ARIEL UNIVERSITY

אוניברסיטת אריאל בשומרון

Foundations of Modern Cryptography

Edward A. Hirsch

https://edwardahirsch.github.io/edwardahirsch

Neapolis University Pafos

Lecture 5: October 31, 2024

Lecture Plan

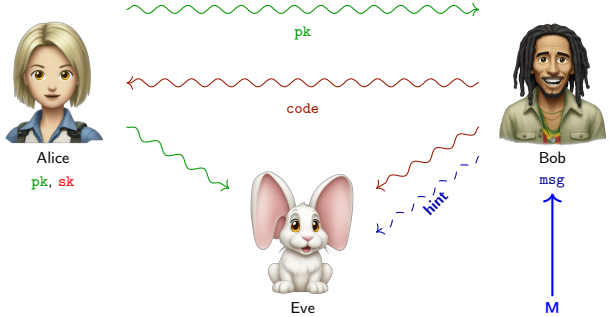
- Indistinguishability implies semantic security — a proof.
- Private-key cryptography: not so simple.

Lecture Plan

- Indistinguishability implies semantic security — a proof.
- Private-key cryptography: not so simple.

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

Computational indistinguishability vs 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~~ (we'll see),
M is a message generator, A is an adversary trying to guess f from `code` (ciphertext),
$\widetilde{A}$ is an "adversary" guessing f without `code`.

Computational indistinguishability vs Semantic security

Definition

Probability distributions P and Q are computationally indistinguishable if $\forall$ adversary A

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

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~~ (we'll see),
M is a message generator, A is an adversary trying to guess f from `code` (ciphertext),
$\widetilde{A}$ is an "adversary" guessing f without `code`.

Computational indistinguishability vs Semantic security

Definition

Probability distributions P and Q are computationally indistinguishable if $\forall$ adversary A

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

Definition

or generated by the adversary?

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

$$\left| \begin{array}{ll} \mathbf{Pr}\{A(E(m_0,e,r_e),e,1^n,m_0,m_1)=1\} & - \\ \mathbf{Pr}\{A(E(m_1,e,r_e),e,1^n,m_0,m_1)=1\} & \end{array} \right| < \varepsilon(n).$$

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

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~~ (we'll see),
M is a message generator, A is an adversary trying to guess f from `code` (ciphertext),
$\widetilde{A}$ is an "adversary" guessing f without `code`.

Computational indistinguishability vs Semantic security

Definition

Probability distributions P and Q are computationally indistinguishable if $\forall$ adversary A

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

Definition

or generated by the adversary?

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

$$\left|\begin{array}{ll} \mathbf{Pr}\{A(E(m_0,e,r_e),e,1^n,m_0,m_1)=1\} & - \\ \mathbf{Pr}\{A(E(m_1,e,r_e),e,1^n,m_0,m_1)=1\} & \end{array}\right| < \varepsilon(n).$$

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

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~~ (we'll see),
M is a message generator, A is an adversary trying to guess f from `code` (ciphertext),
$\widetilde{A}$ is an "adversary" guessing f without `code`.

Semantic security $\Leftrightarrow$ computational indistinguishability *// It remains to prove $\Leftarrow$*

Indistinguishability to semantic security (1)

- Assume A_S can compute $f(m)$, then construct m_0, m_1 with $f(m_0) \neq f(m_1)$ that we will distinguish.
- New adversary A_I given encrypted m distinguishes $m = m_0, m_1$ sent with prob. $\frac{1}{2}$ each:
  `if` $A_S(E(m, \ldots), h(m_0)) = f(m_0)$ `then return 0,`
  `otherwise flip a fair coin and return the result.`
- Note that $f(m_0), h(m_0)$ are **hardwired into the circuit** A_I.

Indistinguishability to semantic security (1)

- Assume A_S can compute $f(m)$, then construct m_0, m_1 with $f(m_0) \neq f(m_1)$ that we will distinguish.
- New adversary A_I given encrypted m distinguishes $m = m_0, m_1$ sent with prob. $\frac{1}{2}$ each:

  `if` $A_S(E(m,\ldots), h(m_0)) = f(m_0)$ `then return 0,`
  `otherwise flip a fair coin and return the result.`

- Note that $f(m_0), h(m_0)$ are **hardwired into the circuit** A_I.
- Let
  - $p_k(x) := \mathbf{Pr}\{A_S(E(x,\ldots), h(m)) = k\}$, *// Normal scenario for A_S*
  - $q_k(x) := \mathbf{Pr}\{A_S(E(x,\ldots), h(m_0)) = k\}$, *// Wrong hint, what would it say?*
  - $f_i := f(m_i)$.
- $\mathbf{Pr}\{A_I \text{ succeeds}\} =$

$$\mathbf{Pr}\{m_0 \text{ is sent}\} \cdot \left(p_{f_0}(m_0) + \frac{1}{2}(1 - p_{f_0}(m_0))\right) + \mathbf{Pr}\{m_1 \text{ is sent}\} \cdot \left(\frac{1}{2}(1 - q_{f_0}(m_1))\right) =$$

$$\frac{1}{2}\left(1 + \frac{p_{f_0}(m_0) - q_{f_0}(m_1)}{2}\right)$$

- It remains to prove that there are m_0, m_1 with substantial $p_{f_0}(m_0) - q_{f_0}(m_1)$.

Indistinguishability to semantic security (2)

Assume the contrary: for all m_0, m_1 success is unlikely:

$$p_{f_0}(m_0) - q_{f_0}(m_1) < \varepsilon.$$

Indistinguishability to semantic security (2)

Assume the contrary: for all m_0, m_1 success is unlikely:

$$p_{f_0}(m_0) - q_{f_0}(m_1) < \varepsilon.$$

Take a weighted sum: $/\!/ p(x) = \boldsymbol{Pr}\{M_S(1^n) = x\}$

$$\sum_{m_0, m_1} p(m_0)p(m_1)(p_{f_0}(m_0) - q_{f_0}(m_1)) \overset{\sum_{m_1} p(m_1)=1}{=\!=\!=} \sum_x p(x)p_{f(x)}(x) - \sum_{m_0, m_1} p(m_0)p(m_1)q_{f_0}(m_1). \qquad (*)$$

The first part is the prob. of success of the original A_S.

