

Computer-aided cryptographic proofs

Gilles Barthe

IMDEA Software Institute, Madrid, Spain

Modern cryptography

Shannon '49

- Mathematical proof of security
- Perfect secrecy is impossible

Diffie & Hellman '76

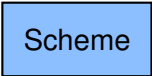
- Computational security
 - Asymptotic guarantees
- PPT adversary has negligible advantage

Goldwasser & Micali '82
Yao '82

Bellare & Rogaway '94

- Concrete bounds
- Aversary advantage to win in time t is $\leq p$

Reductionist proof



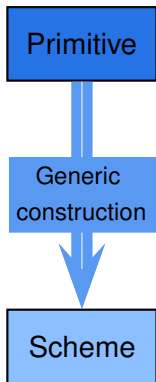
Scheme

Reductionist proof

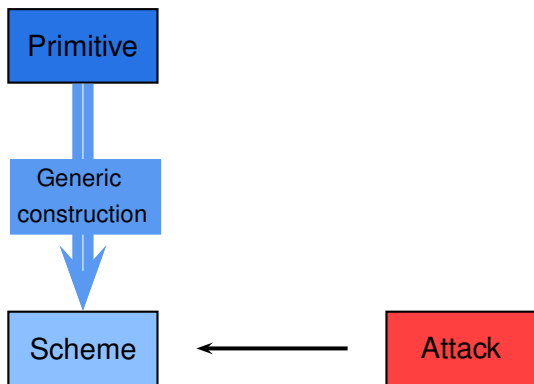
Primitive

Scheme

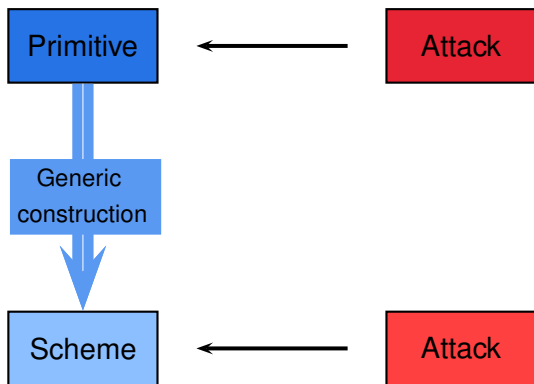
Reductionist proof



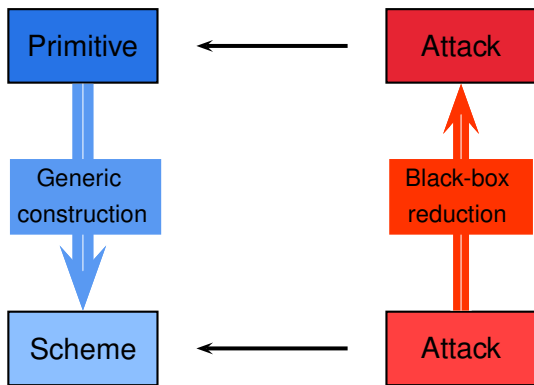
Reductionist proof



Reductionist proof



Reductionist proof



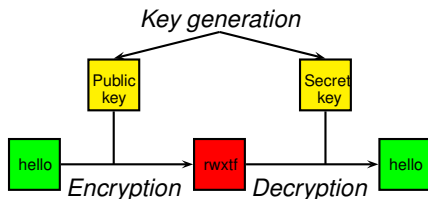
Public-key encryption

Algorithms $(\mathcal{K}, \mathcal{E}_{pk}, \mathcal{D}_{sk})$

- ▶ \mathcal{E} probabilistic
- ▶ \mathcal{D} deterministic and partial

If (sk, pk) is a valid key pair,

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}()$;

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;

$b \xleftarrow{\$} \{0, 1\}$;

$c^* \leftarrow \mathcal{E}_{pk}(m_b)$;

$b' \leftarrow \mathcal{A}_2(c^*)$;

return $(b' = b)$

Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}();$

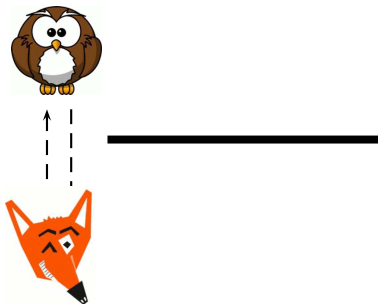
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$

$b \xleftarrow{\$} \{0, 1\};$

$c^* \leftarrow \mathcal{E}_{pk}(m_b);$

$b' \leftarrow \mathcal{A}_2(c^*);$

return $(b' = b)$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}()$;

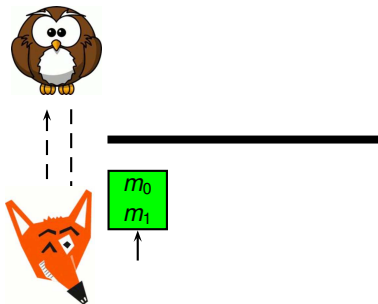
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;

$b \xleftarrow{\$} \{0, 1\}$;

$c^* \leftarrow \mathcal{E}_{pk}(m_b)$;

$b' \leftarrow \mathcal{A}_2(c^*)$;

return $(b' = b)$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}()$;

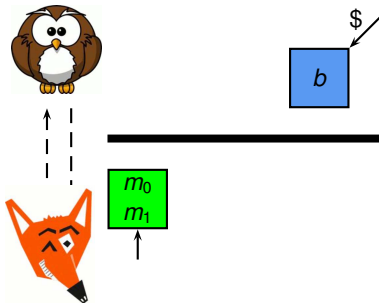
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;

$b \xleftarrow{\$} \{0, 1\}$;

$c^* \leftarrow \mathcal{E}_{pk}(m_b)$;

$b' \leftarrow \mathcal{A}_2(c^*)$;

return $(b' = b)$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}();$

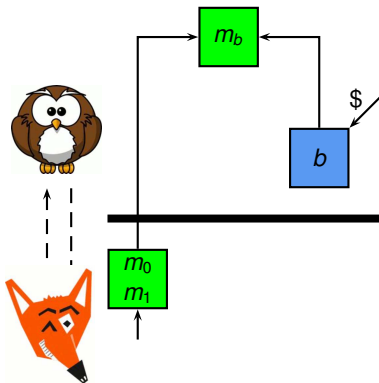
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$

$b \xleftarrow{\$} \{0, 1\};$

$c^* \leftarrow \mathcal{E}_{pk}(m_b);$

$b' \leftarrow \mathcal{A}_2(c^*);$

return $(b' = b)$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}()$;

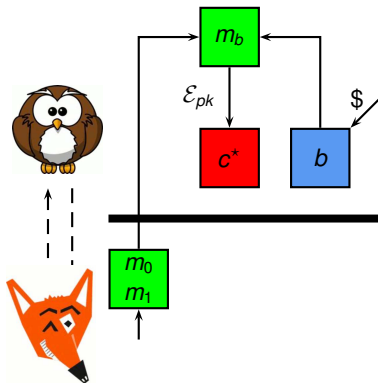
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;

$b \xleftarrow{\$} \{0, 1\}$;

$c^* \leftarrow \mathcal{E}_{pk}(m_b)$;

$b' \leftarrow \mathcal{A}_2(c^*)$;

return $(b' = b)$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}();$

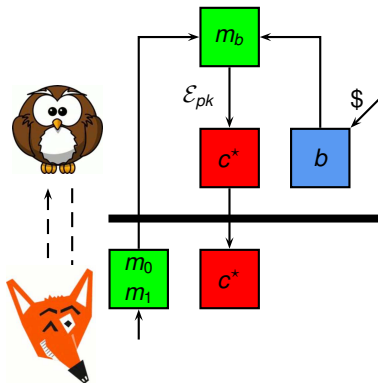
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$

$b \xleftarrow{\$} \{0, 1\};$

$c^* \leftarrow \mathcal{E}_{pk}(m_b);$

$b' \leftarrow \mathcal{A}_2(c^*);$

return $(b' = b)$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}()$;

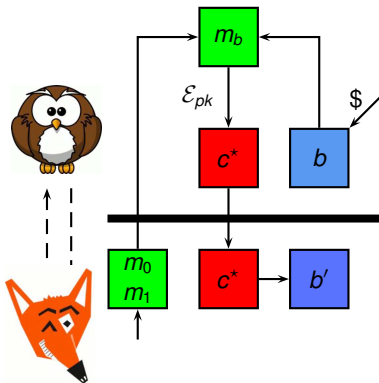
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;

$b \xleftarrow{\$} \{0, 1\}$;

$c^* \leftarrow \mathcal{E}_{pk}(m_b)$;

$b' \leftarrow \mathcal{A}_2(c^*)$;

return $(b' = b)$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}()$;

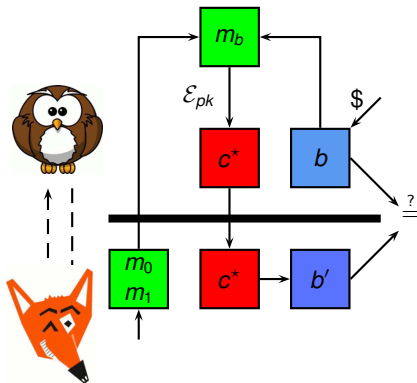
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;

