The Security of Ciphertext Stealing

Phillip Rogaway[1] Mark Wooding[2] Haibin Zhang[1]

[1]Department of Computer Science
University of California at Davis

[2]Thales e-Security Ltd

March 19, 2012

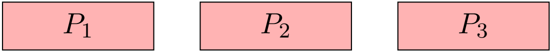
Outline

1. Ciphertext stealing
   - Description
   - Symmetric encryption schemes
   - Security of ciphertext stealing
   - Insecurity of the Meyer–Matyas scheme
2. Online encryption
   - Definitions
   - Delayed CBC
   - Ciphertext stealing redux

Outline

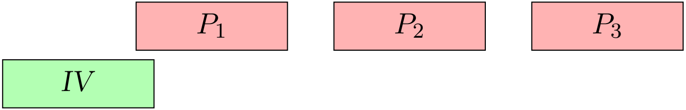
1. Ciphertext stealing
   - Description
   - Symmetric encryption schemes
   - Security of ciphertext stealing
   - Insecurity of the Meyer–Matyas scheme
2. Online encryption
   - Definitions
   - Delayed CBC
   - Ciphertext stealing redux

Ciphertext stealing



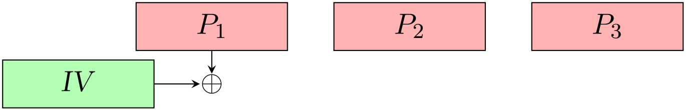
Suppose we have a message to encrypt. We might choose ciphertext block chaining.

Ciphertext stealing



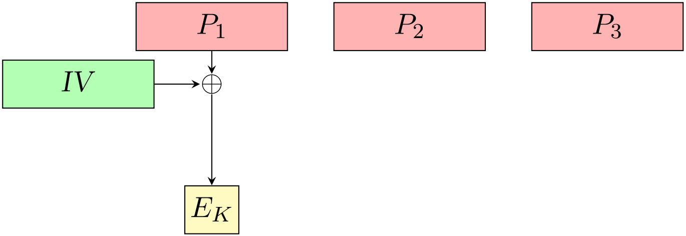
We choose a random initialization vector.

Ciphertext stealing



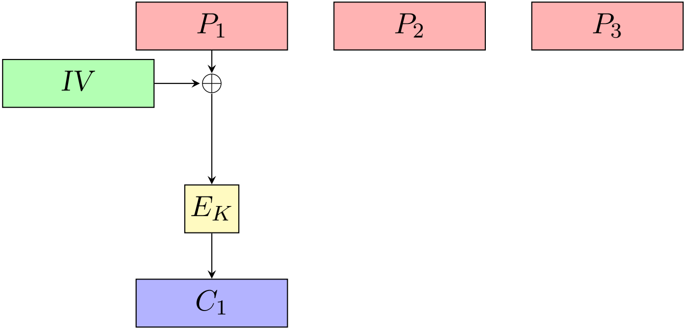
We whiten the first plaintext block by XORing with the IV.

Ciphertext stealing



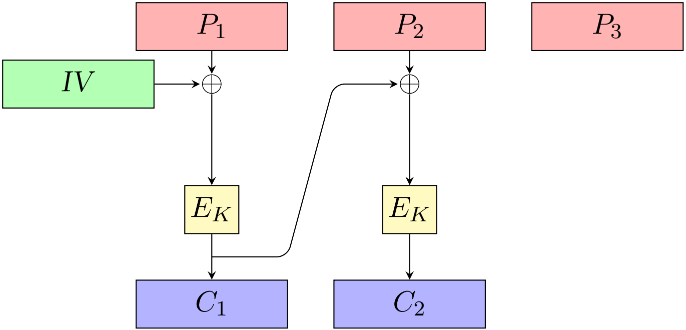
We feed the whitened plaintext through the blockcipher.

Ciphertext stealing



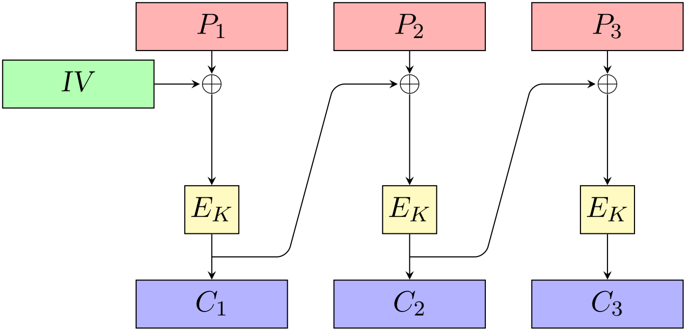
This gives us a ciphertext block.

Ciphertext stealing



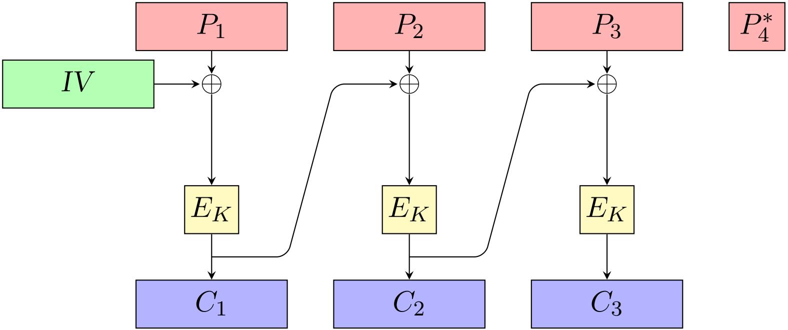
We whiten the next plaintext block using that ciphertext block, apply the blockcipher, and get a new ciphertext block.

Ciphertext stealing



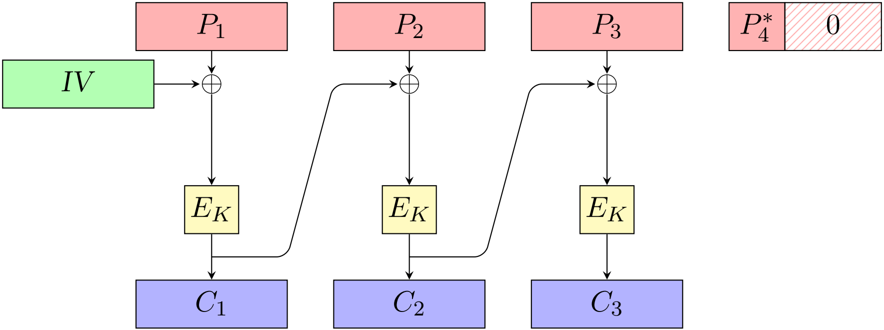
We repeat this for all of the plaintext blocks.

Ciphertext stealing



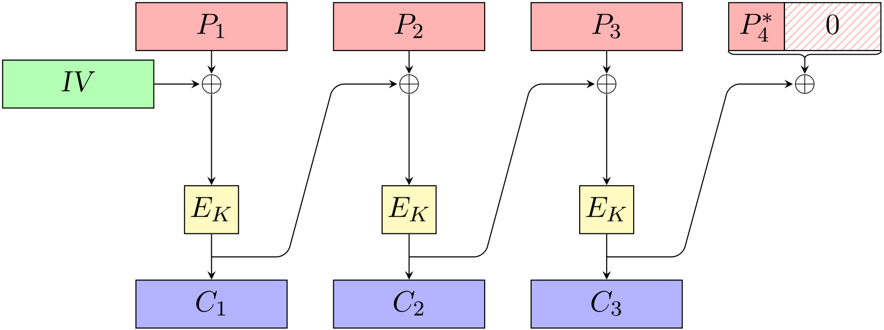
But wait: what if the plaintext isn't a whole number of blocks? There'll be an odd bit left over.

Ciphertext stealing



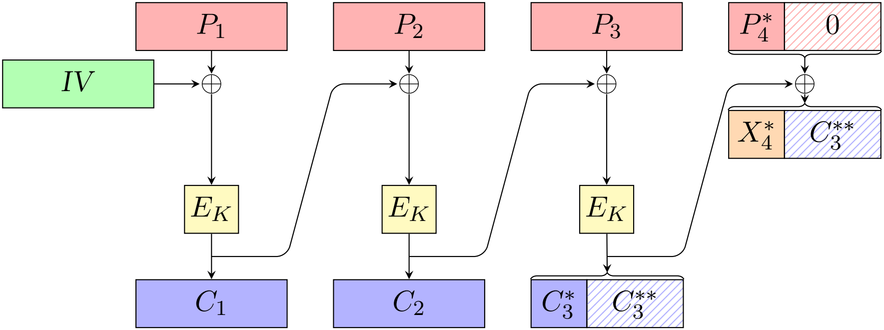
We pad the partial block with zero bits.

