Introduction to

# Cryptography

Lecture 08-11

©Michael Luttenberger

Chair for Foundations of Software Reliability and Theoretical Computer Science
Technische Universität München

2018/11/14, 11:02

## Secure block ciphers

- So far, we have only defined when a "key stretcher" is secure (PRG).
- What's missing is the definition of "secure block cipher".
- As stated before, sCTR mode is often used with a block cipher $B$:

  $G(k, 1^{l \cdot q}) = B_k(\lfloor 1 \rceil) || B_k(\lfloor 2 \rceil) || \ldots || B_k(\lfloor q \rceil)$

▷ As we want sCTR-mode to be CPA-secure, we should require from a "secure block cipher" that

  $G_q(k) = B_k(\lfloor 1 \rceil) || B_k(\lfloor 2 \rceil) || \ldots || B_k(\lfloor q \rceil)$

  is a vlPRG (w.r.t. some concrete $(t, q, \varepsilon)$-bound).

  - Where $\lfloor \cdot \rceil$ is the chosen $l$-bit encoding of the counter value.

- "Technality": Which encoding should we assume? lsbf or msbf? Gray code? big or little endian? ...

## Secure block ciphers

- Idea: Let Eve define a "worst" encoding adaptively, i.e.:
  - Choose some secret $k$.
  - For $i = 1$ to some $q$:

    Eve may choose some $x_i \in \{0,1\}^l$ (her encoding of $i$) for which she is given $B_k(x_i)$.

    Her choice of $x_i$ may depend on $B_k(x_1)|| \dots ||B_k(x_i)$ seen so far.

▷ Still, Eve should not be able to distinguish the "reald world" (above) from the "perfect world":

  - In the "perfect world" she is given $y_i \overset{u}{\in} \{0,1\}^l$ instead of $B_k(x_i)$.
  - except, if $x_i = x_j$ for $i < j$, set $y_j := y_i$ to be consistent.

▷ **Remark**: For now we neglect that $B_k$ should be a permutation.

See later the notion of pseudorandom permutation.

## Keyed functions and permutations

- **Definition**:

▷ A keyed function $F$ ("random access key stretcher")

- takes as input a key $k \in \{0,1\}^n$,
- and a string $x \in \{0,1\}^{l_{\text{in}}(n)}$;
- and outputs a string $F_k(x) := F(k,x) \in \{0,1\}^{l_{\text{out}}(n)}$;
- is DPT-computable w.r.t. $(k,x)$.

$l_{\text{in}}$ is called the input length, $l_{\text{out}}$ the output length of $F$.

▷ A keyed permutation $P$ ("block cipher")

- is a keyed function with $l := l_{\text{in}} = l_{\text{out}}$,
- so that $F_k$ is a permutation on $\{0,1\}^{l(|k|)}$ for every fixed $k$;
- and there is a DPT-algorithm for computing $F_k(x)^{-1}$ from $(k,x)$.

$l$ is called the block length of $P$.

## Pseudorandom functions (PRF)

- **Definition**: Pseudoranom function (PRF)

  Let $F$ be a keyed function of input length $l_{\text{in}}$ and output length $l_{\text{out}}$.

  $F$ is a PRF if every PPT-alg. $\mathcal{D}$ has negl. adv. in the game $\textsc{IndPRF}$:

  1. Alice&Bob set up two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$:

| $\mathcal{O}_0$: "random function oracle (RFO)" | $\mathcal{O}_1$: replies using $F$ |
|---|---|
| Init: create an empty hashmap $T$ | Init: create $k \in \{0,1\}^n$ |
| Query: on input $x \in \{0,1\}^{l_{\text{in}}(n)}$ | Query: on input $x \in \{0,1\}^{l_{\text{in}}(n)}$ |
| if $T[x]$ is undefined, $T[x] := y \overset{u}{\in} \{0,1\}^{l_{\text{out}}(n)}$ | |
| return $T[x]$ | return $F_k(x)$. |

  They toss a fair coin $b \overset{u}{\in} \{0,1\}$ and pass $\mathcal{O}_b$ in a black box $\mathcal{O}$ to Eve.

