An Introduction to Using F* in Cryptographic Proofs

Chris Brzuska, Antoine Delignat-Lavaud, Cedric Fournet,
Konrad Kohbrok, Markulf Kohlweiss

July 19, 2018

Aalto University
Microsoft Research Cambridge
Edinburgh University

Example: Cryptobox

Alice		**Bob**
g^b, a, m		g^a, b
$n \leftarrow_\$ \{0,1\}^{\text{noncelen}}$		
$\boxed{\begin{array}{l} k \leftarrow Hash(g^{ab}) \\ c \leftarrow \mathsf{Enc}_k(m, n) \end{array}}$		
	$\xrightarrow{\text{n, c}}$	
		$\boxed{\begin{array}{l} k \leftarrow Hash(g^{ab}) \\ \mathsf{Dec}_k(c, n) \end{array}}$

Example: Cryptobox cont'd

Cryptobox as a public key authenticated encryption scheme provides three algorithms: $\mathrm{Enc}, \mathrm{Dec}, \mathrm{Gen}$

- Gen(), returns keypair consisting of public and private key
- $\mathrm{Enc}(pk_r, sk_s, m, n)$, returns ciphertext
- $\mathrm{Dec}(sk_r, pk_s, c, n)$, returns message upon successful decryption, else $\perp$

Security Notion: $\mathtt{\$PKAE}^b$

$\underline{\text{GEN}()}$

$(pk, sk) \leftarrow_\$ \text{Gen}()$

$T[pk] \leftarrow sk$

return pk

Security Notion: $\mathtt{\$PKAE}^b$

$$
\begin{array}{ll}
\underline{\text{GEN}()} & \underline{\text{ENC}(pk_s, pk_r, m, n)} \\
(pk, sk) \leftarrow_\$ \text{Gen}() & \textbf{assert } T[pk_s] \neq \bot \\
T[pk] \leftarrow sk & sk_s \leftarrow T[pk_s] \\
\textbf{return } pk & \textbf{if } b = 1 \wedge T[pk] \neq \bot \textbf{ then} \\
 & \quad \boxed{\begin{array}{l} c \leftarrow_\$ \{0,1\}^{|m|} \\ M[\{pk_s, pk_r\}, c, n] \leftarrow m \end{array}} \\
 & \textbf{else} \\
 & \quad \boxed{c \leftarrow \text{Enc}(pk_r, sk_s, m, n)} \\
 & \textbf{return } c
\end{array}
$$

Security Notion: $\$\mathtt{PKAE}^b$

GEN()

$(pk, sk) \leftarrow_{\$} \mathrm{Gen}()$

$T[pk] \leftarrow sk$

return pk

ENC(pk_s, pk_r, m, n)

assert $T[pk_s] \neq \bot$

$sk_s \leftarrow T[pk_s]$

if $b = 1 \wedge T[pk] \neq \bot$ **then**

$c \leftarrow_{\$} \{0,1\}^{|m|}$

$M[\{pk_s, pk_r\}, c, n] \leftarrow m$

else

$c \leftarrow \mathrm{Enc}(pk_r, sk_s, m, n)$

return c

DEC(pk_r, pk_s, c, n)

assert $T[pk_r] \neq \bot$

$sk_r \leftarrow T[pk_r]$

if $b = 1 \wedge T[pk_s] \neq \bot$ **then**

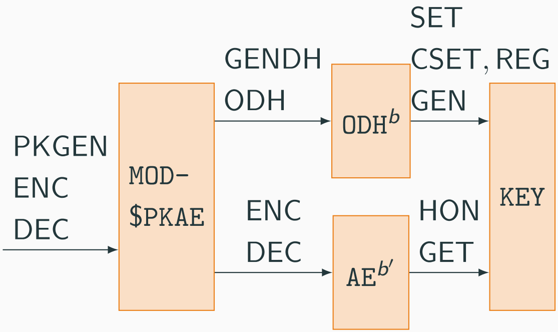
$m \leftarrow M[\{pk_r, pk_s\}, c, n]$

else

$m \leftarrow \mathrm{Dec}(pk_s, sk_r, c, n)$

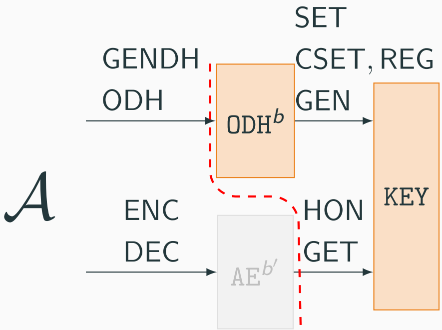
return m

Cryptobox: Formal Construction



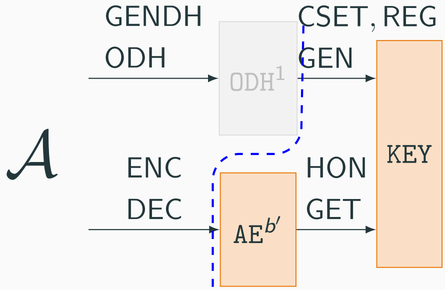
- Standard assumptions: Authenticated Encryption (AE) and Oracle Diffie-Hellman (ODH)
- Both assumptions use the KEY package