$b \xleftarrow{\$} \{0, 1\}$;

$c^* \leftarrow \mathcal{E}_{pk}(m_b)$;

$b' \leftarrow \mathcal{A}_2(c^*)$;

return $(b' = b)$



Indistinguishability

Game IND-CPA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}()$;

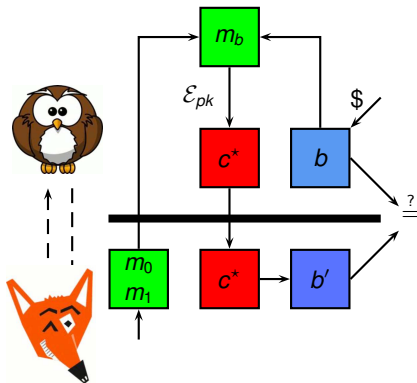
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;

$b \xleftarrow{\$} \{0, 1\}$;

$c^* \leftarrow \mathcal{E}_{pk}(m_b)$;

$b' \leftarrow \mathcal{A}_2(c^*)$;

return $(b' = b)$



$$\Pr_{\text{IND-CPA}(\mathcal{A})} [b' = b] - \frac{1}{2} \text{ small}$$

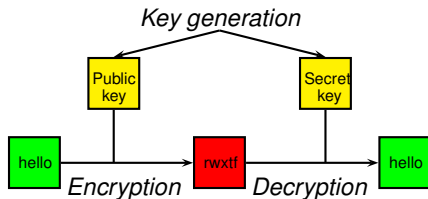
One-way trapdoor permutations

Algorithms $(\mathcal{K}, f_{pk}, f_{sk}^{-1})$

- ▶ f_{pk} and f_{sk}^{-1} deterministic

If (sk, pk) is a valid key pair,

$$f_{sk}^{-1}(f_{pk}(m)) = m$$



One-way trapdoor permutations

Game $\text{OW}(\mathcal{I})$

$(sk, pk) \leftarrow \mathcal{K}();$

$y \xleftarrow{\$} \{0, 1\}^n;$

$x^* \leftarrow f_{pk}(y);$

$y' \leftarrow \mathcal{I}(x^*);$

return $(y' = y)$



One-way trapdoor permutations

Game $\text{OW}(\mathcal{I})$

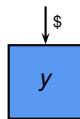
$(sk, pk) \leftarrow \mathcal{K}();$

$y \xleftarrow{\$} \{0, 1\}^n;$

$x^* \leftarrow f_{pk}(y);$

$y' \leftarrow \mathcal{I}(x^*);$

return $(y' = y)$



One-way trapdoor permutations

Game $\text{OW}(\mathcal{I})$

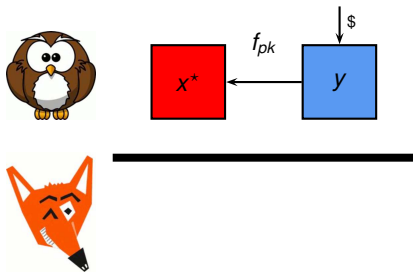
$(sk, pk) \leftarrow \mathcal{K}()$;

$y \xleftarrow{\$} \{0, 1\}^n$;

$x^* \leftarrow f_{pk}(y)$;

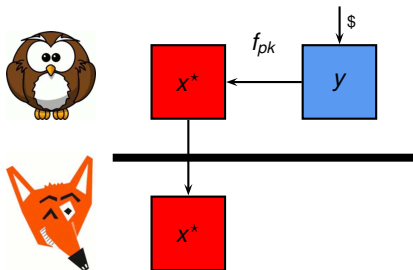
$y' \leftarrow \mathcal{I}(x^*)$;

return $(y' = y)$



One-way trapdoor permutations

Game $\text{OW}(\mathcal{I})$
 $(sk, pk) \leftarrow \mathcal{K}()$;
 $y \xleftarrow{\$} \{0, 1\}^n$;
 $x^* \leftarrow f_{pk}(y)$;
 $y' \leftarrow \mathcal{I}(x^*)$;
return $(y' = y)$



One-way trapdoor permutations

Game $\text{OW}(\mathcal{I})$

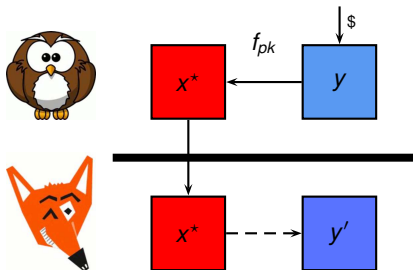
$(sk, pk) \leftarrow \mathcal{K}()$;

$y \xleftarrow{\$} \{0, 1\}^n$;

$x^* \leftarrow f_{pk}(y)$;

$y' \leftarrow \mathcal{I}(x^*)$;

return $(y' = y)$



One-way trapdoor permutations

Game $OW(\mathcal{I})$

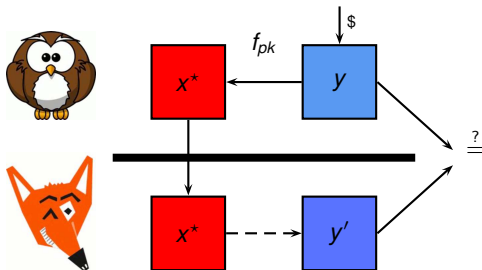
$(sk, pk) \leftarrow \mathcal{K}()$;

$y \xleftarrow{\$} \{0, 1\}^n$;

$x^* \leftarrow f_{pk}(y)$;

$y' \leftarrow \mathcal{I}(x^*)$;

return $(y' = y)$



One-way trapdoor permutations

Game $\text{OW}(\mathcal{I})$

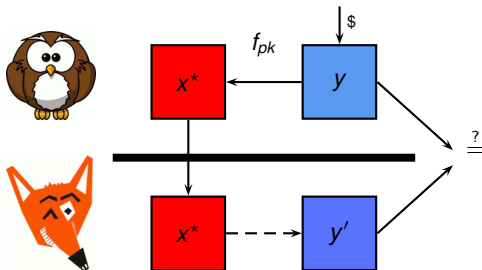
$(sk, pk) \leftarrow \mathcal{K}()$;

$y \xleftarrow{\$} \{0, 1\}^n$;

$x^* \leftarrow f_{pk}(y)$;

$y' \leftarrow \mathcal{I}(x^*)$;

return $(y' = y)$



$\Pr_{\text{OW}(\mathcal{I})}[y' = y]$ small

Random oracles

Oracle $H(x)$:

if $x \notin L$ then

$r \xleftarrow{\$} \{0, 1\}^k$;

$L \leftarrow (x, r) :: L$;

return $L[x]$;

- ▶ Idealized model of hash function
- ▶ Allows practical schemes
- ▶ Not realizable

Example: Bellare and Rogaway 1993 encryption

Game IND-CPA(\mathcal{A}) :

$(sk, pk) \leftarrow \mathcal{K}(\);$

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$

$b \xleftarrow{\$} \{0, 1\};$

$c^* \leftarrow \mathcal{E}_{pk}(m_b);$

$b' \leftarrow \mathcal{A}_2(c^*);$

return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell;$

$s \leftarrow H(r) \oplus m;$

$y \leftarrow f_{pk}(r) \parallel s;$

return y

For every IND-CPA adversary \mathcal{A} , there exists an inverter \mathcal{I} st

$$\Pr_{\text{IND-CPA}(\mathcal{A})}[b' = b] - \frac{1}{2} \leq \Pr_{\text{OW}(\mathcal{I})}[y' = y]$$

Proof

Game hopping technique

Game INDCPA :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $c^* \leftarrow \mathcal{E}_{pk}(m_b)$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell$;
 $h \leftarrow H(r)$;
 $s \leftarrow h \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

Game G :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $c^* \leftarrow \mathcal{E}_{pk}(m_b)$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell$;
 $h \xleftarrow{\$} \{0, 1\}^k$;
 $s \leftarrow h \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

Game G' :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $c^* \leftarrow \mathcal{E}_{pk}(m_b)$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $h \leftarrow s \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

Game OW :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $y \xleftarrow{\$} \{0, 1\}^\ell$;
 $y' \leftarrow \mathcal{I}(f_{pk}(y))$;
return $y = y'$

Adversary $\mathcal{I}(x)$:
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $c^* \leftarrow x \parallel s$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
 $y' \leftarrow [z \in L_H^A \mid f_{pk}(z) = x]$;
return y'

1. Prove a probability claim for each hop
2. Obtain security bound by combining claims
3. Check execution time of constructed adversary

Conditional equivalence

```
 $\mathcal{E}_{pk}(m) :$   
 $r \xleftarrow{\$} \{0, 1\}^{\ell};$   
 $h \leftarrow H(r);$   
 $s \leftarrow h \oplus m;$   
 $c \leftarrow f_{pk}(r) \parallel s;$   
return  $c$ 
```



```
 $\mathcal{E}_{pk}(m) :$   
 $r \xleftarrow{\$} \{0, 1\}^{\ell};$   
 $h \xleftarrow{\$} \{0, 1\}^k;$   
 $s \leftarrow h \oplus m;$   
 $c \leftarrow f_{pk}(r) \parallel s;$   
return  $c$ 
```