Indistinguishability to semantic security (2)

Assume the contrary: for all m_0, m_1 success is unlikely:

$$p_{f_0}(m_0) - q_{f_0}(m_1) < \varepsilon.$$

Take a weighted sum: $\quad /\!/ p(x) = \mathbf{Pr}\{M_S(1^n) = x\}$

$$\sum_{m_0, m_1} p(m_0)p(m_1)(p_{f_0}(m_0) - q_{f_0}(m_1)) \overset{\sum_{m_1} p(m_1)=1}{=\!=\!=} \sum_x p(x)p_{f(x)}(x) - \sum_{m_0, m_1} p(m_0)p(m_1)q_{f_0}(m_1). \qquad (*)$$

The first part is the prob. of success of the original A_S.

Let $F_k = \{x | f(x) = k\}$ and $\mu_k = \mathbf{Pr}\{M_S(1^n) \in F_k\}$, then split the second part by F_k:

$$\sum_k \sum_{m_0 \in F_k} \sum_{m_1} p(m_0)p(m_1)q_k(m_1) = \sum_k \left(\left(\sum_{m_1} p(m_1)q_k(m_1) \right) \sum_{m_0 \in F_k} p(m_0) \right) = \sum_k \left(\mu_k \left(\sum_{m_1} p(m_1)q_k(m_1) \right) \right)$$

Indistinguishability to semantic security (2)

Assume the contrary: for all m_0, m_1 success is unlikely:

$$p_{f_0}(m_0) - q_{f_0}(m_1) < \varepsilon.$$

Take a weighted sum: $/\!/ p(x) = \mathbf{Pr}\{M_S(1^n) = x\}$

$$\sum_{m_0, m_1} p(m_0)p(m_1)(p_{f_0}(m_0) - q_{f_0}(m_1)) \overset{\sum_{m_1} p(m_1)=1}{=\!=\!=} \sum_x p(x)p_{f(x)}(x) - \sum_{m_0, m_1} p(m_0)p(m_1)q_{f_0}(m_1). \qquad (*)$$

The first part is the prob. of success of the original A_S.

Let $F_k = \{x | f(x) = k\}$ and $\mu_k = \mathbf{Pr}\{M_S(1^n) \in F_k\}$, then split the second part by F_k:

$$\sum_k \sum_{m_0 \in F_k} \sum_{m_1} p(m_0)p(m_1)q_k(m_1) = \sum_k \left(\left(\sum_{m_1} p(m_1)q_k(m_1) \right) \sum_{m_0 \in F_k} p(m_0) \right) = \sum_k \left(\mu_k \left(\sum_{m_1} p(m_1)q_k(m_1) \right) \right)$$

. . . and it is the success prob. (on random $m_0 \leftarrow M_S$) for the following oblivious $\tilde{A}_S(h_*)$, where $h_* = h(m_0)$:

take random $m \leftarrow M_S$
encrypt m, run $A_S(E(m, \ldots), h_*)$

Thus (*) is small, contradiction.

Remark (a stronger definition)

We can assume that the oblivious adversary's circuit $\tilde{A}_S$ is generated from the normal adversary's circuit A_S in polynomial time.

Exercise

There are two more things that we can change in the definitions:

- Uniform (algorithms) vs non-uniform (circuits) adversaries (including samplers).
- Messages are generated by an adversary or the system should be secure for every two m_0, m_1.

How do they change the implications ($\Rightarrow$ and $\Leftarrow$) or their proof?

Exercise

We did not say whether the oblivious $\tilde{A}_S$ should get also the public key or not. Is it important?

Remark

Our definitions have been the same for public- and private-key settings so far.

Is it enough for our security?

Multiple messages

Attempted attack:

- E has been used for two messages 0^t and 1^t

$$\begin{array}{rcccc} E_x(0^t) & = & (f^t(x), & B(x) \oplus 0, & B(f(x)) \oplus 0, \quad \ldots) \\ E_y(1^t) & = & (f^t(y), & B(y) \oplus 1, & B(f(y)) \oplus 1, \quad \ldots) \end{array}$$

Any information can be extracted from $E_x(0^t) \oplus E_y(1^t)$?

Is it enough for our security?

Multiple messages

Attempted attack:

- E has been used for two messages 0^t and 1^t

$$\begin{array}{rcllll} E_x(0^t) & = & (f^t(x), & B(x) \oplus 0, & B(f(x)) \oplus 0, & \ldots) \\ E_y(1^t) & = & (f^t(y), & B(y) \oplus 1, & B(f(y)) \oplus 1, & \ldots) \end{array}$$

  Any information can be extracted from $E_x(0^t) \oplus E_y(1^t)$?

- If someone manages to see two encrypted messages for $x = y$ (for example, because the random source for x, y was *not quite uniform and independent*):

Happened in the past!

$$\begin{array}{rcllll} E_x(0^t) & = & (f^t(x), & B(x) \oplus 0, & B(f(x)) \oplus 0, & \ldots) \\ E_x(1^t) & = & (f^t(x), & B(x) \oplus 1, & B(f(x)) \oplus 1, & \ldots) \\ E_x(0^t) \oplus E_x(1^t) & = & (0, & B(x), & B(f(x)), & \ldots) \end{array}$$

  Can many attempts harm the PRG idea?

Is it enough for our security?

Multiple messages

Attempted attack:

- If someone manages to see two encrypted messages for $x = y$ (for example, because the random source for x, y was *not quite uniform and independent*):

Happened in the past!

$$\begin{array}{rcllll} E_x(0^t) & = & (f^t(x), & B(x) \oplus 0, & B(f(x)) \oplus 0, & \ldots) \\ E_x(1^t) & = & (f^t(x), & B(x) \oplus 1, & B(f(x)) \oplus 1, & \ldots) \\ E_x(0^t) \oplus E_x(1^t) & = & (0, & B(x), & B(f(x)), & \ldots) \end{array}$$

Can many attempts harm the PRG idea?

