Anamorphic Encryption, Revisited

Fabio Banfi[1] Konstantin Gegier[2] Martin Hirt[2]

Ueli Maurer[2] Guilherme Rito[3]

[1]Zühlke Engineering AG, Switzerland

[2]ETH Zurich, Switzerland

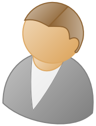
[3]Ruhr-Universität Bochum, Germany

EUROCRYPT 2024
May 27, Zurich, Switzerland

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

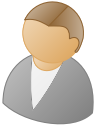
Alice



Bob

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

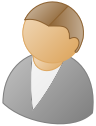

Alice


Bob

Bob uses a *well-established* PKE $\mathbf{\Pi} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]


Alice

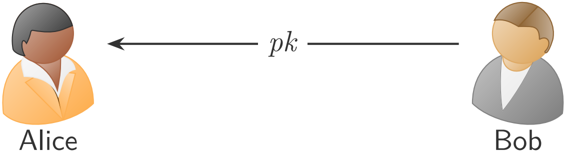

Bob

$(sk, pk) \leftarrow \textbf{Gen}()$

Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

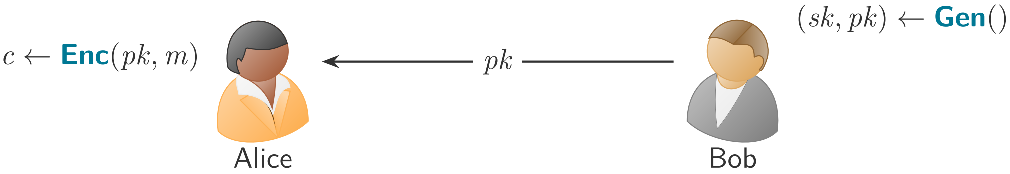
(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$(sk, pk) \leftarrow \textbf{Gen}()$



Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

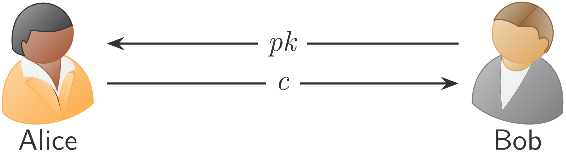


Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$(sk, pk) \leftarrow \textbf{Gen}()$

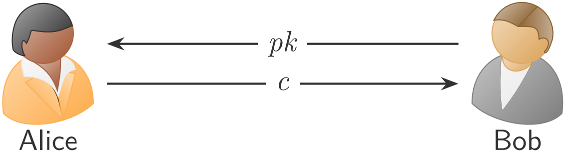
$c \leftarrow \textbf{Enc}(pk, m)$



Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$c \leftarrow \textbf{Enc}(pk, m)$

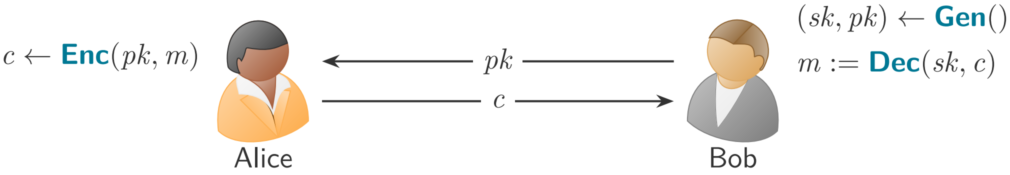


$(sk, pk) \leftarrow \textbf{Gen}()$

$m := \textbf{Dec}(sk, c)$

Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

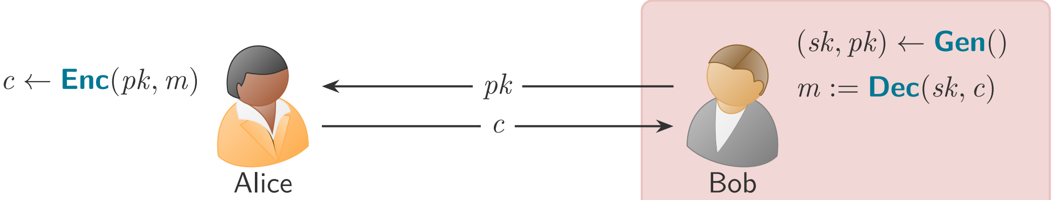
(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]



Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]



Bob uses a *well-established* PKE $\boldsymbol{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

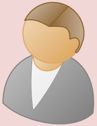
(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$c \leftarrow \textbf{Enc}(pk, m)$

Alice

$\longleftarrow pk \longleftarrow$

$\longrightarrow c \longrightarrow$



Bob

$(sk, pk) \leftarrow \textbf{Gen}()$

$m := \textbf{Dec}(sk, c)$

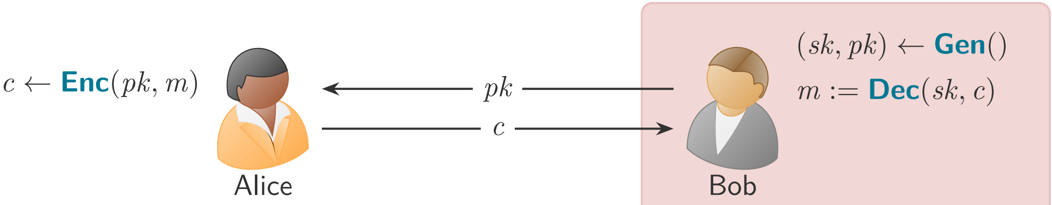
Bob uses a *well-established* PKE $\boldsymbol{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use $\boldsymbol{\Pi}$, but must surrender sk to D

Dictator

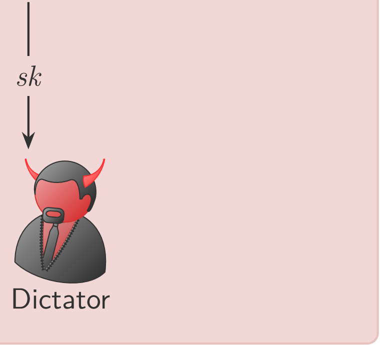
(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]



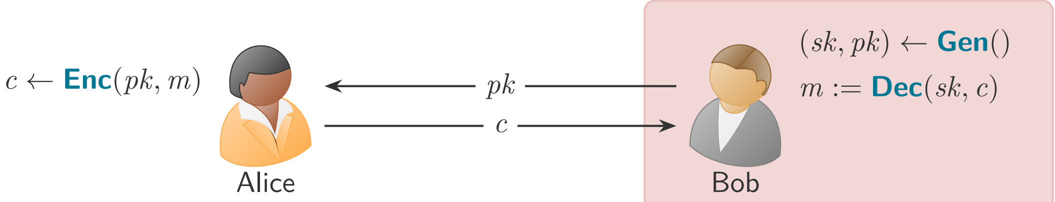
Bob uses a *well-established* PKE $\mathbf{\Pi} = ($**Gen**, **Enc**, **Dec**$)$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use $\mathbf{\Pi}$, but must surrender sk to D



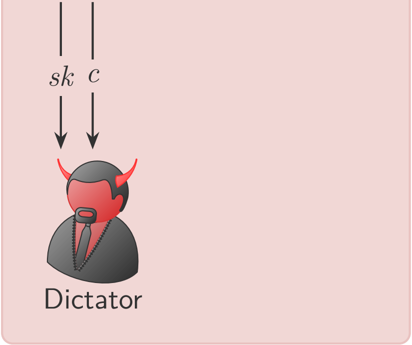
(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]



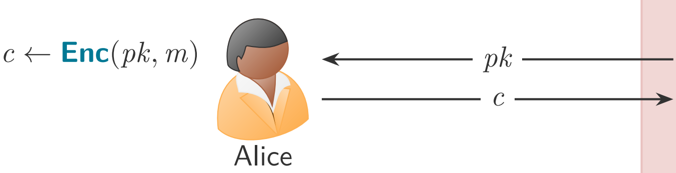
Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

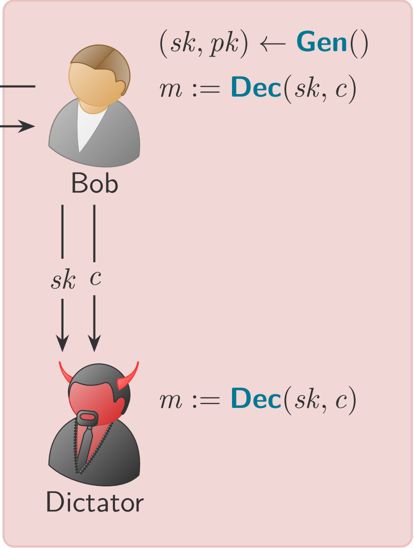
Suddenly, Bob's country is led by a **dictator** D!

Bob can still use $\mathbf{\Pi}$, but must surrender sk to D



(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]



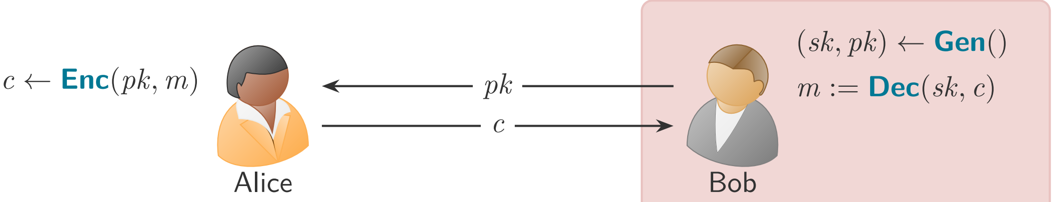


Bob uses a *well-established* PKE $\Pi = ($**Gen**, **Enc**, **Dec**$)$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use Π, but must surrender sk to D

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

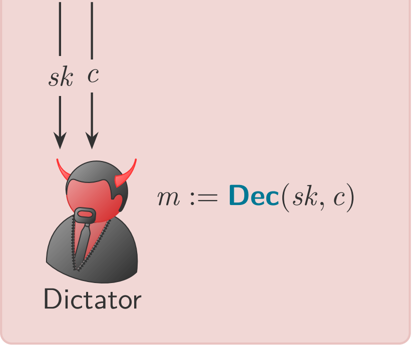


Bob uses a *well-established* PKE $\Pi = (\textsf{Gen}, \textsf{Enc}, \textsf{Dec})$

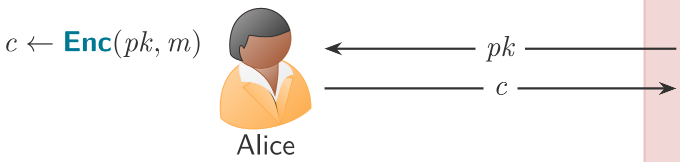
Suddenly, Bob's country is led by a **dictator** D!