By the Fundamental Lemma

$$\Pr_{\text{IND-CPA}}[b' = b] - \Pr_{\mathbf{G}}[b' = b] \leq \Pr_{\mathbf{G}}[r \in L_H^A]$$

Equivalence

```
 $\mathcal{E}_{pk}(m) :$   
 $r \xleftarrow{\$} \{0, 1\}^\ell;$   
 $h \xleftarrow{\$} \{0, 1\}^k;$   
 $s \leftarrow h \oplus m;$   
 $c \leftarrow f_{pk}(r) \parallel s;$   
return  $c$ 
```



```
 $\mathcal{E}_{pk}(m) :$   
 $r \xleftarrow{\$} \{0, 1\}^\ell;$   
 $s \xleftarrow{\$} \{0, 1\}^k;$   
 $h \leftarrow s \oplus m;$   
 $c \leftarrow f_{pk}(r) \parallel s;$   
return  $c$ 
```

By optimistic sampling

$$\Pr_{\mathbf{G}}[r \in L_H^A] = \Pr_{\mathbf{G}'}[r \in L_H^A] \quad \Pr_{\mathbf{G}}[b' = b] = \Pr_{\mathbf{G}'}[b' = b] = \frac{1}{2}$$

Equivalence

```
 $\mathcal{E}_{pk}(m) :$   
 $r \xleftarrow{\$} \{0, 1\}^\ell;$   
 $h \xleftarrow{\$} \{0, 1\}^k;$   
 $s \leftarrow h \oplus m;$   
 $c \leftarrow f_{pk}(r) \parallel s;$   
return  $c$ 
```



```
 $\mathcal{E}_{pk}(m) :$   
 $r \xleftarrow{\$} \{0, 1\}^\ell;$   
 $s \xleftarrow{\$} \{0, 1\}^k;$   
 $h \leftarrow s \oplus m;$   
 $c \leftarrow f_{pk}(r) \parallel s;$   
return  $c$ 
```

By optimistic sampling

$$\Pr_{\text{IND-CPA}}[b' = b] - \frac{1}{2} \leq \Pr_{\mathbf{G}'}[r \in L_H^A]$$

Reduction

Game IND CPA :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $c^* \leftarrow \mathcal{E}_{pk}(m_b)$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

Game OW :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $y \xleftarrow{\$} \{0, 1\}^\ell$;
 $y' \leftarrow \mathcal{I}(f_{pk}(y))$;
return $y = y'$

Adversary $\mathcal{I}(x)$:

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $c^* \leftarrow x \parallel s$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
 $y' \leftarrow [z \in L_H^A \mid f_{pk}(z) = x]$;
return y'

$$\Pr_{\mathbf{G}'} [r \in L_H^A] \leq \Pr_{\text{OW}(\mathcal{I})} [y' = y]$$

Reduction

Game IND CPA :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $c^* \leftarrow \mathcal{E}_{pk}(m_b)$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

Game OW :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $y \xleftarrow{\$} \{0, 1\}^\ell$;
 $y' \leftarrow \mathcal{I}(f_{pk}(y))$;
return $y = y'$

Adversary $\mathcal{I}(x)$:

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $c^* \leftarrow x \parallel s$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
 $y' \leftarrow [z \in L_H^A \mid f_{pk}(z) = x]$;
return y'

$$\Pr_{\text{IND-CPA}(\mathcal{A})}[b' = b] - \frac{1}{2} \leq \Pr_{\text{OW}(\mathcal{I})}[y' = y]$$

Plug-and-pray inverter

Game IND CPA :

$(sk, pk) \leftarrow \mathcal{K}();$
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $b \xleftarrow{\$} \{0, 1\};$
 $c^* \leftarrow \mathcal{E}_{pk}(m_b);$
 $b' \leftarrow \mathcal{A}_2(c^*);$
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell;$
 $s \xleftarrow{\$} \{0, 1\}^k;$
 $c \leftarrow f_{pk}(r) \parallel s;$
return c

Game OW :

$(sk, pk) \leftarrow \mathcal{K}();$
 $y \xleftarrow{\$} \{0, 1\}^\ell;$
 $y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$

Adversary $\mathcal{I}'(x)$:

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $b \xleftarrow{\$} \{0, 1\};$
 $s \xleftarrow{\$} \{0, 1\}^k;$
 $c^* \leftarrow x \parallel s;$
 $b' \leftarrow \mathcal{A}_2(c^*);$
 $i \xleftarrow{\$} [1..q_H];$
 $y' \leftarrow L_H^A[i];$
return y'

$$\Pr_{\text{IND-CPA}(\mathcal{A})}[b' = b] - \frac{1}{2} \leq q_H \Pr_{\text{OW}(\mathcal{I}')}[y' = y]$$

Optimal Asymmetric Encryption Padding

Encryption $\mathcal{E}_{\text{OAEP}(pk)}(m) :$

$r \xleftarrow{\$} \{0, 1\}^{k_0};$

$s \leftarrow G(r) \oplus (m \parallel 0^{k_1});$

$t \leftarrow H(s) \oplus r;$

return $f_{pk}(s \parallel t)$

Decryption $\mathcal{D}_{\text{OAEP}(sk)}(c) :$

$(s, t) \leftarrow f_{sk}^{-1}(c);$

$r \leftarrow t \oplus H(s);$

if $([s \oplus G(r)]_{k_1} = 0^{k_1})$

then $\{m \leftarrow [s \oplus G(r)]^k;\}$

else $\{m \leftarrow \perp;\}$

return m

\oplus exclusive or \parallel concatenation $[\cdot]$ projection 0 zero bitstring

OAEP: provable security

Game IND-CCA(\mathcal{A})

$(sk, pk) \leftarrow \mathcal{K}();$
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $b \xleftarrow{\$} \{0, 1\};$
 $c^* \leftarrow \mathcal{E}_{pk}(m_b);$
 $b' \leftarrow \mathcal{A}_2(c^*);$
return $(b' = b)$

Game SPDOW(\mathcal{I})

$(sk, pk) \leftarrow \mathcal{K}();$
 $y \xleftarrow{\$} \{0, 1\}^{k_2}; z \xleftarrow{\$} \{0, 1\}^{k_3};$
 $x^* \leftarrow f_{pk}(y \| z);$
 $Y' \leftarrow \mathcal{I}(x^*);$
return $(y \in Y')$

FOR ALL IND-CCA adversary \mathcal{A} against $(\mathcal{K}, \mathcal{E}_{\text{OAEP}}, \mathcal{D}_{\text{OAEP}})$,
THERE EXISTS a SPDOW adversary \mathcal{I} against (\mathcal{K}, f, f^{-1}) st

$$\left| \Pr_{\text{IND-CCA}(\mathcal{A})}[b' = b] - \frac{1}{2} \right| \leq \\ \Pr_{\text{SPDOW}(\mathcal{I})}[y \in Y'] + \frac{3q_D q_G + q_D^2 + 4q_D + q_G}{2^{k_0}} + \frac{2q_D}{2^{k_1}}$$

and

$$t_{\mathcal{I}} \leq t_{\mathcal{A}} + q_D q_G q_H T_f$$

Key length

- ▶ Estimation: factoring RSA-768 takes 2^{67} operations
- ▶ Extrapolation: factoring N -bit RSA modulus takes time \propto

$$\exp((1.9229 + o(1)) \log(N)^{1/3} \log(\log(N))^{2/3})$$

Modulus size	Number of Operations
512	2^{58}
768	2^{67}
1024	2^{77}
2048	2^{107}
3072	2^{129}
4096	2^{147}
5120	2^{162}
6144	2^{176}
7680	2^{193}
8192	2^{199}
15360	2^{259}

Practical interpretation for RSA-OAEP

- ▶ Reduction from PDOW to OW. Let $\ell < 2k$

$$\mathbf{Succ}_{RSA}^{\text{SPDOW}_k^q}(t) \left(\mathbf{Succ}_{RSA}^{\text{SPDOW}_k^q}(t) - 2^{\ell-2k+6} \right) \leq \mathbf{Succ}_{RSA}^{\text{OW}_{2k}}(2t + q^2 \ell^3)$$

- ▶ Set bounds to adversary queries

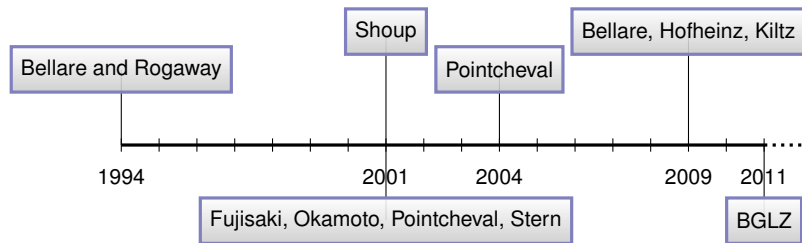
$$q_D \leq 2^{30}$$

$$q_G, q_H \leq 2^{60}$$

- ▶ Derive recommended (overly conservative) key size

4096

OAEP: provable security



1994 Purported proof of chosen-ciphertext security

2001 1994 proof gives weaker security; desired security holds
▶ for a modified scheme ▶ under stronger assumptions

