Private-Key Encryption

Ali El Kaafarani

[1]Mathematical Institute
[2] PQShield Ltd.

Outline

Outline

Pseudo-Random Generators PRGs

- *Pseudorandomness* is a property of a distribution on strings. Say you have a distribution X on ℓ-bit strings that assigns some probability to every string in $\{0,1\}^{\ell}$. Pseudorandomness means that sampling form X is indistinguishable from sampling a uniform string of length ℓ.
- Ideally, we want a PRG to efficiently produce, from short seeds, long sequences of bits that appear to be generated by successive flips of a fair coin.
- Unpredictability is a very important property of sequences of coin tosses. Pseudo-random sequences should be unpredictable to computers with feasible resources. Given a sequence of $(n-1)$ bits, can you guess the n^{th} bit with a probability better than $1/2$?

Pseudo-Random Generators PRGs

- *Pseudorandomness* is a property of a distribution on strings. Say you have a distribution X on ℓ-bit strings that assigns some probability to every string in $\{0,1\}^{\ell}$. Pseudorandomness means that sampling form X is indistinguishable from sampling a uniform string of length ℓ.
- Ideally, we want a PRG to efficiently produce, from short seeds, long sequences of bits that appear to be generated by successive flips of a fair coin.
- Unpredictability is a very important property of sequences of coin tosses. Pseudo-random sequences should be unpredictable to computers with feasible resources. Given a sequence of $(n-1)$ bits, can you guess the n^{th} bit with a probability better than $1/2$?

Pseudo-Random Generators PRGs

- *Pseudorandomness* is a property of a distribution on strings. Say you have a distribution X on ℓ-bit strings that assigns some probability to every string in $\{0,1\}^{\ell}$. Pseudorandomness means that sampling form X is indistinguishable from sampling a uniform string of length ℓ.
- Ideally, we want a PRG to efficiently produce, from short seeds, long sequences of bits that appear to be generated by successive flips of a fair coin.
- Unpredictability is a very important property of sequences of coin tosses. Pseudo-random sequences should be unpredictable to computers with feasible resources. Given a sequence of $(n-1)$ bits, can you guess the n^{th} bit with a probability better than $1/2$?

Pseudo-Random Generators PRGs

- A PRG is an efficient deterministic algorithm that expands a short, uniform seed into a longer, "uniform-looking" output.
- Informally, a PRG is cryptographically secure, if it passes all efficient statistical tests.

Definition

Let $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ a deterministic polynomial-time algorithm where $\ell(n) > n$. G is a secure pseudorandom generator if $\forall$ probabilistic poly-time distinguisher (also called statistical test) D, the advantage

$$\mathrm{Adv}^{\mathrm{prg}}_{G,D}(n) = |\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq \mathsf{negl}(n)$$

where the probabilities are taken over uniform choice of $s \in \{0,1\}^n$, $r \in \{0,1\}^{\ell(n)}$ and the randomness of D.

Pseudo-Random Generators PRGs

- A PRG is an efficient deterministic algorithm that expands a short, uniform seed into a longer, "uniform-looking" output.
- Informally, a PRG is cryptographically secure, if it passes all efficient statistical tests.

Definition

Let $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ a deterministic polynomial-time algorithm where $\ell(n) > n$. G is a secure pseudorandom generator if $\forall$ probabilistic poly-time distinguisher (also called statistical test) D, the advantage

$$\mathrm{Adv}^{\mathrm{prg}}_{G,D}(n) = |\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq \mathsf{negl}(n)$$

where the probabilities are taken over uniform choice of $s \in \{0,1\}^n$, $r \in \{0,1\}^{\ell(n)}$ and the randomness of D.

Pseudo-Random Generators PRGs

- A PRG is an efficient deterministic algorithm that expands a short, uniform seed into a longer, "uniform-looking" output.
- Informally, a PRG is cryptographically secure, if it passes all efficient statistical tests.