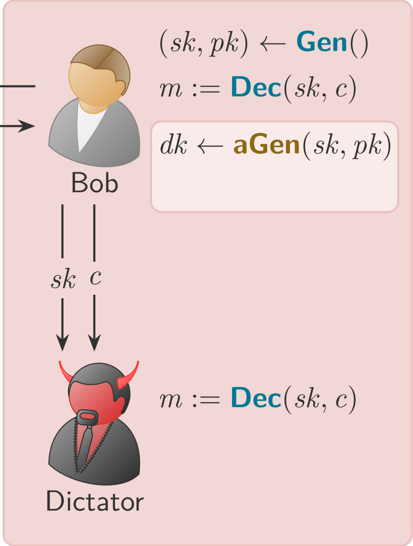
Bob can still use Π, but must surrender sk to D

Use **anamorphic extension** $\Sigma = (\textsf{aGen}, \textsf{aEnc}, \textsf{aDec})$



(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]





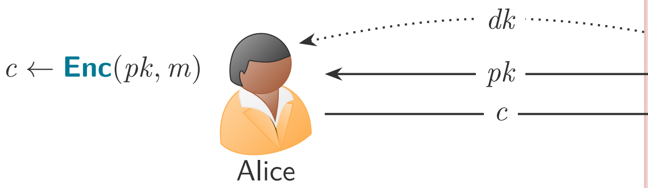
Bob uses a *well-established* PKE $\Pi = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

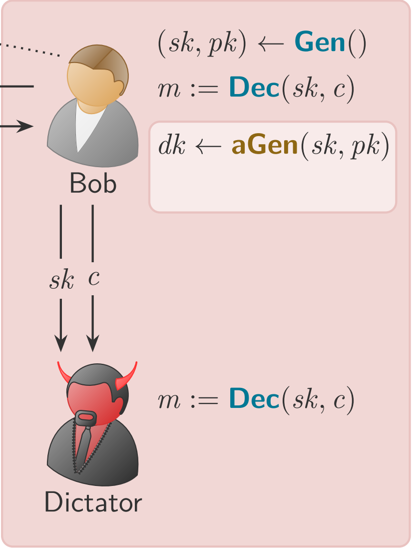
Suddenly, Bob's country is led by a **dictator** D!

Bob can still use Π, but must surrender sk to D

Use **anamorphic extension** $\Sigma = (\textbf{aGen}, \textbf{aEnc}, \textbf{aDec})$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]





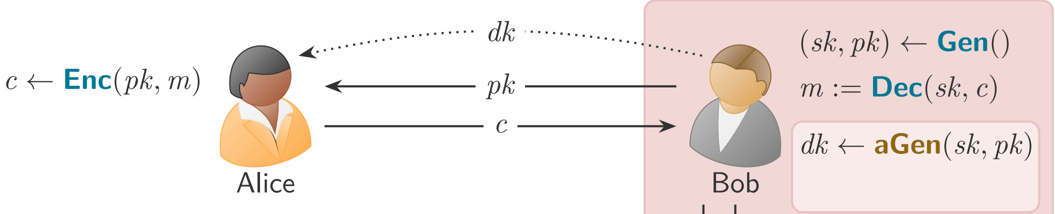
Bob uses a *well-established* PKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use Π, but must surrender sk to D

Use **anamorphic extension** $\Sigma = (\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]



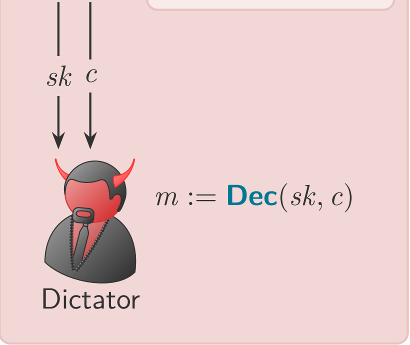
Bob uses a *well-established* PKE $\Pi = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

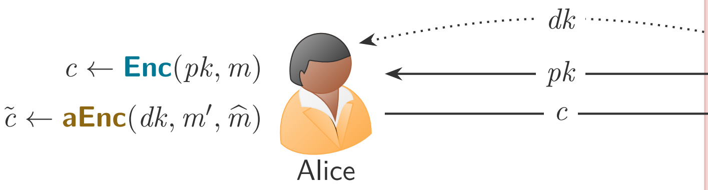
Bob can still use Π, but must surrender sk to D

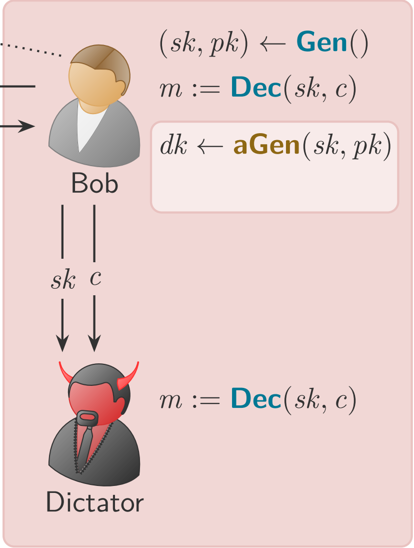
Use **anamorphic extension** $\Sigma = (\textbf{aGen}, \textbf{aEnc}, \textbf{aDec})$

With **double key** dk, Alice embeds **covert message** $\widehat{m}$



(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]





Bob uses a *well-established* PKE $\Pi = ($**Gen**, **Enc**, **Dec**$)$

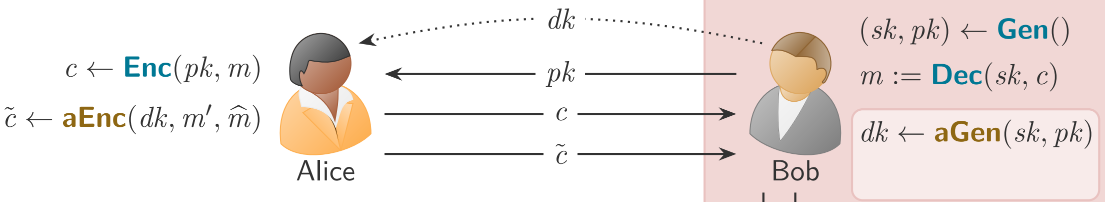
Suddenly, Bob's country is led by a **dictator** D!

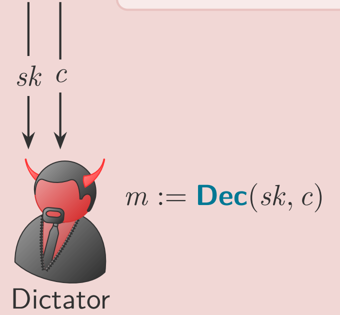
Bob can still use Π, but must surrender sk to D

Use **anamorphic extension** $\Sigma = ($**aGen**, **aEnc**, **aDec**$)$

With **double key** dk, Alice embeds **covert message** $\widehat{m}$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]





Bob uses a *well-established* PKE $\mathbf{\Pi} = ($**Gen**, **Enc**, **Dec**$)$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use $\mathbf{\Pi}$, but must surrender sk to D

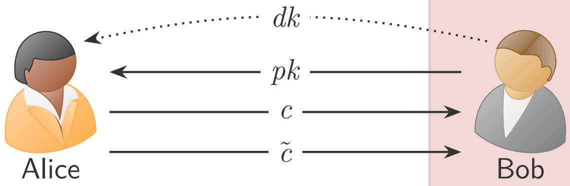
Use **anamorphic extension** $\mathbf{\Sigma} = ($**aGen**, **aEnc**, **aDec**$)$

With **double key** dk, Alice embeds **covert message** $\widehat{m}$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$c \leftarrow \textsf{Enc}(pk, m)$

$\tilde{c} \leftarrow \textsf{aEnc}(dk, m', \widehat{m})$

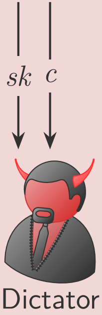


$(sk, pk) \leftarrow \textsf{Gen}()$

$m := \textsf{Dec}(sk, c)$

$dk \leftarrow \textsf{aGen}(sk, pk)$

$\widehat{m} := \textsf{aDec}(dk, \tilde{c})$



$m := \textsf{Dec}(sk, c)$

Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textsf{Gen}, \textsf{Enc}, \textsf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use $\mathbf{\Pi}$, but must surrender sk to D

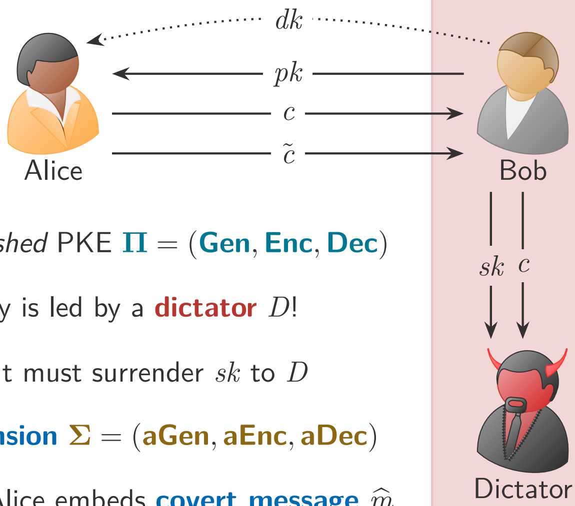
Use **anamorphic extension** $\mathbf{\Sigma} = (\textsf{aGen}, \textsf{aEnc}, \textsf{aDec})$

With **double key** dk, Alice embeds **covert message** $\widehat{m}$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$c \leftarrow \textsf{Enc}(pk, m)$

