Sequence-of-games method for cryptographic proofs

Peeter Laud
peeter.laud@ut.ee
http://www.ut.ee/~peeter_l

Tartu University & Cybernetica AS

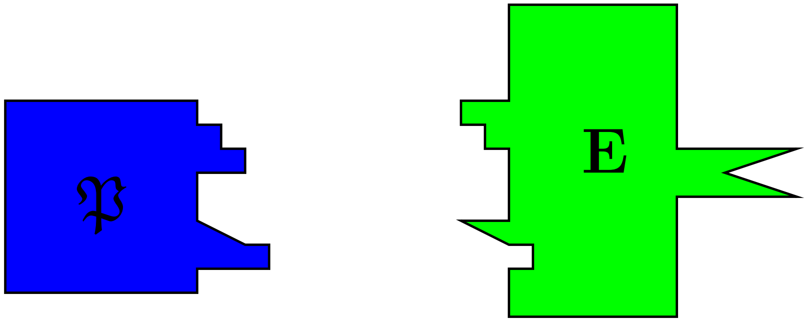
A cryptographic primitive

A primitive is made up of

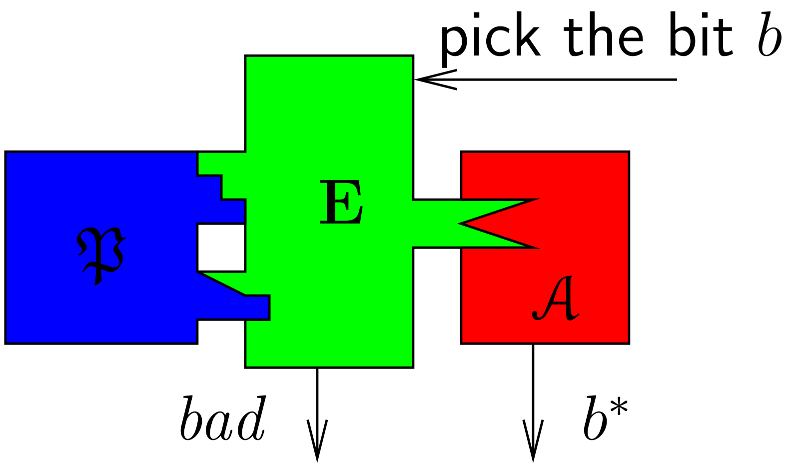
- its interface
  - like an abstract data type
  - method signatures and equalities (e.g. $\mathcal{D}_k(\mathcal{E}_k(x)) = x$)
- its security definition, made up of
  - the interface and implementation of an experiment
  - the success criterion for the adversary
    - either "guess a bit" or "set a bit"

(more complex security definitions are possible, too)

Picture



Picture



$P \in \mathfrak{P}$ is $(\mathfrak{A}, \varepsilon)$-secure if for all $\mathcal{A} \in \mathfrak{A}$:

$$\Pr[bad = 1] \leq \varepsilon \wedge \Pr[b = b^*] \leq \frac{1}{2} + \varepsilon$$

The actual difference of these probabilities from 0 resp. $1/2$ is the advantage of $\mathcal{A}$.

IND-CPA-secure asymm. encryption

- $\mathfrak{P}_{\mathrm{enc}}$ has
  - the methods keygen (nullary, returns a pair), enc, dec (binary);
    - all arguments and values are bit-strings;
    - the plaintext is ℓ_{e} bits long.
  - the equality: $(pk, sk) :=$ keygen(); dec$(sk,$ enc$(pk, x))$ is equivalent to $(pk, sk) :=$ keygen(); x
- $\mathbf{E}_{\mathrm{FtG}}$ has the methods
  - init() is $(pk, \_) := \mathfrak{P}$.keygen(); $b \stackrel{R}{\leftarrow} \{0, 1\}$; $\mathbf{return}\ pk$.
  - lor(M_0, M_1) is $\mathbf{return}\ \mathfrak{P}$.enc$(pk, M_b)$.
  - Both methods can be called only once, in correct order.
  - M_0 and M_1 must be ℓ_{e} bits long.

One-way trapdoor permutation

- $\mathfrak{P}_{\mathrm{tdp}}$ has [same as asymm. enc.]
  - the methods keygen (nullary, returns a pair), enc, dec (binary);
    - all arguments and values are bit-strings;
    - the plaintexts and ciphertexts are ℓ_{p} bits long. $(\ell_{\mathrm{p}} > \ell_{\mathrm{e}})$
  - the equality: $(pk, sk) :=$ keygen(); dec$(sk,$enc$(pk, x))$ is equivalent to $(pk, sk) :=$ keygen(); x
- $\mathbf{E}_{\mathrm{owf}}$ is
  - init() is $(pk, \_) := \mathfrak{P}$.keygen(); $x \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}}$; **return**$(pk, \mathfrak{P}$.enc$(pk, x))$.
  - guess(y) is **if** $x = y$ **then** $bad :=$ true.
  - Both methods can be called only once, in correct order.

Example: “plain” RSA

Guessing only

- Assume that $\mathbf{E}$ does not output the bad-bit.
- This assumption is wlog.:
- Instead, let $\mathbf{E}$ have a method bad$?()$ that, when queried at the end of the execution, returns $b \wedge bad$.
    - $\mathcal{A}$ may not access $\mathbf{E}$ any more after making the bad$?$-query.

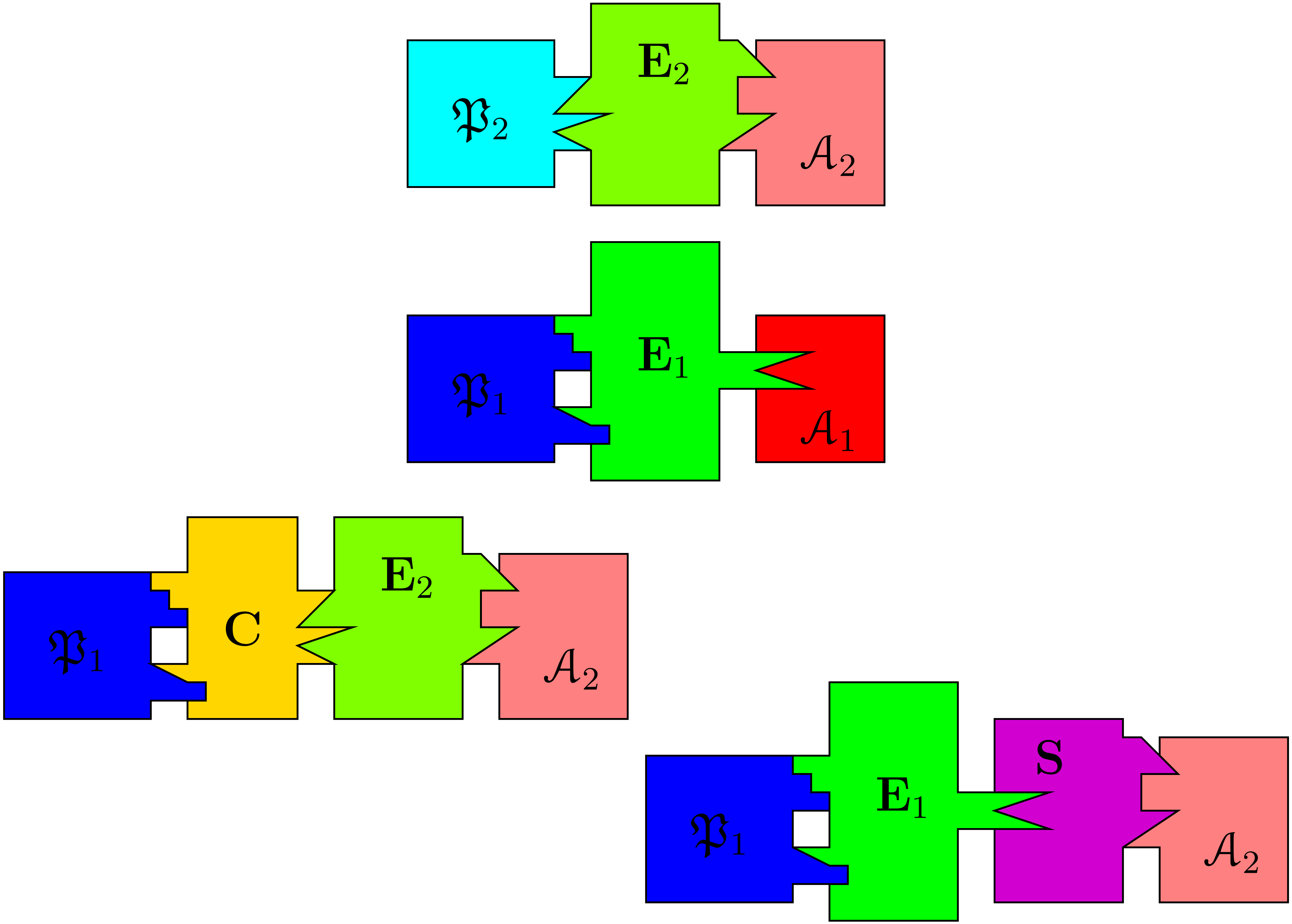
One-way trapdoor permutation

- $\mathfrak{P}_{\mathrm{tdp}}$ has [same as asymm. enc.]
  - the methods keygen (nullary, returns a pair), enc, dec (binary);
    - all arguments and values are bit-strings;
    - the plaintexts and ciphertexts are ℓ_{p} bits long. $(\ell_{\mathrm{p}} > \ell_{\mathrm{e}})$
  - the equality: $(pk, sk) :=$ keygen(); dec$(sk,$ enc$(pk, x))$ is equivalent to $(pk, sk) :=$ keygen(); x
- $\mathbf{E}_{\mathrm{owf}}$ is
  - init() is $(pk, \_) := \mathfrak{P}$.keygen(); $x \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}}$; $b \stackrel{R}{\leftarrow} \{0,1\}$; $\mathbf{return}(pk, \mathfrak{P}.\mathsf{enc}(pk, x))$.
  - guess?(y) is $\mathbf{return}$ $(\mathbf{if}\ x = y\ \mathbf{then}\ b\ \mathit{else}\ 0)$.
  - Both methods can be called only once, in correct order.

Reductions

- Let $\mathfrak{P}_1$ and $\mathfrak{P}_2$ be two primitives, with security definitions $\mathbf{E}_1$ and $\mathbf{E}_2$.
- Let $\mathbf{C}$ be an algorithm, such that for all $P_1 \in \mathfrak{P}_1$ we have $P_1 \| \mathbf{C} \in \mathfrak{P}_2$.
- A cryptographic reduction is a claim of the form "if P_1 is a $(\mathfrak{A}_1, \varepsilon_1)$-secure instance of $\mathfrak{P}_1$ then $P_1 \| \mathbf{C}$ is a $(\mathfrak{A}_2, \varepsilon_2)$-secure instance of $\mathfrak{P}_2$".
- A black-box proof of that claim consists of
  - an algorithm $\mathbf{S}$ (the simulator);
  - proof that if $\mathcal{A}_2 \in \mathfrak{A}_2$ then $\mathbf{S} \| \mathcal{A}_2 \in \mathfrak{A}_1$;
  - proof that if some $\mathcal{A}_2$ has the advantage $\geq \varepsilon_2$ against some $P_1 \| \mathbf{C}$ then $\mathbf{S} \| \mathcal{A}_2$ has the advantage $\geq \varepsilon_1$ against P_1.

