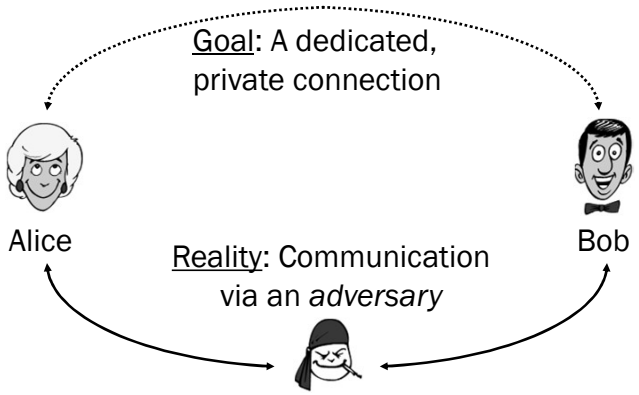
Title Goes Here

An Introduction to Modern Cryptography

Mike Reiter

Cryptography

- **Study of techniques to communicate securely in the presence of an *adversary***
- **Traditional scenario**



Adversary's Goals

1. **Observe what Alice and Bob are communicating**
   - Attacks on "confidentiality" or "secrecy"
2. **Observe that Alice and Bob are communicating, or how much they are communicating**
   - Called "traffic analysis"
3. **Modify communication between Alice and Bob**
   - Attacks on "integrity"
4. **Impersonate Alice to Bob, or vice versa**
5. **Deny Alice and Bob from communicating**
   - Called "denial of service"

- **Cryptography traditionally focuses on preventing (1) and detecting (3) and (4)**

3

Adversary's Goals in Perspective

- **Detecting modification and impersonation attacks is determining who could have *sent* a communication *s***
  - In terms of previous lectures: Bob receives *s* on a channel *C* (i.e., *C* **says** *s*), and must determine if *Alice* **says** *s*
- **Preventing attacks on confidentiality is limiting who can possibly *receive* a communication *s***
  - Not utilized in previous lectures
- **We will cover these topics in this order, unlike most treatments of cryptography**
  - First one builds on what we've already covered
  - Reordering emphasizes independence of two types of goals

4

Who is the Adversary?

- **For our study, we don't really care *who* the attacker is, but we do care about his *resources***
  - The adversary's computational power
  - The resources in the adversary's environment at his disposal
- **Alice and Bob are *resources* for the adversary**
  - How the adversary can interact with them is a core component of the resources available to him
- **Modern cryptography is based on exploiting a gap between**
  - Efficient algorithms for Alice and Bob
  - Computationally infeasible algorithms for the adversary to achieve his goals

5

Computational Resources

- **Alice, Bob and the adversary are (usually) modeled as probabilistic polynomial-time (PPT) Turing machines**
  - For some fixed polynomial p, machine halts in $p(|x|)$ steps on input x
  - Machine can "flip coins", i.e., select from a set of possible transitions randomly
- **Cryptographic algorithms typically specified in terms of a *security parameter* λ that specifies the length of inputs**
- **So, we want cryptographic algorithms such that**
  - Alice and Bob can compute efficiently, i.e., in time $p(\lambda)$ for some polynomial p
  - No (PPT) adversary can defeat with more than "negligible probability" for sufficiently large λ

6

Negligible Probability

- A function $\nu: \mathbb{N} \to \mathbb{R}$ is *negligible* if for every positive polynomial p there exists an N such that for all $\lambda > N$:

$$\nu(\lambda) < \frac{1}{p(\lambda)}$$

- Examples
  - $\nu(\lambda) = 2^{-\sqrt{\lambda}}$ is negligible
  - $\nu(\lambda) = \lambda^{-2}$ is not
- Any event that occurs with negligible probability would still occur with negligible probability if the experiment were repeated polynomially many times (in λ)

One-Way Functions

- Called "preimage resistant" in previous lectures
- A *collection* of one-way functions is a set

$$\{f_i : \text{Domain}_i \to \text{Range}_i\}_{i \in I}$$

such that for every PPT A there is a negligible ν_A where

$$\Pr[\ f_i(z) = y :\ i \leftarrow_R I \cap \{0,1\}^\lambda;\ x \leftarrow_R \text{Domain}_i;\ y \leftarrow f_i(x);\ z \leftarrow A(i,y)\ ] \leq \nu_A(\lambda)$$

for all λ large enough.