Ciphertext stealing



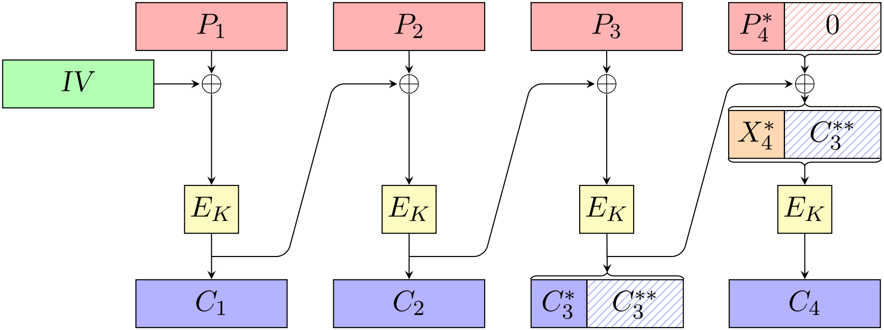
And then whiten using the previous ciphertext block, as usual.

Ciphertext stealing



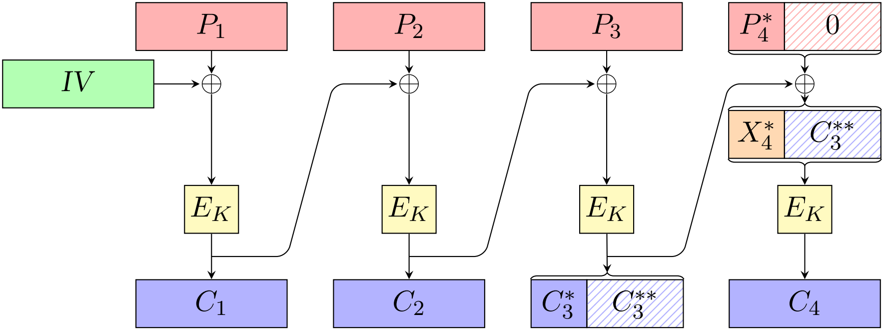
This leaves us with the whitened odd bit of plaintext, and a *copy* of the rest of the previous ciphertext block.

Ciphertext stealing



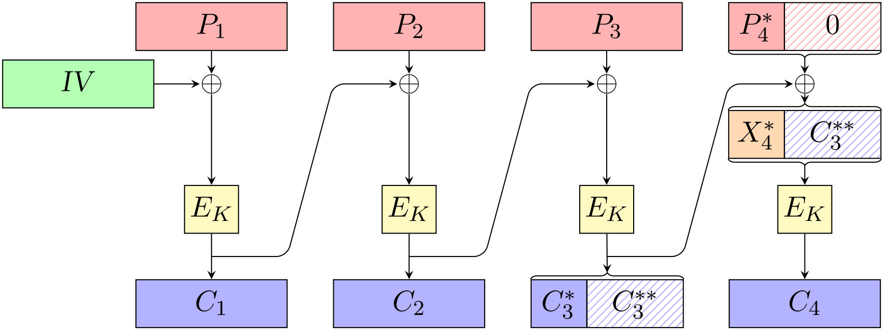
It's the right width, so we can feed it through the blockcipher and get a ciphertext block.

Ciphertext stealing



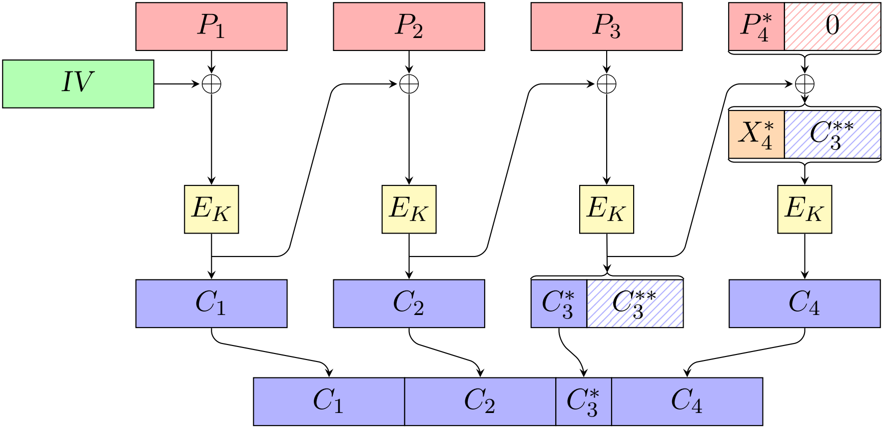
If we decrypt that last ciphertext, we get the end of the penultimate ciphertext block back. So we don't need to transmit that part!

Ciphertext stealing



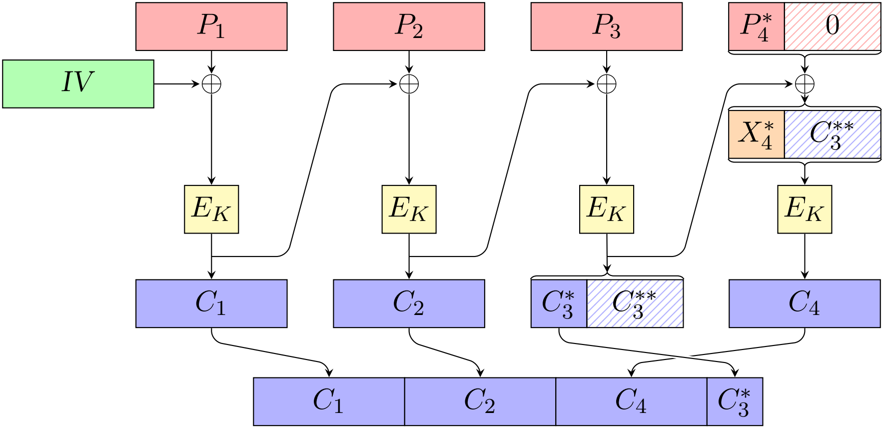
Addendum to NIST SP800–38A describes three variants differing in ciphertext ordering.

Ciphertext stealing



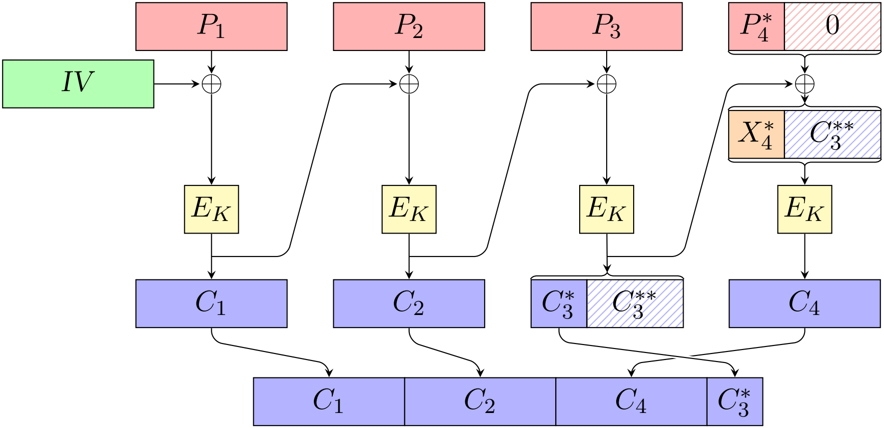
CBC-CS1 preserves *ordering*;

Ciphertext stealing



CBC-CS1 preserves *ordering*; CBC-CS3 preserves *alignment* by swapping;

Ciphertext stealing



CBC-CS1 preserves *ordering*; CBC-CS3 preserves *alignment* by swapping; CBC-CS2 swaps only when necessary, for compatibility.

Ciphertext stealing: history

- Basic idea goes back at least to Meyer and Matyas (1982).

Ciphertext stealing: history

- Basic idea goes back at least to Meyer and Matyas (1982). (Unfortunately their version is broken.)

Ciphertext stealing: history

- Basic idea goes back at least to Meyer and Matyas (1982). (Unfortunately their version is broken.)
- CBC-CS3 described by Schneier (1996).

