Crypto Engineering - verifying crypto primitives and security protocols

Cristian Ene

Grenoble Alpes University, Verimag

Master 2: 1st Semester 2016

Administrative information and contents

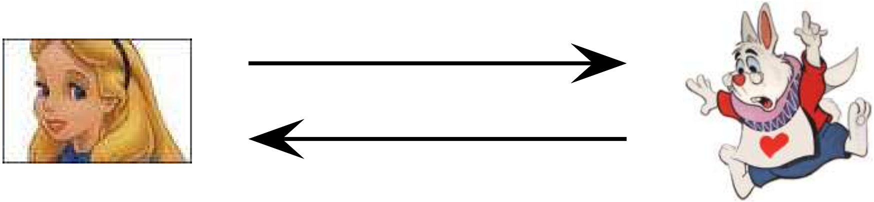
Administrative information:

- E-mail: Cristian.Ene@imag.fr
- Web page: `www-verimag.imag.fr/~ene/m2p/`

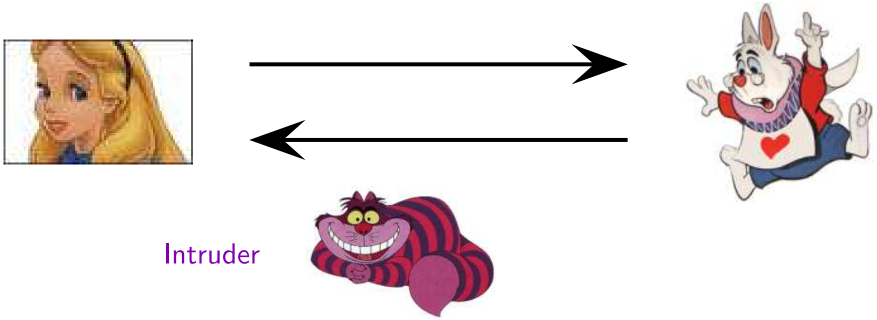
Schedule

1. Indistinguishability and Security Notions
2. Reduction Proofs
3. Hybrid Argument
4. Security of Protocols
5. Passive Intruder
6. Active Intruder
7. Link Between Computational and Symbolic

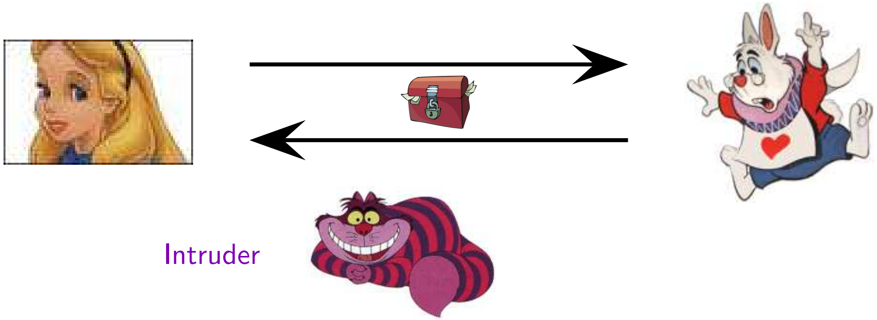
Description of Problem



Description of Problem



Description of Problem



Description of Problem

Intruder

Message cannot be understood by anyone else

Encryption

An encryption scheme $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

- $\mathcal{K}$: **key generation** - a probabilistic algorithm
- $\mathcal{E}$: **encryption** - in general, a probabilistic algorithm
- $\mathcal{D}$: **decryption** - a deterministic algorithm

$$\mathcal{K}(\eta) = (k_e, k_d)$$

$$\mathcal{E}_{k_e}(m, r) = c \qquad (\text{or } \mathcal{E}(k_e, m, r) = c)$$

$$\mathcal{D}(c, k_d) = m$$

Notations

Let $c = \mathcal{E}_{k_e}(m, r)$ and $D_{k_d}(c) = m$.

$k_e = k_d$ symmetric encryption

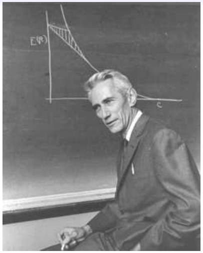
$k_e \neq k_d$ asymmetric (or public key) encryption

A unique m satisfies the relation (with possibly several r)

$\rightarrow$ For asymmetric encryption, given k_e, exhaustive search on m and r can lead to m !

$\Rightarrow$ unconditional secrecy for public key encryption is impossible !

Perfect Security (Shannon)



Definition

Let $m \in \mathcal{M}$ and $c \in E_k(m) \subseteq \mathcal{C}$. For any $m_0 \in \mathcal{M}$ and $c_0 \in \mathcal{C}$, an encryption system is called **perfectly secure** if :

$$Pr(m = m_0 | c = c_0) = Pr(m = m_0)$$

Probability (depends on keys and randomness) of guessing m remains unchanged after seeing the ciphertext c_0.

Exercise:

Messages are $\mathcal{M} = \{0, 1\}$, keys are $\mathcal{K} = \{k_0, k_1\}$ and possible ciphertexts are $\mathcal{C} = \{a, b\}$. Associated probabilities are:
$P(0) = 1/4$, $P(1) = 3/4$, $P(k_0) = 1/4$, $P(k_1) = 3/4$.
The encryption is defined by:

$$E_{k_0}(0) = a, E_{k_0}(1) = b, E_{k_1}(0) = b, E_{k_1}(1) = a.$$

Is this encryption perfectly secure?
If not, can you propose a perfect encryption on the same sets of messages, keys and ciphertexts?

Perfect encryption - One Time Pad (OTP)

Theorem

Let $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme such that $\mid \mathcal{M} \mid = \mid \mathcal{C} \mid = \mid \mathcal{K} \mid$. This scheme is perfect secure if and only if all keys have the same probability $\frac{1}{|\mathcal{K}|}$, and

$$\forall m \in \mathcal{M} \forall c \in \mathcal{C} \exists k \in \mathcal{K} \cdot \mathcal{E}_k(m) = c.$$

Vernam encryption

- $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^k$ for some k.
- $E_k(m) = m \oplus k$ ($\oplus$ is the xor-bitwise operation on strings).
- $D_k(c) = c \oplus k$.

Exercise: Prove that OTP is perfectly secure.

Perfect security vs. Computational security

Perfect security - a necessary condition

Let $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. If this scheme is perfect secure, then

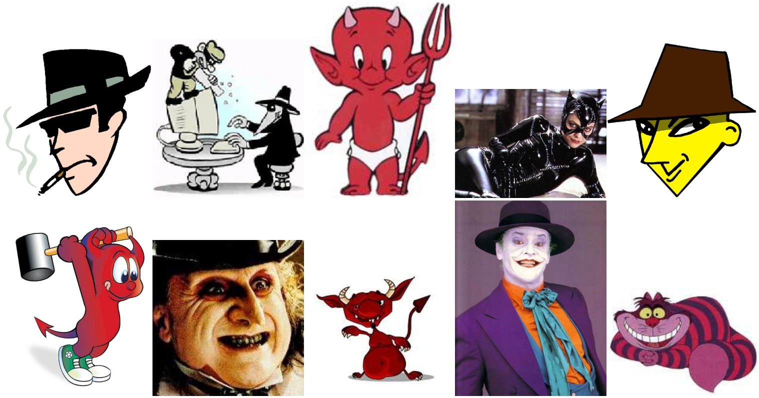
$$\mid \mathcal{M} \mid <= \mid \mathcal{K} \mid .$$

Conclusion: Perfect (unconditional) security is impractical, switch to **Computational security**!.

How to do it?

- Consider “Reasonable” Adversaries.
- Use “Hard” Problems.

Which adversary?



Adversary Model

Qualities of the adversary:

- Clever: Can perform all operations he wants.
- Lucky: Can guess keys/secrets.
- Limited time:
  - Do not consider attack in 2^{60}.
  - Otherwise a Brute force by enumeration is always possible.

Model used:

- **Any Turing Machine**: represents all possible algorithms.
- **Probabilistic**: adversary can guess/generates keys, random numbers...
- **Polynomial-time**.

One-Wayness (OW)

Put your message in a translucent bag, but you cannot read the text.

One-Wayness (OW)

Put your message in a translucent bag, but you cannot read the text.

Without the private key, it is computationally **impossible to recover the plain-text**.

Is it secure ?

Is it secure ?

Is it secure ?

- you cannot read the text but you can distinguish which one has been encrypted.

Is it secure ?

- you cannot read the text but you can distinguish which one has been encrypted.
- Does not exclude to recover half of the plain-text
- Even worse if one has already partial information of the message:
  - Subject: XXXX
  - From: XXXX

Indistinguishability (IND)

Put your message in a black bag, you can not read anything.

Now a black bag is of course IND and it implies OW.

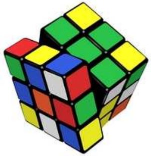
Indistinguishability (IND)

Put your message in a black bag, you can not read anything.

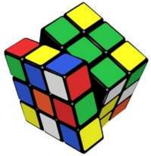
Now a black bag is of course IND and it implies OW.
The adversary is not able to **guess in polynomial-time even a bit of the plain-text knowing the cipher-text**, notion introduced by S. Goldwasser and S.Micali ([GM84]).

Is it secure?

Is it secure?



Is it secure?



- It is possible to scramble it in order to produce a new cipher. In more you know the relation between the two plain text because you know the moves you have done.

Non Malleability (NM)

Put your message in a black box.

But in a black box you cannot touch the cube (message), hence NM implies IND.

Non Malleability (NM)

Put your message in a black box.

But in a black box you cannot touch the cube (message), hence NM implies IND.
The adversary should **not be able to produce a new cipher-text** such that the plain-texts are meaningfully related, notion introduced by D. Dolev, C. Dwork and M. Naor in 1991 ([DDN91,BDPR98,BS99]).

Summary of Security Notions

Non Malleability

$\Downarrow$

Indistinguishability

$\Downarrow$

One-Wayness

Adversary Models

The adversary is given access to oracles :

$\rightarrow$ encryption of all messages of his choice
$\rightarrow$ decryption of all messages of his choice

Three classical security levels:

- Chosen-Plain-text Attacks (CPA)

- Non adaptive Chosen-Cipher-text Attacks (CCA1)
  only before the challenge

- Adaptive Chosen-Cipher-text Attacks (CCA2)
  unlimited access to the oracle (except for the challenge)

Chosen-Plain-text Attacks (CPA)

Adversary can obtain all cipher-texts from any plain-texts.
It is always the case with a Public Encryption scheme.

Non adaptive Chosen-Cipher-text Attacks (CCA1)

Adversary knows the public key, has access to a **decryption oracle multiple times before to get the challenge** (cipher-text), also called "Lunchtime Attack" introduced by M. Naor and M. Yung ([NY90]).

Adaptive Chosen-Cipher-text Attacks (CCA2)

Adversary knows the public key, has access to a **decryption oracle multiple times before and AFTER to get the challenge**, but of course cannot decrypt the challenge (cipher-text) introduced by C. Rackoff and D. Simon ([RS92]).

Summary of Adversaries

CCA2: $\mathcal{O}_1 = \mathcal{O}_2 = \{\mathcal{D}\}$ Adaptive Chosen Cipher text Attack

$\Downarrow$

CCA1: $\mathcal{O}_1 = \{\mathcal{D}\}$, $\mathcal{O}_2 = \emptyset$ Non-adaptive Chosen Cipher-text Attack

$\Downarrow$

CPA: $\mathcal{O}_1 = \mathcal{O}_2 = \emptyset$ Chosen Plain text Attack

Asymmetric Encryption

An asymmetric encryption scheme $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

- $\mathcal{K}$: **key generation** - a probabilistic algorithm
- $\mathcal{E}$: **encryption** - in general, a probabilistic algorithm
- $\mathcal{D}$: **decryption** - a deterministic algorithm

$$\mathcal{K}(\eta) = (k_e, k_d)$$

$$\mathcal{E}_{k_e}(m, r) = c \qquad (\text{or } \mathcal{E}(k_e, m, r) = c)$$

$$\mathcal{D}(c, k_d) = m$$

One-Wayness (OW)

Adversary $\mathcal{A}$: any probabilistic polynomial time Turing Machine (PPTM)

Basic security notion: One-Wayness (OW)

Without the private key, it is computationally impossible to recover the plain text:

$$\Pr_{m,r}[\mathcal{D}(c) = m \mid m \overset{R}{\leftarrow} \mathcal{A}(c)]$$

is negligible.

Indistinguishability (IND)

Game Adversary: $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

1. The adversary $\mathcal{A}_1$ is given the public key pk.
2. The adversary $\mathcal{A}_1$ chooses two messages m_0, m_1.
3. $b \overset{R}{\leftarrow} \{0,1\}$ is chosen at random and $c \overset{R}{\leftarrow} E(\text{pk}, m_b)$ is given to the adversary.
4. The adversary $\mathcal{A}_2$ answers b'.

The probability $Pr[b = b'] - \frac{1}{2}$ should be negligible.

The IND-CPA Games

Let $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme . An adversary is a pair $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of polynomial-time probabilistic algorithms. For any $b \in \{0,1\}$, let $\textsc{Ind}^b_{CPA}(\mathcal{A})$ be the following algorithm:

- Generate $(pk, sk) \overset{R}{\leftarrow} \mathcal{K}(\eta)$.
- $(s, m_0, m_1) \overset{R}{\leftarrow} \mathcal{A}_1(\eta, pk)$
- $b' \overset{R}{\leftarrow} \mathcal{A}_2(\eta, pk, s, \mathcal{E}(pk, m_b))$
- return b'.

Then, we define the advantage against the IND-CPA game by:

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

$$Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \textsc{Ind}^1_{CPA}(\mathcal{A})] - Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \textsc{Ind}^0_{CPA}(\mathcal{A})]$$

The IND-CCA1 Games

Let $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. An adversary is a pair $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of polynomial-time probabilistic algorithms.
For any $b \in \{0, 1\}$, let $\textsc{Ind}^b_{CCA1}(\mathcal{A})$ be the following algorithm:

- Generate $(pk, sk) \overset{R}{\leftarrow} \mathcal{K}(\eta)$.
- $(s, m_0, m_1) \overset{R}{\leftarrow} \mathcal{A}_1^{\mathcal{O}_1}(\eta, pk)$ **where** $\mathcal{O}_1 = \mathcal{D}$
- $b' \overset{R}{\leftarrow} \mathcal{A}_2(\eta, pk, s, \mathcal{E}(pk, m_b))$
- return b'.

Then, we define the advantage against the IND-CCA1 game by:

$$\textsc{Adv}^{\textsc{Ind}_{CCA1}}_{\mathcal{S},\mathcal{A}}(\eta) =$$

$$Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \textsc{Ind}^1_{CCA1}(\mathcal{A})] - Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \textsc{Ind}^0_{CCA1}(\mathcal{A})]$$

The IND-CCA2 Games

Let $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. An adversary is a pair $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of polynomial-time probabilistic algorithms.
For any $b \in \{0,1\}$, let $\text{IND}^b_{CCA2}(\mathcal{A})$ be the following algorithm:

- Generate $(pk, sk) \overset{R}{\leftarrow} \mathcal{K}(\eta)$.
- $(s, m_0, m_1) \overset{R}{\leftarrow} \mathcal{A}_1^{\mathcal{O}_1}(\eta, pk)$ where $\mathcal{O}_1 = \mathcal{D}$
- $b' \overset{R}{\leftarrow} \mathcal{A}_2^{\mathcal{O}_2}(\eta, pk, s, \mathcal{E}(pk, m_b))$ **where** $\mathcal{O}_2 = \mathcal{D}$
- return b'.

Then, we define the advantage against the IND-CCA2 game by:

$$\text{ADV}^{\text{IND}_{CCA2}}_{\mathcal{S},\mathcal{A}}(\eta) =$$

$$Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \text{IND}^1_{CCA2}(\mathcal{A})] - Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \text{IND}^0_{CCA2}(\mathcal{A})]$$

Summary

Given $\mathcal{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\text{IND}^b_{XXX}(\mathcal{A})$ follows:

- Generate $(pk, sk) \overset{R}{\leftarrow} \mathcal{K}(\eta)$.
- $(s, m_0, m_1) \overset{R}{\leftarrow} \mathcal{A}_1^{\mathcal{O}_1}(\eta, pk)$
- $b' \overset{R}{\leftarrow} \mathcal{A}_2^{\mathcal{O}_2}(\eta, pk, s, \mathcal{E}(pk, m_b))$
- return b'.

$$\text{ADV}^{\text{IND}_{XXX}}_{\mathcal{S},\mathcal{A}}(\eta) =$$

$$Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \text{IND}^1_{XXX}(\mathcal{A})] - Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \text{IND}^0_{XXX}(\mathcal{A})]$$

IND-CPA: $\mathcal{O}_1 = \mathcal{O}_2 = \emptyset$ Chosen Plain text Attack

IND-CCA1: $\mathcal{O}_1 = \{\mathcal{D}\}$, $\mathcal{O}_2 = \emptyset$ Non-adaptive Chosen Cipher text Attack

IND-CCA2: $\mathcal{O}_1 = \mathcal{O}_2 = \{\mathcal{D}\}$ Adaptive Chosen Cipher text Attack.

IND-XXX Security

Definition

An encryption scheme is *IND-XXX secure*, if for any adversary $\mathcal{A}$ the function $\textsc{Adv}^{IND-XXX}_{\mathcal{S},\mathcal{A}}$ is negligible.

Exercise

Prove that

$$\begin{aligned}\textsc{Adv}^{\textsc{Ind}XXX}_{\mathcal{S},\mathcal{A}}(\eta) &= Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \textsc{Ind}^1(\mathcal{A})] \\ &- Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \textsc{Ind}^0(\mathcal{A})] \\ &= 2Pr[b' = 1 \mid b' \overset{R}{\leftarrow} \textsc{Ind}^b(\mathcal{A})] - 1\end{aligned}$$

Definition of Non Malleability

Game Adversary: $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

1. The adversary $\mathcal{A}_1$ is given the public key pk.
2. The adversary $\mathcal{A}_1$ chooses a message space $\mathcal{M}$.
3. Two messages m and m^* are chosen at random in $\mathcal{M}$ and $c = E(\mathrm{pk}, m, r)$ is given to the adversary.
4. The adversary $\mathcal{A}_2$ outputs a binary relation R and a cipher-text c'.

Probability $Pr[R(m, m')] - Pr[R(m^*, m')]$ is negligible, where $m' = \mathcal{D}(c')$

Non-Malleability - XXX

- Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and $A = (A_1, A_2)$.
- For $b \in \{0, 1\}$ we define the experiment $\mathbf{Exp}^{atk-b}_{\mathcal{PE},A}(k)$:
  $(pk, sk) \leftarrow \mathcal{K}(k)$; $(\mathcal{M}, s) \leftarrow A_1^{O_1(.)}(pk)$; $x_0, x_1 \leftarrow \mathcal{M}$
  $y \leftarrow \mathcal{E}_{pk}(x_1)$; $(\mathcal{R}, \vec{y}) \leftarrow A_2^{O_2(.)}(\mathcal{M}, s, y)$; $\vec{x} \leftarrow \mathcal{D}_{pk}(\vec{y})$;
  If $y \notin \vec{y} \wedge \perp \notin \vec{x} \wedge \mathcal{R}(x_b, \vec{x})$ then $d \leftarrow 1$ else $d \leftarrow 0$
  Return d
- For $atk \in \{cpa, cca1, cca2\}$ and $k \in \mathbb{N}$, the advantage

$$\mathbf{Adv}^{atk}_{\mathcal{PE},A}(k) = Pr\left[\mathbf{Exp}^{atk-1}_{\mathcal{PE},A}(k) = 1\right] - Pr\left[\mathbf{Exp}^{atk-0}_{\mathcal{PE},A}(k) = 1\right]$$

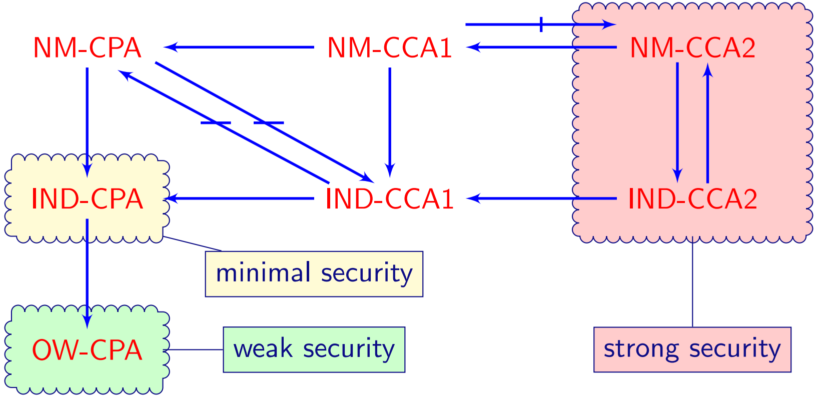
  has to be negligible for $\mathcal{PE}$ to be considered secure, assuming A, $\mathcal{M}$ and $\mathcal{R}$ can be computed in time $p(k)$.

Relations

$$\begin{tikzcd}
\text{NM-CPA} \arrow[d] \arrow[dr, shift left, "\mid" marking] & \text{NM-CCA1} \arrow[l] \arrow[d] \arrow[r, shift left, "\mid" marking] & \text{NM-CCA2} \arrow[l, shift left] \arrow[d, shift right] \\
\text{IND-CPA} \arrow[d] & \text{IND-CCA1} \arrow[l] \arrow[ul, shift left, "\mid" marking] & \text{IND-CCA2} \arrow[l] \arrow[u, shift right] \\
\text{OW-CPA} & &
\end{tikzcd}$$

"Relations Among Notions of Security for Public-Key Encryption Schemes", **Crypto'98**, by Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway [BDPR'98]

Relations



"Relations Among Notions of Security for Public-Key Encryption Schemes", **Crypto'98**, by Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway [BDPR'98]

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

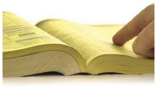
How to prove that an encryption scheme E is secure ?

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

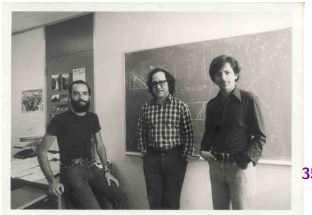
Integer Factoring and RSA



$\rightarrow$ Use of algorithmically hard problems.

Factorization

- $p, q \mapsto n = p.q$ easy (quadratic)
- $n = p.q \mapsto p, q$ difficult



Weak Factor

Experiment $w - Factor_{\mathcal{A}}(n)$

1. Chose randomly 2 integers x_1 and x_2 of size n
2. Compute $N = x_1 x_2$
3. $(x'_1, x'_2) \leftarrow \mathcal{A}(N)$
4. if $x'_1 x'_2 = N$ and $x'_i \neq 1$ then return 1 else return 0

For any probabilistic-polynomial algorithm $\mathcal{A}$,
$Pr[w - Factor_{\mathcal{A}}(n) = 1]$ is negligible (n is the security parameter).

Cyclic Group Definitions (I)

A **group** $(G, *)$ is composed of a set G and a binary operator $*$ on G which satisfy the three following axioms:

$\forall a, b, c \in G,\ a * (b * c) = (a * b) * c$	Associativity
$\exists e \in G,\ \forall a \in G,\ e * a = a * e = a$	Neutral Element
$\forall a \in G,\ \exists b \in G,\ a * b = b * a = e$	Inverse Element

b is called the inverse of a and is denoted by a^{-1}.