2004 Filled gaps in 2001 proof

2009 Security definition needs to be clarified

2011 Fills gaps in 2004 proof

Implementation of OAEP

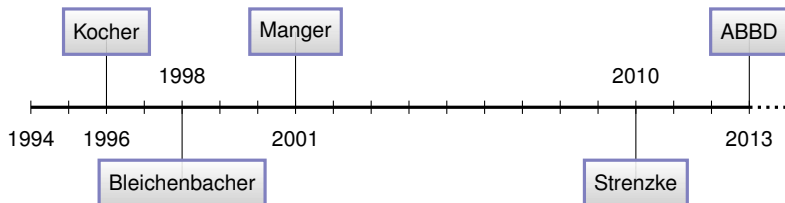
Decryption $\mathcal{D}_{\text{OAEP}(sk)}(c)$:

```
(s, t) ← fsk-1(c);  
r ← t ⊕ H(s);  
if ([s ⊕ G(r)]k1 = 0k1)  
  then {m ← [s ⊕ G(r)]k; }  
  else {m ← ⊥; }  
return m
```

Decryption $\mathcal{D}_{\text{PKCS-C}(sk)}(res, c)$:

```
if (c ∈ MsgSpace(sk)) then  
{ (b0, s, t) ← fsk-1(c);  
  h ← MGF(s, hLen); i ← 0;  
  while (i < hLen + 1)  
  { s[i] ← t[i] ⊕ h[i]; i ← i + 1; }  
  g ← MGF(r, dbLen); i ← 0;  
  while (i < dbLen)  
  { p[i] ← s[i] ⊕ g[i]; i ← i + 1; }  
  l ← payload_length(p);  
  if (b0 = 08 ∧ [p]lhLen = 0..01 ∧  
      [p]hLen = LHash)  
  then  
  { rc ← Success;  
    memcpy(res, 0, p, dbLen - l, l); }  
  else { rc ← DecryptionError; } }  
else { rc ← CiphertextTooLong; }  
return rc;
```

OAEP: real-world security



Attacks from observing

- ▶ error messages
- ▶ execution time
 - ☞ RSA implementations
 - ☞ conversion from integers to bitstrings
(RSA operates on strict subset of $[0..2^k]$)

Problems with cryptographic proofs

Unverifiable proofs

- ▶ Proofs are long and error-prone
- ▶ Rely on unstated and unverified invariants
- ▶ Intricate reasoning steps justified informally

[...] many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor.

Bellare and Rogaway, 2004-2006

Abstraction gap

- ▶ Provable security reasons about algorithmic descriptions
- ▶ Standards constrain implementations
- ▶ Attackers target executable code

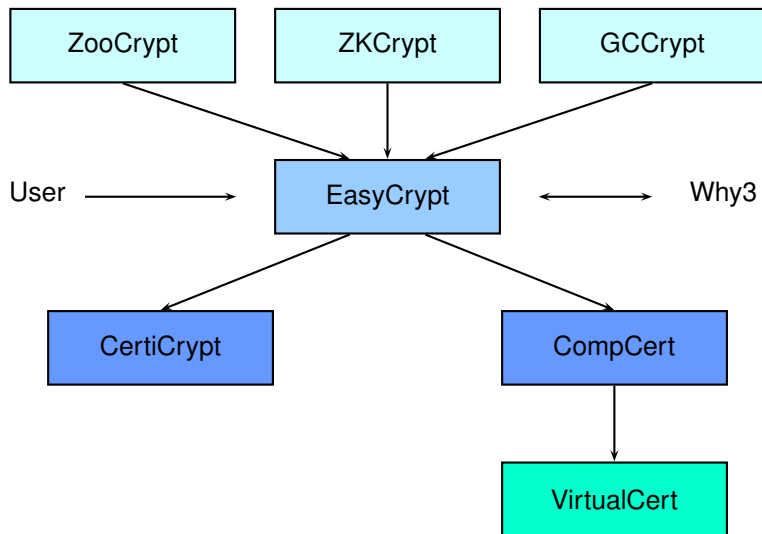
Real-world crypto is breakable; is in fact being broken; is one of many ongoing disaster areas in security. Bernstein, 2013

Computer-aided cryptographic proofs

Provable security as deductive relational verification
of open probabilistic parametrized programs

- ▶ High-confidence reductionist proofs
 - ☞ machine-checked, independently verifiable proofs
 - ☞ adhere to cryptographic practice
(same formalisms, guarantees and proof techniques)
- ▶ Manage complexity of real-world cryptography
- ▶ Increase confidence in implementations
 - ☞ minimize gap between proofs and code
 - ☞ prove effectiveness of countermeasures
(on source code and machine code)
- ▶ Leverage existing verification techniques and tools
 - ☞ program logics, VC generation, invariant generation
 - ☞ SMT solvers, theorem provers, proof assistants
 - ☞ verified compilers

EasyCrypt toolchain



A language for cryptographic games

$\mathcal{C} ::=$	skip	skip
	$\mathcal{V} \leftarrow \mathcal{E}$	assignment
	$\mathcal{V} \xleftarrow{s} \mathcal{D}$	random sampling
	$\mathcal{C}; \mathcal{C}$	sequence
	if \mathcal{E} then \mathcal{C} else \mathcal{C}	conditional
	while \mathcal{E} do \mathcal{C}	while loop
	$\mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \dots, \mathcal{E})$	procedure call

- ▶ \mathcal{E} : (higher-order) expressions
 - ▶ \mathcal{D} : discrete sub-distributions
 - ▶ \mathcal{P} : procedures
- } user extensible
- . oracles: concrete procedures
 - . adversaries: constrained abstract procedures

Program semantics

- ▶ A discrete sub-distribution over A is a map $\mu : A \rightarrow [0, 1]$ st.
 - ☞ $\sum_{a \in A} \mu(a) \leq 1$
 - ☞ $\text{supp}(\mu) = \{a \in A \mid \mu(a) > 0\}$ is discrete
- ▶ Programs as sub-distribution transformers: $\llbracket c \rrbracket$
 - ☞ takes as input a memory m
 - ☞ returns a discrete sub-distribution over memories
- ▶ Probability of an event

$$\text{Pr}_{c,m}[E] = \sum_{m' \in E} \llbracket c \rrbracket_m m'$$

Given $f : A \rightarrow [0, 1]$, define

$$\mu^*(f) = \sum_{a \in A} f(a) \times \mu(\mathbb{1}_a)$$

Program semantics

$$\llbracket x \leftarrow e \rrbracket m = \mathbb{1}_{\{[m|x \mapsto \llbracket e \rrbracket m]\}}$$

$$\llbracket x \stackrel{\$}{\leftarrow} \mu \rrbracket m = \sum_{d \in \text{dom}(\llbracket \mu \rrbracket m)} \mathbb{1}_{\{[m|x \mapsto d]\}}$$

$$\llbracket c_1; c_2 \rrbracket m = \llbracket c_2 \rrbracket^* (\llbracket c_1 \rrbracket m)$$

$$\llbracket \text{if } b \text{ then } c_1 \text{ else } c_2 \rrbracket m = \text{if } \llbracket b \rrbracket m \text{ then } \llbracket c_1 \rrbracket m \text{ else } \llbracket c_2 \rrbracket m$$

$$\llbracket \text{while } b \text{ do } c \rrbracket = \text{fix } (\lambda f. \lambda m. \text{if } \llbracket b \rrbracket m \text{ then } f^* (\llbracket c \rrbracket m) \text{ else } \mathbb{1}_{\{m\}})$$

Alternative characterization

$$\llbracket \text{while } e \text{ do } c \rrbracket m = \lambda f. \sup_{n \in \mathbb{N}} (\llbracket [\text{while } e \text{ do } c]_n \rrbracket m f)$$

where

$$\begin{aligned} \llbracket \text{while } e \text{ do } c \rrbracket_0 &= \text{abort} \\ \llbracket \text{while } e \text{ do } c \rrbracket_{n+1} &= \text{if } e \text{ then } (c; \llbracket \text{while } e \text{ do } c \rrbracket_n) \end{aligned}$$

Reasoning about cryptographic games

- ▶ Probabilistic Relational Hoare Logic
- ▶ Probabilistic Hoare Logic
- ▶ Program optimizations
- ▶ Ambient logic (inc. quantification over modules) allows
 - ☞ hybrid arguments
 - ☞ modular proofs
 - ☞ meta-arguments

(Deductive) program verification

- ▶ Art of proving that programs are correct
- ▶ Origins:
 - ☞ axiomatic semantics (Hoare'69)
 - ☞ weakest precondition calculus (Floyd'67)
- ▶ Major advances in:
 - ☞ language coverage
 - ☞ automation
 - ☞ proof engineering