Ciphertext stealing: history

- Basic idea goes back at least to Meyer and Matyas (1982). (Unfortunately their version is broken.)
- CBC-CS3 described by Schneier (1996).
- CBC-CS3 specified in RFC2040 (1996, Baldwin and Rivest).

Ciphertext stealing: history

- Basic idea goes back at least to Meyer and Matyas (1982). (Unfortunately their version is broken.)
- CBC-CS3 described by Schneier (1996).
- CBC-CS3 specified in RFC2040 (1996, Baldwin and Rivest).
- All three standardized in addendum to NIST SP800–38A (2010).

Definitions: symmetric encryption

Symmetric encryption syntax

We take a functional view of symmetric encryption schemes.

$$\mathcal{E} : \mathcal{K} \times \mathcal{IV} \times \mathcal{P} \to \mathcal{P}$$

Definitions: symmetric encryption

Symmetric encryption syntax

We take a functional view of symmetric encryption schemes.

$$\mathcal{E} : \mathcal{K} \times \mathcal{IV} \times \mathcal{P} \to \mathcal{P}$$

- $\mathcal{K} \subseteq \{0,1\}^*$ is a finite *key space*;
- $\mathcal{IV} = \{0,1\}^v$ is an *IV space*;
- $\mathcal{P} \subseteq \{0,1\}^*$ is the *message space*.
- Require $\mathcal{E}_K^{IV}(\cdot)$ to be a length-preserving permutation on $\mathcal{P}$.

Definitions: symmetric encryption

Symmetric encryption security: ind$

We capture an adversary and play one of two games…

Definitions: symmetric encryption

Symmetric encryption security: ind$

$$IV \stackrel{\$}{\leftarrow} \mathcal{IV}$$
$$c \leftarrow \mathcal{E}_K^{IV}(m)$$

$\xleftarrow{m \in \mathcal{P}}$

$\xrightarrow[IV \parallel c]{}$

The real game: adversary chooses plaintexts m: we give back ciphertexts with fresh random IVs and a consistent random key.

Definitions: symmetric encryption

Symmetric encryption security: ind$

$$m \in \mathcal{P} \longrightarrow \qquad \longleftarrow IV \parallel c$$

$$IV \xleftarrow{\$} \mathcal{IV}$$
$$c \xleftarrow{\$} \{0,1\}^{|m|}$$

The fake game: adversary chooses plaintexts m: we give back fresh random IVs and random fake ciphertexts.

Definitions: symmetric encryption

Symmetric encryption security: ind$

$IV \xleftarrow{\$} \mathcal{IV}$
$c \leftarrow \mathcal{E}_K^{IV}(m)$

$\xleftarrow{m \in \mathcal{P}}$
$\xrightarrow{IV \parallel c}$

$\xrightarrow{m \in \mathcal{P}}$
$\xleftarrow{IV \parallel c}$

$IV \xleftarrow{\$} \mathcal{IV}$
$c \xleftarrow{\$} \{0,1\}^{|m|}$

$$\mathbf{Adv}_{\mathcal{E}}^{\text{ind\$}}(A) = \Pr[A^{\text{Real}(\cdot)} \Rightarrow 1] - \Pr[A^{\text{Fake}(\cdot)} \Rightarrow 1]$$

The adversary's *advantage* measures how well he can distinguish between these games.

Security of ciphertext stealing

Theorem

Let $\mathcal{E}$ *be any of* $\text{CBC-CS1}[\text{Perm}(b)]$, $\text{CBC-CS2}[\text{Perm}(b)]$, *or* $\text{CBC-CS3}[\text{Perm}(b)]$ *and suppose adversary* A *asks queries totalling at most* σ *blocks. Then*

$$\mathbf{Adv}_{\mathcal{E}}^{\text{ind\$}}(A) \leq \sigma^2/2^b$$

Security of ciphertext stealing

Theorem

Let $\mathcal{E}$ be any of CBC-CS1[Perm(b)], CBC-CS2[Perm(b)], *or* CBC-CS3[Perm(b)] *and suppose adversary A asks queries totalling at most σ blocks. Then*

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{ind\$}}(A) \leq \sigma^2/2^b$$

Proof idea

- Factor

$$\text{CBC-CS}n_K^{IV}(m) = \text{POST}_n\big(|m|, \text{CBC}_K^{IV}(\text{PRE}(m))\big)$$

□

Security of ciphertext stealing

Theorem

Let $\mathcal{E}$ be any of $\textsc{cbc-cs}1[\mathrm{Perm}(b)]$, $\textsc{cbc-cs}2[\mathrm{Perm}(b)]$, *or* $\textsc{cbc-cs}3[\mathrm{Perm}(b)]$ *and suppose adversary A asks queries totalling at most σ blocks. Then*

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{ind\$}}(A) \le \sigma^2/2^b$$

Proof idea

- Factor

$$\textsc{cbc-cs}n_K^{IV}(m) = \textsc{post}_n\big(|m|, \textsc{cbc}_K^{IV}(\textsc{pre}(m))\big)$$

- Observe that $\textsc{post}_n$ preserves uniform distribution.

□

Security of ciphertext stealing

Theorem

Let $\mathcal{E}$ *be any of* $\textsc{cbc-cs}1[\mathrm{Perm}(b)]$, $\textsc{cbc-cs}2[\mathrm{Perm}(b)]$, *or* $\textsc{cbc-cs}3[\mathrm{Perm}(b)]$ *and suppose adversary* A *asks queries totalling at most* σ *blocks. Then*

$$\mathbf{Adv}^{\mathrm{ind\$}}_{\mathcal{E}}(A) \le \sigma^2/2^b$$

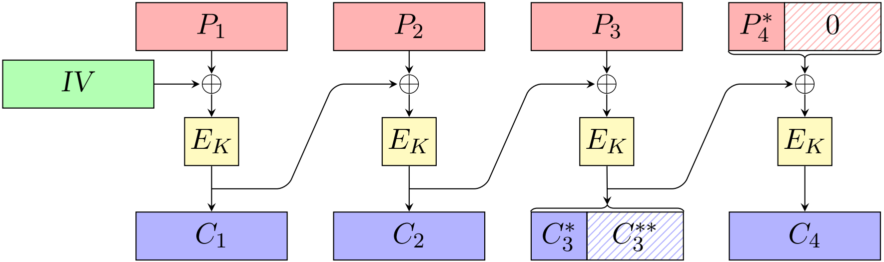
Proof idea

- Factor

$$\textsc{cbc-cs}n^{IV}_{K}(m) = \textsc{post}_n\big(|m|, \textsc{cbc}^{IV}_{K}(\textsc{pre}(m))\big)$$

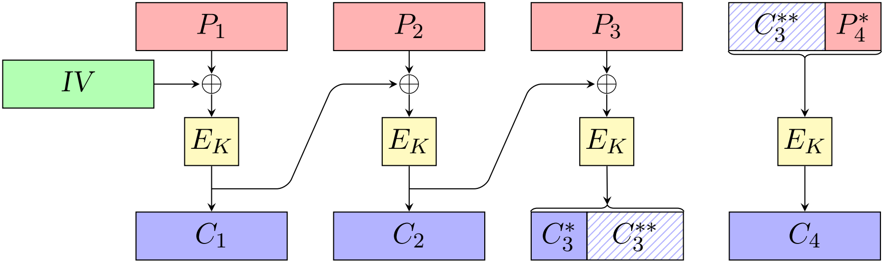
- Observe that $\textsc{post}_n$ preserves uniform distribution.
- Show reduction from CBC security. □

Insecurity of the Meyer–Matyas scheme



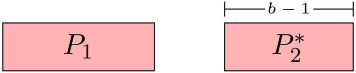
The NIST CBC ciphertext stealing schemes, for comparison.

Insecurity of the Meyer–Matyas scheme



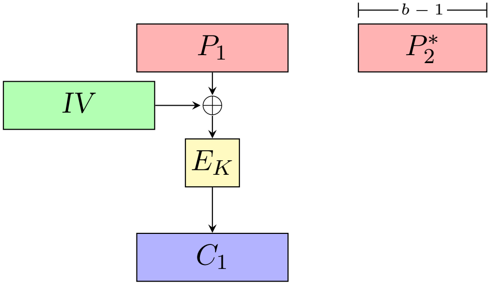
The Meyer–Matyas ciphertext stealing scheme. There's no chaining into the final partial block.