Example

$(\mathbb{Z}, +)$, $(\mathbb{Z}/n\mathbb{Z}, +)$, permutation group, $(\mathcal{M}_{(n,n)}, +)$.

Non-Examples

$(\mathbb{N}, +)$, $(\mathcal{M}_{(n,n)}, *)$

Definitions (II)

Cyclic Group

A group G is **cyclic** if G is finite and there exists an element g of G such that:

$$\forall a \in G,\ \exists n \in \mathbb{N},\ a = g^n$$

Element g is called a *generator* of group G.

Example

If p is a prime number, then $((\mathbb{Z}/p\mathbb{Z})^*, .)$ is a cyclic group.

The Discrete Logarithm (DL)

Let $G = (\langle g \rangle, *)$ be any finite cyclic group of prime order.

Idea: it is hard for any adversary to produce x if he only knows g^x.

The Discrete Logarithm (DL)

Let $G = (\langle g \rangle, *)$ be any finite cyclic group of prime order.

Idea: it is hard for any adversary to produce x if he only knows g^x.

Experiment $DLog_{\mathcal{A}}(n)$

1. Run $G(1^n)$ to get a cyclic group G of order q ($||q|| = n$) and g a generator.
2. Chose randomly x in $\mathbb{Z}_q$ and compute $h = g^x$
3. $x' \leftarrow \mathcal{A}(h, g, q)$
4. if $g^{x'} = h$ return 1 else 0

For any probabilistic-polynomial algorithm $\mathcal{A}$,

$$\mathbf{Adv}^{DLog_{\mathcal{A}}} = Pr[DLog_{\mathcal{A}}(n) = 1]$$

is negligible.

The Diffie-Hellman Key Exchange Protocol

Let g be a generator of a cyclic group of prime order q.

$$\begin{array}{rcl} A \rightarrow B & : & g^a \\ B \rightarrow A & : & g^b \\ A \rightarrow B & : & \{N\}_{g^{ab}} \end{array}$$

Computational Diffie-Hellman (CDH)

Idea: it is hard for any adversary to produce g^{xy} if he only knows g^x and g^y.

Computational Diffie-Hellman (CDH)

Idea: it is hard for any adversary to produce g^{xy} if he only knows g^x and g^y.

Experiment $CDH_{\mathcal{A}}(n)$

1. Run $G(1^n)$ to get a cyclic group G of order q ($||q|| = n$) and g a generator.
2. Chose randomly x, y in $\mathbb{Z}_q$ and compute g^x and g^y
3. $g^z \leftarrow \mathcal{A}(q, g, g^x, g^y)$
4. if $g^z = g^{xy}$ return 1 else 0

For any probabilistic-polynomial algorithm $\mathcal{A}$,

$$\mathbf{Adv}^{CDH_{\mathcal{A}}} = Pr[CDH_{\mathcal{A}}(n) = 1]$$

is negligible.

Decisional Diffie-Hellman (DDH)

Idea: Knowing g^x and g^y, it should be hard for any adversary to distinguish between g^{xy} and g^r for a random value r.

Decisional Diffie-Hellman (DDH)

Idea: Knowing g^x and g^y, it should be hard for any adversary to distinguish between g^{xy} and g^r for a random value r.

Experiment $DDH_{\mathcal{A}}(n)$

1. Run $G(1^n)$ to get a cyclic group G of order q ($||q|| = n$) and g a generator.
2. Chose randomly x, y, r in $\mathbb{Z}_q$ and compute g^x, g^y and g^r.
3. Chose randomly b in $\mathbb{Z}_2$. If $b = 1$ then $s = g^{xy}$ else $s = g^r$.
4. Give q, g, g^x, g^y, s to $\mathcal{A}$ and get the answer c (i.e. $c \leftarrow \mathcal{A}(q, g, g^x, g^y, s)$).
5. if $b = c$ return 1 else 0.

For any probabilistic-polynomial algorithm $\mathcal{A}$,

$$\mathbf{Adv}^{DDH_{\mathcal{A}}} = Pr[DDH_{\mathcal{A}}(n) = 1] - \frac{1}{2}$$

is negligible.

Decisional Diffie-Hellman (DDH)

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

$$-Pr\left[\mathcal{A}(q, g, g^x, g^y, g^r) \rightarrow 1 \middle| x, y, r \overset{R}{\leftarrow} [1, q]\right]$$

is negligible.
This means that an adversary cannot extract a single bit of information on g^{xy} from g^x and g^y.

Hard Problems

Most cryptographic constructions are based on *hard problems*.
Their security is proved by reduction to these problems:

- Discrete Logarithm problem, DL. Given a group $\langle g \rangle$ and g^x, compute x.
- Computational Diffie-Hellman, CDH Given a group $\langle g \rangle$, g^x and g^y, compute g^{xy}.
- Decisional Diffie-Hellman, DDH Given a group $\langle g \rangle$, distinguish between the distributions (g^x, g^y, g^{xy}) and (g^x, g^y, g^r).
- RSA. Given $N = pq$ and $e \in \mathbb{Z}^*_{\varphi(N)}$, compute the inverse of e modulo $\varphi(N) = (p-1)(q-1)$. Factorization

$$DDH \leq CDH \leq DL$$

Relation between the problems

Exercise

$DL \Rightarrow CDH \Rightarrow DDH$.

Prop (Maurer & Wolf 1998)

For generic groups, $DL \Leftrightarrow CDH$

Prop (T. Okamoto and D. Pointcheval 2001)

Gap DH
There are groups for which DDH *is easier than* CDH*.*

Usage of DH assumption

The Diffie-Hellman problems are widely used in cryptography:

- Public key Crypto-systems [ElGamal, Cramer& Shoup]
- Pseudo-random functions [Noar& Reingold, Canetti]
- Pseudo-random generators [Blum& Micali]
- (Group) key exchange protocols [many]

Notion of Indistinguishability

Objects are considered to be computationally equivalent if they cannot be differentiated by any efficient procedure.

Hence, two distributions are said to be computationally indistinguishable if no efficient procedure can tell them apart.

Example with Distributions

Given an efficient algorithm D, we consider the probability that D accepts a string taken from the first distribution, and the probability for the second distribution. If these two probabilities are close, we say that D does not distinguish the two distributions.

Concrete Example (I)

Consider that in Box 1 there are *n* blue numerated balls and in Box 2 there are *n* red numerated balls, with uniform distributions.

Alice picks one ball into one of the two boxes and says the number of the ball.

Where did Alice pick the ball?

$$|Pr[A(Box1|Number) = 1] - Pr[A(Box2|Number) = 1]|$$

is negligible.

Concrete Example (II)

Consider now that Alice has $1/2$ probability to pick ball number 1 between the red balls and $\frac{1}{2(n-1)}$ for the others $(2, \ldots, n)$.

Then we can build an adversary which has a non negligible advantage to know which Box the ball comes from.

Medical Issue

Consider two sets of patients following two indistinguishable distributions of probability. We give in similar conditions to the first set a new medicine and only water to the second set.

If the results are significant then the treatment is efficient, i.e., the probability of distribution for the results whit medicine is distinguishable from the fictive one.

Security parameter

In practice, we use keys, nonces, messages... they all have some length. Also, the algorithms we use, take some time to execute. So, the **security parameter**, denoted η, is an integer such that:

- the key/plaintext length in asymmetric encryption and signing can be η
- the key/block length in block ciphers/symmetric encryption can be η
- the key/tag length in MACs or output length in hash functions can be η
- the execution time of honest participants and of the intruder is **polynomial** in η

Negligible functions

We call a function $\mu : \mathbb{N} \to \mathbb{R}^+$ negligible if for every positive polynomial p there exists an N such that for all $n > N$

$$\mu(n) < \frac{1}{p(n)}$$

Properties

Let f and g be two negligible functions, then

1. $f + g$ is negligible.
2. For any $k > 0$, f^k is negligible.
3. For any polynomial p, $p \times f$ is negligible.

Exercise: Proofs

Probability Ensemble - Distribution

$X = \{X_i\}_{i \in I}$, is an **ensemble** (or a **family**) indexed by I a countable index set, where each X_i is a random variable.

Notations

- $X = \{X_\eta\}_{\eta \in \mathbb{N}}$ has each X_η ranging over strings of length $poly(\eta)$.
- $X = \{X_w\}_{w \in \{0,1\}^*}$ has each X_w ranging over string of length $poly(|w|)$.

A random variable X on some set $\mathcal{X}$ induces a probability distribution $\mathcal{D}$ on the set $\mathcal{X}$, given by the list of probabilities associated with each of its possible values. Hence we will also manipulate families of distributions $\{\mathcal{D}_\eta\}_{\eta \in \mathbb{N}}$.

Notation

Let $\mathcal{A}$ be a probabilistic algorithm, $\{X_\eta\}_{\eta\in\mathbb{N}}$ be a family of random variables, and $\{\mathcal{D}_\eta^X\}_{\eta\in\mathbb{N}}$ be the associated family of distributions.
We will use indifferently:
$[u \overset{R}{\leftarrow} X_\eta; v \overset{R}{\leftarrow} \mathcal{A}(\eta, u) : v]$ or $[u \overset{R}{\leftarrow} \mathcal{D}_\eta^X; v \overset{R}{\leftarrow} \mathcal{A}(\eta, u) : v]$ or
$[v \overset{R}{\leftarrow} \mathcal{A}(\eta, X_\eta) : v]$ or $\mathcal{A}(\eta, X_\eta)$
for the distribution obtained by first sampling some value u using the random variable X_η, then starting $\mathcal{A}$ with u, getting some value v from $\mathcal{A}$ and outputting v.
We also use $\mathcal{A}(X)$ as a shorthand for the distribution family $\{\mathcal{A}(\eta, X_\eta)\}_{\eta\in\mathbb{N}}$.

Computational indistinguishability - definition

- Let $\mathcal{D}^0 = \{\mathcal{D}^0_\eta\}_{\eta\in\mathbb{N}}$ and $\mathcal{D}^1 = \{\mathcal{D}^1_\eta\}_{\eta\in\mathbb{N}}$ be two (families of) distributions. The advantage $Adv^{\mathcal{D}_0,\mathcal{D}_1}(\mathcal{A})$ of an adversary $\mathcal{A}$ is defined by

$$Adv^{\mathcal{D}^0,\mathcal{D}^1}(\mathcal{A}) = Pr[b' = 1 | d \leftarrow^R \mathcal{D}^0_\eta; b' \leftarrow^R \mathcal{A}(\eta, d)]$$
$$- Pr[b' = 1 | d \leftarrow^R \mathcal{D}^1_\eta; b' \leftarrow^R \mathcal{A}(\eta, d)]$$

  The distributions $\mathcal{D}^0$ and $\mathcal{D}^1$ are **computationally indistinguishable** (denoted $\mathcal{D}^0 \approx \mathcal{D}^1$), if the advantage $Adv^{\mathcal{D}_0,\mathcal{D}_1}(\mathcal{A})$ is negligible for any ppt-algorithm $\mathcal{A}$.
- $f : \mathbb{N} \to \mathbb{R}$ is **negligible** if $\forall n.\exists N_n.\forall \eta \geq N_n.f(\eta) \leq \frac{1}{\eta^n}$ (for all polynomials p, $lim_{\eta\to\infty} f(\eta) \times p(\eta) = 0$)

Computational indistinguishability - exemples

- statistical indistinguishability implies computational indistinguishability

$$[d \xleftarrow{R} \{0,1\}^\eta : d] \approx [d \xleftarrow{R} \{0,1\}^\eta;\ \mathit{if}\ d = 0^\eta\ \mathit{then}\ d = 1^\eta : d]$$

- Decisional Diffie Hellman Problem is (assumed) difficult. Let G a cyclic group of order q and generator g (g and q deppend on η). Then

$$[x, y \xleftarrow{R} \mathbb{Z}_q : (g^x, g^y, g^{xy})] \approx [x, y, z \xleftarrow{R} \mathbb{Z}_q : (g^x, g^y, g^z)]$$

We can prove that ElGamal encryption is IND-CPA secure, assuming that the Decisional Diffie Hellman Problem is hard.

Computational indistinguishability : transitivity

Theorem

If $\mathcal{D}^0 \approx \mathcal{D}^1$ and $\mathcal{D}^1 \approx \mathcal{D}^2$ then $\mathcal{D}^0 \approx \mathcal{D}^2$.

Proof

- Suppose that $\mathcal{D}^0 \not\approx \mathcal{D}^2$, and let $\mathcal{A}$ be a ppt-adversary that can distinguish $\mathcal{D}^0$ and $\mathcal{D}^2$ with non-negligible advantage.
- For $i \in \{0, 1, 2\}$, let $p^i_\eta = Pr[b' = 1 | d \leftarrow^R \mathcal{D}^i_\eta; b' \leftarrow^R \mathcal{A}(\eta, d)]$.
- There is a polynomial q, such that for infinitely many η, $p^0_\eta - p^2_\eta \geq 1/q(\eta)$.
- Then, either $p^0_\eta - p^1_\eta \geq 1/(q(\eta)/2)$ or $p^1_\eta - p^2_\eta \geq 1/(q(\eta)/2)$ for infinitely many η.
- Hence $\mathcal{A}$ distinguishes either $\mathcal{D}^0$ and $\mathcal{D}^1$, or $\mathcal{D}^1$ and $\mathcal{D}^2$ with non-negligible advantage.

Computational indistinguishability : proof by reduction

Definition

A family of distributions $\mathcal{E}$ is **polynomial-time constructible**, if there is a ppt-algorithm $\Psi_{\mathcal{E}}$, such that the output of $\Psi_{\mathcal{E}}(\eta)$ is distributed identically to $\mathcal{E}_\eta$.

Definition

Given two families of distributions $\mathcal{D}$ and $\mathcal{E}$, we define $\mathcal{D}\|\mathcal{E}$ by

$$(\mathcal{D}\|\mathcal{E})_\eta = [x \leftarrow^R \mathcal{D}_\eta; y \leftarrow^R \mathcal{E}_\eta :\ (x, y)]$$

Theorem

If $\mathcal{D}^0 \approx \mathcal{D}^1$ and $\mathcal{E}$ is polynomial-time constructible, then $(\mathcal{D}^0\|\mathcal{E}) \approx (\mathcal{D}^1\|\mathcal{E})$.

Computational indistinguishability : proof by reduction

Theorem

If $\mathcal{D}^0 \approx \mathcal{D}^1$ and $\mathcal{E}$ is polynomial-time constructible, then $(\mathcal{D}^0 \| \mathcal{E}) \approx (\mathcal{D}^1 \| \mathcal{E})$.

Proof

- Suppose that $(\mathcal{D}^0 \| \mathcal{E}) \not\approx (\mathcal{D}^1 \| \mathcal{E})$, and let $\mathcal{A}$ be a ppt-adversary that can distinguish $(\mathcal{D}^0 \| \mathcal{E})$ and $(\mathcal{D}^1 \| \mathcal{E})$ with non-negligible advantage.
- Define an adversary $\mathcal{B}$ by

$$\mathcal{B}(\eta, x) = [y \xleftarrow{R} \Psi_{\mathcal{E}}(\eta); b' \xleftarrow{R} \mathcal{A}(\eta, (x, y)) : \ b']$$

- We can see that if x is distributed according to $\mathcal{D}^i_\eta$, then the argument of $\mathcal{A}$ is distributed according to $(\mathcal{D}^i \| \mathcal{E})_\eta$.
- Hence the advantage of $\mathcal{B}$ is equal to the advantage of $\mathcal{A}$.

Indistinguishability by Repeated Sampling

Repeated sampling

Let $\mathcal{D}$ be a family of distributions, and p be a positive polynomial. We define the family of distributions $\overrightarrow{\mathcal{D}}$ by

$$\overrightarrow{\mathcal{D}}_\eta = [x_1 \leftarrow^R \mathcal{D}_\eta; \ldots; x_{p(\eta)} \leftarrow^R \mathcal{D}_\eta : \ (x_1, \ldots, x_{p(\eta)})]$$

To sample $\overrightarrow{\mathcal{D}}_\eta$, sample $\mathcal{D}_\eta$ $p(\eta)$ times, and construct the tuple of sampled values.

Two families ${\mathcal{D}^0}_\eta$ and ${\mathcal{D}^1}_\eta$ are said **indistinguishable by polynomial-time sampling** if $\overrightarrow{\mathcal{D}^0}$ and $\overrightarrow{\mathcal{D}^1}$ are indistinguishable.

Theorems

- If $\overrightarrow{\mathcal{D}^0} \approx \overrightarrow{\mathcal{D}^1}$ then $\mathcal{D}^0 \approx \mathcal{D}^1$.
- If $\mathcal{D}^0 \approx \mathcal{D}^1$ and $\mathcal{D}^i$ are polynomial-time constructible, then $\overrightarrow{\mathcal{D}^0} \approx \overrightarrow{\mathcal{D}^1}$.

Computational indistinguishability : from many samplings to one

If $\overrightarrow{\mathcal{D}^0} \approx \overrightarrow{\mathcal{D}^1}$ then $\mathcal{D}^0 \approx \mathcal{D}^1$.

Theorem.

Proof.

- Suppose that $\mathcal{D}^0 \not\approx \mathcal{D}^1$, and let $\mathcal{A}$ be a ppt-adversary that can distinguish $\mathcal{D}^0$ and $\mathcal{D}^1$ with non-negligible advantage.
- Define an adversary $\mathcal{B}$ by

$$\mathcal{B}(\eta, (x_1, \ldots, x_{p(\eta)})) = [b' \xleftarrow{R} \mathcal{A}(\eta, x_1) : \ b']$$

- We can see that if $(x_1, \ldots, x_{p(\eta)})$ is distributed according to $\overrightarrow{\mathcal{D}^i}_\eta$, then the argument x_1 of $\mathcal{A}$ is distributed according to $\mathcal{D}^i_\eta$.
- Hence the advantage of $\mathcal{B}$ is equal to the advantage of $\mathcal{A}$.

□

Hybrid Argument: A digest

- Extreme hybrids collide with the complex ensembles
- Neighboring hybrids are easily related to the basic ensembles
- Number of hybrid is “small” (polynomial)

From one sampling to many : hybrid argument 1

Theorem. If $\mathcal{D}^0 \approx \mathcal{D}^1$ and $\mathcal{D}^i$ are polynomial-time constructible, then $\overrightarrow{\mathcal{D}^0} \approx \overrightarrow{\mathcal{D}^1}$.

Proof.

- Suppose that $\overrightarrow{\mathcal{D}^0} \not\approx \overrightarrow{\mathcal{D}^1}$, and let $\mathcal{A}$ be a ppt-adversary that can distinguish $\overrightarrow{\mathcal{D}^0}$ and $\overrightarrow{\mathcal{D}^1}$ with non-negligible advantage.
- Hence for some polynomial q, and for infinitely many η,
  $Pr[\mathcal{A}(\eta, x) = 1 | x \leftarrow^R \overrightarrow{\mathcal{D}^0}_\eta] - Pr[\mathcal{A}(\eta, x) = 1 | x \leftarrow^R \overrightarrow{\mathcal{D}^1}_\eta] \geq 1/q(\eta)$.
- Assume for now that p is a constant.
- For $k \in \{0, \dots, p\}$, define $\overrightarrow{\mathcal{E}^k}_\eta$ by
  $\overrightarrow{\mathcal{E}^k}_\eta = [x_1 \leftarrow^R \mathcal{D}^0_\eta; \dots; x_k \leftarrow^R \mathcal{D}^0_\eta; x_{k+1} \leftarrow^R \mathcal{D}^1_\eta; \dots; x_p \leftarrow^R \mathcal{D}^1_\eta : (x_1, \dots, x_p)]$
- Thus $\overrightarrow{\mathcal{E}^0}_\eta = \overrightarrow{\mathcal{D}^1}$ and $\overrightarrow{\mathcal{E}^p}_\eta = \overrightarrow{\mathcal{D}^0}$. Define $P^k_\eta = Pr[\mathcal{A}(\eta, x) = 1 | x \leftarrow^R \overrightarrow{\mathcal{E}^k}_\eta]$.
- Then for infinitely many η, $1/q(\eta) \leq P^p_\eta - P^0_\eta = \sum_{i=1}^{p} (P^i_\eta - P^{i-1}_\eta)$.
- Then for some j and infinitely many η, $(P^j_\eta - P^{j-1}_\eta)] \geq 1/(p \times q(\eta))$.
- Then for some j, $\overrightarrow{\mathcal{E}^{j-1}}_\eta \not\approx \overrightarrow{\mathcal{E}^j}_\eta$, and $\mathcal{A}$ can distinguish them.

□

From one sampling to many : hybrid argument 2

Proof.

(Continuation 1)

- There is some j, such that $\overrightarrow{\mathcal{E}^{j-1}}_\eta \not\approx \overrightarrow{\mathcal{E}^{j}}_\eta$, and $\mathcal{A}$ can distinguish them.
- Now if $\mathcal{A}$ can distinguish $\overrightarrow{\mathcal{E}^{j-1}}_\eta$ and $\overrightarrow{\mathcal{E}^{j}}_\eta$, how can we construct $\mathcal{B}$ that can distinguish $\overrightarrow{\mathcal{D}^{0}} = \overrightarrow{\mathcal{E}^{p}}_\eta$ and $\overrightarrow{\mathcal{D}^{1}} = \overrightarrow{\mathcal{E}^{0}}_\eta$.
- Define $\mathcal{B}$ as follows. On input (η, x):

  1. Let $x_1 \xleftarrow{R} \mathcal{D}^0_\eta; \ldots ; x_{j-1} \xleftarrow{R} \mathcal{D}^0_\eta$.
  2. Let $x_j := x$.
  3. Let $x_j \xleftarrow{R} \mathcal{D}^1_\eta; \ldots ; x_p \xleftarrow{R} \mathcal{D}^1_\eta$.
  4. Call $b' \xleftarrow{R} \mathcal{A}(\eta, (x_1, \ldots , x_p))$ and return b'.

- The advantage of this distinguisher is at least $1/(p \times q(\eta))$.
- Remark: the above prove was not constructive.

□

Being constructive

Proof.

(Continuation 2)

- Suppose that $\overrightarrow{\mathcal{D}^0} \not\approx \overrightarrow{\mathcal{D}^1}$, and let $\mathcal{A}$ be a ppt-adversary that can distinguish $\overrightarrow{\mathcal{D}^0}$ and $\overrightarrow{\mathcal{D}^1}$ with non-negligible advantage.
- Hence for some polynomial q, and for infinitely many η, $Pr[\mathcal{A}(\eta, x) = 1 | x \leftarrow^R \overrightarrow{\mathcal{D}^0}_\eta] - Pr[\mathcal{A}(\eta, x) = 1 | x \leftarrow^R \overrightarrow{\mathcal{D}^1}_\eta] \geq 1/q(\eta)$.
- Define $\mathcal{B}$ as follows. On input (η, x):

```
0. Let $j \leftarrow^R \{1, \ldots, p\}$.
1. Let $x_1 \leftarrow^R \mathcal{D}^0_\eta; \ldots ; x_{j-1} \leftarrow^R \mathcal{D}^0_\eta$.
2. Let $x_j := x$.
3. Let $x_j \leftarrow^R \mathcal{D}^1_\eta; \ldots ; x_p \leftarrow^R \mathcal{D}^1_\eta$.
4. Call $b' \leftarrow^R \mathcal{A}(\eta, (x_1, \ldots, x_p))$ and return $b'$.
```

- The advantage of this distinguisher is at least $1/(p \times q(\eta))$.