Hoare logic

- ▶ Judgments $c : P \Longrightarrow Q$
(P and Q are f.o. formulae over program variables)
- ▶ A judgment $c : P \Longrightarrow Q$ is valid iff

$$\forall m, m' \models P \Rightarrow \llbracket c \rrbracket m = m' \Rightarrow m' \models Q$$

Selected rules

$$\frac{c : P \Longrightarrow Q \quad P' \Rightarrow P \quad Q \Rightarrow Q'}{c : P' \Longrightarrow Q'}$$

$$\frac{}{x \leftarrow e : Q[e/x] \Longrightarrow Q}$$

$$\frac{c_1 : P \Longrightarrow Q \quad c_2 : Q \Longrightarrow R}{c_1; c_2 : P \Longrightarrow R}$$

$$\frac{c_1 : P \wedge e \Longrightarrow Q \quad c_2 : P \wedge \neg e \Longrightarrow Q}{\text{if } e \text{ then } c_1 \text{ else } c_2 : P \Longrightarrow Q}$$

$$\frac{c : I \wedge e \Longrightarrow I}{\text{while } e \text{ do } c : I \Longrightarrow I \wedge \neg e}$$

Verification condition generation

- ▶ Generate a set of verification conditions from annotated command and postcondition
- ▶ If all VCs are valid and $P \Rightarrow wp(c, Q)$ then $c : P \Longrightarrow Q$

Selected rules

$$wp(x \leftarrow e, Q) = Q[e/x]$$

$$wp(c_1; c_2, R) = wp(c_1, wp(c_2, R))$$

$$wp(\text{if } e \text{ then } c_1 \text{ else } c_2, Q) = e \Rightarrow wp(c_1, Q) \wedge \neg e \Rightarrow wp(c_2, Q)$$

$$wp(\text{while}_I e \text{ do } c, Q) = I$$

The while rule generates two proof obligations

$$I \wedge e \Rightarrow wp(c, I) \quad I \wedge \neg e \Rightarrow Q$$

Beyond safety

2-safety and 2-programs safety

- ▶ 2 executions of the same program: information flow
- ▶ 2 programs: program equivalence

- ▶ Judgments

$$\models \{P\} c_1 \sim c_2 \{Q\}$$

(P and Q are f.o. formulae over tagged program variables)

- ▶ Validity

$$\forall m_1, m_2. (m_1, m_2) \models P \implies (\llbracket c_1 \rrbracket m_1, \llbracket c_2 \rrbracket m_2) \models Q$$

- ▶ Verification methods

- ☞ embedding into Hoare logic
- ☞ relational Hoare logic

pRHL: probabilistic relational Hoare logic

- ▶ Judgment

$$\vDash \{P\} c_1 \sim c_2 \{Q\}$$

where P and Q denote relations on memories

- ▶ Validity

$$\forall m_1, m_2. (m_1, m_2) \vDash P \implies (\llbracket c_1 \rrbracket m_1, \llbracket c_2 \rrbracket m_2) \vDash Q^\sharp$$

- ▶ Definition of \cdot^\sharp drawn from probabilistic process algebra

Deriving probability claims

Assume $\models \{P\} c_1 \sim c_2 \{Q\}$ and $(m_1, m_2) \models P$

Equivalence

- ▶ If $Q \stackrel{\text{def}}{=} \bigwedge_{x \in X} x\langle 1 \rangle = x\langle 2 \rangle$ and $\text{FV}(A) \subseteq X$ then

$$\Pr_{c_1, m_1}[A] = \Pr_{c_2, m_2}[A]$$

- ▶ If $Q \stackrel{\text{def}}{=} A\langle 1 \rangle \Leftrightarrow B\langle 2 \rangle$ then

$$\Pr_{c_1, m_1}[A] = \Pr_{c_2, m_2}[B]$$

Conditional equivalence

- ▶ If $Q \stackrel{\text{def}}{=} \neg F\langle 2 \rangle \Rightarrow \bigwedge_{x \in X} x\langle 1 \rangle = x\langle 2 \rangle$ and $\text{FV}(A) \subseteq X$ then

$$\Pr_{c_1, m_1}[A] - \Pr_{c_2, m_2}[A] \leq \Pr_{c_2, m_2}[F]$$

- ▶ If $Q \stackrel{\text{def}}{=} \neg F\langle 2 \rangle \Rightarrow (A\langle 1 \rangle \Leftrightarrow B\langle 2 \rangle)$ then

$$\Pr_{c_1, m_1}[A] - \Pr_{c_2, m_2}[B] \leq \Pr_{c_2, m_2}[F]$$

Lifting Relations to sub-distributions

Existential definition

$(\mu_1, \mu_2) \models Q^\sharp$ iff there exists $\mu \in \mathcal{D}(\mathcal{M} \times \mathcal{M})$ s.t.

- ▶ $\pi_i(\mu) = \mu_i$, where $\pi_1(\mu)(a) = \sum_{b \in B} \mu(a, b)$
- ▶ $\text{supp}(\mu) \subseteq Q$, i.e. $\mu(a, b) > 0 \Rightarrow Q(a, b)$

Inductive definition

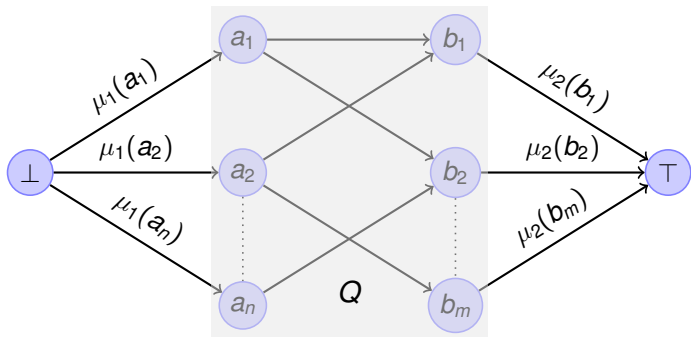
- ▶ If $(s, t) \models Q$ then $(\delta_s, \delta_t) \models Q^\sharp$
- ▶ If $(\mu_i, \nu_i) \models Q^\sharp$ and $\sum_i p_i = 1$, then

$$\left(\sum_i p_i \mu_i, \sum_i p_i \nu_i \right) \models Q^\sharp$$

Flow network definition

$(\mu_1, \mu_2) \models Q^\sharp$ iff the maximum flow in the induced network is 1

Flow networks



Proof rules: random assignment

Intuition

Let A be a finite set and let $f, g : A \rightarrow B$. Define

- ▶ $c = x \stackrel{\$}{\leftarrow} \mu; y \leftarrow f x$
- ▶ $c' = x \stackrel{\$}{\leftarrow} \mu'; y \leftarrow g x$

Then $\llbracket c \rrbracket = \llbracket c' \rrbracket$ (extensionally) iff there exists $h : A \xrightarrow{1-1} A$ st

- ▶ $f = g \circ h$
- ▶ for all a , $\mu(a) = \mu'(h(a))$

$$\frac{h \text{ is 1-1 and } \forall a, \mu(a) = \mu'(h(a))}{\models \{\forall v, Q[h v/x\langle 1 \rangle][v/x\langle 2 \rangle]\} \quad x \stackrel{\$}{\leftarrow} \mu \sim x \stackrel{\$}{\leftarrow} \mu' \quad \{Q\}}$$

- ▶ Rule captures a special case of lifting
- ▶ General rule might lead to untractable arithmetic equalities

Proof rule: assignments and conditionals

Assignments

$$\frac{}{\models \{Q[e\langle 1 \rangle/x\langle 1 \rangle][e'\langle 2 \rangle/x'\langle 2 \rangle]\} x \leftarrow e \sim x' \leftarrow e' \{Q\}}$$

$$\frac{}{\models \{Q[x\langle 1 \rangle := e\langle 1 \rangle]\} x \leftarrow e \sim \text{skip} \{Q\}}$$

Conditionals

$$\frac{\begin{array}{l} P \Rightarrow e\langle 1 \rangle = e'\langle 2 \rangle \\ \models \{P \wedge e\langle 1 \rangle\} c_1 \sim c'_1 \{Q\} \quad \models \{P \wedge \neg e\langle 1 \rangle\} c_2 \sim c'_2 \{Q\} \end{array}}{\models \{P\} \text{ if } e \text{ then } c_1 \text{ else } c_2 \sim \text{if } e' \text{ then } c'_1 \text{ else } c'_2 \{Q\}}$$

$$\frac{\models \{P \wedge e\langle 1 \rangle\} c_1 \sim c \{Q\} \quad \models \{P \wedge \neg e\langle 1 \rangle\} c_2 \sim c \{Q\}}{\models \{P\} \text{ if } e \text{ then } c_1 \text{ else } c_2 \sim c \{Q\}}$$

Loops

Two-sided rule

$$\frac{P \Rightarrow e\langle 1 \rangle = e'\langle 2 \rangle \quad \models \{P \wedge e\langle 1 \rangle\} c \sim c' \{P\}}{\models \{P\} \text{ while } e \text{ do } c \sim \text{ while } e' \text{ do } c' \{P \wedge \neg e\langle 1 \rangle\}}$$

- ▶ rule is incomplete: same number of iterations

One sided-rules

- ▶ standard rule with losslessness verification condition

Optimizations

- ▶ unrolling
- ▶ loop fission/fusion/. . .

