ARIEL UNIVERSITY

אוניברסיטת אריאל בשומרון

Foundations of Modern Cryptography

Edward A. Hirsch

https://edwardahirsch.github.io/edwardahirsch

Neapolis University Pafos

Lecture 3: October 17, 2024

Lecture Plan

Public-Key Encryption Schemes

- Constructions
- Definitions of security

Lecture Plan

Public-Key Encryption Schemes

- Constructions
- Definitions of security

If the screen seems frozen and I do not respond,
please call me in Telegram.

Task: Public-Key Encryption

Alice
public key, secret key

Eve

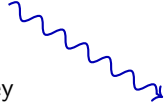
Bob
message

Task: Public-Key Encryption

public key

Alice

public key, secret key



Eve

Bob

message

Task: Public-Key Encryption

Alice
public key, secret key

public key

encrypted message

Eve

Bob
message

Pictures courtesy AI Emojis Generator

Task: Public-Key Encryption

Alice
public key, secret key

public key

encrypted message

Bob
message

Eve

Eve should be unable to decrypt (not to say recover the secret key)
even though she has seen a lot of communication (and perhaps some messages).

Pictures courtesy AI Emojis Generator

Hardcore Predicate

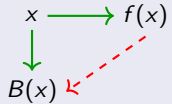
- One-way functions are hard to invert.
- What if we want to recover a *single* (say, the first) bit of the argument? Is it still hard?

Hardcore Predicate

- One-way functions are hard to invert.
- What if we want to recover a *single* (say, the first) bit of the argument? Is it still hard?

Definition

$B : \{0,1\}^* \to \{0,1\}$ is a hardcore predicate for $f : \{0,1\}^* \to \{0,1\}^*$ if
no adversary can guess $B(x)$ from $f(x)$ with probability non-negligibly better than 50%

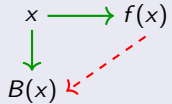


Hardcore Predicate

- One-way functions are hard to invert.
- What if we want to recover a *single* (say, the first) bit of the argument? Is it still hard?

Definition

A family of polynomial-time computable functions
$B_n\colon \{0,1\}^n \to \{0,1\}$ is a hardcore predicate for $f_n\colon \{0,1\}^n \to \{0,1\}^s$ if
no adversary can guess $B_n(x)$ from $f_n(x)$ with probability non-negligibly better than 50%:



$$\forall k\ \forall A\ \exists N\ \forall n > N \quad \mathbf{Pr}\{A(f_n(x)) = B_n(x)\} < \frac{1}{2} + \frac{1}{n^k},$$

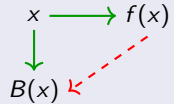
where A is a randomized polynomial-time adversary;
the probability is taken over $x \in U_n$ and over the randomness used by A.

Hardcore Predicate

- One-way functions are hard to invert.
- What if we want to recover a *single* (say, the first) bit of the argument? Is it still hard?

Definition

A family of polynomial-time computable functions
$B_n\colon \{0,1\}^n \to \{0,1\}$ is a hardcore predicate for $f_n\colon \{0,1\}^n \to \{0,1\}^s$ if
no adversary can guess $B_n(x)$ from $f_n(x)$ with probability non-negligibly better than 50%:



$$\forall k\ \forall A\ \exists N\ \forall n > N\quad \mathbf{Pr}\{A(f_n(x)) = B_n(x)\} < \frac{1}{2} + \frac{1}{n^k},$$

where A is a randomized polynomial-time adversary;
the probability is taken over $x \in U_n$ and over the randomness used by A.

Theorem (Oded Goldreich, Leonid Levin)

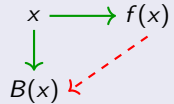
If f is a strongly one-way function, then $\tilde{f}(x,r) = (f(x), r)$ is a strongly one-way function and $B(x,r) = \langle x, r\rangle \bmod 2 = x_1 r_1 \oplus x_2 r_2 \oplus \ldots$ is its hardcore predicate. *// Proof delayed.*

Hardcore Predicate

- One-way functions are hard to invert.
- What if we want to recover a *single* (say, the first) bit of the argument? Is it still hard?

Definition

A family of polynomial-time computable functions $B_n\colon \{0,1\}^n \to \{0,1\}$ is a hardcore predicate for $f_n\colon \{0,1\}^n \to \{0,1\}^s$ if no adversary can guess $B_n(x)$ from $f_n(x)$ with probability non-negligibly better than 50%:



$$\forall k\ \forall A\ \exists N\ \forall n > N\quad \mathbf{Pr}\{A(f_n(x)) = B_n(x)\} < \frac{1}{2} + \frac{1}{n^k},$$

where A is a randomized polynomial-time adversary;
the probability is taken over $x \in U_n$ and over the randomness used by A.

Theorem (Oded Goldreich, Leonid Levin)

If f is a strongly one-way function, then $\tilde{f}(x,r) = (f(x), r)$ is a strongly one-way function and $B(x,r) = \langle x, r\rangle \bmod 2 = x_1 r_1 \oplus x_2 r_2 \oplus \ldots$ is its hardcore predicate. *// Proof delayed.*

Exercise

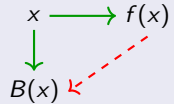
The construction works for trapdoor permutation families as well. Formulate and prove this.