Picture



$\mathfrak{P}_2$
$\mathbf{E}_2$
$\mathcal{A}_2$
$\mathfrak{P}_1$
$\mathbf{E}_1$
$\mathcal{A}_1$
$\mathfrak{P}_1$
$\mathbf{C}$
$\mathbf{E}_2$
$\mathcal{A}_2$
$\mathfrak{P}_1$
$\mathbf{E}_1$
$\mathbf{S}$
$\mathcal{A}_2$

Example: OAEP

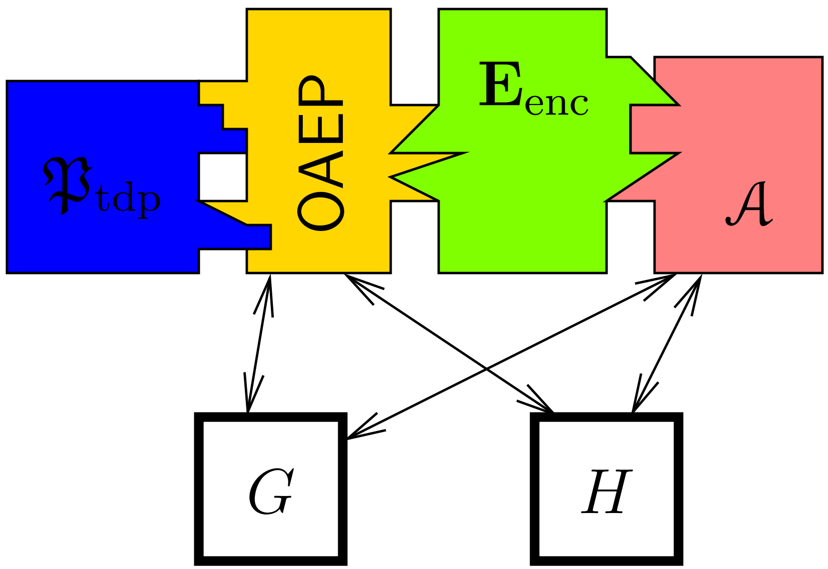
- Implements $\mathfrak{P}_{\mathrm{enc}}$ using $\mathfrak{P}_{\mathrm{tdp}}$.
- Uses two functions (random oracles) $\mathbf{G} : \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}} \rightarrow \{0,1\}^{\ell_{\mathrm{e}}}$ and $\mathbf{H} : \{0,1\}^{\ell_{\mathrm{e}}} \rightarrow \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$.
- $\mathbf{C}$.keygen simply calls $\mathfrak{P}_{\mathrm{tdp}}$.keygen.
- $\mathbf{C}$.enc(pk, x) is $r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$; $s := \mathbf{G}(r) \oplus x$; $t := \mathbf{H}(s) \oplus r$; $\mathbf{return}$ $\mathfrak{P}_{\mathrm{tdp}}$.enc$(pk, s \| t)$.

Under encryption:

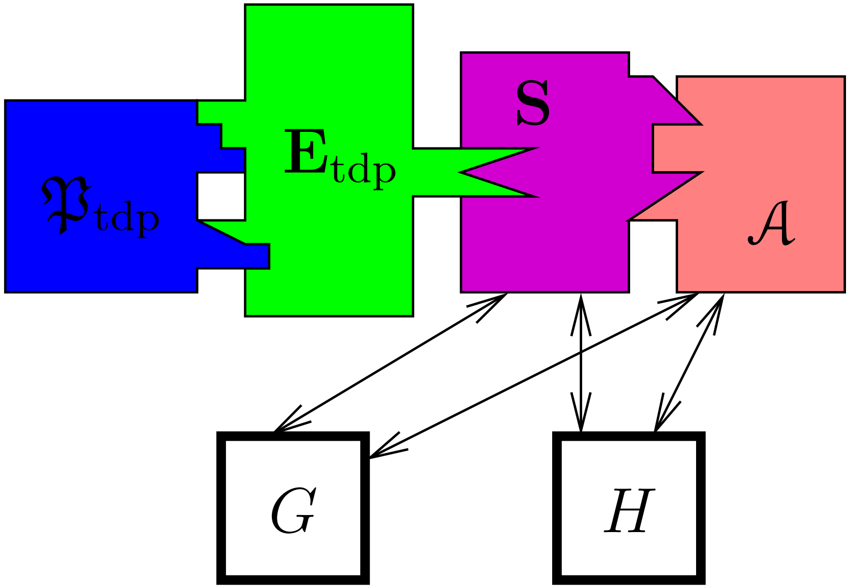
$$\underbrace{G(r) \oplus x}_{s} \;\Big\|\; H(s) \oplus r$$

- $\mathbf{C}$.dec is its inverse...
- A simulator has been proposed...

Picture



Picture



$\mathfrak{P}_{\mathrm{tdp}}$
$\mathbf{E}_{\mathrm{tdp}}$
$\mathbf{S}$
$\mathcal{A}$
G
H

Problems with the approach

- $\mathbf{S}$ may be quite complex.
  - Other components may be complex, too:
  - We are comparing $\mathbf{C}\|\mathbf{E}_2$ with $\mathbf{E}_1\|\mathbf{S}$.
- It may be quite hard to prove that $\mathbf{S}\|\mathcal{A}_2$ has large advantage.
- Example: OAEP was proposed in Eurocrypt '94. The flaw in the proof (of IND-CCA-security) was found in 2000.

Problems with the approach

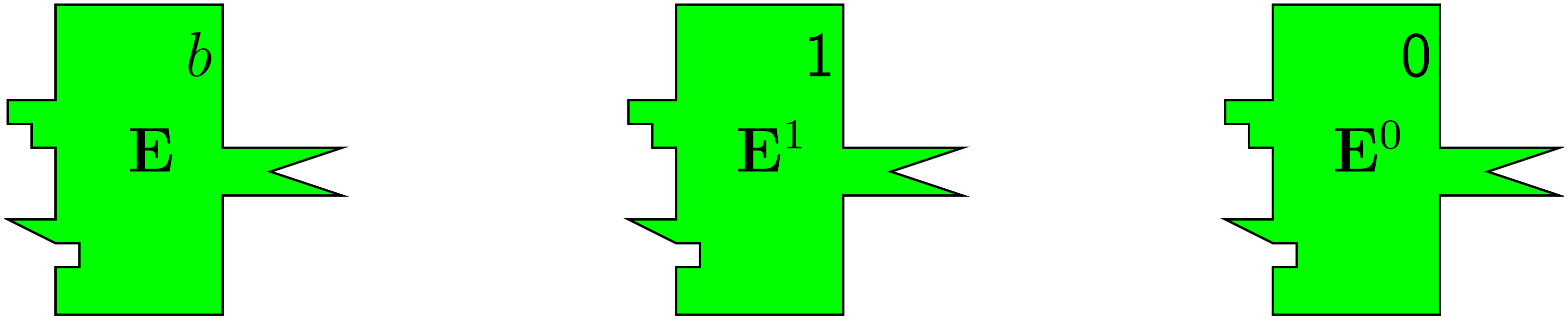
- $\mathbf{S}$ may be quite complex.
  - Other components may be complex, too:
  - We are comparing $\mathbf{C}\|\mathbf{E}_2$ with $\mathbf{E}_1\|\mathbf{S}$.
- It may be quite hard to prove that $\mathbf{S}\|\mathcal{A}_2$ has large advantage.
- The proof is even more complex if $\mathbf{C}$ is parameterized somehow. E.g.
  - $(\mathfrak{P}_1, \mathbf{E}_1)$ is secure encryption;
  - $\mathfrak{P}_2$ are programs in some programming language;
  - $\mathbf{E}_2$ requires a program to have (computationally) secure information flow;
  - $\mathbf{C} \in \mathfrak{C}$ are the programs that are accepted by some static checking mechanism.
- Example: my PhD-thesis (year 2002):
  - $\mathfrak{C}$ — 9 pages. $\mathbf{S}$ and the correctness proof — 65 pages.
  - Do you think that the proof is correct?

Everything is program code

- $\mathbf{E}_1$, $\mathbf{E}_2$ can be written down in some programming language.
  - We can be very precise here.
  - We fix the semantics of the programming language.
- The same holds for $\mathbf{C}$.
- If we are given a family $\mathfrak{C}$ then we still can consider its elements as programs.
- The programs $\mathbf{E}_1$ and $\mathbf{E}_2$ generate the random bit b somewhere inside their code.

Specializing $\mathbf{E}$

- Given $\mathbf{E}$, define $\mathbf{E}^1$ and $\mathbf{E}^0$ as follows:
  - $\mathbf{E}^1$ is like $\mathbf{E}$, but instead of randomly generating b, it has $b := 1$.
  - $\mathbf{E}^0$ is like $\mathbf{E}$, but instead of randomly generating b, it has $b := 0$.
- Let $\mathbf{b}_i$ be the random bit output by $\mathcal{A}$ if it runs in parallel with $P \in \mathfrak{P}$ and $\mathbf{E}^i$.
- $P \in \mathfrak{P}$ is $(\mathfrak{A}, \varepsilon)$-secure if for all $\mathcal{A} \in \mathfrak{A}$, the distance of the distributions $\mathbf{b}_0$ and $\mathbf{b}_1$ is at most ε.



Program transformations

- Let P_1 be $(\mathfrak{A}_1, \varepsilon_1)$-secure instance of $\mathfrak{P}_1$ with security definition $\mathbf{E}_1$.
- Consider the construction $\mathbf{C}$ and the environment $\mathbf{E}_2$.
- Suppose that $\mathbf{C}\|\mathbf{E}_2 \equiv \mathbf{D}\|\mathbf{E}_1^b$.
- Suppose that for all $\mathcal{A} \in \mathfrak{A}_2$ we have $\mathbf{D}\|\mathcal{A} \in \mathfrak{A}_1$.
- Then no $\mathcal{A} \in \mathfrak{A}_2$ can distinguish

$$P_1\|\mathbf{D}\|\mathbf{E}_1^1 \quad \text{and} \quad P_1\|\mathbf{D}\|\mathbf{E}_1^0$$

with advantage of more than ε_1.

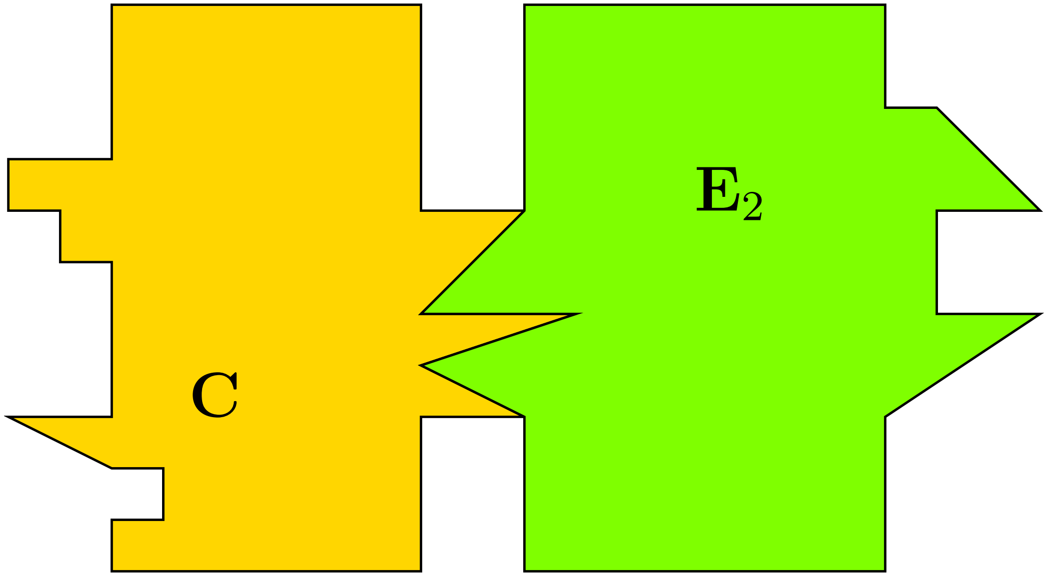
Program transformations

- Let P_1 be $(\mathfrak{A}_1, \varepsilon_1)$-secure instance of $\mathfrak{P}_1$ with security definition $\mathbf{E}_1$.
- Consider the construction $\mathbf{C}$ and the environment $\mathbf{E}_2$.
- Suppose that $\mathbf{C}\|\mathbf{E}_2 \equiv \mathbf{D}\|\mathbf{E}_1^b$.
- Suppose that for all $\mathcal{A} \in \mathfrak{A}_2$ we have $\mathbf{D}\|\mathcal{A} \in \mathfrak{A}_1$.
- Then no $\mathcal{A} \in \mathfrak{A}_2$ can distinguish