Insecurity of the Meyer–Matyas scheme



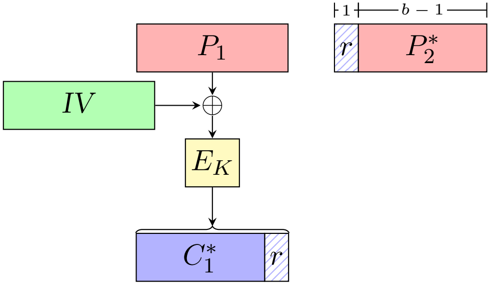
Start with a message m which is 1 bit short of two whole blocks.

Insecurity of the Meyer–Matyas scheme



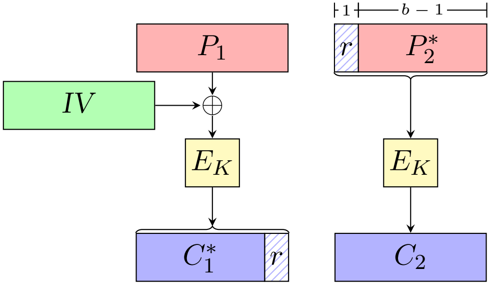
The first block is whitened with a fresh random IV and fed through the blockcipher.

Insecurity of the Meyer–Matyas scheme



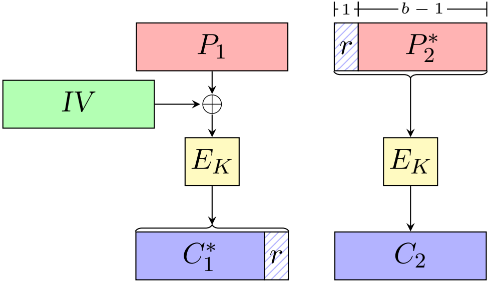
The second block is padded by prefixing with the final bit r of the previous ciphertext.

Insecurity of the Meyer–Matyas scheme



And then fed through the blockcipher.

Insecurity of the Meyer–Matyas scheme



But there are only two possible values for r. If we do this twice, we expect the C_2 values to be equal with probability at least $\frac{1}{2}$.

Insecurity of the Meyer–Matyas scheme

$$m = 1^b \,\|\, 0^{b-1}$$

Our adversary starts with such a message.

Insecurity of the Meyer–Matyas scheme

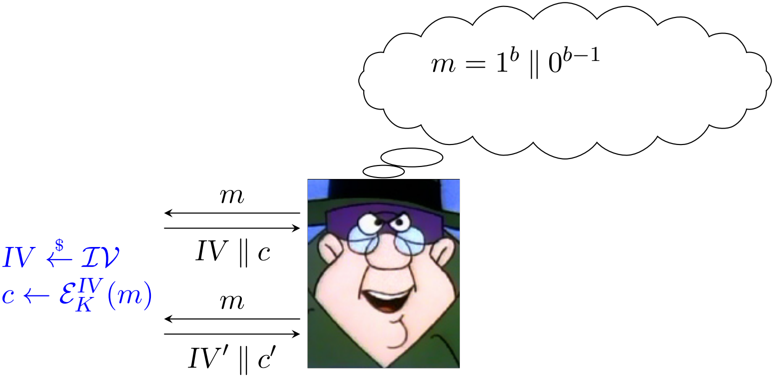
$$m = 1^b \parallel 0^{b-1}$$

$$IV \xleftarrow{\$} \mathcal{IV}$$
$$c \leftarrow \mathcal{E}_K^{IV}(m)$$

$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad IV \parallel c \quad}$

And asks its encryption oracle to encrypt it, getting a ciphertext c.



Then it asks to encrypt the same message again, getting a new ciphertext c'.

Insecurity of the Meyer–Matyas scheme

$m = 1^b \parallel 0^{b-1}$

$\mathrm{LSB}_b(c) = \mathrm{LSB}_b(c')$ **?**

$IV \xleftarrow{\$} \mathcal{IV}$

$c \leftarrow \mathcal{E}_K^{IV}(m)$

m ←

→ $IV \parallel c$

m ←

→ $IV' \parallel c'$

$$\mathbf{Adv}^{\mathrm{ind\$}}_{\mathrm{CBC\text{-}CSX}}(A) = \Pr[A^{\mathrm{Real}(\cdot)} \Rightarrow 1] - \Pr[A^{\mathrm{Fake}(\cdot)} \Rightarrow 1]$$

The adversary declares 'real' if the last b bits of c and c' are equal.

$m = 1^b \,\|\, 0^{b-1}$

$\mathrm{LSB}_b(c) = \mathrm{LSB}_b(c')$ **?**

$IV \xleftarrow{\$} \mathcal{IV}$

$c \leftarrow \mathcal{E}_K^{IV}(m)$

$\xleftarrow{\;m\;}$

$\xrightarrow[IV \,\|\, c]{}$

$\xleftarrow{\;m\;}$

$\xrightarrow[IV' \,\|\, c']{}$

$$\mathbf{Adv}^{\mathrm{ind\$}}_{\mathrm{CBC\text{-}CSX}}(A) \geq \frac{1}{2} - \Pr[A^{\mathrm{Fake}(\cdot)} \Rightarrow 1]$$

If this is indeed the real game, we've just seen that they're equal with probability at least $\frac{1}{2}$.

Insecurity of the Meyer–Matyas scheme

$m = 1^b \parallel 0^{b-1}$
$\mathrm{LSB}_b(c) = \mathrm{LSB}_b(c')$ **?**

Real oracle	Messages to the real oracle	Adversary	Messages to the fake oracle	Fake oracle
$IV \xleftarrow{\$} \mathcal{IV}$ $c \leftarrow \mathcal{E}_K^{IV}(m)$	m ← → $IV \parallel c$ m ← → $IV' \parallel c'$		→ m $IV \parallel c$ ← → m $IV' \parallel c'$ ←	$IV \xleftarrow{\$} \mathcal{IV}$ $c \xleftarrow{\$} \{0,1\}^{\lvert m \rvert}$

$$\mathbf{Adv}^{\text{ind\$}}_{\text{CBC-CSX}}(A) \geq \frac{1}{2} - \Pr[A^{\text{Fake}(\cdot)} \Rightarrow 1]$$

If this is the fake game, then the ciphertexts are simply random strings.

Insecurity of the Meyer–Matyas scheme

$m = 1^b \,\|\, 0^{b-1}$

$\mathrm{LSB}_b(c) = \mathrm{LSB}_b(c')$ **?**

Real oracle		Adversary		Random oracle
$IV \xleftarrow{\$} \mathcal{IV}$ $c \leftarrow \mathcal{E}_K^{IV}(m)$	$\leftarrow m$ $IV \,\|\, c \rightarrow$ $\leftarrow m$ $IV' \,\|\, c' \rightarrow$		$m \rightarrow$ $\leftarrow IV \,\|\, c$ $m \rightarrow$ $\leftarrow IV' \,\|\, c'$	$IV \xleftarrow{\$} \mathcal{IV}$ $c \xleftarrow{\$} \{0,1\}^{\|m\|}$

$$\mathbf{Adv}^{\mathrm{ind\$}}_{\mathrm{CBC\text{-}CSX}}(A) \geq \frac{1}{2} - \frac{1}{2^b}$$

So they're equal with probability exactly $1/2^b$.

Outline

1. Ciphertext stealing
   - Description
   - Symmetric encryption schemes
   - Security of ciphertext stealing
   - Insecurity of the Meyer–Matyas scheme

2. Online encryption
   - Definitions
   - Delayed CBC
   - Ciphertext stealing redux

Background

Idea

- Conventional definitions treat encryption as processing an entire message in one go.

Background

Idea

- Conventional definitions treat encryption as processing an entire message in one go.
- In real life, messages are often processed in chunks.

Background

Idea

- Conventional definitions treat encryption as processing an entire message in one go.
- In real life, messages are often processed in chunks.
  - Keys held by memory-constrained devices.

Idea

- Conventional definitions treat encryption as processing an entire message in one go.
- In real life, messages are often processed in chunks.
  - Keys held by memory-constrained devices.
  - Reducing end-to-end latency.