Hardcore Predicate

- One-way functions are hard to invert.
- What if we want to recover a *single* (say, the first) bit of the argument? Is it still hard?

Definition

A family of polynomial-time computable functions $B_n\colon \{0,1\}^n \to \{0,1\}$ is a hardcore predicate for $f_n\colon \{0,1\}^n \to \{0,1\}^s$ if no adversary can guess $B_n(x)$ from $f_n(x)$ with probability non-negligibly better than 50%:



$$\forall k\ \forall A\ \exists N\ \forall n > N\quad \mathbf{Pr}\{A(f_n(x)) = B_n(x)\} < \frac{1}{2} + \frac{1}{n^k},$$

where A is a randomized polynomial-time adversary;
the probability is taken over $x \in U_n$ and over the randomness used by A.

Theorem (Oded Goldreich, Leonid Levin)

If f is a strongly one-way function, then $\tilde{f}(x,r) = (f(x), r)$ is a strongly one-way function and $B(x,r) = \langle x, r\rangle \bmod 2 = x_1 r_1 \oplus x_2 r_2 \oplus \ldots$ is its hardcore predicate. *// Proof delayed.*

Problem

This theorem does not claim that there is a hardcore predicate for f. Is it possible to build one?

Public-Key Encryption Scheme

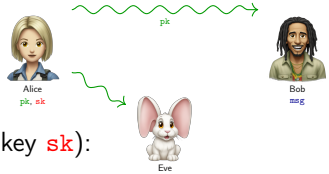
All algorithms below take time polynomial in n.
They can be randomized (and thus make errors) or deterministic.

INGREDIENTS:

- G (Alice) generates a pair of keys (public key pk, secret/private key sk):

Public-Key Encryption Scheme

All algorithms below take time polynomial in n.
They can be randomized (and thus make errors) or deterministic.



INGREDIENTS:

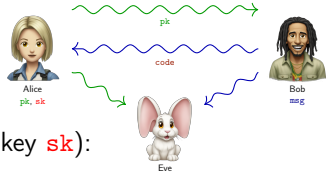
- G (Alice) generates a pair of keys (public key $\mathtt{pk}$, secret/private key $\mathtt{sk}$):

$$G\colon (1^n, r_g) \mapsto (\mathtt{pk}, \mathtt{sk})$$

($r_g \in \{0,1\}^*$ is a random string privately used by G).

Public-Key Encryption Scheme

All algorithms below take time polynomial in n.
They can be randomized (and thus make errors) or deterministic.



INGREDIENTS:

- G (Alice) generates a pair of keys (public key $\texttt{pk}$, secret/private key $\texttt{sk}$):

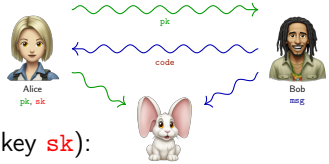
$$G\colon (1^n, r_g) \mapsto (\texttt{pk}, \texttt{sk})$$

  ($r_g \in \{0,1\}^*$ is a random string privately used by G).

- E (Bob) encrypts a one-bit message $\texttt{msg}$ using the public key: $\texttt{code} := E(\texttt{pk}, \texttt{msg})$.

Public-Key Encryption Scheme

All algorithms below take time polynomial in n.
They can be randomized (and thus make errors) or deterministic.



INGREDIENTS:

- G (Alice) generates a pair of keys (public key $\texttt{pk}$, secret/private key $\texttt{sk}$):

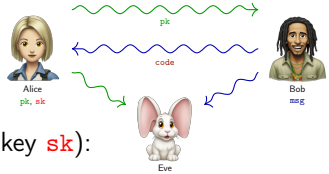
$$G\colon (1^n, r_g) \mapsto (\texttt{pk}, \texttt{sk})$$

  ($r_g \in \{0,1\}^*$ is a random string privately used by G).

- E (Bob) encrypts a one-bit message $\texttt{msg}$ using the public key: $\texttt{code} := E(\texttt{pk}, \texttt{msg})$.
- D (Alice) decrypts a one-bit message $\texttt{msg}$ using the secret key: $D(\texttt{sk}, \texttt{code}) \approx \texttt{msg}$.

Public-Key Encryption Scheme

All algorithms below take time polynomial in n.
They can be randomized (and thus make errors) or deterministic.



INGREDIENTS:

- G (Alice) generates a pair of keys (public key $\mathtt{pk}$, secret/private key $\mathtt{sk}$):

$$G\colon (1^n, r_g) \mapsto (\mathtt{pk}, \mathtt{sk})$$

  ($r_g \in \{0,1\}^*$ is a random string privately used by G).

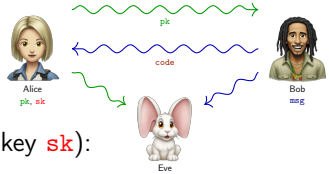
- E (Bob) encrypts a one-bit message $\mathtt{msg}$ using the public key: $\mathtt{code} := E(\mathtt{pk}, \mathtt{msg})$.