$\square$

When p depends on η

Proof.

(Continuation 3)

- Suppose that $\overrightarrow{\mathcal{D}^0} \not\approx \overrightarrow{\mathcal{D}^1}$, and let $\mathcal{A}$ be a ppt-adversary that can distinguish $\overrightarrow{\mathcal{D}^0}$ and $\overrightarrow{\mathcal{D}^1}$ with non-negligible advantage.
- Hence for some polynomial q, and for infinitely many η, $Pr[\mathcal{A}(\eta, x) = 1 | x \leftarrow^R \overrightarrow{\mathcal{D}^0}_\eta] - Pr[\mathcal{A}(\eta, x) = 1 | x \leftarrow^R \overrightarrow{\mathcal{D}^1}_\eta] \geq 1/q(\eta)$.
- Define $\mathcal{B}$ as follows. On input (η, x):

```
0. Let $j \leftarrow^R \{1, \ldots, p(\eta)\}$.
1. Let $x_1 \leftarrow^R \mathcal{D}^0_\eta; \ldots; x_{j-1} \leftarrow^R \mathcal{D}^0_\eta$.
2. Let $x_j := x$.
3. Let $x_j \leftarrow^R \mathcal{D}^1_\eta; \ldots; x_{p(\eta)} \leftarrow^R \mathcal{D}^1_\eta$.
4. Call $b' \leftarrow^R \mathcal{A}(\eta, (x_1, \ldots, x_{p(\eta)}))$ and return $b'$.
```

- The advantage of this distinguisher is at least $1/(p(\eta) \times q(\eta))$.

□

Pseudo-random Ensembles

Definition

The ensemble $X = \{X_n\}_{n \in N}$ is called pseudo random ensemble if there exists a uniform ensemble $U = \{U_{l(n)}\}_{n \in N}$ such that X and U are indistinguishable in polynomial time

Pseudo-random Generator

Definition

A pseudo-random generator is a deterministic polynomial-time algorithm G satisfying:

- Expansion: There exists a function $l : N \rightarrow N$ such that $l(n) > n$ for all $n \in N$ and $|G(s)| = l(|s|)$ for all $s \in \{0, 1\}^*$.
- Pseudo-randomness: The ensemble $\{G(U_n)\}_{n \in N}$ is pseudo-random.

l is called the expansion factor of G.

Increasing the Expansion Factor

Given a pseudo-random generator G_1 with expansion function $l_1(n) = n + 1$, we construct a PRG G with arbitrary polynomial expansion factor

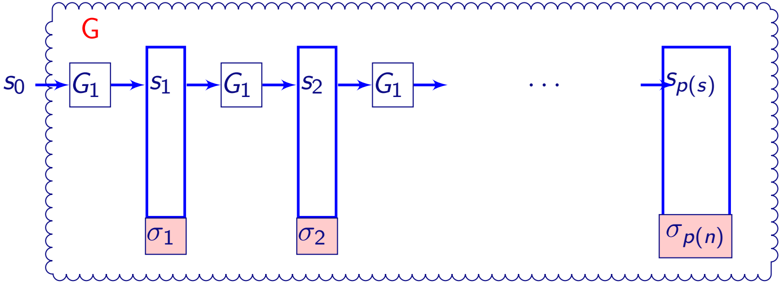
Construction

Let G_1 be a deterministic polynomial-time algorithm mapping strings of length n into strings of length $n + 1$, and let $p(.)$ be a polynomial. Define $G(s) = \sigma_1 \sigma_2 \ldots \sigma_{p(|s|)}$ where $s_0 = s$, the bit σ_i is the first bit of $G_1(s_{i-1})$, and s_i is the $|s|$-bit-long suffix of $G_1(s_{i-1})$ for every $1 \leq i \leq p(|s|)$.

Increasing the Expansion Factor

Algorithm: $G(s_0) = \sigma_1\sigma_2 \dots \sigma_{p(n)}$

```
Let $s_0 = s$ and $n = |s|$
For $i = 1$ to $p(n)$ do
    $\sigma_i s_i \leftarrow G_1(s_{i-1})$, where $\sigma_i \in \{0,1\}$ and $|s_i| = |s_{i-1}|$
Output $\sigma_1\sigma_2 \dots \sigma_{p(n)}$
```



Application of Hybrid Argument

Theorem

Let G_1,$p(.)$, and G defined as in previous construction such that $p(n) > n$. If G_1 is a PRG then G is also a PRG.

Proof uses an hybrid argument.
Intuitively, we can see that each application of G_1 can be replaced by a random process. The Indistinguishability of each applications of G_1 implies that polynomially many applications of G_1 are indinstiguishable from a random process.

Proof: Idea

To the contrary, suppose G is not a PRG then $\{G(U_n)\}_{n \in N}$ and $\{U_{p(n)}\}_{n \in N}$ are distiguisahble, i.e.

$$\Delta(n) = |Pr[D(G(U_n)) = 1] - Pr[D(U_{p(n)}])| > \frac{1}{q(n)}$$

It will contradict the fact that G_1 is PRG.

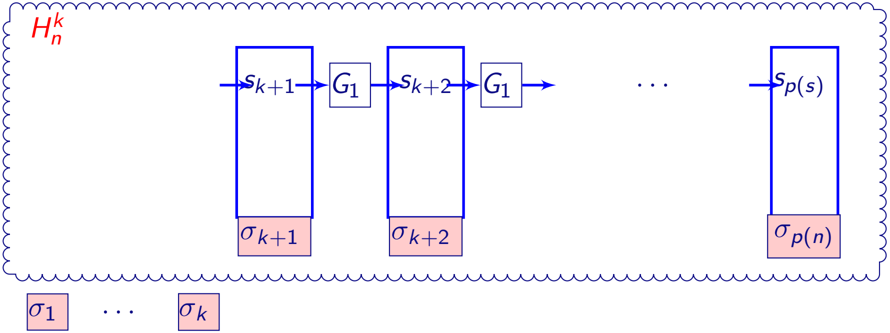
Hybrid Term

We define $\forall k, 0 \leq k \leq p(n)$

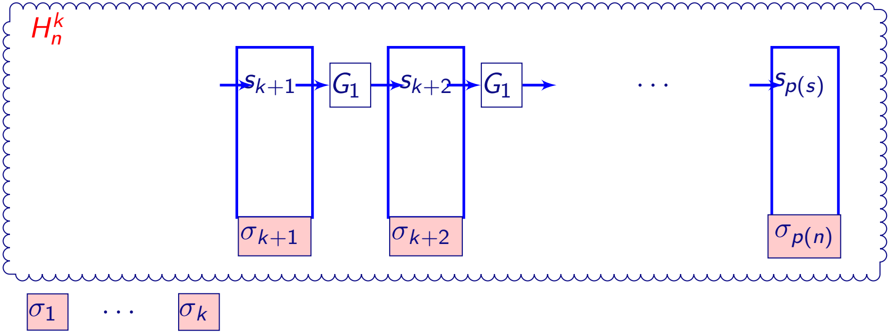
$$H_n^k = U_k^{(1)}.pref_{p(n)-k}(G(U_n^{(2)}))$$

where $U_k^{(1)}$ and $U_n^{(2)}$ are independent random variables

Proof: Other Representation of H_n^k



Proof: Other Representation of H_n^k



It is clear that:

- $H_n^0 = G(U_n)$
- $H_n^{p(n)} = U_{p(n)}$

Proof

Idea: If an algorithm D can distinguish extreme hybrid, it can do it for two neighboring hybrids.

By construction

$$pref_{j+1}(G(x)) = pref_1(G_1(x)).pref_j(G(suff_n(G_1(x))))$$

$$H_n^k = U_k^{(1)}.pref_{p(n)-k}(G(U_n^{(2)}))$$

$$H_n^{k+1} = U_{k+1}^{(1)}.pref_{p(n)-(k+1)}(G(U_n^{(2)}))$$

Notation:

$$f_{p(n)-k}(\alpha) = pref_1(\alpha).pref_{p(n)-k-1}(G(suff_n(\alpha)))$$

Claims

Two Easy Claims

- H_n^k is distributed identically to $U_k^{(1)}.f_{p(n)-k}(G_1(U_n^{(2)}))$
- H_n^{k+1} is distributed identically to $U_k^{(1)}.f_{p(n)-k}(U_{n+1}^{(2)})$

Proof of

Claim 1

- H_n^k is distributed identically to $U_k^{(1)}.f_{p(n)-k}(G_1(U_n^{(2)}))$

$$\begin{aligned} H_n^k &= U_k^{(1)}.pref_{(p(n)-k-1)+1}(G(U_n^{(2)})) \\ &= U_k^{(1)}.pref_1(G_1(U_n^{(2)})).pref_{p(n)-k-1}(G(suff_n(G_1(U_n^{(2)})))) \\ &= U_k^{(1)}.f_{p(n)-k}(G_1(U_n^{(2)})) \end{aligned}$$

Proof of

Claim 2

- H_n^{k+1} is distributed identically to $U_k^{(1)}.f_{p(n)-k}(U_{n+1}^{(2)})$

$$\begin{aligned}
H_n^{k+1} &= U_{k+1}^{(1)}.pref_{(p(n)-(k+1))}(G(U_n^{(2)})) \\
&= U_k^{(1')}.U_1^{(1'')}.pref_{(p(n)-k-1)}(G(suff_n(U_{n+1}^{(2')}))) \\
&= U_k^{(1')}.pref_1(U_{n+1}^{(2')}).pref_{(p(n)-k-1)}(G(suff_n(U_{n+1}^{(2')}))) \\
&= U_k^{(1')}.f_{p(n)-k}(U_{n+1}^{(2')})
\end{aligned}$$

Proof

We derive from D' an algorithm that distinguishes $G_1(U_n)$ from U_{n+1}.

Algorithm D'

Input $\alpha \in \{0,1\}^{n+1}$

1. D' picks uniformly an integer k in $\{0, 1, \ldots, p(n) - 1\}$
2. D' samples β uniformly in $\{0,1\}^k$
3. D' halts with output $D(\beta.f_{p(n)-k}(\alpha))$

Two Last Claims

Claims

$$Pr[D'(G_1(U_n)) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D(H_n^k) = 1]$$

$$Pr[D'(U_{n+1}) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D(H_n^{k+1}) = 1]$$

Two Last Claims

Claims

$$Pr[D'(G_1(U_n)) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D(H_n^k) = 1]$$

$$Pr[D'(U_{n+1}) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D(H_n^{k+1}) = 1]$$

Proof: By construction of D', we get for every $\alpha \in \{0,1\}^{n+1}$

$$Pr[D'(\alpha) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D(U_k.f_{p(n)-k}(\alpha)) = 1]$$

Last Part

$$\begin{aligned}
\delta(n) &= |Pr[D'(G_1(U_n)) = 1] - Pr[D'(U_{n+1}) = 1]| \\
&= \frac{1}{p(n)} | \sum_{k=0}^{p(n)-1} Pr[D(H_n^k) = 1] - \sum_{k=0}^{p(n)-1} Pr[D(H_n^{k+1}) = 1]| \\
&= \frac{1}{p(n)} |Pr[D(G(U_n)) = 1] - Pr[D(U_{p(n)}) = 1]| \\
&= \frac{\Delta(n)}{p(n)} > \frac{1}{q(n)p(n)}
\end{aligned}$$

Contradiction

Definitions (recall)

DL

$$\mathbf{Adv}^{DL}(\mathcal{A}) = Pr\Big[r = x \mid x \overset{R}{\leftarrow} \mathbb{Z}_q; r \overset{R}{\leftarrow} \mathcal{A}(g^x)\Big].$$

Assumption *DL*: $\mathbf{Adv}^{DL}(\mathcal{A})$ is negligible for any ppt-algorithm $\mathcal{A}$.

CDH

$$\mathbf{Adv}^{CDH}(\mathcal{A}) = Pr\Big[r = g^{xy} \mid x \overset{R}{\leftarrow} \mathbb{Z}_q; y \overset{R}{\leftarrow} \mathbb{Z}_q; r \overset{R}{\leftarrow} \mathcal{A}(g^x, g^y)\Big].$$

Assumption *CDH*: $\mathbf{Adv}^{CDH}(\mathcal{A})$ is negligible for any ppt-algorithm $\mathcal{A}$.

DDH

$$\mathbf{Adv}^{DDH}(\mathcal{A}) = Pr\Big[b' = 1 \mid x \overset{R}{\leftarrow} \mathbb{Z}_q; y \overset{R}{\leftarrow} \mathbb{Z}_q; b' \overset{R}{\leftarrow} \mathcal{A}(g^x, g^y, g^{xy})\Big]$$
$$- Pr\Big[b' = 1 \mid x \overset{R}{\leftarrow} \mathbb{Z}_q; y \overset{R}{\leftarrow} \mathbb{Z}_q; r \overset{R}{\leftarrow} \mathbb{Z}_q; b' \overset{R}{\leftarrow} \mathcal{A}(g^x, g^y, g^r)\Big].$$

Assumption *DDH*: $\mathbf{Adv}^{DDH}(\mathcal{A})$ is negligible for any ppt-algorithm $\mathcal{A}$.

Security parameter η is implicit in all these definitions.

Proof of $CDH \leq DL$

We denote by $X = g^x, Y = g^y$ the inputs.
We have to compute g^{xy}.

Proof of $CDH \leq DL$

We denote by $X = g^x, Y = g^y$ the inputs.
We have to compute g^{xy}.

Consider that there is an adversary $\mathcal{A}$ who breaks DL.

Proof of $CDH \leq DL$

We denote by $X = g^x, Y = g^y$ the inputs.
We have to compute g^{xy}.

Consider that there is an adversary $\mathcal{A}$ who breaks DL.
Using $\mathcal{A}$ with input Y, we get y.

Proof of $CDH \leq DL$

We denote by $X = g^x, Y = g^y$ the inputs.
We have to compute g^{xy}.

Consider that there is an adversary $\mathcal{A}$ who breaks DL.
Using $\mathcal{A}$ with input Y, we get y.
Then it is enough to compute X^y, since $X^y = (g^x)^y = g^{xy}$.
We have solved CDH.
Formally, we got an adversary $\mathcal{B}$ defined by

$$\mathcal{B}(\eta, X, Y) = y \overset{R}{\leftarrow} \mathcal{A}(\eta, Y);\ \mathit{return}\ X^y$$

such that

$$\mathbf{Adv}^{CDH}(\mathcal{B}) = \mathbf{Adv}^{DL}(\mathcal{A}).$$

Experiments

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

$x, y \overset{R}{\leftarrow} \mathbb{Z}_q; b' \overset{R}{\leftarrow} \mathcal{A}(g^x, g^y, g^{xy})$:
return b'.

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

$x, y, z \overset{R}{\leftarrow} \mathbb{Z}_q; b' \overset{R}{\leftarrow} \mathcal{A}(g^x, g^y, g^z)$:
return b'.

$$\mathbf{Adv}^{DDH}(\mathcal{A}) = Pr\left[b' = 1 \mid b' \overset{R}{\leftarrow} Exp^{ddh-1}(\mathcal{A})\right] - Pr\left[b' = 1 \mid b' \overset{R}{\leftarrow} Exp^{ddh-0}(\mathcal{A})\right]$$

DDH implies CDH

DDH implies CDH

Let $\mathcal{A}$ be an adversary against the CDH assumption. We build an adversary $\mathcal{B}$ against DDH:

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

DDH implies CDH

Let $\mathcal{A}$ be an adversary against the CDH assumption. We build an adversary $\mathcal{B}$ against DDH:

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

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

DDH implies CDH

Let $\mathcal{A}$ be an adversary against the CDH assumption. We build an adversary $\mathcal{B}$ against DDH:

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

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

$$Pr\left[\mathcal{B}(g^x, g^y, g^{xy}) \to 1 \middle| x, y \overset{R}{\leftarrow} \mathbb{Z}_q\right] - Pr\left[\mathcal{B}(g^x, g^y, g^r) \to 1 \middle| x, y, r \overset{R}{\leftarrow} \mathbb{Z}_q\right] =$$

DDH implies CDH

Let $\mathcal{A}$ be an adversary against the CDH assumption. We build an adversary $\mathcal{B}$ against DDH:

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

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

$$Pr\left[\mathcal{B}(g^x, g^y, g^{xy}) \to 1 \middle| x, y \overset{R}{\leftarrow} \mathbb{Z}_q\right] - Pr\left[\mathcal{B}(g^x, g^y, g^r) \to 1 \middle| x, y, r \overset{R}{\leftarrow} \mathbb{Z}_q\right] =$$

$$Pr\left[\mathcal{A}(g^x, g^y) \to g^{xy} \middle| x, y \overset{R}{\leftarrow} \mathbb{Z}_q\right] - Pr\left[\mathcal{A}(g^x, g^y) \to g^r \middle| x, y, r \overset{R}{\leftarrow} \mathbb{Z}_q\right] =$$

DDH implies CDH

Let $\mathcal{A}$ be an adversary against the CDH assumption. We build an adversary $\mathcal{B}$ against DDH:

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

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

$$Pr\left[\mathcal{B}(g^x, g^y, g^{xy}) \to 1 \middle| x, y \overset{R}{\leftarrow} \mathbb{Z}_q\right] - Pr\left[\mathcal{B}(g^x, g^y, g^r) \to 1 \middle| x, y, r \overset{R}{\leftarrow} \mathbb{Z}_q\right] =$$

$$Pr\left[\mathcal{A}(g^x, g^y) \to g^{xy} \middle| x, y \overset{R}{\leftarrow} \mathbb{Z}_q\right] - Pr\left[\mathcal{A}(g^x, g^y) \to g^r \middle| x, y, r \overset{R}{\leftarrow} \mathbb{Z}_q\right] =$$

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

The number of elements in G is supposed large, hence $1/q$ is negligible. If $\mathcal{A}$ would break CDH with non-negligible advantage, then $\mathcal{B}$ would break DDH with non-negligible advantage. Hence, if DDH assumption holds, then CDH assumption must hold.

Hardness assumptions (II)

Factorization

It is computational hard to factorize a large number.

RSA

public	private
$n = pq$	(p, q) (prime numbers)
$1 < e < \phi(n)$, e co-prime with $\phi(n)$	$d = e^{-1} \mod \phi(n)$

- RSA Encryption : $E(m) = m^e \mod n$
- RSA Decryption : $D(c) = c^d \mod n$

OW-CPA = RSA problem by definition!

RSA implies Factorization.

Example: ElGamal Encryption Scheme

ElGamal scheme

Key generation: Alice chooses a prime number p, a group generator g of $(\mathbb{Z}_p)$ and $a \in (\mathbb{Z}_{p-1})^*$.

Public key: (p, g, h), where $h = g^a \mod p$.

Private key: a.

Encryption of M: Bob chooses $r \in_R (\mathbb{Z}_{p-1})^*$ and computes $(u, v) = (g^r, Mh^r)$.

Decryption of (u, v): Alice computes $M \equiv_p v \div u^a$

Justification: $v \div u^a = Mh^r \div g^{ra} \equiv_p M$

Remarque: re-usage of the same random r leads to a security flaw:

$$M_1 h^r \div M_2 h^r \equiv_p M_1 \div M_2$$

Practical Inconvenience: Cipher is twice as long as plain text.

Example: ElGamal Encryption Scheme

Exercise

Prove that ElGamal is OW-CPA under CDH Assumption.

Example: ElGamal Encryption Scheme

Exercise

Prove that ElGamal is OW-CPA under CDH Assumption.

Exercise

Prove that ElGamal is IND-CPA under DDH Assumption.

Relations

$$\begin{tikzcd}
\text{NM-CPA} \arrow[d] \arrow[dr, shift left, "/" marking] & \text{NM-CCA1} \arrow[l] \arrow[d] \arrow[r, bend left, "|" marking] & \text{NM-CCA2} \arrow[l] \arrow[d, shift right] \\
\text{IND-CPA} \arrow[d] & \text{IND-CCA1} \arrow[l] \arrow[ul, shift left, "/" marking] & \text{IND-CCA2} \arrow[l] \arrow[u, shift right] \\
\text{OW-CPA} & &
\end{tikzcd}$$

"Relations Among Notions of Security for Public-Key Encryption Schemes", **Crypto'98**, by Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway [BDPR'98]

Non-Malleability - recall

- Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and $A = (A_1, A_2)$.
- For $b \in \{0, 1\}$ we define the experiment $\mathbf{Exp}^{atk-b}_{\mathcal{PE},A}(k)$:

  $(pk, sk) \leftarrow \mathcal{K}(k)$; $(\mathcal{M}, s) \leftarrow A_1^{O_1(.)}(pk)$; $x_0, x_1 \leftarrow \mathcal{M}$
  $y \leftarrow \mathcal{E}_{pk}(x_1)$; $(\mathcal{R}, \vec{y}) \leftarrow A_2^{O_2(.)}(\mathcal{M}, s, y)$; $\vec{x} \leftarrow \mathcal{D}_{pk}(\vec{y})$;
  If $y \notin \vec{y} \wedge \perp \notin \vec{x} \wedge \mathcal{R}(x_b, \vec{x})$ then $d \leftarrow 1$ else $d \leftarrow 0$
  Return d
- For $atk \in \{cpa, cca1, cca2\}$ and $k \in \mathbb{N}$, the advantage

  $$\mathbf{Adv}^{atk}_{\mathcal{PE},A}(k) = Pr\left[\mathbf{Exp}^{atk-1}_{\mathcal{PE},A}(k) = 1\right] - Pr\left[\mathbf{Exp}^{atk-0}_{\mathcal{PE},A}(k) = 1\right]$$

  has to be negligible for $\mathcal{PE}$ to be considered secure, assuming A, $\mathcal{M}$ and $\mathcal{R}$ can be computed in time $p(k)$.

Intuition for IND-CCA2 $\Rightarrow$ NM-CCA2

- Non-malleability deals with the ability to produce ciphertexts whose the plaintexts are related to the plaintext of a challenge ciphertext.

Intuition for IND-CCA2 ⇒ NM-CCA2

- Non-malleability deals with the ability to produce ciphertexts whose the plaintexts are related to the plaintext of a challenge ciphertext.
- Recall that in $CCA2$, the adversary is granted access to the decryption oracle during its whole attack, hence she can decrypt any ciphertext it obtains.

Intuition for IND-CCA2 $\Rightarrow$ NM-CCA2

- Non-malleability deals with the ability to produce ciphertexts whose the plaintexts are related to the plaintext of a challenge ciphertext.
- Recall that in $CCA2$, the adversary is granted access to the decryption oracle during its whole attack, hence she can decrypt any ciphertext it obtains.
- Now let suppose that the adversary is given an encryption of either m_0 or m_1 (the challenge). If the encryption would not be NM-CCA2, the adversary could get some ciphertexts $\vec{y}$ whose the plaintexts $\vec{x}$ are related in some manner $\mathcal{R}$ to the plaintext of the challenge ciphertext.