Definition

Let $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ *a deterministic polynomial-time algorithm where* $\ell(n) > n$. G *is a secure pseudorandom generator if* $\forall$ *probabilistic poly-time distinguisher (also called statistical test)* D*, the advantage*

$$\mathsf{Adv}^{\mathrm{prg}}_{G,D}(n) = |\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq \mathsf{negl}(n)$$

where the probabilities are taken over uniform choice of $s \in \{0,1\}^n$, $r \in \{0,1\}^{\ell(n)}$ *and the randomness of* D.

Pseudo-Random Generators

- Do PRGs exist? Can we construct them?
- Not if $NP = P$
- What is the weakest assumption under which we can construct PRGs?
- It is the existence of *one-way functions* (i.e. easy to compute-hard to invert)

Pseudo-Random Generators

- Do PRGs exist? Can we construct them?
- Not if $NP = P$
- What is the weakest assumption under which we can construct PRGs?
- It is the existence of *one-way functions* (i.e. easy to compute-hard to invert)

Pseudo-Random Generators

- Do PRGs exist? Can we construct them?
- Not if $NP = P$
- What is the weakest assumption under which we can construct PRGs?
- It is the existence of *one-way functions* (i.e. easy to compute-hard to invert)

Pseudo-Random Generators

- Do PRGs exist? Can we construct them?
- Not if $NP = P$
- What is the weakest assumption under which we can construct PRGs?
- It is the existence of *one-way functions* (i.e. easy to compute-hard to invert)

PRG: examples

- *Linear Congruential Generator* (Non-Crypto PRG)
  - $X_{n+1} = aX_n + b \mod m$, where a, b, m are the constants and X_0 is the seed.
- Cryptographically suitable PRGs: /dev/random, Fortuna, Intel RdRand (available in Ivy Bridge processors), etc.
  - They continuously add entropy to internal state
  - Example of entropy source: Timing (hardware interrupts, e.g. keyboard, mouse, etc.).

Security of PRGs

- Attackers abilities: What they can observe or influence/manipulate in the inputs/outputs/state.
- Types of attacks:
  - Input Based Attacks (security goals: minimize the number of possible outputs, so guessing becomes easier, or force the generator to produce a particular output),
  - State Based attacks (security goals: backward/forward secrecy, i.e. predict past/future outputs).

- Remember, the goal is to “make our *secure* OTP practical”...
- OTP is perfectly secure. But $|k| \geq |m|$...!
- What if we use pseudo-random generators?
- And use a different security definition...
- (t, ϵ)-indistinguishability: We allow that the security may fail with probability $\leq \epsilon$ and we restrict attention to adversaries running in time $\leq t$.
- This is the basic idea of a new security definition called *semantic security* or *indistinguishable encryptions*!
- Perhaps we can now encrypt a 1 MB file using only 128-bit key!

- Remember, the goal is to “make our *secure* OTP practical”...
- OTP is perfectly secure. But $|k| \geq |m|$...!
- What if we use pseudo-random generators?
- And use a different security definition...
- (t, ϵ)-indistinguishability: We allow that the security may fail with probability $\leq \epsilon$ and we restrict attention to adversaries running in time $\leq t$.
- This is the basic idea of a new security definition called *semantic security* or *indistinguishable encryptions*!
- Perhaps we can now encrypt a 1 MB file using only 128-bit key!

- Remember, the goal is to “make our *secure* OTP practical”...
- OTP is perfectly secure. But $|k| \geq |m|$...!
- What if we use pseudo-random generators?
- And use a different security definition...
- (t, ϵ)-indistinguishability: We allow that the security may fail with probability $\leq \epsilon$ and we restrict attention to adversaries running in time $\leq t$.
- This is the basic idea of a new security definition called *semantic security* or *indistinguishable encryptions*!
- Perhaps we can now encrypt a 1 MB file using only 128-bit key!

- Remember, the goal is to “make our *secure* OTP practical”...
- OTP is perfectly secure. But $|k| \geq |m|$...!
- What if we use pseudo-random generators?
- And use a different security definition...
- (t, ϵ)-indistinguishability: We allow that the security may fail with probability $\leq \epsilon$ and we restrict attention to adversaries running in time $\leq t$.
- This is the basic idea of a new security definition called *semantic security* or *indistinguishable encryptions*!
- Perhaps we can now encrypt a 1 MB file using only 128-bit key!