- D (Alice) decrypts a one-bit message $\mathtt{msg}$ using the secret key: $D(\mathtt{sk}, \mathtt{code}) \approx \mathtt{msg}$.
  We assume that the message is decrypted correctly:

$$D(\mathtt{sk}, E(\mathtt{pk}, \mathtt{msg})) = \mathtt{msg}$$

Public-Key Encryption Scheme

All algorithms below take time polynomial in n.
They can be randomized (and thus make errors) or deterministic.



INGREDIENTS:

- G (Alice) generates a pair of keys (public key $\mathtt{pk}$, secret/private key $\mathtt{sk}$):

$$G\colon (1^n, r_g) \mapsto (\mathtt{pk}, \mathtt{sk})$$

  ($r_g \in \{0,1\}^*$ is a random string privately used by G).

- E (Bob) encrypts a one-bit message $\mathtt{msg}$ using the public key: $\mathtt{code} := E(\mathtt{pk}, \mathtt{msg})$.

- D (Alice) decrypts a one-bit message $\mathtt{msg}$ using the secret key: $D(\mathtt{sk}, \mathtt{code}) \approx \mathtt{msg}$.
  We assume that the message is decrypted correctly:

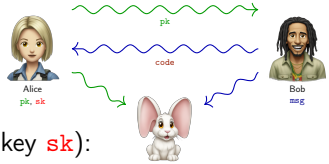
$$D(\mathtt{sk}, E(\mathtt{pk}, \mathtt{msg})) = \mathtt{msg}$$

  or at least

$$\mathbf{Pr}\{D(\mathtt{sk}, E(\mathtt{pk}, \mathtt{msg})) = \mathtt{msg}\} \geqslant 0.9.$$

Public-Key Encryption Scheme

All algorithms below take time polynomial in n.
They can be randomized (and thus make errors) or deterministic.



INGREDIENTS:

- G (Alice) generates a pair of keys (public key $\mathtt{pk}$, secret/private key $\mathtt{sk}$):

$$G\colon (1^n, r_g) \mapsto (\mathtt{pk}, \mathtt{sk})$$

  ($r_g \in \{0,1\}^*$ is a random string privately used by G).

- E (Bob) encrypts a one-bit message $\mathtt{msg}$ using the public key: $\mathtt{code} := E(\mathtt{pk}, \mathtt{msg})$.

- D (Alice) decrypts a one-bit message $\mathtt{msg}$ using the secret key: $D(\mathtt{sk}, \mathtt{code}) \approx \mathtt{msg}$.
  We assume that the message is decrypted correctly:

$$D(\mathtt{sk}, E(\mathtt{pk}, \mathtt{msg})) = \mathtt{msg}$$

  or at least

$$\mathbf{Pr}\{D(\mathtt{sk}, E(\mathtt{pk}, \mathtt{msg})) = \mathtt{msg}\} \geqslant 0.9.$$

- An adversary $A(\mathtt{pk}, \mathtt{code})$ (Eve) should not be able to decrypt (without $\mathtt{sk}$) with non-negligible probability, at least

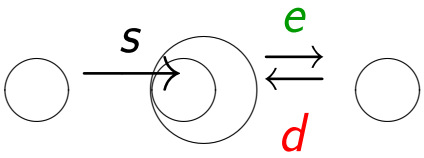
$$\mathbf{Pr}\{A(\mathtt{pk}, E(\mathtt{pk}, \mathtt{msg})) = \mathtt{msg}\} < \varepsilon(n).$$

  negligible

  (more on this later).

Public-Key Encryption Scheme

A construction for encrypting 1 bit based on trapdoor permutations and hardcore predicates



Caution: discrepancy in the literature what is "permutation"

Assume that G generates a trapdoor permutation family:
deterministic polynomial-time $G\colon (1^n, r_g) \mapsto (e, d, s)$ (Boolean circuits) s.t.

- $e\colon \{0,1\}^n \to \{0,1\}^{\zeta(n)}$
- $d\colon \{0,1\}^{\zeta(n)} \to \{0,1\}^n$
- $s\colon \{0,1\}^{\sigma(n)} \to \{0,1\}^n$

such that $\forall x \in \mathrm{Im}(s)\ d(e(x)) = x$.

Let $B(x)$ be a hardcore predicate, $B(x)$ is hard to guess from $e(x)$.

Public-Key Encryption Scheme

Caution: discrepancy in the literature what is "permutation"

A construction for encrypting 1 bit based on trapdoor permutations and hardcore predicates

Assume that G generates a trapdoor permutation family:
Let $B(x)$ be a hardcore predicate, $B(x)$ is hard to guess from $e(x)$.

Key generation: G is the same, $\mathtt{pk} = (e, s)$, $\mathtt{sk} = d$.

Public-Key Encryption Scheme

Caution: discrepancy in the literature what is "permutation"

A construction for encrypting 1 bit based on trapdoor permutations and hardcore predicates

Assume that G generates a trapdoor permutation family:
Let $B(x)$ be a hardcore predicate, $B(x)$ is hard to guess from $e(x)$.