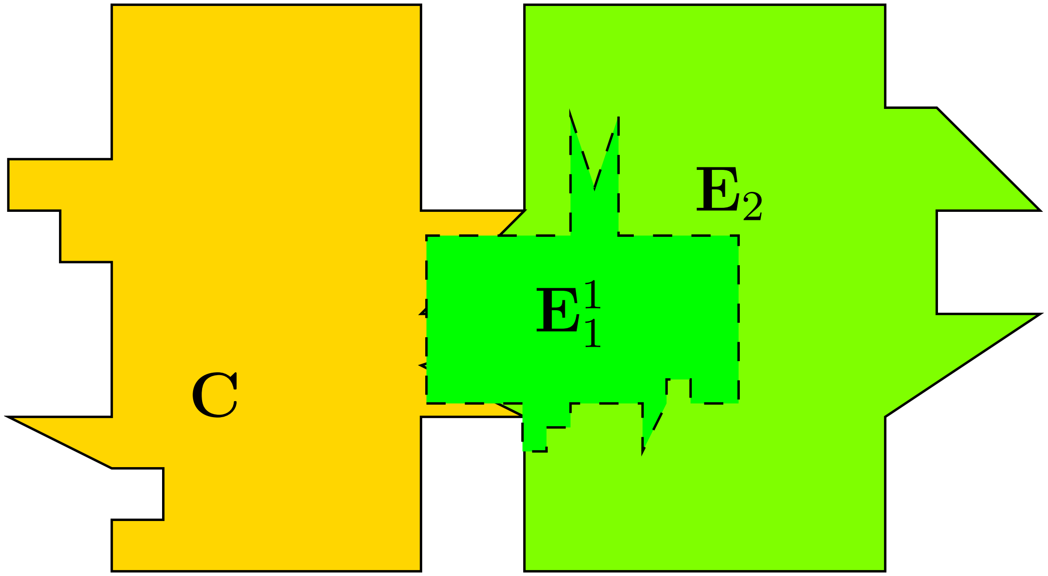
  $$P_1\|\mathbf{D}\|\mathbf{E}_1^1 \quad \text{and} \quad P_1\|\mathbf{D}\|\mathbf{E}_1^0$$

  with advantage of more than ε_1.
- Suppose now that $\mathbf{D}\|\mathbf{E}_1^{1-b} \equiv \mathbf{D}'\|\mathbf{E}_1^{b'}$ with $\mathbf{D}'\|\mathcal{A} \in \mathfrak{A}_1$ for all $\mathcal{A} \in \mathfrak{A}_2$.
- Then no $\mathcal{A} \in \mathfrak{A}_2$ can distinguish $P_1\|\mathbf{D}'\|\mathbf{E}_1^1$ and $P_1\|\mathbf{D}'\|\mathbf{E}_1^0$ with an advantage of more that ε_1.
- By triangle inequality, no $\mathcal{A} \in \mathfrak{A}_2$ can distinguish $P_1\|\mathbf{D}\|\mathbf{E}_1^b$ and $P_1\|\mathbf{D}'\|\mathbf{E}_1^{1-b'}$ with an advantage of more than $2\varepsilon_1$.

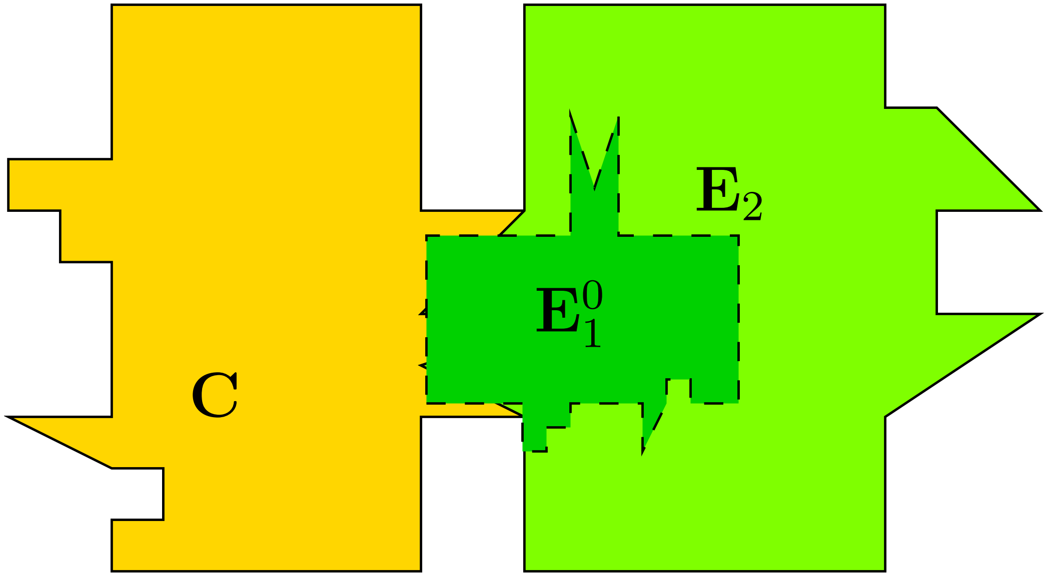
Picture



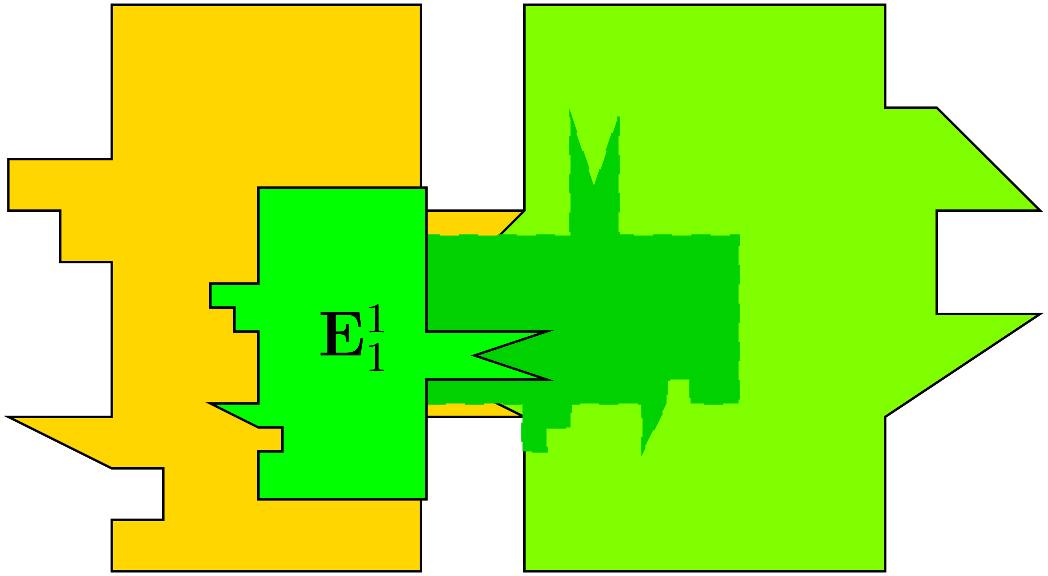
Picture



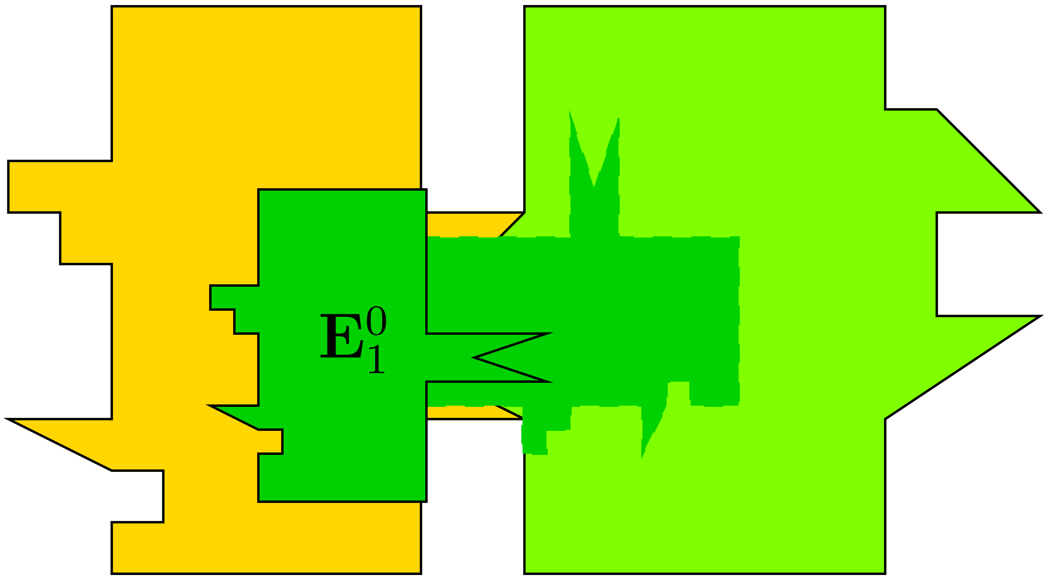
Picture



Picture



Picture



Analysis strategy

- Transform $\mathbf{C}\|\mathbf{E}_2$ until we reach a program that does not use b.
- Allowed transformations are given by the security definition(s) of the primitive(s) that $\mathbf{C}$ uses.
  - These transformations come with the upper bound of the advantage of distinguishing the original and final program.
- Also allowed are changes that do not change the observable semantics of the program.
- The probability that b can be guessed in the original program is no more than the sum of advantages associated with transformations.

OAEP

$\mathbf{G}(x)$:
if $G[x] \neq \perp$
    return $G[x]$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $H[y] \neq \perp$
    return $H[y]$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():
$(pk, \_)$:= keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s := \mathbf{G}(r) \oplus M_b$
$t := \mathbf{H}(s) \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Assume wlog. that $\mathcal{A}$ does not repeat queries to $\mathbf{G}$ and $\mathbf{H}$

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $G[x] \neq \bot$
    return $G[x]$
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $H[y] \neq \bot$
    return $H[y]$
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\mathsf{init}()\text{:}}$
$(pk, \_) := \mathsf{keygen}()$
$b \xleftarrow{R} \{0,1\}$
return pk

$\underline{\mathsf{lor}(M_0, M_1)\text{:}}$
$r \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s := \mathbf{G}(r) \oplus M_b$
$t := \mathbf{H}(s) \oplus r$
$z := s \| t$
$w := \mathsf{enc}(pk, z)$
return w

Inline $\mathbf{G}$ and $\mathbf{H}$...

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $G[x] \neq \bot$
  return $G[x]$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $H[y] \neq \bot$
  return $H[y]$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\text{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
return pk

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
(if $G[r] = \bot$ then $R := G[r] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$ else $R := G[r]$); $s := R \oplus M_b$
(if $H[s] = \bot$ then $S := H[s] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$ else $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:
if $G[x] \neq \bot$
  return $G[x]$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $H[y] \neq \bot$
  return $H[y]$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():
$(pk, \_)$:= keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
(if $G[r] = \bot$ then $R := G[r] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$ else $R := G[r]$); $s := R \oplus M_b$
(if $H[s] = \bot$ then $S := H[s] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$ else $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Pre-generate R and S...