A Candidate One-Way Function

- Candidate one-way function (collection):

$$f_{g,p}(x) = g^x \bmod p$$

where

p is a prime number

g is a "generator" of $Z_p^* = \{1, 2, \ldots p\text{-}1\}$,

i.e., $\{g^1 \bmod p, g^2 \bmod p, \ldots\} = Z_p^*$

- This one-way function underlies numerous important cryptographic algorithms
  - Notably Diffie-Hellman and ElGamal

Trapdoor One-Way Functions

- A collection of trapdoor one-way functions is a set

$$\{f_i\}_{i \in I}$$

that is one-way, but for which there is an efficient algorithm B and trapdoor t_i for each $i \in I$ such that

$$x \leftarrow B(i, t_i, f_i(x))$$

- Intuition: Trapdoor t_i permits f_i to be inverted efficiently (i.e., in time polynomial in λ), but otherwise $\{f_i\}_{i \in I}$ is one-way

A Candidate Trapdoor One-Way Function

- **Candidate one-way function (collection):**

$$f_{n,e}(x) = x^e \bmod n$$

where

$n = pq$ where p, q are primes with $|p| = |q|$

$\gcd(e, (p-1)(q-1)) = 1$

and the trapdoor for $<n,e>$ is $<p,q>$, so that

Algorithm $B(<n,e>, <p,q>, y)$:

return $y^d \bmod n$ where $ed \bmod (p-1)(q-1) = 1$

- **Why does it work?**

$y^d \bmod n = x^{ed} \bmod n = x^{ed \bmod (p-1)(q-1)} \bmod n = x^1 \bmod n = x$

- **This is the famous "RSA" trapdoor function**

Why "Candidate"?

- **Because there is no proof that these functions are one-way**
  - They just seem to be
- **Best known algorithm for**
  - inverting $f_{g,p}$ runs in expected time proportional to

$$e^{\sqrt{2(\ln p)(\ln\ln p)}}$$

  - inverting $f_{n,e}$ (without the trapdoor) runs in expected time proportional to

$$e^{1.9(\ln n)^{1/3}(\ln\ln n)^{2/3}}$$

- **In fact, there is no proof that one-way functions exist!**
  - Though it is widely believed that they do

Applications

- **We have already seen applications for one-way functions**
  - To make a public identifier for private information
- **What about applications for trapdoor functions?**
- **One application is a digital signature**
  - Let K be i (the index of f_i) and K^{-1} be the trapdoor t_i
  - To sign a message x, create $\sigma = f^{-1}(x)$ using the trapdoor
  - Signer verifies signature by checking that $f_i(\sigma) = x$
  - (This is just for illustration. This is not a secure signature scheme.)

Practice

- **Security has been defined for λ "large enough"**
- **In practice, λ must be chosen**
  - How big should p or n be when used in practice?
- **It depends on numerous factors, including**
  - Life span: How long the information must be protected
  - Security margin: Computational and financial power of the attacker
  - Cryptanalysis: Algorithmic progress during lifetime of information

An Analysis for Commercial Systems (1)

[Lenstra & Verheul 1999]

Based their analysis on four hypotheses

1. **5×10^5 MIPS Years (MY) was an adequate security margin for commercial applications up to 1982**
   - 1 MY = one year of computation on a VAX 11/780 ≈ 20 hours on a 450 MHz P-II
   - 5×10^5 MY ≈ 14,000 months on a 450 MHz P-II ≈ 2 months on 7000 such processors
   - This number was derived from the assumption that the Data Encryption Standard was sufficient for commercial apps in 1982
2. **The amount of computing power and RAM one gets for a dollar doubles every 18 months.**
   - A slight variation of Moore's law
   - One expects $2^{10 \times (12/18)} \approx 100$ times more power and RAM for the same cost every 10 years.

15

An Analysis for Commercial Systems (2)

[Lenstra & Verheul 1999]

3. **The budgets of organizations (i.e., attackers) doubles every 10 years.**
   - Derived from the trend that the U.S. Gross National Product doubles every ten years (measured in contemporary dollars).