Key generation: G is the same, $\mathtt{pk} = (e, s)$, $\mathtt{sk} = d$.
Encryption: $E(\mathtt{pk}, \mathtt{msg}) = (e(x), B(x) \oplus \mathtt{msg})$. // $e(x)$ and $B(x) \oplus \mathtt{msg}$

Public-Key Encryption Scheme

Caution: discrepancy in the literature what is "permutation"

A construction for encrypting 1 bit based on trapdoor permutations and hardcore predicates

Assume that G generates a trapdoor permutation family:
Let $B(x)$ be a hardcore predicate, $B(x)$ is hard to guess from $e(x)$.

Key generation: G is the same, $\mathtt{pk} = (e, s)$, $\mathtt{sk} = d$.

Encryption: $E(\mathtt{pk}, \mathtt{msg}) = (e(s(r)), B(s(r)) \oplus \mathtt{msg})$. // $e(x)$ and $B(x) \oplus \mathtt{msg}$

Decryption: $D(\mathtt{sk}, \mathtt{code}) = B(d(\mathtt{code}_1)) \oplus \mathtt{code}_2$. // $B(d(e(x))) \oplus B(x) \oplus \mathtt{msg}$

Public-Key Encryption Scheme

Caution: discrepancy in the literature what is "permutation"

A construction for encrypting 1 bit based on trapdoor permutations and hardcore predicates

Assume that G generates a trapdoor permutation family:
Let $B(x)$ be a hardcore predicate, $B(x)$ is hard to guess from $e(x)$.

Key generation: G is the same, $\mathtt{pk} = (e, s)$, $\mathtt{sk} = d$.
Encryption: $E(\mathtt{pk}, \mathtt{msg}) = (e(s(r)), B(s(r)) \oplus \mathtt{msg})$. $/\!/ e(x)$ and $B(x) \oplus \mathtt{msg}$
Decryption: $D(\mathtt{sk}, \mathtt{code}) = B(d(\mathtt{code}_1)) \oplus \mathtt{code}_2$. $/\!/ B(d(e(x))) \oplus B(x) \oplus \mathtt{msg}$

If A breaks it, we break the hardcore predicate:

Take $b \in \{0, 1\}$ at random.
Compute $B(x)$ from y as $\beta := A(\mathtt{pk}, (y, b)) \oplus b$ $/\!/ x = s(r)$, $y = e(x)$

Public-Key Encryption Scheme

Caution: discrepancy in the literature what is "permutation"

A construction for encrypting 1 bit based on trapdoor permutations and hardcore predicates

Assume that G generates a trapdoor permutation family:
Let $B(x)$ be a hardcore predicate, $B(x)$ is hard to guess from $e(x)$.

Key generation: G is the same, $\mathtt{pk} = (e, s)$, $\mathtt{sk} = d$.
Encryption: $E(\mathtt{pk}, \mathtt{msg}) = (e(s(r)), B(s(r)) \oplus \mathtt{msg})$. $\quad /\!/ e(x)$ and $B(x) \oplus \mathtt{msg}$
Decryption: $D(\mathtt{sk}, \mathtt{code}) = B(d(\mathtt{code}_1)) \oplus \mathtt{code}_2$. $\quad /\!/ B(d(e(x))) \oplus B(x) \oplus \mathtt{msg}$

If A breaks it, we break the hardcore predicate:

Take $b \in \{0,1\}$ at random.
Compute $B(x)$ from y as $\beta := A(\mathtt{pk}, (y, b)) \oplus b$ $\quad /\!/ x = s(r),\ y = e(x)$

$$
\begin{aligned}
\mathbf{Pr}\{\beta = B(x)\} &= \mathbf{Pr}\{A(\mathtt{pk}, (y, b)) \oplus b = B(x)\} \\
&= \mathbf{Pr}\{A(\mathtt{pk}, (e(x), b)) \oplus b = B(x)\} \\
&= \mathbf{Pr}\{A(\mathtt{pk}, E_x(\mathtt{pk}, B(x) \oplus b) \oplus b = B(x)\} = \mathbf{Pr}\{A(\mathtt{pk}, E_x(\mathtt{pk}, \underbrace{B(x) \oplus b}_{\mathtt{msg}}) = \underbrace{B(x) \oplus b}_{\mathtt{msg}}\}
\end{aligned}
$$

Public-Key Encryption Scheme

Caution: discrepancy in the literature what is "permutation"

A construction for encrypting 1 bit based on trapdoor permutations and hardcore predicates

Assume that G generates a trapdoor permutation family:
Let $B(x)$ be a hardcore predicate, $B(x)$ is hard to guess from $e(x)$.

Key generation: G is the same, $\mathtt{pk} = (e, s)$, $\mathtt{sk} = d$.
Encryption: $E(\mathtt{pk}, \mathtt{msg}) = (e(s(r)), B(s(r)) \oplus \mathtt{msg})$. $/\!/ e(x)$ and $B(x) \oplus \mathtt{msg}$
Decryption: $D(\mathtt{sk}, \mathtt{code}) = B(d(\mathtt{code}_1)) \oplus \mathtt{code}_2$. $/\!/ B(d(e(x))) \oplus B(x) \oplus \mathtt{msg}$

If A breaks it, we break the hardcore predicate:

