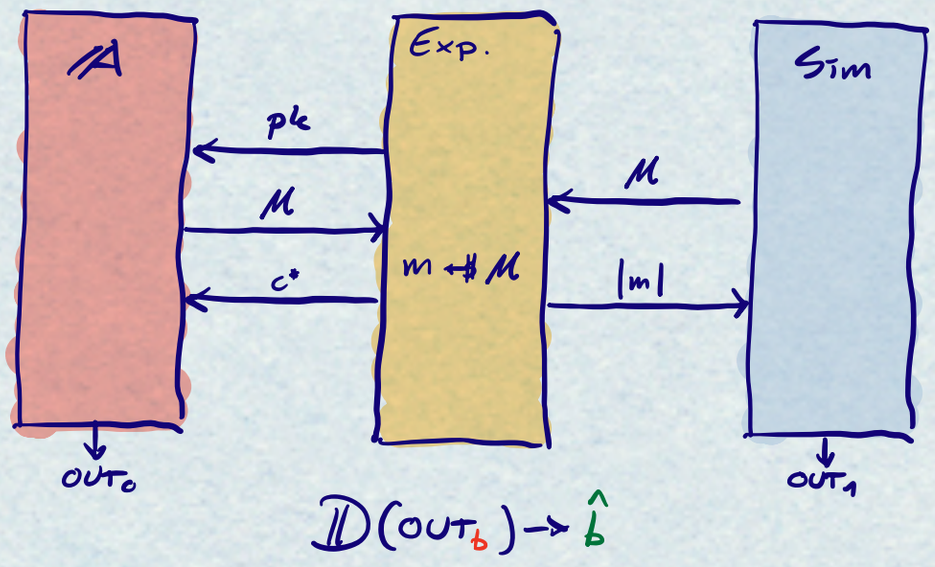
A PRIORI INDISTINGUISHABILITY (IND)



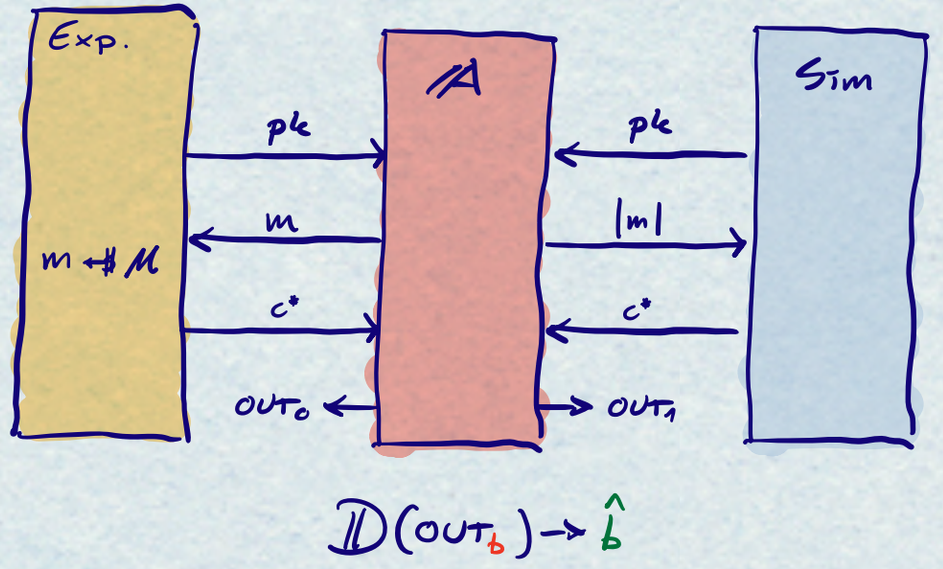
A POSTERIORI INDISTINGUISHABILITY (ISO)



A POSTERIORI SIMULATABILITY (SSO)



A PRIORI SIMULATABILITY (NCE)