Intuition for IND-CCA2 $\Rightarrow$ NM-CCA2

- Non-malleability deals with the ability to produce ciphertexts whose the plaintexts are related to the plaintext of a challenge ciphertext.
- Recall that in *CCA*2, the adversary is granted access to the decryption oracle during its whole attack, hence she can decrypt any ciphertext it obtains.
- Now let suppose that the adversary is given an encryption of either m_0 or m_1 (the challenge). If the encryption would not be NM-CCA2, the adversary could get some ciphertexts $\vec{y}$ whose the plaintexts $\vec{x}$ are related in some manner $\mathcal{R}$ to the plaintext of the challenge ciphertext.
- Now the adversary can use her decryption oracle in order to get the plaintexts $\vec{x}$, and she can test by herself if m_0 (or m_1) is related by $\mathcal{R}$ to these plaintexts $\vec{x}$.

$+$ $\Rightarrow$ $+$

Main Idea

- We want to prove that if $\mathcal{PE}$ is IND-CCA2 secure, then it is also NM-CC2 secure
- Let $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ be an NM-CCA2 adversary attacking $\mathcal{PE}$.
- We build an IND-CCA2 adversary $\mathcal{A}$ attacking the same scheme, that uses $\mathcal{B}$ as a procedure.
- Then, we will prove that

$$\mathbf{Adv}^{NM-CCA2}_{\mathcal{PE},\mathcal{B}}(k) = 2 \cdot \mathbf{Adv}^{IND-CCA2}_{\mathcal{PE},\mathcal{A}}(k)$$

- If $\mathbf{Adv}^{NM-CCA2}_{\mathcal{PE},\mathcal{B}}(k)$ would be non-negligible, then $\mathbf{Adv}^{IND-CCA2}_{\mathcal{PE},\mathcal{A}}(k)$ would be non-negligible too, contradiction with the assumption that $\mathcal{PE}$ is IND-CCA2 secure!

Algorithm of Attack

Algorithm of Attack

Algorithm $\mathcal{A}_1^{\mathcal{D}_{sk}}(pk)$

```
$(s, \mathcal{M}) \overset{R}{\leftarrow} \mathcal{B}_1^{\mathcal{D}_{sk}}(pk)$; $(m_0, m_1) \overset{R}{\leftarrow} \mathcal{M}$; $s' \leftarrow (m_0, m_1, \mathcal{M}, s)$;
Return $(m_0, m_1, s')$.
```

Algorithm $\mathcal{A}_2^{\mathcal{D}_{sk}}(s', y)$

```
$(\mathcal{R}, \vec{y}) \overset{R}{\leftarrow} \mathcal{B}_2^{\mathcal{D}_{sk}}(\mathcal{M}, s, y)$; $\vec{x} \leftarrow \mathcal{D}_{sk}(\vec{y})$;
if $\mathcal{R}(m_0, \vec{x})$ then $d \leftarrow 0$ else $d \overset{R}{\leftarrow} \{0, 1\}$;
Return $d$.
```

Notation

$$\mathbf{Adv}^{IND-CCA2}_{\mathcal{PE},\mathcal{A}}(k) = pk(0) - pk(1)$$

where

$$p_k(b) = Pr\Big[b' = 0 \mid (pk, sk) \overset{R}{\leftarrow} \mathcal{K}(\eta); (s', m_0, m_1) \overset{R}{\leftarrow} \mathcal{A}_1^{\mathcal{D}_{sk}}(pk); \\ c \overset{R}{\leftarrow} \mathcal{E}(pk, m_b); b' \overset{R}{\leftarrow} \mathcal{A}_2^{\mathcal{D}_{sk}}(s', c)\Big]$$

$$\mathbf{Adv}^{NM-CCA2}_{\mathcal{PE},\mathcal{B}}(k) = pk'(0) - pk'(1)$$

where

$$p'_k(b) = Pr\Big[\mathcal{R}(m_0, \vec{x}) \mid (pk, sk) \overset{R}{\leftarrow} \mathcal{K}(\eta); (s, \mathcal{M}) \overset{R}{\leftarrow} \mathcal{B}_1^{\mathcal{D}_{sk}}(pk); (m_0, m_1) \overset{R}{\leftarrow} \mathcal{M}; \\ c \overset{R}{\leftarrow} \mathcal{E}(pk, m_b); (\mathcal{R}, \vec{y}) \overset{R}{\leftarrow} \mathcal{B}_2^{\mathcal{D}_{sk}}(\mathcal{M}, s, c); \vec{x} \leftarrow \mathcal{D}_{sk}(\vec{y})\Big]$$

End of the Proof

$$\begin{aligned}
p_k(0) &= p'_k(0) \times Pr[d = 0|\mathcal{R}(m_0, \vec{x})] + (1 - p'_k(0)) \times Pr[d = 0|d \overset{R}{\leftarrow} \{0,1\}] \\
&= p'_k(0) + \frac{1}{2} - \frac{1}{2} \cdot p'_k(0) \\
&= \frac{1}{2} \cdot (1 + p'_k(0)) \\
p_k(1) &= \frac{1}{2} \cdot (1 + p'_k(1))
\end{aligned}$$

$$\begin{aligned}
\mathbf{Adv}^{IND-CCA2}_{\mathcal{PE},\mathcal{A}}(k) &= pk(0) - pk(1) \\
&= \frac{1}{2} \cdot (1 + p'_k(0)) - \frac{1}{2} \cdot (1 + p'_k(1)) \\
&= \frac{1}{2} \cdot (p'_k(0) - p'_k(1)) \\
&= \frac{1}{2} \cdot \mathbf{Adv}^{NM-CCA2}_{\mathcal{PE},\mathcal{B}}(k)
\end{aligned}$$

Idea: IND-CCA1 $\not\Rightarrow$ NM-CPA

Assume there exists some IND-CCA1 secure encryption $P\mathcal{E}$.

Modify $P\mathcal{E}$ to build $P\mathcal{E}'$ which is also IND-CCA1 secure but not NM-CPA secure.

$+$

$\not\Rightarrow$

$+$

Algorithm $P\mathcal{E}'$:

Algorithm $P\mathcal{E}'$:

Notation : $y_1||y_2$ is the concatenation of y_1 and y_2, and $\overline{x}$ is the bitwise complement of x.

Algorithm $\mathcal{E}'_{pk}(x)$

```
$y_1 \overset{R}{\leftarrow} \mathcal{E}_{pk}(x)$; $y_2 \overset{R}{\leftarrow} \mathcal{E}_{pk}(\overline{x})$;
Return $y_1||y_2$.
```

Algorithm $\mathcal{D}'_{sk}(y_1||y_2)$

```
Return $\mathcal{D}_{sk}(y_1)$.
```

$P\mathcal{E}'$ is not NM-CPA: Idea

Given a cipher text $y_1||y_2$ of a message x it is easy to create a cipher of $\overline{x}$: just output $y_2||y_1$.Thus the scheme is malleable.

Formally: let $\mathcal{M}$ be the uniform distribution on $\{0,1\}^k$, and let $\mathcal{R}$ be the relation defined by $\mathcal{R}(m_1, m_2) = 1$ if and only if $m_1 = \overline{m_2}$

Let us consider the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where

$$\mathcal{A}_1(pk) = \text{ Return } (\emptyset, \mathcal{M}).$$

$$\mathcal{A}_2(\emptyset, \mathcal{M}, y_1||y_2) = \text{ Return } (\mathcal{R}, y_2||y_1).$$

Then

$$\mathbf{Adv}^{NM-CPA}_{\mathcal{PE}',\mathcal{A}}(k) = 1 - 2^{-k}$$

and hence $\mathcal{A}$ breaks $P\mathcal{E}'$ in the sense of NM-CPA.

$P\mathcal{E}'$ is IND-CCA1: Idea

Let $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ be some polynomial time adversary attacking $P\mathcal{E}'$ in the IND-CCA1 sense. Show that $\mathbf{Adv}^{IND-CCA1}_{\mathcal{PE}',\mathcal{B}}(k)$ is negligible, using an hybrid argument.

$$p_k(i,j) = Pr[b' = 1 \mid (s, m_0, m_1) \overset{R}{\leftarrow} \mathcal{B}_1^{\mathcal{D}_{sk}}; y_1 \overset{R}{\leftarrow} \mathcal{E}_{p_k}(m_i); y_2 \overset{R}{\leftarrow} \mathcal{E}_{p_k}(\overline{m_j});$$

$$b' \overset{R}{\leftarrow} \mathcal{B}_2(s, m_0, m_1, y_1||y_2)]$$

$$\mathbf{Adv}^{IND-CCA1}_{\mathcal{PE}',\mathcal{B}}(k) = p_k(1,1) - p_k(0,0)$$

$$\mathbf{Adv}^{IND-CCA1}_{\mathcal{PE}',\mathcal{B}}(k) = p_k(1,1) - p_k(1,0) + p_k(1,0) - p_k(0,0)$$

Claim 1: $p_k(1,1) - p_k(1,0)$ is negligible

Algorithm $\mathcal{A}_1^{\mathcal{D}_{sk}}(pk)$

```
$(s, m_0, m_1) \overset{R}{\leftarrow} \mathcal{B}_1^{\mathcal{D'}_{sk}}(pk)$;
Return $(\overline{m_0}, \overline{m_1}, s)$.
```

Algorithm $A_2(s, m_0, m_1, y)$

```
$y_1 \overset{R}{\leftarrow} \mathcal{E}_{pk}(\overline{m_1})$; $d \overset{R}{\leftarrow} \mathcal{B}_2(\overline{m_0}, \overline{m_1}, s, y_1||y)$;
Return $d$.
```

$$Pr[b' = 1 \mid (m_0, m_1, s) \overset{R}{\leftarrow} \mathcal{A}_1^{\mathcal{D}_{sk}}(pk); c \overset{R}{\leftarrow} \mathcal{E}_{pk}(m_1); b' \overset{R}{\leftarrow} \mathcal{A}_2(m_0, m_1, s, c)] = p_k(1,1)$$

$$Pr[b' = 1 \mid (m_0, m_1, s) \overset{R}{\leftarrow} \mathcal{A}_1^{\mathcal{D}_{sk}}(pk) : c \overset{R}{\leftarrow} \mathcal{E}_{pk}(m_0); b' \overset{R}{\leftarrow} \mathcal{A}_2(m_0, m_1, s, c)] = p_k(1,0)$$

$\mathbf{Adv}_{\mathcal{PE},\mathcal{A}}^{IND-CCA1}(k) = p_k(1,1) - p_k(0,0)$ is negligible, assuming security of $P\mathcal{E}$ in the IND-CCA1 sense.

Claim 2: $p_k(1,0) - p_k(0,0)$ is negligible

Algorithm $\mathcal{A}_1^{\mathcal{D}_{sk}}(pk)$

```
$(x_0, x_1, s) \overset{R}{\leftarrow} \mathcal{B}_1^{\mathcal{D}'_{sk}}(pk)$;
Return $(x_0, x_1, s)$.
```

Algorithm $\mathcal{A}_2(x_0, x_1, s, y)$

```
$y_0 \overset{R}{\leftarrow} \mathcal{E}_{pk}(\overline{x_0})$; $d \overset{R}{\leftarrow} \mathcal{B}_2(x_0, x_1, s, y || y_0)$;
Return $d$.
```

$$Pr[b' = 1 \mid (x_0, x_1, s) \overset{R}{\leftarrow} \mathcal{A}_1^{\mathcal{D}_{sk}}(pk); c \overset{R}{\leftarrow} \mathcal{E}_{pk}(x_1); b' \overset{R}{\leftarrow} \mathcal{A}_2(x_0, x_1, s, c)] = p_k(1,0)$$

$$Pr[b' = 1 \mid (x_0, x_1, s) \overset{R}{\leftarrow} \mathcal{A}_1^{\mathcal{D}_{sk}}(pk); c \overset{R}{\leftarrow} \mathcal{E}_{pk}(x_0); b' \overset{R}{\leftarrow} \mathcal{A}_2(x_0, x_1, s, c)] = p_k(0,0)$$

$\mathbf{Adv}_{\mathcal{PE},\mathcal{A}}^{IND-CCA1}(k) = p_k(1,0) - p_k(0,0)$ is negligible, assuming security of $P\mathcal{E}$ in the IND-CCA1 sense.

References

- Relations Among Notions of Security for Public-Key Encryption Schemes Crypto'98 by Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway
- The Foundations of Cryptography by Oded Goldreich
- Bellare Rogaway online book Introduction to modern cryptography
- On the Security Notions for Public-Key Encryption Schemes SCN '04, by Duong Hieu Phan and David Pointcheval.

Idea and Motivations

Idea

$A\mathcal{E} = (\mathcal{K}^a, \mathcal{E}^a, \mathcal{D}^a)$ an asymmetric encryption scheme (pk, sk).
$S\mathcal{E} = (\mathcal{K}^s, \mathcal{E}^s, \mathcal{D}^s)$ a symmetric encryption scheme K.

We define $\overline{\mathcal{E}_{pk}}(M)$ using $\mathcal{E}^s_K(M)$ and $\mathcal{E}^a_{pk}(K)$

Motivation

Costly operation (asymmetric encryption) is applied on a message of fixed size, and after efficient algorithm are used to encrypt the data.

Encryption Algorithm

```
**Algorithm** $\overline{\mathcal{E}_{pk}}(M)$
$K \leftarrow K^s$;
$C^a \leftarrow \mathcal{E}^a_{pk}(K)$;
$C^s \leftarrow \mathcal{E}^s_K(M)$;
$C \leftarrow (C^a, C^s)$;
Return $C$
```

Decryption Algorithm

```
**Algorithm** $\overline{\mathcal{D}_{sk}}(C)$
Parse $C$ as $(C^a, C^s)$;
$K \leftarrow \mathcal{D}^a_{sk}(C^a)$;
$M \leftarrow \mathcal{D}^s_K(C^s)$;
Return $M$
```

Security

Property

If $A\mathcal{E}$ and $S\mathcal{E}$ are each secure against chosen-plain-text attack, then $\overline{A\mathcal{E}}$ the hybrid encryption is also secure against chosen-plain-text attack.

Let B be an IND-CPA adversary attacking $\overline{A\mathcal{E}}$. Then there exist IND-CPA adversaries $A_{00,01}, A_{11,10}$ attacking $A\mathcal{E}$, and an adversary A attacking $S\mathcal{E}$, such that:

$$Adv_{\overline{A\mathcal{E}}}^{IND-CPA}(B) \leq Adv_{A\mathcal{E}}^{IND-CPA}(A_{00,01}) + Adv_{A\mathcal{E}}^{IND-CPA}(A_{11,10})$$

$$+Adv_{S\mathcal{E}}^{IND-CPA}(A)$$

Idea of the Proof

$$P(\alpha, \beta) = Pr[Exp_{A\mathcal{E}}^{\alpha\beta}(B) = 1]$$

$$P(1, 0) = Pr[Exp_{A\mathcal{E}}^{IND-CPA-1}(B) = 1]$$

$$P(0, 0) = Pr[Exp_{A\mathcal{E}}^{IND-CPA-0}(B) = 1]$$

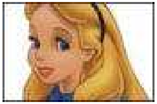
$$Adv_{A\mathcal{E}}^{IND-CPA}(B) = P(1, 0) - P(0, 0)$$

General Scheme of the Proof

$$P(1,0) - P(0,0) =$$
$$[P(1,0) - P(1,1)] + [P(1,1) - P(0,1)] + [P(0,1) - P(0,0)]$$

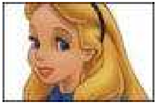
$$P(1,0) - P(1,1) \leq Adv_{A\mathcal{E}}^{IND-CPA}(A_{11,10})$$
$$P(1,1) - P(0,1) \leq Adv_{S\mathcal{E}}^{IND-CPA}(A)$$
$$P(0,1) - P(0,0) \leq Adv_{A\mathcal{E}}^{IND-CPA}(A_{00,01})$$

Challenge: Find a flaw on a simple protocol!



$$\xrightarrow{\quad} \{N_A, A\}_{K_B} \xrightarrow{\quad}$$

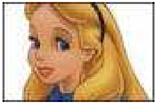
Challenge: Find a flaw on a simple protocol!



$$\xrightarrow{\qquad\qquad} \{N_A, A\}_{K_B} \xrightarrow{\qquad\qquad}$$

$$\xleftarrow{\qquad\qquad} \{N_A, N_B\}_{K_A} \xleftarrow{\qquad\qquad}$$

Challenge: Find a flaw on a simple protocol!

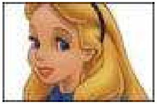


$$\xrightarrow{\qquad \{N_A, A\}_{K_B} \qquad}$$

$$\xleftarrow{\qquad \{N_A, N_B\}_{K_A} \qquad}$$

$$\xrightarrow{\qquad \{N_B\}_{K_B} \qquad}$$

Challenge: Find a flaw on a simple protocol!



$\longrightarrow \{N_A, A\}_{K_B} \longrightarrow$

$\longleftarrow \{N_A, N_B\}_{K_A} \longleftarrow$

$\longrightarrow \{N_B\}_{K_B} \longrightarrow$

Question

- Is N_B a shared secret between A et B?

Communication Protocols

- **What are communication protocols?**
  - A set of rules that governs the interaction and transmission of data exchanged between agents.
  - **Examples**
    - TCP and UDP govern computer interactions over an IP network
    - HTTP governs the exchange of data in the hypertext format.
    - SMTP governs the exchange of e-mail.

Security Protocols

- **What are security protocols?**
  - Communication protocols that operate in an untrusted network or among (some) untrusted agents (principals), and that are used to achieve security goals against a threat model
  - Ingredients:
    - Principal: a protocol participant, typically human or computer
    - Security Goal: the confidentiality or integrity of data, the authentication of a principal
    - Threat model: the capabilities of the attacker
  - They use cryptographic primitives as basic primitives
  - Examples: TLS, IPsec, SSH, WPA, Kerberos, Needham-Schroeder,...

Example - Online Banking

- Cryptographic protocol: TLS (HTTPS)
  - Principals: Web Browser, Bank Website
  - Security Goal: the confidentiality and integrity of data, server authentication
  - Threat model: network attacker, phishing website
- Password-based authentication protocol
  - Principals: Bank Client, Bank Website
  - Security Goal: client authentication
  - Threat model: dishonest client

Credit card payment

- Online Transaction Authorization
  - Principals: Credit Card, Terminal, Bank Website
  - Security Goal: transaction data integrity, card authentication
  - Threat model: fake card, tampered terminal
- Cardholder verification (PIN Entry)
  - Principals: Client, Credit Card, Terminal
  - Security Goal: client authentication
  - Threat model: stolen credit card

Modelling security protocols

- Security protocols are small (but **critical**) components embedded within large distributed applications: for example, TLS within a web browser
- the security of the system depends on their correctness
- small recipes, but non trivial to design and understand: a long history of attacks on academic or real word protocols (many attacks, several years later after deployment!)
- why is it so difficult to correctly design security protocols?
  - *cryptografic guarantees* provided by primitives are often misunderstood
  - *rich threat models* are difficult to reason about and to test

Building a secret establishment protocol

- an attempt to design a simple good protocol
- principals:
    - a set of users, denoted by a, b, c, . . . (not all are necessarily honest!)
    - an **honest** server denoted by s
- security goal: to establish a new secret
    - this can be used to generate new session keys (further communication protocols between the same principals may be based on these keys)

A secret establishment protocol - security goals

- we consider a protocol with 3 roles A, B (intended to be played by users) and S (intended to be played by the server)
- we suppose that there is an intruder i (but nobody knows that i is not honest!)
- aims of the protocol:
  - at the end of the protocol, k_{AB} should be known to A and B, and possibly to S (who is supposed to forget it immediately), but to no other parties
  - A and B can assume that k_{AB} is newly (freshly) generated

A secret establishment protocol - first attempt

- A protocol is informally specified as a sequence of messages m exchanged between roles (principals) A and B

$$A \longrightarrow B : \quad m \qquad \text{(also called Alice-Bob notation)}$$

- first attempt: a protocol that consists of 3 messages

$1.A \longrightarrow S : \ A, B$	1. A contacts S by sending the identities of the 2 parties who are going to share the secret
$2.S \longrightarrow A : \ K_{AB}$	2. S sends the secret K_{AB} to A
$3.A \longrightarrow B : \ K_{AB}, A$	3. A passes K_{AB} together with its identity to B

- K_{AB} does not contain information about A and B, it is simply a symbolic name
- to notice the significantly incomplete protocol specification
- we use both M, N and $\langle M, N \rangle$ to denote concatenation

A secret establishment protocol - a naive attempt

- Is the protocol correct? Recall that only the users that play the roles *A* and *B* may learn the secret
- In an ideal world, maybe... but a realistic assumption on typical communication networks is:

  Security Assumption 1: the intruder is able to steal all messages sent in the protocol.

$\Rightarrow$ Use cryptography (symmetric/asymmetric encryption, signatures, hashes ...).

A secret establishment protocol - using crypto

- assume that S initially shares a secret key $sk(U, S)$ with each user U of the system. $\{\ m\ \}_k$ is symmetric encryption of m using the key k (more on this later...)

$$
\begin{aligned}
&1. A \longrightarrow S: \ A, B \\
&2. S \longrightarrow A: \ \{\ K_{AB}\ \}_{sk(A,S)}, \{\ K_{AB}\ \}_{sk(B,S)} \\
&3. A \longrightarrow B: \ \{\ K_{AB}\ \}_{sk(B,S)}, A
\end{aligned}
$$

- Problems? No for secrecy, since:
  Perfect Cryptography Assumption : an encrypted message can be decrypted only with the appropriate key.
- Is really B, sharing K_{AB} (only) with A?
  Security Assumption 2: the intruder is able to intercept and send messages to anybody, under any sender name (that is, he can impersonate any other participant in the protocol.

A secret establishment protocol - despite an intruder

- so basically, we assume two things (such an adversary is called a Dolev-Yao adversary):
  - the adversary has complete control over the network
    - **interception** $A \longrightarrow O(B) : m$
    - **injection** $O(A) \longrightarrow B : m$
    - **compromise** $A \longrightarrow O : m$
  - the adversary cannot break cryptography
- although generally there at most 4 or 5 messages exchanged in a legitimate run of the prootcol, there are an infinite number of variations in which the adversary can participate
- there are several sources of infinity: number of sessions, number of messages that adversary can create, number of nonces

A secret establishment protocol - a simple attack

$$\begin{array}{ll} 1.1 & a \longrightarrow s : \ a, b \\ 1.2 & s \longrightarrow a : \ \{\, k_{ab} \,\}_{sk(a,s)}, \{\, k_{ab} \,\}_{sk(b,s)} \\ 1.3 & a \longrightarrow i(b) : \ \{\, k_{ab} \,\}_{sk(b,s)}, a \\ 2.3 & i(x) \longrightarrow b : \ \{\, k_{ab} \,\}_{sk(b,s)}, x \end{array}$$

- we denote by $m.n$ the nth step of the mth session
- The intruder i intercepts the message from a to b and replaces x for a's identity (x is any agent name).

$\rightarrow$ Problem: b believes that he is sharing the key with x, whereas he is sharing it with a.