Take $b \in \{0,1\}$ at random.
Compute $B(x)$ from y as $\beta := A(\mathtt{pk}, (y, b)) \oplus b$ $/\!/ x = s(r),\ y = e(x)$

$$
\begin{aligned}
\mathbf{Pr}\{\beta = B(x)\} &= \mathbf{Pr}\{A(\mathtt{pk}, (y, b)) \oplus b = B(x)\} \\
&= \mathbf{Pr}\{A(\mathtt{pk}, (e(x), b)) \oplus b = B(x)\} \\
&= \mathbf{Pr}\{A(\mathtt{pk}, E_x(\mathtt{pk}, B(x) \oplus b) \oplus b = B(x)\} = \mathbf{Pr}\{A(\mathtt{pk}, E_x(\mathtt{pk}, \underbrace{B(x) \oplus b}_{\mathtt{msg}}) = \underbrace{B(x) \oplus b}_{\mathtt{msg}}\}
\end{aligned}
$$

Distributions: $x \leftarrow s(r)$, $r \leftarrow U_{s(n)}$, $b \leftarrow U_2$, $r_g \leftarrow U_{\ldots}$; A, D, E are randomized.

Exercise

It is hard to decrypt a *random* message (bit). Is it enough?

Is it secure enough?

A TCS approach: Computational indistinguishability

How to distinguish...

- a Geiger counter output from a general-purpose computer output,

Who tries to distinguish?

Is it secure enough?

A TCS approach: Computational indistinguishability

How to distinguish...

- a Geiger counter output from a general-purpose computer output,
- one random source from another random source?

Who tries to distinguish?

Is it secure enough?

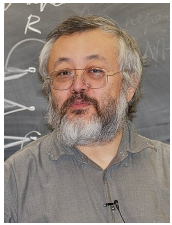
A TCS approach: Computational indistinguishability

How to distinguish...

- a Geiger counter output from a general-purpose computer output,
- one random source from another random source?

Who tries to distinguish?

- a mathematician? (ask Sasha Shen!) ⟶



Alexander Shen

Taken by A. Smal

Is it secure enough?

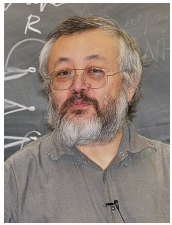
A TCS approach: Computational indistinguishability

How to distinguish...

- a Geiger counter output from a general-purpose computer output,
- one random source from another random source?

Who tries to distinguish?

- a mathematician? (ask Sasha Shen!) ⟶
- a computer?



Alexander Shen

Taken by A. Smal

Is it secure enough?

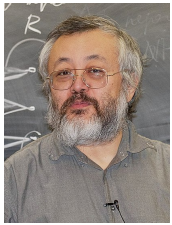
A TCS approach: Computational indistinguishability

How to distinguish...

- a Geiger counter output from a general-purpose computer output,
- one random source from another random source?

Who tries to distinguish?

- a mathematician? (ask Sasha Shen!)
- a computer?
- a polynomial-time bounded computer!



Alexander Shen

Taken by A. Smal

typically polynomial-time

Definition

Probability distributions P and Q are computationally indistinguishable if $\forall$ adversary A // *Outputs 0 or 1.*

$$\left|\mathbf{Pr}_{x\leftarrow \mathsf{P}}\{A(x)=1\} - \mathbf{Pr}_{x\leftarrow \mathsf{Q}}\{A(x)=1\}\right| < \varepsilon(n)$$

.

Is it secure enough?

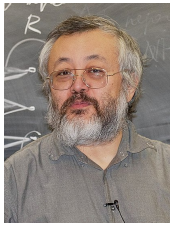
A TCS approach: Computational indistinguishability

How to distinguish...

- a Geiger counter output from a general-purpose computer output,
- one random source from another random source?

Who tries to distinguish?

- a mathematician? (ask Sasha Shen!)
- a computer?
- a polynomial-time bounded computer!



Alexander Shen

Taken by A. Smal

typically polynomial-time

Definition

Probability distributions P and Q are computationally indistinguishable if $\forall k$ $\forall$ adversary A // *Outputs 0 or 1.*

$$\left|\mathbf{Pr}_{x \leftarrow P_n}\{A(x) = 1\} - \mathbf{Pr}_{x \leftarrow Q_n}\{A(x) = 1\}\right| < \frac{1}{n^k}$$

for n big enough ($P = \{P_n\}_n$, $Q = \{Q_n\}_n$ are ensembles of distributions).

Indistinguishability as a security criterion (not the only one!)

Not just single-bit messages: their bits are dependent

PKES = PKCS

Definition

or generated by the adversary?

A PKCS is computationally indistinguishable if $\forall$ messages (m_0, m_1) of polynomial length $\forall$ adversary A

$$\Big| \; \mathbf{Pr}\{A(E(m_0, e, r_e), e, 1^n, m_0, m_1) = 1\} \; - \; \mathbf{Pr}\{A(E(m_1, e, r_e), e, 1^n, m_0, m_1) = 1\} \; \Big| \; < \varepsilon(n).$$

The probability is taken over r_g, r_e, and A's randomness.

Indistinguishability of 1-bit PKCS

Theorem (already proved)

For the 1-bit PKCS we constructed, $\forall A$

$$\mathbf{Pr}\{A(e(\mathtt{msg}, r_e), 1^n, e) = \mathtt{msg}\} < \frac{1}{2} + \varepsilon(n),$$

where $G(1^n, r_g) = (e, d)$, and **Pr** is over everything including msg.