- Remember, the goal is to “make our *secure* OTP practical”...
- OTP is perfectly secure. But $|k| \geq |m|$...!
- What if we use pseudo-random generators?
- And use a different security definition...
- (t, ϵ)-indistinguishability: We allow that the security may fail with probability $\leq \epsilon$ and we restrict attention to adversaries running in time $\leq t$.
- This is the basic idea of a new security definition called *semantic security* or *indistinguishable encryptions*!
- Perhaps we can now encrypt a 1 MB file using only 128-bit key!

- Remember, the goal is to “make our *secure* OTP practical”...
- OTP is perfectly secure. But $|k| \geq |m|$...!
- What if we use pseudo-random generators?
- And use a different security definition...
- (t, ϵ)-indistinguishability: We allow that the security may fail with probability $\leq \epsilon$ and we restrict attention to adversaries running in time $\leq t$.
- This is the basic idea of a new security definition called *semantic security* or *indistinguishable encryptions*!
- Perhaps we can now encrypt a 1 MB file using only 128-bit key!

- Remember, the goal is to “make our *secure* OTP practical”...
- OTP is perfectly secure. But $|k| \geq |m|$...!
- What if we use pseudo-random generators?
- And use a different security definition...
- (t, ϵ)-indistinguishability: We allow that the security may fail with probability $\leq \epsilon$ and we restrict attention to adversaries running in time $\leq t$.
- This is the basic idea of a new security definition called *semantic security* or *indistinguishable encryptions*!
- Perhaps we can now encrypt a 1 MB file using only 128-bit key!

Fixed-length Encryption scheme using a PRG

Let G be a pseudorandom generator with expansion factor ℓ. For messages of length ℓ, we define the following encryption scheme $E = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{KeyGen}(n)$: It randomly picks random bit string of length n, i.e. $k \in \{0,1\}^n$.
- Enc : it takes as input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, it outputs

$$c \leftarrow \mathsf{Enc}(G(k), m) = G(k) \oplus m$$

- Dec : it takes as input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, it outputs

$$m \leftarrow \mathsf{Dec}(G(k), c) = G(k) \oplus c.$$

Semantic Security

Security Game

Challenger Ch | **Adversary** A

$\xleftarrow{m_0, m_1, |m_0| = |m_1|}$

$b \leftarrow\!\!\$ \{0, 1\}$

$\xrightarrow{c = \mathsf{Enc}(k, m_b)}$ Outputs his guess b'

Definition

An encryption scheme is semantically secure if for all efficient $\mathcal{A}$ the following holds:

$$\mathsf{Adv}^{\mathrm{eav}}_{E,\mathcal{A}}(n) = \Pr[\mathsf{A}\ \textit{wins}] - 1/2 \leq \mathsf{negl}(n)$$

What about security for multiple encryptions?

Semantic Security (also called Computational indistinguishability)

More formally,

- Let us distinguish between the two cases of $b = 0$ or $b = 1$. Call the first EXP(0) and the second EXP(1).
- For $b = 0, 1$, let G_b be the event that the output of the experiment EXP(b)=1. Now

$$\mathsf{Adv}^{\mathrm{SS}}_{E,\mathcal{A}}(n) = |\Pr(G_0) - \Pr(G_1)|$$

Theorem

If G is a secure PRG, then the encryption scheme derived from G is semantically secure.

Lemma

If $\mathcal{A}$ is an adversary against a semantic secure encryption scheme E , then there exists an adversary $\mathcal{B}$ against the PRG G of E s.t.

$$\mathsf{Adv}^{\mathrm{ss}}_{E,\mathcal{A}}(n) \leq 2 \cdot \mathsf{Adv}^{\mathrm{prg}}_{G,\mathcal{B}}(n)$$

Proof.

(By reduction)

Let G_b the event that the adversary outputs $b' = 1$ if the challenger uses the PRG while encrypting and R_b is the same but when using a truly random generator.