Cryptobox: Formal Construction



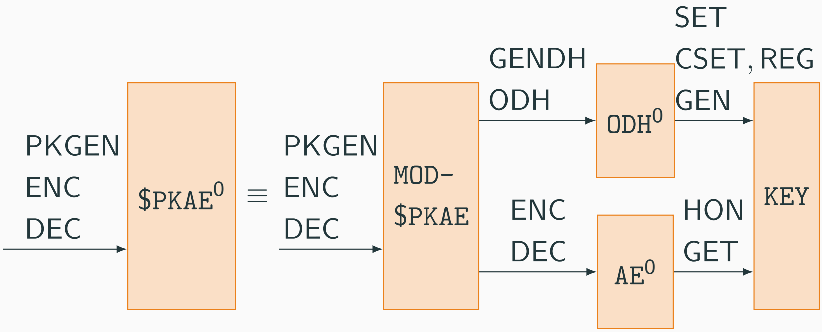
$$\left(\mathtt{ODH}^0 \middle| \mathtt{ID}_{\mathsf{HON,GET}}\right) \circ \mathtt{KEY} \overset{\epsilon_{\mathtt{ODH}}}{\approx} \left(\mathtt{ODH}^1 \middle| \mathtt{ID}_{\mathsf{HON,GET}}\right) \circ \mathtt{KEY}$$

Cryptobox: Formal Construction



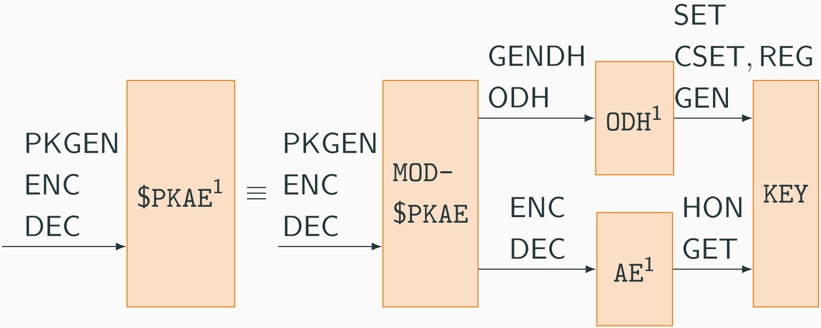
$$\left(\mathtt{ID}_{\mathsf{CSET},\mathsf{GEN},\mathsf{REG}} \middle| \mathtt{AE}^0\right) \circ \mathtt{KEY} \overset{\epsilon_{\mathtt{AE}}}{\approx} \left(\mathtt{ID}_{\mathsf{CSET},\mathsf{GEN},\mathsf{REG}} \middle| \mathtt{AE}^1\right) \circ \mathtt{KEY}$$

Cryptobox: Formal Construction



$$\$\mathtt{PKAE}^0 \equiv \mathtt{MOD\text{-}\$PKAE} \circ (\mathtt{ODH}^0 | \mathtt{AE}^0) \circ \mathtt{KEY}$$

Cryptobox: Formal Construction



$$\mathtt{\$PKAE}^0 \equiv \mathtt{MOD\text{-}\$PKAE} \circ (\mathtt{ODH}^0 | \mathtt{AE}^0) \circ \mathtt{KEY}$$

$$\wr\!\wr\, \epsilon_{\mathtt{ODH}} + \epsilon_{\mathtt{AE}}$$

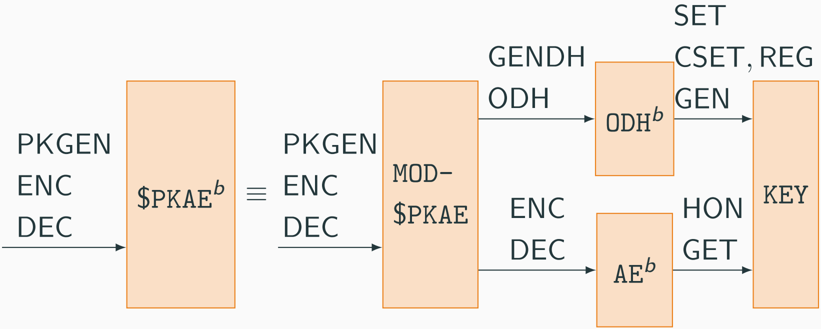
$$\mathtt{\$PKAE}^1 \equiv \mathtt{MOD\text{-}\$PKAE} \circ (\mathtt{ODH}^1 | \mathtt{AE}^1) \circ \mathtt{KEY}$$

What does F* do for us?