Definition

1-bit PKCS is computationally indistinguishable, if $\forall A$

$$\left|\mathbf{Pr}\{A(e(1, r_e), 1^n, e) = 1\} - \mathbf{Pr}\{A(e(0, r_e), 1^n, e) = 1\}\right| < \varepsilon(n),$$

where $G(1^n, r_g) = (e, d)$, and **Pr** is over everything excluding msg.

Exercise (do it now?)

Is our PKCS indistinguishable?

Public-Key Encryption Scheme for arbitrary messages

Let $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ (the same key pair!).

Break the old (1-bit) cryptosystem using an adversary A^* for this one.

Public-Key Encryption Scheme for arbitrary messages

Let $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ (the same key pair!).

Break the old (1-bit) cryptosystem using an adversary A^* for this one.

We are given $E(0)$ and $E(1)$, as two codewords $\texttt{code}_a, \texttt{code}_b$. How to distinguish them?

Public-Key Encryption Scheme for arbitrary messages

Let $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ (the same key pair!).

Break the old (1-bit) cryptosystem using an adversary A^* for this one.

We are given $E(0)$ and $E(1)$, as two codewords $\texttt{code}_a, \texttt{code}_b$. How to distinguish them?

A^* can distinguish two messages $m_0 = s_1 \ldots s_p$ and $m_1 = t_1 \ldots t_p$,

$\frac{1}{n^k} < |\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} s_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(t_1 t_2 \ldots t_{p-1} t_p \ldots) = 1\}| \leqslant$

Public-Key Encryption Scheme for arbitrary messages

Let $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ (the same key pair!).

Break the old (1-bit) cryptosystem using an adversary A^* for this one.

We are given $E(0)$ and $E(1)$, as two codewords $\texttt{code}_a, \texttt{code}_b$. How to distinguish them?

A^* can distinguish two messages $m_0 = s_1 \ldots s_p$ and $m_1 = t_1 \ldots t_p$, consider everything between them.

$$\frac{1}{n^k} < |\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} s_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(t_1 t_2 \ldots t_{p-1} t_p \ldots) = 1\}| \leqslant$$

$$\begin{aligned}
&|\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} s_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} t_p \ldots) = 1\}| + \\
&|\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} t_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(s_1 s_2 \ldots t_{p-1} t_p \ldots) = 1\}| + \\
&\qquad \ldots \\
&|\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots t_{p-1} t_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(s_1 t_2 \ldots t_{p-1} t_p \ldots) = 1\}| + \\
&|\mathbf{Pr}\{A^*(E(s_1 t_2 \ldots t_{p-1} t_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(t_1 t_2 \ldots t_{p-1} t_p, \ldots) = 1\}|.
\end{aligned}$$

There is a substantial difference somewhere here: $|\ldots s_{i_*} \ldots - \ldots t_{i_*} \ldots| > \frac{1}{n^k}\frac{1}{p} \geqslant \frac{1}{n^r}$.

Public-Key Encryption Scheme for arbitrary messages

Let $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ (the same key pair!).

Break the old (1-bit) cryptosystem using an adversary A^* for this one.

We are given $E(0)$ and $E(1)$, as two codewords $\texttt{code}_a, \texttt{code}_b$. How to distinguish them?

A^* can distinguish two messages $m_0 = s_1 \ldots s_p$ and $m_1 = t_1 \ldots t_p$, consider everything between them.

$$\frac{1}{n^k} < |\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} s_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(t_1 t_2 \ldots t_{p-1} t_p \ldots) = 1\}| \leqslant$$

$$\begin{aligned}
&|\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} s_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} t_p \ldots) = 1\}| + \\
&|\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots s_{p-1} t_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(s_1 s_2 \ldots t_{p-1} t_p \ldots) = 1\}| + \\
&\qquad \ldots \\
&|\mathbf{Pr}\{A^*(E(s_1 s_2 \ldots t_{p-1} t_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(s_1 t_2 \ldots t_{p-1} t_p \ldots) = 1\}| + \\
&|\mathbf{Pr}\{A^*(E(s_1 t_2 \ldots t_{p-1} t_p \ldots) = 1\} - \mathbf{Pr}\{A^*(E(t_1 t_2 \ldots t_{p-1} t_p, \ldots) = 1\}|.
\end{aligned}$$

There is a substantial difference somewhere here: $|\ldots s_{i_*} \ldots - \ldots t_{i_*} \ldots| > \frac{1}{n^k}\frac{1}{p} \geqslant \frac{1}{n^{k'}}$.

Substitute $\texttt{code}_a$ for s_i and $\texttt{code}_b$ for t_i, generate other bits s_j, t_j at random.

Where is i_*? Try a random i. Overall success:

$$\mathbf{Pr}\{i = i_*\} \cdot \frac{1}{n^{k'}}.$$

$\mathbf{Pr}\{i = i_*\} = \frac{1}{p}$ is good if the messages were of polynomial length.

Is it secure enough?