OAEP

$\mathbf{G}(x)$:
if $G[x] \neq \bot$
  return $G[x]$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $H[y] \neq \bot$
  return $H[y]$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():
$(pk, \_)$:= keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$; $R \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$; $S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
(if $G[r] = \bot$ then $G[r] := R$ else $R := G[r]$); $s := R \oplus M_b$
(if $H[s] = \bot$ then $H[s] := S$ else $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:
if $G[x] \neq \bot$
  return $G[x]$
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $H[y] \neq \bot$
  return $H[y]$
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():
$(pk, \_)$:= keygen()
$b \xleftarrow{R} \{0,1\}$
return pk

lor(M_0, M_1):
$r \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$; $R \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$; $S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
(if $G[r] = \bot$ then $G[r] := R$ else $R := G[r]$); $s := R \oplus M_b$
(if $H[s] = \bot$ then $H[s] := S$ else $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Create R and S during initialization...

OAEP

$\mathbf{G}(x)$:
if $G[x] \neq \perp$
  return $G[x]$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $H[y] \neq \perp$
  return $H[y]$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():
$(pk, \_)$:= keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
$R \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$;
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$;
(if $G[r] = \perp$ then $G[r] := R$ else $R := G[r]$); $s := R \oplus M_b$
(if $H[s] = \perp$ then $H[s] := S$ else $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Distinguishing advantage: 0

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $G[x] \neq \bot$
    return $G[x]$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $H[y] \neq \bot$
    return $H[y]$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\text{init}()\text{:}}$
$(pk, \_) := \text{keygen}()$
$b \stackrel{R}{\leftarrow} \{0,1\}$
$R \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$;
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return pk

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$;
(if $G[r] = \bot$ then $G[r] := R$ else $R := G[r]$); $s := R \oplus M_b$
(if $H[s] = \bot$ then $H[s] := S$ else $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w := \text{enc}(pk, z)$
return w

Why could this be?

OAEP

G(x):

if $x = r$
    return R
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

H(y):

if $y = s$
    return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():

$(pk, \_)$:= keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
$R \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$;
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return pk

lor(M_0, M_1):

$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$;
(if $G[r] = \perp$ then $G[r] := R$ else $R := G[r]$); $s := R \oplus M_b$
(if $H[s] = \perp$ then $H[s] := S$ else $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:
if $x = r$
  return R
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $y = s$
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():
$(pk, \_)$:= keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
$R \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$;
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$;
(if $G[r] = \bot$ then $G[r] := R$ else $R := G[r]$); $s := R \oplus M_b$
(if $H[s] = \bot$ then $H[s] := S$ else $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

These assignments are dead

OAEP

$\mathbf{G}(x)$:
if $x = r$
    return R
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $y = s$
    return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():
$(pk, \_) :=$ keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
$R \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$;
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$;
(if $G[r] \neq \bot$ then $R := G[r]$); $s := R \oplus M_b$
(if $H[s] \neq \bot$ then $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Distinguishing advantage: 0

More analysis strategy

- The transformed program contains a bit bad, initially false.
  - Similar to, but formally not related to the bit bad we had earlier for expressing integrity properties.
- The program may contain statements setting bad to true.
- The program never reads bad.

$\Rightarrow$ Setting bad does not change the observable behaviour.

More analysis strategy

- The transformed program contains a bit bad, initially false.
  - Similar to, but formally not related to the bit bad we had earlier for expressing integrity properties.
- The program may contain statements setting bad to true.
- The program never reads bad.

$\Rightarrow$ Setting bad does not change the observable behaviour.

- A transformation may not remove the settings of bad.
- We may freely change the code that is executed only if bad is set.
- The distinguishing advantage is assumed to be 0.

More analysis strategy

- The transformed program contains a bit bad, initially false.
  - Similar to, but formally not related to the bit bad we had earlier for expressing integrity properties.
- The program may contain statements setting bad to true.
- The program never reads bad.

$\Rightarrow$ Setting bad does not change the observable behaviour.

- A transformation may not remove the settings of bad.
- We may freely change the code that is executed only if bad is set.
- The distinguishing advantage is assumed to be 0.
- Actually we may remove occurrencies of setting bad.
  - But we must pay as the distinguishing advantage $\Pr_{\circ}[bad = \mathsf{true}] - \Pr_{\bullet}[bad = \mathsf{true}]$.

OAEP

$\mathbf{G}(x)$:
if $x = r$
  return R
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{e}}}$
return $G[x]$

$\mathbf{H}(y)$:
if $y = s$
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
return $H[y]$

init():
$(pk, \_)$:= keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
$R \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{e}}}$;
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$;
(if $G[r] \neq \bot$ then $R := G[r]$); $s := R \oplus M_b$
(if $H[s] \neq \bot$ then $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Collisions are generally bad...

OAEP

G(x):
if $x = r$
  $bad :=$ true
  return R
$G[x] \xleftarrow{R} \{0,1\}^{\ell_e}$
return $G[x]$

H(y):
if $y = s$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

init():
$(pk, \_) :=$ keygen()
$b \xleftarrow{R} \{0,1\}$
$R \xleftarrow{R} \{0,1\}^{\ell_e}$;
$S \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return pk

lor(M_0, M_1):
$r \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$;
(if $G[r] \neq \perp$ then $bad :=$ true; $R := G[r]$); $s := R \oplus M_b$
(if $H[s] \neq \perp$ then $bad :=$ true; $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Distinguishing advantage: 0

OAEP

$\underline{\mathbf{G}(x)}$:
if $x = r$
  $bad :=$ true
  return R
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $y = s$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\text{init}()}$:
$(pk, \_) :=$ keygen()
$b \xleftarrow{R} \{0,1\}$
$R \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$;
$S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return pk

$\underline{\text{lor}(M_0, M_1)}$:
$r \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$;
(if $G[r] \neq \perp$ then $bad :=$ true; $R := G[r]$); $s := R \oplus M_b$
(if $H[s] \neq \perp$ then $bad :=$ true; $S := H[s]$); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

After bad, nothing matters

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $x = r$
    $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $y = s$
    return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\text{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$b \stackrel{R}{\leftarrow} \{0,1\}$
$R \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$;
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return pk

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$;
(if $G[r] \neq \bot$ then $bad :=$ true); $s := R \oplus M_b$
(if $H[s] \neq \bot$ then $bad :=$ true); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Distinguishing advantage: 0

OAEP

G(x):
if $x = r$
  $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_e}$
return $G[x]$

H(y):
if $y = s$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

init():
$(pk, \_) :=$ keygen()
$b \xleftarrow{R} \{0,1\}$
$R \xleftarrow{R} \{0,1\}^{\ell_e}$;
$S \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return pk

lor(M_0, M_1):
$r \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$;
(if $G[r] \neq \bot$ then $bad :=$ true); $s := R \oplus M_b$
(if $H[s] \neq \bot$ then $bad :=$ true); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

s is just a random value (and R is dead)

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $x = r$
  $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $y = s$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\text{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$b \xleftarrow{R} \{0,1\}$
$S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return pk

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$r \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$;
(if $G[r] \neq \perp$ then $bad :=$ true); $s \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
(if $H[s] \neq \perp$ then $bad :=$ true); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Distinguishing advantage: 0

The bit b has disappeared. Now we have to bound the probability of bad.

OAEP

G(x):
if $x = r$
  $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

H(y):
if $y = s$
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

init():
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$;
(if $G[r] \neq \perp$ then $bad :=$ true); $s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
(if $H[s] \neq \perp$ then $bad :=$ true); $t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

This does not happen often. . .

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $x = r$
  bad := true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $y = s$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\text{init}()\text{:}}$
$(pk, \_)$:= keygen()
$S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return pk

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$r \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$;
$s \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
$t := S \oplus r$
$z := s \| t$
w := enc(pk, z)
return w

Distinguishing advantage: $q_G/2^{\ell_\mathrm{p}-\ell_\mathrm{e}} + q_H/2^{\ell_\mathrm{e}}$

OAEP

$\mathbf{G}(x)$:
if $x = r$
  $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $y = s$
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

init():
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return pk

lor(M_0, M_1):
$r \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$;
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
$t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return w

Regroup. . .

OAEP

$\underline{\mathbf{G}(x)}$:
if $lor \wedge (x = r)$
  $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $lor \wedge (y = s)$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)}$:
$lor :=$ true
return w

$\underline{\mathsf{init}()}$:
$(pk, \_) :=$ keygen()
$S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$r \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
$t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $lor \wedge (x = r)$
  $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $lor \wedge (y = s)$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\mathsf{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$S \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
$r \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
$s \xleftarrow{R} \{0,1\}^{\ell_e}$
$t := S \oplus r$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Regroup. . .

OAEP

$\underline{\mathbf{G}(x)}$:
if $lor \wedge (x = r)$
    $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $lor \wedge (y = s)$
    return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)}$:
$lor :=$ true
return w

$\underline{\mathsf{init}()}$:
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
$r := S \oplus t$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\underline{\mathbf{G}(x)}$:
if $lor \wedge (x = r)$
  $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $lor \wedge (y = s)$
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)}$:
$lor :=$ true
return w

$\underline{\mathsf{init}()}$:
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
$r := S \oplus t$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Copy propagation. . .

OAEP

$\underline{\mathbf{G}(x)}$:
if $lor \wedge (x = S \oplus t)$
  $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $lor \wedge (y = s)$
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)}$:
$lor :=$ true
return w

$\underline{\mathsf{init}()}$:
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\underline{\mathbf{G}(x)}$:
if $lor \wedge (x = S \oplus t)$
  $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $lor \wedge (y = s)$
  return $S \leftarrow$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)}$:
$lor :=$ true
return w

$\underline{\mathsf{init}()}$:
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Mark that we have been there. Regroup

OAEP

$\underline{\mathbf{G}(x)}$:
if $lor \wedge (x \oplus t = S)$
  $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $lor \wedge (y = s)$
  $d :=$ true
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)}$:
$lor :=$ true
return w

$\underline{\mathsf{init}()}$:
$(pk, \_) := \mathsf{keygen}()$
$S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$t \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
$z := s \| t$
$w := \mathsf{enc}(pk, z)$
return pk

Distinguishing advantage: 0

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $lor \wedge (x \oplus t = S)$
  $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $lor \wedge (y = s)$
  $d :=$ true
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\text{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Consider both cases of d

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if d
  if $lor \wedge (x \oplus t = S)$
    $bad :=$ true
else
  if $lor \wedge (x \oplus t = S)$
    $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $lor \wedge (y = s)$
  $d :=$ true
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\text{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:
if d
  if $lor \wedge (x \oplus t = S)$
    $bad :=$ true
else
  if $lor \wedge (x \oplus t = S)$
    $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $lor \wedge (y = s)$
  $d :=$ true
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

lor(M_0, M_1):
$lor :=$ true
return w

init():
$(pk, \_) :=$ keygen()
$S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$t \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

No previous use of S.