- **Illustration of combining hypotheses 1–3**
  - If 5×10^5 MY was infeasible to break in 1982
  - … then $100 \times 2 \times (5 \times 10^5 \text{ MY}) = 10^8$ MY infeasible in 1992
  - … then $100 \times 2 \times (10^8 \text{ MY}) = 2 \times 10^{10}$ MY infeasible in 2002
  - … then $100 \times 2 \times (2 \times 10^{10} \text{ MY}) = 4 \times 10^{12}$ MY infeasible in 2012

4. **The computational effort required to invert $f_{p,g}$ or $f_{n,e}$ halves every 18 months.**
   - Consistent with cryptanalytic progress from 1970 to 1999.

16

Lenstra & Verheul [1999] Recommendations

Year	$\|n\|$ or $\|p\|$	Year	$\|n\|$ or $\|p\|$	Year	$\|n\|$ or $\|p\|$	Year	$\|n\|$ or $\|p\|$
1982	417	2006	1191	2016	1664	2026	2236
1985	488	2007	1235	2017	1717	2027	2299
1990	622	2008	1279	2018	1771	2028	2362
1995	777	2009	1323	2019	1825	2029	2427
2000	952	2010	1369	2020	1881	2030	2493
2001	990	2011	1416	2021	1937	2031	2560
2002	1028	2012	1464	2022	1995	2032	2629
2003	1068	2013	1513	2023	2054	2033	2698
2004	1108	2014	1562	2024	2113	2034	2768
2005	1149	2015	1613	2025	2174	2035	2840

17

"Oracles" in the Adversary's Environment

- **The adversary does not work in a vacuum**
  - Trivial example: Adversary may be able to sign a message as Bob by tricking Bob into signing it for him
- **The environment of the adversary can be augmented with oracles that compute certain functions for the adversary**
  - Formally, a PPT adversary is augmented with new query and response tapes for each oracle
  - Notation: if A is an adversary, then A^f is an adversary with access to an oracle for function f

18

Recall Informal Definition of a Digital Signature

- **A digital signature scheme is a triple $\langle G, S, V \rangle$ of efficiently computable algorithms**
  - G outputs a "public key" K and a "private key" K^{-1}

    $$\langle K, K^{-1} \rangle \leftarrow G(\cdot)$$
  - S takes a "message" m and K^{-1} as input and outputs a "signature" σ

    $$\sigma \leftarrow S_{K^{-1}}(m)$$
  - V takes a message m, signature σ and public key K as input, and outputs a bit b

    $$b \leftarrow V_K(m, \sigma)$$
  - If $\sigma \leftarrow S_{K^{-1}}(m)$ then $V_K(m, \sigma)$ outputs 1 ("valid")
  - Given only K and message/signature pairs $\{\langle m_i, S_{K^{-1}}(m_i)\rangle\}_i$, it is computationally infeasible to compute $\langle m, \sigma \rangle$ such that

    $$V_K(m, \sigma) = 1$$

    any new $m \neq m_i$

Attacks Against Digital Signature Schemes

- **Types of attacks**
  - Key-Only Attack: Adversary knows only K
  - Known Signature Attack: The adversary knows K and has seen $\langle m, \sigma \rangle$ pairs made by $S_{K^{-1}}$, *but not chosen by the adversary*
  - Chosen Message Attack: The adversary knows K and is given an oracle for $S_{K^{-1}}$
- **When does the adversary succeed?**
  - Existential Forgery: Adversary succeeds in forging the signature of one message, not necessarily of his choice.
  - Selective Forgery: The adversary succeeds in forging the signature of some message of his choice.
  - Universal Forgery: The adversary is able to forge the signature of any message.
  - Total Break: The adversary computes the signer's secret key.

Example Definition

- A signature scheme secure against *existential forgery* under *chosen message attack* is a triple

$$\langle G, S, V \rangle$$

such that for every PPT A there is a negligible ν_A where

$$\Pr[\ V_K(m, \sigma) = 1 : \quad \langle K, K^{-1} \rangle \leftarrow G(1^\lambda); \quad \langle m, \sigma \rangle \leftarrow A^{S_{K^{-1}}}(K); \quad S_{K^{-1}}(m) \text{ not queried }] \ \leq \ \nu_A(\lambda)$$

for all λ large enough.