  2. Eve runs $\mathcal{D}^{\mathcal{O}}(1^n)$ to obtain a reply $r$.

  ▷ Let $\text{Win}_{n,F}^{\textsc{IndPRF}}(\mathcal{D})$ denoe the event that $r = b$.

- **Remark**: We could also let RFO determine also all answers a-priori, i.e. choose a function from $\{0,1\}^{l_{\text{in}}(n)}$ to $\{0,1\}^{l_{\text{out}}}$ uniformly at random.

## Pseudorandom functions (PRF)

- **Ex**: $F_k(x) = x \oplus k$ with $n = l_{\text{in}}(n) = l_{\text{out}}(n)$ is not a PRF.
- **Ex**: Let $d(n) = l_{\text{in}}(n) = l_{\text{out}}(n) = \lfloor \sqrt{n} \rfloor$.

  Given $k \in \{0,1\}^n$, take the first $d(n)$ bits of $k$ to define a $d(n) \times d(n)$-matrix $M_k$:

$$M_k = \begin{pmatrix} k_0 & \cdots & k_{d(n)-1} \\ k_{d(n)} & \cdots & k_{2d(n)-1} \\ & \vdots & \\ k_{d(n)(d(n)-1)} & \cdots & k_{d(n)^2-1} \end{pmatrix}$$

  and define $F_k(x) = M_k \cdot x$ with matrix-vector mulitplication $\mathrm{mod} 2$.

  Is this a PRF?

## Pseudorandom functions (PRF)

- **Ex**: Assume Alice receives a phone call by someone claiming to be Bob. She now wants to verify that the caller is indeed Bob. How can she use a PRF $F$ and a key $k \in \{0,1\}^n$ exclusively known to her and Bob to do so?

  Your authentication method should work in the presence of an efficient eavesdropper Eve who may eavesdrop several phone calls between Alice and Bob and who eventually tries to impersonate Bob. Try to give a formal definition of the security you would require in this scenario.

## Pseudorandom functions (PRF)

- **Ex**: Let $F$ be PRF with $n = l_{\text{in}}(n) = l_{\text{out}}(n)$.

  For any PPT-encoding $\lfloor\rceil : \mathbb{Z}_{2^n} \to \{0,1\}^n$ and any polynomial $l(n)$

  $G_l(x) = F_k(\lfloor 1\rceil)||F_k(\lfloor 2\rceil)|| \ldots ||F_k(\lfloor l(n)\rceil)$ is a PRG of stretch $nl(n)$.

  Conclude that $F$-sCTR is CPA-secure if $F$ is a PRF.
- **Ex**: Show that PRFs with $l_{\text{out}}(n) \cdot 2^{l_{\text{in}}(n)} \leq n$ exist (unconditionally).
- **Ex**: Let $G$ be a PRG of stretch $l_G(n) = 2n$.

  Split $G(k) =: G_0(k)||G_1(k)$ into two $n$ bit strings.
  - Define $F_k^{(1)}(0) := G_0(k)$ and $F_k^{(1)}(1) := G_1(k)$.

    Show: $F^{(1)}$ is a PRF with $l_{\text{in}}(n) = 1$ and $l_{\text{out}}(n) = n$.
  - Set $F_k^{(2)}(x_1x_2) := G_{x_2}(F_k^{(1)}(x_1))$ for $x_1x_2 \in \{0,1\}^2$.

    Show*: $F^{(2)}$ is a PRF with $l_{\text{in}}(n) = 2$ and $l_{\text{out}}(n) = n$.

## Pseudorandom functions (PRF)

$$\begin{tikzcd}
 & & & k \arrow[dll, "0"'] \arrow[drr, "1"] & & & & \\
 & \begin{array}{c} G_0(k) \\ =: F_k^{(1)}(0) \end{array} \arrow[dl] \arrow[dr] & & & & \begin{array}{c} G_1(k) \\ =: F_k^{(1)}(1) \end{array} \arrow[dl] \arrow[dr] & & \\
\begin{array}{c} G_0(G_0(k)) \\ =: F_k^{(2)}(00) \end{array} \arrow[d] \arrow[dr] & & \begin{array}{c} G_1(G_0(k)) \\ =: F_k^{(2)}(01) \end{array} \arrow[d] \arrow[dr] & & \begin{array}{c} G_0(G_1(k)) \\ =: F_k^{(2)}(10) \end{array} \arrow[d] \arrow[dr] & & \begin{array}{c} G_1(G_1(k)) \\ =: F_k^{(2)}(11) \end{array} \arrow[d] \arrow[dr] & \\
\cdots & \cdots & \cdots & \begin{array}{c} G_1(G_1(G_0(k))) \\ =: F_k^{(3)}(011) \end{array} & \cdots & \cdots & \cdots & \cdots
\end{tikzcd}$$