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if d
    if $lor \wedge (x \oplus t = S)$
        $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{e}}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $lor \wedge (y = s)$
    $d :=$ true
    return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
return $H[y]$

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\text{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{e}}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: $1/2^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$

OAEP

$\underline{\mathbf{G}(x)}$:
if d
    if $lor \land (x \oplus t = S)$
        $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $lor \land (y = s)$
    $d :=$ true
    return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\text{lor}(M_0, M_1)}$:
$lor :=$ true
return w

$\underline{\text{init}()}$:
$(pk, \_) :=$ keygen()
$S \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
$t \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
$s \xleftarrow{R} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Clean up control flow. Regroup

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $d \wedge (x \oplus S = t)$
    $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_{\mathrm{e}}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $lor \wedge (y = s)$
    $d :=$ true
    return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\mathsf{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$S \xleftarrow{R} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
$t \xleftarrow{R} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
$s \xleftarrow{R} \{0,1\}^{\ell_{\mathrm{e}}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:
if $d \wedge (x \oplus S = t)$
  $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $lor \wedge (y = s)$
  $d :=$ true $\leftarrow$
  return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

lor(M_0, M_1):
$lor :=$ true
return w

init():
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Save y after guessing s

OAEP

$\mathbf{G}(x)$:
if $d \wedge (x \oplus S = t)$
  $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $lor \wedge (y = s)$
  $d :=$ true
  $y^* := y$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

lor(M_0, M_1):
$lor :=$ true
return w

init():
$(pk, \_) :=$ keygen()
$S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$t \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $d \wedge (x \oplus S = t)$
    $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
if $lor \wedge (y = s)$
    $d :=$ true
    $y^* := y$
    return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\text{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$S \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
$t \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
$s \xleftarrow{R} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Here $d \equiv lor \wedge (y^* = s)$

OAEP

G(x):
if $lor \wedge (y^* = s) \wedge$
$(x \oplus S = t)$
$bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

H(y):
if $lor \wedge (y = s)$
$d :=$ true
$y^* := y$
return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

lor(M_0, M_1):
$lor :=$ true
return w

init():
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:
if $lor \wedge (y^* = s) \wedge$
$(x \oplus S = t)$
$bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $lor \wedge (y = s)$
$d :=$ true
$y^* := y$
return S
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

lor(M_0, M_1):
$lor :=$ true
return w

init():
$(pk, \_) :=$ keygen()
$S \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Regroup. This is dead. First use of S

OAEP

$\underline{\mathbf{G}(x)}$:

if $lor \wedge$
$(y^* \| (x \oplus S) = s \| t)$
  bad := true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:

if $lor \wedge (y = s)$
  $y^* := y$
  $S \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

lor(M_0, M_1):

lor := true
return w

init():

$(pk, \_)$:= keygen()
$t \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
$s \xleftarrow{R} \{0,1\}^{\ell_e}$
$z := s \| t$
w := enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:

if $lor \wedge$
$(y^* \| (x \oplus S) = s \| t)$
  $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:

if $lor \wedge (y = s)$
  $y^* := y$
  $S \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
  return S
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

lor(M_0, M_1):

$lor :=$ true
return w

init():

$(pk, \_) :=$ keygen()
$t \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Store S as $H[y^*]$

OAEP

$\mathbf{G}(x)$:
if $lor \wedge (y^* \| (x \oplus H[y^*]) = s \| t)$
  $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\mathbf{H}(y)$:
if $lor \wedge (y = s)$
  $y^* := y$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

Distinguishing advantage: 0

lor(M_0, M_1):
$lor :=$ true
return w

init():
$(pk, \_) :=$ keygen()
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

OAEP

$\mathbf{G}(x)$:
if $lor \wedge (y^* \| (x \oplus H[y^*]) = s \| t)$
    $bad :=$ true
$G[x] \overset{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\mathbf{H}(y)$:
if $lor \wedge (y = s)$
    $y^* := y$
$H[y] \overset{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

lor(M_0, M_1):
$lor :=$ true
return w

init():
$(pk, \_) :=$ keygen()
$t \overset{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
$s \overset{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

Consider any y^* where $H[y^*]$ has been defined

OAEP

$\underline{\mathbf{G}(x)}$:
if $lor \wedge \exists y^* : (y^* \| (x \oplus H[y^*]) = s \| t)$
    $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)}$:
if $lor \wedge (y = s)$
    $y^* := y$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

Distinguishing advantage: 0

$\underline{\text{lor}(M_0, M_1)}$:
$lor :=$ true
return w

$\underline{\text{init}()}$:
$(pk, \_) :=$ keygen()
$t \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
$s \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

OAEP

$\mathbf{G}(x)$:
if $lor \wedge \exists y^* : (y^* \| (x \oplus H[y^*]) = s \| t)$
$\quad bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_e}$
return $G[x]$

$\mathbf{H}(y)$:
if $lor \wedge (y = s)$
$\quad y^* := y$
$H[y] \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

This is dead. Directly generate z

lor(M_0, M_1):
$lor :=$ true
return w

init():
$(pk, \_) :=$ keygen()
$t \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
$s \xleftarrow{R} \{0,1\}^{\ell_e}$
$z := s \| t$
$w :=$ enc(pk, z)
return pk

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
if $lor \land \exists y^* : (y^* \| (x \oplus H[y^*]) = z)$
    $bad :=$ true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\text{init}()\text{:}}$
$(pk, \_) :=$ keygen()
$z \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}}$
$w :=$ enc(pk, z)
return pk

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:

if $lor \wedge \exists y^* : (y^* \| (x \oplus H[y^*]) = z)$

  $bad :=$ true

$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{e}}}$

return $G[x]$

$\mathbf{H}(y)$:

$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$

return $H[y]$

lor(M_0, M_1):

$lor :=$ true

return w

init():

$(pk, \_) :=$ keygen()

$z \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}}$

$w :=$ enc(pk, z)

return pk

Check only a single y^* (randomly chosen)

OAEP

$\underline{\mathbf{G}(x)\text{:}}$

$y^* \xleftarrow{R} \{y \mid \mathsf{def}(H[y])\}$
if $lor \wedge (y^* \| (x \oplus H[y^*]) = z)$
  $bad := \mathsf{true}$
$G[x] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$

$H[y] \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

Distinguishing advantage: q^H times

$\underline{\mathsf{lor}(M_0, M_1)\text{:}}$

$lor := \mathsf{true}$
return w

$\underline{\mathsf{init}()\text{:}}$

$(pk, \_) := \mathsf{keygen}()$
$z \xleftarrow{R} \{0,1\}^{\ell_\mathrm{p}}$
$w := \mathsf{enc}(pk, z)$
return pk

OAEP

$\underline{\mathbf{G}(x)\text{:}}$

$y^* \xleftarrow{R} \{y \,|\, \mathsf{def}(H[y])\}$
if $lor \wedge (y^* \| (x \oplus H[y^*]) = z)$
  bad := true
$G[x] \xleftarrow{R} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$

$H[y] \xleftarrow{R} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

$\mathbf{E}_{\mathrm{owf}}$.guess. $\mathbf{E}_{\mathrm{owf}}$.init.

$\underline{\mathsf{lor}(M_0, M_1)\text{:}}$

lor := true
return w

$\underline{\mathsf{init}()\text{:}}$

$(pk, \_)$:= keygen()
$z \xleftarrow{R} \{0,1\}^{\ell_p}$
w := enc(pk, z)
return pk

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
$y^* \stackrel{R}{\leftarrow} \{y \mid \mathsf{def}(H[y])\}$
if $lor \wedge \mathbf{E}_{\mathrm{owf}}.\mathsf{guess}(y^* \| (x \oplus H[y^*]))$
    $bad := \mathsf{true}$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{e}}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
return $H[y]$

$\underline{\mathsf{lor}(M_0, M_1)\text{:}}$
$lor := \mathsf{true}$
return w

$\underline{\mathsf{init}()\text{:}}$
$(pk, w) := \mathbf{E}_{\mathrm{owf}}.\mathsf{init}()$
return pk

Distinguishing advantage: 0

OAEP

$\mathbf{G}(x)$:

$y^* \stackrel{R}{\leftarrow} \{y \,|\, \mathsf{def}(H[y])\}$
if $lor \wedge \mathbf{E}_{\mathrm{owf}}.\mathsf{guess}(y^* \| (x \oplus H[y^*]))$
    $bad :=$ true
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{e}}}$
return $G[x]$

$\mathbf{H}(y)$:

$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_{\mathrm{p}}-\ell_{\mathrm{e}}}$
return $H[y]$

lor(M_0, M_1):

$lor :=$ true
return w

init():

$(pk, w) := \mathbf{E}_{\mathrm{owf}}.\mathsf{init}()$
return pk

This is always false. This is dead

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_e}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_p - \ell_e}$
return $H[y]$

Distinguishing advantage: 0

$\underline{\mathsf{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\mathsf{init}()\text{:}}$
$(pk, w) := \mathbf{E}_{\mathrm{owf}}.\mathsf{init}()$
return pk

OAEP

$\underline{\mathbf{G}(x)\text{:}}$
$G[x] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{e}}$
return $G[x]$

$\underline{\mathbf{H}(y)\text{:}}$
$H[y] \stackrel{R}{\leftarrow} \{0,1\}^{\ell_\mathrm{p}-\ell_\mathrm{e}}$
return $H[y]$

$\underline{\text{lor}(M_0, M_1)\text{:}}$
$lor :=$ true
return w

$\underline{\text{init}()\text{:}}$
$(pk, w) := \mathbf{E}_\mathrm{owf}.\text{init}()$
return pk

Probability of setting bad is at most $q_G \cdot \varepsilon_\mathrm{owf}$

We should now add up all the advantages we encountered. The result will be

$$q_G q_H \varepsilon_\mathrm{owf} + \frac{q_G + 1}{2^{\ell_\mathrm{p}-\ell_\mathrm{e}}} + \frac{q_H}{2^{\ell_\mathrm{e}}}$$

Secure information flow

- Programs in a simple imperative language. The set of variables $\mathbf{Var}$ is partitioned into $\mathbf{Var}_{\mathrm{H}}$ and $\mathbf{Var}_{\mathrm{L}}$.
- $P ::= x := E \,|\, \mathbf{skip} \,|\, P_1 \,;\, P_2 \,|\, \mathbf{if}\ b\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2 \,|\, \mathbf{while}\ b\ \mathbf{do}\ P$
- $E ::= x \,|\, o(E_1, \ldots, E_k)$.
- The semantics of o is a probabilistic function from $(\{0,1\}^*)^k$ to $\{0,1\}^*$.
- Program state — a mapping from $\mathbf{Var}$ to $\{0,1\}^*$.
- The semantics $[\![P]\!]$ maps the initial state S_0 to the probability distribution D of final states.
- $S_1 \sim_{\mathrm{L}} S_2$ iff $S_1(x) = S_2(x)$ for all $x \in \mathbf{Var}_{\mathrm{L}}$.
- $D_1 \sim_{\mathrm{L}} D_2$ iff for all states S, $\Pr[S_1 \sim_{\mathrm{L}} S \,|\, S_1 \leftarrow D_1] = \Pr[S_2 \sim_{\mathrm{L}} S \,|\, S_2 \leftarrow D_2]$.
- P has secure information flow if $S_1 \sim_{\mathrm{L}} S_2$ implies $[\![P]\!](S_1) \sim_{\mathrm{L}} [\![P]\!](S_2)$.

Types for secure information flow

- Variable, expression and command types: h and l.
  - Expression of type l does not depend on secret data.
  - Variable of type t may store and stores data of level t.
  - Command of type h does not assign to variables of type l.
- A typing Γ maps variables to types.
- The types are ordered: $l \leq h$.
- This defines us the operations for least upper bound and greatest lower bound.