"Vanilla RSA" Signature Scheme

Algorithm $G(1^\lambda)$:
$p, q \leftarrow_R \{\lambda/2\text{-bit primes}\}$
$n \leftarrow pq$
Choose e: $\gcd(e,(p-1)(q-1)) = 1$
Compute d : $ed = 1 \bmod (p-1)(q-1)$
Return $\langle\langle n, e\rangle, \langle n, d\rangle\rangle$

Algorithm $S_{\langle n, d\rangle}(m)$, $m \in \mathbf{Z}_n$:
return $m^d \bmod n$

Algorithm $V_{\langle n, e\rangle}(m, \sigma)$:
$m' \leftarrow \sigma^e \bmod n$
if $m = m'$ return 1 else return 0

- How secure is this?

Security of Vanilla RSA Signatures

- **Vanilla RSA is existentially forgeable under a known message attack.**
  - Given $\langle m_1, \sigma_1 \rangle$ and $\langle m_2, \sigma_2 \rangle$, consider $\langle m_1 m_2 \bmod n,\ \sigma_1 \sigma_2 \bmod n \rangle$:

$$(\sigma_1 \sigma_2)^e \bmod n = ((m_1)^d (m_2)^d)^e \bmod n = m_1 m_2 \bmod n$$

- **Vanilla RSA is universally forgeable under a chosen message attack.**

RSA in Practice

- **Other measures are taken to strengthen RSA**
  - Versions are used that are existentially unforgeable under chosen message attack, under reasonable assumptions
- **One approach is called "hash-then-sign"**
  - Let h be a collision-resistant hash function

Algorithm $S_{\langle n, d \rangle}(m)$:
  return $h(m)^d \bmod n$

Algorithm $V_{\langle n, e \rangle}(m, \sigma)$:
  $m' \leftarrow \sigma^e \bmod n$
  if $h(m) = m'$ return 1 else return 0

- **Fully specified RSA signatures can be found in PKCS #1**

Pseudorandom Functions

- Intuitively, a ***pseudorandom function*** is a function

$$f\text{: Keys} \times \text{Domain} \rightarrow \text{Range}$$

  that is indistinguishable from a random function to anyone not knowing the key (the first input)
  - A useful primitive for a range of "higher level" crypto functions
  - Notation: Let $f_K(x) = f(K,x)$

- To define this precisely, let

$$F(\text{Domain} \rightarrow \text{Range})$$

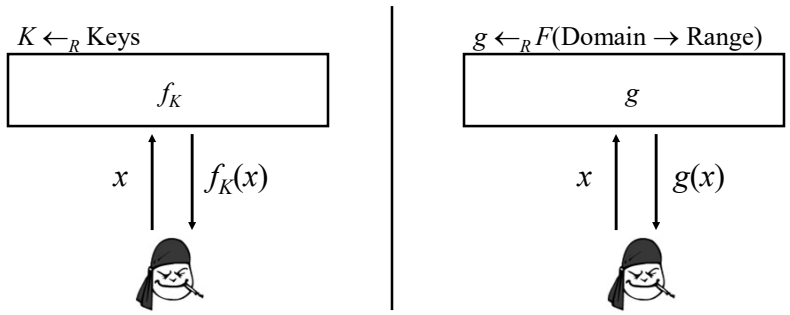
  denote the set of all functions from Domain to Range

25

25

Adversary for Pseudorandom Functions

- Adversary participates in one of two experiments



- Adversary queries oracle on inputs of its choice
- At end of experiment, adversary outputs a guess (0 or 1) as to which experiment he was participating in