A secret establishment protocol - a simple attack (II)

$$\begin{array}{ll}
1.1 & a \longrightarrow s :\ a, b \\
1.2 & s \longrightarrow a :\ \{\, k_{ab} \,\}_{sk(a,s)}, \{\, k_{ab} \,\}_{sk(b,s)} \\
1.3 & a \longrightarrow i(b) :\ \{\, k_{ab} \,\}_{sk(b,s)}, a \\
2.3 & i(x) \longrightarrow b :\ \{\, k_{ab} \,\}_{sk(b,s)}, x
\end{array}$$

- Consequences: it depends on the context in which the protocol is used, for example, b can give away to a information which should have been shared only with x
- Hence, even if i does not get k_{ab}, the protocol is broken since it does not satisy the requirement that the users should know who else knows the session key.

A secret establishment protocol - a clever attack

$$\begin{array}{lll}
1.1 & a \longrightarrow i(s): & a, b \\
2.1 & i \longrightarrow s: & i, a \\
2.2 & s \longrightarrow i: & \{\, k_{ia} \,\}_{sk(i,s)}, \{\, k_{ia} \,\}_{sk(a,s)} \\
1.2 & i(s) \longrightarrow a: & \{\, k_{ia} \,\}_{sk(a,s)}, \{\, k_{ia} \,\}_{sk(i,s)} \\
1.3 & a \longrightarrow i(b): & \{\, k_{ia} \,\}_{sk(i,s)}, a
\end{array}$$

- i intercepts the message intended to be for s, and replaces b's identity with its own identity, so that s generates a key (a bit-string) k_{ia}, and encrypts it with the keys of the two supposed users i and a
- Since a cannot distinguish between encrypted messages (if she does not have the good key!), she will not detect the alteration

A secret establishment protocol - a clever attack (II)

$$
\begin{array}{lll}
1.1 & a \longrightarrow i(s): & a, b \\
2.1 & i \longrightarrow s: & i, a \\
2.2 & s \longrightarrow i: & \{\, k_{ia} \,\}_{sk(i,s)}, \{\, k_{ia} \,\}_{sk(a,s)} \\
1.2 & i(s) \longrightarrow a: & \{\, k_{ia} \,\}_{sk(a,s)}, \{\, k_{ia} \,\}_{sk(i,s)} \\
1.3 & a \longrightarrow i(b): & \{\, k_{ia} \,\}_{sk(i,s)}, a
\end{array}
$$

- Hence, a will belive that the session has been successfully completed with b, and that she is sharing the key k_{ia} with b, whereas i knows this key; so i can further masquerade as b and learn all information that a intends to send to b.
- In this attack we suppose that i is a legitimate user registered to s.

Security Assumption 3: the intruder can be a legitimate protocol participant, or is able to corrupt legitimate participants (an insider), an external party (an outsider)

A secret establishment protocol - a clever attempt

- to prevent this kind of attacks, a good principle is to bound cryptographically the names of the participants A and B to the key k_{ab} intended to be shared by them.

$$\begin{array}{ll}1. A \longrightarrow S: & A, B \\ 2. S \longrightarrow A: & \{\, K_{AB}, B \,\}_{sk(A,S)}, \{\, K_{AB}, A \,\}_{sk(B,S)} \\ 3. A \longrightarrow B: & \{\, K_{AB}, A \,\}_{sk(B,S)}\end{array}$$

- Neither of the previous two attacks does works. Can you justify that the intruder is unable to attack it by eavesdropping, altering and injecting his own messages?
- Still a problem: What about old keys...?

The intruder has memory

- The problem is that the quality of long-term encrypting keys is much better than that of the session keys generated during each exection of the protocol.
- Consequences: communications in different sessions should be separated; in particular, it should be impossible to replay messages from old sessions.

Security Assumption 4: the intruder is able to obtain the value of any "sufficiently old" session key K_{AB} generated in a previous run of the protocol.

The intruder has memory(II)

$$
\begin{array}{lll}
1.1 & a \longrightarrow s : & a, b \\
1.2 & s \longrightarrow a : & \{\ k_{ab}, b\ \}_{sk(a,s)}, \{\ k_{ab}, a\ \}_{sk(b,s)} \\
1.3 & a \longrightarrow b : & \{\ k_{ab}, a\ \}_{sk(b,s)} \\
\ldots & & \\
2014.1 & a \longrightarrow i(s) : & a, b \\
2014.2 & i(s) \longrightarrow a : & \{\ k_{ab}, b\ \}_{sk(a,s)}, \{\ k_{ab}, a\ \}_{sk(b,s)} \\
2014.3 & a \longrightarrow b : & \{\ k_{ab}, a\ \}_{sk(b,s)}
\end{array}
$$

- i masquerades as s, and replays a “very old” key k_{ab}
- Consequences:
  - By **Assumption 1**, i can be expected to know the older encrypted messages $\{\ k_{ab}, b\ \}_{sk(a,s)}$ and $\{\ k_{ab}, a\ \}_{sk(b,s)}$.
  - By **Assumption 4**, i can be expected to know the value of k_{ab} (by cryptanalysis ... or blackmail).
  - Thus i is able to decrypt subsequent messages encrypted with k_{ab} or even to inject messages whose integrity must be protected by k_{ab}.

The intruder has memory(III)

- Even if i does not obtain k_{ab}, the previous attack can be regarded as successful
  - i has succeded in making a and b to accept an old session key!
  - i can now replay messages protected by k_{ab} in the previous session.
- Imagine a further step in the context of the protocol:

$$4. A \longrightarrow B : \quad \{\ request\ \}_{K_{AB}}$$

The intruder has memory(IV)

1.1	$a \longrightarrow s$:	a, b
1.2	$s \longrightarrow a$:	$\{\ k_{ab}, b\ \}_{sk(a,s)}, \{\ a, k_{ab}\ \}_{sk(b,s)}$
1.3	$a \longrightarrow b$:	$\{\ k_{ab}, a\ \}_{sk(b,s)}$
1.4	$a \longrightarrow b$:	$\{\ \text{transfer 10000 euro to i}\ \}_{k_{ab}}$
. . .		
2014.3	$i(a) \longrightarrow b$:	$\{\ k_{ab}, a\ \}_{sk(b,s)}$
2014.4	$i(a) \longrightarrow b$:	$\{\ \text{transfer 10000 euro to i}\ \}_{k_{ab}}$

- You could say: is b memoryless (he should remember k_{ab})?
- “The Principals don’t think”, but they only follow the protocol!

A secret establishment protocol - using timestamps

- To prevent replays, we could add a timestamp T to each message:

$$\begin{aligned}
&1. A \longrightarrow S: && A, B \\
&2. S \longrightarrow A: && \{ K_{AB}, B, T \}_{sk(A,S)}, \{ K_{AB}, A, T \}_{sk(B,S)} \\
&3. A \longrightarrow B: && \{ K_{AB}, A, T \}_{sk(B,S)}
\end{aligned}$$

- B should reject messages that are older than some threshold Δ
- This requires the clocks at A and B to be synchronized
- It still leaves a window of opportunity for replay attacks (within Δ)
- Another solution: use nonces (fresh randomly-generated values intended to be used only once).

A secret establishment protocol - using nonces

$$\begin{array}{ll}
1. A \longrightarrow S : & A, B, N_A \\
2. S \longrightarrow A : & \{ K_{AB}, B, N_A, \{ K_{AB}, A \}_{sk(B,S)} \}_{sk(A,S)} \\
3. A \longrightarrow B : & \{ K_{AB}, A \}_{sk(B,S)} \\
4. B \longrightarrow A : & \{ N_B \}_{K_{AB}} \\
5. A \longrightarrow B : & \{ N_B - 1 \}_{K_{AB}}
\end{array}$$

- A sends her nonce N_A to S in order to get a new key
- B, once he get the new key, challenges A with a fresh nonce N_B
- N_A and N_B do not identify who created them, are just variable names!
- This is a famous protocol proposed by Needham and Schroeder. Is it secure?

A secret establishment protocol - still attacks

$$\begin{array}{lll}
1. & a \longrightarrow s : & a, b, n_a \\
2. & s \longrightarrow a : & \{\, k_{ab}, b, n_a, \{\, k_{ab}, a \,\}_{sk(b,s)} \,\}_{sk(a,s)} \\
3. & a \longrightarrow b : & \{\, k_{ab}, a \,\}_{sk(b,s)} \\
4. & b \longrightarrow a : & \{\, n_b \,\}_{k_{ab}} \\
5. & a \longrightarrow b : & \{\, n_b - 1 \,\}_{k_{ab}} \\
\ldots & & \\
2014.4 & i(a) \longrightarrow b : & \{\, k_{ab}, a \,\}_{sk(b,s)} \\
2014.4. & b \longrightarrow i(a) : & \{\, n_b \,\}_{k_{ab}} \\
2014.5. & i(a) \longrightarrow b : & \{\, n_b - 1 \,\}_{k_{ab}}
\end{array}$$

- the same attack as before: *i* masquerades *a* and can convince *b* to use an old key

A secret establishment protocol - a good protocol?

$$\begin{array}{ll}
1. B \longrightarrow A: & B, N_B \\
2. A \longrightarrow S: & A, B, N_A, N_B \\
3. S \longrightarrow A: & \{ K_{AB}, B, N_A \}_{sk(A,S)}, \{ K_{AB}, A, N_B \}_{sk(B,S)} \\
4. A \longrightarrow B: & \{ K_{AB}, A, N_B \}_{sk(B,S)}
\end{array}$$

- Is "the good" protocol?
- The protocol avoids all previous attack, as long as the encryption provides confidentiality and integrity...
- It is not a proof... it is just intuition.
- But neither A nor B can deduce at the end of a session that the other actually received the key K_{AB}

A secret establishment protocol - a perfect protocol?

$$
\begin{array}{ll}
1. B \longrightarrow A: & B, N_B \\
2. A \longrightarrow S: & A, B, N_A, N_B \\
3. S \longrightarrow A: & \{ K_{AB}, B, N_A \}_{sk(A,S)}, \{ K_{AB}, A, N_B \}_{sk(B,S)} \\
4. A \longrightarrow B: & \{ K_{AB}, A, N_B \}_{sk(B,S)}, \{ A, N_B \}_{K_{AB}} \\
5. B \longrightarrow A: & \{ N_B - 1 \}_{K_{AB}}
\end{array}
$$

- We added a final communication exchange that allows to both A and B to check that the other received the key K_{AB}
- Is this protocol “perfect”?

Other attacks

- All previous attacks were logical or symbolic attacks: they supposed that cryptography is perfect!
- Suppose now that K_{AB} is a password of 8 characters, and that the encryption algorithm is deterministic.
The protocol (recall).

$$\begin{array}{llll} \dots & & & \\ 1. & b \longrightarrow a : & & b, n_b \\ \dots & & & \\ 4. & a \longrightarrow b : & & \dots, \{\, a, n_b \,\}_{k_{ab}} \\ \dots & & & \end{array}$$

- i generates all strings of 8 characters pwd, and since he knows both a, n_b and the encryption algorithm, he can check whether $\{\, a, n_b \,\}_{k_{ab}}$ and $\{\, a, n_b \,\}_{pwd}$ are equal. This is called a "dictionary attack".

Needham-Schroeder protocol

- assume that each user U has a pair of public/private keys $(pk(U), sk(U))$ such that everybody knows $pk(U)$, but U is the only one to know $sk(U)$. Moreover, we denote by $\{\!|m|\!\}_{pk}$ the asymmetric encryption of m using the public key pk.
- NS protocol (1978)

$$\begin{array}{lll} 1. & A \longrightarrow B : & \{\!|A, N_A|\!\}_{pk(B)} \\ 2. & B \longrightarrow A : & \{\!|N_A, N_B|\!\}_{pk(A)} \\ 3. & A \longrightarrow B : & \{\!|N_B|\!\}_{pk(B)} \end{array}$$

- Is really B, sharing N_A, N_B (only) with A?
- Gawin Lowe found an attack in 1995 (17 years later!). Can you see it?

Needham-Schroeder protocol - an attack

- man-in-the-middle attack (1995)

1.1.	$a \longrightarrow i :$	$\{\!\{a, n_a\}\!\}_{pk(i)}$
2.1.	$i(a) \longrightarrow b :$	$\{\!\{a, n_a\}\!\}_{pk(b)}$
2.2.	$b \longrightarrow i(a) :$	$\{\!\{n_a, n_b\}\!\}_{pk(a)}$
1.2.	$i \longrightarrow a :$	$\{\!\{n_a, n_b\}\!\}_{pk(a)}$
1.3.	$a \longrightarrow i :$	$\{\!\{n_b\}\!\}_{pk(i)}$
2.3.	$i(a) \longrightarrow b :$	$\{\!\{n_b\}\!\}_{pk(b)}$

- b "thinks" he is talking to a, while he is talking to i, and moreover, n_b (which maybe is intended to be the shared session secret) was disclosed to i

Needham-Schroeder-Lowe protocol

- Fixing the Lowe attack

$$\begin{array}{lll} 1. & A \longrightarrow B : & \{\!|A, N_A|\!\}_{pk(B)} \\ 2. & B \longrightarrow A : & \{\!|N_A, N_B, B|\!\}_{pk(A)} \\ 3. & A \longrightarrow B : & \{\!|N_B|\!\}_{pk(B)} \end{array}$$

- ... and a type-flaw attack

$$\begin{array}{lll} 1.1. & i(a) \longrightarrow b : & \{\!|a, i|\!\}_{pk(b)} \\ 1.2. & b \longrightarrow i(a) : & \{\!|i, n_b, b|\!\}_{pk(a)} \\ 2.1. & i \longrightarrow a : & \{\!|i, \langle n_b, b\rangle|\!\}_{pk(a)} \\ 2.2. & a \longrightarrow i : & \{\!|\langle n_b, b\rangle, n_a, a|\!\}_{pk(i)} \\ 1.3. & i(a) \longrightarrow b : & \{\!|n_b|\!\}_{pk(b)} \end{array}$$

- To fix: change $\{\!|N_A, N_B, B|\!\}_{pk(A)}$ to $\{\!|B, N_A, N_B|\!\}_{pk(A)}$ or encode messages correctly (tag each field with its type)

The perfect cryptography and the real world

- And if cryptography is not perfect?
- Let us take a simple El-Gamal encryption scheme:
  - a cyclic group G of order q and generator g
  - $pk(U) = g^{sk(u)}$, and messages are elements of G: $m = g^{m'}$ for some m'
  - $\{\!\{m\}\!\}_{pk} = (pk^r \times g^{m'}, g^r)$ for some randomly sampled r (where $g^{m'}$ is the encoding of m).
  - let us assume that $\langle m_1, m_2 \rangle$ is obtained by concatenating the bit-strings, i.e. is mapped to $g^{m'_1 + 2^{|m'_1|} \times m'_2}$
  - then encryption is malleable: if one knows $|m|$ and $n = g^{n'}$, then one can transform $\{\!\{m, n\}\!\}_{pk}$ into $\{\!\{m, p\}\!\}_{pk}$ for any known $p = g^{p'}$: if $\{\!\{m, n\}\!\}_{pk} = (bs_1, bs_2)$, then $\{\!\{m, p\}\!\}_{pk} = (bs_1 \times (g^{n'})^{-2^{|m'|}} \times (g^{p'})^{2^{|m'|}}, bs_2)$.

A computational attack

- ... and we get a computational attack

1.1. $a \longrightarrow i$:	$\{\!\vert a, n_a \vert\!\}_{pk(i)}$
2.1. $i(a) \longrightarrow b$:	$\{\!\vert a, n_a \vert\!\}_{pk(b)}$
2.2. $b \longrightarrow a$:	$\{\!\vert n_a, n_b, b \vert\!\}_{pk(a)}$
computation step :	i *computes* $\{\!\vert n_a, n_b, i \vert\!\}_{pk(a)}$
1.2. $i \longrightarrow a$:	$\{\!\vert n_a, n_b, i \vert\!\}_{pk(a)}$
1.3. $a \longrightarrow i$:	$\{\!\vert n_b \vert\!\}_{pk(i)}$
2.3. $i(a) \longrightarrow b$:	$\{\!\vert n_b \vert\!\}_{pk(b)}$

- b is convinced that he is sharing a secret value n_b and that is talking with a
- Symbolic model vs. Computational model

Security protocols - general goals

- **Confidentiality:** Can the intruder *i* learn a secret that is meant to be shared only by *a* and *b*?
- **Integrity:** Can the intruder *i* modify a message from *a* and get it accepted by *b*?
- **Authentication:** Can the intruder *i* convince *b* that is talking to *a*?
- **Anonymity:** If *a* wishes to remain anonymous, can the intruder *i* disclose its identity?
- **Non-repudiation:** If *a* sends a message, can she later denies that she sent the message? Or, if *b* receives a message, can he later denies that he got the message?
- **Non-repudiation:** Can one of *a* or *b* obtain an unfair advantage befor the transaction is completed?

E-vote protocols - specific goals

- **Eligibility:** Only legitimate voter can vote, and only once.
- **Vote privacy:** the vote of any honest voter is not revealed
- **Receipt-freenes:** a voter cannot prove she vote in a certain way
- **Coercion-resistance:** a voter cannot prove she vote in a certain way, even if the intruder ineracts with the voter during the voting process
- **Individual verifiability:** a voter can verify that her vote was counted
- **Universal verifiability:** the result of the vote was correctly computed
- **Fairness:** no partial results can be obtained before the end of the process vote

Cryptographic primitives : symmetric encryption

$$\begin{tikzcd}
{} \arrow[r, "\text{Bonjour Alice}"] & \text{encrypting} \arrow[r, "\textit{ugtrytt kjhvlk}"] & \text{decrypting} \arrow[r, "\text{Bonjour Alice}"] & {} \\
& \text{shared key } sk \arrow[u] & \text{shared key } sk \arrow[u] &
\end{tikzcd}$$

- **Notation** $\{\ m\ \}_{sk}$
- **Decrypting:** $dec(\{\ m\ \}_{sk}, sk) = m$
- **Security property:** Informally, a ciphertext cannot be decrypted without knowing the key.
- It is fast, useful to encrypt large messages
- **Examples:** DES, AES, RC4 ,...

Cryptographic primitives : asymmetric encryption

$$\begin{tikzcd}
{} \arrow[r, "\text{Bonjour Alice}"] & \text{encrypting} \arrow[r, "\textit{ugtrytt kjhvlk}"] & \text{decrypting} \arrow[r, "\text{Bonjour Alice}"] & {} \\
& \text{public key } pk \arrow[u] & \text{private key } sk \arrow[u] &
\end{tikzcd}$$

- **Notation** $[\![m]\!]_{sk}$
- **Decrypting:** $dec([\![m]\!]_{sk}, pk) = m$, where $pk = sk^{-1}$
- **Security property:** Informally, a ciphertext encrypted with the public key cannot be decrypted without knowing the inverse private key.
- It is slower, but allows to communicate with an unknown participant (by means of a public key infrastructure PKI)
- **Examples:** RSA, ElGamal, Cramer-Shoup, OAEP+,...

Cryptographic primitives : signature

$$\begin{tikzcd} {} \arrow[r, "\text{Bonjour Alice}"] & \text{signing} \arrow[r, "\textit{ugtrytt kjhvlk}"] & \text{verifying} \arrow[r, "\text{Bonjour Alice}"] & {} \\ & \text{private key } sk \arrow[u] & \text{public key } pk \arrow[u] & \end{tikzcd}$$

- **Notation $[m]_{pk}$**
- **Verifying:** $ver([m]_{sk}, pk) = true$, where $sk = pk^{-1}$
- **Security property:** Informally, a signature that can be verified using a public key pk, cannot be created without knowing the inverse private key sk.
- **Examples:** RSA, DSA, ECDSA,...

Other cryptographic primitives

- A hashing function h produces for a large message m, a small message $h(m)$ called hash. **Informal security properties:**
  - If $m_1 \neq m_2$, then $h(m_1) \neq h(m_2)$.
  - Given only $h(m)$, it is impossible to find m.
- Nonces, denoted $n_a, n_b, \ldots, n_1, n_2, \ldots$ are randomly generated large values. **Informal security properties:**
  - $n_1 \neq n_2$ for any independent generated nonces.
  - Given only partial information about n, it is impossible to find entire n.

From attacks to proofs

- Can we be confident that a protocol given in the Alice-Bob notation is correct?
- How can we rigorously prove that the protocol achieves his security goals?
  - What does $A \longrightarrow B$ mean? How A and B parses the received messages and what exactly they do?
  - How we specify formally the security goals?
  - How do we capture the threat model?
- Using our informal notation, we can find and describe attacks.
- To prove mathematically that a given protocol satisfy a security goal, we have to move to a more formal setting.

Proofs of protocols : the symbolic model

- The **Dolev-Yao model** was proposed in 1983:
  - messages are terms in a free algebra: $\{ m \}_k$, $[\![m]\!]_k$, $\mathbf{[}m\mathbf{]}_k$, $h(m)$...
  - no secret value (nonce or key) can be guessed
  - the intruder can only apply functions in the given signature
  - ... but has complete control of the network
- One can add equations between primitives (functions), but anyway, we assume that the only equalities are those given by these equations.
- Proofs in this model can be automated.

Proofs of protocols : the computational model

- The **computational model** was proposed in the '80s:
  - the messages are *bitstrings*
  - the cryptographic primitives are *operations on bitstrings*
  - the intruder is any *probabilistic polynomial-time Turing machine*
  - has complete control of the network
- The model is much closer to reality, but proofs in this model are mostly manual.

Proofs of protocols : (no) side channels

- The **computational model** is still an abstraction, which does not exactly match the reality. It ignores:
  - timing, power consumption, noise
  - tampering attacks
  - errors in implementation: what to answer if a message does not have the expected form?
- In this course we will ignore such kind of attacks.

Proofs of protocols in the symbolic model

- One must compute **the set of terms** that the attacker can obtain.
- This set can be infinite for several reasons:
  - unbounded number of sessions
  - unbounded size of messages
  - unbounded number of fresh values

Complexity

- If everything is unbounded, then proving security is undecidable.
- If everything is bounded, then we get a finite state transition system, and we can apply model checking!
- Bounding only the number of sessions: insecurity is typically **NP-complete**

Solutions to undecidability in the general case

- Uses approximations (generally preserving soundness):
  - abstract interpretation
  - typing
- Allow non-termination.

Proverif uses approximations and allow non-termination.

Relevance of the symbolic model

- Advantages:
  - Numerous attacks have been found.
  - An attack in the symbolic model can be easily translated in a practical atttack in the computational model.
  - Proofs are simpler and can be automated.
- Drawback: in general, a proof in the symbolic model, does not imply a proof in the computational model.

Link between the two models

- Computational soundness theorems:

  Proof in the symbolic model $\Longrightarrow$ Proof in the computational model

  modulo additional assumptions (initiated by Abadi and Rogaway[2000]).