Type system

$$\frac{\Gamma(x) = t}{\Gamma \vdash x : t} \qquad \frac{\Gamma \vdash E : t_1 \quad t_1 \leq t_2}{\Gamma \vdash E : t_2} \qquad \frac{\Gamma \vdash E_1 : t \quad \cdots \quad \Gamma \vdash E_k : t}{\Gamma \vdash o(E_1, \ldots, E_k) : t}$$

$$\frac{\Gamma(x) = t \quad \Gamma \vdash E : t}{\Gamma \vdash x := E : t} \qquad \frac{}{\Gamma \vdash \mathbf{skip} : h} \qquad \frac{\Gamma \vdash P : t_1 \quad t_1 \geq t_2}{\Gamma \vdash P : t_2}$$

$$\frac{\Gamma \vdash P_1 : t \quad \Gamma \vdash P_2 : t}{\Gamma \vdash P_1; P_2 : t} \qquad \frac{\Gamma \vdash b : t \quad \Gamma \vdash P_1 : t \quad \Gamma \vdash P_2 : t}{\Gamma \vdash \mathbf{if}\ b\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2 : t}$$

$$\frac{\Gamma \vdash b : t \quad \Gamma \vdash P : t}{\Gamma \vdash \mathbf{while}\ b\ \mathbf{do}\ P : t}$$

If $\Gamma \vdash P : t$ then P has secure information flow, where the levels of the variables are given by Γ.

Type system and program transformation

$$\frac{\Gamma(x) = l \quad \Gamma \vdash E : l}{\Gamma \vdash x := E : l \hookrightarrow x := E} \qquad \frac{\Gamma(x) = h}{\Gamma \vdash x := E : h \hookrightarrow \mathbf{skip}}$$

$$\frac{}{\Gamma \vdash \mathbf{skip} : h \hookrightarrow \mathbf{skip}} \qquad \frac{\Gamma \vdash P : t_1 \hookrightarrow P' \quad t_1 \geq t_2}{\Gamma \vdash P : t_2 \hookrightarrow P'}$$

$$\frac{\Gamma \vdash P_1 : t \hookrightarrow P_1' \quad \Gamma \vdash P_2 : t \hookrightarrow P_2'}{\Gamma \vdash P_1; P_2 : t \hookrightarrow P_1'; P_2'}$$

$$\frac{\Gamma \vdash b : l \quad \Gamma \vdash P_1 : l \hookrightarrow P_1' \quad \Gamma \vdash P_2 : l \hookrightarrow P_2'}{\Gamma \vdash \mathbf{if}\ b\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2 : l \hookrightarrow \mathbf{if}\ b\ \mathbf{then}\ P_1'\ \mathbf{else}\ P_2'}$$

$$\frac{\Gamma \vdash b : l \quad \Gamma \vdash P : l \hookrightarrow P'}{\Gamma \vdash \mathbf{while}\ b\ \mathbf{do}\ P : l \hookrightarrow \mathbf{while}\ b\ \mathbf{do}\ P'}$$

$$\frac{\Gamma \vdash b : h \quad \Gamma \vdash P_1 : h \hookrightarrow P_1' \quad \Gamma \vdash P_2 : h \hookrightarrow P_2'}{\Gamma \vdash \mathbf{if}\ b\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2 : h \hookrightarrow \mathbf{skip}}$$

$$\frac{\Gamma \vdash b : h \quad \Gamma \vdash P : h \hookrightarrow P'}{\Gamma \vdash \mathbf{while}\ b\ \mathbf{do}\ P : h \hookrightarrow \mathbf{skip}}$$

Type system and program transformation

$$\frac{\Gamma(x) = l \quad \Gamma \vdash E : l}{\Gamma \vdash x := E : l \hookrightarrow x := E}$$

$$\frac{\Gamma(x) = h}{\Gamma \vdash x := E : h \hookrightarrow \mathbf{skip}}$$

$$\Gamma \vdash P : t_1 \hookrightarrow P' \quad t_1 \geq t_2$$

$$P : t_2 \hookrightarrow P'$$

$$: t \hookrightarrow P_2'$$

$$; P_2'$$

$$\vdash P_2 : l \hookrightarrow P_2'$$

$$b \ \mathbf{then} \ P_1' \ \mathbf{else} \ P_2'$$

$$\to P'$$

$$\mathbf{ile} \ b \ \mathbf{do} \ P'$$

$$\vdash P_2 : h \hookrightarrow P_2'$$

$$\Gamma \vdash \mathbf{if} \ b \ \mathbf{then} \ P_1 \ \mathbf{else} \ P_2 : h \hookrightarrow \mathbf{skip}$$

$$\frac{\Gamma \vdash b : h \quad \Gamma \vdash P : h \hookrightarrow P'}{\Gamma \vdash \mathbf{while} \ b \ \mathbf{do} \ P : h \hookrightarrow \mathbf{skip}}$$

If $\Gamma \vdash P : t \hookrightarrow P'$ then

- P has secure information flow;
  - levels of variables are given by Γ
- For all states S, $[\![P]\!](S) \sim_{\mathrm{L}} [\![P']\!](S)$
  - Easy to prove inductively.
- P' does not use variables in $\Gamma^{-1}(\{h\})$

Hence P' justifies the typing of P.

Symmetric encryption in programs

- Operations kgen (nullary) and enc (binary).
- IND-CPA security: $\mathbf{E}_{\mathrm{enc}}$ has methods:
  - init(): $b \stackrel{R}{\leftarrow} \{0,1\}$; $k_1 :=$ kgen(); if $b = 1$ then $k_2 :=$ kgen() else $k_2 := k_1$.
  - encrypt(i, x): if $b = 1$ then return enc(k_i, x) else return enc(k_i, C), where C is a fixed constant.

Symmetric encryption in programs

- Operations kgen (nullary) and enc (binary).
- IND-CPA security: $\mathbf{E}_{\text{enc}}$ has methods:
  - init(): $b \stackrel{R}{\leftarrow} \{0,1\}$; $k_1 :=$ kgen(); if $b = 1$ then $k_2 :=$ kgen() else $k_2 := k_1$.
  - encrypt(i,x): if $b = 1$ then return enc(k_i, x) else return enc(k_i, C), where C is a fixed constant.
- In terms of programs it means that
  - Let ℓ_1, ℓ_2 be two locations in the program where a key is generated.
  - Let the keys generated at ℓ_1, ℓ_2 be used only in encryptions.
  - Then
    - the second location may be deleted (made an assignment of a key generated at the first location)
    - All encryptions with keys generated at ℓ_1 or ℓ_2 may be replaced by the encryptions of C.

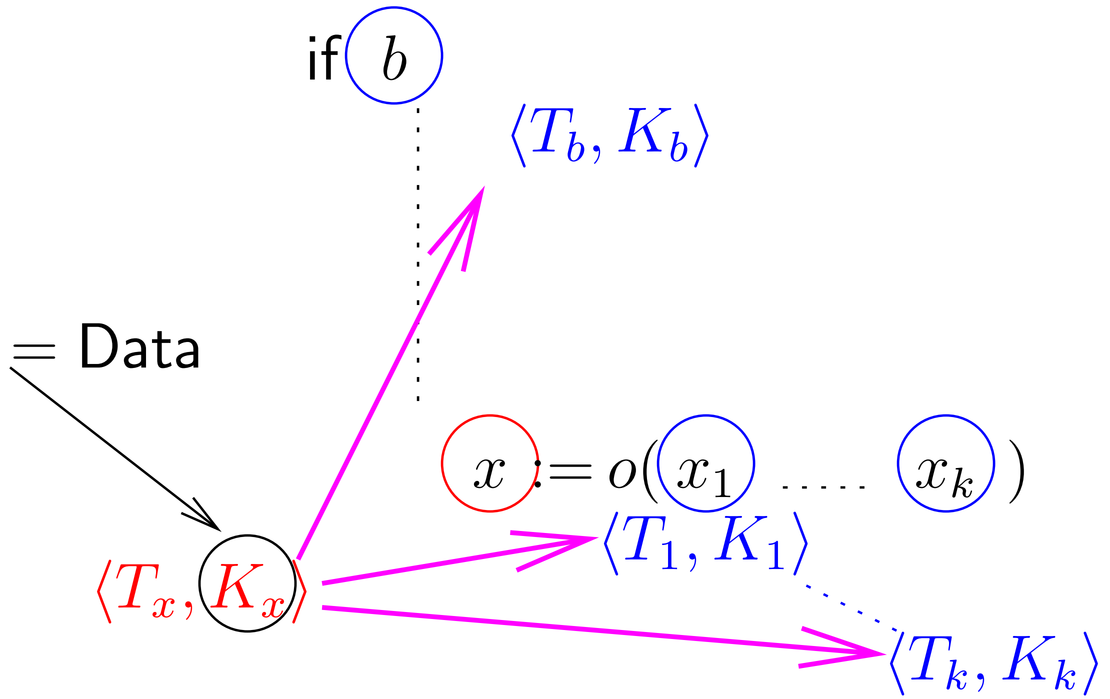
Computational SIF

- Previous definition of secure information flow is too strong.
- Computationally secure information flow:
  - $b \stackrel{R}{\leftarrow} \{0,1\}$;
  - adversary chooses states S_0, S_1, such that $S_0 \sim_{\mathrm{L}} S_1$.
  - $S \leftarrow [\![P]\!](S_b)$, give $S|_{\mathbf{Var}_{\mathrm{L}}}$ to adverary.
  - Adversary tries to guess b.

A type system for CSIF

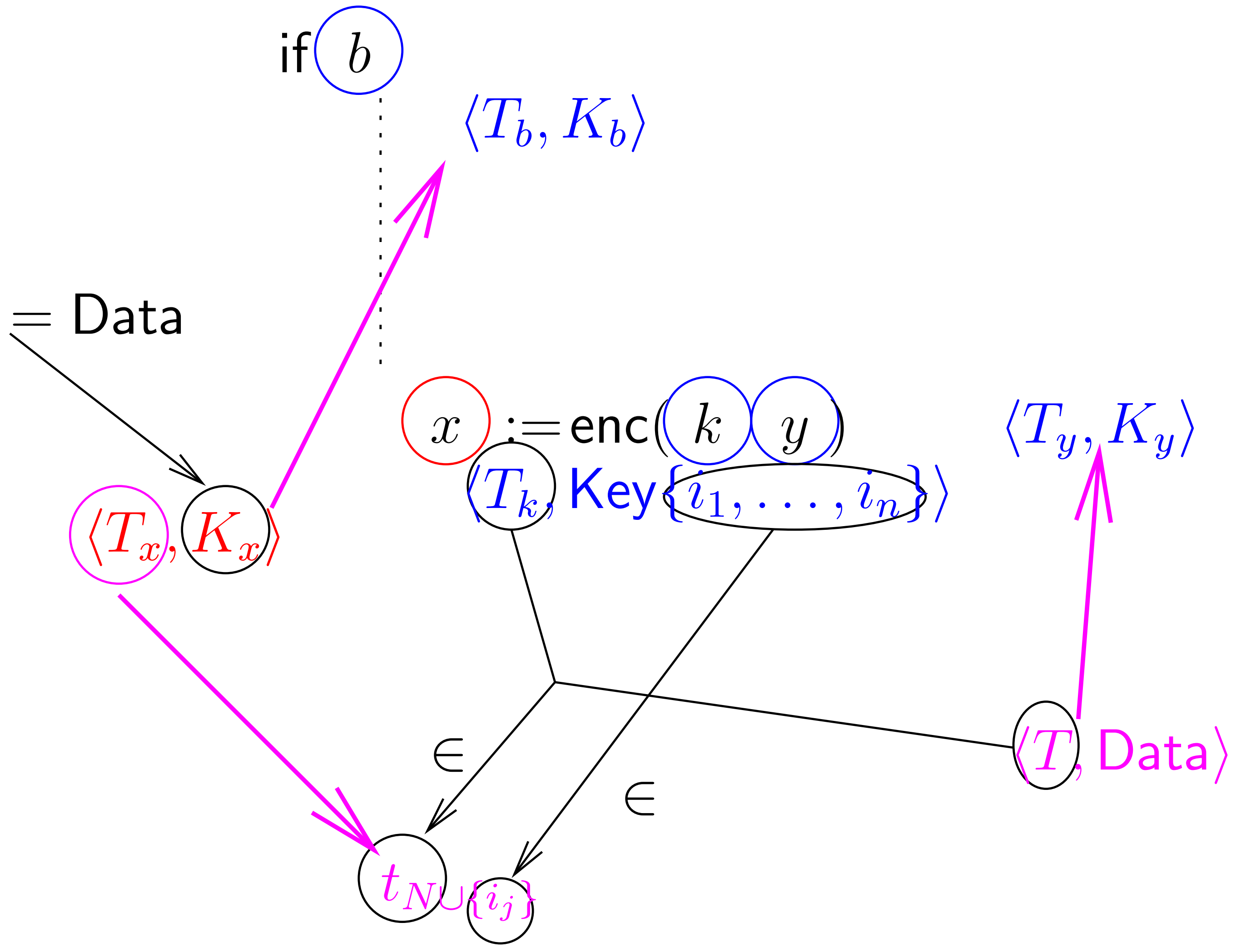
- Let $\mathcal{L}$ be the set of points in program P where a key is generated.
- Resources $\mathcal{R} = \{h\} \cup \mathcal{L}$.
- Basic types $\mathcal{T}_0 = \{t_K \,|\, t \in \mathcal{R}, K \subseteq \mathcal{L}\}$.
  - Order: $t_K \leq t'_{K'}$ if $t = t'$ and $K \supseteq K'$.
- Information types $\mathcal{T} = \mathcal{P}(\mathcal{T}_0)/\equiv$.
  - $\{\ell_{\ell'}, \ell'\} \equiv \{\ell, \ell'\}$ and similar...
  - $\equiv$ expresses our ability to use keys for decryption.
  - Order: $T_1 \leq T_2$ if for all $t_K \in T_1$ exists $t'_{K'} \in T_2$ such that $t_K \leq t'_{K'}$.
- Usage types $\mathcal{U} = \{\mathsf{Key}_K \,|\, K \subseteq \mathcal{L}\}$
- We assign an information type $\Gamma_I(x)$ and a usage type $\Gamma_U(x)$ to each variable.
  - Pairs of information and usage types are ordered, too.
- The program statements constrain the possible types of variables.
- The program has CSIF if a valid typing exists.