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{ss}}_{E,\mathcal{A}}(n) &= |\Pr(G_0) - \Pr(G_1)| \\
&= |\Pr(G_0) - \Pr(R_0) + \Pr(R_0) - \Pr(R_1) \\
&\quad + \Pr(R_1) - \Pr(G_1)| \\
&\leq |\Pr(G_0) - \Pr(R_0)| + \underbrace{|\Pr(R_0) - \Pr(R_1)|}_{= 0 \text{ as OTP is sem. secure.}} \\
&\quad + |\Pr(R_1) - \Pr(G_1)| \\
&= 2\mathsf{Adv}^{\mathrm{prg}}_{G,\mathcal{B}}(n).
\end{aligned}$$

□

Stream Ciphers

- Terminology is not standard: it is either considered to be practical instantiations of pseudo-random generators or the encryption scheme which uses it.
- They produce as many random bits as exactly needed.
- They are more flexible (no upper bound on the number of bits) and efficient (each application takes the exact number of random bits that it requests)
- A stream cipher consists of two main deterministic algorithms:
- Init(s, IV): takes a seed s and an optional *initialization vector IV* and outputs an initial state st_0
- GetBits(st_i): takes the i-th state information st_i and outputs a bit y and an updated state, i.e. st_{i+1}

Stream Ciphers

Construction of a PRG G_ℓ:

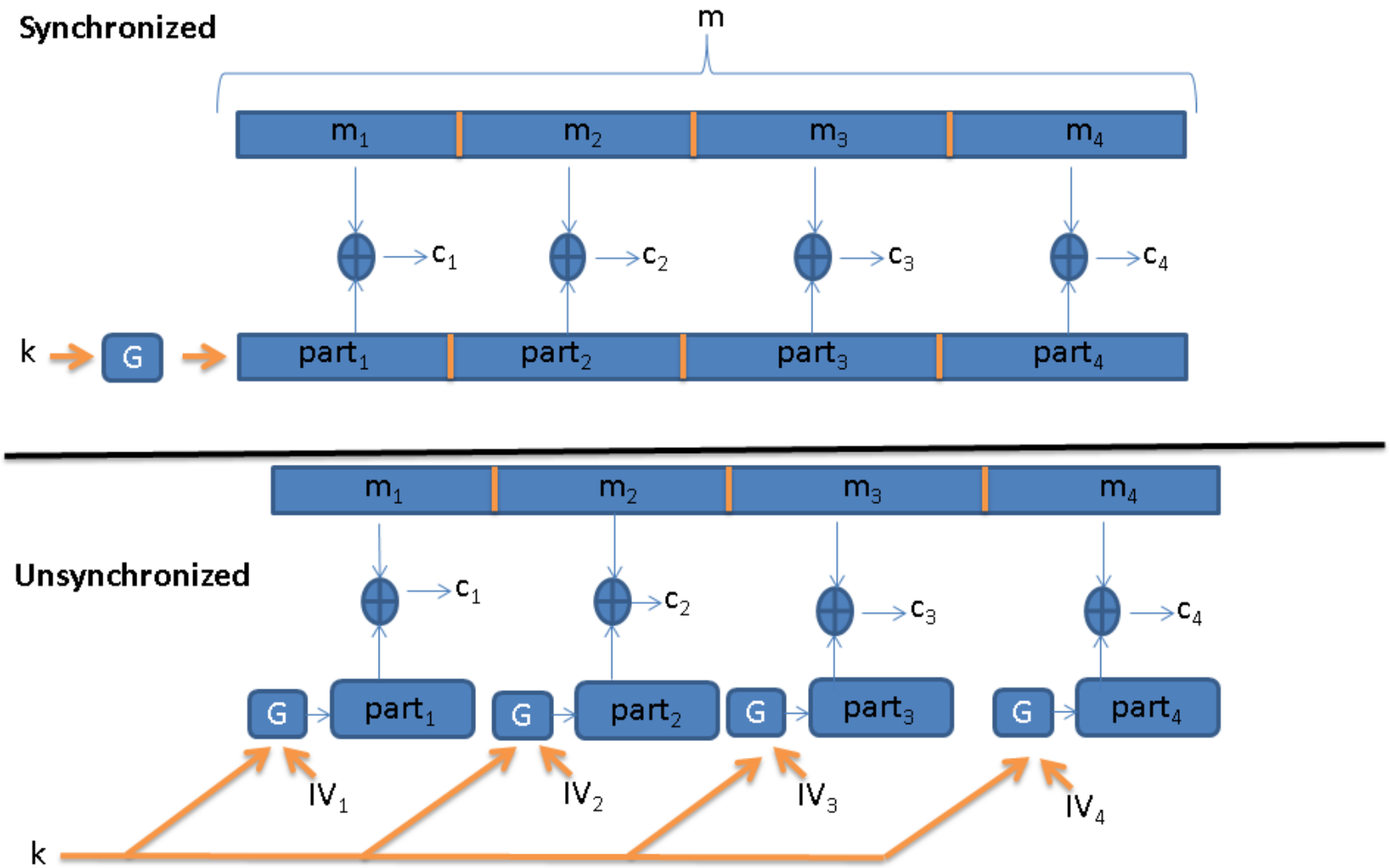
$\mathsf{st}_0 \leftarrow \mathsf{Init}(s, IV)$

