Security Models
Lecture 2
Computational Security

Pascal Lafourcade

2020-2021

Outline

Reduction Proof Technique

How to prove that an encryption scheme E is secure ?

Reduction Proof Technique

How to prove that an encryption scheme E is secure ?

1. Hypothesis: HARD problem P (RSA, DL,DDH,CDH)
2. Reduction:
   - If an adversary A breaks the encryption scheme E
   - Then A can be used it to solve P in polynomial time.
3. Conclusion: Under this assumption there does not exist an adversary in polynomial time which can break the security of the scheme.

Reduction Proof Technique

How to prove that an encryption scheme E is secure ?

1. Hypothesis: HARD problem P (RSA, DL,DDH,CDH)
2. Reduction:
   - If an adversary A breaks the encryption scheme E
   - Then A can be used it to solve P in polynomial time.
3. Conclusion: Under this assumption there does not exist an adversary in polynomial time which can break the security of the scheme.

Application: ElGamal is IND-CPA secure under DDH assumption.

Consider an adversary breaking IND-CPA game for ElGamal then he can solve DDH

Definitions (recall)

$$\mathrm{Adv}^{DL}(\mathcal{A}) = Pr\left[\mathcal{A}(g^x) \rightarrow x \middle| x \xleftarrow{R} [1, q]\right]$$

$$\mathrm{Adv}^{CDH}(\mathcal{A}) = Pr\left[\mathcal{A}(g^x, g^y) \rightarrow g^{xy} \middle| x, y \xleftarrow{R} [1, q]\right]$$

$$\mathrm{Adv}^{DDH}(\mathcal{A}) = Pr\left[\mathcal{A}(g^x, g^y, g^{xy}) \rightarrow 1 \middle| x, y \xleftarrow{R} [1, q]\right]$$
$$-Pr\left[\mathcal{A}(g^x, g^y, g^r) \rightarrow 1 \middle| x, y, r \xleftarrow{R} [1, q]\right]$$

Proof of $CDH \leq DL$

We denote by $X = g^x, Y = g^y$

Proof of $CDH \leq DL$

We denote by $X = g^x, Y = g^y$

Consider there is an adversary A who breaks DL.

Proof of $CDH \leq DL$

We denote by $X = g^x, Y = g^y$

Consider there is an adversary A who breaks DL.
Using A for breaking DL, we get y

Proof of $CDH \leq DL$

We denote by $X = g^x, Y = g^y$

Consider there is an adversary A who breaks DL.
Using A for breaking DL, we get y
Hence $Z = g^{xy} = (g^x)^y = X^y$
We have solved CDH

Experiments

$Exp^{ddh-1}(A)$

$x, y \xleftarrow{R} [1, q]$
$returnA(g^x, g^y, g^{xy})$

$Exp^{ddh-0}(A)$

$x, y, z \xleftarrow{R} [1, q]$
$returnA(g^x, g^y, g^z)$

CDH implies DDH

CDH implies DDH

Let $\mathcal{A}$ be an adversary against the CDH assumption and $\mathcal{B}$ against DDH

Adversary $\mathcal{B}(X, Y, Z)$:
  if $Z = \mathcal{A}(X, Y)$ **then return** 1
  else return 0

CDH implies DDH

Let $\mathcal{A}$ be an adversary against the CDH assumption and $\mathcal{B}$ against DDH

Adversary $\mathcal{B}(X, Y, Z)$:
  if $Z = \mathcal{A}(X, Y)$ **then return** 1
  else return 0

$$\mathrm{Adv}^{DDH}(\mathcal{B}) = Pr[Exp^{DDH-1}(\mathcal{B}) = 1] - Pr[Exp^{DDH-0}(\mathcal{B}) = 1]$$

$$Pr\left[\mathcal{B}(g^x, g^y, g^{xy}) \to 1 \middle| x, y \xleftarrow{R} [1, q]\right] - Pr\left[\mathcal{B}(g^x, g^y, g^r) \to 1 \middle| x, y, r \xleftarrow{R} [1, q\right]$$

$$\mathrm{Adv}^{CDH}(\mathcal{A}) - \frac{1}{q}$$

The number of elements in G is supposed large hence $1/q$ is negligible. As the DDH assumption holds, the advantage of $\mathcal{B}$ is negligible. Hence the advantage of $\mathcal{A}$ against CDH is also negligible and the CDH assumption holds.