- For our PKCS, no. If someone can distinguish sequences,

$$E(m_1), \quad E(m_2), \quad \ldots, \quad E(m_{i^*}), \quad \ldots, \quad E(m_k)$$

$$E(s_1), \quad E(s_2), \quad \ldots, \quad E(s_{i^*}), \quad \ldots, \quad E(s_k)$$

Is it enough for our security?

Multiple messages

Attempted attack:

- If someone manages to see two encrypted messages for $x = y$ (for example, because the random source for x, y was *not quite uniform and independent*):

Happened in the past!

$$\begin{array}{rcllll} E_x(0^t) & = & (f^t(x), & B(x) \oplus 0, & B(f(x)) \oplus 0, & \ldots) \\ E_x(1^t) & = & (f^t(x), & B(x) \oplus 1, & B(f(x)) \oplus 1, & \ldots) \\ E_x(0^t) \oplus E_x(1^t) & = & (0, & B(x), & B(f(x)), & \ldots) \end{array}$$

Can many attempts harm the PRG idea?

- For our PKCS, no. If someone can distinguish sequences,

$$\begin{array}{ccccccc} E(m_1), & E(m_2), & \ldots, & E(m_{i^*}), & \ldots, & E(m_k) \\ & & & \ldots & & \\ E(m_1), & E(m_2), & \ldots, & E(m_{i^*}), & \ldots, & E(s_k) \\ E(m_1), & E(m_2), & \ldots, & E(s_{i^*}), & \ldots, & E(s_k) \\ & & & \ldots & & \\ E(s_1), & E(s_2), & \ldots, & E(s_{i^*}), & \ldots, & E(s_k) \end{array}$$

then one can distinguish sequences of almost similar messages, there is a crucial transition point i^*.

Is it enough for our security?

Multiple messages

Attempted attack:

- If someone manages to see two encrypted messages for $x = y$ (for example, because the random source for x, y was *not quite uniform and independent*):

  Happened in the past!

$$\begin{array}{rcllll} E_x(0^t) & = & (f^t(x), & B(x) \oplus 0, & B(f(x)) \oplus 0, & \ldots) \\ E_x(1^t) & = & (f^t(x), & B(x) \oplus 1, & B(f(x)) \oplus 1, & \ldots) \\ E_x(0^t) \oplus E_x(1^t) & = & (0, & B(x), & B(f(x)), & \ldots) \end{array}$$

  Can many attempts harm the PRG idea?

- For our PKCS, no. If someone can distinguish sequences,

$$\begin{array}{ccccccc} E(m_1), & E(m_2), & \ldots, & E(m_{i^*}), & \ldots, & E(m_k) \\ & & & \ldots & & \\ E(m_1), & E(m_2), & \ldots, & E(???_1), & \ldots, & E(s_k) \\ E(m_1), & E(m_2), & \ldots, & E(???_2), & \ldots, & E(s_k) \\ & & & \ldots & & \\ E(s_1), & E(s_2), & \ldots, & E(s_{i^*}), & \ldots, & E(s_k) \end{array}$$

  then one can distinguish sequences of almost similar messages, there is a crucial transition point i^*.

- Now generate everything ourselves (using the public key) and distinguish $E(???_1), E(???_2)$ given to us.

Is it enough for our security?

Multiple messages

Attempted attack:

- If someone manages to see two encrypted messages for $x = y$ (for example, because the random source for x, y was *not quite uniform and independent*):

  Happened in the past!

$$\begin{array}{rcllll} E_x(0^t) & = & (f^t(x), & B(x) \oplus 0, & B(f(x)) \oplus 0, & \ldots) \\ E_x(1^t) & = & (f^t(x), & B(x) \oplus 1, & B(f(x)) \oplus 1, & \ldots) \\ E_x(0^t) \oplus E_x(1^t) & = & (0, & B(x), & B(f(x)), & \ldots) \end{array}$$

  Can many attempts harm the PRG idea?

- For our PKCS, no. If someone can distinguish sequences,

$$\begin{array}{lllll} E(m_1), & E(m_2), & \ldots, & E(m_{i^*}), & \ldots, & E(m_k) \\ & & & \ldots & & \\ E(m_1), & E(m_2), & \ldots, & E(???_1), & \ldots, & E(s_k) \\ E(m_1), & E(m_2), & \ldots, & E(???_2), & \ldots, & E(s_k) \\ & & & \ldots & & \\ E(s_1), & E(s_2), & \ldots, & E(s_{i^*}), & \ldots, & E(s_k) \end{array}$$

  then one can distinguish sequences of almost similar messages, there is a crucial transition point i^*.

- Now generate everything ourselves (using the public key) and distinguish $E(???_1), E(???_2)$ given to us.
- True for public-key encryption only (Eve can't encrypt in the private-key setting).

Private-key (Symmetric-key) encryption

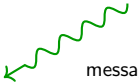
Alice

shared secret key

encrypted message(s)

Bob

message(s), shared secret key



Eve

Definition

Indistinguishable or Semantically secure

Symmetric-key encryption scheme is secure in the multiple-messages setting if

[. . . all the same words as for a single message but . . .]

an adversary is given a sequence of messages (generated by M_S) encrypted with the same key

[. . . and is asked to break them all: f and h depend on all the messages]

Symmetric-key cryptography using owp / PRG: How NOT to do it

. . . and why it is important in practice

- An "old spy" idea: XOR each bit with a bit from the secret book.

Symmetric-key cryptography using owp / PRG: How NOT to do it

. . . and why it is important in practice

- An "old spy" idea: XOR each bit with a bit from the secret book.
- Now a book can be compressed using a PRG: it can generate it from a short seed.

Symmetric-key cryptography using owp / PRG: How NOT to do it

. . . and why it is important in practice

- An "old spy" idea: XOR each bit with a bit from the secret book.
- Now a book can be compressed using a PRG: it can generate it from a short seed.
- Practical: the (short) seed can be agreed upon just once using public-key cryptography.