for $i = 1, \cdots, \ell$;

$\quad (y_i, \mathsf{st}_i) \leftarrow \mathsf{GetBits}(\mathsf{st}_{i-1})$

return $y_1, \cdots, y_\ell$

Stream Cipher Mode of Operations



Stream Cipher Modes of Operations

- synchronized mode.
  - It gives a stateful CPA-secure encryption scheme.
  - Sender and Receiver must be synchronized.
  - It generates a long pseudo-random stream- different parts of it are used to encrypt different messages.
  - Therefore, messages should be received in order.
- unsynchronized mode.
  - It needs initialization vectors.
  - It gives *stateless* CPA-secure encryption.

Examples of Stream Ciphers

- Linear-Feedback Shift Registers (LFSR)
- RC4 by Ron Rivest 1987 (recent attack: AlFardan et al. 2013)
- eStream: Salsa 20, ChaCha (2008), and SOSEMANUK.
- eStream competition page:
  `http://competitions.cr.yp.to/estream.html`

RC4

- The state in RC4 consists of the triplet (S, i, j). S is a 256-byte array that contains a permutation of the numbers $0, \cdots, 255$. both $i, j \in \{0, \cdots, 255\}$.
- The key can be up to 256 byte long.

RC4- the Init() algorithm

Input: a 16-byte key

Output: Initial state (S, i, j)

for $i = 1, \cdots, 255$:

$S[i] \leftarrow i$ ▶ it sets S to the identity permutation

$k[i] \leftarrow k[i \mod 16]$ ▶ it expands the key to 256 bytes by repetition

$j = 0$

for $i = 1, \cdots, 255;$

$j \leftarrow j + S[i] + k[i] \mod 256$

Swap $S[i]$ and $S[j]$ ▶ "pseudo-random" swapping of S's elements

$i \leftarrow 0, j \leftarrow 0$

return (S, i, j)

RC4- the GetBits() algorithm

Input: Current state (S, i, j)

Output: byte y**, updated state** (S, i, j)

$i \leftarrow i + 1 \mod 256$

$j \leftarrow j + S[i] \mod 256$ ▶ changing j in a “pseudo-random” way

Swap $S[i]$ and $S[j]$

$t \leftarrow S[i] + S[j] \mod 256$

$y \leftarrow S[t]$

return $(S, i, j), y$

RC4: Security Analysis

- Weaknesses in its key scheduling algorithm, i.e. the Init() algorithm.
- Biases in the second output byte of RC4: the probability that it is 0 is $1/128$ instead of $1/256$ for $S = 256$.
- Biases in further bytes: *double-byte* or *adjacent-byte* biases.
- Conclusion: not secure, nevertheless, "its usage is still running at about 30% of all TLS traffic" (Garman et al. March 2015)

RC4: Security Analysis [AlFardan et al. 2013]