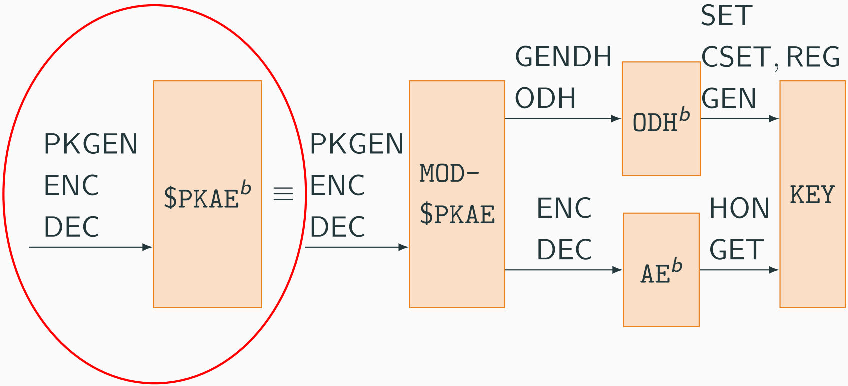
What *is* $F^\star$

- Functional programming language
- Prototype developed by Microsoft Research and INRIA Paris
- Strong type system

What does F⋆ do for us?

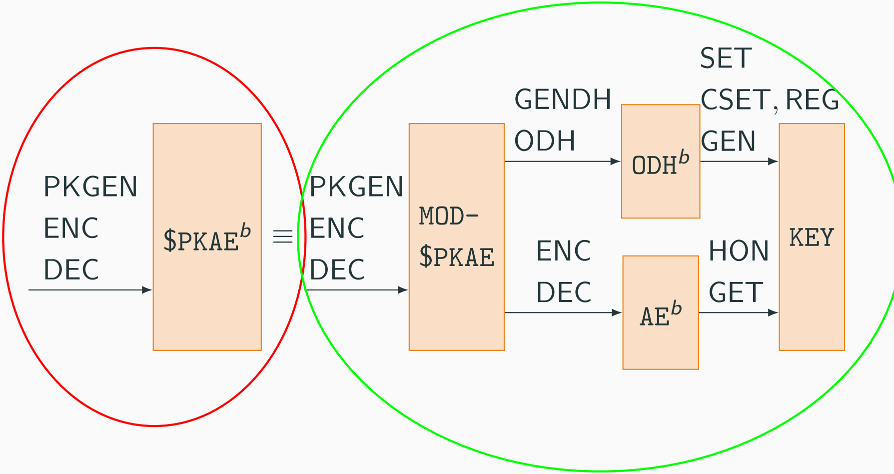


What does F* do for us?



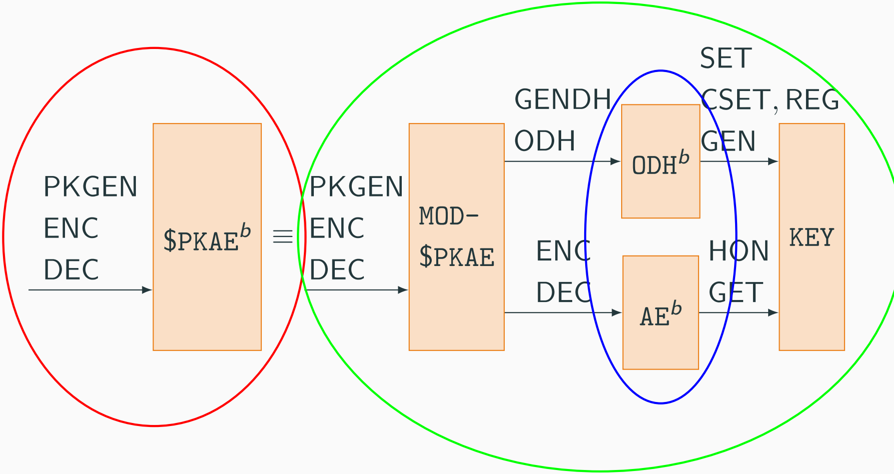
- Perfect indistinguishability

What does F⋆ do for us?