Example: RSA

public	private
$n = pq$	$d = e^{-1} \mod \phi(n)$
e (public key)	(private key)

RSA Encryption

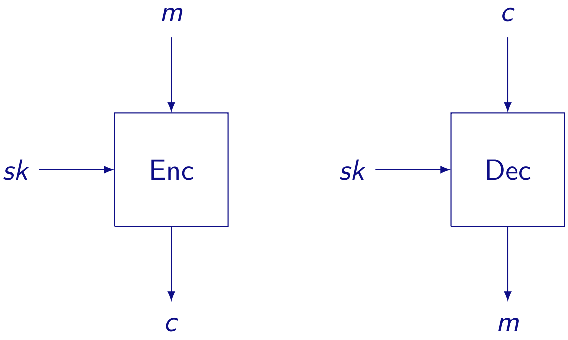
- $E(m) = m^e \mod n$
- $D(c) = c^d \mod n$

OW-CPA = RSA problem by definition!

Outline

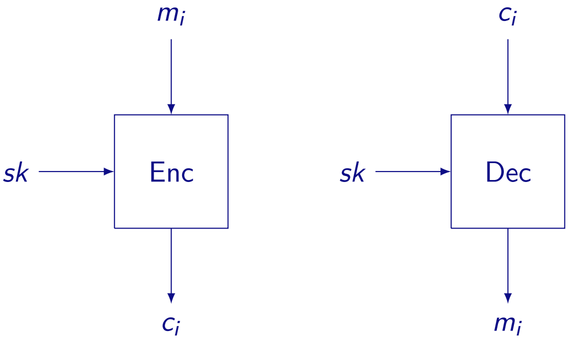
Chiffrement par bloc

- m, un message clair
- c, le chiffré de m
- $|m| = |c| = n$ bits



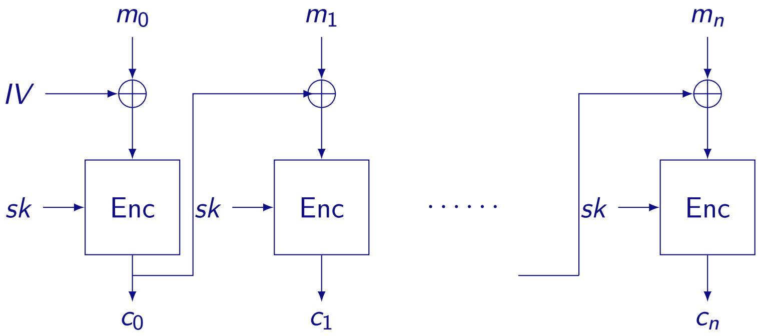
Mode ECB (*Electronic CodeBook*)

Let $|m| = k \cdot n,\ k > 1$.
$m = (m_1, \ldots, m_k)$, $|m_i| = n$ bits.



Mode CBC (*Cipher Block Chaining*)

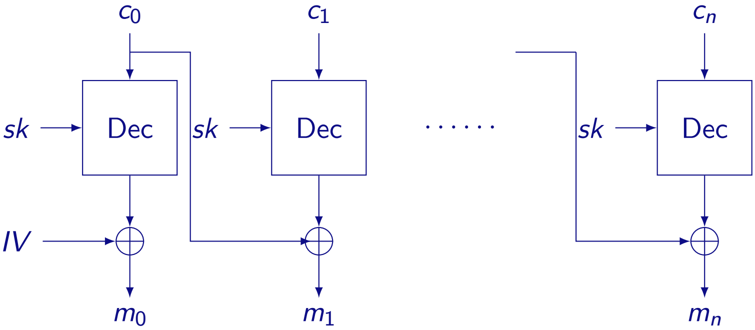
Encryption:



If the first block has index 1, $C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV$
$P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV$

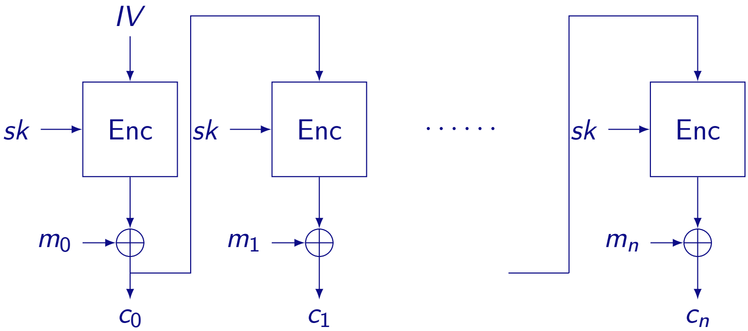
Mode CBC (*Cipher Block Chaining*)

Decryption:



Mode CFB (*Cipher FeedBack*)