- **Theorem**: Goldreich-Goldwasser-Micali theorem

  PRGs exist iff PRFs with $l_{\mathsf{in}}(n) = l_{\mathsf{out}}(n) = n$ exist.

▷ See e.g. Theorem 3.6.5 in [12].

- **Ex**: Why can't we simply take a PRG of stretch $n \cdot 2^n$, and read its output $G(k)$ as the table of a PRF $F_k$ with $n = l_{\mathsf{in}}(n) = l_{\mathsf{out}}(n)$?

  How does the tree construction avoid this problem?

## Pseudorandom functions (PRF)

- **Remarks**:
- Our definition of PRF is asymptotic, i.e. adjustable key (and block) length.
- **Ex**: Read here (Section 3.6) for the definition of PRF for fixed key and block length.
- PRFs are almost "secure block ciphers" – almost, as they do not need to be a permutation for a given key.
- Better think of PRFs as vlPRGs plus random access to their output.
- The random access allows to build stateless ES which are CPA-secure.

▷ **Ex**: Why might a stateless ES be more convenient?

## Randomized counter (rCTR) mode

▷ **Definition**: rCTR mode for a PRF $F$

- $\mathcal{K}_n = \{0,1\}^n$, $\mathcal{M}_n = \left(\{0,1\}^{l_{\mathsf{out}}(n)}\right)^*$, $\mathcal{C}_n = \{0,1\}^{l_{\mathsf{in}}(n)} \left(\{0,1\}^{l_{\mathsf{out}}(n)}\right)^*$
- Gen: On input $1^n$, output $k \overset{u}{\in} \mathcal{K}_n$.
- Enc: On input $k \in \mathcal{K}_n$ and $m = m^{(1)} || \ldots || m^{(t)} \in \mathcal{M}_n$,

  choose $\mathrm{ctr} \overset{u}{\in} \mathbb{Z}_{2^{l_{\mathsf{in}}(n)}}$, and set $c^{(0)} := \lfloor \mathrm{ctr} \rceil$;

  for $i = 1$ to $i = t$ compute $c^{(i)} := m^{(i)} \oplus F_k(\lfloor \mathrm{ctr} + i \rceil)$;

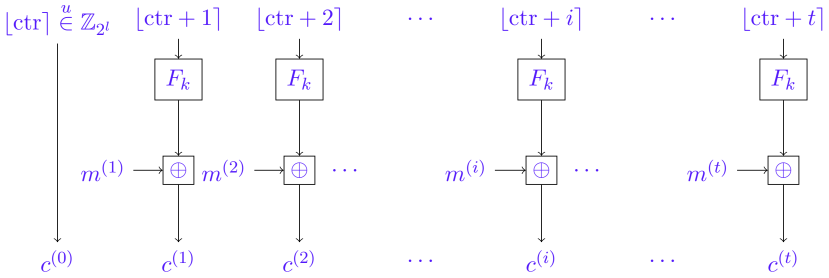
  output $c = c^{(0)} || c^{(1)} || \ldots || c^{(t)}$.
- Dec: **Ex**

- **Theorem**: $F$-rCTR is CPA-secure if $F$ is a PRF.
- **Ex**: Above definition assumes for simplicity that messages are a multiple of $l_{\mathsf{out}}(n)$.

  Adapt the definition s.t. $\mathcal{M}_n = \{0,1\}^*$ and $|c| = |m| + l_{\mathsf{out}}(n)$.