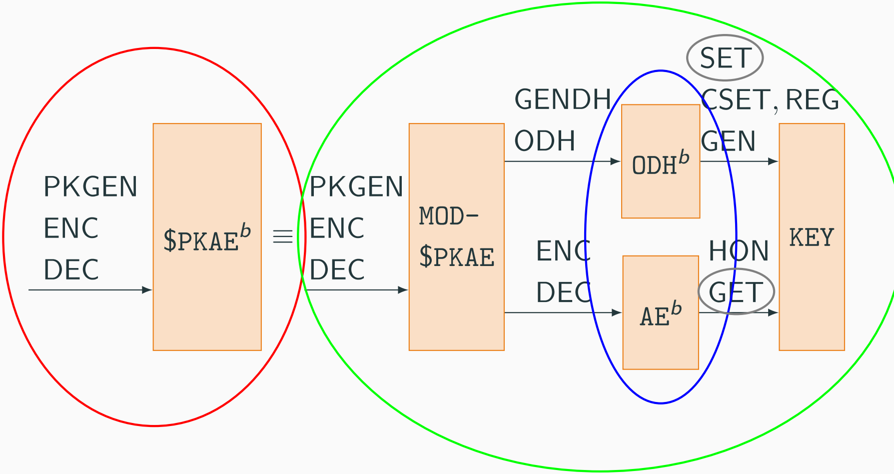
- Perfect indistinguishability
- Code packaging

What does F⋆ do for us?



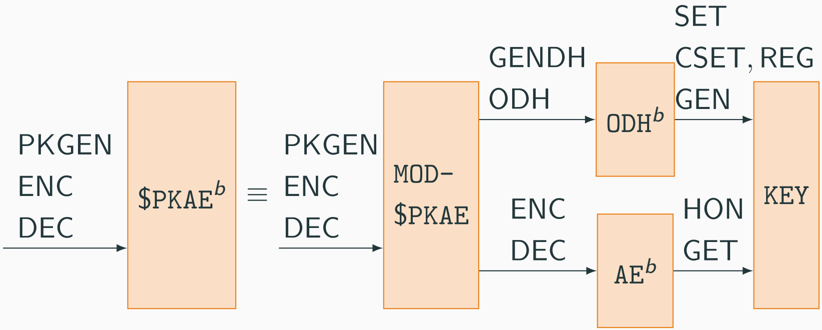
- Perfect indistinguishability
- Code packaging
- proof order

What does F* do for us?



- Perfect indistinguishability
- Code packaging
- proof order
- key security

What does F* do for us?



$$\mathtt{\$PKAE}^0 \equiv \mathtt{MOD\text{-}\$PKAE} \circ (\mathtt{ODH}^0 | \mathtt{AE}^0) \circ \mathtt{KEY} \equiv \text{“Concrete Code”}$$

$$\approx \epsilon_{\mathtt{ODH}} + \epsilon_{\mathtt{AE}}$$

$$\mathtt{\$PKAE}^1 \equiv \mathtt{MOD\text{-}\$PKAE} \circ (\mathtt{ODH}^1 | \mathtt{AE}^1) \circ \mathtt{KEY}$$

Useful F* Properties: Abstract Types

```
abstract type skey = bytes
```

- implementation of abstract types is hidden from external modules
- external modules can not instantiate abstract types

We will use abstract types to enforce state separation for certain types of state.

Useful F* Properties: Dependent Types

```
type my_int = x:int{x>5}
```

Useful F* Properties: Dependent Types

```
type my_int = x:int{x>5}

val my_add: x:my_int → y:my_int → z:my_int{z = x + y}
let my_add x y = x + y
```

- dependent types can fully specify a functions behaviour
- $\mathrm{F}^{\star}$ makes sure that pre- and postconditions are met

Useful F⋆ Properties: F⋆-Interfaces

Interface

```
module MyAdd

type my_int = x:int{x>5}

val my_add: x:my_int → y:my_int
              → z:my_int{z = x + y}
```

Implementation

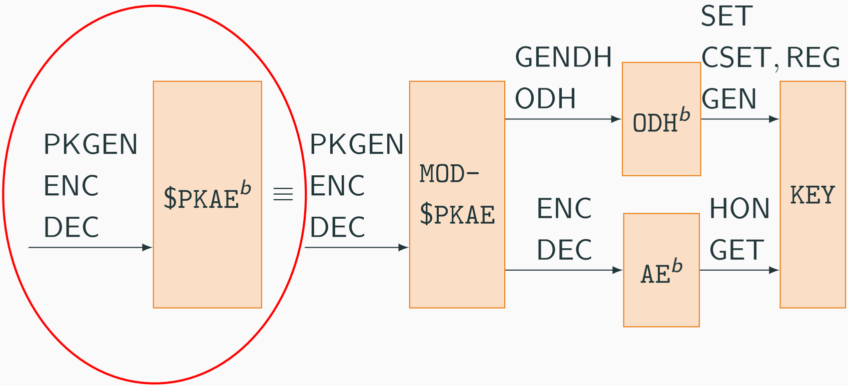
```
module MyAdd

let my_add x y = x + y
```

- interfaces can be used to describe the functionality of a package completely

We will use $\mathrm{F}^{\star}$-interfaces to prove perfect indistinguishability between packages.