Background

Idea

- Conventional definitions treat encryption as processing an entire message in one go.
- In real life, messages are often processed in chunks.
  - Keys held by memory-constrained devices.
  - Reducing end-to-end latency.
- We should have definitions which capture this behaviour so that we can analyse the security of schemes.

Background

Idea

- Conventional definitions treat encryption as processing an entire message in one go.
- In real life, messages are often processed in chunks.
  - Keys held by memory-constrained devices.
  - Reducing end-to-end latency.
- We should have definitions which capture this behaviour so that we can analyse the security of schemes.

History

Background

Idea

- Conventional definitions treat encryption as processing an entire message in one go.
- In real life, messages are often processed in chunks.
  - Keys held by memory-constrained devices.
  - Reducing end-to-end latency.
- We should have definitions which capture this behaviour so that we can analyse the security of schemes.

History

- Blockwise-adaptive attacks: [BKN02], [JMV02], [FMP03], [FJP04].

Background

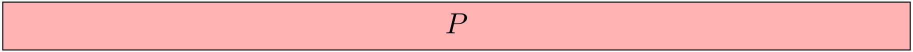
Idea

- Conventional definitions treat encryption as processing an entire message in one go.
- In real life, messages are often processed in chunks.
  - Keys held by memory-constrained devices.
  - Reducing end-to-end latency.
- We should have definitions which capture this behaviour so that we can analyse the security of schemes.

History

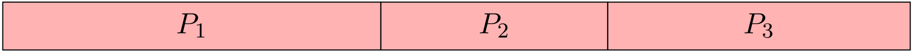
- Blockwise-adaptive attacks: [BKN02], [JMV02], [FMP03], [FJP04].
- Our stream-based approach from [GR97].

How it looks



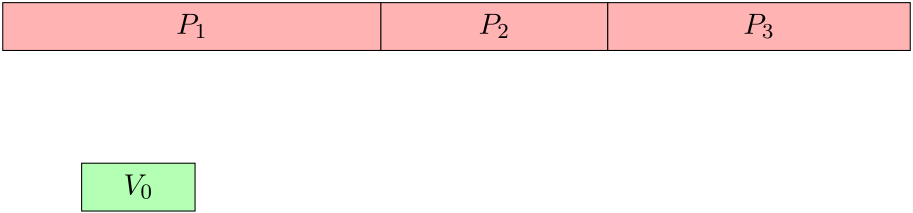
Suppose we have a plaintext message P. Maybe we don't even know all of it yet.

How it looks



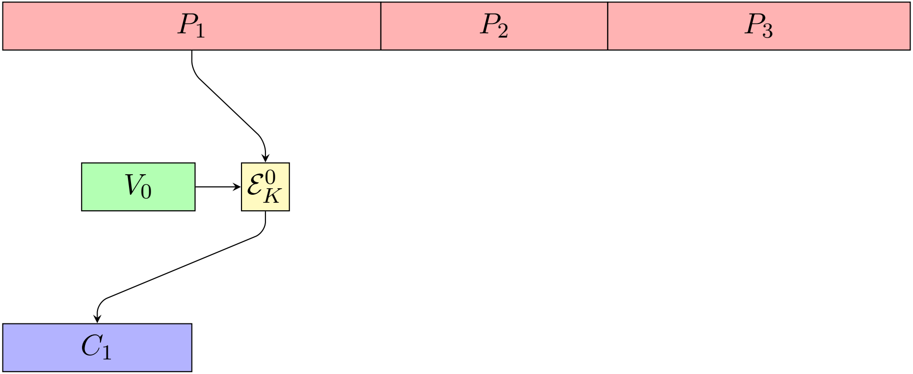
Split it into chunks P_1, P_2, … of arbitrary sizes.

How it looks



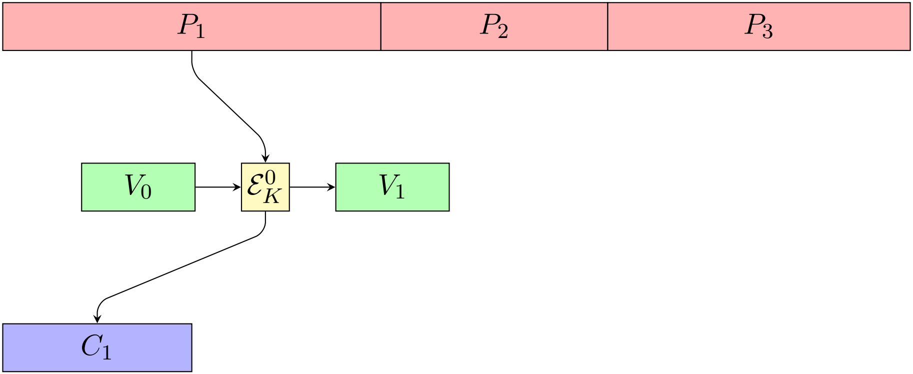
Sample an initial state ('initialization vector') V_0 appropriate for the encryption scheme.

How it looks



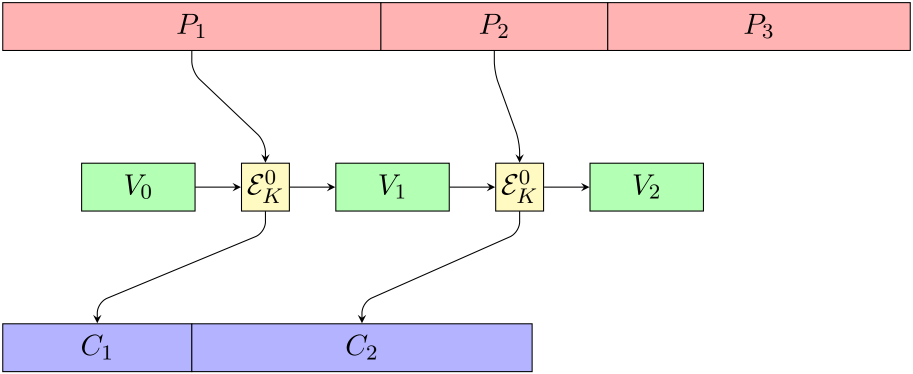
Feed the first plaintext chunk to the encryption scheme. It gives us a ciphertext chunk C_1. In general, C_1 might not be the same length as P_1.

How it looks



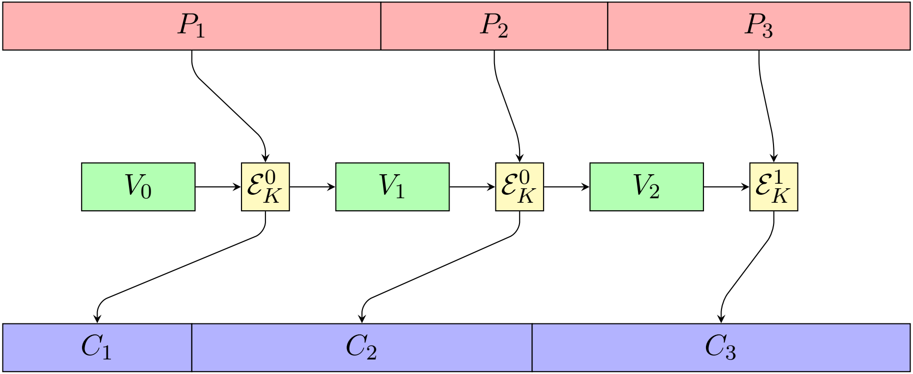
It also gives us a *state* V_1.

How it looks



We can feed the next plaintext P_2 to the encryption scheme, along with the previous state V_1. We get a ciphertext chunk C_2 and a new state V_2.

How it looks



And so on. . . Indicate to the encryption scheme when there are no more chunks to process.

What's new about our definition

- We don't depend on chunks being single blocks, or aligned to block boundaries.

What's new about our definition

- We don't depend on chunks being single blocks, or aligned to block boundaries.
- Indeed, we don't assume there's a blockcipher involved at all.

What's new about our definition

- We don't depend on chunks being single blocks, or aligned to block boundaries.
- Indeed, we don't assume there's a blockcipher involved at all.
- Security is defined in terms of indistinguishability from random strings of appropriate lengths.