$\tilde{c} \leftarrow \textsf{aEnc}(dk, m', \widehat{m})$



$(sk, pk) \leftarrow \textsf{Gen}()$

$m := \textsf{Dec}(sk, c)$

$dk \leftarrow \textsf{aGen}(sk, pk)$

$\widehat{m} := \textsf{aDec}(dk, \tilde{c})$

$m' := \textsf{Dec}(sk, \tilde{c})$

$m := \textsf{Dec}(sk, c)$

Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textsf{Gen}, \textsf{Enc}, \textsf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use $\mathbf{\Pi}$, but must surrender sk to D

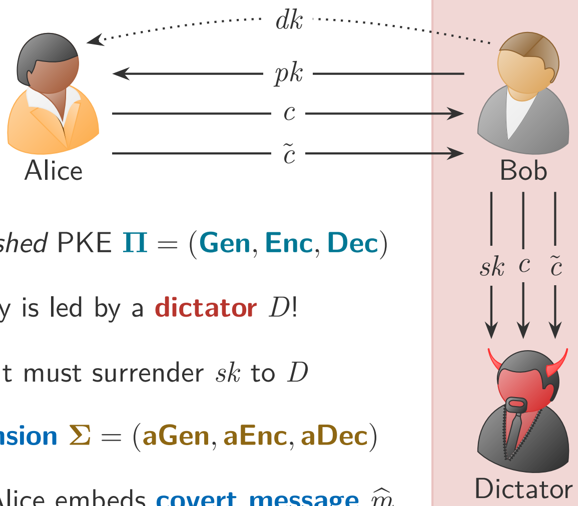
Use **anamorphic extension** $\mathbf{\Sigma} = (\textsf{aGen}, \textsf{aEnc}, \textsf{aDec})$

With **double key** dk, Alice embeds **covert message** $\widehat{m}$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$c \leftarrow \textbf{Enc}(pk, m)$

$\tilde{c} \leftarrow \textbf{aEnc}(dk, m', \widehat{m})$



$(sk, pk) \leftarrow \textbf{Gen}()$

$m := \textbf{Dec}(sk, c)$

$dk \leftarrow \textbf{aGen}(sk, pk)$

$\widehat{m} := \textbf{aDec}(dk, \tilde{c})$

$m' := \textbf{Dec}(sk, \tilde{c})$

$m := \textbf{Dec}(sk, c)$

Bob uses a *well-established* PKE $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use $\mathbf{\Pi}$, but must surrender sk to D

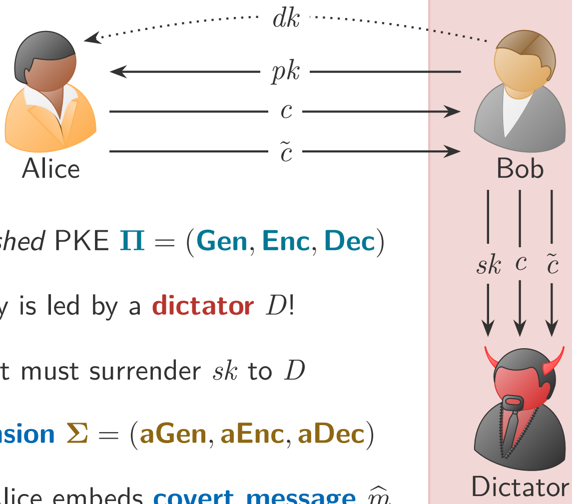
Use **anamorphic extension** $\mathbf{\Sigma} = (\textbf{aGen}, \textbf{aEnc}, \textbf{aDec})$

With **double key** dk, Alice embeds **covert message** $\widehat{m}$

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$c \leftarrow \textbf{Enc}(pk, m)$

$\tilde{c} \leftarrow \textbf{aEnc}(dk, m', \widehat{m})$



$(sk, pk) \leftarrow \textbf{Gen}()$

$m := \textbf{Dec}(sk, c)$

$dk \leftarrow \textbf{aGen}(sk, pk)$

$\widehat{m} := \textbf{aDec}(dk, \tilde{c})$

$m' := \textbf{Dec}(sk, \tilde{c})$

$m := \textbf{Dec}(sk, c)$

$m' := \textbf{Dec}(sk, \tilde{c})$

Bob uses a *well-established* PKE $\Pi = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use Π, but must surrender sk to D

Use **anamorphic extension** $\Sigma = (\textbf{aGen}, \textbf{aEnc}, \textbf{aDec})$

With **double key** dk, Alice embeds **covert message** $\widehat{m}$

Decoupling Keys & Security

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk):

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation:

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages:

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow \textbf{aGen}()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow \textbf{aGen}(sk, pk)$ to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:**

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow$ **Enc**(pk, m);

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow \textbf{aGen}()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow \textbf{aGen}(sk, pk)$ to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \textbf{Enc}(pk, m)$; $m := \textbf{Dec}(sk, c)$

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \textbf{Enc}(pk, m);$ $m := \textbf{Dec}(sk, c)$
- **Anamorphic:**

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow \textbf{aGen}()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow \textbf{aGen}(sk, pk)$ to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \textbf{Enc}(pk, m)$; $m := \textbf{Dec}(sk, c)$
- **Anamorphic:** $\tilde{c} \leftarrow \textbf{aEnc}(dk, m, \widehat{m})$;

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \textbf{Enc}(pk, m)$; $m := \textbf{Dec}(sk, c)$
- **Anamorphic:** $\tilde{c} \leftarrow \textbf{aEnc}(dk, m, \widehat{m})$; $\widehat{m} := \textbf{aDec}(dk, \tilde{c})$

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \mathbf{Enc}(pk, m)$; $m := \mathbf{Dec}(sk, c)$
- **Anamorphic:** $\tilde{c} \leftarrow \mathbf{aEnc}(dk, m, \widehat{m})$; $\widehat{m} := \mathbf{aDec}(dk, \tilde{c})$, $m := \mathbf{Dec}(sk, \tilde{c})$

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \mathbf{Enc}(pk, m)$; $m := \mathbf{Dec}(sk, c)$
- **Anamorphic:** $\tilde{c} \leftarrow \mathbf{aEnc}(dk, m, \widehat{m})$; $\widehat{m} := \mathbf{aDec}(dk, \tilde{c})$, $m := \mathbf{Dec}(sk, \tilde{c})$

Security:

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow \mathbf{aGen}()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow \mathbf{aGen}(sk, pk)$ to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \mathbf{Enc}(pk, m)$; $m := \mathbf{Dec}(sk, c)$
- **Anamorphic:** $\tilde{c} \leftarrow \mathbf{aEnc}(dk, m, \widehat{m})$; $\widehat{m} := \mathbf{aDec}(dk, \tilde{c})$, $m := \mathbf{Dec}(sk, \tilde{c})$

Security: The two modes must be indistinguishable: $\tilde{c} \approx c$!

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \mathbf{Enc}(pk, m)$; $m := \mathbf{Dec}(sk, c)$
- **Anamorphic:** $\tilde{c} \leftarrow \mathbf{aEnc}(dk, m, \widehat{m})$; $\widehat{m} := \mathbf{aDec}(dk, \tilde{c})$, $m := \mathbf{Dec}(sk, \tilde{c})$

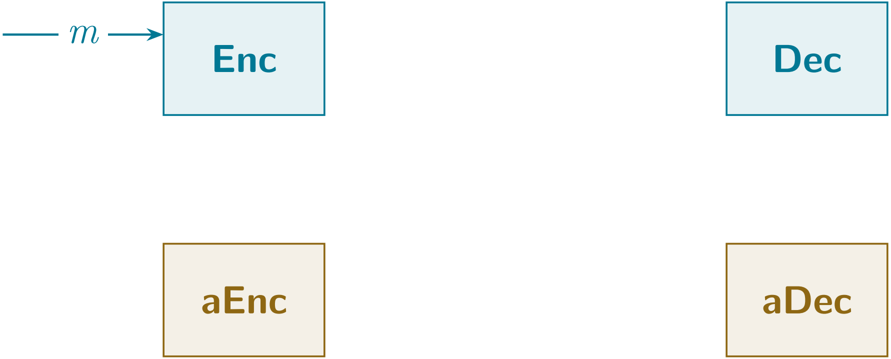
Security: The two modes must be indistinguishable: $\tilde{c} \approx c$! **Is this all?**