Paper short presentation and short report

- How: by team (2 by team for M2R, 3 by team for M2P).
- When: presentation to be held at 23 November, report to be sent by 7 December.
- Where to chose an article (the list): `www-verimag.imag.fr/~ene/m2p/papers.html`

Proofs of protocols in the symbolic model

- A decidability result for intruder deduction
  - Deduction is PTIME complete
- A decidability result for secrecy for bounded number of sessions
  - Bounded protocol security is NP complete
- A undecidability result for secrecy for unbounded number of sessions
  - Proof techniques that require user input or allow non-termination

Proofs of protocols in the symbolic (Dolev-Yao) model

- Intruder controls the network and can:
  - intercept messages
  - modify messages
  - block messages
  - generate new messages
  - insert new messages
- Perfect cryptography:
  - Abstraction with terms algebra
  - Decryption only if inverse key is known
- Protocol has
  - Arbitrary number of principals
  - Arbitrary number of parallel sessions
  - Messages with arbitrary size

Passive Intruder

Let as assume

- bounded number of sessions
- adversary can only read messages

Symbols and arities

- We consider
  - $\mathcal{F}$ is a finite set
  - *Arity* is a mapping from $\mathcal{F}$ into $\mathbb{N}$
  - $(\mathcal{F}, \mathit{Arity})$ is a **ranked alphabet** or **signature** denoted Σ
- Then:
  - The **arity** of a symbol $f \in \mathcal{F}$ is $\mathit{Arity}(f)$
  - The set of symbols of arity p is denoted by $\mathcal{F}_p$.
  - Elements of arity 0, 1, $\ldots p$ are respectively called constants, unary, $\ldots p$-ary symbols.
- In Dolev-Yao theory we consider $\mathcal{F} = \{\mathbf{enc}/2, \mathbf{pair}/2, s/0, i/0, a/0, b/0, \ldots\}$.
  We denote $\{\_\}_{\_}$ for $\mathbf{enc}(\_,\_)$ and $\langle \_, \_ \rangle$ for $\mathbf{pair}(\_,\_)$.

Terms

Let $\mathcal{X}$ be a set of symbols called **variables**.

Definition

The set $\mathcal{T}(\mathcal{F},\mathcal{X})$ of **terms** over the ranked alphabet $\mathcal{F}$ and the set of variables $\mathcal{X}$ is the smallest set defined by:

- $\mathcal{F}_0 \cup \subseteq \mathcal{T}(\mathcal{F},\mathcal{X})$
- if $f \in \mathcal{F}_p$ and $t_1,\ldots,t_p \in \mathcal{T}(\mathcal{F},\mathcal{X})$, then $f(t_1,\ldots,t_p) \in \mathcal{T}(\mathcal{F},\mathcal{X})$.

- If $\mathcal{X} = \emptyset$ then $\mathcal{T}(\mathcal{F},\mathcal{X})$ is also written $\mathcal{T}(\mathcal{F})$. Terms in $\mathcal{T}(\mathcal{F})$ are called **ground terms** or simply messages.

Example

Let $\mathcal{F} = \{\{\ \_\ \}_\_/2, \langle \_, \_ \rangle/2, \mathbf{a}/0, \mathbf{k_1}/0, \mathbf{k_2}/0, \mathbf{0}/0, \mathbf{1}/0\}$ and $\mathcal{X} = \{x, y, z\}$.

$\langle x, \mathbf{1} \rangle$, $\{\ x\ \}_{\mathbf{k_1}}$ $\{\ \langle z, \mathbf{a} \rangle\ \}_{\mathbf{k_2}}$, $\{\ \langle \mathbf{a}, \mathbf{0} \rangle\ \}_{\mathbf{k_1}}$ are terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$.

Dolev-Yao Deduction System

We denote $T_0 \vdash s$ for "term s is deducible from the set of terms T_0".

Deduction System for Dolev Yao theory: $T_0 \vdash^? s$

(A) $\dfrac{u \in T_0}{T_0 \vdash u}$

(UL) $\dfrac{T_0 \vdash \langle u, v\rangle}{T_0 \vdash u}$

(P) $\dfrac{T_0 \vdash u \qquad T_0 \vdash v}{T_0 \vdash \langle u, v\rangle}$

(UR) $\dfrac{T_0 \vdash \langle u, v\rangle}{T_0 \vdash v}$

(C) $\dfrac{T_0 \vdash u \qquad T_0 \vdash v}{T_0 \vdash \{u\}_v}$

(D) $\dfrac{T_0 \vdash \{u\}_v \qquad T_0 \vdash v}{T_0 \vdash u}$

Proof System

- A **sequent** is an expression of the form $T \vdash u$.
- A **proof** of a sequent $T \vdash u$ is a tree whose nodes are labeled by either sequents or expressions of the form "$v \in T$", such that:
  - Each leaf is labeled by an expression of the form $v \in T$, and each non-leaf node is labeled by an sequent.
  - Each node labeled by a sequent $T \vdash v$ has n children labeled by $T \vdash s_1, \ldots, T \vdash s_n$ such that there is an instance of an inference rule with conclusion $T \vdash v$ and **hypotheses** $T \vdash s_1, \ldots, T \vdash s_n$.
  - The **root** of the tree is labeled by $T \vdash u$.
- A **subproof** of a proof P is a subtree of P.

Notions for Proof System

Definition

- The **size of a proof** P of $T \vdash u$, denoted by $|P|$, is the number of nodes in the proof.
- A proof P of $T \vdash u$ is **minimal** if there does not exist a proof P' of $T \vdash u$ such that $|P'| < |P|$.

Example: $T_0 \vdash^? s$

Example

$T_0 = \{k, \{b\}_c, \langle a, \{c\}_k \rangle\}$ and $s = b$

$$(D)\dfrac{(A)\dfrac{\{b\}_c \in T_0}{T_0 \vdash \{b\}_c} \quad (D)\dfrac{(UR)\dfrac{(A)\dfrac{\langle a, \{c\}_k \rangle \in T_0}{T_0 \vdash \langle a, \{c\}_k \rangle}}{T_0 \vdash \{c\}_k} \quad (A)\dfrac{k \in T_0}{T_0 \vdash k}}{T_0 \vdash c}}{T_0 \vdash b}$$

Exercise: $T_0 \vdash^{?} s$

Is it possible from T_0 to deduce s

- $T_0 = \{a, k\}$ and $s = \langle a, \{a\}_k \rangle$
- $T_0 = \{a, k\}$ and $s = \langle b, \{k\}_a \rangle$
- $T_0 = \{\{k\}_a, b\}$ and $s = \langle \{b\}_{\{k\}_a}, \{k\}_a \rangle$
- $T_0 = \{\langle a, \{k\}_a \rangle\}$ and $s = \{\langle a, \{k\}_a \rangle\}_k$

Syntactic Subterms

$S(t)$ is the smallest set such that:

- $t \in S(t)$
- $\langle u, v \rangle \in S(t) \Rightarrow u, v \in S(t)$
- $\{u\}_v \in S(t) \Rightarrow u, v \in S(t)$

Example

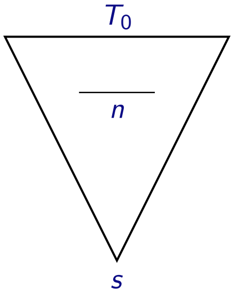
Let $t = \{\langle a, \{b\}_{k_2} \rangle\}_{k_1}$
Then

$$S(t) = \{t, a, b, k_1, k_2, \{b\}_{k_2}, \langle a, \{b\}_{k_2} \rangle\}$$

Definition of S-Locality

- A proof P of $T_0 \vdash s$ is S-local :

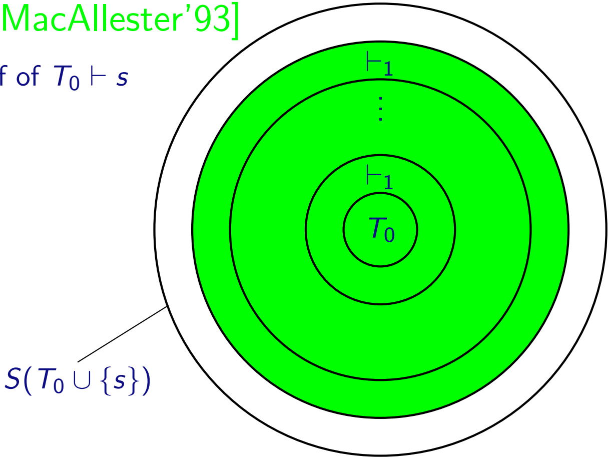
$$\forall n \in P, n \in S(T_0 \cup \{s\})$$



- A proof P of $T \vdash w$ is **S-local** if all nodes are in $S(T \cup \{w\})$.
- A proof system is **S-local** if whenever there is a proof of $T \vdash w$ then there is also a S-local proof of $T \vdash w$.

Locality Idea [MacAllester'93]

P a S-local proof of $T_0 \vdash s$



Intruder Deduction Problem : $T_0 \vdash^? s$

- S-locality
- One-step deductibility

Example: a local proof of $T_0 \vdash s$

Example

$T_0 = \{k, \{b\}_c, \langle a, \{c\}_k \rangle\}$ and $s = b$

$$
(D)\dfrac{(D)\dfrac{(UR)\dfrac{(A)\dfrac{\langle a, \{c\}_k \rangle \in T_0}{T_0 \vdash \langle a, \{c\}_k \rangle}}{T_0 \vdash \{c\}_k} \quad (A)\dfrac{k \in T_0}{T_0 \vdash k}}{T_0 \vdash c} \quad (A)\dfrac{\{b\}_c \in T_0}{T_0 \vdash \{b\}_c}}{T_0 \vdash b}
$$

$S(T_0 \cup \{s\}) = T_0 \cup \{a, b, c, \{c\}_k\}$

Example of necessity of $S(T \cup \{s\})$

Example

$T_0 = \{k, \{b\}_c, \langle a, \{c\}_k\rangle\}$ and $s = \langle b, k\rangle$

$$(P)\dfrac{(D)\dfrac{(A)\dfrac{\{b\}_c \in T_0}{T_0 \vdash \{b\}_c}\;(D)\dfrac{(UR)\dfrac{(A)\dfrac{\langle a, \{c\}_k\rangle \in T_0}{T_0 \vdash \langle a, \{c\}_k\rangle}}{T_0 \vdash \{c\}_k} \quad (A)\dfrac{k \in T_0}{T_0 \vdash k}}{T_0 \vdash c}}{T_0 \vdash b} \quad (A)\dfrac{k \in T_0}{T_0 \vdash k}}{T_0 \vdash \langle b, k\rangle}$$

$S(T_0) = T_0 \cup \{a, b, c, k, \{b\}_k, \{c\}_k\}$ but $\langle b, k\rangle \notin S(T_0)$
It is not enough.

Notice that $\langle b, k\rangle \in S(T_0 \cup \{s\})$

Example non minimal proof is not S-local

Example

$T_0 = \{k, \{c\}_k\}$ and $s = c$
$S(T_0) = T_0 \cup \{c\}$ but $\langle \{c\}_k, \{c\}_k \rangle$ it is not in $S(T_0 \cup \{s\})$

Locality Theorem

Theorem

Theorem of Locality [McAllester 93]
If a proof system P is S-local and $S(t)$ can be computed in polytime and $\vdash^{\leq 1}$ is P-time, then there is a P-time procedure to decide the deductibility in P.

McAllester's Algorithm

```
Input : $T_0, w$
$T \leftarrow T_0$;
while ($\exists s \in S(T_0, w)$ such that $T \vdash^{\leq 1} s$ and $s \notin T$)
  $T \leftarrow T \cup \{s\}$;
Output :$w \in T$
```

Locality for Dolev-Yao theory

Lemma

Let T be a set of terms, and w a term. A minimal proof of $T \vdash w$ is S-local. Moreover, if it is reduced to a leaf or ends with a decomposition rule, then all nodes are in $S(T)$.

Proof.

Let Π a minimal proof of $T \vdash w$. We prove the Lemma by induction on the size of Π $\square$

Prop

The intruder detection problem is decidable in P-time for the Dolev-Yao theory.

Active intruder and bounded number of sessions

Let as assume

- bounded number of sessions
- adversary can intercept, modify, block, generate, insert new messages

Substitution

Definition

- A **substitution** (respectively a **ground substitution**) σ is a mapping from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F}, \mathcal{X})$ (respectively into $\mathcal{T}(\mathcal{F})$) where there are only finitely many variables not mapped to themselves.
- Substitutions can be extended to $\mathcal{T}(\mathcal{F}, \mathcal{X})$ in such a way that $\forall f \in \mathcal{F}_n, \forall t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$:

$$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n)).$$

The **domain** of a substitution σ is the subset of variables $x \in \mathcal{X}$ such that $\sigma(x) \neq x$.

Example:

Let $\sigma = \{x \to N_A, y \to \{\langle N_A, N_B\rangle\}_{k_B}\}$ and $t = \langle x, \langle y, \langle x, x\rangle\rangle\rangle$.

Then,

$$\sigma(t) = \langle N_A, \langle \{\langle N_A, N_B\rangle\}_{k_B}, \langle N_A, N_A\rangle\rangle\rangle$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:
$s = a \quad t = X$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:
$s = a \quad t = X$
$\quad \sigma = \{X \to a\}$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:
$s = a \quad t = X$
$\quad \sigma = \{X \rightarrow a\}$
$s = a \quad t = p(X)$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:
$s = a \quad t = X$
$\quad \sigma = \{X \rightarrow a\}$
$s = a \quad t = p(X)$
No unifier

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:
$s = a \quad t = X$
$\quad \sigma = \{X \rightarrow a\}$
$s = a \quad t = p(X)$
No unifier
$s = p(a, X) \quad t = p(Y, b)$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:
$s = a \quad t = X$
$\quad \sigma = \{X \to a\}$
$s = a \quad t = p(X)$
No unifier
$s = p(a, X) \quad t = p(Y, b)$
$\quad \sigma = \{X \to b; Y \to a\}$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$s = a \quad t = X$

$\quad \sigma = \{X \rightarrow a\}$

$s = a \quad t = p(X)$

No unifier

$s = p(a, X) \quad t = p(Y, b)$

$\quad \sigma = \{X \rightarrow b; Y \rightarrow a\}$

$s = p(f(X), g(Z)) \quad t = p(f(a), Y)$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$s = a \quad t = X$

$\quad \sigma = \{X \rightarrow a\}$

$s = a \quad t = p(X)$

No unifier

$s = p(a, X) \quad t = p(Y, b)$

$\quad \sigma = \{X \rightarrow b; Y \rightarrow a\}$

$s = p(f(X), g(Z)) \quad t = p(f(a), Y)$

$\quad \sigma = \{X \rightarrow a; Y \rightarrow g(Z)\}$ or $\sigma = \{X \rightarrow a; Y \rightarrow g(b); Z \rightarrow b\}$

Most General Unifier

Definition

The most general unification between two terms s and t, denoted by $mgu(s,t)$ if: $\forall\sigma$ such that $s\sigma = t\sigma, \exists\theta$ such that $\sigma = mgu(s,t)\theta$

$$s = p(f(X), g(Z)) \quad t = p(f(a), Y)$$

$$\sigma_1 = \{X \to a; Y \to g(Z)\} \; \sigma_2 = \{X \to a; Y \to g(b); Z \to b\}$$

Goal

Design an algorithm that for a given unification problem $s =^? t$

- returns an mgu of s and t if they are unifiable.
- reports failure otherwise.

Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

1. Move the markers simultaneously, one symbol at a time, until both move off the end of the term (success), or until they point to two different symbols;

Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

1. Move the markers simultaneously, one symbol at a time, until both move off the end of the term (success), or until they point to two different symbols;
2. If the two symbols are both non-variables, then fail;

Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

1. Move the markers simultaneously, one symbol at a time, until both move off the end of the term (success), or until they point to two different symbols;
2. If the two symbols are both non-variables, then fail; otherwise, one is a variable (call it x) and the other one is the first symbol of a subterm (call it t):

Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

1. Move the markers simultaneously, one symbol at a time, until both move off the end of the term (success), or until they point to two different symbols;
2. If the two symbols are both non-variables, then fail; otherwise, one is a variable (call it x) and the other one is the first symbol of a subterm (call it t):
   - If x occurs in t, then fail;

Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

1. Move the markers simultaneously, one symbol at a time, until both move off the end of the term (success), or until they point to two different symbols;
2. If the two symbols are both non-variables, then fail; otherwise, one is a variable (call it x) and the other one is the first symbol of a subterm (call it t):
    - If x occurs in t, then fail;
    - Otherwise, replace x everywhere by t (including in the solution), write down "$x \to t$" as a part of the solution, and return to 1.

Example:

$$f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$$

Example:

$f(x, g(a), g(z)) =^? f(g(y), g(y), g(g(x)))$
$f(x, g(a), g(z))$

$f(g(y), g(y), g(g(x)))$

Example:

$f(x, g(a), g(z)) =^{?} f(g(y), g(y), g(g(x)))$
$f(x, g(a), g(z))$

$f(g(y), g(y), g(g(x)))$

Example:

$f(x, g(a), g(z)) =^? f(g(y), g(y), g(g(x)))$
$f(x, g(a), g(z))$

$f(g(y), g(y), g(g(x)))$

$\sigma = \{x \rightarrow g(y)\}$

Example:

$f(x, g(a), g(z)) =^? f(g(y), g(y), g(g(x)))$
$f(g(y), g(a), g(z))$

$f(g(y), g(y), g(g(g(y))))$

$\sigma = \{x \to g(y)\}$

Example:

$f(x, g(a), g(z)) =^? f(g(y), g(y), g(g(x)))$
$f(g(y), g(a), g(z))$

$f(g(y), g(y), g(g(g(y))))$

$\sigma = \{x \rightarrow g(y)\}$

Example:

$f(x, g(a), g(z)) =^? f(g(y), g(y), g(g(x)))$
$f(g(a), g(a), g(z))$

$f(g(a), g(a), g(g(g(a))))$

$\sigma = \{x \rightarrow g(a), y \rightarrow a\}$

Example:

$f(x, g(a), g(z)) =^? f(g(y), g(y), g(g(x)))$
$f(g(a), g(a), g(z))$

$f(g(a), g(a), g(g(g(a))))$

$\sigma = \{x \rightarrow g(a), y \rightarrow a\}$

Example:

$f(x, g(a), g(z)) =^? f(g(y), g(y), g(g(x)))$
$f(g(a), g(a), g(g(g(a))))$

$f(g(a), g(a), g(g(g(a))))$

$\sigma = \{x \to g(a), y \to a, z \to g(g(a))\}$

Questions

1. Correctness:
   - Does the algorithm always terminate?
   - Does it always produce an mgu for two unifiable terms, and fail for non-unifiable terms?
   - Do these answers depend on the order of operations?
2. Extension with equational theory, e.g., $ab = ba$.

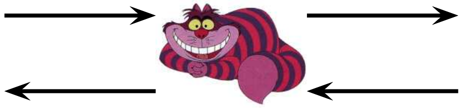
Syntactic Unification is Unitary

Theorem (Robinson)

Without equational theory there exists an unique mgu for syntactic unification (modulo renaming). Unification is called unitary.

The Intruder is the Network (Worst Case)

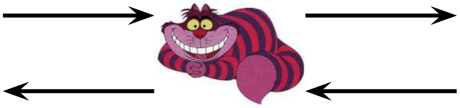
Listen



Passive: Intruder deduction problem

The Intruder is the Network (Worst Case)

Listen



Passive: Intruder deduction problem

Active Intruder Security problem

- Intercept messages (add messages to his knowledge)
- Play messages from his knowledge
- Start new sessions

Execution tree has:

- infinite branching (size of messages is not bounded)
- infinite depth (number of sessions is not bounded)

Dolev-Yao Deduction System

Deduction System : $T_0 \vdash^? s$

(A) $\dfrac{u \in T_0}{T_0 \vdash u}$

(UL) $\dfrac{T_0 \vdash \langle u, v \rangle}{T_0 \vdash u}$

(P) $\dfrac{T_0 \vdash u \qquad T_0 \vdash v}{T_0 \vdash \langle u, v \rangle}$

(UR) $\dfrac{T_0 \vdash \langle u, v \rangle}{T_0 \vdash v}$

(C) $\dfrac{T_0 \vdash u \qquad T_0 \vdash v}{T_0 \vdash \{u\}_v}$

(D) $\dfrac{T_0 \vdash \{u\}_v \qquad T_0 \vdash v}{T_0 \vdash u}$

Model: actions, roles and protocol

Action: a couple $(recv(u), send(v))$ such that $u \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{init\}$, $v \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{ok\}$. Denoted $(u \longrightarrow v)$.

Role: $R(p_1, \ldots, p_m)$ is a finite parameterized sequence of actions:

$$R(p_1, \ldots, p_m) = (u_1 \longrightarrow v_1), \ldots, (u_n \longrightarrow v_n)$$

such that $vars(v_i) \subseteq \bigcup_{1 \leq j \leq i} vars(u_j)$.

Protocol: is a finite set of roles: $P = \{R_1, \ldots, R_k\}$

Scenario: Bounded session scenario: a parallel composition of instantiated roles (also called processes):

$$||_{i=1}^{n} R^i_{j_i}(n^i_1/p^i_1, \ldots, n^i_{m_i}/p^i_{m_i})$$

Example: Needham-Schroeder protocol

Example (Needham-Schroeder)

$$\begin{array}{llcll} 1. & A & \longrightarrow & B & : \{N_a, A\}_{pk(B)} \\ 2. & B & \longrightarrow & A & : \{N_a, N_b\}_{pk(A)} \\ 3. & A & \longrightarrow & B & : \{N_b\}_{pk(B)} \end{array}$$

Two roles: R_1 initiatior, R_2 responder.

Example: Needham-Schroeder protocol

Example (Needham-Schroeder)

$$\begin{array}{llcll} 1. & A & \longrightarrow & B & : \{N_a, A\}_{pk(B)} \\ 2. & B & \longrightarrow & A & : \{N_a, N_b\}_{pk(A)} \\ 3. & A & \longrightarrow & B & : \{N_b\}_{pk(B)} \end{array}$$

Two roles: R_1 initiatior, R_2 responder.

$$\begin{array}{rl} R_1(A, B, N_a) = & (init \longrightarrow \{N_a, A\}_{pk(B)}), \\ & (\{N_a, X_{N_b}\}_{pk(A)} \longrightarrow \{X_{N_b}\}_{pk(B)}), \\ \\ R_2(B, N_b) = & (\{X_{N_a}, X_A\}_{pk(B)} \longrightarrow \{X_{N_a}, N_b\}_{pk(X_A)}) \\ & (\{N_b\}_{pk(B)} \longrightarrow ok) \end{array}$$

Example: Needham-Schroeder attack scenario

1.1.	$a \longrightarrow i$:	$\{\!|a, n_a|\!\}_{pk(i)}$
2.1.	$i(a) \longrightarrow b$:	$\{\!|a, n_a|\!\}_{pk(b)}$
2.2.	$b \longrightarrow i(a)$:	$\{\!|n_a, n_b|\!\}_{pk(a)}$
1.2.	$i \longrightarrow a$:	$\{\!|n_a, n_b|\!\}_{pk(a)}$
1.3.	$a \longrightarrow i$:	$\{\!|n_b|\!\}_{pk(i)}$
2.3.	$i(a) \longrightarrow b$:	$\{\!|n_b|\!\}_{pk(b)}$

A scenario where the Lowe attack is possible.