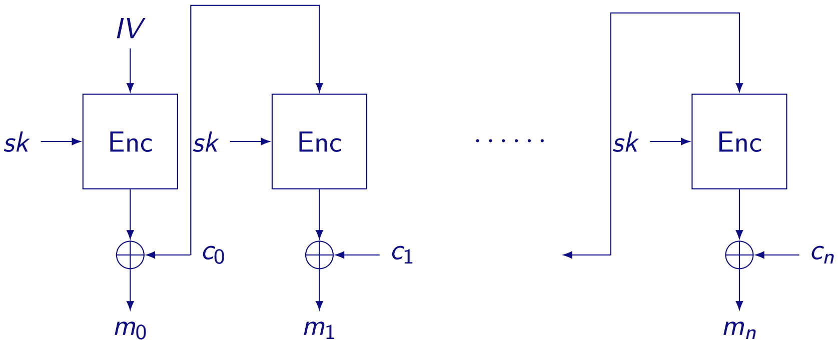
Encryption:



$$C_i = E_K(C_{i-1}) \oplus P_i$$
$$P_i = E_K(C_{i-1}) \oplus C_i$$
$$C_0 = \text{ IV}$$

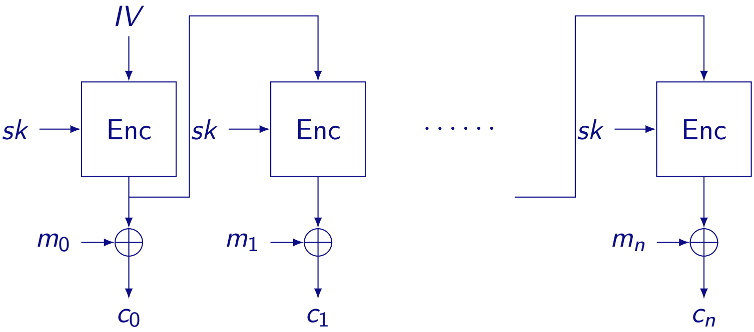
Mode CFB (*Cipher FeedBack*)

Decryption:



Mode OFB (*Output FeedBack*)

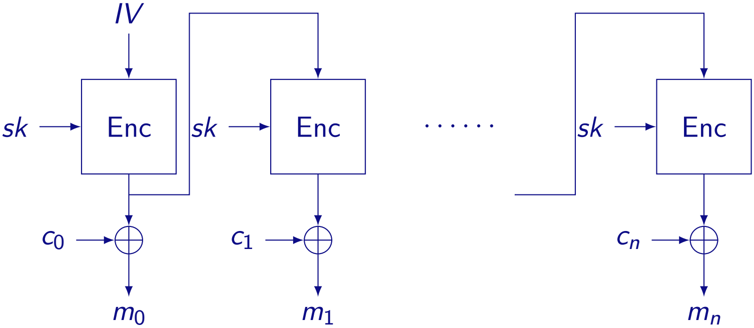
Encryption:



$$C_i = P_i \oplus O_i$$
$$P_i = C_i \oplus O_i$$
$$O_i = E_K(O_{i-1})$$
$$O_0 = \text{IV}$$

Mode OFB (*Output FeedBack*)

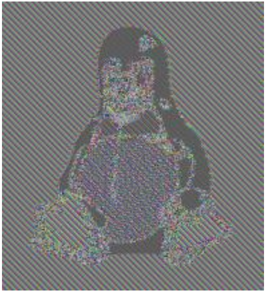
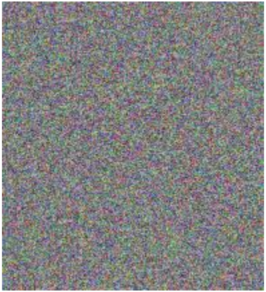
Decryption:



Counter Mode (CTR)

$$C_0 = IV$$
$$C_i = P_i \oplus \mathcal{E}_k(IV + i - 1)$$
$$P_i = C_i \oplus \mathcal{E}_k(IV + i - 1)$$

ECB vs Others

Outline

History

- Proposed in 1998 by Ronald Cramer and Victor Shoup
- First efficient scheme proven to be IND-CCA2 in standard model.
- Extension of Elgamal Cryptosystem.
- Use of a collision-resistant hash function

Ronald Cramer and Victor Shoup. "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack." in proceedings of Crypto 1998, LNCS 1462.