Using Anamorphic Encryption

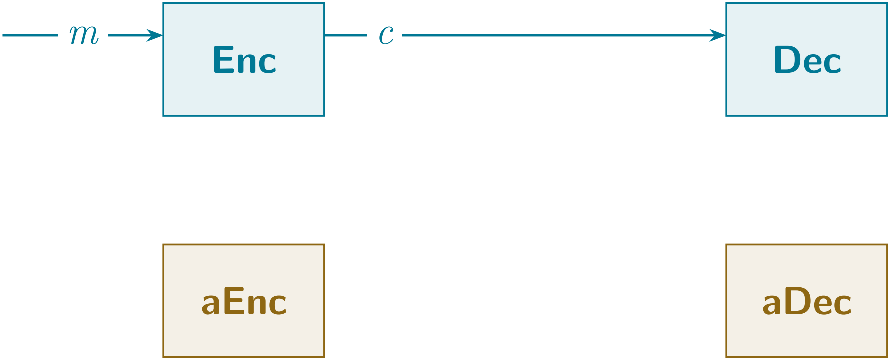
Using Anamorphic Encryption

Using Anamorphic Encryption

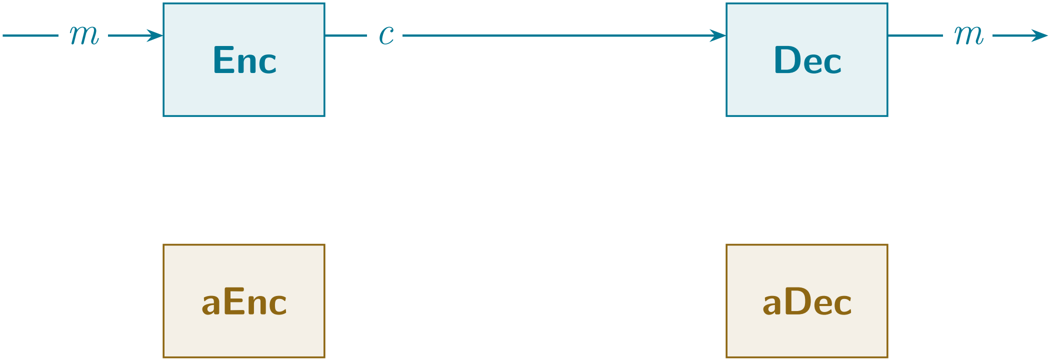
Using Anamorphic Encryption



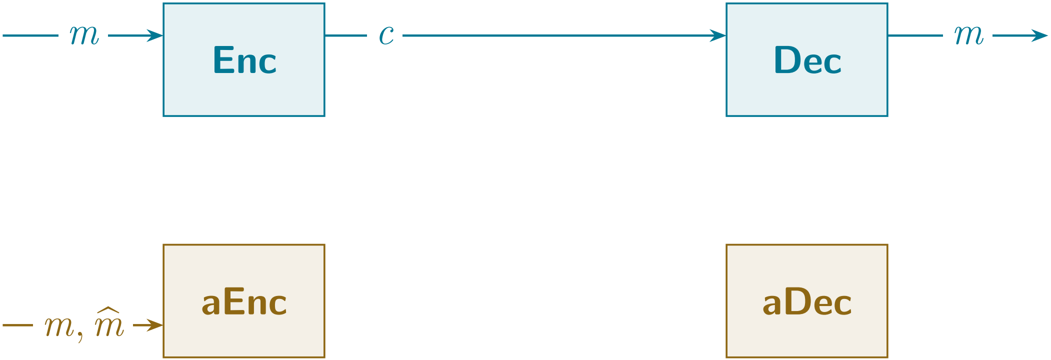
Using Anamorphic Encryption



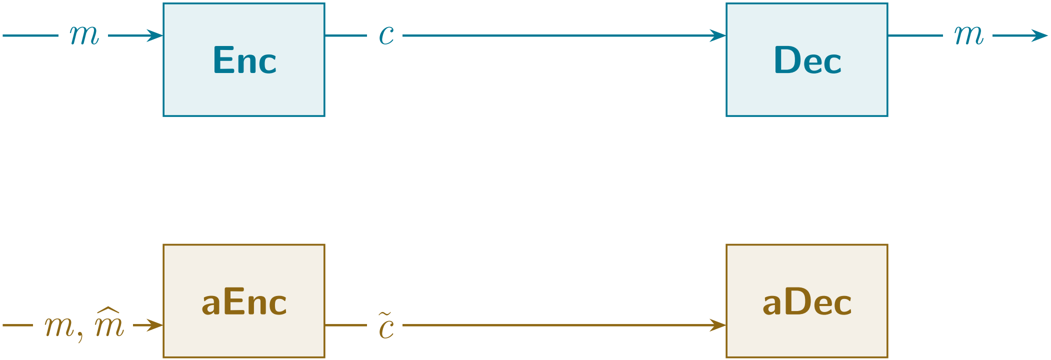
Using Anamorphic Encryption



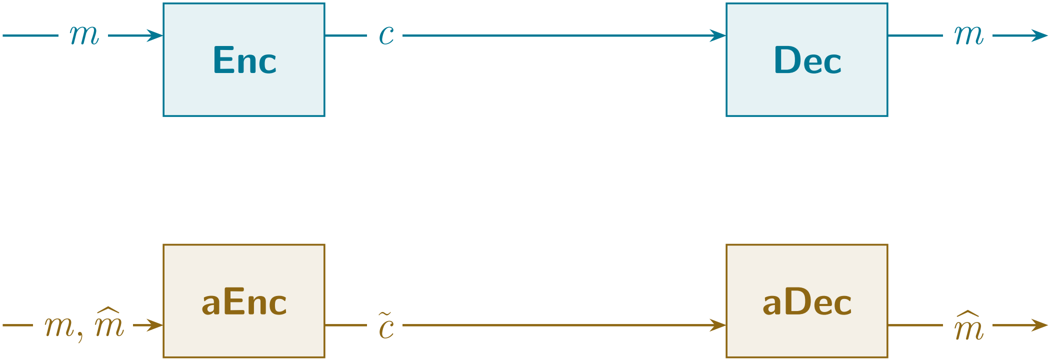
Using Anamorphic Encryption



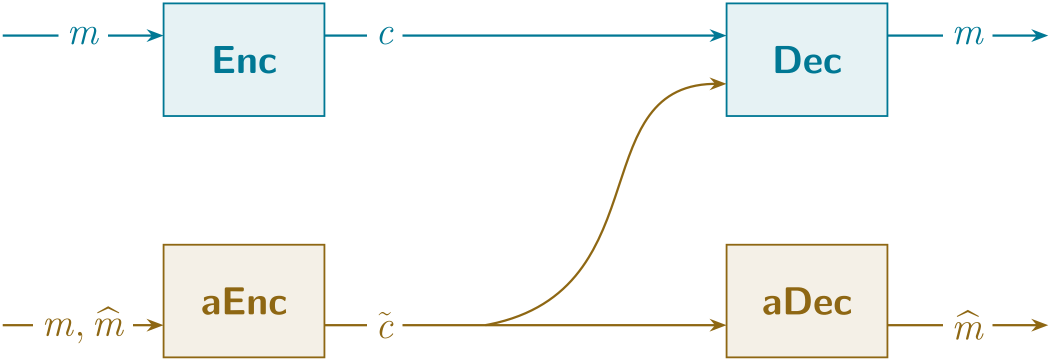
Using Anamorphic Encryption



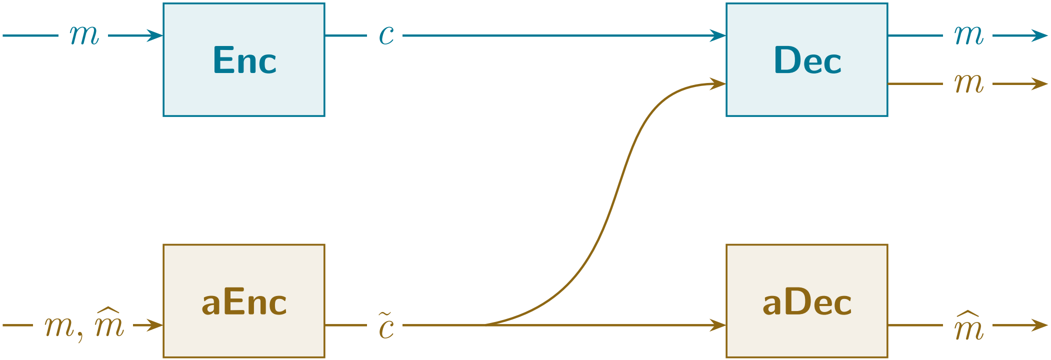
Using Anamorphic Encryption



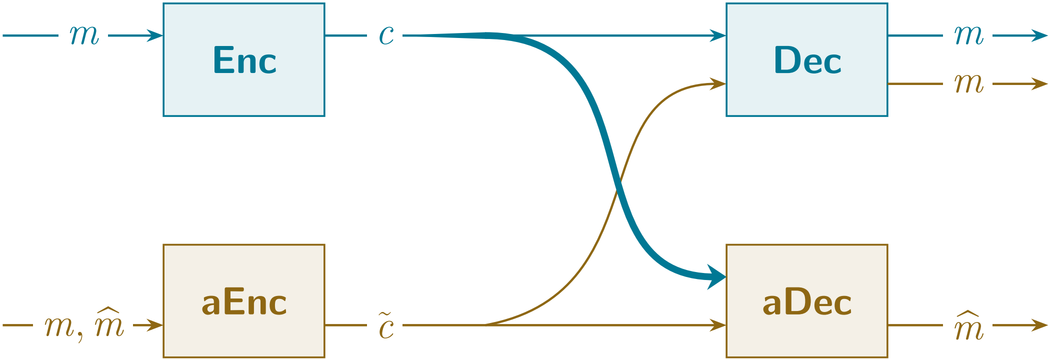
Using Anamorphic Encryption



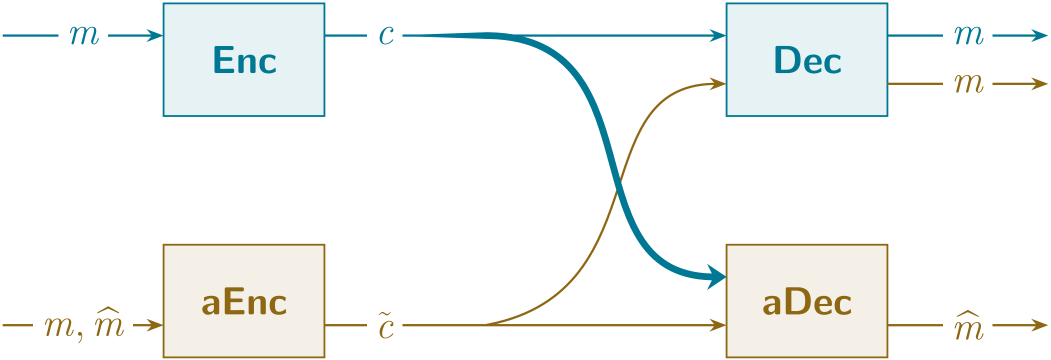
Using Anamorphic Encryption



Using Anamorphic Encryption

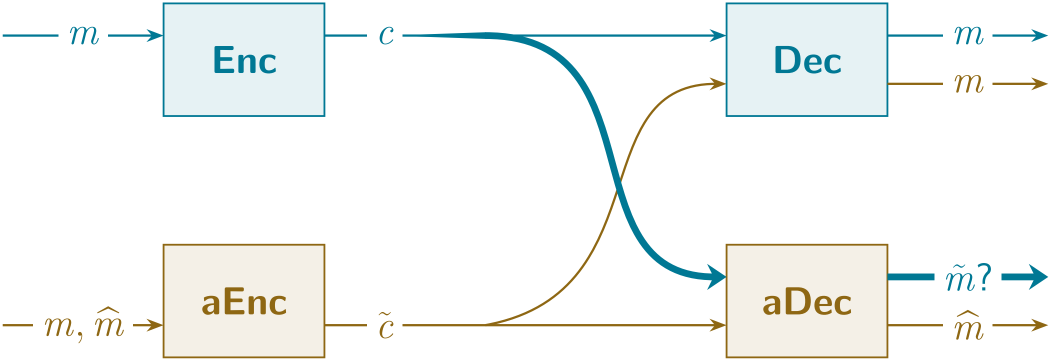


Using Anamorphic Encryption



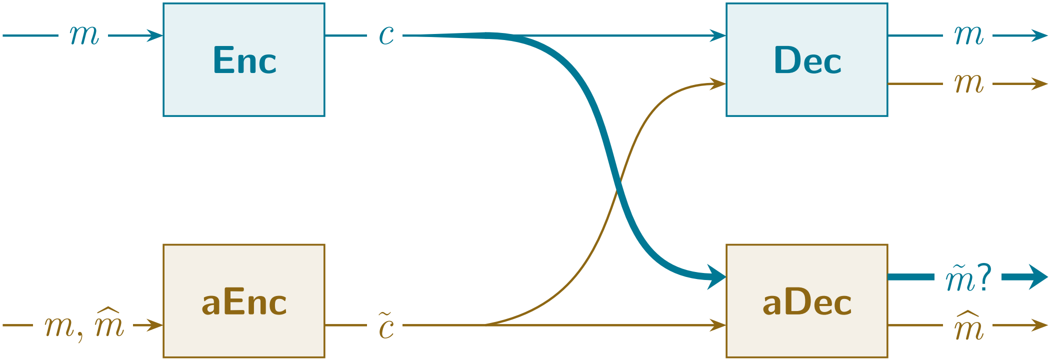
This case was not considered!

Using Anamorphic Encryption



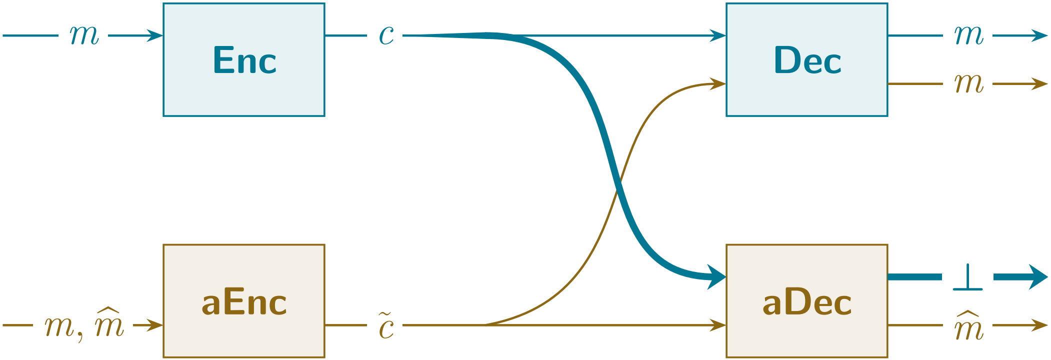
This case was not considered!

Using Anamorphic Encryption



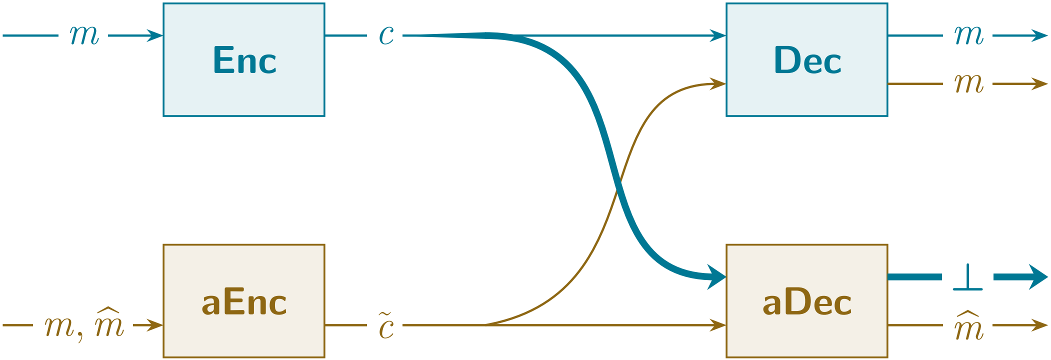
This case was not considered! Need to signal *"no covert message"*

Using Anamorphic Encryption



This case was not considered! Need to signal *"no covert message"*

Using Anamorphic Encryption



This case was not considered! Need to signal *"no covert message"* $\Longrightarrow$ **Robustness**

Why Robustness?

Why Robustness?

- **Functionality:**

Why Robustness?

- **Functionality:** Bob might use $\mathbf{\Pi}$ *regularly* and $\mathbf{\Sigma}$ *sporadically*

Why Robustness?

- **Functionality:** Bob might use $\mathbf{\Pi}$ *regularly* and $\mathbf{\Sigma}$ *sporadically*

  Therefore, more often than not:

Why Robustness?

- **Functionality:** Bob might use $\mathbf{\Pi}$ *regularly* and $\mathbf{\Sigma}$ *sporadically*

  Therefore, more often than not: *ciphertexts carry <u>no</u> (intentional) covert message!*

Why Robustness?

- **Functionality:** Bob might use $\mathbf{\Pi}$ *regularly* and $\mathbf{\Sigma}$ *sporadically*

  Therefore, more often than not: *ciphertexts carry <u>no</u> (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

Why Robustness?

- **Functionality:** Bob might use Π *regularly* and Σ *sporadically*

  Therefore, more often than not: *ciphertexts carry <u>no</u> (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

  Is this satisfactory?

Why Robustness?

- **Functionality:** Bob might use Π *regularly* and Σ *sporadically*

  Therefore, more often than not: *ciphertexts carry <u>no</u> (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

  Is this satisfactory? **No!**

Why Robustness?

- **Functionality:** Bob might use Π *regularly* and Σ *sporadically*

  Therefore, more often than not: *ciphertexts carry <u>no</u> (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

  Is this satisfactory? **No!**

- **Security:**

Why Robustness?

- **Functionality:** Bob might use Π *regularly* and Σ *sporadically*

  Therefore, more often than not: *ciphertexts carry no (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

  Is this satisfactory? **No!**

- **Security:** it could get even worse!

Why Robustness?

- **Functionality:** Bob might use $\mathbf{\Pi}$ *regularly* and $\mathbf{\Sigma}$ *sporadically*

  Therefore, more often than not: *ciphertexts carry <u>no</u> (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

  Is this satisfactory? **No!**