Definitions: online encryption

Online encryption syntax

We define online encryption schemes as functions:

$$\mathcal{E}\colon \mathcal{K} \times \mathcal{V} \times \{0,1\} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{V} \qquad (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},\delta}(P_i)$$

Definitions: online encryption

Online encryption syntax

We define online encryption schemes as functions:

$$\mathcal{E}\colon \mathcal{K} \times \mathcal{V} \times \{0,1\} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{V} \qquad (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},\delta}(P_i)$$

- $\mathcal{K}$ is the key space. We require that it be finite.

Definitions: online encryption

Online encryption syntax

We define online encryption schemes as functions:

$$\mathcal{E} : \mathcal{K} \times \mathcal{V} \times \{0,1\} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{V} \qquad (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},\delta}(P_i)$$

- $\mathcal{K}$ is the key space. We require that it be finite.
- $\mathcal{V} \subseteq \bigcup_{0 \le i < v} \{0,1\}^i$ is the *state* space.

Definitions: online encryption

Online encryption syntax

We define online encryption schemes as functions:

$$\mathcal{E}\colon \mathcal{K} \times \mathcal{V} \times \{0,1\} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{V} \qquad (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},\delta}(P_i)$$

- $\mathcal{K}$ is the key space. We require that it be finite.
- $\mathcal{V} \subseteq \bigcup_{0\le i<v} \{0,1\}^i$ is the *state* space.
- $\delta \in \{0,1\}$ is the *end-of-message indicator*: 0 means more chunks are coming; 1 means this is the last one.

Definitions: online encryption

Online encryption syntax

We define online encryption schemes as functions:

$$\mathcal{E} \colon \mathcal{K} \times \mathcal{V} \times \{0,1\} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{V} \qquad (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},\delta}(P_i)$$

- $\mathcal{K}$ is the key space. We require that it be finite.
- $\mathcal{V} \subseteq \bigcup_{0 \le i < v} \{0,1\}^i$ is the *state* space.
- $\delta \in \{0,1\}$ is the *end-of-message indicator*: 0 means more chunks are coming; 1 means this is the last one.
- Also a *message space* $\mathcal{P} \subseteq \{0,1\}^*$ and *IV space* $\mathcal{IV} \subseteq \mathcal{V}$.

Definitions: online encryption

Online encryption syntax

We define online encryption schemes as functions:

$$\mathcal{E}\colon \mathcal{K} \times \mathcal{V} \times \{0,1\} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{V} \qquad (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},\delta}(P_i)$$

Well-formedness requirements

Ciphertexts The ciphertext is always the same whichever way you split up the plaintext.

Definitions: online encryption

Online encryption syntax

We define online encryption schemes as functions:

$$\mathcal{E} \colon \mathcal{K} \times \mathcal{V} \times \{0,1\} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{V} \qquad (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},\delta}(P_i)$$

Well-formedness requirements

Ciphertexts The ciphertext is always the same whichever way you split up the plaintext.

Invertibility Ciphertexts can be decrypted uniquely.

Definitions: online encryption

Online encryption syntax

We define online encryption schemes as functions:

$$\mathcal{E}\colon \mathcal{K} \times \mathcal{V} \times \{0,1\} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{V} \qquad (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},\delta}(P_i)$$

Well-formedness requirements

- **Ciphertexts** The ciphertext is always the same whichever way you split up the plaintext.
- **Invertibility** Ciphertexts can be decrypted uniquely.
- **Lengths** The lengths of ciphertext chunks depend only on the history of plaintext lengths.

Well-formedness (formalities)

Notation

- $(C_1, C_2, \ldots, C_n) \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$

$$
\begin{array}{l}
V_0 \leftarrow IV; \\
\textbf{for } 1 \leq i \leq n-1 \textbf{ do } (C_i, V_i) \leftarrow \mathcal{E}_K^{V_{i-1},0}(P_i); \\
(C_n, V_n) \leftarrow \mathcal{E}_K^{V_{n-1},1}(P_n); \\
\textbf{return } (C_1, C_2, \ldots, C_n);
\end{array}
$$

Well-formedness (formalities)

Notation

- $(C_1, C_2, \ldots, C_n) \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$
- $C \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$

$$(C_n, V_n) \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n);$$
$$\textbf{return } C_1 \parallel C_2 \parallel \cdots \parallel C_n;$$

Well-formedness (formalities)

Notation

- $(C_1, C_2, \ldots, C_n) \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$
- $C \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$

Ciphertext consistency

If

$$P = P_1 \parallel P_2 \parallel \cdots \parallel P_n = P'_1 \parallel P'_2 \parallel \cdots \parallel P'_{n'}$$

then

$$\mathcal{E}_K^{IV}(P_1, P_2, \cdots, P_n) = \mathcal{E}_K^{IV}(P'_1, P'_2, \cdots, P'_{n'})$$

Well-formedness (formalities)

Notation

- $(C_1, C_2, \ldots, C_n) \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$
- $C \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$
- $C \leftarrow \mathcal{E}_K^{IV}(P)$

Ciphertext consistency

If

$$P = P_1 \parallel P_2 \parallel \cdots \parallel P_n = P'_1 \parallel P'_2 \parallel \cdots \parallel P'_{n'}$$

then

$$\mathcal{E}_K^{IV}(P_1, P_2, \cdots, P_n) = \mathcal{E}_K^{IV}(P'_1, P'_2, \cdots, P'_{n'})$$

Well-formedness (formalities)

Notation

- $(C_1, C_2, \ldots, C_n) \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$
- $C \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$
- $C \leftarrow \mathcal{E}_K^{IV}(P)$

Invertibility

$\mathcal{E}_K^{IV}(\cdot)$ must be injective on $\mathcal{P}$.

Well-formedness (formalities)

Notation

- $(C_1, C_2, \ldots, C_n) \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$
- $C \leftarrow \mathcal{E}_K^{IV}(P_1, P_2, \ldots, P_n)$
- $C \leftarrow \mathcal{E}_K^{IV}(P)$

Length consistency

Let $(C, \bar{V}) = \mathcal{E}_K^{V,\delta}$ and $(C', \bar{V}') = \mathcal{E}_{K'}^{V',\delta}(P')$. If $|P| = |P'|$ and $|V| = |V'|$ then $|C| = |C'|$ and $|\bar{V}| = |\bar{V}'|$.

Online encryption security: IND$

Initialization:
$V \xleftarrow{\$} \mathcal{IV}$

$(c, V) \leftarrow \mathcal{E}_K^{V,\delta}(m)$ $\xleftarrow{m,\,\delta}$ $\xrightarrow[c]{}$

Adversary submits message chunks and a 'done' flag to an oracle, which returns ciphertext chunks.

Online encryption security: IND$

Initialization:
$V \stackrel{\$}{\leftarrow} \mathcal{IV}$

$\xrightarrow{m, \delta}$ $(c, V) \leftarrow \mathcal{E}_K^{V,\delta}(m)$

$\xleftarrow{c'}$ $c' \stackrel{\$}{\leftarrow} \{0,1\}^{|c|}$

… or maybe it just returns random strings of the right length.

Online encryption security: IND$

Initialization:
$V \xleftarrow{\$} \mathcal{IV}$

$(c, V) \leftarrow \mathcal{E}_K^{V,\delta}(m)$ $\xleftarrow{m,\delta}$ $\xrightarrow[c]{}$

$\xrightarrow{m,\delta}$ $\xleftarrow[c']{}$ $(c, V) \leftarrow \mathcal{E}_K^{V,\delta}(m)$
$c' \xleftarrow{\$} \{0,1\}^{|c|}$

We'd like these to be hard to distinguish.

Online encryption security: IND$

Initialization:
$V_i \xleftarrow{\$} \mathcal{IV}$ for $i \in \mathbb{N}$

$(c, V_i) \leftarrow \mathcal{E}_K^{V_i,\delta}(m)$ $\underset{c}{\overset{i, m, \delta}{\leftrightarrows}}$

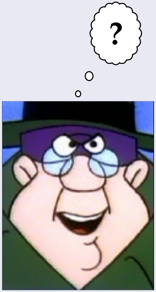
$\underset{c'}{\overset{i, m, \delta}{\rightleftarrows}}$ $(c, V_i) \leftarrow \mathcal{E}_K^{V_i,\delta}(m)$
$c' \xleftarrow{\$} \{0,1\}^{|c|}$

… even when the adversary can contribute to multiple messages concurrently.