Example: Needham-Schroeder attack scenario

1.1.	$a \longrightarrow i$:	$[\![a, n_a]\!]_{pk(i)}$
2.1.	$i(a) \longrightarrow b$:	$[\![a, n_a]\!]_{pk(b)}$
2.2.	$b \longrightarrow i(a)$:	$[\![n_a, n_b]\!]_{pk(a)}$
1.2.	$i \longrightarrow a$:	$[\![n_a, n_b]\!]_{pk(a)}$
1.3.	$a \longrightarrow i$:	$[\![n_b]\!]_{pk(i)}$
2.3.	$i(a) \longrightarrow b$:	$[\![n_b]\!]_{pk(b)}$

A scenario where the Lowe attack is possible.

$$
\begin{aligned}
R_1(a, i, n_a) || R_2(b, n_b) = \quad & \\
& (init \longrightarrow \{n_a, a\}_{pk(i)}), \\
& (\{n_a, X_{N_b}\}_{pk(A)} \longrightarrow \{X_{N_b}\}_{pk(i)}), || \\
& \\
& (\{X_{N_a}, X_A\}_{pk(b)} \longrightarrow \{X_{N_a}, n_b\}_{pk(X_A)}), \\
& (\{n_b\}_{pk(b)} \longrightarrow ok)
\end{aligned}
$$

Denning-Sacco Protocol

$$\begin{array}{lll}
1. & A \longrightarrow S : & \langle A, B\rangle \\
2. & S \longrightarrow A : & \{\langle\langle B, N_{AB}\rangle, \langle N_s, \{\langle N_{AB}, \langle A, N_s\rangle\rangle\}_{K_{BS}}\rangle\rangle\}_{K_{AS}} \\
3. & A \longrightarrow B : & \{\langle N_{AB}, \langle A, N_s\rangle\rangle\}_{K_{BS}} \\
4. & B \longrightarrow A : & \{S_{AB}\}_{N_{AB}}
\end{array}$$

$P_{DS} = \{R_A, R_B, R_S\}$ models one session of A, B and S.

Denning-Sacco Protocol

$$\begin{array}{lll}
1. & A \longrightarrow S: & \langle A, B\rangle \\
2. & S \longrightarrow A: & \{\langle\langle B, N_{AB}\rangle, \langle N_s, \{\langle N_{AB}, \langle A, N_s\rangle\rangle\}_{K_{BS}}\rangle\rangle\}_{K_{AS}} \\
3. & A \longrightarrow B: & \{\langle N_{AB}, \langle A, N_s\rangle\rangle\}_{K_{BS}} \\
4. & B \longrightarrow A: & \{S_{AB}\}_{N_{AB}}
\end{array}$$

$P_{DS} = \{R_A, R_B, R_S\}$ models one session of A, B and S.

$$\begin{array}{ll}
R_A(A, B, S) = & (init \longrightarrow \langle A, B\rangle), \\
& (\{\langle\langle B, X_{N_{AB}}\rangle, \langle X_{N_S}, Z_A\rangle\rangle\}_{sk(A,S)} \longrightarrow Z_A), \\
& (\{W_A\}_{X_{N_{AB}}} \longrightarrow ok) \\
 & \\
R_B(B, S, S_{AB}) = & (\{Y_{N_{AB}}, \langle X_A, Y_{N_S}\rangle\rangle\}_{sk(B,S)} \longrightarrow \{S_{AB}\}_{Y_{N_{AB}}}) \\
 & \\
R_S(S, N_{AB}, N_S) = & (\langle X_A, X_B\rangle \longrightarrow \\
& \quad \{\langle\langle X_B, N_{AB}\rangle, \langle N_S, \{\langle N_{AB}, \langle X_A, N_S\rangle\rangle\}_{sk(X_B,S)}\rangle\rangle\}_{sk(X_A,S)})
\end{array}$$

Semantics of scenarii

Definition (States and Transitions)

A state is a couple (T, Q) where T is a set of ground terms (intruder knowledge) and Q a bounded scenario.
Then we have a transition between states $(T, Q) \longrightarrow (T', Q')$ if there are $R \in Q$ and a substitution σ such that:

- $R = (u \longrightarrow v), R'$
- $T \vdash u\sigma \quad (dom(\sigma) = vars(u))$
- $T' = T \cup \{v\sigma\}$
- $Q' = (Q \setminus \{R\}) \cup R'\sigma$

Example

Example

Let $T = \{a, b, k_I\}$ and $Q = \{R\}$ where
$R = (\langle X, Y\rangle \longrightarrow \langle\{Y\}_k, X\rangle), (Z \longrightarrow \langle X, \langle Y, Z\rangle\rangle)$.

- $(T, Q) \longrightarrow (T \cup \{\langle\{b\}_k, a\rangle\}, \{(Z \longrightarrow \langle a, \langle b, Z\rangle\rangle)\})$
- $(T, Q) \longrightarrow (T \cup \{\langle\{\{a\}_{k_I}\}_k, a\rangle\}, \{(Z \longrightarrow \langle a, \langle\{a\}_{k_I}, Z\rangle\rangle)\}$
- $(T, Q) \not\longrightarrow (T \cup \{\langle\{\{a\}_k\}_k, a\rangle\}, \{(Z \longrightarrow \langle a, \langle\{a\}_k, Z\rangle\rangle)\})$

Example

Example

Let $T = \{a, b, k_I\}$ and $Q = \{R\}$ where
$R = (\langle X, Y\rangle \longrightarrow \langle \{Y\}_k, X\rangle), (Z \longrightarrow \langle X, \langle Y, Z\rangle\rangle)$.

- $(T, Q) \longrightarrow (T \cup \{\langle \{b\}_k, a\rangle\}, \{(Z \longrightarrow \langle a, \langle b, Z\rangle\rangle)\})$
- $(T, Q) \longrightarrow (T \cup \{\langle \{\{a\}_{k_I}\}_k, a\rangle\}, \{(Z \longrightarrow \langle a, \langle \{a\}_{k_I}, Z\rangle\rangle)\}$
- $(T, Q) \not\longrightarrow (T \cup \{\langle \{\{a\}_k\}_k, a\rangle\}, \{(Z \longrightarrow \langle a, \langle \{a\}_k, Z\rangle\rangle)\})$

Each branch has a finite depth, but possibly a infinite branching.

Preservation of the secrecy

Definition (Secrecy)

Let T_1 be a ground set of terms (Initial knowledge of the intruder). A bounded scenario Q does not preserve the secrecy of a ground term s for T_1 if there exists a state (T', Q'), such that

- $T' \vdash s$
- $(T_1, Q) \longrightarrow^* (T', Q')$

where $\longrightarrow^*$ is the reflexive and transitive closure of $\longrightarrow$.

If there does not exist a such state (T', Q') we say that Q preserves the secrecy of s for the initial intruder knowledge T_1.

Interleaving

Given a bounded scenario (a set of instantiated roles) Q, we obtain several execution traces deppending on the chosen interleaving of actions

$$I = (u_1 \longrightarrow v_1) <_E \ldots <_E (u_n \longrightarrow v_n)$$

Interleaving

Given a bounded scenario (a set of instantiated roles) Q, we obtain several execution traces deppending on the chosen interleaving of actions

$$I = (u_1 \longrightarrow v_1) <_E \ldots <_E (u_n \longrightarrow v_n)$$

Then, I does not preserve the secrecy of s, given T_1 if there exists $\sigma_1, \ldots, \sigma_n$ such that

$$(Q, T_1) \longrightarrow (Q_1, T_1 \cup \{v_1\sigma_1\}) \longrightarrow \ldots \longrightarrow (Q_n, T_1 \cup \{v_1\sigma_1, \ldots, v_n\sigma_n\})$$

and $T_1 \cup \{v_1\sigma_1, \ldots, v_n\sigma_n\} \vdash s$.

Interleaving

Given a bounded scenario (a set of instantiated roles) Q, we obtain several execution traces deppending on the chosen interleaving of actions

$$I = (u_1 \longrightarrow v_1) <_E \ldots <_E (u_n \longrightarrow v_n)$$

Then, I does not preserve the secrecy of s, given T_1 if there exists $\sigma_1, \ldots, \sigma_n$ such that

$$(Q, T_1) \longrightarrow (Q_1, T_1 \cup \{v_1\sigma_1\}) \longrightarrow \ldots \longrightarrow (Q_n, T_1 \cup \{v_1\sigma_1, \ldots, v_n\sigma_n\})$$

and $T_1 \cup \{v_1\sigma_1, \ldots, v_n\sigma_n\} \vdash s$.

A bounded scenario Q does preserve the secrecy of s if and only if, any possible interleaving I does preserve the secrecy of s.

Constraints System

Symbolic representation of execution tree by constraints system.

Definition (Constraints System)

A constraint is an expression $T \Vdash u$ where T is a set of terms and u a term.

A constraints system C is a finite set of constraints $\cup_{1 \leq i \leq n} T_i \Vdash u_i$ such that

- $T_i \subseteq T_{i+1} \qquad (1 \leq i \leq n)$
- if $T_i \Vdash u_i \in C$ and $x \in vars(T_i)$ then there exists $j < i$, such that $T_j \Vdash u_j \in C$ and $x \in vars(u_j)$

A substitution σ is a solution of C if $T\sigma \vdash u\sigma$ for all $T \Vdash u \in C$.

We denote by $\perp$ a constraints system unsatisfiable.

From Protocols to Constraints system

Let Q a bounded scenario, I the following interleaving of the actions of Q and s a secret term.

$$I = (u_1 \longrightarrow v_1) <_E (u_2 \longrightarrow v_2) <_E \ldots <_E (u_n \longrightarrow v_n)$$

From Protocols to Constraints system

Let Q a bounded scenario, I the following interleaving of the actions of Q and s a secret term.

$$I = (u_1 \longrightarrow v_1) <_E (u_2 \longrightarrow v_2) <_E \ldots <_E (u_n \longrightarrow v_n)$$

We associate C:

$$\begin{array}{rcl} T_1 & \Vdash & u_1 \\ T_2 = T_1 \cup \{v_1\} & \Vdash & u_2 \\ & \vdots & \\ T_n = T_{n-1} \cup \{v_{n-1}\} & \Vdash & u_n \\ T_{n+1} = T_n \cup \{v_n\} & \Vdash & s \end{array}$$

We can show that C has a solution iff I does not preserve the secret of the term s.

Exercises

$$\begin{array}{rl}
A \longrightarrow B: & \langle A, N_A \rangle \\
B \longrightarrow A: & \{\langle N_A, N_B \rangle\}_{K_{ab}} \\
A \longrightarrow B: & N_B \\
B \longrightarrow A: & \{\langle K, N_B \rangle\}_{K_{ab}} \\
A \longrightarrow B: & \{s\}_K
\end{array}$$

Intruder knows only identities of A and B.

- Give a role specification of this protocol.
- Give a constraint system associated to the "normal" interleaving of a single session of this protocol between a and b.

Solution(1): Role specification

$$\begin{array}{rl} A \longrightarrow B: & \langle A, N_A \rangle \\ B \longrightarrow A: & \{\langle N_A, N_B \rangle\}_{K_{ab}} \\ A \longrightarrow B: & N_B \\ B \longrightarrow A: & \{\langle K, N_B \rangle\}_{K_{ab}} \\ A \longrightarrow B: & \{s\}_K \end{array}$$

Roles:

$$\begin{array}{rll} R_A(A, B, N_A, S) = & (init \longrightarrow \langle A, N_A \rangle), & \mathbf{a1} \\ & (\{\langle N_A, X_{N_B} \rangle\}_{sk(A,B)} \longrightarrow X_{N_B}), & \mathbf{a2} \\ & (\{\langle X_K, X_{N_B} \rangle\}_{sk(A,B)} \longrightarrow \{s\}_{X_K}) & \mathbf{a3} \end{array}$$

$$\begin{array}{rll} R_B(B, N_B, K) = & (\langle X_A, X_{N_A} \rangle \longrightarrow \{\langle X_{N_A}, N_B \rangle\}_{sk(X_A,B)}) & \mathbf{b1} \\ & (N_B \longrightarrow \{\langle K, N_B \rangle\}_{sk(X_A,B)}), & \mathbf{b2} \\ & (\{X_s\}_K \longrightarrow ok) & \mathbf{b3} \end{array}$$

Solution(1): constraints system for a single session

The constraint system associated to the interleaving

$$I = a1 <_E b1 <_E a2 <_E b2 <_E a3 <_E b3$$

of $R_A(a, b, n_a, s) || R_B(b, n_b, k)$ where
$T_1 = \{a, b, i, sk(a, i), sk(b, i), init, ok\}$

$$\begin{array}{lll}
T_1 & \Vdash & init \\
T_2 = T_1 \cup \{\langle a, n_A \rangle\} & \Vdash & \langle X_A, X_{N_A} \rangle \\
T_3 = T_2 \cup \{\{\langle X_{N_A}, n_B \rangle\}_{sk(X_A,b)}\} & \Vdash & \{\langle n_A, X_{N_B} \rangle\}_{sk(a,X_B)} \\
T_4 = T_3 \cup \{X_{N_B}\} & \Vdash & n_B \\
T_5 = T_4 \cup \{\{\langle k, n_B \rangle\}_{sk(X_A,b)}\} & \Vdash & \{\langle X_K, X_{N_B} \rangle\}_{sk(a,X_B)} \\
T_6 = T_5 \cup \{\{s\}_{X_K}\} & \Vdash & ok \\
T_7 = T_6 \cup \{ok\} & \Vdash & s
\end{array}$$

Solved constraints

A constraints system C is solved if it is of the form
$T_1 \Vdash x_1, \ldots, T_n \Vdash x_n$ such that $\emptyset \neq T_1 \subseteq T_2 \subseteq \cdots \subseteq T_n$
or

$$\perp$$

Solved constraints

A constraints system C is solved if it is of the form
$T_1 \Vdash x_1, \ldots, T_n \Vdash x_n$ such that $\emptyset \neq T_1 \subseteq T_2 \subseteq \cdots \subseteq T_n$
or

$$\perp$$

Solved constraints of the first form have simple solutions: let $t \in T_1 \neq \emptyset$ and $\sigma = \{x_1 \to t, \ldots, x_n \to t\}$. Then σ is a solution of $T_1 \Vdash x_1, \ldots, T_n \Vdash x_n$.

Resolution of Constraints systems

Definition (Rules of simplification: $C \rightsquigarrow_\sigma C'$)

R_1	$C \cup \{T \Vdash u\}$	$\rightsquigarrow$	C	if $T \cup \{x \mid T' \Vdash x \in C, T' \subset T\} \vdash u$
R_2	$C \cup \{T \Vdash u\}$	$\rightsquigarrow_\sigma$	$C\sigma \cup \{T\sigma \Vdash u\sigma\}$	$\sigma = mgu(t, u), t \in st(T)$, $t \neq u$; t, u not variables
R_3	$C \cup \{T \Vdash u\}$	$\rightsquigarrow_\sigma$	$C\sigma \cup \{T\sigma \Vdash u\sigma\}$	$\sigma = mgu(t_1, t_2), t_1, t_2 \in st(T)$, $t_1 \neq t_2$; t_1, t_2 not variables
R_4	$C \cup \{T \Vdash \{u\}_v\}$	$\rightsquigarrow$	$C \cup \{T \Vdash u, T \Vdash v\}$	
R_5	$C \cup \{T \Vdash \langle u, v\rangle\}$	$\rightsquigarrow$	$C \cup \{T \Vdash u, T \Vdash v\}$	
R_6	$C \cup \{T \Vdash u\}$	$\rightsquigarrow$	$\bot$	if $T = \emptyset$ or $var(T, u) = \emptyset$ and $T \nvdash u$

Properties of simplification rules

Lemma (Preservation)

Simplification rules transform a constraints system into a constraints system.

Properties of simplification rules

Lemma (Preservation)

Simplification rules transform a constraints system into a constraints system.

Lemma (Correctness)

If $C \rightsquigarrow_{\sigma} C'$ then if θ is a solution of C', $\sigma\theta$ is also a solution of C.

Properties of simplification rules

Lemma (Preservation)

Simplification rules transform a constraints system into a constraints system.

Lemma (Correctness)

If $C \rightsquigarrow_\sigma C'$ then if θ is a solution of C', $\sigma\theta$ is also a solution of C.

Lemma (Termination)

Simplification rules always terminate: There does not exist infinite chain $C \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} C_2 \rightsquigarrow_{\sigma_3} \ldots$.

Properties of simplification rules

Lemma (Preservation)

Simplification rules transform a constraints system into a constraints system.

Lemma (Correctness)

If $C \rightsquigarrow_\sigma C'$ then if θ is a solution of C', $\sigma\theta$ is also a solution of C.

Lemma (Termination)

Simplification rules always terminate: There does not exist infinite chain $C \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} C_2 \rightsquigarrow_{\sigma_3} \ldots$.

Lemma (Completeness)

If C is a constraint system not in solved form and if σ is a solution of C then there exists θ, τ such that $C \rightsquigarrow_\theta C'$, $\sigma = \theta\tau$ and τ is a solution of C'.

Decidability

Theorem

Preservation of the secrecy for protocol with bounded number of sessions is decidable.

- Guess an interleaving and build constraints system associated.
- Using previous lemma C has a solution iff there exists C' in solved form such that $C' \neq \perp$ and $C \rightsquigarrow_{\tau} C'$
- Using termination lemma to conclude.

We also can show that the problem is in co-NP.

3-SAT Problem

Definition

Input: set of propositional variables $\{x_1, \ldots, x_n\}$ and a conjunction of clauses with 3 literals.

$$f(\vec{x}) = \bigwedge_{1 \leq i \leq l} (x_{i,1}^{\epsilon_{i,1}} \vee x_{i,2}^{\epsilon_{i,2}} \vee x_{i,3}^{\epsilon_{i,3}})$$

where $\epsilon_{i,j} \in \{+, -\}$ and $x^+ = x, x^- = \neg x$.
Question : Does exist a valuation V of $\{x_1, \ldots, x_n\}$, such that $V(f(\vec{x})) = \top$.

Theorem

3-SAT problem is NP-complete.

Theorem

Decide if a protocol P does not preserve the secrecy of a ground

NP-hardness

We build a protocol such that an intruder can deduce s iff $f(\vec{x})$ is satisfaisable.

$$g(x_{i,j}^{\epsilon_{i,j}}) = \begin{cases} x_{i,j} & \text{if } \epsilon_{i,j} = + \\ \{x_{i,j}\}_K & \text{if } \epsilon_{i,j} = - \end{cases}$$
$$\forall 1 \leq i \leq l: \; f_i(\vec{x}) = \langle g(x_{i,1}^{\epsilon_{i,1}}), \langle g(x_{i,2}^{\epsilon_{i,2}}), g(x_{i,3}^{\epsilon_{i,3}})\rangle\rangle$$

We suppose Initial intruder knowledge is $\{\bot, \top\}$.

$$\begin{array}{rl}
A: & \langle x_1, \langle \ldots, x_n\rangle\rangle \rightarrow \{\langle f_1(\vec{x}), \langle f_2(\vec{x}), \langle \ldots, \langle f_n(\vec{x}), end\rangle \ldots\rangle\rangle\}_p \\
\forall 1 \leq i \leq l: & \\
B_i: & \{\langle\langle \top, \langle x, y\rangle\rangle, z\rangle\}_p \longrightarrow \{z\}_p \\
\overline{B}_i: & \{\langle\langle \{\bot\}_K, \langle x, y\rangle\rangle, z\rangle\}_p \longrightarrow \{z\}_p \\
C_i: & \{\langle\langle x, \langle \top, y\rangle\rangle, z\rangle\}_p \longrightarrow \{z\}_p \\
\overline{C}_i: & \{\langle\langle x, \langle \{\bot\}_K, y\rangle\rangle, z\rangle\}_p \longrightarrow \{z\}_p \\
D_i: & \{\langle\langle x, \langle y, \top\rangle\rangle, z\rangle\}_p \longrightarrow \{z\}_p \\
\overline{D}_i: & \{\langle\langle x, \langle y, \{\bot\}_K\rangle\rangle, z\rangle\}_p \longrightarrow \{z\}_p \\
E: & \{end\}_p \longrightarrow s
\end{array}$$

Active intruder and unbounded number of sessions

Let as assume

- unbounded number of sessions
- adversary can intercept, modify, block, generate, insert new messages

PCP is undecidable

Definition (Post Correspondence Problem (PCP))

Let Σ be a finite alphabet.
Input : Sequence of pairs $\langle u_i, v_i \rangle_{1 \leq i \leq n}$ $u_i, v_i \in \Sigma^*$, $n \in \mathbb{N}$
Question : Existence of $k, i_1, \ldots, i_k \in \mathbb{N}$ such that $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$?

Example

u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
aba	bbb	aab	bb	a	aaa	$abab$	$babba$

Solution: **1431**

$u_1 \cdot u_4 \cdot u_3 \cdot u_1 = aba \cdot bb \cdot aab \cdot aba = a \cdot babba \cdot abab \cdot a = v_1 \cdot v_4 \cdot v_3 \cdot v_1$

But no solution for $\langle \mathbf{u_1}, \mathbf{v_1} \rangle, \langle \mathbf{u_2}, \mathbf{v_2} \rangle, \langle \mathbf{u_3}, \mathbf{v_3} \rangle$

Undecidability for Protocols

We construct a protocol such that decidability of secret implies decidability of PCP.

$$A: \quad (init \longrightarrow \{\langle u_1, v_1\rangle\}_{K_{AB}} \ldots \{\langle u_n, v_n\rangle\}_{K_{AB}})$$

$$\begin{aligned} B: \quad & (\{\langle x, y\rangle\}_{K_{AB}} \longrightarrow \\ & \langle\{\langle x\cdot u_1, y\cdot v_1\rangle\}_{K_{AB}}, \{s\}_{\langle\{\langle x\cdot u_1, x\cdot u_1\rangle\}_{K_{AB}}\rangle} \\ & \ldots \\ & \langle\{\langle x\cdot u_n, y\cdot v_n\rangle\}_{K_{AB}}, \{s\}_{\langle\{\langle x\cdot u_n, x\cdot u_n\rangle\}_{K_{AB}}\rangle}) \end{aligned}$$

We assume that $\mathbf{K_{AB}}$ is a shared key between **A** and **B**.
Intruder can find s if and only if she can solve PCP.

Proofs of protocols : the symbolic model

- The **Dolev-Yao model** was proposed in 1983:
  - messages are terms in a free algebra: (m_1, m_2), $\{m\}_k$, $\mathbf{[}m\mathbf{]}_k$, $h(m)$...
  - no secret value (nonce or key) can be guessed
  - the intruder can only apply functions in the given signature
  - ... but has complete control of the network
  - choice in semantics : non - deterministic
- One can add equations between primitives (functions), but anyway, we assume that the only equalities are those given by these equations.
- Proofs in this model are simpler and can be automated.

Proofs of protocols : the computational model

- The **computational model** was proposed in the '80s:
  - the messages are *bitstrings*
  - the cryptographic primitives (encryption, decryption, key-generation, signing..) are *probabilistic polynomial-time algorithms on bitstrings*
  - secrets can be guessed ... with *very small* probability
  - the intruder is any *interactive probabilistic polynomial-time algorithme*
  - has complete control of the network
- The model is much closer to reality, but proofs in this model are mostly manual.

Proofs

- Computational world:
  Done by hand, long and often complex and hard to understand. (Sometime proofs are error-prone). Often based on reduction to cryptographic assumptions.
- Symbolic world:
  Based on constraint solving, First-order Logic, Tree automata etc ... these proofs are automatic.