## Randomized counter (rCTR) mode

- Schematic representation of $F$-rCTR:


$\lfloor \mathrm{ctr} \rceil \overset{u}{\in} \mathbb{Z}_{2^l}$
$\lfloor \mathrm{ctr}+1 \rceil$
$\lfloor \mathrm{ctr}+2 \rceil$
$\cdots$
$\lfloor \mathrm{ctr}+i \rceil$
$\cdots$
$\lfloor \mathrm{ctr}+t \rceil$
$F_k$
$F_k$
$F_k$
$F_k$
$m^{(1)}$
$m^{(2)}$
$m^{(i)}$
$m^{(t)}$
$c^{(0)}$
$c^{(1)}$
$c^{(2)}$
$c^{(i)}$
$c^{(t)}$

- Analogous to $F$-sCTR mode: parallel processing of message blocks.

## Randomized counter (rCTR) mode

- Proof sketch:
- Assume that the used PRF $F$ represents a block cipher of block length $l(n) = l_{\text{in}}(n) = l_{\text{out}}(n)$.
- Canonical approach:

  Let $\mathcal{A}$ be any PPT-adversary for $F$-rCTR in the game $\textsc{IndCPA}$.

  Build a PPT-adversay $\mathcal{D}$ for $F$ in the game $\textsc{IndPRF}$.

  $\mathcal{D}$ plays the game $\textsc{IndCPA}$ vs. $\mathcal{A}$.

  $\mathcal{D}$ uses $\mathcal{O}$-rCTR to encrypt all messages.

  - Both the queries by $\mathcal{A}$ and the encryption by Alice&Bob.

▷ Recall:

W.l.o.g. we may assume that $\mathcal{A}$ only outputs a single message pair.

## Randomized counter (rCTR) mode

| Alice&Bob | $\mathcal{D}$ | sub: $\mathcal{A}$ |
|---|---|---|
| $b \overset{u}{\in} \{0,1\}$ | | |
| $b=0$: $\mathcal{O} :=$ RFO | | |
| $b=1$: $k \overset{u}{\in} \{0,1\}^n$; $\mathcal{O} := F_k$ | | |
| pass $\mathcal{O}$ to $\mathcal{D}$ | | |
| | (use $\mathcal{O}$ for $\mathsf{Enc}_k$) | |
| | run $\mathcal{A}^{\mathsf{Enc}_k}(1^n)$ | |
| | | return $(m_0, m_1)$ ($\lvert m_0\rvert = \lvert m_1\rvert$) |
| | $b' \overset{u}{\in} \{0,1\}$ | |
| | $c = \mathsf{Enc}_k(m_{b'})$ | |
| | run $\mathcal{A}^{\mathsf{Enc}_k}(1^n, c)$ | |
| | | return $r'$ |
| | return $r := (r' \overset{?}{=} b')$ | |

- “Real world” ($b=1$): $\mathcal{D}$ wins iff $\mathcal{A}$ wins vs. $F$-rCTR.
- “Perfect world” ($b=0$): $\mathcal{D}$ wins iff $\mathcal{A}$ loses vs. RFO-rCTR.
    - Previously (prOTP/sCTR): Both schemes yield a OTP in the perfect world so that $\mathcal{A}$ wins with prob. $1/2$.
    - Now (rCTR): Argue that rCTR mode with a perfect block cipher RFO yields a OTP almost all the time.

## Randomized counter (rCTR) mode

- RFO-rCTR:

▷ on input $m = m^{(1)}||\ldots||m^{(t)}$, choose $\mathrm{ctr} \overset{u}{\in} \mathbb{Z}_{2^{l(n)}}$,

output $\lfloor \mathrm{ctr} \rceil || m^{(1)} \oplus \mathsf{RFO}(\lfloor \mathrm{ctr}+1 \rceil)||\ldots||m^{(1)} \oplus \mathsf{RFO}(\lfloor \mathrm{ctr}+t \rceil)$.

▷ Let $\vec{x} = (x_1, \ldots, x_{q(n)})$ be the sequence of all values for which RFO is queried:

- $x_i$ is the $i$-th query; the queries do not need to be distinct.
- $q(n)$ is the total number of encrypted message blocks in the course of game $\textsc{IndCPA}$.

▷ Overapproximation: If $\vec{x}$ consists of $q(n)$ distinct values,

then just as in sCTR mode:

the sequence of all message blocks is encrypted using one big OTP by definition of RFO.

## Randomized counter (rCTR) mode

- Prob. $\Pr[\text{“no OTP”}]$, i.e. some $x_i$ is used twice.
- For simplicity, assume that all $x_i$ are chosen uniformly at random and independently of each other.
  - E.g. when $\mathcal{A}$ only uses messages of length exactly $0^{l(n)}$.