4 KINDS OF CORRUPTION A.K.A. "OPENING"

TRANSMISSION OPENING (◇): MESSAGE LEAKS

4 KINDS OF CORRUPTION A.K.A. "OPENING"

TRANSMISSION OPENING (◇): MESSAGE LEAKS

SENDER OPENING (⊙): MESSAGE AND RANDOMNESS LEAKS

4 KINDS OF CORRUPTION A.K.A. "OPENING"

TRANSMISSION OPENING (◇): MESSAGE LEAKS

SENDER OPENING (⊙): MESSAGE AND RANDOMNESS LEAKS

RECEIVER OPENING (★): SECRET KEY LEAKS

4 KINDS OF CORRUPTION A.K.A. "OPENING"

TRANSMISSION OPENING (◇): MESSAGE LEAKS

SENDER OPENING (⊙): MESSAGE AND RANDOMNESS LEAKS

RECEIVER OPENING (★): SECRET KEY LEAKS

BI-OPENING (✪): ALL OF THE ABOVE LEAK

4 KINDS OF CORRUPTION A.K.A. "OPENING"

TRANSMISSION OPENING (◇): MESSAGE LEAKS

SENDER OPENING (⊙): MESSAGE AND RANDOMNESS LEAKS

RECEIVER OPENING (★): SECRET KEY LEAKS

BI-OPENING (✪): ALL OF THE ABOVE LEAK

COMBINING: $\{\text{IND}, \text{ISO}, \text{SSO}, \text{NCE}\} - \{\text{CPA}, \text{CCA}\} - \{◇, ⊙, ★, ✪\}$

4 KINDS OF CORRUPTION A.K.A. "OPENING"

TRANSMISSION OPENING (◇): MESSAGE LEAKS

SENDER OPENING (⊙): MESSAGE AND RANDOMNESS LEAKS

RECEIVER OPENING (★): SECRET KEY LEAKS

BI-OPENING (✪): ALL OF THE ABOVE LEAK

COMBINING: $\{\text{IND}, \text{ISO}, \text{SSO}, \text{NCE}\} - \{\text{CPA}, \text{CCA}\} - \{◇, ⊙, ★, ✪\}$ ← 32 NOTIONS OF SECURITY!

4 KINDS OF CORRUPTION A.K.A. "OPENING"

TRANSMISSION OPENING (◇): MESSAGE LEAKS

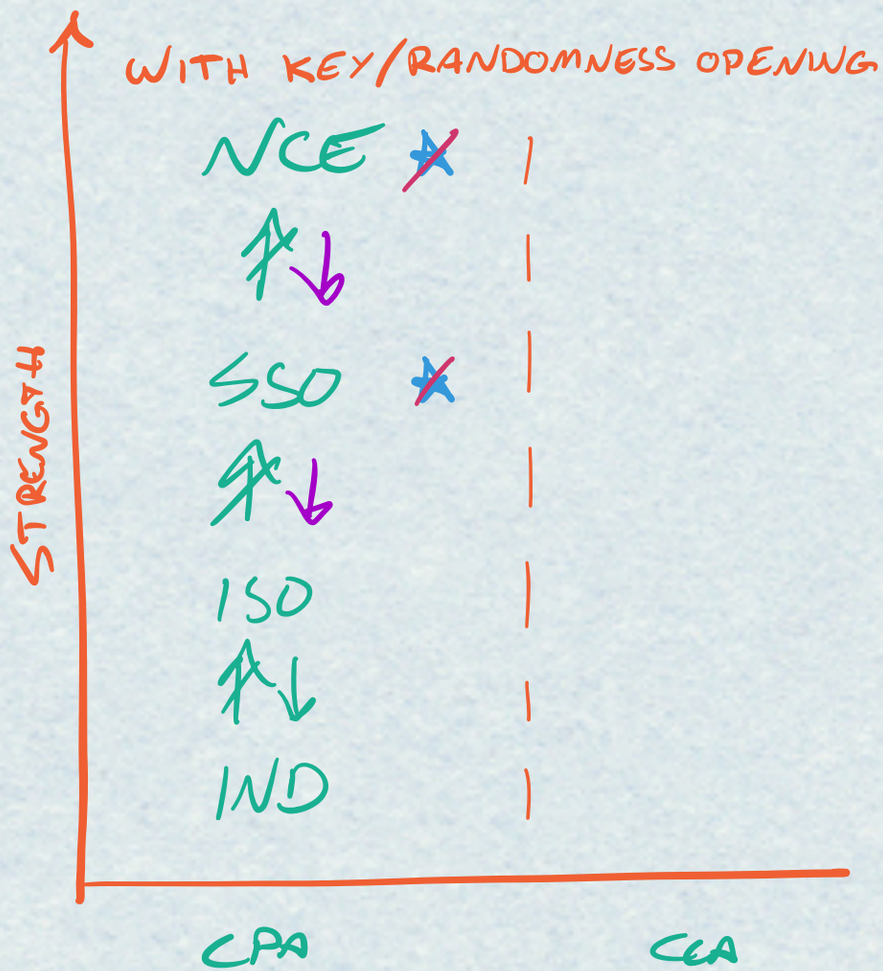
SENDER OPENING (⊙): MESSAGE AND RANDOMNESS LEAKS