Online encryption security: IND$

Initialization:
$V_i \xleftarrow{\$} \mathcal{IV}$ for $i \in \mathbb{N}$

$(c, V_i) \leftarrow \mathcal{E}_K^{V_i,\delta}(m)$ $\quad \xleftarrow{m,\,\delta} \quad \xrightarrow{c}$

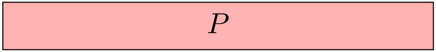


$\xrightarrow{m,\,\delta} \quad \xleftarrow{c'}$ $\quad (c, V_i) \leftarrow \mathcal{E}_K^{V_i,\delta}(m)$; $c' \xleftarrow{\$} \{0,1\}^{|c|}$

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{IND\$}}(A) = \Pr[A^{\mathrm{Real}(\cdot)} \Rightarrow 1] - \Pr[A^{\mathrm{Fake}(\cdot)} \Rightarrow 1]$$

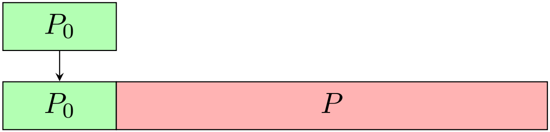
The adversary's advantage measures how well he can distinguish between these two games.

CBC online – wrong version



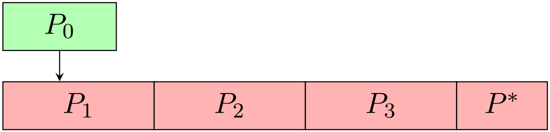
We're given a plaintext chunk.

CBC online – wrong version

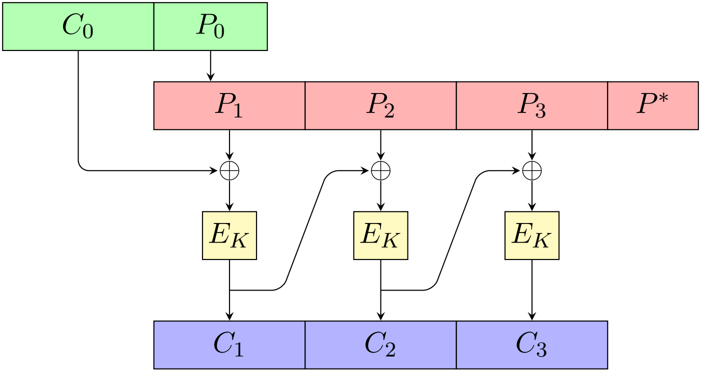


In general, we have a partial plaintext left over from the previous call. Tack this on the front.

CBC online – wrong version

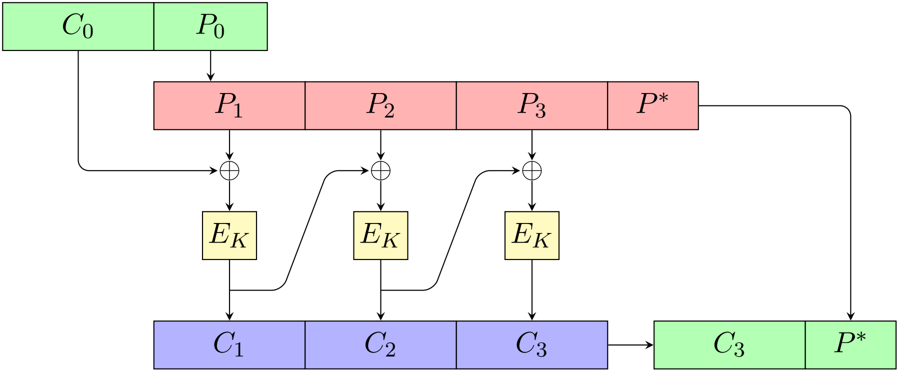


And split the plaintext into blocks. There'll be a bit left over.



Encrypt the whole blocks using CBC mode, using an IV maintained in the state.

CBC online – wrong version



The new state is the last ciphertext block, and the leftover bit of plaintext.

CBC online – insecurity of the wrong version

- Of course, this is insecure. The adversary learns the IV to be used to encrypt the next plaintext chunk as part of this ciphertext.

CBC online – insecurity of the wrong version

- Of course, this is insecure. The adversary learns the IV to be used to encrypt the next plaintext chunk as part of this ciphertext.
- Suppose this is V; suppose also that the adversary guesses that the plaintext corresponding to some ciphertext C_i is P^*, i.e., that $C_i = E_K(P^* \oplus C_{i-1})$.

CBC online – insecurity of the wrong version

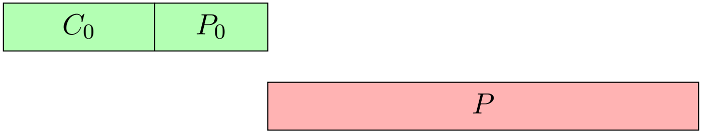
- Of course, this is insecure. The adversary learns the IV to be used to encrypt the next plaintext chunk as part of this ciphertext.
- Suppose this is V; suppose also that the adversary guesses that the plaintext corresponding to some ciphertext C_i is P^*, i.e., that $C_i = E_K(P^* \oplus C_{i-1})$.
- So he arranges for the first block encrypted as part of the next plaintext chunk to be $P^* \oplus V \oplus C_{i-1}$. If the resulting ciphertext is C_i then his guess is confirmed.

CBC online – insecurity of the wrong version

- Of course, this is insecure. The adversary learns the IV to be used to encrypt the next plaintext chunk as part of this ciphertext.
- Suppose this is V; suppose also that the adversary guesses that the plaintext corresponding to some ciphertext C_i is P^*, i.e., that $C_i = E_K(P^* \oplus C_{i-1})$.
- So he arranges for the first block encrypted as part of the next plaintext chunk to be $P^* \oplus V \oplus C_{i-1}$. If the resulting ciphertext is C_i then his guess is confirmed.
- It's sufficient to hold one block back [FMP03].

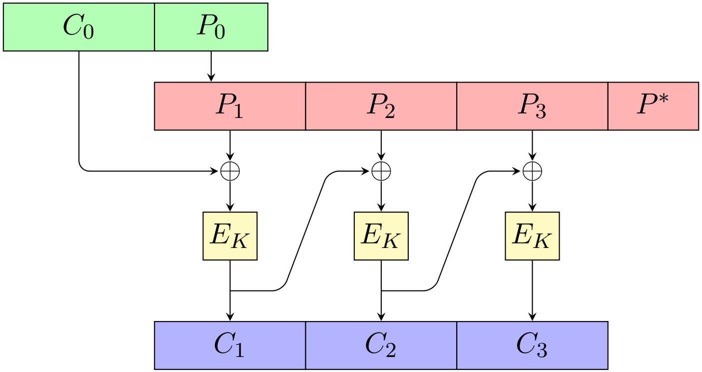
- Of course, this is insecure. The adversary learns the IV to be used to encrypt the next plaintext chunk as part of this ciphertext.
- Suppose this is V; suppose also that the adversary guesses that the plaintext corresponding to some ciphertext C_i is P^*, i.e., that $C_i = E_K(P^* \oplus C_{i-1})$.
- So he arranges for the first block encrypted as part of the next plaintext chunk to be $P^* \oplus V \oplus C_{i-1}$. If the resulting ciphertext is C_i then his guess is confirmed.
- It's sufficient to hold one block back [FMP03]. Intuition: CBC output is indistinguishable from random data, so the last block should be unpredictable, which is sufficient for security.

Delayed CBC [FMP03]



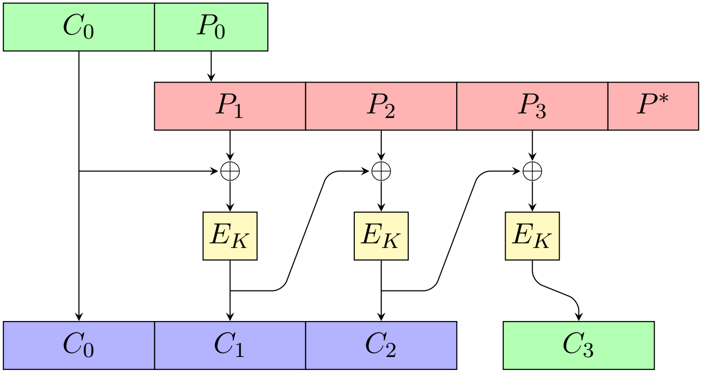
The state looks the same: previous ciphertext, and leftover plaintext.