- **Security:** it could get even worse!

  Without robustness, D might find out that Bob has established a covert channel!

Why Robustness?

- **Functionality:** Bob might use Π *regularly* and Σ *sporadically*

  Therefore, more often than not: *ciphertexts carry no (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

  Is this satisfactory? **No!**

- **Security:** it could get even worse!

  Without robustness, D might find out that Bob has established a covert channel!

  1. Send encryption of random message to Bob

Why Robustness?

- **Functionality:** Bob might use $\boldsymbol{\Pi}$ *regularly* and $\boldsymbol{\Sigma}$ *sporadically*

  Therefore, more often than not: *ciphertexts carry no (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

  Is this satisfactory? **No!**

- **Security:** it could get even worse!

  Without robustness, D might find out that Bob has established a covert channel!

  1. Send encryption of random message to Bob
  2. If D is lucky, the covert message is not "garbage" and Bob detectably reacts!

Construction Σ_1: A Naive Robust Scheme

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter $\texttt{ctr}$, use r to encrypt m into $\tilde{c}$:

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter $\mathtt{ctr}$, use r to encrypt m into $\tilde{c}$:

$$\textbf{aEnc}(dk, m, \widehat{m}) := \textbf{Enc}(pk, m; F_K(\mathtt{ctr} \| \widehat{m}))$$

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter $\texttt{ctr}$, use r to encrypt m into $\tilde{c}$:

  $$\textbf{aEnc}(dk, m, \widehat{m}) := \textbf{Enc}(pk, m; F_K(\texttt{ctr} \| \widehat{m}))$$

- **Bob:** decrypt $\tilde{c}$ into m, and check which $\widehat{m} \in \widehat{\mathcal{M}}$ yields $\tilde{c}$:

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter $\texttt{ctr}$, use r to encrypt m into $\tilde{c}$:

  $$\textbf{aEnc}(dk, m, \widehat{m}) := \textbf{Enc}(pk, m; F_K(\texttt{ctr} \| \widehat{m}))$$

- **Bob:** decrypt $\tilde{c}$ into m, and check which $\widehat{m} \in \widehat{\mathcal{M}}$ yields $\tilde{c}$:

  $$\textbf{aDec}(dk, \tilde{c}) := \{ \text{let } m := \textbf{Dec}(sk, \tilde{c});$$

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter $\mathtt{ctr}$, use r to encrypt m into $\tilde{c}$:

  $$\textbf{aEnc}(dk, m, \widehat{m}) := \textbf{Enc}(pk, m; F_K(\mathtt{ctr} \| \widehat{m}))$$

- **Bob:** decrypt $\tilde{c}$ into m, and check which $\widehat{m} \in \widehat{\mathcal{M}}$ yields $\tilde{c}$:

  $$\textbf{aDec}(dk, \tilde{c}) := \{\, \text{let } m := \textbf{Dec}(sk, \tilde{c});$$
  $$\text{find } \widehat{m} \text{ s.t. } \textbf{Enc}(pk, m; F_K(\mathtt{ctr} \| \widehat{m})) = \tilde{c} \text{ or return } \perp; \}$$

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter $\underline{\texttt{ctr}}$, use r to encrypt m into $\tilde{c}$:

$$\textbf{aEnc}(dk, m, \widehat{m}) := \textbf{Enc}(pk, m; F_K(\underline{\texttt{ctr}} \| \widehat{m}))$$

- **Bob:** decrypt $\tilde{c}$ into m, and check which $\widehat{m} \in \widehat{\mathcal{M}}$ yields $\tilde{c}$:

$$\textbf{aDec}(dk, \tilde{c}) := \{ \text{ let } m := \textbf{Dec}(sk, \tilde{c});$$

$$\text{find } \widehat{m} \text{ s.t. } \textbf{Enc}(pk, m; F_K(\underline{\texttt{ctr}} \| \widehat{m})) = \tilde{c} \text{ or return } \perp; \}$$

Problem: Alice and Bob need to keep **synchronized** counters and **aDec** uses **Dec**!