Perfect Indistinguishability in F*



F*-Interfaces and Perfect Indistinguishability

$\underline{\mathrm{ENC}(pk_s, pk_r, m, n)}$

assert $T[pk_s] \neq \perp$

$sk_s \leftarrow T[pk_s]$

if $b = 1 \wedge T[pk_r] \neq \perp$ **then**

$\quad c \leftarrow\$ \{0,1\}^{|m|}$

$\quad M[\{pk_s, pk_r\}, c, n] \leftarrow m$

else

$\quad c \leftarrow \mathrm{Enc}(pk_r, sk_s, m, n)$

return c

F⋆-Interfaces and Perfect Indistinguishability

$\underline{\mathrm{ENC}(pk_s, pk_r, m, n)}$

assert $T[pk_s] \neq \perp$

$sk_s \leftarrow T[pk_s]$

if $b = 1 \wedge T[pk_r] \neq \perp$ **then**

$\quad \boxed{\begin{array}{l} c \leftarrow_\$ \{0,1\}^{|m|} \\ M[\{pk_s, pk_r\}, c, n] \leftarrow m \end{array}}$

else

$\quad \boxed{c \leftarrow \mathrm{Enc}(pk_r, sk_s, m, n)}$

return c

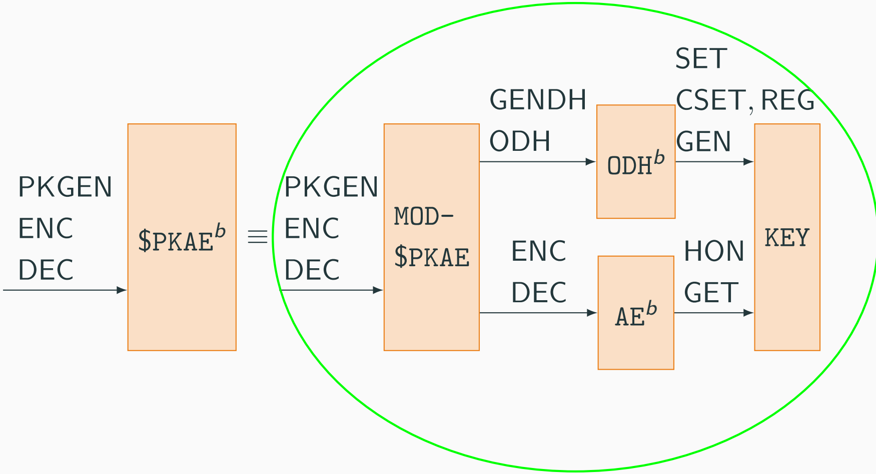
$PKAE Interface

```
module PKAE

abstract type skey = bytes
...

val enc: sk:skey →
         pk:pkey →
         p:plaintext →
         n:nonce →
         c:ciphertext{
  if hon pk ∧ hon sk ∧ b then
    c = random_bytes (length p)
    extend_log ({pk,sk},n,c) p
  else
    c = Cryptobox.enc pk sk p n
}
```

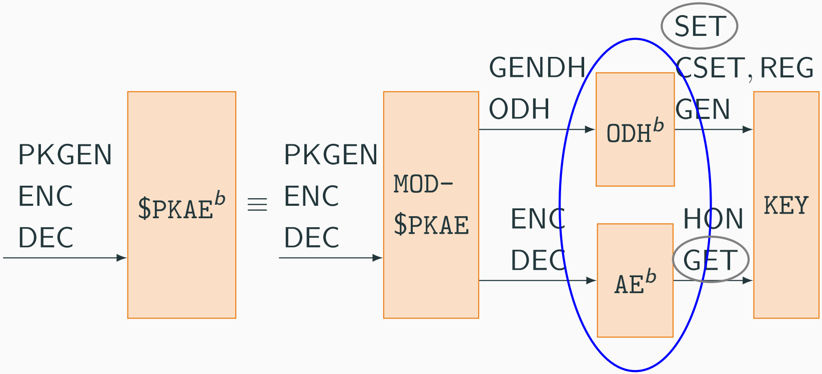
Composition in F*

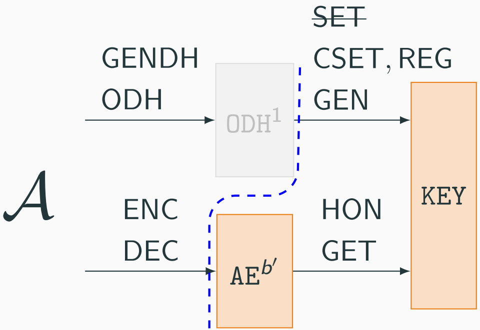


Code Packaging in F⋆

- One F⋆-module per package
  - F⋆-modules contain functions, variables
- F⋆-module dependencies form a DAG
- Same basic properties as packages in the framework