Behavioural approach: Semantic security

Definition

A PKCS is semantically secure if $\forall f\ \forall h\ \forall A\ \forall M$

$$\mathbf{Pr}\{A(E(e,m),e,h(m)) = f(m)\} \leqslant \mathbf{Pr}\{\widetilde{A}(e,h(m)) = f(m)\} + \varepsilon(n),$$

where the hint h and the "sense" f are polynomial-time computable,
M is a message generator,
A is an adversary trying to guess f from `code`,
$\widetilde{A}$ is an "adversary" guessing f without `code`.

Alice

pk, sk

Bob

msg

Eve

Is it secure enough?

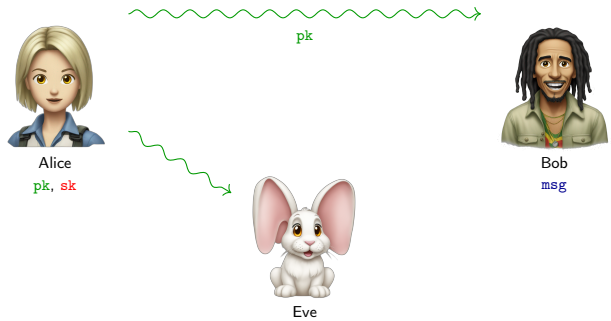
Behavioural approach: Semantic security

Definition

A PKCS is semantically secure if $\forall f\ \forall h\ \forall A\ \forall M$

$$\mathbf{Pr}\{A(E(e,m),e,h(m)) = f(m)\} \leqslant \mathbf{Pr}\{\widetilde{A}(e,h(m)) = f(m)\} + \varepsilon(n),$$

where the hint h and the "sense" f are polynomial-time computable,
M is a message generator,
A is an adversary trying to guess f from `code`,
$\widetilde{A}$ is an "adversary" guessing f without `code`.



Is it secure enough?

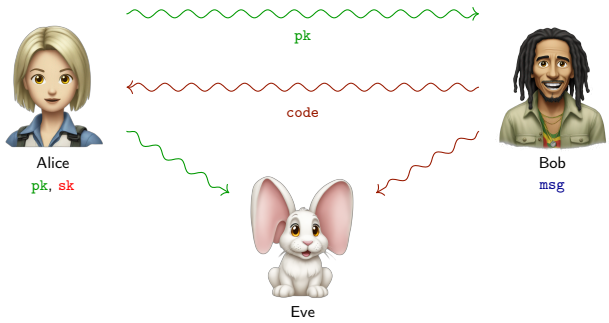
Behavioural approach: Semantic security

Definition

A PKCS is semantically secure if $\forall f\ \forall h\ \forall A\ \forall M$

$$\mathbf{Pr}\{A(E(e,m),e,h(m)) = f(m)\} \leqslant \mathbf{Pr}\{\widetilde{A}(e,h(m)) = f(m)\} + \varepsilon(n),$$

where the hint h and the "sense" f are polynomial-time computable,
M is a message generator,
A is an adversary trying to guess f from `code`,
$\widetilde{A}$ is an "adversary" guessing f without `code`.



Is it secure enough?

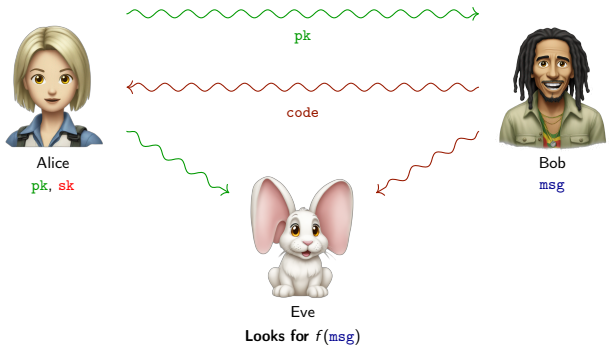
Behavioural approach: Semantic security

Definition

A PKCS is semantically secure if $\forall f\ \forall h\ \forall A\ \forall M$

$$\mathbf{Pr}\{A(E(e,m),e,h(m)) = f(m)\} \leqslant \mathbf{Pr}\{\widetilde{A}(e,h(m)) = f(m)\} + \varepsilon(n),$$

where the hint h and the "sense" f are polynomial-time computable,
M is a message generator,
A is an adversary trying to guess f from `code`,
$\widetilde{A}$ is an "adversary" guessing f without `code`.



Is it secure enough?

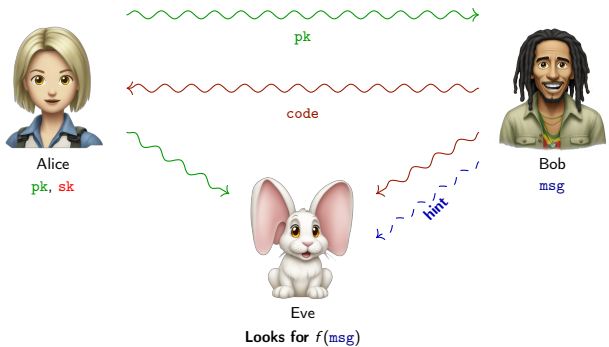
Behavioural approach: Semantic security

Definition

A PKCS is semantically secure if $\forall f\ \forall h\ \forall A\ \forall M$

$$\mathbf{Pr}\{A(E(e,m),e,h(m)) = f(m)\} \leqslant \mathbf{Pr}\{\widetilde{A}(e,h(m)) = f(m)\} + \varepsilon(n),$$

where the hint h and the "sense" f are polynomial-time computable,
M is a message generator,
A is an adversary trying to guess f from `code`,
$\widetilde{A}$ is an "adversary" guessing f without `code`.