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter $\underline{\texttt{ctr}}$, use r to encrypt m into $\tilde{c}$:

$$\textbf{aEnc}(dk, m, \widehat{m}) := \textbf{Enc}(pk, m; F_K(\underline{\texttt{ctr}} \| \widehat{m}))$$

- **Bob:** decrypt $\tilde{c}$ into m, and check which $\widehat{m} \in \widehat{\mathcal{M}}$ yields $\tilde{c}$:

$$\textbf{aDec}(dk, \tilde{c}) := \{ \text{let } m := \textbf{Dec}(sk, \tilde{c});$$

$$\text{find } \widehat{m} \text{ s.t. } \textbf{Enc}(pk, m; F_K(\underline{\texttt{ctr}} \| \widehat{m})) = \tilde{c} \text{ or return } \perp; \}$$

Problem: Alice and Bob need to keep **synchronized** counters and **aDec** uses **Dec**!

Solution: use PKEs with a special property:

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter $\underline{\texttt{ctr}}$, use r to encrypt m into $\tilde{c}$:

$$\mathbf{aEnc}(dk, m, \widehat{m}) := \mathbf{Enc}(pk, m; F_K(\underline{\texttt{ctr}} \| \widehat{m}))$$

- **Bob:** decrypt $\tilde{c}$ into m, and check which $\widehat{m} \in \widehat{\mathcal{M}}$ yields $\tilde{c}$:

$$\mathbf{aDec}(dk, \tilde{c}) := \{ \text{let } m := \mathbf{Dec}(sk, \tilde{c});$$
$$\text{find } \widehat{m} \text{ s.t. } \mathbf{Enc}(pk, m; F_K(\underline{\texttt{ctr}} \| \widehat{m})) = \tilde{c} \text{ or return } \perp; \}$$

Problem: Alice and Bob need to keep **synchronized** counters and **aDec** uses **Dec**!

Solution: use PKEs with a special property: **Selective Randomness Recoverability**

Selective Randomness Recoverability (SRR)

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$ is SRR if the following conditions are met:

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

Selective Randomness Recoverability (SRR)

PKE scheme $\boldsymbol{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*:

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \mathsf{Enc}(pk, m; r)$ we want $c = (A, B)$ where:

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \mathbf{Enc}(pk, m; r)$ we want $c = (A, B)$ where:

- Part A depends on pk, m, and r:

Selective Randomness Recoverability (SRR)

PKE scheme $\boldsymbol{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \textbf{Enc}(pk, m; r)$ we want $c = (A, B)$ where:

- Part A depends on pk, m, and r: $A = \alpha(pk, m, r)$

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\textsf{\textbf{Gen}}, \textsf{\textbf{Enc}}, \textsf{\textbf{Dec}})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \textsf{\textbf{Enc}}(pk, m; r)$ we want $c = (A, B)$ where:

- Part A depends on pk, m, and r: $A = \alpha(pk, m, r)$
- Part B depends **only** on r:

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \textbf{Enc}(pk, m; r)$ we want $c = (A, B)$ where:

- Part A depends on pk, m, and r: $A = \alpha(pk, m, r)$
- Part B depends **only** on r: $B = \beta(r)$

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \mathbf{Enc}(pk, m; r)$ we want $c = (A, B)$ where:

- Part A depends on pk, m, and r: $A = \alpha(pk, m, r)$
- Part B depends **only** on r: $B = \beta(r)$

(iii) Can compute $\beta(a)$ from $\beta(a \star b)$ and b:

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \mathbf{Enc}(pk, m; r)$ we want $c = (A, B)$ where:

- Part A depends on pk, m, and r: $A = \alpha(pk, m, r)$
- Part B depends **only** on r: $B = \beta(r)$

(iii) Can compute $\beta(a)$ from $\beta(a \star b)$ and b:

- There exists an efficiently computable function γ s.t. $\gamma(\beta(a \star b), b) = \beta(a)$

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \textbf{Enc}(pk, m; r)$ we want $c = (A, B)$ where:

- Part A depends on pk, m, and r: $A = \alpha(pk, m, r)$
- Part B depends **only** on r: $B = \beta(r)$

(iii) Can compute $\beta(a)$ from $\beta(a \star b)$ and b:

- There exists an efficiently computable function γ s.t. $\gamma(\beta(a \star b), b) = \beta(a)$

Both **ElGamal** and **Cramer-Shoup** are SRR

Construction Σ_2: Using an SRR Scheme

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$:

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$: set $\mathbf{T}[\beta(\widehat{m})] := \widehat{m}$ for each $\widehat{m} \in \widehat{\mathcal{M}}$

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$: set $\mathbf{T}[\beta(\widehat{m})] := \widehat{m}$ for each $\widehat{m} \in \widehat{\mathcal{M}}$
- **Alice:** use $F_K(\mathtt{ctr})$ as otp for $\widehat{m}$ and use result as r to enc. m into $\tilde{c} = (A, B)$:

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$: set $\mathbf{T}[\beta(\widehat{m})] := \widehat{m}$ for each $\widehat{m} \in \widehat{\mathcal{M}}$
- **Alice:** use $F_K(\mathtt{ctr})$ as otp for $\widehat{m}$ and use result as r to enc. m into $\tilde{c} = (A, B)$:

$$\textbf{aEnc}(dk, m, \widehat{m}; \mathtt{ctr}) := \textbf{Enc}(pk, m; \widehat{m} \star F_K(\mathtt{ctr}))$$

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$: set $\mathbf{T}[\beta(\widehat{m})] := \widehat{m}$ for each $\widehat{m} \in \widehat{\mathcal{M}}$
- **Alice:** use $F_K(\texttt{ctr})$ as otp for $\widehat{m}$ and use result as r to enc. m into $\tilde{c} = (A, B)$:

  $$\textbf{aEnc}(dk, m, \widehat{m}; \texttt{ctr}) := \textbf{Enc}(pk, m; \widehat{m} \star F_K(\texttt{ctr}))$$

- **Bob:** use F_K and γ to extract $\widehat{m}$ from B:

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$: set $\mathbf{T}[\beta(\widehat{m})] := \widehat{m}$ for each $\widehat{m} \in \widehat{\mathcal{M}}$
- **Alice:** use $F_K(\texttt{ctr})$ as otp for $\widehat{m}$ and use result as r to enc. m into $\tilde{c} = (A, B)$:

$$\textbf{aEnc}(dk, m, \widehat{m}; \texttt{ctr}) := \textbf{Enc}(pk, m; \widehat{m} \star F_K(\texttt{ctr}))$$

- **Bob:** use F_K and γ to extract $\widehat{m}$ from B:

$$\textbf{aDec}(dk, (A, B); \texttt{ctr}) := \mathbf{T}[\gamma(B, F_K(\texttt{ctr}))]$$

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$: set $\mathbf{T}[\beta(\widehat{m})] := \widehat{m}$ for each $\widehat{m} \in \widehat{\mathcal{M}}$
- **Alice:** use $F_K(\texttt{ctr})$ as otp for $\widehat{m}$ and use result as r to enc. m into $\tilde{c} = (A, B)$:

$$\textbf{aEnc}(dk, m, \widehat{m}; \texttt{ctr}) := \textbf{Enc}(pk, m; \widehat{m} \star F_K(\texttt{ctr}))$$

- **Bob:** use F_K and γ to extract $\widehat{m}$ from B:

$$\textbf{aDec}(dk, (A, B); \texttt{ctr}) := \mathbf{T}[\gamma(B, F_K(\texttt{ctr}))]$$

[**Dec** not needed!]

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$: set $\mathbf{T}[\beta(\widehat{m})] := \widehat{m}$ for each $\widehat{m} \in \widehat{\mathcal{M}}$
- **Alice:** use $F_K(\underline{\mathtt{ctr}})$ as otp for $\widehat{m}$ and use result as r to enc. m into $\tilde{c} = (A, B)$:

$$\mathbf{aEnc}(dk, m, \widehat{m}; \underline{\mathtt{ctr}}) := \mathbf{Enc}(pk, m; \widehat{m} \star F_K(\underline{\mathtt{ctr}}))$$

- **Bob:** use F_K and γ to extract $\widehat{m}$ from B:

$$\mathbf{aDec}(dk, (A, B); \underline{\mathtt{ctr}}) := \mathbf{T}[\gamma(B, F_K(\underline{\mathtt{ctr}}))] \qquad [\mathbf{Dec} \text{ not needed!}]$$

Still need to keep **synchronized** counters!