RECEIVER OPENING (★): SECRET KEY LEAKS

BI-OPENING (✪): ALL OF THE ABOVE LEAK

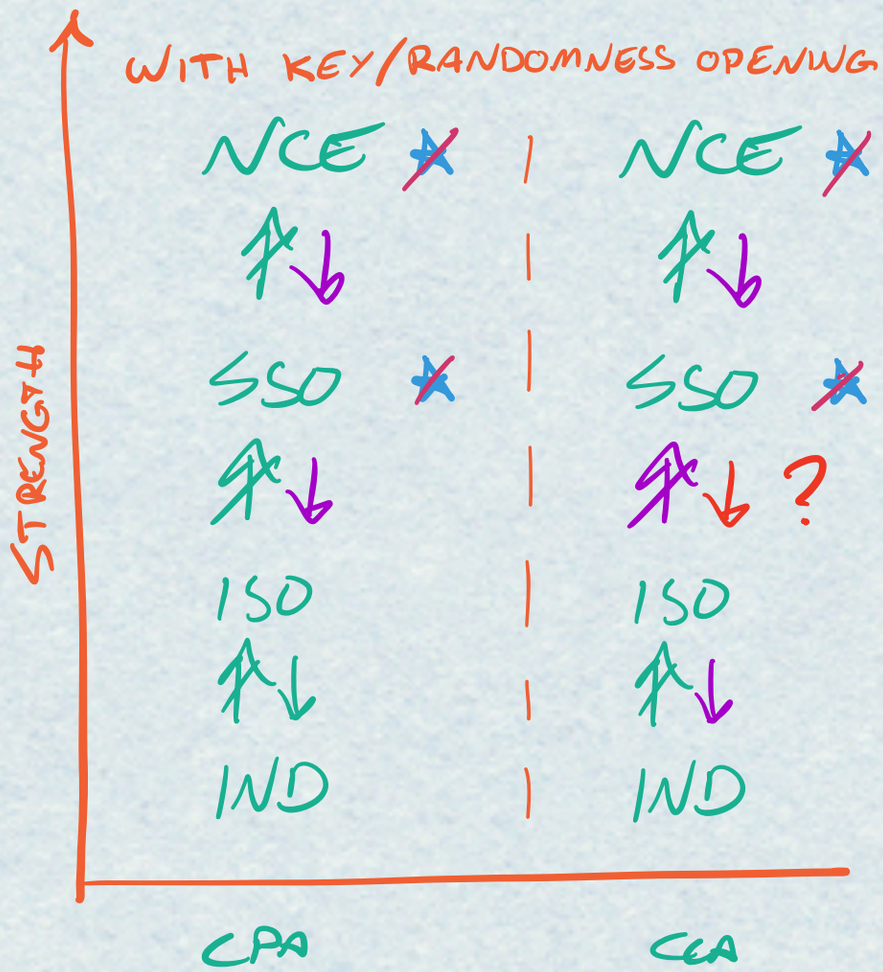
COMBINING: $\{\text{IND}, \text{ISO}, \text{SSO}, \text{NCE}\} - \{\text{CPA}, \text{CCA}\} - \{◇, ⊙, ★, ✪\}$ ← ~~32~~ 30 NOTIONS OF SECURITY!