General assignments

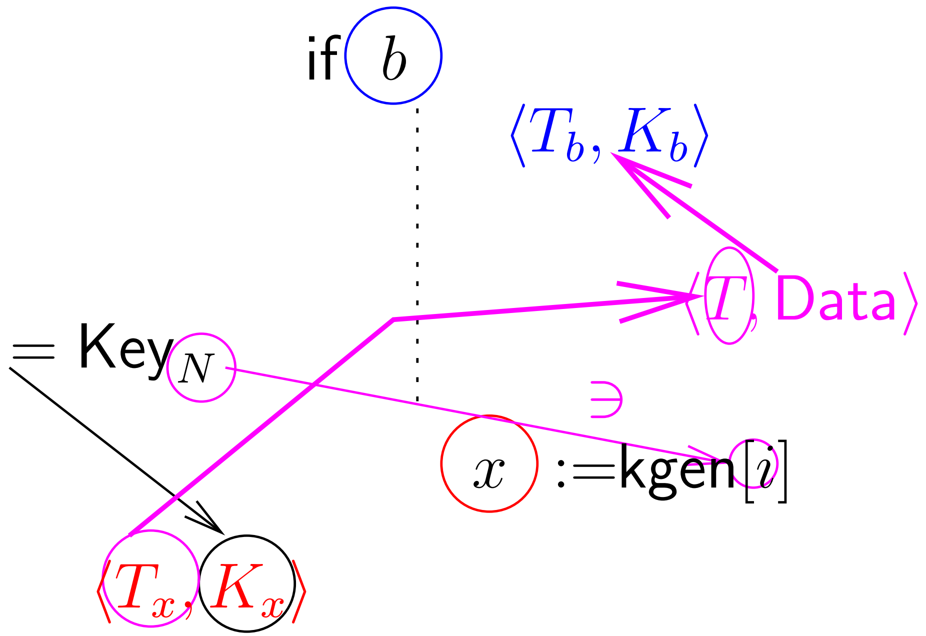


- Here $\longrightarrow$ means $\geq$.
- The next slides will present special cases. These are alternatives to the general scheme.

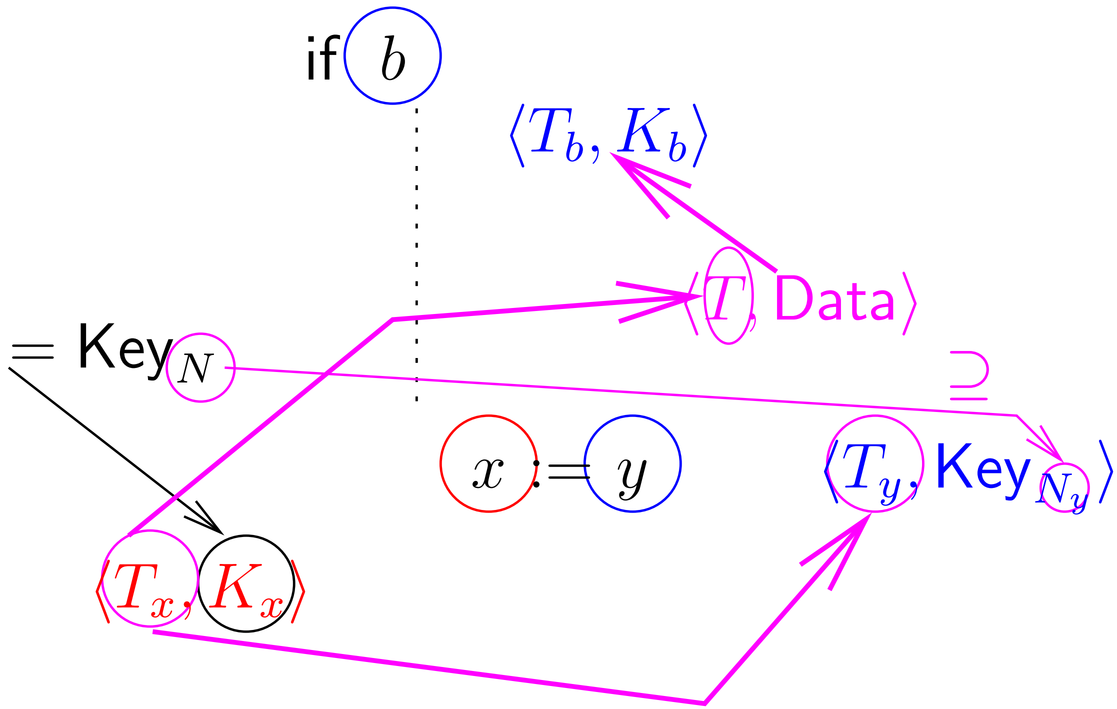
Encryptions



Key generations



Assigning one key to another



Example program

$k := \mathsf{kgen}[1]$
if b *then*
  $l := k$
  $y := \mathsf{kgen}[2]$
else
  $l := \mathsf{kgen}[3]$
  $y := \mathsf{kgen}[4]$
$x := \mathsf{enc}(l, y)$
$z := \mathsf{enc}(y, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle$ $s : \langle \{h\}, \mathsf{Data} \rangle$
$k : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l : \langle \{h\}, \mathsf{Key}_{1,3} \rangle$ $y : \langle \{h\}, \mathsf{Key}_{2,4} \rangle$
$x : \langle \{h_1, h_3, 2_1, 2_3, 4_1, 4_3\}, \mathsf{Data} \rangle$
$z : \langle \{h_2, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_1, h_2, h_3, h_4, 2_1, 2_3, 4_1, 4_3\}$

- $\Gamma(\mathbf{Var}_{\mathrm{L}}) = \bigvee_{x \in \mathbf{Var}_{\mathrm{L}}} \Gamma(x)$.
- If $\Gamma(\mathbf{Var}_{\mathrm{L}}) \not\geq \{h\}$ then the program has CSIF.

Keep key generations separate

$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
  $l^{(t)} := k^{(t)}; l_1 := k_1$
  $y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
  $l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
  $y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
  $v := y_2$
else
  $v := y_4$
$x := \mathsf{case}_{\mathsf{enc}}(l^{(t)} \| 1, l_1, v | 3, l_3, v)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle$ $s : \langle \{h\}, \mathsf{Data} \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle$ $k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle$ $l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle$ $y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \{h, 2, 4\}, \mathsf{Data} \rangle$
$x : \langle \{h_1, h_3, 2_1, 2_3, 4_1, 4_3\}, \mathsf{Data} \rangle$
$z : \langle \{h_2, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_1, h_2, h_3, h_4, 2_1, 2_3, 4_1, 4_3\}$

Add a key to encrypt C

$\mathfrak{k} :=$ kgen$[0]$
$k^{(t)} := 1; k_1 :=$ kgen$[1]$
if b *then*
  $l^{(t)} := k^{(t)}; l_1 := k_1$
  $y^{(t)} := 2; y_2 :=$ kgen$[2]$
else
  $l^{(t)} := 3; l_3 :=$ kgen$[3]$
  $y^{(t)} := 4; y_4 :=$ kgen$[4]$
if $y^{(t)} = 2$ *then*
  $v := y_2$
else
  $v := y_4$
$x :=$ case$_{\mathsf{enc}}(l^{(t)} \| 1, l_1, v | 3, l_3, v)$
$z :=$ case$_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle$ $s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle$ $k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle$ $l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle$ $y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \{h, 2, 4\}, \mathsf{Data} \rangle$
$x : \langle \{h_1, h_3, 2_1, 2_3, 4_1, 4_3\}, \mathsf{Data} \rangle$
$z : \langle \{h_2, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_1, h_2, h_3, h_4, 2_1, 2_3, 4_1, 4_3\}$

Find a key generation to handle

- We had $\Gamma_I(\mathbf{Var}_{\mathrm{L}}) = \{h_1, h_2, h_3, h_4, 2_1, 2_3, 4_1, 4_3\}$.
- It still contains h.
- Find some $\ell \in \mathfrak{L}$ that only appears as a key in $\Gamma_I(\mathbf{Var}_{\mathrm{L}})$.
  - There always must be one.
  - These are 1 ja 3. Let us choose 1.
- Remove all accesses to variables x where $\Gamma_I(x)$ contains 1 at the position of data.
- In our case, this operation does not do anything.
- Apply the cryptographic transformation.

Choose encryptions with keys 0 and 1

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
  $l^{(t)} := k^{(t)}; l_1 := k_1$
  $y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
  $l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
  $y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
  $v := y_2$
else
  $v := y_4$
$x := \mathsf{case}_{\mathsf{enc}}(l^{(t)} \| 1, l_1, v | 3, l_3, v)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle$ $s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle$ $k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle$ $l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle$ $y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \{h, 2, 4\}, \mathsf{Data} \rangle$
$x : \langle \{h_1, h_3, 2_1, 2_3, 4_1, 4_3\}, \mathsf{Data} \rangle$
$z : \langle \{h_2, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_1, h_2, h_3, h_4, 2_1, 2_3, 4_1, 4_3\}$

Replace plaintext with C

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
$\quad l^{(t)} := k^{(t)}; l_1 := k_1$
$\quad y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
$\quad l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
$\quad y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
$\quad v := y_2$
else
$\quad v := y_4$
$x := \mathsf{case}_{\mathsf{enc}}(l^{(t)} \| 1, \mathfrak{k}, C | 3, l_3, v)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle \quad s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle \quad k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle \quad l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle \quad y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \{h, 2, 4\}, \mathsf{Data} \rangle$
$x : \langle \{h_0, h_3, 2_3, 4_3\}, \mathsf{Data} \rangle$
$z : \langle \{h_2, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_0, h_2, h_3, h_4, 2_3, 4_3\}$

Note how the types change.