26

26

Rough Definition of Pseudorandom Functions

- A collection of pseudorandom functions is a set

$$\{f^{\lambda} : \text{Keys}(\lambda) \times \text{Domain}(\lambda) \rightarrow \text{Range}(\lambda)\}_{\lambda \in \mathbf{N}}$$

such that for every PPT A there is a negligible ν_A where

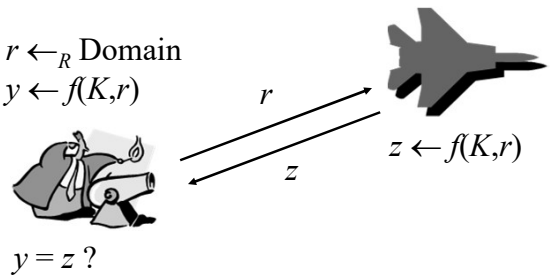
$$\Pr[A^{f_K^{\lambda}} = 1] - \Pr[A^{g^{\lambda}} = 1] \leq \nu_A(\lambda)$$

for all λ large enough, where the probabilities are taken over the choices of

$K \leftarrow_R \text{Keys}(\lambda)$
$g^{\lambda} \leftarrow_R F(\text{Domain}(\lambda) \rightarrow \text{Range}(\lambda))$

Application of Pseudorandom Functions

"Friend or foe" identification



- If friendly aircraft know K, then they can be challenged to respond with $f(K,r)$.
  - Requires Domain and Range to be large

Recall Informal Definition of a MAC

- **A message authentication code (MAC) scheme is a triple $\langle G, T, V \rangle$ of efficiently computable functions**
  - G outputs a "secret key" K

    $$K \leftarrow G(\cdot)$$
  - T takes a key K and "message" m as input, and outputs a "tag" t

    $$t \leftarrow T_K(m)$$
  - V takes a message m, tag t and key K as input, and outputs a bit b

    $$b \leftarrow V_K(m, t)$$
  - If $t \leftarrow T_K(m)$ then $V_K(m, t)$ outputs 1 ("valid")
  - Given only message/tag pairs $\{\langle m_i, T_K(m_i)\rangle\}_i$, it is computationally infeasible to compute $\langle m, t\rangle$ such that

    $$V_K(m, t) = 1$$

    for any new $m \neq m_i$

Pseudorandom Functions Make Good MACs

- **Let f be a pseudorandom function (for an appropriate λ)**
- **Select $K \leftarrow_R$ Keys**
- **Define $T_K(m) = f_K(m)$ for $m \in$ Domain**
- **Define**

$$V_K(m,t) = \begin{cases} 1 & \text{if } f_K(m) = t \\ 0 & \text{otherwise} \end{cases}$$

MACs for Longer Messages

- **Creating a suitable MAC is trickier than you might think**
- **Suppose Domain = $\{0,1\}^L$, Range = $\{0,1\}^{L'}$**
- **Proposal (where "|" denotes concatenation)**

Algorithm $T_K(m)$:
let $m_1|\ldots|m_n = m$, $m_i \in \{0,1\}^L$
for $i = 1\ldots n$ do $y_i \leftarrow f_K(m_i)$
$t \leftarrow y_1 \oplus \ldots \oplus y_n$
return t

Algorithm $V_K(m, t)$:
let $m_1|\ldots|m_n = m$, $m_i \in \{0,1\}^L$
for $i = 1\ldots n$ do $y_i \leftarrow f_K(m_i)$
$t' \leftarrow y_1 \oplus \ldots \oplus y_n$
if $t = t'$ return 1 else return 0

- **Is this secure?**

Two Simple Attacks

- **Attack #1**

A^{T_K} : Choose $m \in \{0,1\}^L$
$t \leftarrow T_K(m)$
Output $(m|m, 0^{L'})$

- **Attack #2**

A^{T_K} : Choose $m_1, m_2 \in \{0,1\}^L$
$t \leftarrow T_K(m_1|m_2)$
Output $(m_2|m_1, t)$

Another Proposal

Algorithm $T_K(m)$:
  let $m_1|\ldots|m_n = m$, $m_i \in \{0,1\}^{L-l}$
  for $i = 1\ldots n$ do $y_i \leftarrow f_K(\langle i \rangle \mid m_i)$
  $t \leftarrow y_1 \oplus \ldots \oplus y_n$
  return t

Algorithm $V_K(m, t)$:
  let $m_1|\ldots|m_n = m$, $m_i \in \{0,1\}^{L-l}$
  for $i = 1\ldots n$ do $y_i \leftarrow f_K(\langle i \rangle \mid m_i)$
  $t' \leftarrow y_1 \oplus \ldots \oplus y_n$
  if $t = t'$ return 1 else return 0