Exemple: asymmetric encryption

In the symbolic world:

$$\textbf{(AD)}\quad \frac{T_0 \vdash \{u\}_{pk(v)} \qquad T_0 \vdash sk(v)}{T_0 \vdash u} \qquad\qquad \textbf{(AC)}\quad \frac{T_0 \vdash u \qquad T_0 \vdash pk(v)}{T_0 \vdash \{u\}_{pk(v)}}$$

In the real world: ElGamal Encryption:

- a cyclic group G of order q and generator g (g and q deppend on some integer η - the security parameter)
- $\mathcal{K}_\eta() = x \leftarrow^R \mathbb{Z}_q; sk \leftarrow x; pk \leftarrow g^x; return(sk, pk)$
- $\{m\}_{pk} = r \leftarrow^R \mathbb{Z}_q; return(pk^r \times g^m, g^r)$.

But what can we assume about this primitive?

Back to the symbolic world

Messages are defined by the following grammar

$$
\begin{aligned}
\text{Keys} &= \{k, k_1, k_2, \ldots,\} \\
\text{Coins} &= \{r, s, r_1, r_2, \ldots,\} \\
\text{Bits} &= \{0, 1\} \\
e &::= k \in \text{Keys} \\
&\mid b \in \text{Bits} \\
&\mid (e_1, e_2) \\
&\mid \{e\}_k^r
\end{aligned}
$$

- If $\{e\}_k^r$ and $\{e'\}_{k'}^r$ both occur in the same context, then $e = e'$ and $k = k'$.

Recall: Dolev-Yao Deduction System

We denote $T_0 \vdash e$ for "message e is deducible from the set of messages T_0".

Deduction System for Dolev Yao theory: $T_0 \vdash^{?} s$

$$\text{(C0)}\quad \frac{}{T_0 \vdash 0} \qquad\qquad \text{(C1)}\quad \frac{}{T_0 \vdash 1}$$

$$\text{(A)}\quad \frac{u \in T_0}{T_0 \vdash u} \qquad\qquad \text{(UL)}\quad \frac{T_0 \vdash (u,v)}{T_0 \vdash u}$$

$$\text{(P)}\quad \frac{T_0 \vdash u \qquad T_0 \vdash v}{T_0 \vdash (u,v)} \qquad\qquad \text{(UR)}\quad \frac{T_0 \vdash (u,v)}{T_0 \vdash v}$$

$$\text{(C)}\quad \frac{T_0 \vdash u \qquad T_0 \vdash v}{T_0 \vdash \{u\}_v^r} \qquad\qquad \text{(D)}\quad \frac{T_0 \vdash \{u\}_v^r \qquad T_0 \vdash v}{T_0 \vdash u}$$

Symbolic world: meaning of a message

- Now we want do define an equivalence $\sim$ between messages, such that $e_1 \sim e_2$ when e_1 and e_2 *"look the same"*. Intuitevely, by doing any permitted operation, we cannot distinguish e_1 and e_2.
- Examples:
  - $\{0\}_v^r \sim \{1\}_v^{r'}$.
  - $(\{0\}_v^r, v) \not\sim (\{1\}_v^{r'}, v)$.
  - $(\{0\}_v^r, (\{1\}_v^s) \sim (\{1\}_v^{r'}, \{1\}_u^{s'})$.
- How can we define formally $\sim$?

Key Recovery **rec** function

- We define $\mathbf{rec}(E)$ as being the set of keys that can be recovered from E using information available in E only: $\mathbf{rec}(E) = \{k \in \text{Keys} | E \vdash k\}$.
- Example:

$$\mathbf{rec}(((\{\{k_2\}^{r}_{k_1}\}^{r'}_{k_1}, \{(k_3, 0)\}^{r''}_{k_2}), k_1)) = \{k_1, k_2, k_3\}$$

- But we want a constructive definition of $\mathbf{rec}(E)$!

Key Recovery Function in one pass $\mathbf{F}_{kr}(E, K)$

- Following the approach of [Micciancio,Warinschi], for an expression E and a set K of keys, we define Key Recovery Function $\mathbf{F}_{kr}(E, K)$ as follows:

$$\begin{aligned}
\mathbf{F}_{kr}(b, K) &= K \\
\mathbf{F}_{kr}(k, K) &= \{k\} \cup K \\
\mathbf{F}_{kr}((E_1, E_2), K) &= \mathbf{F}_{kr}(E_1, K) \cup \mathbf{F}_{kr}(E_2, K) \\
\mathbf{F}_{kr}(\{E\}_k, K) &= \begin{cases} K & \text{if } k \notin K \\ \mathbf{F}_{kr}(E, K) & \text{otherwise.} \end{cases}
\end{aligned}$$

Example

$$E = ((\{\{k_2\}_{k_1}^{r}\}_{k_1}^{r'}, \{(k_3, 0)\}_{k_2}^{r"}), k_1)$$

1. $\mathbf{F}_{kr}(E, \emptyset) = \{k_1\}$
2. $\mathbf{F}_{kr}(E, \{k_1\}) = \{k_1, k_2\}$
3. $\mathbf{F}_{kr}(E, \{k_1, k_2\}) = \{k_1, k_2, k_3\}$
4. $\mathbf{F}_{kr}(E, \{k_1, k_2, k_3\}) = \{k_1, k_2, k_3\}$

Inductive Definition of Key Recovery **rec**

- Let E be an expression, then we define **rec** by $\mathbf{rec}(E) = \bigcup_i G_i(E) = G_{|E|}(E)$ where

$$\begin{aligned} G_0(E) &= \emptyset \\ G_i(E) &= \mathbf{F}_{kr}(E, G_{i-1}(E)). \end{aligned}$$

Example

$E = ((\{\{k_2\}_{k_1}^{r}\}_{k_1}^{r'}, \{(k_3, 0)\}_{k_2}^{r''}), k_1)$

1. $G_1(E) = \mathbf{F}_{kr}(E, \emptyset) = \{k_1\}$
2. $G_2(E) = \mathbf{F}_{kr}(E, \{k_1\}) = \{k_1, k_2\}$
3. $G_3(E) = \mathbf{F}_{kr}(E, \{k_1, k_2\}) = \{k_1, k_2, k_3\}$
4. $G_4(E) = \mathbf{F}_{kr}(E, \{k_1, k_2, k_3\}) = \{k_1, k_2, k_3\} = G_3(E) = \mathbf{rec}(E)$

Patterns

- *Patterns* are defined by the following grammar

$$
\begin{aligned}
\text{Keys} &= \{k, k_1, k_2, \dots,\} \\
\text{Coins} &= \{r, s, r_1, r_2, \dots,\} \\
\text{Bits} &= \{0, 1\} \\
P &::= k \in \text{Keys} \\
&\mid b \in \text{Bits} \\
&\mid (P_1, P_2) \\
&\mid \{P\}_k^r \\
&\mid \square^r
\end{aligned}
$$

- Remark: a message is a pattern with no occurrence of $\square$
- e.g., $\{(\{\square^r\}_{k_2}^s, 1)\}_{k_1}^{s'}$ is a pattern

The pattern of an expression

- We denote by **keys**(E) the set of all keys occurring in E, and let **hidden**$(E) =$ **keys**$(E) -$ **rec**(E). Define $\textsc{Pat}(E, K)$ as follows:

$$\begin{aligned}
\textsc{Pat}(b, K) &= b \\
\textsc{Pat}(k, K) &= k \\
\textsc{Pat}((E_1, E_2), K) &= (\textsc{Pat}(E_1, K), \textsc{Pat}(E_2, K)) \\
\textsc{Pat}(\{E\}_k^r, K) &= \begin{cases} \{\textsc{Pat}(E, K)\}_k^r & \text{if } k \in K \\ \Box^r & \text{otherwise} \end{cases}
\end{aligned}$$

Finally, we define **pat** by $\mathbf{pat}(E) = \textsc{Pat}(E, \mathbf{rec}(E))$. Intuitively, **pat**$(E)$ is the **pattern** corresponding to E, given the knowledge of any keys that may be recovered from E.

Examples

$$\mathbf{pat}(E) = \textsc{Pat}(E, \mathbf{rec}(E))$$

$$E = ((\{\{k_2\}_{k_1}^{r}\}_{k_1}^{r'}, \{k_3\}_{k_2}^{r''}), k_1)$$

$$\mathbf{rec}(E) = \{k_1, k_2, k_3\}$$

$$\mathbf{pat}(E) = E$$

$$E = ((\{\{k_2\}_{k_1}^{r}\}_{k_1}^{r'}, \{k_3\}_{k_2}^{r''}), k_2)$$

$$\mathbf{rec}(E) = \{k_2, k_3\}$$

$$\mathbf{pat}(E) = (\Box^{r'}, \{k_3\}_{k_2}^{r''}, k_2)$$

Pattern - exercise

Example (Exercise)

$$\begin{aligned} pat((\mathbf{0},\mathbf{1})) &= ? \\ pat((\{\mathbf{0}\}^{r}_{K_1},(\{\mathbf{1}\}^{r'}_{K_2},K_1))) &= ? \\ pat((\{\{K_1\}^{r'}_{K_2}\}^{r''}_{K_3},K_3)) &= ? \end{aligned}$$

Pattern Equivalence

- We define E to be **symbolically equivalent** to F (and we write $E \sim F$) if there is a renaming (a bijection) σ_K of keys of F and a renaming σ_R of random coins of F such that $\mathbf{pat}(E) = \mathbf{pat}(F\sigma_K\sigma_R)$

Examples:

- $k' \sim k$
- $(k_1, k_2) \not\sim (k_3, k_3)$ (no bijection !)
- but $(k_1, k_2) \sim (k_4, k_1)$
- $(\{0\}_k^r, \{0\}_k^{r'}) \sim (\{0\}_k^{r'}, \{1\}_k^s)$
- but $(\{0\}_k^r, \{0\}_k^r) \not\sim (\{0\}_k^{r'}, \{1\}_k^s)$
- $\{((1,1),(1,1))\}_k^r \sim \{1\}_k^r$
- $(\{0\}_k^{r'}, \{0\}_k^s) \sim (\{0\}_{k'}^{r'}, \{0\}_{k''}^s)$

Terms and the computational world

What is the meaning of a term in the computational world?

- 0, 1 ? … are constants, hence deterministic
- $((0, 1), 1)$ … using some deterministic concatenation
- k … keys are randomly sampled
- r … coins are randomly sampled
- $\{0\}_k^r$ … in general, encryption is probabilistic

Hence to a term, we will associate a **distribution**.

Implementation - primitives

- Let $\langle u, v\rangle : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^*$ be an easily computable and invertible injective function.
- Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme:
  - $\mathcal{K}(1^\eta)$: generates random key.
  - $\mathcal{E}^r(1^\eta, k, x)$: encrypts x with k using random coins r.
  - $\mathcal{D}(1^\eta, k, y)$: decrypts y with k.
  - Correctness:

$$\forall \eta, \forall x, \forall r$$

$$k := \mathcal{K}(1^\eta); y := \mathcal{E}^r(1^\eta, k, x); x' := \mathcal{D}(1^\eta, k, y) :$$

$$(x = x')?$$

Implementation of formal terms

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $\eta \in \mathbb{N}$ be the security parameter. We associate to the formal term M a distribution $[\![M]\!]_\eta$ and by consequence a family of distributions $[\![M]\!]$.

- For each $k \in Keys(M)$ do $\tau_K(k) \overset{R}{\leftarrow} \mathcal{K}_\eta()$.
- For each $r \in Coins(M)$ do $\tau_R(r) \overset{R}{\leftarrow} \{0,1\}^\omega$.
- $[\![k]\!]_\eta = \tau_K(k)$.
- For $b \in \{0,1\}$, $[\![b]\!]_\eta = b$.
- $[\![(e_1, e_2)]\!]_\eta = \langle [\![e_1]\!]_\eta, [\![e_2]\!]_\eta \rangle$.
- $[\![\{e\}_k^r]\!]_\eta = \mathcal{E}^{\tau_R(r)}(1^\eta, \tau_K(k), [\![e]\!]_\eta)$.

Computational equivalence of terms

We say that two terms E and F are **computationally equivalent**, and we write $E \approx F$, when the associated distributions are computationally indistinguishable

$$[\![E]\!] \approx [\![F]\!]$$

Corectness

Do we have that $E \sim F$ implies that $E \approx F$?
YES, if we avoid key cycle in terms, and if encryption is *type-0*-secure.

Encryption Cycles - examples:

- $\{k\}_k^r$
- $\{k_2\}_{k_1}^r, \{k_1\}_{k_2}^{r'}$
- $\{k_2\}_{k_1}^r, \{k_3\}_{k_2}^{r'}, \{k_1\}_{k_3}^{r''}$
- etc ...

Definition

Definition (Key Cycles)

Let $k, k' \in \text{Keys}$. We say that k encrypts k' in the term M, denoted by $k \succ_M k'$, if $\{N\}_k^r \in st(M)$ and $k' \in st(N)$.
A term M is acyclic iff the relation $\succ_M$ is acyclic.

Example

.

- Let $M = (\{\{k_1\}_{k_2}^{r1}\}_{k_3}^{r2}, \mathbf{0})$. $\succ_M$ is defined by $k_3 \succ_M k_2 \succ_M k_1$. M is acyclic.
- Let $M = \{k\}_k^r$. We get $k \succ_M k$. M has a cycle of size 1.
- Let $M = (\{k_1\}_{k_2}^{r1}, \{k_2\}_{k_1}^{r2})$. We have that $k_1 \succ_M k_2$ and $k_2 \succ_M k_1$. M has a cycle of size 2.

Type-0 Scheme - intuition

- A *type-0* encryption scheme has the following properties:
  1. message-hiding: given two messages m and m' of equal length and a cipher-text c of one of them, one cannot tell which of m or m' corresponds to c (IND-CPA)
  2. Which-key-hiding: given cipher-texts c, c', one cannot tell if they were encrypted under the same key
  3. Message-length-hiding: given cipher-text c, one cannot determine the length of the corresponding plain-text

Type-0 Scheme - formal definition

Definition

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and $\mathcal{A}$ be an adversary:

$$Adv^0_{\Pi,\eta}(\mathcal{A}) = Pr[\mathcal{A}^{\mathcal{O}^0_1(.),\mathcal{O}^0_2(.)}(\eta) = 1 | k \leftarrow^R \mathcal{K}_\eta()]$$

$$- Pr[\mathcal{A}^{\mathcal{O}^1_1(.),\mathcal{O}^1_2(.)}(\eta) = 1 | k_1, k_2 \leftarrow^R \mathcal{K}_\eta()]$$

Π is **Type-0 secure** if $Adv^0_{\Pi,\eta}(\mathcal{A})$ is negligible, for any ppt $\mathcal{A}$.

- $\mathcal{A}^{\mathcal{O}}$ is an adversary having acces to a (or several) oracle(s) $\mathcal{O}$.
- $\mathcal{O}^1_1(.)$ is an oracle that takes a message m and answers $c \leftarrow^R \mathcal{E}_{k_1}(m)$.
- $\mathcal{O}^1_2(.)$ is an oracle that takes a message m and answers $c \leftarrow^R \mathcal{E}_{k_2}(m)$.
- $\mathcal{O}^0_1(.)$ is an oracle that takes a message m and answers $c \leftarrow^R \mathcal{E}_{k}(0)$.
- $\mathcal{O}^0_2(.)$ is an oracle that takes a message m and answers $c \leftarrow^R \mathcal{E}_{k}(0)$.

Soundness

Theorem

Soundness If e_1,e_2 do not contain encryption cycles, and if encryption is implemented using a type-0 secure scheme, then

$$e_1 \sim e_2 \quad \textit{implies} \quad e_1 \approx e_2$$

Proof is by a hybrid reduction-style argument.

From now on we suppose that expressions do not contain encryption cycles, and that encryption is implemented using a type-0 secure scheme.

Implementation of formal patterns

- For each $k \in Keys(M)$ do $\tau_K(k) \overset{R}{\leftarrow} \mathcal{K}_\eta()$.
- $k_\Box \overset{R}{\leftarrow} \mathcal{K}_\eta()$.
- For each $r \in Coins(M)$ do $\tau_R(r) \overset{R}{\leftarrow} \{0,1\}^\omega$.
- $[\![k]\!]_\eta = \tau_K(k)$.
- For $b \in \{0,1\}$, $[\![b]\!]_\eta = b$.
- $[\![(e_1, e_2)]\!]_\eta = \langle [\![e_1]\!]_\eta, [\![e_2]\!]_\eta \rangle$.
- $[\![\{e\}_k^r]\!]_\eta = \mathcal{E}^{\tau_R(r)}(1^\eta, \tau_K(k), [\![e]\!]_\eta)$.
- $[\![\Box^r]\!]_\eta = \mathcal{E}^{\tau_R(r)}(1^\eta, k_\Box, 0)$.

Replacing one key

For a key $k_0 \in \text{Keys}$, define

- $\mathit{replace}(k, k_0) = k$.
- $\mathit{replace}(b, k_0) = b$.
- $\mathit{replace}((e_1, e_2), k_0) = (\mathit{replace}(e_1, k_0), \mathit{replace}(e_2, k_0))$.
- $\mathit{replace}(\{e\}_k^r, k_0) = \{\mathit{replace}(e, k_0)\}_k^r$ if $k \neq k_0$.
- $\mathit{replace}(\{e\}_{k_0}^r, k_0) = \Box^r$.
- $\mathit{replace}(\Box^r, k_0) = \Box^r$.

Replacing one key is sound

Theorem

Soundness Let e an expression, and k_0 a key occuring in e only as encryption key. Then $[\![e]\!] \approx [\![replace(e, k_0)]\!]$.

Proof.

Assume that $\mathcal{B}$ distinguishes $[\![e]\!]$ from $[\![replace(e, k_0)]\!]$, and let construct an adversary $\mathcal{A}^{\mathcal{O}}$ that wins against the security of the type-0 encryption scheme. $\mathcal{A}^{\mathcal{O}}(1^\eta)$ works as follows:

- For each $k \in Keys(e)$ do $\tau_K(k) \overset{R}{\leftarrow} \mathcal{K}_\eta()$.
- For each $r \in Coins(e)$ do $\tau_R(r) \overset{R}{\leftarrow} \{0,1\}^\omega$.
- Let $L = \{\}$ (the empty mapping).
- Compute the "semantics" v of e by invoking $\mathbf{S}em^O(e)$.
- Return $\mathcal{B}(1^\eta, v)$.

Replacing one key is sound

Theorem

Let e an expression, and k_0 a key occuring in e only as encryption key. Then $[\![e]\!] \approx [\![replace(e, k_0)]\!]$.

Proof.

(Continuation)

$\mathbf{S}em^{O}(e)$ is: case e of

- k: return $\tau_K(k)$ (note that $k \neq k_0$).
- b: return b.
- (e_1, e_2): Let $v_i = \mathbf{S}em^{O}(e_i)$; return $\langle v_1, v_2 \rangle$.
- $\square^r$: return $\mathcal{O}_2(0)$.
- $\{e\}_k^r$: Let $w = \mathbf{S}em^{O}(e)$;
  - If $k \neq k_0$, then return $\mathcal{E}^{\tau_R(r)}(1^\eta, \tau_K(k), w)$.
  - If $k = k_0$ and $L(r)$ is not defined, then $L(r) = \mathcal{O}_1(w)$; return $L(r)$.

One can see that $\mathbf{S}em^{\mathcal{O}^1}(e) = [\![e]\!]$ and $\mathbf{S}em^{\mathcal{O}^0}(e) = [\![replace(e, k_0)]\!]$, hence $\mathcal{A}$ distinguishes $\mathcal{O}^0$ from $\mathcal{O}^1$, with the same advantage that $\mathcal{B}$ distinguishes $[\![e]\!]$ from $[\![replace(e, k_0)]\!]$.

□

Soundness

Theorem

If e_1,e_2 do not contain encryption cycles, and if encryption is implemented using a type-0 secure scheme, then

$$e_1 \sim e_2 \quad \textit{implies} \quad e_1 \approx e_2$$

Proof.

- If $\{k_1, \ldots, k_n\} = Keys(e)$, and for all i, j, $i < j \Longrightarrow k_j \not\succ_e k_i$, then
$$replace(\cdots replace(replace(e, k_1), k_2) \cdots, k_n) = \mathbf{pat}(e)$$
- Apply the previous lemma to each key in **hidden**(e), we get
$$[\![e]\!] \approx [\![\mathbf{pat}(e)]\!]$$
- Permuting the formal keys and coins does not change the generated probability distribution over bit-strings. If $e_1 \sim e_2$, then $\mathbf{pat}(e_1) = \mathbf{pat}(e_2\sigma_K\sigma_R)$ for some permutations σ_k on keys and σ_R on coins, and by transitivity $[\![e_1]\!] \approx [\![\mathbf{pat}(e_1)]\!] = [\![\mathbf{pat}(e_2\sigma_K\sigma_R)]\!] = [\![\mathbf{pat}(e_2)]\!] \approx [\![e_2]\!]$,

hence $e_1 \approx e_2$. $\square$

Example

$$\llbracket((\{k_4, 0\}^{r_1}_{k_3}, \{k_3\}^{r_2}_{k_2}), (\{\{11\}^{r_4}_{k_4}\}^{r_3}_{k_1}, k_1))\rrbracket$$

$$\approx$$

$$\llbracket((\{k_4, 0\}^{r_1}_{k_3}, \Box^{r_2}), (\{\{11\}^{r_4}_{k_4}\}^{r_3}_{k_1}, k_1))\rrbracket$$

$$\approx$$

$$\llbracket((\Box^{r_1}, \Box^{r_2}), (\{\{11\}^{r_4}_{k_4}\}^{r_3}_{k_1}, k_1))\rrbracket$$

$$\approx$$

$$\llbracket((\Box^{r_1}, \Box^{r_2}), (\{\Box^{r_4}\}^{r_3}_{k_1}, k_1))\rrbracket$$

$$\approx$$

$$\llbracket((\{1\}^{r_1}_{k_2}, \Box^{r_2}), (\{\{0\}^{r_4}_{k_2}\}^{r_3}_{k_1}, k_1))\rrbracket$$

$$\approx$$

$$\llbracket((\{1\}^{r_1}_{k_2}, \{k_2\}^{r_2}_{k_3}), (\{\{0\}^{r_4}_{k_2}\}^{r_3}_{k_1}, k_1))\rrbracket$$

Abadi Rogaway 2000 Paper

- Introduced in [AR 00] M. Abadi et P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology, 15(2):103–127, 2002.
- Models security of encryption against a *passive* computational adversary
- Adversary observes two *expressions* which have the same *pattern* assuming secure encryption – goal is to distinguish them with non-negligible probability

RESULT:
Symbolic indistinguishability implies computational security

(Some) Extensions

- [Micciancio, Warinschi] give completeness for modified (authenticated) encryption, and also soundness for active adversaries
- [Micciancio, Panjwani] give a soundness theorem for an adversary which can adaptively attack the encryption scheme
- [Backes, Pfitzmann, Waidner] give an cryptographic implementation of Dolev-Yao terms in a general (UC) setting
- [Canneti, Herzog] give a formal (automated) approach to universal composability