Construction Σ_3: Getting Rid of Synchronization

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random `ctr`, until can *partially* extract `ctr` from B via some function δ

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random `ctr`, until can *partially* extract `ctr` from B via some function δ

aEnc$(dk, m, \widehat{m})$:

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random `ctr`, until can *partially* extract `ctr` from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$,

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random $\texttt{ctr}$, until can *partially* extract $\texttt{ctr}$ from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\texttt{ctr} := \mathtt{x} \| \mathtt{y}$,

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random `ctr`, until can *partially* extract `ctr` from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\mathtt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\mathtt{ctr})$,

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random $\mathtt{ctr}$, until can *partially* extract $\mathtt{ctr}$ from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\mathtt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\mathtt{ctr})$, and $B := \beta(r)$

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random $\texttt{ctr}$, until can *partially* extract $\texttt{ctr}$ from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\texttt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\texttt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \mathtt{x}$,

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random $\texttt{ctr}$, until can *partially* extract $\texttt{ctr}$ from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\texttt{x}, \texttt{y}) \in [\sigma] \times [\tau]$, set $\texttt{ctr} := \texttt{x} \| \texttt{y}$, $r := \widehat{m} \star F_K(\texttt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \texttt{x}$, let r^* be the such first r

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random $\texttt{ctr}$, until can *partially* extract $\texttt{ctr}$ from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\texttt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\texttt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \mathtt{x}$, let r^* be the such first r
3. Return $(A, B) :=$ **Enc**$(pk, m; r^*)$

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random `ctr`, until can *partially* extract `ctr` from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\mathtt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\mathtt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \mathtt{x}$, let r^* be the such first r
3. Return $(A, B) := \mathbf{Enc}(pk, m; r^*)$

aDec$(dk, (A, B))$:

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random $\texttt{ctr}$, until can *partially* extract $\texttt{ctr}$ from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\texttt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\texttt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \mathtt{x}$, let r^* be the such first r
3. Return $(A, B) :=$ **Enc**$(pk, m; r^*)$

aDec$(dk, (A, B))$:

1. Set $\mathtt{x} := \delta(B)$

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random `ctr`, until can *partially* extract `ctr` from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\mathtt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\mathtt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \mathtt{x}$, let r^* be the such first r
3. Return $(A, B) := \textbf{Enc}(pk, m; r^*)$

aDec$(dk, (A, B))$:

1. Set $\mathtt{x} := \delta(B)$
2. For each possible value $\mathtt{y}$:

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random ctr, until can *partially* extract ctr from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\mathtt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\mathtt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \mathtt{x}$, let r^* be the such first r
3. Return $(A, B) := \mathbf{Enc}(pk, m; r^*)$

aDec$(dk, (A, B))$:

1. Set $\mathtt{x} := \delta(B)$
2. For each possible value y: if $\widehat{m} := \mathbf{T}[\gamma(B, F_K(\mathtt{x} \| \mathtt{y}))] \neq \perp$, return $\widehat{m}$

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random $\texttt{ctr}$, until can *partially* extract $\texttt{ctr}$ from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\texttt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\texttt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \mathtt{x}$, let r^* be the such first r
3. Return $(A, B) :=$ **Enc**$(pk, m; r^*)$

aDec$(dk, (A, B))$:

1. Set $\mathtt{x} := \delta(B)$
2. For each possible value $\mathtt{y}$: if $\widehat{m} := \mathbf{T}[\gamma(B, F_K(\mathtt{x} \| \mathtt{y}))] \neq \perp$, return $\widehat{m}$
3. If no such $\mathtt{y}$ found, return $\perp$

Security-Efficiency Trade-Off for Σ_3

Security-Efficiency Trade-Off for $\mathbf{\Sigma}_3$

Security of $\mathbf{\Sigma}_3$:

Security-Efficiency Trade-Off for $\boldsymbol{\Sigma}_3$

Security of $\boldsymbol{\Sigma}_3$: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Security-Efficiency Trade-Off for $\mathbf{\Sigma}_3$

Security of $\mathbf{\Sigma}_3$: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of $\mathbf{\Sigma}_3$:

Security-Efficiency Trade-Off for Σ_3

Security of Σ_3: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of Σ_3:

- **aEnc** takes σ tries *in expectation*

Security-Efficiency Trade-Off for $\mathbf{\Sigma}_3$

Security of $\mathbf{\Sigma}_3$: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of $\mathbf{\Sigma}_3$:

- **aEnc** takes σ tries *in expectation*
- **aDec** takes *at most* τ tries

Security-Efficiency Trade-Off for Σ_3

Security of Σ_3: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of Σ_3:

- **aEnc** takes σ tries *in expectation*
- **aDec** takes *at most* τ tries

Trade-off:

Security-Efficiency Trade-Off for Σ_3

Security of Σ_3: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of Σ_3:

- **aEnc** takes σ tries *in expectation*
- **aDec** takes *at most* τ tries

Trade-off:

- For **aEnc** and **aDec** to be *efficient*, σ and τ must be small (poly.)

Security-Efficiency Trade-Off for Σ_3

Security of Σ_3: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of Σ_3:

- **aEnc** takes σ tries *in expectation*
- **aDec** takes *at most* τ tries

Trade-off:

- For **aEnc** and **aDec** to be *efficient*, σ and τ must be small (poly.)
- This means, the limit on transmitted covert messages $\sigma \cdot \tau$ will also be small

Security-Efficiency Trade-Off for Σ_3

Security of Σ_3: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of Σ_3:

- **aEnc** takes σ tries *in expectation*
- **aDec** takes *at most* τ tries

Trade-off:

- For **aEnc** and **aDec** to be *efficient*, σ and τ must be small (poly.)
- This means, the limit on transmitted covert messages $\sigma \cdot \tau$ will also be small

Mitigation:

Security-Efficiency Trade-Off for $\mathbf{\Sigma}_3$

Security of $\mathbf{\Sigma}_3$: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of $\mathbf{\Sigma}_3$:

- **aEnc** takes σ tries *in expectation*
- **aDec** takes *at most* τ tries

Trade-off:

- For **aEnc** and **aDec** to be *efficient*, σ and τ must be small (poly.)
- This means, the limit on transmitted covert messages $\sigma \cdot \tau$ will also be small

Mitigation: in our new model, we can simply *update the double key!*

Conclusions

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**
- We also show how to make (fully) rand. recoverable schemes robustly anamorphic

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**
- We also show how to make (fully) rand. recoverable schemes robustly anamorphic
  - Use small subset of randomness as covert message space (concrete for **RSA-OAEP**)

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**
- We also show how to make (fully) rand. recoverable schemes robustly anamorphic
  - Use small subset of randomness as covert message space (concrete for **RSA-OAEP**)
- **Open questions:**

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**
- We also show how to make (fully) rand. recoverable schemes robustly anamorphic
  - Use small subset of randomness as covert message space (concrete for **RSA-OAEP**)
- **Open questions:**
  - Is the trade-off between security and efficiency for Σ_3 optimal?

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**
- We also show how to make (fully) rand. recoverable schemes robustly anamorphic
  - Use small subset of randomness as covert message space (concrete for **RSA-OAEP**)
- **Open questions:**
  - Is the trade-off between security and efficiency for Σ_3 optimal?
  - Are there more robust anamorphic schemes?

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**
- We also show how to make (fully) rand. recoverable schemes robustly anamorphic
  - Use small subset of randomness as covert message space (concrete for **RSA-OAEP**)
- **Open questions:**
  - Is the trade-off between security and efficiency for Σ_3 optimal?
  - Are there more robust anamorphic schemes? (see next talk 😉)

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**
- We also show how to make (fully) rand. recoverable schemes robustly anamorphic
  - Use small subset of randomness as covert message space (concrete for **RSA-OAEP**)
- **Open questions:**
  - Is the trade-off between security and efficiency for Σ_3 optimal?
  - Are there more robust anamorphic schemes? (see next talk 😉)

Thank You For Your Attention!

Anamorphic Encryption, Revisited

Fabio Banfi[1] Konstantin Gegier[2] Martin Hirt[2]

Ueli Maurer[2] Guilherme Rito[3]

[1]Zühlke Engineering AG, Switzerland

[2]ETH Zurich, Switzerland

[3]Ruhr-Universität Bochum, Germany

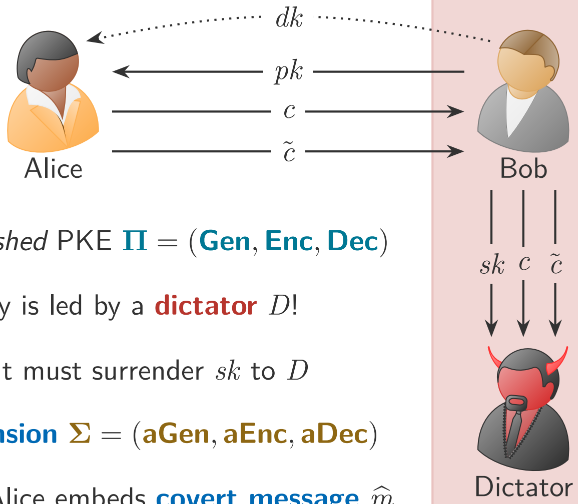
EUROCRYPT 2024

May 27, Zurich, Switzerland

(Receiver-)Anamorphic Encryption [Persiano et al., EUROCRYPT 2022]

$c \leftarrow \textbf{Enc}(pk, m)$

$\tilde{c} \leftarrow \textbf{aEnc}(dk, m', \widehat{m})$



$(sk, pk) \leftarrow \textbf{Gen}()$

$m := \textbf{Dec}(sk, c)$

$dk \leftarrow \textbf{aGen}(sk, pk)$

$\widehat{m} := \textbf{aDec}(dk, \tilde{c})$

$m' := \textbf{Dec}(sk, \tilde{c})$

$m := \textbf{Dec}(sk, c)$

$m' := \textbf{Dec}(sk, \tilde{c})$

Bob uses a *well-established* PKE $\boldsymbol{\Pi} = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec})$

Suddenly, Bob's country is led by a **dictator** D!

Bob can still use $\boldsymbol{\Pi}$, but must surrender sk to D

Use **anamorphic extension** $\boldsymbol{\Sigma} = (\textbf{aGen}, \textbf{aEnc}, \textbf{aDec})$

With **double key** dk, Alice embeds **covert message** $\widehat{m}$

Decoupling Keys & Security

In Persiano et al., double key dk was bound to key pair (sk, pk): $(sk, pk, dk) \leftarrow$ **aGen**$()$

Limitation: impossible to associate a new double key to an *already deployed* key pair

We redefine **aGen** so that Bob can *later* associate $dk \leftarrow$ **aGen**(sk, pk) to his key pair

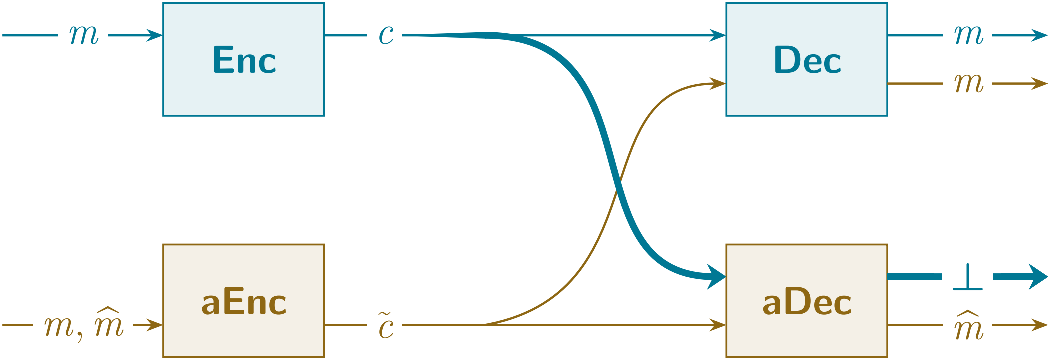
Advantages: can associate *multiple* double keys to a key pair and enables **deniability**

Recall the two modes Alice and Bob can use to communicate:

- **Normal:** $c \leftarrow \mathbf{Enc}(pk, m)$; $m := \mathbf{Dec}(sk, c)$
- **Anamorphic:** $\tilde{c} \leftarrow \mathbf{aEnc}(dk, m, \widehat{m})$; $\widehat{m} := \mathbf{aDec}(dk, \tilde{c})$, $m := \mathbf{Dec}(sk, \tilde{c})$

Security: The two modes must be indistinguishable: $\tilde{c} \approx c$! **Is this all?**

Using Anamorphic Encryption



This case was not considered! Need to signal *"no covert message"* $\Longrightarrow$ **Robustness**

Why Robustness?