Delayed CBC [FMP03]



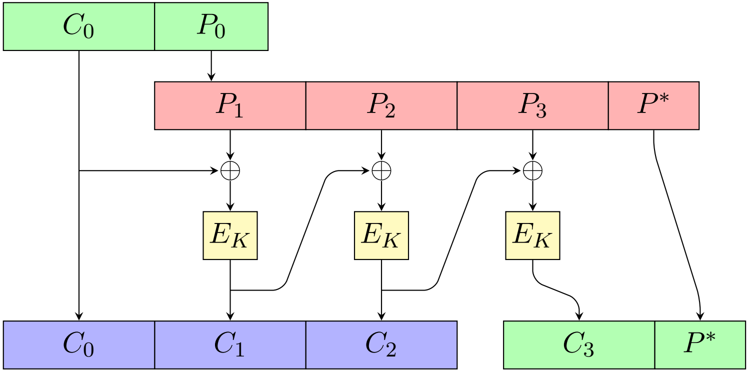
Prefix the leftover plaintext to the new chunk, split into blocks, and encrypt.

Delayed CBC [FMP03]



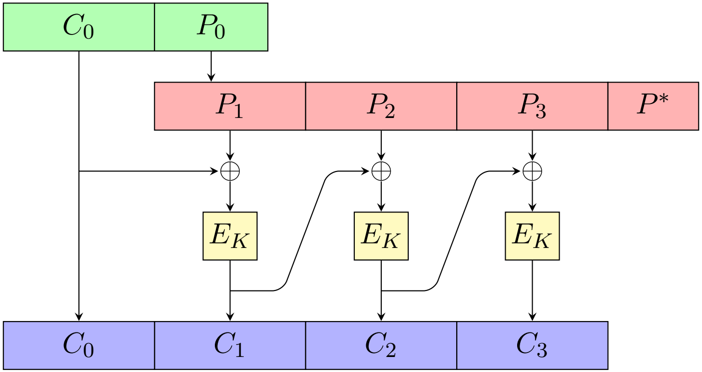
We must output the previous-ciphertext block. We shouldn't output the last new ciphertext block, just store it for later.

Delayed CBC [FMP03]



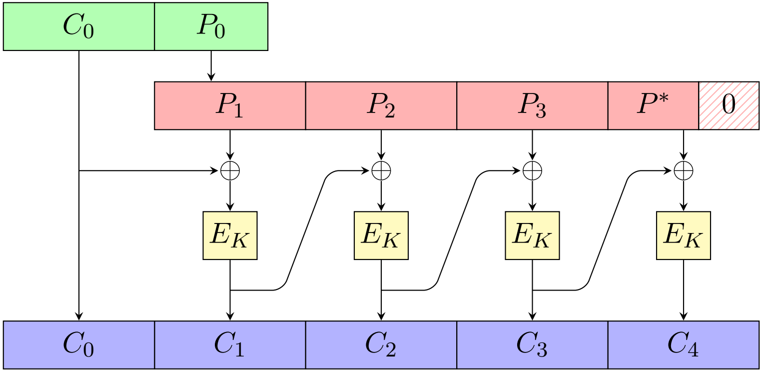
And we keep the leftover piece of plaintext.

Delayed CBC with ciphertext stealing



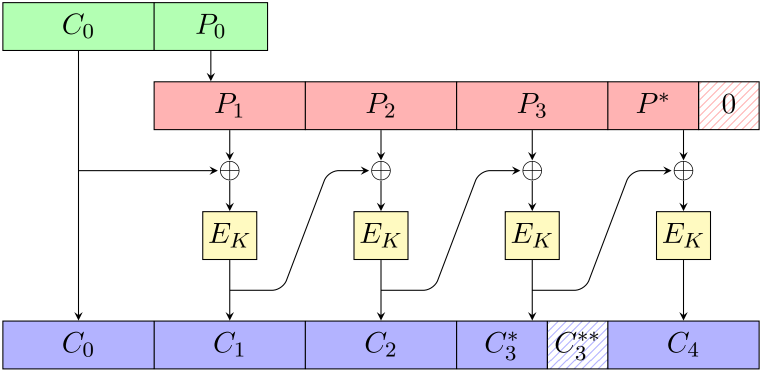
So, we've got to the end of a message, and we've not filled up the last block.

Delayed CBC with ciphertext stealing



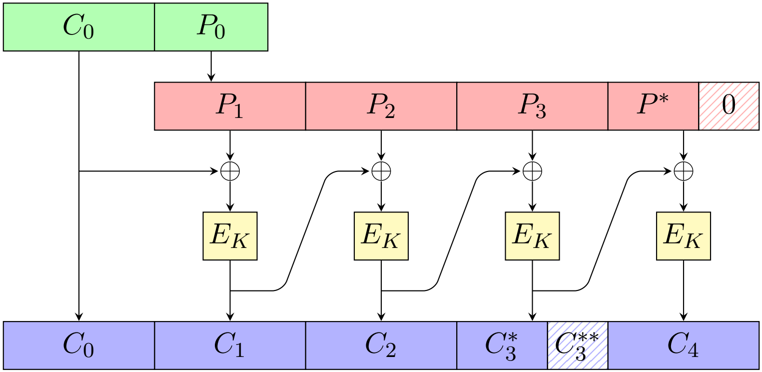
So we pad it with zero bits.

Delayed CBC with ciphertext stealing



The recipient can recover the tail of the next-to-last ciphertext block by decrypting the final one.

Delayed CBC with ciphertext stealing



Again, there are variants which differ in how they order the last two ciphertext blocks.

Delayed CBC with ciphertext stealing

- Actually the natural implementation. You have to hold back the last ciphertext block anyway, because you might have to truncate it. Indeed, for DCBC-CS3, you sometimes have to hold back *two* ciphertexts blocks.

Security of delayed CBC with ciphertext stealing

Theorem

Let $\mathcal{E}$ be any of DCBC-CS1$[\mathrm{Perm}(b)]$, DCBC-CS2$[\mathrm{Perm}(b)]$, *or* DCBC-CS3$[\mathrm{Perm}(b)]$ *and suppose adversary A asks queries totalling at most σ blocks. Then*

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{IND\$}}(A) \le \sigma^2/2^b$$

Security of delayed CBC with ciphertext stealing

Theorem

Let $\mathcal{E}$ be any of $\text{DCBC-CS1}[\text{Perm}(b)]$, $\text{DCBC-CS2}[\text{Perm}(b)]$, *or* $\text{DCBC-CS3}[\text{Perm}(b)]$ *and suppose adversary A asks queries totalling at most σ blocks. Then*

$$\mathbf{Adv}_{\mathcal{E}}^{\text{IND\$}}(A) \leq \sigma^2/2^b$$

Proof idea

- Describe DCBC-CSn in terms of a DCBC oracle. (Not quite as simple as the offline case: the state needs to be structured differently, and parts of it duplicated.)

□

Security of delayed CBC with ciphertext stealing

Theorem

Let $\mathcal{E}$ be any of $\textsc{DCBC-CS1}[\mathrm{Perm}(b)]$, $\textsc{DCBC-CS2}[\mathrm{Perm}(b)]$, *or* $\textsc{DCBC-CS3}[\mathrm{Perm}(b)]$ *and suppose adversary A asks queries totalling at most σ blocks. Then*

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{IND\$}}(A) \leq \sigma^2/2^b$$

Proof idea

- Describe $\textsc{DCBC-CS}n$ in terms of a $\textsc{DCBC}$ oracle. (Not quite as simple as the offline case: the state needs to be structured differently, and parts of it duplicated.)
- Observe that the postprocessing applied to the ciphertext preserves uniform distribution.

□

Security of delayed CBC with ciphertext stealing

Theorem

Let $\mathcal{E}$ be any of DCBC-CS1[Perm(b)], DCBC-CS2[Perm(b)], *or* DCBC-CS3[Perm(b)] *and suppose adversary A asks queries totalling at most σ blocks. Then*

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{IND\$}}(A) \le \sigma^2/2^b$$

Proof idea

- Describe DCBC-CSn in terms of a DCBC oracle. (Not quite as simple as the offline case: the state needs to be structured differently, and parts of it duplicated.)
- Observe that the postprocessing applied to the ciphertext preserves uniform distribution.
- Show reduction from DCBC security. □

The end