Key Generation

- G a cyclic group of order q with two distinct, random generators g_1, g_2
- Pick 5 random values (x_1, x_2, y_1, y_2, z) in $\{0, \ldots, q-1\}$
- $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, $h = g_1^z$
- Public key: (c, d, h), with G, q, g_1, g_2
- Secret key: (x_1, x_2, y_1, y_2, z)

Encryption of $m \in G$ with $(G, q, g_1, g_2, c, d, h)$

- Pick a random $k \in \{0, \ldots, q-1\}$
- Calculate: $u_1 = g_1^k$, $u_2 = g_2^k$
- $e = h^k m$
- $\alpha = H(u_1, u_2, e)$
- $v = c^k d^{k\alpha}$
- Ciphertext: (u_1, u_2, e, v)

Decryption of (u_1, u_2, e, v) with (x_1, x_2, y_1, y_2, z)

- Compute $\alpha = H(u_1, u_2, e)$
- Verify $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = v$
- $m = e/(u_1^z)$

It works because

$$u_1^z = g_1^{kz} = h^k$$

$$m = e/h^k$$

Outline

Key Privacy or Key Anonymity

Key Privacy or Key Anonymity

Key Privacy or Key Anonymity

SOLUTION

IKA-XXX Games




Given an encryption scheme $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. An adversary is a pair $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of polynomial-time probabilistic algorithms, $b \in \{0, 1\}$.

Let $IKA^{b}_{CPA}(\mathcal{A})$ be the following algorithm:

- $(pk_0, sk_0) \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$; $(pk_1, sk_1) \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$.
- $(s, m) \stackrel{R}{\leftarrow} \mathcal{A}_1(\eta, pk_0, pk_1)$
- Sample $b \stackrel{R}{\leftarrow} \{0, 1\}$.
- $b' \stackrel{R}{\leftarrow} \mathcal{A}_2(\eta, pk, s, \mathcal{E}(pk_b, m))$
- return b'.

$$\textsc{Adv}^{IKA_{XXX}}_{\mathcal{S},\mathcal{A}}(\eta) =$$

$$Pr[b' \stackrel{R}{\leftarrow} IKA^{1}_{XXX}(\mathcal{A}) : b' = 1] - Pr[b' \stackrel{R}{\leftarrow} IKA^{0}_{XXX}(\mathcal{A}) : b' = 1]$$

For CCA Adversary has access to the oracles D_{sk_0} and D_{sk_1}.

Example of Key-privacy or anonymity

- El Gamal and Cramer-Shoup are IKA secure under DDH assumption
- RSA trapdoor permutation is not anonymous
- variant of RSA-OAEP is IKA secure under assumption RSA is one-way

Reference : Key-Privacy in Public-Key Encryption by Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval

Outline

Digital Signature

Syntax: algorithms (KGen, Enc, Dec) such that:

- KGen(1^λ) : given security parameters, outputs tuple, (sk, pk) consisting of a private/public key
- $Sign(sk; m)$: given plaintext and secret key, outputs signature σ
- Vf(pk; m, σ) : given message, signature and public key, outputs a bit 1 if σ checks for m, 0 otherwise

Signature Security

Correctness:

- For all tuples (sk,pk) = KGen(1^λ) and for all messages $m \in M$, it must hold that Vf(pk; m, Sign(sk, m)) = 1
- Sometimes we degrade it to ϵ-correctness in which the verification of a signed message fails with probability ϵ

EUF-CMA:

Adversary can't forge fresh signature
$(sk, pk) = KGen(1^\lambda)$
$(m, \sigma) = A^{Sign(*)}(pk, 1^\lambda)$
Storelist $Q = \{(m_1, \sigma_1), \ldots, (m_k, \sigma_k)\}$ of queries to Sign
A wins iff $(m, *) \notin Q$ and $Vf(pk, m, \sigma) = 1$

RSA signature is Not EUF-CMA

Recall

$pk = (n, e)$ and $sk = d$
$\sigma = m^d \mod n$
verify : $m = \sigma^e \mod n$

Attack 1 : Pick a random string s compute $m' = s^e \mod N$ outputs (m', s) as forgery.
Attack2 : goal forge a signature for a given message m
Pick m_1 randomly, ask $\sigma_1 = m_1^d \mod n$
Compute m_2 such that $m_1 m_2 = m \mod N$, and ask $\sigma_2 = m_2^d \mod n$
Output $(m, \sigma_1\sigma_2 \mod N)$

How to get EUF-CMA : Probabilistic Full-Domain-Hash RSA (PFRSA)

Sign: $\sigma = (r, s) = (r, y^d \mod n)$, where $y = H(r||m)$ and r random
Verification : $s^e = H(r||m)$

Outline

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

Birthday Paradox : Hash Function

Let an Hash function $H : D \to 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

- Try $1 : 1 - 0$

Birthday Paradox : Hash Function

Let an Hash function $H : D \to 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