The Hierarchy



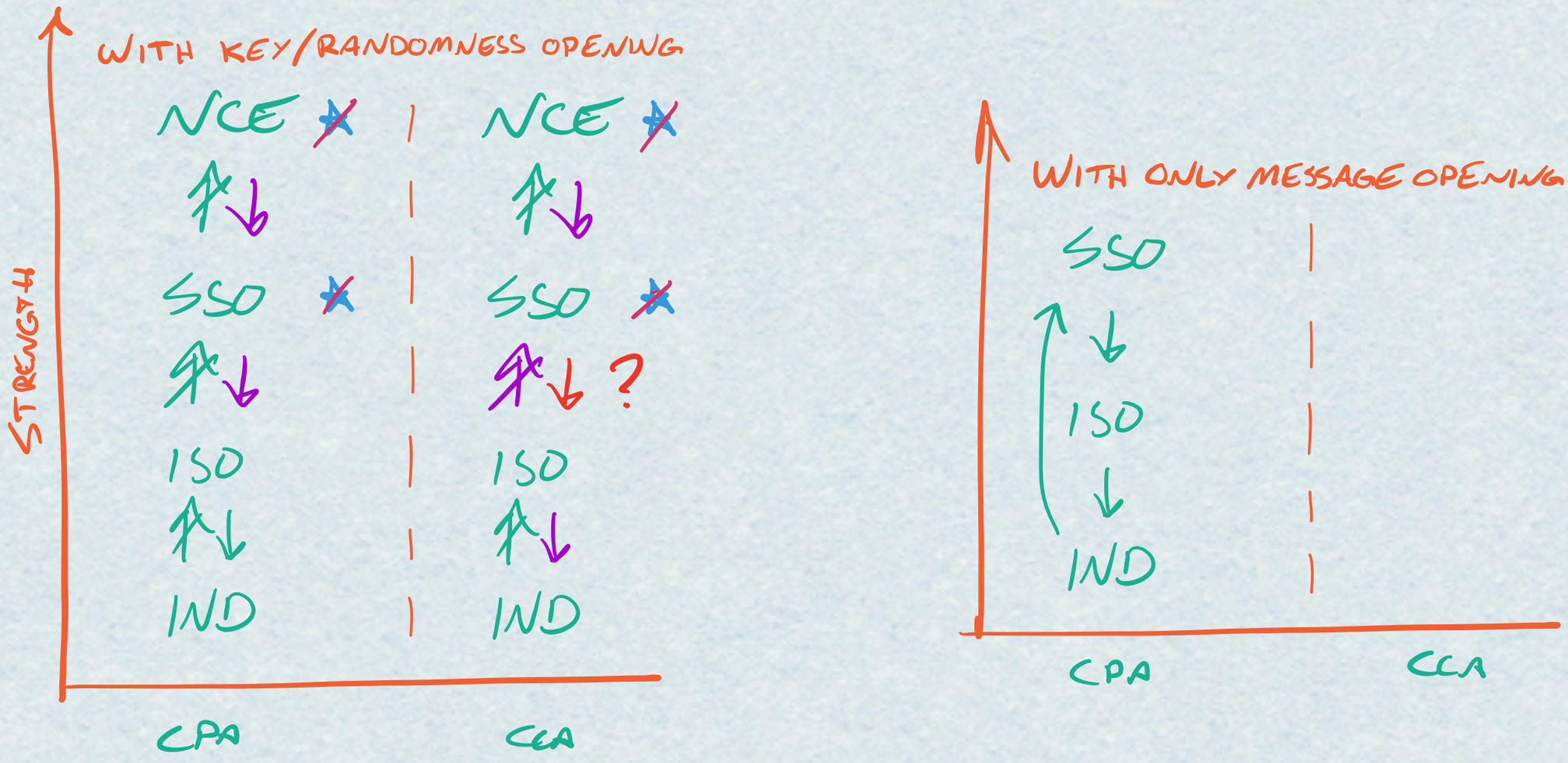
PURPLE = NOVEL, RED = OPEN, ~~X~~ = STANDARD MODEL UNACHIEVABLE

The Hierarchy



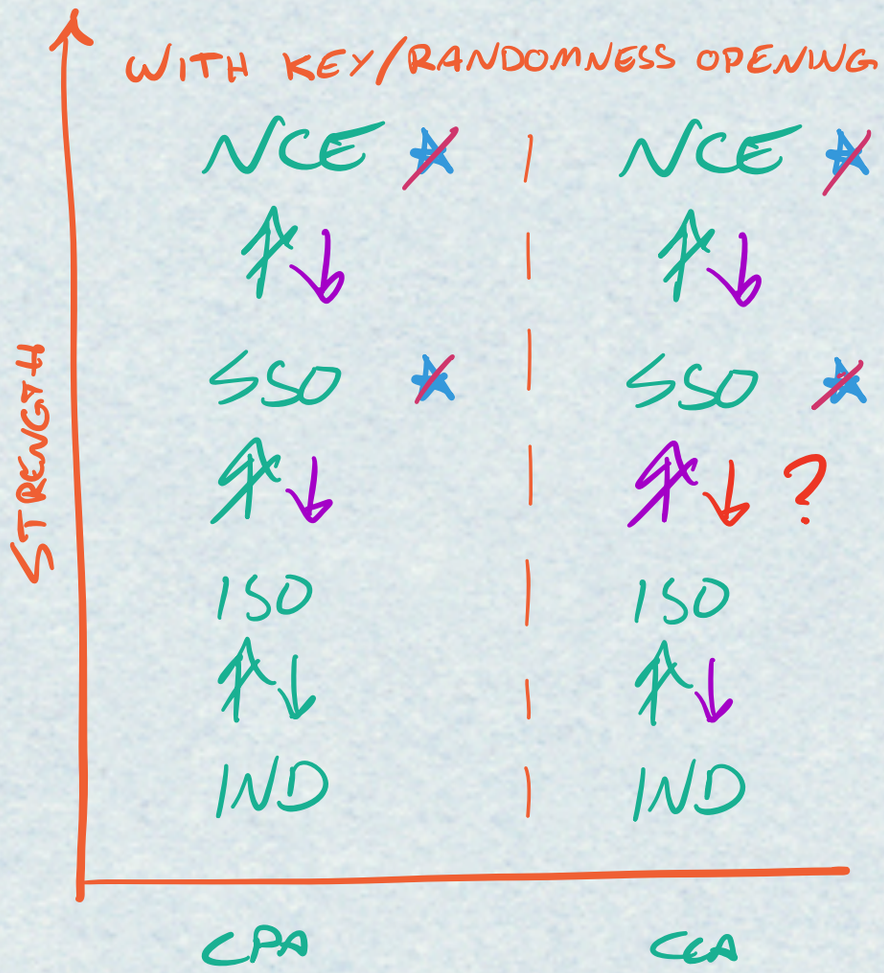
PURPLE = NOVEL, RED = OPEN, ~~★~~ = STANDARD MODEL UNACHIEVABLE