- **Functionality:** Bob might use $\mathbf{\Pi}$ *regularly* and $\mathbf{\Sigma}$ *sporadically*

  Therefore, more often than not: *ciphertexts carry <u>no</u> (intentional) covert message!*

  When Bob sees "garbage" covert messages, he could guess they were not meant ...

  Is this satisfactory? **No!**

- **Security:** it could get even worse!

  Without robustness, D might find out that Bob has established a covert channel!

  1. Send encryption of random message to Bob
  2. If D is lucky, the covert message is not "garbage" and Bob detectably reacts!

Construction Σ_1: A Naive Robust Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Alice:** map $\widehat{m} \in \widehat{\mathcal{M}}$ to $r \in \mathcal{R}$ via F_K and counter `ctr`, use r to encrypt m into $\tilde{c}$:

  $$\textbf{aEnc}(dk, m, \widehat{m}) := \textbf{Enc}(pk, m; F_K(\texttt{ctr} \| \widehat{m}))$$

- **Bob:** decrypt $\tilde{c}$ into m, and check which $\widehat{m} \in \widehat{\mathcal{M}}$ yields $\tilde{c}$:

  $$\textbf{aDec}(dk, \tilde{c}) := \{ \text{let } m := \textbf{Dec}(sk, \tilde{c});$$
  $$\text{find } \widehat{m} \text{ s.t. } \textbf{Enc}(pk, m; F_K(\texttt{ctr} \| \widehat{m})) = \tilde{c} \text{ or return } \perp; \}$$

Problem: Alice and Bob need to keep **synchronized** counters and **aDec** uses **Dec**!

Solution: use PKEs with a special property: **Selective Randomness Recoverability**

Selective Randomness Recoverability (SRR)

PKE scheme $\mathbf{\Pi} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ is SRR if the following conditions are met:

(i) Randomness space $\mathcal{R}$ must form a group with some operation $\star$

(ii) Ciphertexts *"have two parts"*: for $c := \mathbf{Enc}(pk, m; r)$ we want $c = (A, B)$ where:

- Part A depends on pk, m, and r: $A = \alpha(pk, m, r)$
- Part B depends **only** on r: $B = \beta(r)$

(iii) Can compute $\beta(a)$ from $\beta(a \star b)$ and b:

- There exists an efficiently computable function γ s.t. $\gamma(\beta(a \star b), b) = \beta(a)$

Both **ElGamal** and **Cramer-Shoup** are SRR

Construction Σ_2: Using an SRR Scheme

Keep $\widehat{\mathcal{M}}$ small (poly. size), share key K of PRF F as part of double key dk, and then:

- **Bob:** precompute β^{-1} in *table* $\mathbf{T}$:   set $\mathbf{T}[\beta(\widehat{m})] := \widehat{m}$ for each $\widehat{m} \in \widehat{\mathcal{M}}$
- **Alice:** use $F_K(\mathtt{ctr})$ as otp for $\widehat{m}$ and use result as r to enc. m into $\tilde{c} = (A, B)$:

$$\textbf{aEnc}(dk, m, \widehat{m}; \mathtt{ctr}) := \textbf{Enc}(pk, m; \widehat{m} \star F_K(\mathtt{ctr}))$$

- **Bob:** use F_K and γ to extract $\widehat{m}$ from B:

$$\textbf{aDec}(dk, (A, B); \mathtt{ctr}) := \mathbf{T}[\gamma(B, F_K(\mathtt{ctr}))] \qquad [\textbf{Dec} \text{ not needed!}]$$

Still need to keep **synchronized** counters!

Construction Σ_3: Getting Rid of Synchronization

Idea: pick random ctr, until can *partially* extract ctr from B via some function δ

aEnc$(dk, m, \widehat{m})$:

1. Pick u.a.r. $(\mathtt{x}, \mathtt{y}) \in [\sigma] \times [\tau]$, set $\mathtt{ctr} := \mathtt{x} \| \mathtt{y}$, $r := \widehat{m} \star F_K(\mathtt{ctr})$, and $B := \beta(r)$
2. Repeat until $\delta(B) = \mathtt{x}$, let r^* be the such first r
3. Return $(A, B) :=$ **Enc**$(pk, m; r^*)$

aDec$(dk, (A, B))$:

1. Set $\mathtt{x} := \delta(B)$
2. For each possible value $\mathtt{y}$: if $\widehat{m} := \mathbf{T}[\gamma(B, F_K(\mathtt{x}\|\mathtt{y}))] \neq \bot$, return $\widehat{m}$
3. If no such $\mathtt{y}$ found, return $\bot$

Security-Efficiency Trade-Off for $\mathbf{\Sigma}_3$

Security of $\mathbf{\Sigma}_3$: can safely transmit *at most* $\sigma \cdot \tau$ covert messages

Efficiency of $\mathbf{\Sigma}_3$:

- **aEnc** takes σ tries *in expectation*
- **aDec** takes *at most* τ tries

Trade-off:

- For **aEnc** and **aDec** to be *efficient*, σ and τ must be small (poly.)
- This means, the limit on transmitted covert messages $\sigma \cdot \tau$ will also be small

Mitigation: in our new model, we can simply *update the double key!*

Conclusions

- Our abstract scheme can be made concrete for **ElGamal** and **Cramer-Shoup**
- We also show how to make (fully) rand. recoverable schemes robustly anamorphic
  - Use small subset of randomness as covert message space (concrete for **RSA-OAEP**)
- **Open questions:**
  - Is the trade-off between security and efficiency for Σ_3 optimal?
  - Are there more robust anamorphic schemes? (see next talk 😉)

Thank You For Your Attention!

Appendix: The Evolution of Anamorphic Encryption

- Persiano et al. [EUROCRYPT 2022]: first receiver- and sender-anam. schemes
- Kutyłowski et al. [CRYPTO 2023]: sender-anamorphic signatures
- Kutyłowski et al. [PoPETs 2023(4)]: more receiver-anamorphic PKE schemes
- Wang et al. [ASIACRYPT 2023]: sender-anam. **robustness** (inspired by our work)
- Our work [EUROCRYPT 2024]: receiver-anamorphic **robustness**
- Catalano et al. [EUROCRYPT 2024]: receiver-anam. homomorphic encryption

  + new receiver-anamorphic **robust** schemes
- More to come ...

Appendix: Deniability

Why does decoupling key-pair (sk, pk) and double key dk enable **deniability**?

Assume $dk \leftarrow$ **aGen**(pk) instead of $dk \leftarrow$ **aGen**(sk, pk) (true for all our constructions)

Then, a malicious sender holding dk *cannot* convince D that Bob also holds dk:

- The double key dk can be generated either by the sender or the receiver
- The sender can simulate dk and some ciphertexts, without the help of the receiver

This is **not true** for Persiano et al.'s anamorphic Naor-Yung transform:

- The malicious sender hands dk to the dictator
- The dictator can then detect whether key-pair was deployed in *anamorphic mode*

Appendix: An SRR Scheme

ElGamal on cyclic group $\mathbb{G} = \langle g \rangle$ of order q is SRR:

(i) $\mathcal{R} = \mathbb{Z}_q$, and $\langle \mathbb{Z}_q; \oplus \rangle$ is a group with $\oplus$ addition modulo q

(ii) With $A = \alpha(pk, m, r) = m \cdot pk^r$ and $B = \beta(r) = g^r$: **Enc**$(pk, m; r) = (A, B)$

(iii) With $\gamma(a, b) := a \cdot g^{-b}$: $\gamma(\beta(a \oplus b), b) = \gamma(g^{a \oplus b}, b) = g^{a \oplus b} \cdot g^{-b} = g^a = \beta(a)$

Analogously for **Cramer-Shoup**

Appendix: Correctness and Robustness of Σ_2

Correctness: with $(A, \beta(\widehat{m} \star F_K(\texttt{ctr}))) := \textbf{aEnc}(dk, m, \widehat{m}; \texttt{ctr})$:

$$\textbf{aDec}(dk, (A, \beta(\widehat{m} \star F_K(\texttt{ctr})); \texttt{ctr}) = \mathbf{T}[\gamma(\beta(\widehat{m} \star F_K(\texttt{ctr})), F_K(\texttt{ctr}))]$$

$$= \mathbf{T}[\beta(\widehat{m})] = \widehat{m}$$

Robustness: with $(A, \beta(r)) := \textbf{Enc}(pk, m; r)$, for $r \xleftarrow{\$} \mathcal{R}$:

$$\textbf{aDec}(dk, (A, \beta(r)); \texttt{ctr}) = \mathbf{T}[\gamma(\beta(r), F_K(\texttt{ctr}))] = \mathbf{T}[\beta(r \star F_K(\texttt{ctr})^{-1})] \overset{(*)}{\approx} \perp$$

$(*)$: w.o.p., since $r \star F_K(\texttt{ctr})^{-1} \notin \widehat{\mathcal{M}}$ with probability $1 - |\widehat{\mathcal{M}}|/|\mathcal{R}|$