- **Is this secure?**

An Attack

- **No!**

A^{T_K} : Choose $m_1, m_1' \in \{0,1\}^L$, $m_1 \neq m_1'$
Choose $m_2, m_2' \in \{0,1\}^L$, $m_2 \neq m_2'$
$t_1 \leftarrow T_K(m_1|m_2)$
$t_2 \leftarrow T_K(m_1|m_2')$
$t_3 \leftarrow T_K(m_1'|m_2)$
Output $(m_1'|\, m_2'\,,\, t_1 \oplus t_2 \oplus t_3)$

A Third Proposal

- **Algorithm $T_K(m)$ outputs $t = \langle r, s \rangle$ where**

$$
\begin{aligned}
r &\leftarrow_R \{0,1\}^{l-1} \\
s &\leftarrow f_K(0 \mid r) \oplus f_K(1 \mid \langle 1 \rangle \mid m_1) \\
&\quad \oplus f_K(1 \mid \langle 2 \rangle \mid m_2) \\
&\quad \oplus \ldots \\
&\quad \oplus f_K(1 \mid \langle n \rangle \mid m_n)
\end{aligned}
$$

- **Is this secure?**
  - Yes, but we will not prove it here
  - Intuition: since A^{T_K} can invoke only T_K and not f_K, A^{T_K} cannot recover $f_K(0 \mid r)$

MACs from Cryptographic Hash Functions

- **Creating MACs using only hash functions is desirable since**
  - Popular hash functions (SHA-1, MD5) are faster than implementations of (thought-to-be) pseudorandom functions
  - Implementations of hash functions are readily and freely available, and are not subject to export controls of U.S. and other countries
- **The HMAC algorithm is an example**
  - Described in Internet RFC 2104
  - Mandatory to implement for Internet security protocols

HMAC

- **Let h be a cryptographic hash function (preimage resistant, 2nd preimage resistant, collision resistant)**

Algorithm $T_K(m)$:
  let ipad = the byte 0x36 repeated 64 times
  let opad = the byte 0x5C repeated 64 times
  $t \leftarrow h((K \oplus \text{opad}) \mid h((K \oplus \text{ipad}) \mid m))$
  return t

- **Security can be shown under non-standard but plausible assumptions about the hash function**

Informal Definition of Symmetric Encryption

- **A symmetric encryption scheme is a triple $\langle G, E, D \rangle$ of efficiently computable functions**
  - G outputs a "secret key" K
    $$K \leftarrow G(\cdot)$$
  - E takes a key K and "plaintext" m as input, and outputs a "ciphertext"
    $$c \leftarrow E_K(m)$$
  - D takes a ciphertext c and key K as input, and outputs $\perp$ or a plaintext
    $$m \leftarrow D_K(c)$$
  - If $c \leftarrow E_K(m)$ then $m \leftarrow D_K(c)$
  - If $c \leftarrow E_K(m)$, then c should reveal "no information" about m

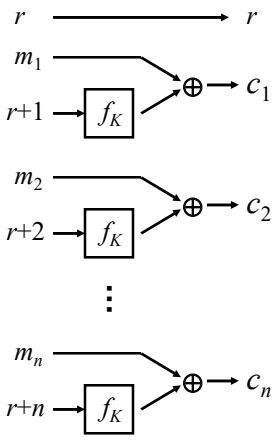
Example: "Counter Mode" Encryption

- Let f: $\mathbf{Keys} \times \{0,1\}^l \rightarrow \{0,1\}^L$ be a pseudorandom function

Algorithm $G()$:
  $K \leftarrow_R$ Keys
  return K

Algorithm $E_K(m)$:
  let $m_1|\ldots|m_n = m : m_i \in \{0,1\}^L$
  let $r \leftarrow_R \{0,1\}^l$
  for $i = 1\ldots n$ do $c_i \leftarrow f_K(r+i \bmod 2^l) \oplus m_i$
  return $r \mid c_1 \mid \ldots \mid c_n$

Algorithm $D_K(c)$:
  let $r \mid c_1 \mid \ldots \mid c_n = c : r \in \{0,1\}^l$ and $c_i \in \{0,1\}^L$
  for $i = 1\ldots n$ do $m_i \leftarrow f_K(r+i \bmod 2^l) \oplus c_i$
  return $m_1 \mid \ldots \mid m_n$



What Does "No Information" Mean?