▷ $\Pr_{x_i,x_j \overset{u}{\in} \{0,1\}^{l(n)}}[x_i = x_j] = 2^{-l(n)}$

▷ $\Pr[\text{“no OTP”}] = \Pr_{x_1,\ldots,x_q \overset{u}{\in} \{0,1\}^{l(n)}}\left[\bigvee_{i \neq j} x_i = x_j\right] \leq \binom{q(n)}{2} 2^{-l(n)}$.

- **Ex**: Assume that in total $e(n)$ messages are encrypted, each consisting of at most $s(n)$ blocks.

  Show that $\Pr[\text{“no OTP”}] \leq \frac{e(n)^2 s(n)}{2^{l(n)}}$.

- Safe upper bound: $\Pr[\text{“no OTP”}] \leq \frac{q(n)^2}{2^{l(n)}}$.

▷ “rCTR mode yields a OTP in the perfect world except for a negl. fraction of cases.”

## Randomized counter (rCTR) mode

- So, with $\mathcal{E}$ the ES used by $\mathcal{D}$ in the "perfect world":

$$\begin{aligned}\Pr\left[\mathsf{Win}^{\textsc{IndCPA}}_{n,\mathsf{RFO\text{-}rCTR}}(\mathcal{A})\right] &\le \tfrac{1}{2}\cdot\Pr[\text{"OTP"}] + 1\cdot\Pr[\text{"no OTP"}]\\ &\le \tfrac{1}{2} + \tfrac{q(n)^2}{2^{l(n)}}\end{aligned}$$

- All in all:

$$\begin{aligned}&2\Pr\left[\mathsf{Win}^{\textsc{IndPRF}}_{n,F}(\mathcal{D})\right]\\ =\ & \Pr\left[\mathsf{Win}^{\textsc{IndCPA}}_{n,F\text{-}\mathsf{rCTR}}(\mathcal{A})\right] + 1 - \Pr\left[\mathsf{Win}^{\textsc{IndCPA}}_{n,\mathsf{RFO\text{-}rCTR}}(\mathcal{A})\right]\\ \ge\ & \Pr\left[\mathsf{Win}^{\textsc{IndCPA}}_{n,F\text{-}\mathsf{rCTR}}(\mathcal{A})\right] + \tfrac{1}{2} - \tfrac{q(n)^2}{2^{l(n)}}\end{aligned}$$

▷ I.e.:

$$\left|\Pr\left[\mathsf{Win}^{\textsc{IndCPA}}_{n,F\text{-}\mathsf{rCTR}}(\mathcal{A})\right] - \tfrac{1}{2}\right| \le 2\left|\Pr\left[\mathsf{Win}^{\textsc{IndPRF}}_{n,F}(\mathcal{D})\right] - \tfrac{1}{2}\right| + \tfrac{q(n)^2}{2^{l(n)}}$$

## Randomized counter (rCTR) mode

▷ Consquence of this analysis:

Besides the key length, also the block length of a block cipher should be sufficiently large – at least when used in rCTR mode.

- Probability for a collision (see appendix) when choosing $q$ values uniformly from $\{1, \ldots, N\}$ independently of each other:

  $\Theta(\frac{q^2}{2N})$.

- E.g. for DES we have $l = 64$, i.e. $N = 2^{64}$.

  If Eve can obtain $q = 2^{32}$ blocks ($32$ GiB) of encrypted data, the security bound becomes meaningless, and with prob. roughly $1/2$ some random value for ctr is used at least twice.