Symmetric-key cryptography using owp / PRG: How NOT to do it

. . . and why it is important in practice

- An "old spy" idea: XOR each bit with a bit from the secret book.
- Now a book can be compressed using a PRG: it can generate it from a short seed.
- Practical: the (short) seed can be agreed upon just once using public-key cryptography.
- Symmetric key `sk`: a seed for a (publicly known) PRG G.

$$E(\texttt{sk}, \texttt{msg}) = \texttt{msg} \oplus G(\texttt{sk}) = (b_1 \oplus G_1(\texttt{sk})) \circ (b_2 \oplus G_2(\texttt{sk})) \circ \ldots \circ (b_p \oplus G_p(\texttt{sk})),$$

where b_i and G_i are consequtive bits of `msg` and G.

Symmetric-key cryptography using owp / PRG: How NOT to do it

. . . and why it is important in practice

- An "old spy" idea: XOR each bit with a bit from the secret book.
- Now a book can be compressed using a PRG: it can generate it from a short seed.
- Practical: the (short) seed can be agreed upon just once using public-key cryptography.
- Symmetric key $\mathtt{sk}$: a seed for a (publicly known) PRG G.

$$E(\mathtt{sk}, \mathtt{msg}) \;=\; \mathtt{msg} \oplus G(\mathtt{sk}) \;=\; (b_1 \oplus G_1(\mathtt{sk})) \circ (b_2 \oplus G_2(\mathtt{sk})) \circ \ldots \circ (b_p \oplus G_p(\mathtt{sk})),$$

  where b_i and G_i are consequtive bits of $\mathtt{msg}$ and G.
- Attack: it happens that

$$\mathtt{msg} = E(\mathtt{sk}, 0^p) \oplus E(\mathtt{sk}, \mathtt{msg}).$$

Symmetric-key cryptography using owp / PRG: How NOT to do it

. . . and why it is important in practice

- An "old spy" idea: XOR each bit with a bit from the secret book.
- Now a book can be compressed using a PRG: it can generate it from a short seed.
- Practical: the (short) seed can be agreed upon just once using public-key cryptography.
- Symmetric key $\mathtt{sk}$: a seed for a (publicly known) PRG G.

$$E(\mathtt{sk}, \mathtt{msg}) = \mathtt{msg} \oplus G(\mathtt{sk}) = (b_1 \oplus G_1(\mathtt{sk})) \circ (b_2 \oplus G_2(\mathtt{sk})) \circ \ldots \circ (b_p \oplus G_p(\mathtt{sk})),$$

  where b_i and G_i are consequtive bits of $\mathtt{msg}$ and G.
- Attack: it happens that

$$\mathtt{msg} = E(\mathtt{sk}, 0^p) \oplus E(\mathtt{sk}, \mathtt{msg}).$$

- Even worse, if we know the ciphertext $E(\mathtt{sk}, \mathtt{msg})$ for any known message $\mathtt{msg}$:

$$G(\mathtt{sk}) = E(\mathtt{sk}, \mathtt{msg}) \oplus \mathtt{msg}.$$

Symmetric-key cryptography using owp / PRG: How NOT to do it

. . . and why it is important in practice

- An "old spy" idea: XOR each bit with a bit from the secret book.
- Now a book can be compressed using a PRG: it can generate it from a short seed.
- Practical: the (short) seed can be agreed upon just once using public-key cryptography.
- Symmetric key $\mathtt{sk}$: a seed for a (publicly known) PRG G.

$$E(\mathtt{sk}, \mathtt{msg}) = \mathtt{msg} \oplus G(\mathtt{sk}) = (b_1 \oplus G_1(\mathtt{sk})) \circ (b_2 \oplus G_2(\mathtt{sk})) \circ \ldots \circ (b_p \oplus G_p(\mathtt{sk})),$$

  where b_i and G_i are consequtive bits of $\mathtt{msg}$ and G.
- Attack: it happens that

$$\mathtt{msg} = E(\mathtt{sk}, 0^p) \oplus E(\mathtt{sk}, \mathtt{msg}).$$

- Even worse, if we know the ciphertext $E(\mathtt{sk}, \mathtt{msg})$ for any known message $\mathtt{msg}$:

$$G(\mathtt{sk}) = E(\mathtt{sk}, \mathtt{msg}) \oplus \mathtt{msg}.$$

- Easy to break — complete the details (exercise).

Exercise

Show formally that this system is not semantically secure for multiple messages.

Sometimes it suffices: A state-based stream cipher

- Assume that we have indeed two participants and most messages are not lost.

Sometimes it suffices: A state-based stream cipher

- Assume that we have indeed two participants and most messages are not lost.
- Let $G\colon \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG.
- Choose $s \leftarrow U_n$. This is a permanent key.
- Maintain a counter and change the "book" fragment accordingly.

Sometimes it suffices: A state-based stream cipher

- Assume that we have indeed two participants and most messages are not lost.
- Let $G\colon \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG.
- Choose $s \leftarrow U_n$. This is a permanent key.
- Maintain a counter and change the "book" fragment accordingly.
- Initial state: $(0, s)$.
- State $(i, s') \mapsto$ Next state $(i+1, G(s')_{1..n})$.

Sometimes it suffices: A state-based stream cipher

- Assume that we have indeed two participants and most messages are not lost.
- Let $G\colon \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG.
- Choose $s \leftarrow U_n$. This is a permanent key.
- Maintain a counter and change the "book" fragment accordingly.
- Initial state: $(0, s)$.
- State $(i, s') \mapsto$ Next state $(i+1, G(s')_{1..n})$.
- The last bit $b_{n+1} := G(s')_{n+1}$ is used to encrypt, $E(b) = b_{n+1} \oplus b$.

Sometimes it suffices: A state-based stream cipher

- Assume that we have indeed two participants and most messages are not lost.
- Let $G\colon \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG.
- Choose $s \leftarrow U_n$. This is a permanent key.
- Maintain a counter and change the "book" fragment accordingly.
- Initial state: $(0, s)$.
- State $(i, s') \mapsto$ Next state $(i+1, G(s')_{1..n})$.
- The last bit $b_{n+1} := G(s')_{n+1}$ is used to encrypt, $E(b) = b_{n+1} \oplus b$.
- If messages are lost, it takes proportional time to recover s' by repeating G.