Adversaries

$$\frac{\forall \mathcal{O}. \models \{Q \wedge =_W\} z \leftarrow \mathcal{O}(\vec{w}) \sim z \leftarrow \mathcal{O}(\vec{w}) \{Q \wedge =_{\{z\}}\}}{\models \{Q \wedge =_Y\} x \leftarrow \mathcal{A}(\vec{y}) \sim x \leftarrow \mathcal{A}(\vec{y}) \{Q \wedge =_{\{x\}}\}}$$

- ▶ Adversaries perform arbitrary sequences of oracle calls (and intermediate computations)
- ▶ No functional specification
- ▶ Given the same inputs, provide the same outputs

Reasoning about Failure Events

Lemma (Fundamental Lemma)

Let A, B, F be events and G_1, G_2 be two games such that

$$\Pr_{G_1}[A \wedge \neg F] = \Pr_{G_2}[B \wedge \neg F]$$

Then

$$|\Pr_{G_1}[A] - \Pr_{G_2}[B]| \leq \max(\Pr_{G_1}[F], \Pr_{G_2}[F])$$

Fundamental Lemma: syntactic criterion

Let $A = B$ and $F = \text{bad}$. If G_0 and G_1 are syntactically identical until bad is set, e.g.

Game G_0 :

...
 $\text{bad} \leftarrow \text{true}; c_0$
...

Game G_1 :

...
 $\text{bad} \leftarrow \text{true}; c_1$
...

then

- ▶ $\Pr_{G_1, m}[A \mid \neg \text{bad}] = \Pr_{G_2, m}[A \mid \neg \text{bad}]$
- ▶ $\Pr_{G_1, m}[\text{bad}] = \Pr_{G_2, m}[\text{bad}]$

Corollary

$$|\Pr_{G_1, m}[A] - \Pr_{G_2, m}[A]| \leq \Pr_{G_1, 2}[\text{bad}]$$

Fundamental Lemma: syntactic criterion

Let $A = B$ and $F = \text{bad}$. If G_0 and G_1 are syntactically identical until bad is set, e.g.

Game G_0 :
...
 $\text{bad} \leftarrow \text{true}; c_0$
...

Game G_1 :
...
 $\text{bad} \leftarrow \text{true}; c_1$
...

then

- ▶ $\Pr_{G_1, m}[A \mid \neg \text{bad}] = \Pr_{G_2, m}[A \mid \neg \text{bad}]$
- ▶ $\Pr_{G_1, m}[\text{bad}] = \Pr_{G_2, m}[\text{bad}]$

Corollary

$$|\Pr_{G_1, m}[A] - \Pr_{G_2, m}[A]| \leq \Pr_{G_1, 2}[\text{bad}]$$

Fundamental Lemma in pRHL

Let A, B, F be events and G_1, G_2 be two games such that

$$\models \{\text{true}\} G_1 \sim G_2 \{(F\langle 1 \rangle \leftrightarrow F\langle 2 \rangle) \wedge (\neg F\langle 2 \rangle \rightarrow (A\langle 1 \rangle \leftrightarrow B\langle 2 \rangle))\}$$

then

$$|\Pr_{G_1}[A] - \Pr_{G_2}[B]| \leq \Pr_{G_2}[F] \qquad \Pr_{G_1}[F] = \Pr_{G_2}[F]$$

Simpler variants

$$\models \{\text{true}\} G_1 \sim G_2 \{\neg F\langle 2 \rangle \rightarrow (A\langle 1 \rangle \leftrightarrow B\langle 2 \rangle)\}$$

then

$$|\Pr_{G_1}[A] - \Pr_{G_2}[B]| \leq \Pr_{G_2}[F]$$

Even simpler

$$\models \{\text{true}\} G_1 \sim G_2 \{\neg F\langle 2 \rangle \rightarrow B\langle 2 \rangle \rightarrow A\langle 1 \rangle\}$$

then

$$\Pr_{G_1}[A] - \Pr_{G_2}[B] \leq \Pr_{G_2}[F]$$

Fundamental lemma: adversary

For an adversary \mathcal{A} with oracle access to oracles \mathcal{O}^i , it suffices to check that:

- ▶ F is monotonic

$$\{F\} \mathcal{O}_1^i \{F\} \qquad \{F\} \mathcal{O}_2^i \{F\}$$

- ▶ Oracle calls preserve equivalence up to failure

$$\begin{aligned} & \models y \leftarrow \mathcal{A}^{\mathcal{O}_1^i}(\vec{x}) \sim y \leftarrow \mathcal{A}^{\mathcal{O}_2^i}(\vec{x}) : \\ & (\neg F\langle 2 \rangle \wedge Q \wedge \vec{x}\langle 1 \rangle = \vec{x}\langle 2 \rangle) \implies (\neg F\langle 2 \rangle \rightarrow Q \wedge y\langle 1 \rangle = y\langle 2 \rangle) \end{aligned}$$

pHL: probabilistic Hoare logic

$$[c : P \Longrightarrow Q] \leq \delta \quad [c : P \Longrightarrow Q] = \delta \quad [c : P \Longrightarrow Q] \geq \delta$$

- ▶ P, Q predicates on memories
- ▶ δ a real expression evaluated on initial memory

Interpretation (\leq)

$$\forall m, m \models P \Rightarrow \llbracket c \rrbracket m Q \leq \delta m$$

Sample rules

Assignment

$$[x \leftarrow e : P[e/x] \Longrightarrow P] = 1$$

Sequential composition for \geq

$$\frac{[c_1 : P \Longrightarrow R] \geq \delta_1 \quad [c_2 : R \Longrightarrow Q] \geq \delta_2}{[c_1; c_2 : P \Longrightarrow Q] \geq \delta_1 \delta_2}$$

Sequential composition for \leq

$$\frac{\begin{array}{ll} [c_1 : P \Longrightarrow R] \leq \delta_1 & [c_2 : R \Longrightarrow Q] \leq \delta_2 \\ [c_1 : P \Longrightarrow \neg R] \leq \delta_3 & [c_2 : \neg R \Longrightarrow Q] \leq \delta_4 \end{array}}{[c_1; c_2 : P \Longrightarrow Q] \leq \delta_1 \delta_2 + \delta_3 \delta_4}$$

Applications of probabilistic Hoare Logic

Let P be a precondition.

- ▶ Termination wrt P :

$$[c : P \Longrightarrow \top] = 1$$

- ▶ Cost:

$$[\bar{c} : P \Longrightarrow \text{cost} \leq p] = 1$$

where \bar{c} is an annotated version of c

- ▶ Observational equivalence $c_1 \equiv_{P, \vec{x}} c_2$

$$\forall \vec{a} p, [c_1 : P \Longrightarrow \vec{x} = \vec{a}] \leq p \Leftrightarrow [c_2 : P \Longrightarrow \vec{x} = \vec{a}] \leq p$$

Example: Bellare and Rogaway 1993 encryption

Game IND-CPA(\mathcal{A}) :

$(sk, pk) \leftarrow \mathcal{K}(\);$

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$

$b \xleftarrow{\$} \{0, 1\};$

$c^* \leftarrow \mathcal{E}_{pk}(m_b);$

$b' \leftarrow \mathcal{A}_2(c^*);$

return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell;$

$s \leftarrow H(r) \oplus m;$

$y \leftarrow f_{pk}(r) \parallel s;$

return y

For every IND-CPA adversary \mathcal{A} , there exists an inverter \mathcal{I} st

$$\Pr_{\text{IND-CPA}(\mathcal{A})}[b' = b] - \frac{1}{2} \leq \Pr_{\text{OW}(\mathcal{I})}[y' = y]$$

Proof

Game hopping technique

Game INDCPA :

$(sk, pk) \leftarrow \mathcal{K}();$
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $b \xleftarrow{\$} \{0, 1\};$
 $c^* \leftarrow \mathcal{E}_{pk}(m_b);$
 $b' \leftarrow \mathcal{A}_2(c^*);$
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell;$
 $h \leftarrow H(r);$
 $s \leftarrow h \oplus m;$
 $c \leftarrow f_{pk}(r) \parallel s;$
return c

Game G :

$(sk, pk) \leftarrow \mathcal{K}();$
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $b \xleftarrow{\$} \{0, 1\};$
 $c^* \leftarrow \mathcal{E}_{pk}(m_b);$
 $b' \leftarrow \mathcal{A}_2(c^*);$
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell;$
 $h \xleftarrow{\$} \{0, 1\}^k;$
 $s \leftarrow h \oplus m;$
 $c \leftarrow f_{pk}(r) \parallel s;$
return c

Game G' :

$(sk, pk) \leftarrow \mathcal{K}();$
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $b \xleftarrow{\$} \{0, 1\};$
 $c^* \leftarrow \mathcal{E}_{pk}(m_b);$
 $b' \leftarrow \mathcal{A}_2(c^*);$
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell;$
 $s \xleftarrow{\$} \{0, 1\}^k;$
 $h \leftarrow s \oplus m;$
 $c \leftarrow f_{pk}(r) \parallel s;$
return c