Is it secure enough?

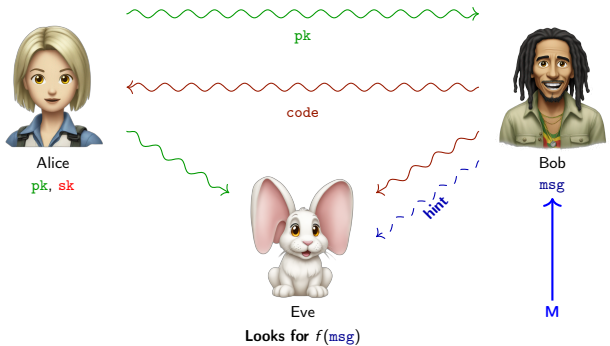
Behavioural approach: Semantic security

Definition

A PKCS is semantically secure if $\forall f\ \forall h\ \forall A\ \forall M$

$$\mathbf{Pr}\{A(E(e,m),e,h(m)) = f(m)\} \leqslant \mathbf{Pr}\{\widetilde{A}(e,h(m)) = f(m)\} + \varepsilon(n),$$

where the hint h and the "sense" f are polynomial-time computable,
M is a message generator,
A is an adversary trying to guess f from `code`,
$\widetilde{A}$ is an "adversary" guessing f without `code`.



Is it secure enough?

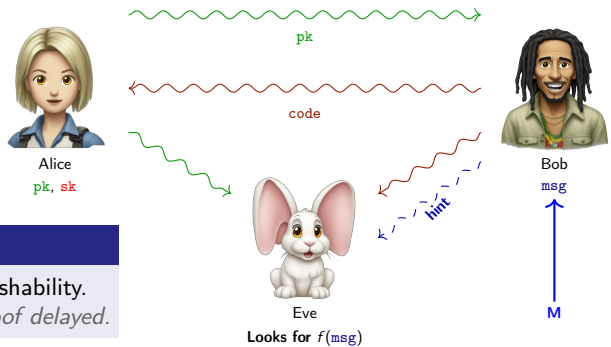
Behavioural approach: Semantic security

Definition

A PKCS is semantically secure if $\forall f\ \forall h\ \forall A\ \forall M$

$$\mathbf{Pr}\{A(E(e,m),e,h(m)) = f(m)\} \leqslant \mathbf{Pr}\{\widetilde{A}(e,h(m)) = f(m)\} + \varepsilon(n),$$

where the hint h and the "sense" f are polynomial-time computable,
M is a message generator,
A is an adversary trying to guess f from `code`,
$\widetilde{A}$ is an "adversary" guessing f without `code`.



Theorem

Semantic security $\Leftrightarrow$ computational indistinguishability.

// Proof delayed.

A more efficient PKCS

- $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ has a separate codeword for each bit.

A more efficient PKCS

- r is short for $s(r)$ (a random string generated by the sampler)
- $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ has a separate codeword for each bit.
- $E^*(b_1 b_2 \ldots) = ((\underbrace{e(r_1)}_{n \text{ bits}}, B(r_1) \oplus b_1), (\underbrace{e(r_2)}_{n \text{ bits}}, B(r_2) \oplus b_2), \ldots)$.

A more efficient PKCS

- r is short for $s(r)$ (a random string generated by the sampler)
- $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ has a separate codeword for each bit.
- $E^*(b_1 b_2 \ldots) = (\underbrace{(e(r_1)}_{n \text{ bits}}, B(r_1) \oplus b_1), \underbrace{(e(r_2)}_{n \text{ bits}}, B(r_2) \oplus b_2), \ldots)$.
- $E^{**}(b_1 \ldots b_m, e, r) = (e^m(r), B(r) \oplus b_1, B(e(r)) \oplus b_2, \ldots)$, is better.

A more efficient PKCS

- r is short for $s(r)$ (a random string generated by the sampler)
- $E^*(b_1 b_2 \ldots) = (E(b_1), E(b_2) \ldots)$ has a separate codeword for each bit.
- $E^*(b_1 b_2 \ldots) = (\underbrace{(e(r_1)}_{n \text{ bits}}, B(r_1) \oplus b_1), \underbrace{(e(r_2)}_{n \text{ bits}}, B(r_2) \oplus b_2), \ldots)$.
- $E^{**}(b_1 \ldots b_m, e, r) = (e^m(r), B(r) \oplus b_1, B(e(r)) \oplus b_2, \ldots)$, is better.
- We will later see that its codewords are indistinguishable from codewords of random messages.

Takeout

- PKCS from a trapdoor permutation and a hardcore predicate.
- Indistinguishability vs security.

Coming next:

- *Goldreich–Levin Theorem proof.*
- *Indistinguishability vs security proof.*
- *PRGs (pseudorandom generators).*