Exercise

Prove that this scheme is secure.

Pseudorandom functions

IDEA:

- Prove that your construction is secure from a truly random function.

Pseudorandom functions

IDEA:

- Prove that your construction is secure from a truly random function.
- Generate a function f_ζ that looks random.
- Prove that it is indistinguishable from a truly random function.

Pseudorandom functions

IDEA:

- Prove that your construction is secure from a truly random function.
- Generate a function f_ζ that looks random.
- Prove that it is indistinguishable from a truly random function.
- Your construction should not distinguish it as well $\rightarrow$ you can use f_ζ instead.

Pseudorandom functions

Definition (Pseudorandom function family, prff)

A prff is a family of functions $\{f_\zeta\}_\zeta$, where $f_\zeta \colon \{0,1\}^n \to \{0,1\}^n$, computed by Boolean circuits generated by a polynomial-time computable algorithm $Z \colon (1^n, r_Z) \mapsto f_\zeta$ such that f_ζ is computationally indistinguishable from a random function: for any adversary A,

$$|\mathbf{Pr}_{A,y}\{A(y,1^n) = 1 \mid y \leftarrow f_\zeta(U_n)\} - \mathbf{Pr}_{A,R,x}\{A(x,1^n) = 1 \mid x \leftarrow R(U_n)\}| < \varepsilon(n),$$

where $R \colon \{0,1\}^n \to \{0,1\}^n$ has a uniformly random truth table.

Small ζ serves as an exponentially large "code book".
We use ζ and f_ζ interchangably.

Definition (Pseudorandom function family, prff)

A prff is a family of functions $\{f_\zeta\}_\zeta$, where $f_\zeta \colon \{0,1\}^n \to \{0,1\}^n$, computed by Boolean circuits generated by a polynomial-time computable algorithm $Z \colon (1^n, r_Z) \mapsto f_\zeta$ such that f_ζ is computationally indistinguishable from a random function: for any adversary A,

$$|\mathbf{Pr}_{A,y}\{A(y,1^n) = 1 \mid y \leftarrow f_\zeta(U_n)\} - \mathbf{Pr}_{A,R,x}\{A(x,1^n) = 1 \mid x \leftarrow R(U_n)\}| < \varepsilon(n),$$

where $R \colon \{0,1\}^n \to \{0,1\}^n$ has a uniformly random truth table.

Small ζ serves as an exponentially large "code book".
We use ζ and f_ζ interchangably.

Construction from PRGs

- One can construct a prff using a $2n$-PRG $G \colon \{0,1\}^n \to \{0,1\}^{2n}$.
- Split G's output into two n-bit parts $G(x) = G_0(x) \circ G_1(x)$.
- Take $s \leftarrow U_n$, compute
$$\zeta_s(b_1 b_2 \dots b_n) = G_{b_n}(G_{b_{n-1}}(\dots(G_{b_1}(s)\dots)).$$
- (Proof delayed.)

A standard definition says that Z generates an index, and $f_\zeta(x)$ is computed by a polynomial-time algorithm $F(\zeta, x)$, observe that it is equivalent to our definition.

A symmetric-key encryption scheme

Based on prff

- To encode an n-bit message, define a "proto-scheme"

$$E(\zeta, \texttt{msg}, r_E) = (\zeta(r_E) \oplus \texttt{msg}, r_E),$$

where $\zeta \leftarrow Z(1^n)$ serves as a key and $r_E \leftarrow U$ is our randomness.

A symmetric-key encryption scheme

Based on prff

- To encode an n-bit message, define a "proto-scheme"

$$E(\zeta, \texttt{msg}, r_E) = (\zeta(r_E) \oplus \texttt{msg}, r_E),$$

  where $\zeta \leftarrow Z(1^n)$ serves as a key and $r_E \leftarrow U$ is our randomness.

- To encode messages of arbitrary length, cut them into pieces:

$$E'(\zeta, \texttt{msg}, \underbrace{r_1 r_2 \ldots}_{r_E}) = (|\texttt{msg}|, E(\zeta, \texttt{msg}[1..n], r_1), E(\zeta, \texttt{msg}[n+1..2n], r_2), \ldots)$$

A symmetric-key encryption scheme

Based on prff

- To encode an n-bit message, define a "proto-scheme"

$$E(\zeta, \mathtt{msg}, r_E) = (\zeta(r_E) \oplus \mathtt{msg}, r_E),$$

  where $\zeta \leftarrow Z(1^n)$ serves as a key and $r_E \leftarrow U$ is our randomness.

- To encode messages of arbitrary length, cut them into pieces:

$$E'(\zeta, \mathtt{msg}, \underbrace{r_1 r_2 \ldots}_{r_E}) = (|\mathtt{msg}|, E(\zeta, \mathtt{msg}[1..n], r_1), E(\zeta, \mathtt{msg}[n+1..2n], r_2), \ldots)$$

- E' reuses the key, we need to show that E is secure in the multiple-messages setting.

A symmetric-key encryption scheme

Based on prff

- To encode an n-bit message, define a "proto-scheme"

$$E(\zeta, \mathtt{msg}, r_E) = (\zeta(r_E) \oplus \mathtt{msg}, r_E),$$

where $\zeta \leftarrow Z(1^n)$ serves as a key and $r_E \leftarrow U$ is our randomness.