Proof Ordering and Key Security in F*




~~SET~~
GENDH
CSET, REG
ODH
GEN
$\mathtt{ODH}^1$
$\mathcal{A}$
KEY
ENC
HON
DEC
GET
$\mathtt{AE}^{b'}$

$\mathtt{AE}^b$ in F*

Major Design Decision

```
abstract type key = bytes
```

$\mathtt{AE}^b$ in F*

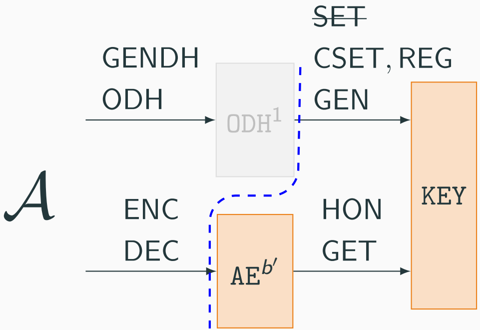
Major Design Decision

```
abstract type key = bytes
```

Why are we doing this?

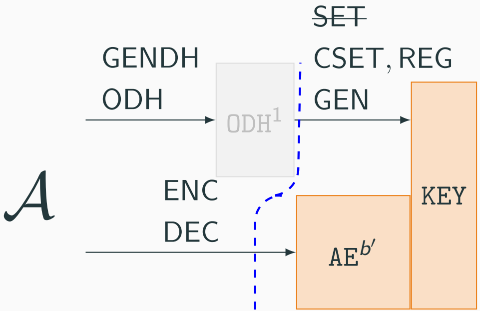
- We can handle keys instead of indices.
  - $\mathrm{F}^\star$ makes sure the concrete key is invisible to any package but $\mathtt{AE}^b$.
- We can get rid of a GET and a SET oracle.
  - Security of keys can be confirmed just by studying one file.

Consequences of an Abstract Key Type



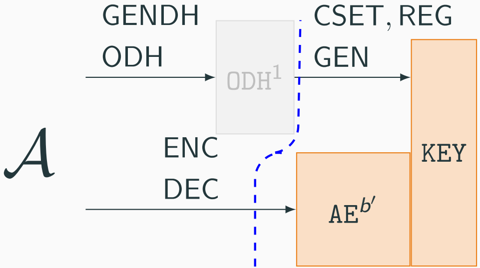
Consequences of an Abstract Key Type

- No separate KEY package



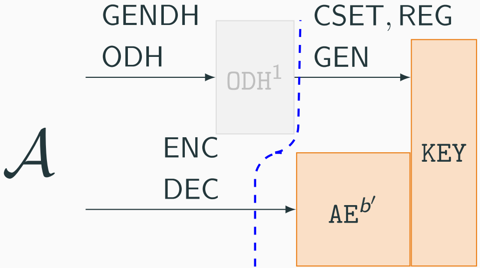
Consequences of an Abstract Key Type

- No separate KEY package
- No SET oracles



Consequences of an Abstract Key Type

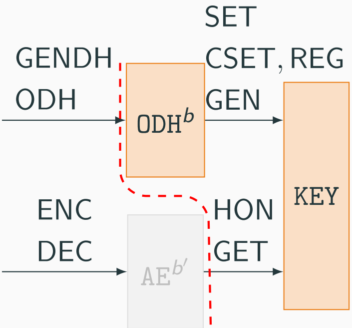
- No separate KEY package
- No SET oracles



- No cross-use of keys
- Some trickery needed to bind a key to an index
- More trickery needed to have a proper ODH assumption

(Old) $\mathtt{ODH}$ Assumption

$\underline{\text{GENDH}}$	$\underline{\text{ODH}(\widetilde{X}, \widetilde{Y})}$
$a \leftarrow\$ \mathbb{Z}_p$	**assert** $T[\widetilde{X}] \neq \bot$
$T[g^a] \leftarrow a$	$x \leftarrow T[\widetilde{X}]$
return g^a	**if** $T[\widetilde{Y}] \neq \bot$ **then**
	if b **then**
	$\text{GEN}(\{\widetilde{X}, \widetilde{Y}\})$
	else
	$\text{SET}(\{\widetilde{X}, \widetilde{Y}\}, \text{H}(\widetilde{Y}^x))$
	else
	$\text{CSET}(\{\widetilde{X}, \widetilde{Y}\}, \text{H}(\widetilde{Y}^x))$



A New $\texttt{KEY}$ Package

Standard $\texttt{KEY}$ package:

$\underline{\mathrm{REG}(i, b')}$

$H[i] \leftarrow b'$

$\underline{\mathrm{GET}(i)}$

return $K[i]$

$\underline{\mathrm{SET}(i, k)}$

assert $H[i] = 1$

$K[i] \leftarrow k$

$\underline{\mathrm{CSET}(i, k)}$

assert $H[i] = 0$

$K[i] \leftarrow k$

$\underline{\mathrm{GEN}(i)}$

$K[i] \leftarrow\$ \{0, 1\}^\lambda$

$\underline{\mathrm{HON}(i)}$

return $H[i]$

New $\texttt{KEY}$ package, now part of $\texttt{AE}^b$:

$\underline{\mathrm{REG}(i, b')}$

$H[i] \leftarrow b'$

$\underline{\mathrm{COERCE}(i, k)}$

assert $H[i] = 0$

$K[i] \leftarrow k$

$\underline{\mathrm{GEN}(i)}$

$K[i] \leftarrow\$ \{0, 1\}^\lambda$

$\underline{\mathrm{HON}(i)}$

return $H[i]$

A New KEY Package

Standard KEY package:

$\underline{\mathrm{REG}(i, b')}$

$H[i] \leftarrow b'$

$\underline{\mathrm{GET}(i)}$

return $K[i]$

$\underline{\mathrm{SET}(i, k)}$

assert $H[i] = 1$

$K[i] \leftarrow k$

$\underline{\mathrm{CSET}(i, k)}$

assert $H[i] = 0$

$K[i] \leftarrow k$

$\underline{\mathrm{GEN}(i)}$

$K[i] \leftarrow_{\$} \{0, 1\}^{\lambda}$

$\underline{\mathrm{HON}(i)}$

return $H[i]$

New KEY package, now part of $\mathtt{AE}^b$:

$\underline{\mathrm{REG}(i, b')}$

$H[i] \leftarrow b'$

$\underline{\mathrm{COERCE}(i, k)}$

assert $H[i] = 0 \vee \neg b$

$K[i] \leftarrow k$

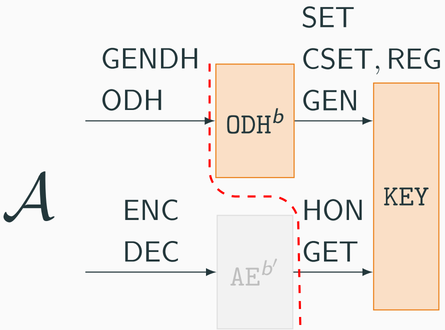
$\underline{\mathrm{GEN}(i)}$

$K[i] \leftarrow_{\$} \{0, 1\}^{\lambda}$

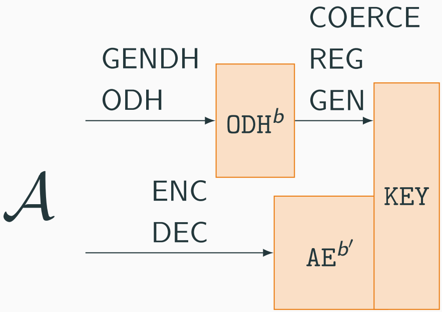
$\underline{\mathrm{HON}(i)}$

return $H[i]$

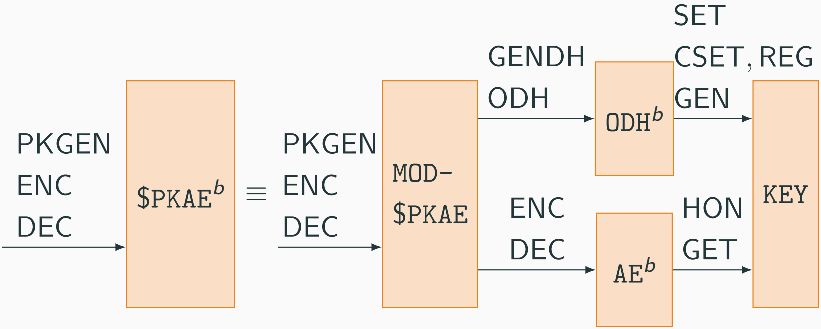
New $\mathtt{ODH}^b$ Assumption



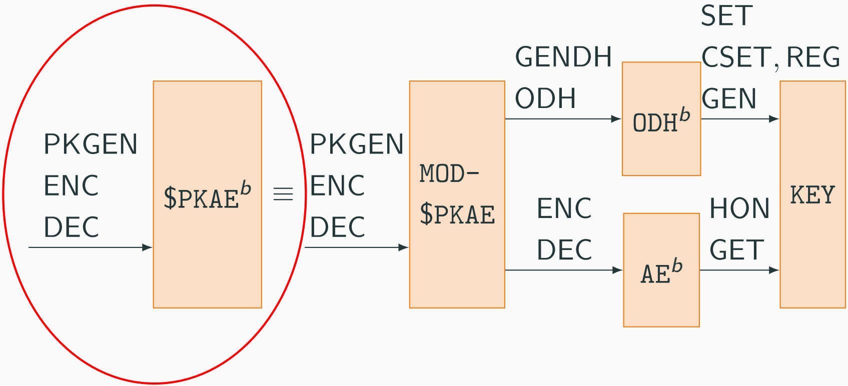
New $\mathtt{ODH}^b$ Assumption



What does F* do for us?