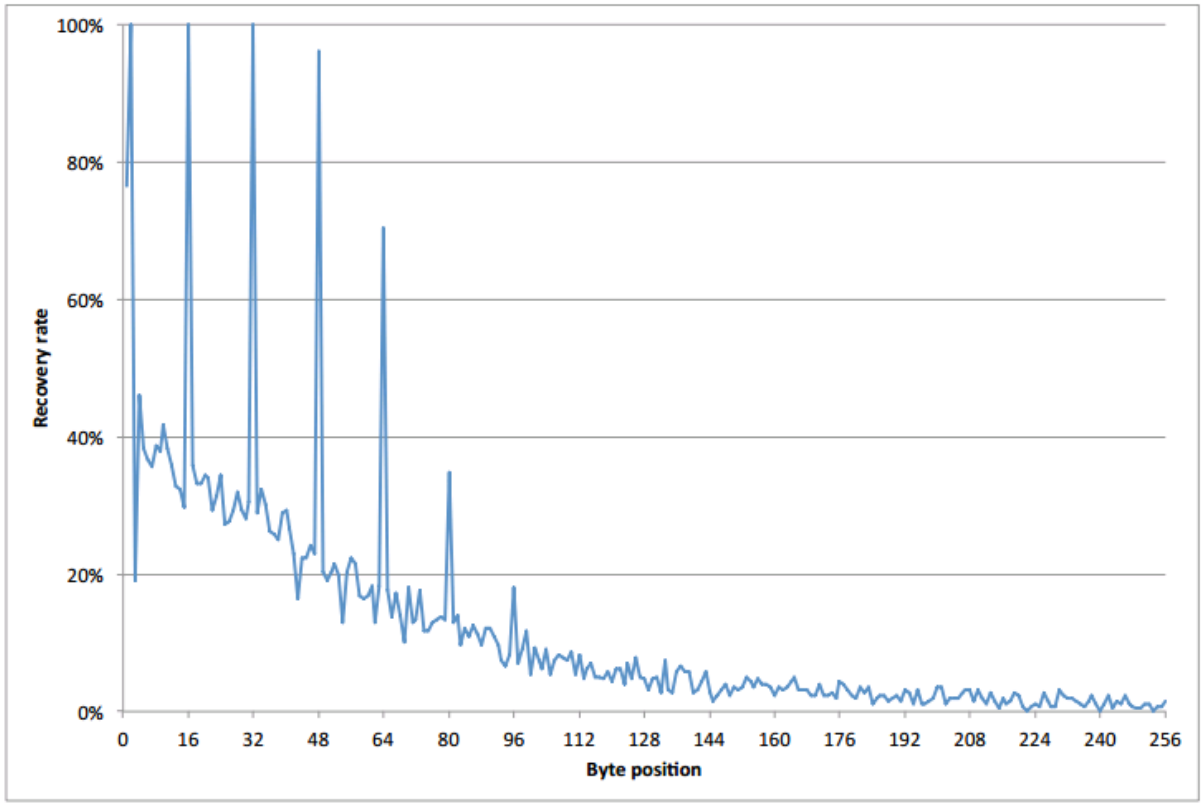


Figure: Recovery rate of the single-byte bias attack (based on 256 experiments)

More security definitions are needed...

- Stream ciphers are semantically secure.
- But what about multiple encryptions?
- What if the adversary wants to be challenged on two vectors of messages instead of two single messages?
- Obviously, he can trivially win the game (why?)
- Conclusion: deterministic encryption schemes are NOT secure under the multiple encryptions model.

More security definitions are needed...

- Stream ciphers are semantically secure.
- But what about multiple encryptions?
- What if the adversary wants to be challenged on two vectors of messages instead of two single messages?
- Obviously, he can trivially win the game (why?)
- Conclusion: deterministic encryption schemes are NOT secure under the multiple encryptions model.