Game OW :

$(sk, pk) \leftarrow \mathcal{K}();$
 $y \xleftarrow{\$} \{0, 1\}^\ell;$
 $y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$

Adversary $\mathcal{I}(x)$:

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $s \xleftarrow{\$} \{0, 1\}^k;$
 $c^* \leftarrow x \parallel s;$
 $b' \leftarrow \mathcal{A}_2(c^*);$
 $y' \leftarrow [z \in L_H^A \mid f_{pk}(z) = x];$
return y'

1. For each hop
 - ▶ prove validity of pRHL judgment
 - ▶ derive probability claims
 - ▶ (possibly) resolve some probability expressions using pHL
2. Obtain security bound by combining claims
3. Check execution time of constructed adversary

Conditional equivalence

$\mathcal{E}_{pk}(m)$:
 $r \xleftarrow{\$} \{0, 1\}^\ell$;
 $h \leftarrow H(r)$;
 $s \leftarrow h \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c



$\mathcal{E}_{pk}(m)$:
 $r \xleftarrow{\$} \{0, 1\}^\ell$;
 $h \xleftarrow{\$} \{0, 1\}^k$;
 $s \leftarrow h \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

$$\models \{T\} \text{ IND-CPA} \sim \mathbf{G} \left\{ (\neg r \in L_H^A) \langle 2 \rangle \rightarrow =_{b,b'} \right\}$$

$$\Pr_{\text{IND-CPA}} [b' = b] - \Pr_{\mathbf{G}} [b' = b] \leq \Pr_{\mathbf{G}} [r \in L_H^A]$$

Equivalence

$\mathcal{E}_{pk}(m)$:
 $r \xleftarrow{\$} \{0, 1\}^\ell$;
 $h \xleftarrow{\$} \{0, 1\}^k$;
 $s \leftarrow h \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c



$\mathcal{E}_{pk}(m)$:
 $r \xleftarrow{\$} \{0, 1\}^\ell$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $h \leftarrow s \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

$$\models \{T\} \mathbf{G} \sim \mathbf{G}' \left\{ =_{b, b', r, L_H^A} \right\}$$

$$\Pr_{\mathbf{G}} [r \in L_H^A] = \Pr_{\mathbf{G}'} [r \in L_H^A] \quad \Pr_{\mathbf{G}} [b' = b] = \Pr_{\mathbf{G}'} [b' = b] = \frac{1}{2}$$

Equivalence

$\mathcal{E}_{pk}(m)$:
 $r \xleftarrow{\$} \{0, 1\}^\ell$;
 $h \xleftarrow{\$} \{0, 1\}^k$;
 $s \leftarrow h \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c



$\mathcal{E}_{pk}(m)$:
 $r \xleftarrow{\$} \{0, 1\}^\ell$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $h \leftarrow s \oplus m$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

$$\models \{T\} \mathbf{G} \sim \mathbf{G}' \left\{ =_{b, b', r, L_H^A} \right\}$$

$$\Pr_{\text{IND-CPA}}[b' = b] - \frac{1}{2} \leq \Pr_{\mathbf{G}'}[r \in L_H^A]$$

Reduction

Game IND CPA :

$(sk, pk) \leftarrow \mathcal{K}();$
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $b \xleftarrow{\$} \{0, 1\};$
 $c^* \leftarrow \mathcal{E}_{pk}(m_b);$
 $b' \leftarrow \mathcal{A}_2(c^*);$
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m) :$

$r \xleftarrow{\$} \{0, 1\}^\ell;$
 $s \xleftarrow{\$} \{0, 1\}^k;$
 $c \leftarrow f_{pk}(r) \parallel s;$
return c

Game OW :

$(sk, pk) \leftarrow \mathcal{K}();$
 $y \xleftarrow{\$} \{0, 1\}^\ell;$
 $y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$

Adversary $\mathcal{I}(x) :$

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
 $b \xleftarrow{\$} \{0, 1\};$
 $s \xleftarrow{\$} \{0, 1\}^k;$
 $c^* \leftarrow x \parallel s;$
 $b' \leftarrow \mathcal{A}_2(c^*);$
 $y' \leftarrow [z \in L_H^A \mid f_{pk}(z) = x];$
return y'

$$\models \{T\} \mathbf{G}' \sim \text{OW} \left\{ (r \in L_H^A) \langle 1 \rangle \rightarrow (y' = y) \langle 2 \rangle \right\}$$

$$\Pr_{\mathbf{G}'} [r \in L_H^A] \leq \Pr_{\text{OW}(\mathcal{I})} [y' = y]$$

Reduction

Game INDCPA :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $c^* \leftarrow \mathcal{E}_{pk}(m_b)$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:

$r \xleftarrow{\$} \{0, 1\}^\ell$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $c \leftarrow f_{pk}(r) \parallel s$;
return c

Game OW :

$(sk, pk) \leftarrow \mathcal{K}()$;
 $y \xleftarrow{\$} \{0, 1\}^\ell$;
 $y' \leftarrow \mathcal{I}(f_{pk}(y))$;
return $y = y'$

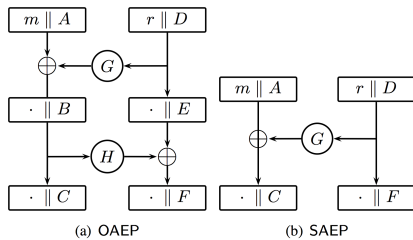
Adversary $\mathcal{I}(x)$:

$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
 $b \xleftarrow{\$} \{0, 1\}$;
 $s \xleftarrow{\$} \{0, 1\}^k$;
 $c^* \leftarrow x \parallel s$;
 $b' \leftarrow \mathcal{A}_2(c^*)$;
 $y' \leftarrow [z \in L_H^A \mid f_{pk}(z) = x]$;
return y'

$$\models \{\text{T}\} \mathbf{G}' \sim \text{OW} \left\{ (r \in L_H^A) \langle 1 \rangle \rightarrow (y' = y) \langle 2 \rangle \right\}$$

$$\Pr_{\text{IND-CPA}(\mathcal{A})}[b' = b] - \frac{1}{2} \leq \Pr_{\text{OW}(\mathcal{I})}[y' = y]$$

Variants of OAEP



Automated proofs and exploration

The next 700 cryptosystems (after Landin, 1966)

Do the cryptosystems reflect [...] the situations that are being catered for? Or are they accidents of history and personal background that may be obscuring fruitful developments?

[...]

We must think in terms, not of cryptosystems, but of families of cryptosystems. That is to say we must systematize their design so that a new system is a point chosen from a well-mapped space, rather than a laboriously devised construction.

The two views of cryptography

Computational cryptography

- ▶ Strong ties with complexity theory
- ▶ Feasible adversary breaks scheme with small probability
- ▶ Design of secure and efficient primitives and protocols
- ▶ Complex and manual proofs

The two views of cryptography

Computational cryptography

- ▶ Strong ties with complexity theory
- ▶ Feasible adversary breaks scheme with small probability
- ▶ Design of secure and efficient primitives and protocols
- ▶ Complex and manual proofs

Symbolic cryptography (Dolev and Yao, 1983)

- ▶ Assume perfect cryptography
- ▶ Adversary cannot win
- ▶ Discovery of logical bugs in protocols
- ▶ Strong ties with verification
- ▶ Automated proofs

Reconciling the two views

(Abadi and Rogaway, 2000)

Computational soundness

Security in symbolic model implies computational security

- ▶ ... under non-standard assumptions on primitives
- ▶ Symbolic tools deliver asymptotic guarantees
- ▶ ... but no concrete guarantees
- ▶ Applicable to many settings
- ▶ ... but some impossibility results

Our approach

Judgment

$$C :_p \varphi$$

- ▶ Reasons about probability of events
- ▶ Concrete probability can be computed
 - ☞ on the fly
 - ☞ a posteriori (judgments use only $p = 0, \frac{1}{2}$)
- ▶ Side-conditions are discharged by symbolic methods
 - ☞ what is the entropy of e ?
 - ☞ can I compute e' from e ?