- Option 1: Adversary cannot recover m from $E_K(m)$?
  - What if adversary can get first bit of m?
- Option 2: Adversary cannot recover any bit of m from $E_K(m)$?
  - What if adversary can get sum of bits in m?
- Here we will define security in terms of *indistinguishability*

Chosen Plaintext Attack (CPA)

- **Suppose the adversary is given two oracles**
  - An encryption oracle E_K
  - A test oracle $T_K(m_0, m_1)$ that can be called only once

Oracle $T_K(m_0, m_1)$:
if $|m_0| \neq |m_1|$ then return $\perp$
$b \leftarrow_R \{0,1\}$
return $E_K(m_b)$

- **The adversary must guess whether $b = 0$ or $b = 1$**

Rough Definition of a CPA-Secure Scheme

- **An CPA-secure encryption scheme is a triple**

$$\langle G, E, D \rangle$$

such that for every PPT A there is a negligible ν_A where

$$\Pr[\, A^{E_K, T_K} = 0 : b \leftarrow 0 \,] - \Pr[\, A^{E_K, T_K} = 0 : b \leftarrow 1 \,] \leq \nu_A(\lambda)$$

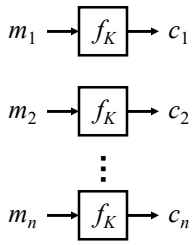
for all λ large enough, where the probabilities are taken over $K \leftarrow G(1^\lambda)$.

Example: ECB Encryption

- **ECB = "Electronic Code Book"**
- **Let f: Keys $\times \{0,1\}^L \rightarrow \{0,1\}^L$ be a pseudorandom *permutation***
  - f_K^{-1} exists and can be computed efficiently if k is known

Algorithm $E_K(m)$:
  let $m_1|\ldots|m_n = m : m_i \in \{0,1\}^L$
  for $i = 1\ldots n$ do $c_i \leftarrow f_K(m_i)$
  return $c_1 \mid \ldots \mid c_n$

Algorithm $D_K(c)$:
  let $c_1 \mid \ldots \mid c_n = c : c_i \in \{0,1\}^L$
  for $i = 1\ldots n$ do $m_i \leftarrow f_K^{-1}(c_i)$
  return $m_1 \mid \ldots \mid m_n$



- **Is this CPA-secure?**

CPA Security

- **ECB is not CPA-secure**
- **In fact, if E_K is deterministic and stateless, then it is not CPA secure**
- **To be CPA secure, E_K must be either nondeterministic or stateful**
  - Example of nondeterministic: Counter-mode encryption
  - Counter-mode encryption also has a deterministic, stateful version

Example: "Stateful Counter Mode" Encryption

- **Let f: Keys $\times \{0,1\}^l \rightarrow \{0,1\}^L$ be a pseudorandom function**

Algorithm $E_K(m)$:
  let $m_1|\ldots|m_n = m : m_i \in \{0,1\}^L$
  if $r = \perp$ then $r \leftarrow_R \{0,1\}^l$
  for $i = 1\ldots n$ do $c_i \leftarrow f_K(r+i \bmod 2^l) \oplus m_i$
  $c \leftarrow r \mid c_1 \mid \ldots \mid c_n$
  $r \leftarrow r + n \bmod 2^l$
  return c

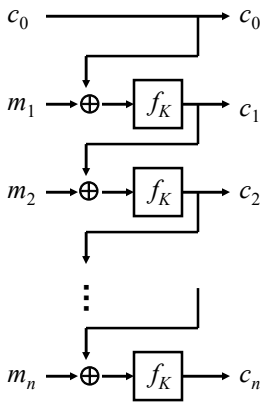
Algorithm $D_K(c)$:
  let $r \mid c_1 \mid \ldots \mid c_n = c : r \in \{0,1\}^l$ and $c_i \in \{0,1\}^L$
  for $i = 1\ldots n$ do $m_i \leftarrow f_K(r+i \bmod 2^l) \oplus c_i$
  return $m_1 \mid \ldots \mid m_n$

Example: "Cipher Block Chaining" Encryption

- **Let f: Keys $\times \{0,1\}^L \rightarrow \{0,1\}^L$ be a pseudorandom *permutation***