CPA Indistinguishability Experiment $\text{PrivK}^{\text{cpa}}_{\mathcal{A},E}$

Challenger Ch	**Adversary** A
$k = \text{KeyGen}(n)$	Access to an oracle $\text{Enc}(k, \cdot)$
$\xleftarrow{m_0, m_1, \lvert m_0\rvert = \lvert m_1\rvert}$	
$b \leftarrow \{0, 1\}$	
$\xrightarrow{c = \text{Enc}(k, m_b)}$	Outputs his guess b'

Definition

An encryption scheme is CPA-secure if for all efficient $\mathcal{A}$ the following holds:

$$\text{Adv}^{\text{cpa}}_{\mathcal{A},E}(n) = \Pr[\text{PrivK}^{\text{cpa}}_{\mathcal{A},E}(n) = 1] - 1/2 = \text{negl}(n)$$

Where $\text{PrivK}^{\text{cpa}}_{\mathcal{A},E}(n) = 1$ if $b' = b$, and 0 otherwise.

Note: CPA-secure $\Rightarrow$ CPA secure for multiple encryptions.

CCA Indistinguishability Experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},E}$

Challenger Ch

$k = \mathsf{KeyGen}(n)$

$\xleftarrow{m_0, m_1, |m_0| = |m_1|}$

$b \leftarrow \{0, 1\}$

$\xrightarrow{c = \mathsf{Enc}(k, m_b)}$

Adversary A

Access to two oracles $\mathsf{Enc}(k, \cdot), \mathsf{Dec}(k, \cdot)$

Access to two oracles $\mathsf{Enc}(k, \cdot), \mathsf{Dec}(k, \cdot)^c$

Outputs his guess b'

Definition

An encryption scheme is CCA-secure if for all efficient $\mathcal{A}$ the following holds:

$$\mathsf{Adv}^{\mathsf{cca}}_{\mathcal{A},E}(n) = \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},E}(n) = 1] - 1/2 = \mathsf{negl}(n)$$

Outline

Pseudo-Random Functions

- A generalization of the notion of pseudo-random generators.
- We now consider a “random-looking” function.
- It is the pseudo-randomness of a distribution on functions.
- We are interested in *keyed functions*, i.e. $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$.
- Once we choose k, we fix it, and then use $F_k : \{0,1\}^* \to \{0,1\}^*$.
- F is *length-preserving* if the lengths of the key, input, output are equal.
- F is pseudo-random if the function F_k, for a uniform key k, is indistinguishable from a function chosen uniformally at random from the set of all functions with the same domain and range.

Pseudo-Random Functions

Definition

Let $Func[X, Y]$ be the set of all functions from X to Y. A function $F : K \times X \to Y$ is a secure Pseudo-Random Function (PRF) if F is efficiently computable and for all PPT distinguishers $\mathcal{A}$

$$\mathsf{Adv}^{\mathrm{prf}}_{F,\mathcal{A}}(n) = |\Pr[\mathcal{A}^{F()}(n) = 1] - \Pr[\mathcal{A}^{F_k()}(n) = 1]| < \mathsf{negl}(n)$$

where $F \in Func[X, Y]$, $k \in K$, and $\mathcal{A}$ has access to the function in question, i.e. either $F()$ and $F_k()$.

Note that $|Func[X, Y]| = |X|^{|Y|}$.

Pseudo-Random Permutations

A pseudo-random function $F : K \times X \to Y$ is an efficient pseudo-random permutation if the following hold:

- F is injective and $|X| = |Y|$.
- F is deterministic and efficiently computable.
- F^{-1} is efficiently computable.

In practice: $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, where;

- 3DES: $n = 64$ bits, $k = 168$ bits
- AES: $n = 128$ bits, $k = 128, 192, 256$ bits

Pseudo-Random Permutations

A pseudo-random function $F : K \times X \to Y$ is an efficient pseudo-random permutation if the following hold:

- F is injective and $|X| = |Y|$.
- F is deterministic and efficiently computable.
- F^{-1} is efficiently computable.