Also use symbolic methods for

- ▶ Finding attacks
- ▶ Computing decryption algorithm

An algebraic view of padding-based schemes

Encryption algorithms are modelled as algebraic expressions

\mathcal{E}	::=	m	input message
		0	zero bitstring
		\mathcal{R}	uniform random bitstring
		$\mathcal{E} \oplus \mathcal{E}$	xor
		$\mathcal{E} \parallel \mathcal{E}$	concatenation
		$[\mathcal{E}]_s^s$	projection
		$H(\mathcal{E})$	hash
		$f(\mathcal{E})$	trapdoor permutation

Decryption algorithms are modelled using list comprehension

$$\vec{x} \stackrel{c}{\leftarrow} \vec{L}_H^A : T \triangleright e$$

where $T ::= e = e \mid e \in L_H \mid e \in L_H^A \mid T \wedge T$

Semantics

Left-to-right evaluation with sharing, yields a pWHILE procedure

Example

$$f((G(r) \oplus (m \parallel 0)) \parallel H(G(r) \oplus (m \parallel 0)) \oplus r)$$

interpreted as:

```
 $r \xleftarrow{s} \{0, 1\}^k;$   
 $g \leftarrow G(r);$   
 $s \leftarrow g \oplus (m \parallel 0);$   
 $h \leftarrow H(s);$   
return  $f_{pk}(s \parallel (h \oplus r))$ 
```

Proof principles: chosen-plaintext security

Failure event	Replace $H(e)$ by fresh r
Optimistic sampling	Replace $e \oplus r$, where r is fresh, by r
Probability	Compute probability of $b = b'$ or $e \in L$
Reduction	Find inverter and apply one-wayness

plus a few additional rules

Deducibility

$$\frac{}{e \vdash e} \quad \frac{e \vdash e_1 \quad e \vdash e_2}{e \vdash e_1 \parallel e_2} [\text{Conc}] \quad \frac{e \vdash e_1 \quad e \vdash e_2}{e \vdash e_1 \oplus e_2} [\text{Xor}]$$
$$\frac{e \vdash e'}{e \vdash [e']_n^\ell} [\text{Proj}] \quad \frac{e \vdash e_1 \quad \vdash e_1 \doteq e_2}{e \vdash e_2} [\text{Conv}]$$
$$\frac{e \vdash e'}{e \vdash H(e')} [\text{H}] \quad \frac{e \vdash e'}{e \vdash f(e')} [\text{F}] \quad \boxed{\frac{e \vdash e'}{e \vdash f^{-1}(e')} [\text{FInv}]}$$

Convertibility

- ▶ Based on equational theory of bitstrings
- ▶ Decidable for probabilistic expressions without H, f, f^{-1}

Useful for

- ▶ Finding decryption algorithm and attacks
- ▶ Proof rules: symbolic entropy and symbolic reduction

Attack finding

- ▶ Apply correctness check
 - ☞ is decryption possible with a key? $r \parallel f(r)$
- ▶ Apply simple filters, eg
 - ☞ is decryption possible without a key? $m \parallel f(r)$
 - ☞ is encryption randomized? $f(m)$
 - ☞ is randomness extractable without a key? $r \parallel f(m \oplus r)$
- ▶ Apply static equivalence

Proof system for IND-CPA

$$\frac{m \notin c^*}{c^* : \text{Guess}} [\text{Indep}] \qquad \frac{c^* : \varphi \quad r \text{ fresh}}{(c^* : \varphi)\{e \oplus r/r\}} [\text{Opt}]$$

$$\frac{e \vdash \vec{r} \quad \vec{r} \cap \mathcal{R}(c^*) = \emptyset}{c^* : e \in L_H^A} [\text{Indom}]$$

$$\frac{e \vdash^A \vec{r} \quad f(\vec{r}) \parallel \vec{r}_0 \parallel m \vdash_{t'}^A c^* \quad \vec{r} \cap \vec{r}_0 = \emptyset}{c^* : e \in L_H^A} [\text{OW}]$$

$$\frac{c^* : \phi \quad c^* : e \in L_H^A \quad r \notin \mathcal{R}(e) \quad H \notin \mathcal{H}(c^*, \phi, e)}{c^*[H(e)/r] : \varphi[H(e)/r]} [\text{Fail}]$$

+ a few more rules that are seldom needed
(eg a variant of [Indom] to prove IND-CPA of OAEP under OW)

Chosen-ciphertext security

Principles

Extensionality Replace e or d by equivalent ones

Plaintext awareness Reject invalid ciphertexts

Plaintext extractor “Public” decryption oracle can be eliminated

Currently restricted to non-programmable random oracles.

Attack finding

- ▶ is encryption malleable? $f(r) || m \oplus G(r)$

Soundness

- ▶ Once and for all
 - + “global” guarantee
 - + avoids resorting to intermediate framework
- ▶ By generating a pRHL/EasyCrypt proof for each scheme
 - + limits Trusted Computing Base
 - + proofs can be combined and reused
 - currently restricted to IND-CPA

Systematic exploration

- ▶ Generate well-typed terms up to user-defined constraints
- ▶ Check for decryption algorithm and attacks
- ▶ Launch proof search (strategy + backtracking)
- ▶ Compile successful runs to EasyCrypt (for IND-CPA)
- ▶ Practical interpretation

Evaluation

- ▶ Precise (no spurious attack, few unknowns)
- ▶ Efficient (attacks and proofs found instantaneously)

Minimality in cryptography

- ▶ OAEP (1994):

$$f((m \| 0) \oplus G(r) \| r \oplus H((m \| 0) \oplus G(r)))$$

not that Optimal; needs redundancy

- ▶ SAEP (2001):

$$f(r \| (m \| 0) \oplus G(r))$$

tighter reduction; needs redundancy

- ▶ ZAEP (with David Pointcheval):

$$f(r \| m \oplus G(r))$$

tighter reduction, bit-optimal, redundancy-free

ZAEP

For every INDCCA adversary \mathcal{A} there exists an inverter \mathcal{I} st

$$\left| \Pr_{\text{IND-CCA}}[b = b'] - \frac{1}{2} \right| \leq \mathbf{Succ}_f^{\text{OW}}(\mathcal{I}) + \frac{q_D}{2^n}$$

Based on existence of two efficient algorithms:

- ▶ CIE: given $f(r, s_1), f(r, s_2)$ with $s_1 \neq s_2$, returns s_1, s_2 and r
- ▶ SIE: given $f(r, s)$ and r returns s

Algorithms exist for RSA with exponents 2 and 3

Provable security of executable code

Proof by reduction:

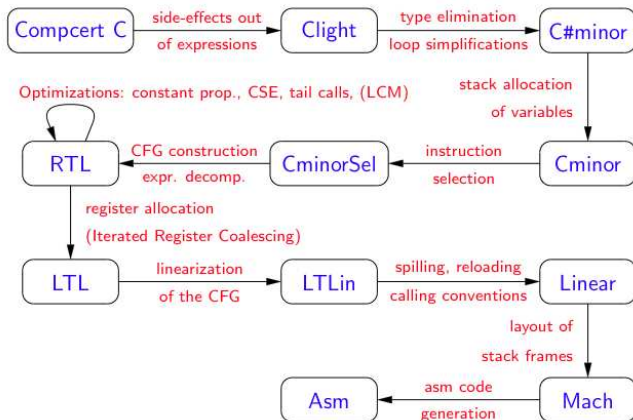
FOR ALL adversary that breaks the assembly code,
THERE EXISTS an adversary that breaks the source code

Proof relies on

- ▶ adversary model: low-level adversary does not observe computations (more than allowed in the source program)
- ▶ semantic preservation

CompCert (Leroy, 2006)

- ▶ Optimizing C compiler implemented in Coq
- ▶ Formal proof of semantic preservation



Semantic preservation in CompCert

Preservation of event traces

- ☞ system calls (“external calls”),
- ☞ I/O from and to the environment, and
- ☞ user-defined events

Issues with semantics preservation

Ideal (probabilistic) operations

- ▶ random sampling of bitstrings
- ▶ hash function (random oracle)

Libraries

- ▶ Implementations use arithmetic libraries

Approach

Proved semantics preservation using

- ▶ environment to model ideal operations
- ▶ trusted libraries to implement arithmetic

Side-channels and countermeasures

Well-known recipes for security disasters

- ▶ branching on secrets
- ▶ array accesses with secret indices

Want obliviousness

- ▶ control flow does not depend on secrets
- ▶ memory accesses do not depend on secrets

Problems

- ▶ algorithms might fail to satisfy obliviousness
- ▶ compilers might break both properties

Approach

- ▶ develop property-specific solutions
- ▶ formally verified using CompCert

Control flow obliviousness

- ▶ Prove security wrt “program counter model” game
 - ☞ add left/right tags to all branching statements
 - ☞ return traces to adversary
- ▶ Check compiler does not introduce new branches

Applied to implementation of OAEP based on LIP library

Memory obliviousness

- ▶ AES, DES. . . do not satisfy memory obliviousness
- ▶ Does compiler preserve memory obliviousness?

Solution

- ▶ Flow-sensitive information flow analysis of assembly code
- ▶ Stealth memory for accesses breaking obliviousness

Applied to implementations of AES

Case studies

- ▶ Encryption: OAEP, ZAEP, Hashed ElGamal, Cramer-Shoup, Boneh-Franklin IBE
- ▶ Signature: Full Domain Hash, BLS
- ▶ Hash functions:
 - ▶ Merkle-Damgard, Keccak
 - ▶ hashing into elliptic curves
- ▶ Zero knowledge protocols
- ▶ Authenticated Key Exchange
- ▶ Differential privacy

Conclusion

- ▶ Solid foundation for cryptographic proofs
- ▶ Formal verification of emblematic case studies
- ▶ Automated exploration of classes of constructions
- ▶ Discovery of new practical schemes
- ▶ Narrowing the gap between proofs and code

Cryptography is

- ▶ a thriving research area at the crossroads of many fields
- ▶ a great source of challenging problems
- ▶ an exciting opportunity to apply PL and PV techniques

<http://www.easycrypt.info>