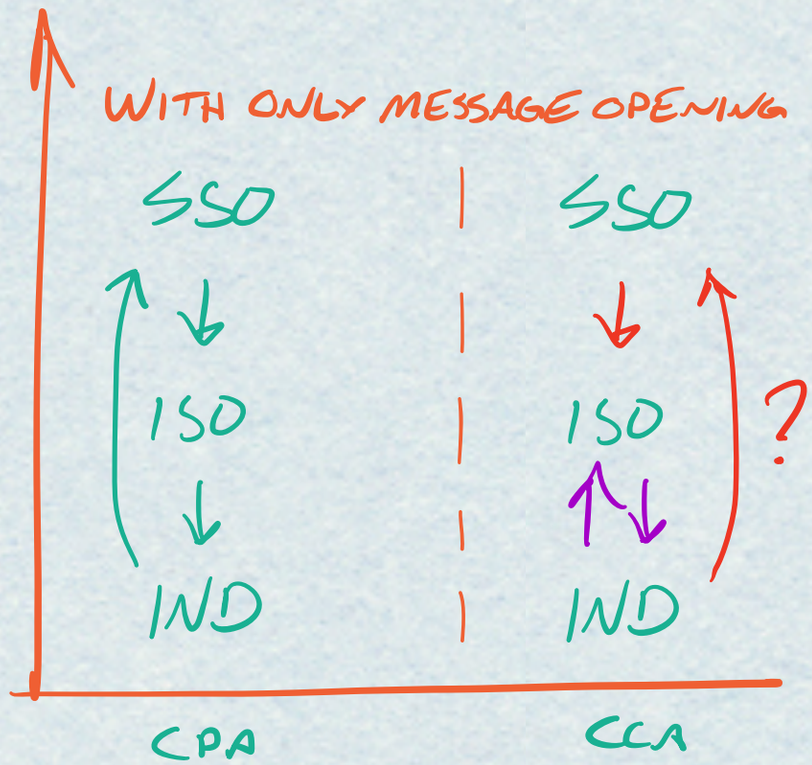
The Hierarchy



Purple = novel, Red = open, ✗ = standard model unachievable

The Hierarchy