In practice: $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, where;

- 3DES: $n = 64$ bits, $k = 168$ bits
- AES: $n = 128$ bits, $k = 128, 192, 256$ bits

Encryption using PRP

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudo-random permutation. We define the following encryption scheme $E = (\text{KeyGen}, \text{Enc}, \text{Dec})$:

- KeyGen : it takes n and outputs a key $k \in \{0,1\}^n$.
- Enc: it takes a key $k \in \{0,1\}^n$ and message m, it picks a random $r \leftarrow \{0,1\}^n$, and outputs

$$(c_0, c_1) \leftarrow (r, F_k(r) \oplus m).$$

- Dec: it takes a key k and a ciphertext $c = (c_0, c_1)$ and outputs

$$m \leftarrow (F_k(c_0) \oplus c_1).$$

Theorem

If F is a pseudo-random permutation then the encryption scheme E is CPA-secure.

Proof.

We will first prove that a version E' of this encryption scheme would be indeed CPA-secure if the function F was truly random, and then show that if E was insecure then we can distinguish F from a truly random function.
For E', there are two cases:

- r_c didn't appear in any of the encryption queries, and therefore the probability to win the game in this case is exactly 1/2.
- r_c appeared in *at least* one of the queries. Assuming that the adversary is restricted to $q(n)$ queries, then the probability of this event is at most $q(n)/2^n$.

Thus $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},E'}(n) = 1] \leq 1/2 + q(n)/2^n$.

□

Proof.

F is a PRP $\Rightarrow$

$$\left| \Pr[\mathcal{A}^{F()}(n) = 1] - \Pr[\mathcal{A}^{F_k()}(n) = 1] \right| < \mathsf{negl}(n)$$

One can easily see that $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},E'}(n) = 1] = \Pr[\mathcal{A}^{F()}(n) = 1]$ and $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},E}(n) = 1] = \Pr[\mathcal{A}^{F_k()}(n) = 1]$. Thus

$$\left| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},E}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},E'}(n) = 1] \right| \leq \mathsf{negl}(n)$$

Therefore

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},E}(n) = 1] \leq 1/2 + q(n)/2^n + \mathsf{negl}(n)\,.$$

□

Strong Pseudo-Random Permutations

Definition

Let Perm_n *be the set of all permutations from* $\{0,1\}^n$ *to* $\{0,1\}^n$*. Let* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *be an efficient length-preserving, keyed permutation.* f *is a strong pseudo-random permutation if for all probabilistic polynomial-time distinguishers D, there exists a negligible function negl such that*

$$|\Pr[D^{f(),f^{-1}()}(n) = 1] - \Pr[D^{f_k(),f_k^{-1}()}(n) = 1]| < \text{negl}(n)$$

where $f \in_R \text{Perm}_n$*,* $k \in_R \{0,1\}^k$*, and* D *has access to the functions in question, i.e. either* $f(), f^{-1}()$ *and* $f_k(), f_k^{-1}()$*.*

Notes:

- Strong PRP $\Rightarrow$ PRP
- $|\text{Perm}_n| = 2^n!$

Further Reading (1)


Nadhem J AlFardan, Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob CN Schuldt.
On the security of RC4 in TLS.
In *USENIX Security*, pages 305–320, 2013.


Boaz Barak and Shai Halevi.
A model and architecture for pseudo-random generation with applications to/dev/random.
In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 203–212. ACM, 2005.


Daniel J Bernstein.
The Salsa20 Family of Stream Ciphers.
In *New stream cipher designs*, pages 84–97. Springer, 2008.

Further Reading (2)


Lenore Blum, Manuel Blum, and Mike Shub.
A simple unpredictable pseudo-random number generator.
SIAM Journal on computing, 15(2):364–383, 1986.


Christian Cachin.
Entropy measures and unconditional security in cryptography.
PhD thesis, SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH, 1997.


Scott Fluhrer, Itsik Mantin, and Adi Shamir.
Weaknesses in the key scheduling algorithm of RC4.
In *Selected areas in cryptography*, pages 1–24. Springer, 2001.

Further Reading (3)

Christina Garman, Kenneth G Paterson, and Thyla van der Merwe.
Attacks only get better: Password recovery attacks against RC4 in TLS.
2015.

Itsik Mantin and Adi Shamir.
A practical attack on broadcast RC4.
In *Fast Software Encryption*, pages 152–164. Springer, 2002.