Algorithm $E_K(m)$:
  let $m_1|\ldots|m_n = m : m_i \in \{0,1\}^L$
  $c_0 \leftarrow_R \{0,1\}^L$
  for $i = 1\ldots n$ do $c_i \leftarrow f_K(c_{i-1} \oplus m_i)$
  return $c_0 \mid c_1 \mid \ldots \mid c_n$

Algorithm $D_K(c)$:
  let $c_0 \mid c_1 \mid \ldots \mid c_n = c : c_i \in \{0,1\}^L$
  for $i = 1\ldots n$ do $m_i \leftarrow f_K^{-1}(c_i) \oplus c_{i-1}$
  return $m_1 \mid \ldots \mid m_n$



Chosen Ciphertext Attack (CCA)

- **Suppose the adversary is given three oracles**
  - An encryption oracle E_K
  - A test oracle $T_K(m_0, m_1)$ that can be called only once

    Oracle $T_K(m_0, m_1)$:
    if $|m_0| \neq |m_1|$ then return $\perp$
    $b \leftarrow_R \{0,1\}$
    return $E_K(m_b)$

  - A decryption oracle D_K
- **The adversary must guess whether $b = 0$ or $b = 1$, but if**

  $$c \leftarrow T_K(m_0, m_1)$$

  then adversary cannot query $D_K(c)$
- **CCA is powerful enough to break all standard modes of operation (Counter, CBC, …)**

47

47

Informal Definition of Public Key Encryption

- **A public key encryption scheme is a triple $\langle G, E, D \rangle$ of efficiently computable functions**
  - G outputs a "public key" K and a "private key" K^{-1}

    $$\langle K, K^{-1} \rangle \leftarrow G(\cdot)$$

  - E takes public key K and plaintext m as input, and outputs a ciphertext

    $$c \leftarrow E_K(m)$$

  - D takes a ciphertext c and private key K^{-1} as input, and outputs $\perp$ or a plaintext

    $$m \leftarrow D_{K^{-1}}(c)$$

  - If $c \leftarrow E_K(m)$ then $m \leftarrow D_{K^{-1}}(c)$
  - If $c \leftarrow E_K(m)$, then c and K should reveal "no information" about m

48

48

"Vanilla RSA" Encryption Scheme

- **Frequently, K is made public**
- **Therefore, CPA security is *mandatory***
  - Defined precisely as for a symmetric cipher
- **Consider "Vanilla RSA" encryption**

Algorithm $G(1^\lambda)$:
  $p, q \leftarrow_R \{\lambda/2\text{-bit primes}\}$
  $n \leftarrow pq$
  Choose e: $\gcd(e,(p-1)(q-1)) = 1$
  Compute d : $ed = 1 \bmod (p-1)(q-1)$
  Return $\langle\langle n, e\rangle, \langle n, d\rangle\rangle$

Algorithm $E_{\langle n, e\rangle}(m)$, $m \in \mathbf{Z}_n$:
  return $m^e \bmod n$

Algorithm $D_{\langle n, d\rangle}(c)$:
  return $c^d \bmod n$

  - Is this CPA secure?

CCA-Security Can Be Achieved

- **PKCS #1 specifies RSA encryption that is secure against *chosen* ciphertext attacks under plausible assumptions**
  - Let G: $\{0,1\}^{k_0} \rightarrow \{0,1\}^{|n|-k_0}$
  - Let H: $\{0,1\}^{|n|-k_0} \rightarrow \{0,1\}^{k_0}$

Algorithm $E_{\langle n, e\rangle}(m)$, $|m| = |n| - k_0 - k_1$:
  $r \leftarrow_R \{0,1\}^{k_0}$
  $a \leftarrow m|0^{k1} \oplus G(r)$
  $b \leftarrow r \oplus H(a)$
  return $(a|b)^e \bmod n$

CCA-secure only for single "block"

Algorithm $D_{\langle n, d\rangle}(c)$:
  $a|b \leftarrow c^d \bmod n : a \in \{0,1\}^{|n|-k_0}, b \in \{0,1\}^{k_0}$
  $r \leftarrow H(a) \oplus b$
  $y \leftarrow G(r) \oplus a$
  if $y = m|0^{k1}$ then return m, else return $\perp$