- Try 1 : $1 - 0$
- Try 2 : $1 - 1/2^k$

Birthday Paradox : Hash Function

Let an Hash function $H : D \to 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(\mathit{at\ least\ 1\ collision}) = 1 - P(\mathit{no\ collision})$$

Probability of no collision

- Try 1 : $1 - 0$
- Try 2 : $1 - 1/2^k$
- Try 3 : $1 - 2/2^k$

Birthday Paradox : Hash Function

Let an Hash function $H : D \to 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(\textit{at least } 1 \textit{ collision}) = 1 - P(\textit{no collision})$$

Probability of no collision

- Try 1 : $1 - 0$
- Try 2 : $1 - 1/2^k$
- Try 3 : $1 - 2/2^k$
- $\vdots$
- Try q : $1 - (q-1)/2^k$

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

- Try 1 : $1 - 0$
- Try 2 : $1 - 1/2^k$
- Try 3 : $1 - 2/2^k$
- $\vdots$
- Try q : $1 - (q-1)/2^k$

$$P(no\ collision) = \prod_{i=1}^{i=q} (1 - i/2^k)$$

Birthday Paradox : Hash Function

$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$
Using $1 - x \approx e^{-x}$ we have

$$1 - e^{-\sum_{i=1}^{i=q}(1-i/2^k)} = 1 - e^{-q(q-1)/2^{k+1}}$$

If you want a probability of ϵ to have a collision

Need ot solve

$$\epsilon = 1 - e^{-q(q-1)/2^{k+1}}$$
$$q(q-1) = 2^{k+1} ln(1/(1-\epsilon)$$
$$k \approx sqrt(2^{k+1} ln(1/(1-\epsilon))$$

Examples

- $\epsilon = \frac{1}{2} \Rightarrow k \approx 1.177 sqrt(2^{k+1})$
- $\epsilon = \frac{3}{4} \Rightarrow k \approx 1.665 sqrt(2^{k+1})$
- $\epsilon = 0.9 \Rightarrow k \approx 2.146 sqrt(2^{k+1})$

Remark: if 2^{k+1} is 365 among $1.77 sqrt(365)\ approx 23$
So should be at least > 64 or even 80. > 128 or 160 to resist birthday attack.

Outline

Several Tools for computational proofs

- CryptoVerif
- Easycrypt
- F*

Example: Prove properties of primitives in EasyCrypt, and use them to prove protocols in CryptoVerif.

CryptoVerif by Bruno Blanchet (2006)

Automatic tool for the automatic reasoning about security protocols[1]

- Messages are bitstrings
- Cryptographic primitives are functions from bitstrings to bitstrings
- The adversary is a probabilistic Turing machine

Version 2.04, released on Nov. 30, 2020

[1] `http://cryptoverif.inria.fr/`

CryptoVerif

- generates proofs by sequences of games.
- proves secrecy, authentication, and indistinguishability properties.
- works for N sessions (polynomial in the security parameter), with an active adversary.
- gives a bound on the probability of an attack (exact security).
- has an automatic proof strategy and can also be manually guided.

Included

A generic method for specifying properties of cryptographic primitives:

- MACs (message authentication codes)
- symmetric encryption
- public-key encryption
- signatures
- hash functions,
- Diffie-Hellman key agreements
- ...

Workflow of CryptoVerif

Prepare the input file containing

- the specification of the protocol to study (initial game),
- the security assumptions on the cryptographic primitives,
- the security properties to prove.

Run CryptoVerif

CryptoVerif outputs

the sequence of games that leads to the proof, a succinct explanation of the transformations performed between games, an upper bound of the probability of success of an attack.

F* (2016)

A general-purpose functional programming language with effects aimed at program verification.
https://www.fstar-lang.org/
Semi-automated verification system
Interactive proof assistant based on dependent types
F* is programmed in F*, but not (yet) verified

Project Everest

verify and deploy new, efficient HTTPS stack

- miTLS*: Verified reference implementation of TLS (1.2 and 1.3)
- HACL*: High-Assurance Cryptographic Library
- Vale: Verified Assembly Language for Everest

EasyCrypt 2009

- A toolset for reasoning about relational properties of probabilistic computations with adversarial code.
- Views cryptographic proofs as relational verification of open parametric probabilistic programs

`https://www.easycrypt.info/trac/`

Outline

Today

1. Modes
2. Reduction RSA, Elgamal
3. Cramer Shoup
4. Key Privacy
5. Signature security
6. Birhtday Paradox
7. Tools

Thank you for your attention.

Questions ?