Choose the next key (3)

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
$\quad l^{(t)} := k^{(t)}; l_1 := k_1$
$\quad y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
$\quad l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
$\quad y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
$\quad v := y_2$
else
$\quad v := y_4$
$x := \mathsf{case}_{\mathsf{enc}}(l^{(t)} \| 1, \mathfrak{k}, C | 3, l_3, v)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle \quad s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle \quad k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle \quad l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle \quad y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \{h, 2, 4\}, \mathsf{Data} \rangle$
$x : \langle \{h_0, h_3, 2_3, 4_3\}, \mathsf{Data} \rangle$
$z : \langle \{h_2, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_0, h_2, h_3, h_4, 2_3, 4_3\}$

Replace plaintext with C

$\mathfrak{k}$:= kgen$[0]$
$k^{(t)} := 1; k_1 :=$ kgen$[1]$
if b *then*
  $l^{(t)} := k^{(t)}; l_1 := k_1$
  $y^{(t)} := 2; y_2 :=$ kgen$[2]$
else
  $l^{(t)} := 3; l_3 :=$ kgen$[3]$
  $y^{(t)} := 4; y_4 :=$ kgen$[4]$
if $y^{(t)} = 2$ *then*
  $v := y_2$
else
  $v := y_4$
$x :=$ case$_{\mathsf{enc}}(l^{(t)} \| 1, \mathfrak{k}, C | 3, \mathfrak{k}, C)$
$z :=$ case$_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle$ $s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle$ $k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle$ $l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle$ $y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \{h, 2, 4\}, \mathsf{Data} \rangle$
$x : \langle \{h_0\}, \mathsf{Data} \rangle$
$z : \langle \{h_2, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_0, h_2, h_4\}$

$\mathsf{case}_{\mathsf{enc}}$ with equal branches $\Rightarrow$ enc

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
$\quad l^{(t)} := k^{(t)}; l_1 := k_1$
$\quad y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
$\quad l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
$\quad y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
$\quad v := y_2$
else
$\quad v := y_4$
$x := \mathsf{enc}(\mathfrak{k}, C)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle\{h\}, \mathsf{Data}\rangle$ $\quad s : \langle\{h\}, \mathsf{Data}\rangle$
$\mathfrak{k} : \langle\emptyset, \mathsf{Key}_0\rangle$
$k^{(t)} : \langle\emptyset, \mathsf{Data}\rangle$ $\quad k_1 : \langle\emptyset, \mathsf{Key}_1\rangle$
$l^{(t)} : \langle\{h\}, \mathsf{Data}\rangle$
$l_1 : \langle\{h\}, \mathsf{Key}_1\rangle$ $\quad l_3 : \langle\{h\}, \mathsf{Key}_3\rangle$
$y^{(t)} : \langle\{h\}, \mathsf{Data}\rangle$
$y_2 : \langle\{h\}, \mathsf{Key}_2\rangle$ $\quad y_4 : \langle\{h\}, \mathsf{Key}_4\rangle$
$v : \langle\{h, 2, 4\}, \mathsf{Data}\rangle$
$x : \langle\emptyset, \mathsf{Data}\rangle$
$z : \langle\{h_2, h_4\}, \mathsf{Data}\rangle$
$\mathrm{P} : \{h_2, h_4\}$

Choose a key

- $\Gamma_I(\mathbf{Var}_{\mathrm{L}})$ contains 2 and 4 as keys only. Let us choose 2.
- 2 occurs in $\gamma(v)$ as data.
- Remove all accesses to v — it cannot affect the values of $\mathbf{Var}_{\mathrm{L}}$.

Deleting 2 occurring as data

$\mathfrak{k} :=$ kgen$[0]$
$k^{(t)} := 1; k_1 :=$ kgen$[1]$
if b *then*
  $l^{(t)} := k^{(t)}; l_1 := k_1$
  $y^{(t)} := 2; y_2 :=$ kgen$[2]$
else
  $l^{(t)} := 3; l_3 :=$ kgen$[3]$
  $y^{(t)} := 4; y_4 :=$ kgen$[4]$
if $y^{(t)} = 2$ *then*
  skip
else
  skip
$x :=$ enc$(\mathfrak{k}, C)$
$z :=$ case$_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle$ $s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle$ $k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle$ $l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle$ $y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \emptyset, \mathsf{Data} \rangle$
$x : \langle \emptyset, \mathsf{Data} \rangle$
$z : \langle \{h_2, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_2, h_4\}$

Encryptions with keys 0 and 2. . .

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
$\quad l^{(t)} := k^{(t)}; l_1 := k_1$
$\quad y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
$\quad l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
$\quad y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
$\quad$ *skip*
else
$\quad$ *skip*
$x := \mathsf{enc}(\mathfrak{k}, C)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, y_2, s | 4, y_4, s)$

$b : \langle\{h\}, \mathsf{Data}\rangle$ $\quad s : \langle\{h\}, \mathsf{Data}\rangle$
$\mathfrak{k} : \langle\emptyset, \mathsf{Key}_0\rangle$
$k^{(t)} : \langle\emptyset, \mathsf{Data}\rangle$ $\quad k_1 : \langle\emptyset, \mathsf{Key}_1\rangle$
$l^{(t)} : \langle\{h\}, \mathsf{Data}\rangle$
$l_1 : \langle\{h\}, \mathsf{Key}_1\rangle$ $\quad l_3 : \langle\{h\}, \mathsf{Key}_3\rangle$
$y^{(t)} : \langle\{h\}, \mathsf{Data}\rangle$
$y_2 : \langle\{h\}, \mathsf{Key}_2\rangle$ $\quad y_4 : \langle\{h\}, \mathsf{Key}_4\rangle$
$v : \langle\emptyset, \mathsf{Data}\rangle$
$x : \langle\emptyset, \mathsf{Data}\rangle$
$z : \langle\{h_2, h_4\}, \mathsf{Data}\rangle$
$\mathrm{P} : \{h_2, h_4\}$

... are replaced with $\mathsf{enc}(\mathfrak{k}, C)$

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
  $l^{(t)} := k^{(t)}; l_1 := k_1$
  $y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
  $l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
  $y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
  skip
else
  skip
$x := \mathsf{enc}(\mathfrak{k}, C)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, \mathfrak{k}, C | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle$ $s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle$ $k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle$ $l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle$ $y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \emptyset, \mathsf{Data} \rangle$
$x : \langle \emptyset, \mathsf{Data} \rangle$
$z : \langle \{h_0, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_0, h_4\}$

Encryptions with keys 0 and 4...

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
$\quad l^{(t)} := k^{(t)}; l_1 := k_1$
$\quad y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
$\quad l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
$\quad y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
$\quad$ *skip*
else
$\quad$ *skip*
$x := \mathsf{enc}(\mathfrak{k}, C)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, \mathfrak{k}, C | 4, y_4, s)$

$b : \langle \{h\}, \mathsf{Data} \rangle \quad s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle \quad k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle \quad l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle \quad y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \emptyset, \mathsf{Data} \rangle$
$x : \langle \emptyset, \mathsf{Data} \rangle$
$z : \langle \{h_0, h_4\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_0, h_4\}$

… are replaced with $\mathsf{enc}(\mathfrak{k}, C)$

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
$\quad l^{(t)} := k^{(t)}; l_1 := k_1$
$\quad y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
$\quad l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
$\quad y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
$\quad$ *skip*
else
$\quad$ *skip*
$x := \mathsf{enc}(\mathfrak{k}, C)$
$z := \mathsf{case}_{\mathsf{enc}}(y^{(t)} \| 2, \mathfrak{k}, C | 4, \mathfrak{k}, C)$

$b : \langle \{h\}, \mathsf{Data} \rangle$ $\quad s : \langle \{h\}, \mathsf{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \mathsf{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \mathsf{Data} \rangle$ $\quad k_1 : \langle \emptyset, \mathsf{Key}_1 \rangle$
$l^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$l_1 : \langle \{h\}, \mathsf{Key}_1 \rangle$ $\quad l_3 : \langle \{h\}, \mathsf{Key}_3 \rangle$
$y^{(t)} : \langle \{h\}, \mathsf{Data} \rangle$
$y_2 : \langle \{h\}, \mathsf{Key}_2 \rangle$ $\quad y_4 : \langle \{h\}, \mathsf{Key}_4 \rangle$
$v : \langle \emptyset, \mathsf{Data} \rangle$
$x : \langle \emptyset, \mathsf{Data} \rangle$
$z : \langle \{h_0\}, \mathsf{Data} \rangle$
$\mathrm{P} : \{h_0\}$

$\mathsf{case}_{\mathsf{enc}}$ with equal branches $\Rightarrow$ enc

$\mathfrak{k} := \mathsf{kgen}[0]$
$k^{(t)} := 1; k_1 := \mathsf{kgen}[1]$
if b *then*
    $l^{(t)} := k^{(t)}; l_1 := k_1$
    $y^{(t)} := 2; y_2 := \mathsf{kgen}[2]$
else
    $l^{(t)} := 3; l_3 := \mathsf{kgen}[3]$
    $y^{(t)} := 4; y_4 := \mathsf{kgen}[4]$
if $y^{(t)} = 2$ *then*
    skip
else
    skip
$x := \mathsf{enc}(\mathfrak{k}, C)$
$z := \mathsf{enc}(\mathfrak{k}, C)$

$b : \langle\{h\}, \mathsf{Data}\rangle$ $\quad s : \langle\{h\}, \mathsf{Data}\rangle$
$\mathfrak{k} : \langle\emptyset, \mathsf{Key}_0\rangle$
$k^{(t)} : \langle\emptyset, \mathsf{Data}\rangle$ $\quad k_1 : \langle\emptyset, \mathsf{Key}_1\rangle$
$l^{(t)} : \langle\{h\}, \mathsf{Data}\rangle$
$l_1 : \langle\{h\}, \mathsf{Key}_1\rangle$ $\quad l_3 : \langle\{h\}, \mathsf{Key}_3\rangle$
$y^{(t)} : \langle\{h\}, \mathsf{Data}\rangle$
$y_2 : \langle\{h\}, \mathsf{Key}_2\rangle$ $\quad y_4 : \langle\{h\}, \mathsf{Key}_4\rangle$
$v : \langle\emptyset, \mathsf{Data}\rangle$
$x : \langle\emptyset, \mathsf{Data}\rangle$
$z : \langle\emptyset, \mathsf{Data}\rangle$
$\mathrm{P} : \emptyset$

SIF transformation (h goes away)

$\mathfrak{k} :=$ kgen$[0]$
$k^{(t)} := 1; k_1 :=$ kgen$[1]$
$skip$
$skip$
$x :=$ enc$(\mathfrak{k}, C)$
$z :=$ enc$(\mathfrak{k}, C)$

$b : \langle \{h\}, \text{Data} \rangle$ $\quad s : \langle \{h\}, \text{Data} \rangle$
$\mathfrak{k} : \langle \emptyset, \text{Key}_0 \rangle$
$k^{(t)} : \langle \emptyset, \text{Data} \rangle$ $\quad k_1 : \langle \emptyset, \text{Key}_1 \rangle$
$l^{(t)} : \langle \emptyset, \text{Data} \rangle$
$l_1 : \langle \emptyset, \text{Data} \rangle$ $\quad l_3 : \langle \emptyset, \text{Data} \rangle$
$y^{(t)} : \langle \emptyset, \text{Data} \rangle$
$y_2 : \langle \emptyset, \text{Data} \rangle$ $\quad y_4 : \langle \emptyset, \text{Data} \rangle$
$v : \langle \emptyset, \text{Data} \rangle$
$x : \langle \emptyset, \text{Data} \rangle$
$z : \langle \emptyset, \text{Data} \rangle$
$\mathrm{P} : \emptyset$