- To encode messages of arbitrary length, cut them into pieces:

$$E'(\zeta, \mathtt{msg}, \underbrace{r_1 r_2 \ldots}_{r_E}) = (|\mathtt{msg}|, E(\zeta, \mathtt{msg}[1..n], r_1), E(\zeta, \mathtt{msg}[n+1..2n], r_2), \ldots)$$

- E' reuses the key, we need to show that E is secure in the multiple-messages setting.
- If an adversary distinguishes

$$\{(\zeta(r_i) \oplus \mathtt{msg}^{(i)}, r_i)\}_i \text{ and } \{(\zeta(r'_i) \oplus \underset{\text{random}}{\rho^{(i)}}, r'_i)\}_i$$

then we can distinguish either

$$\zeta \text{ and } R$$

or

$$\{(R(r_i) \oplus \mathtt{msg}^{(i)}, r_i)\}_i \text{ and } \{(R(r'_i) \oplus \underset{\text{random}}{\rho^{(i)}}, r'_i)\}_i$$

A symmetric-key encryption scheme

Based on prff

- To encode an n-bit message, define a "proto-scheme"

$$E(\zeta, \mathtt{msg}, r_E) = (\zeta(r_E) \oplus \mathtt{msg}, r_E),$$

where $\zeta \leftarrow Z(1^n)$ serves as a key and $r_E \leftarrow U$ is our randomness.

- To encode messages of arbitrary length, cut them into pieces:

$$E'(\zeta, \mathtt{msg}, \underbrace{r_1 r_2 \ldots}_{r_E}) = (|\mathtt{msg}|, E(\zeta, \mathtt{msg}[1..n], r_1), E(\zeta, \mathtt{msg}[n+1..2n], r_2), \ldots)$$

- E' reuses the key, we need to show that E is secure in the multiple-messages setting.

- If an adversary distinguishes

$$\{(\zeta(r_i) \oplus \mathtt{msg}^{(i)}, r_i)\}_i \text{ and } \{(\zeta(r'_i) \oplus \underset{\text{random}}{\rho^{(i)}}, r'_i)\}_i$$

then we can distinguish either

$$\zeta \text{ and } R$$

or

$$\{(R(r_i) \oplus \mathtt{msg}^{(i)}, r_i)\}_i \text{ and } \{(R(r'_i) \oplus \underset{\text{random}}{\rho^{(i)}}, r'_i)\}_i$$

The latter two are indistinguishable: with prob. $\geqslant 1 - t^2 \cdot 2^{-n}$ all r_i's are distinct and thus $R(r_i) \oplus \ldots$ are distributed uniformly and independently.

Symmetric-key schemes: how to exchange the key?

- (Temporary) secure channel: just send a key.

Symmetric-key schemes: how to exchange the key?

- (Temporary) secure channel: just send a key.
- Insecure trusted channel (Eve, passive adversary): authenticated key-exchange protocol.
  - Bob generates a symmetric key and uses (unrelated) public-key encryption to send it to Alice.

Symmetric-key schemes: how to exchange the key?

- (Temporary) secure channel: just send a key.
- Insecure trusted channel (Eve, passive adversary): authenticated key-exchange protocol.
  - Bob generates a symmetric key and uses (unrelated) public-key encryption to send it to Alice.
  - Dual-side key-exchange protocol (both sides influence the key):

Diffie-Hellman key-exchange protocol

- Prime $p \in \mathbb{P}$, generator g: $\mathbb{Z}_p^* = \{g^k\}_k$ (public).

Symmetric-key schemes: how to exchange the key?

- (Temporary) secure channel: just send a key.
- Insecure trusted channel (Eve, passive adversary): authenticated key-exchange protocol.
  - Bob generates a symmetric key and uses (unrelated) public-key encryption to send it to Alice.
  - Dual-side key-exchange protocol (both sides influence the key):

Diffie-Hellman key-exchange protocol

- Prime $p \in \mathbb{P}$, generator g: $\mathbb{Z}_p^* = \{g^k\}_k$ (public).
- Alice chooses a and sends $g^a \mod p$.
- Bob chooses b and sends $g^b \mod p$.
- Now they both have $g^{ab} \equiv (g^a)^b \equiv (g^b)^a \mod p$ (their private key).

Symmetric-key schemes: how to exchange the key?

- (Temporary) secure channel: just send a key.
- Insecure trusted channel (Eve, passive adversary): authenticated key-exchange protocol.
  - Bob generates a symmetric key and uses (unrelated) public-key encryption to send it to Alice.
  - Dual-side key-exchange protocol (both sides influence the key):

Diffie-Hellman key-exchange protocol

- Prime $p \in \mathbb{P}$, generator g: $\mathbb{Z}_p^* = \{g^k\}_k$ (public).
- Alice chooses a and sends $g^a \mod p$.
- Bob chooses b and sends $g^b \mod p$.
- Now they both have $g^{ab} \equiv (g^a)^b \equiv (g^b)^a \mod p$ (their private key).
- Assumption: even given h, it is infeasible to check that $h = g^{ab} \mod p$ from public info.

Symmetric-key schemes: how to exchange the key?

- (Temporary) secure channel: just send a key.
- Insecure trusted channel (Eve, passive adversary): authenticated key-exchange protocol.
  - Bob generates a symmetric key and uses (unrelated) public-key encryption to send it to Alice.
  - Dual-side key-exchange protocol (both sides influence the key):

  Diffie-Hellman key-exchange protocol
  - Prime $p \in \mathbb{P}$, generator g: $\mathbb{Z}_p^* = \{g^k\}_k$ (public).
  - Alice chooses a and sends $g^a \mod p$.
  - Bob chooses b and sends $g^b \mod p$.
  - Now they both have $g^{ab} \equiv (g^a)^b \equiv (g^b)^a \mod p$ (their private key).
  - Assumption: even given h, it is infeasible to check that $h = g^{ab} \mod p$ from public info.
- Untrusted channel (Eve, passive adversary): unauthenticated key-exchange protocol.

Symmetric-key schemes: how to exchange the key?

- (Temporary) secure channel: just send a key.
- Insecure trusted channel (Eve, passive adversary): authenticated key-exchange protocol.
  - Bob generates a symmetric key and uses (unrelated) public-key encryption to send it to Alice.
  - Dual-side key-exchange protocol (both sides influence the key):

  Diffie-Hellman key-exchange protocol
  - Prime $p \in \mathbb{P}$, generator g: $\mathbb{Z}_p^* = \{g^k\}_k$ (public).
  - Alice chooses a and sends $g^a \mod p$.
  - Bob chooses b and sends $g^b \mod p$.
  - Now they both have $g^{ab} \equiv (g^a)^b \equiv (g^b)^a \mod p$ (their private key).
  - Assumption: even given h, it is infeasible to check that $h = g^{ab} \mod p$ from public info.
- Untrusted channel (Eve, passive adversary): unauthenticated key-exchange protocol.
- Untrusted channel is a problem for public-key cryptography as well (requires certification).