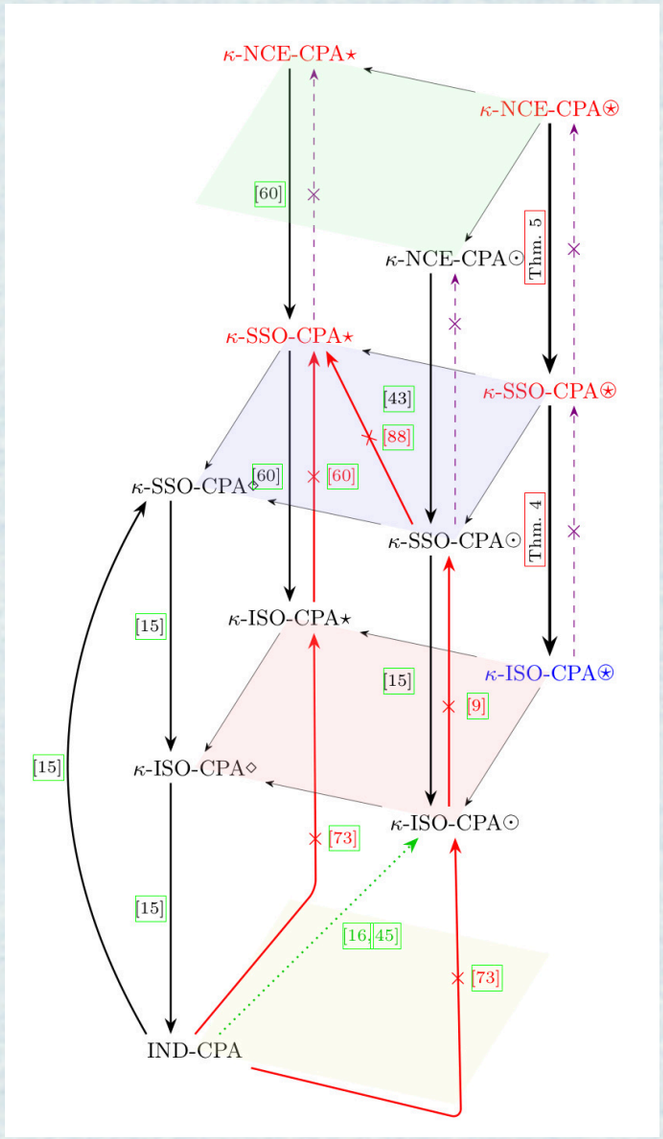




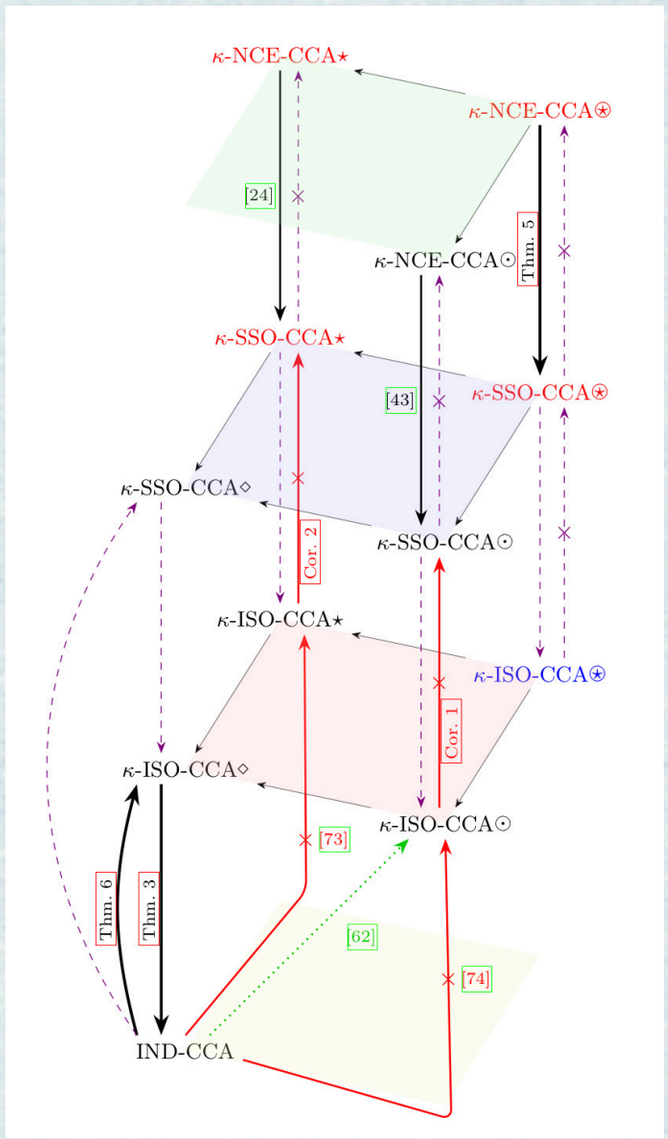
Purple = novel, Red = open, ~~X~~ = standard model unachievable