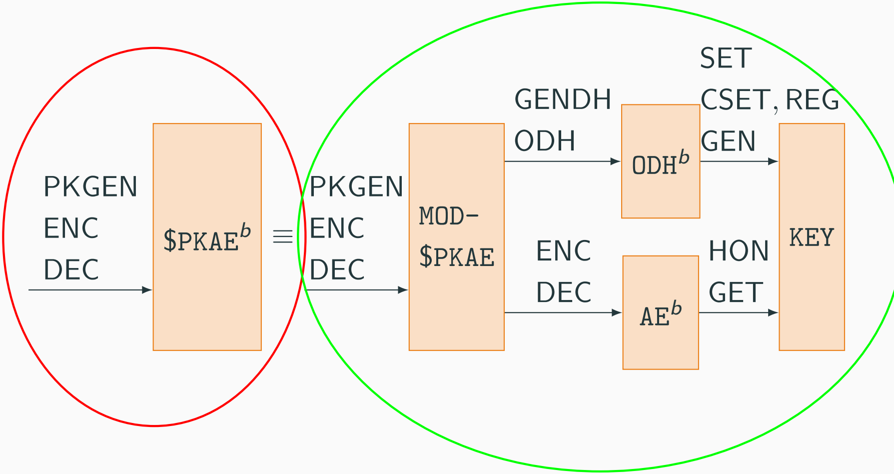


What does F* do for us?



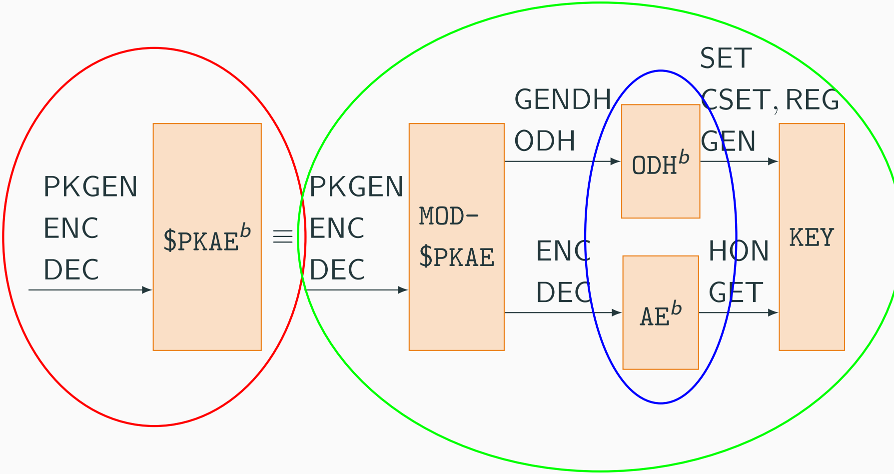
- Perfect indistinguishability

What does F* do for us?



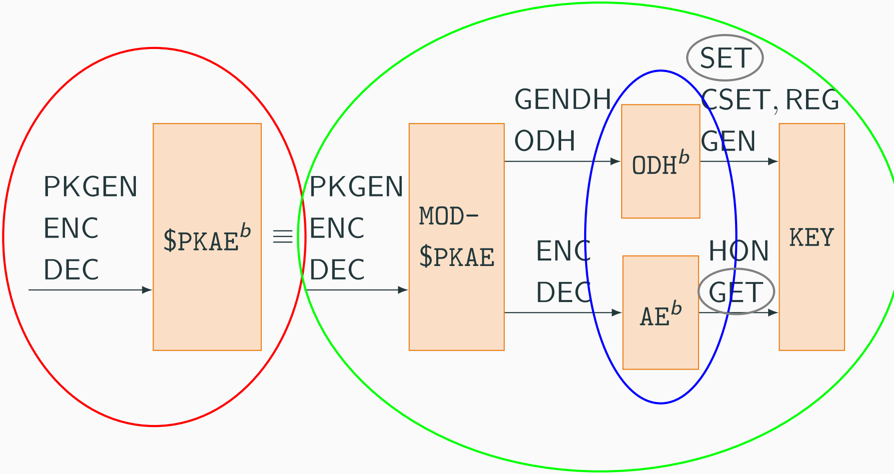
- Perfect indistinguishability
- Code packaging

What does F⋆ do for us?



- Perfect indistinguishability
- Code packaging
- proof order

What does F⋆ do for us?



- Perfect indistinguishability
- Code packaging
- proof order
- key security