Possible practical (not purely mathematical) attacks on encryption

Unencrypted message

- CPA: Chosen Plaintext Attack.
  - Adversary can encrypt.
  - Does not add power in public-key cryptography.

Possible practical (not purely mathematical) attacks on encryption

Unencrypted message

- CPA: Chosen Plaintext Attack.
  - Adversary can encrypt.
  - Does not add power in public-key cryptography.

Encrypted message

- CCA: Chosen Ciphertext Attack.
  - Adversary can decrypt (everything but the target message).
  - Either before (a priori) or after (a posteriori) getting the target ciphertext.
    - The prff-base construction is already secure under a-priori CCA, can be made secure under a-posteriori CCA.
    - For public-key encryption, additional work is required.

Possible practical (not purely mathematical) attacks on encryption

Unencrypted message

- CPA: Chosen Plaintext Attack.
  - Adversary can encrypt.
  - Does not add power in public-key cryptography.

Encrypted message

- CCA: Chosen Ciphertext Attack.
  - Adversary can decrypt (everything but the target message).
  - Either before (a priori) or after (a posteriori) getting the target ciphertext.
    - The prff-base construction is already secure under a-priori CCA, can be made secure under a-posteriori CCA.
    - For public-key encryption, additional work is required.
- There are other nuances, e.g., are we attacking a specific key?

Possible practical (not purely mathematical) attacks on encryption

Unencrypted message

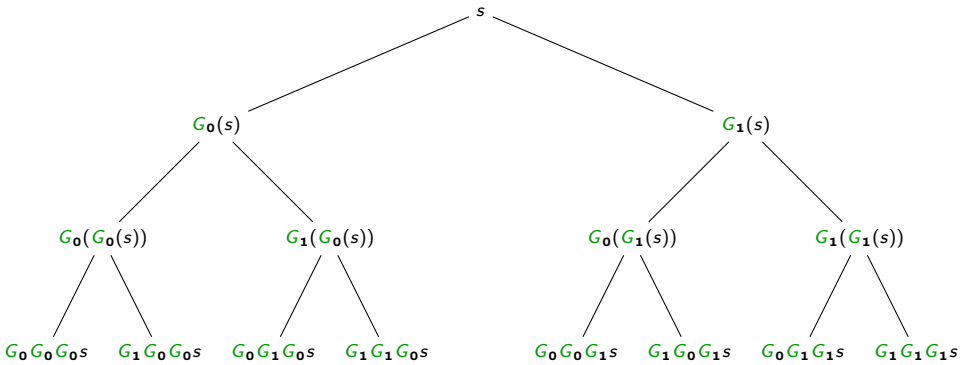
- CPA: Chosen Plaintext Attack.
  - Adversary can encrypt.
  - Does not add power in public-key cryptography.

Encrypted message

- CCA: Chosen Ciphertext Attack.
  - Adversary can decrypt (everything but the target message).
  - Either before (a priori) or after (a posteriori) getting the target ciphertext.
    - The prff-base construction is already secure under a-priori CCA, can be made secure under a-posteriori CCA.
    - For public-key encryption, additional work is required.
- There are other nuances, e.g., are we attacking a specific key?
- More practical issues: leakage due to interference with other programs.

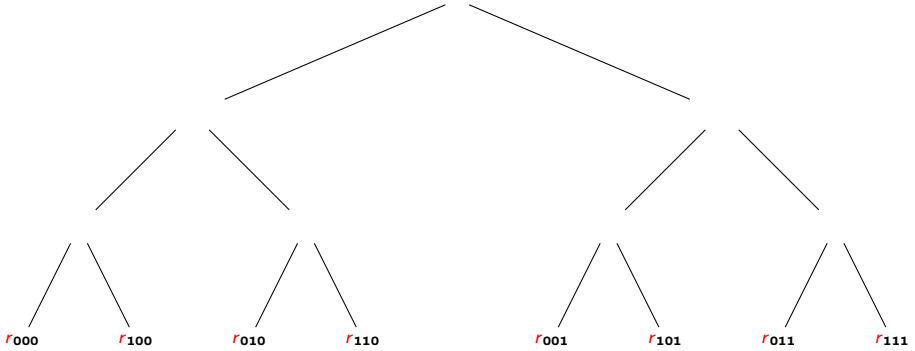
Prff from PRG: Why it works

Distinguish $(x_1^\ell, x_1^r), (x_2^\ell, x_2^r), \ldots, (x_p^\ell, x_p^r)$ from truly random source using a distinguisher D of $\zeta_s(b_1 b_2 \ldots b_n) = G_{b_n}(G_{b_{n-1}}(\ldots(G_{b_2}(G_{b_1}(\zeta))\ldots))$ from $R(b_1 b_2 \ldots b_n)$



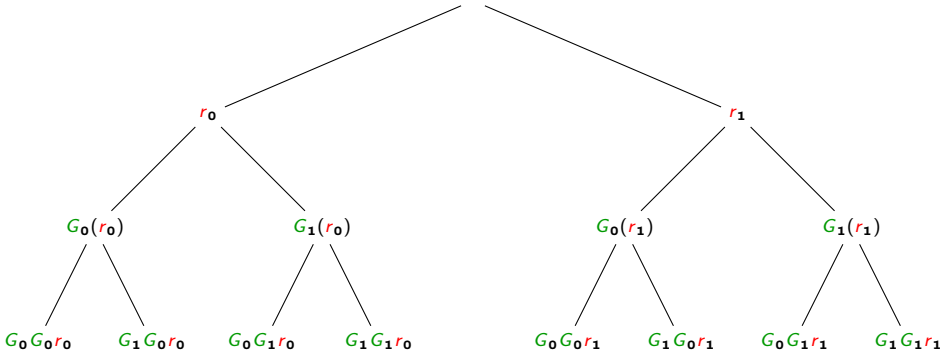
Prff from PRG: Why it works

Distinguish $(x_1^\ell, x_1^r), (x_2^\ell, x_2^r), \ldots, (x_p^\ell, x_p^r)$ from truly random source using a distinguisher D of $\zeta_s(b_1 b_2 \ldots b_n) = G_{b_n}(G_{b_{n-1}}(\ldots(G_{b_2}(G_{b_1}(\zeta)))\ldots))$ from $R(b_1 b_2 \ldots b_n)$