RELATIONS

CPA:



CCA:



ACHIEVABILITY

	◇	○	☆	⊛
IND	=	=	=	=
ISO	=			
SSO	=			
NCE	(=)			

= : IND-CCA EQUIVALENT

✓ : ACHIEVED IN THE STANDARD MODEL

∪ : UNACHIEVABLE IN THE STANDARD MODEL

ACHIEVABILITY

	◇	○	★	✪
IND	=	=	=	=
ISO	=	√	√	?
SSO	=	√		
NCE	(=)	√		

= : IND-CCA EQUIVALENT

√ : ACHIEVED IN THE STANDARD MODEL

∪ : UNACHIEVABLE IN THE STANDARD MODEL

ACHIEVABILITY

	◇	○	★	⊛
IND	=	=	=	=
ISO	=	√	√	?
SSO	=	√	U	U
NCE	(=)	√	U	U

= : IND-CCA EQUIVALENT

√ : ACHIEVED IN THE STANDARD MODEL

U : UNACHIEVABLE IN THE STANDARD MODEL

Conclusion

So... which model of corruptions is the right one?

Conclusion

So... which model of corruptions is the right one?

Short answer: stop worrying and use IND-CCA(✪)

CONCLUSION

SO... WHICH MODEL OF CORRUPTIONS IS THE RIGHT ONE?

SHORT ANSWER: STOP WORRYING AND USE IND-CCA(★)

... UNLESS YOU NEED TO OPEN CHALLENGES
AND STAY SINGLE-CHALLENGE-BIT

CONCLUSION

SO... WHICH MODEL OF CORRUPTIONS IS THE RIGHT ONE?

SHORT ANSWER: STOP WORRYING AND USE IND-CCA(★)

... UNLESS YOU NEED TO OPEN CHALLENGES
AND STAY SINGLE-CHALLENGE-BIT

THEN: USE IND-CCA(★)!

CONCLUSION

SO... WHICH MODEL OF CORRUPTIONS IS THE RIGHT ONE?

SHORT ANSWER: STOP WORRYING AND USE IND-CCA(★)

... UNLESS YOU NEED TO OPEN CHALLENGES
AND STAY SINGLE-CHALLENGE-BIT

THEN: USE ISO-CCA(★)!

... UNLESS YOUR MESSAGE SPACE IS NOT
EFFICIENTLY CONDITIONALLY RESAMPLEABLE

Conclusion

So... which model of corruptions is the right one?

Short answer: Stop worrying and use IND-CCA(⋆)

... unless you need to open challenges
and stay single-challenge-bit

Then: use ISO-CCA⋆!

... unless your message space is not
efficiently conditionally resampleable

Then: use SSO-CCA⋆!

Conclusion

So... which model of corruptions is the right one?

Short answer: Stop worrying and use IND-CCA(⋆)

... unless you need to open challenges
and stay single-challenge-bit

Then: use ISO-CCA(⋆)!

... unless your message space is not
efficiently conditionally resampleable

Then: use SSO-CCA(⋆)!

... unless you need to avoid message sampling

Conclusion

So... which model of corruptions is the right one?

Short answer: Stop worrying and use IND-CCA(★)

... unless you need to open challenges
AND stay single-challenge-bit

Then: use ISO-CCA★!

... Unless your message space is not
efficiently conditionally resampleable

Then: use SSO-CCA★!

... unless you need to avoid message sampling

Then: use NCE-CCA★!

Thank You!

Questions?

Hans.Heum@ntnu.no

New Discord server:
Understanding Chen's Algorithm

bitly

Already >350 members!