Prff from PRG: Why it works

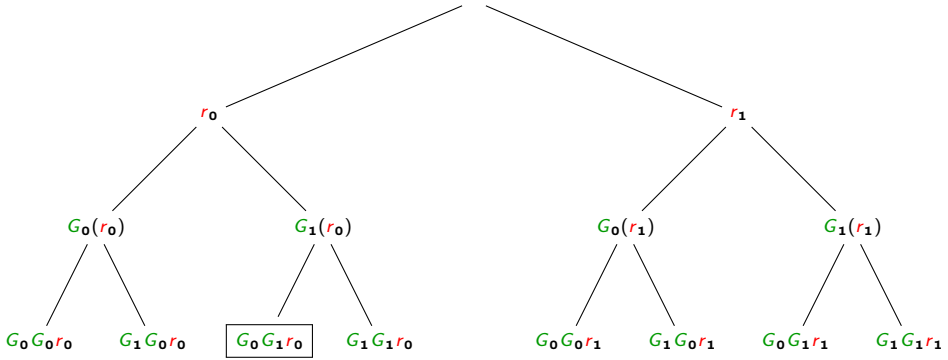
Distinguish $(x_1^\ell, x_1^r), (x_2^\ell, x_2^r), \ldots, (x_p^\ell, x_p^r)$ from truly random source using a distinguisher D of $\zeta_s(b_1 b_2 \ldots b_n) = G_{b_n}(G_{b_{n-1}}(\ldots(G_{b_2}(G_{b_1}(\zeta))\ldots))$ from $R(b_1 b_2 \ldots b_n)$



Prff from PRG: Why it works

Distinguish $(x_1^\ell, x_1^r), (x_2^\ell, x_2^r), \ldots, (x_p^\ell, x_p^r)$ from truly random source using a distinguisher D of $\zeta_s(b_1 b_2 \ldots b_n) = G_{b_n}(G_{b_{n-1}}(\ldots(G_{b_2}(G_{b_1}(\zeta))\ldots))$ from $R(b_1 b_2 \ldots b_n)$

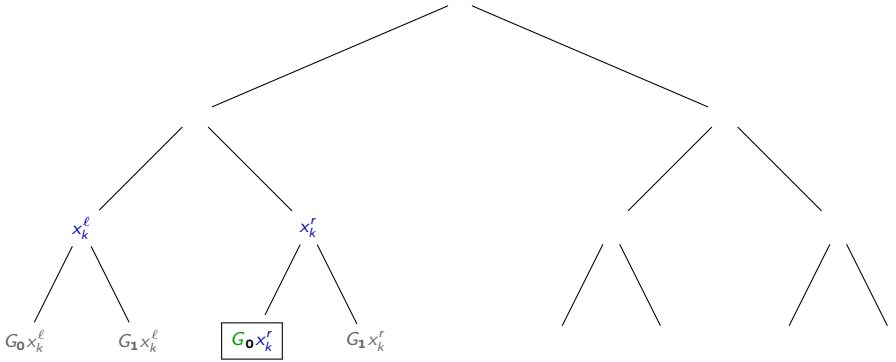
Suppose D asks for $\boxed{f(010)}\ldots$



Prff from PRG: Why it works

Distinguish $(x_1^\ell, x_1^r), (x_2^\ell, x_2^r), \ldots, (x_p^\ell, x_p^r)$ from truly random source using a distinguisher D of $\zeta_s(b_1 b_2 \ldots b_n) = G_{b_n}(G_{b_{n-1}}(\ldots(G_{b_2}(G_{b_1}(\zeta)))\ldots))$ from $R(b_1 b_2 \ldots b_n)$

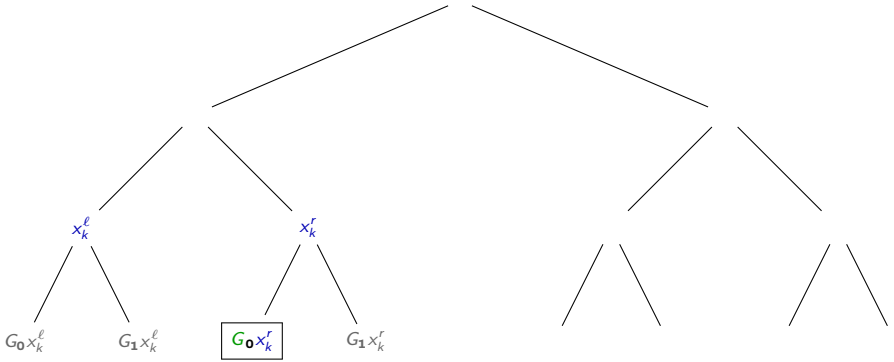
Suppose D asks for $\boxed{f(010)}\ldots$



Prff from PRG: Why it works

Distinguish $(x_1^\ell, x_1^r), (x_2^\ell, x_2^r), \ldots, (x_p^\ell, x_p^r)$ from truly random source using a distinguisher D of $\zeta_s(b_1 b_2 \ldots b_n) = G_{b_n}(G_{b_{n-1}}(\ldots(G_{b_2}(G_{b_1}(\zeta)))\ldots))$ from $R(b_1 b_2 \ldots b_n)$

Suppose D asks for $\boxed{f(010)}\ldots$



Note that a PRG stays secure if queried polynomially many times.

Takeout

- Indistinguishability implies semantic security — a proof.
- Symmetric-key encryption based on pseudorandom function families.
- Diffie–Hellman key-exchange protocol.
- Prff from PRG.

Coming next:

- *Digital signatures.*