## Randomized output feedback (rOFB) mode

▷ **Definition**: rOFB for a PRF $F$ with $l(n) = l_{\text{in}}(n) = l_{\text{out}}(n)$

- $\mathcal{K}_n = \{0,1\}^n$, $\mathcal{M}_n = \left(\{0,1\}^{l(n)}\right)^*$, $\mathcal{C}_n = \{0,1\}^{l(n)} \left(\{0,1\}^{l(n)}\right)^*$
- Gen: On input $1^n$, output $k \overset{u}{\in} \mathcal{K}_n$.
- Enc: On input $k \in \mathcal{K}_n$ and $m = m^{(1)}||\ldots||m^{(t)} \in \mathcal{M}_n$,

  choose $\mathrm{IV} \overset{u}{\in} \{0,1\}^{l(n)}$ and set $c^{(0)} = \mathrm{IV}$, and

  for $i = 1$ to $i = t$ compute $c^{(i)} := m^{(i)} \oplus F_k^i(\mathrm{IV})$;

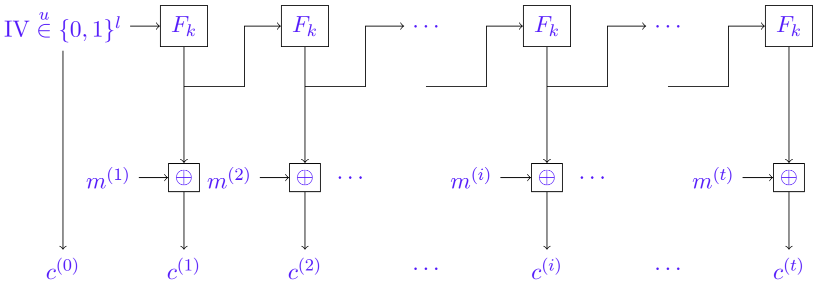
  output $c = c^{(0)}||c^{(1)}||\ldots||c^{(t)}$.
- Dec: **Ex**

- **Theorem** (w/o proof): $F$-rOFB is CPA-secure if $F$ is a PRF.

▷ Idea: in the perfect world (RFO instead of $F_k$) – except for negl. prob. of a collision – RFO is never queried on the same input twice. So, we get a prOTP again.

## Randomized output feedback (rOFB) mode

- Schematic representation of OFB mode for $l(n) = l_{\text{in}}(n) = l_{\text{out}}(n)$



- To process the $i$-th block of data, we need to know $F_k^i(\mathrm{IV})$ for OFB while s/rCTR mode only requires knowledge of $F_k(\lfloor \mathrm{ctr} + i \rceil)$.

▷ **Ex**: Is $F$-rOFB CCA-secure for $F$ a PRF?

- **Ex**: Again, adapt the definition of rOFB s.t. $\mathcal{M}_n = \{0,1\}^*$ and $|c| = |m| + l_{\text{out}}(n)$.

## Randomized cipher feedback (rCFB) mode

▷ **Definition**: rCFB for a PRF $F$ with $l(n) = l_{\text{in}}(n) = l_{\text{out}}(n)$

- $\mathcal{K}_n = \{0,1\}^n$, $\mathcal{M}_n = \left(\{0,1\}^{l(n)}\right)^*$, $\mathcal{C}_n = \{0,1\}^{l(n)} \left(\{0,1\}^{l(n)}\right)^*$
- Gen: On input $1^n$, output $k \overset{u}{\in} \mathcal{K}_n$.
- Enc: On input $k \in \mathcal{K}_n$ and $m = m^{(1)}||\ldots||m^{(t)} \in \mathcal{M}_n$,

  choose $\mathrm{IV} \overset{u}{\in} \{0,1\}^{l(n)}$ and set $c^{(0)} = \mathrm{IV}$, and

  for $i = 1$ to $i = t$ compute $c^{(i)} := m^{(i)} \oplus F_k^i(c^{(i-1)})$;

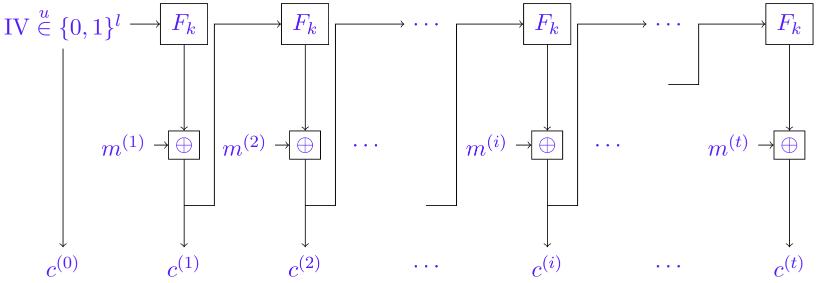
  output $c = c^{(0)}||c^{(1)}||\ldots||c^{(t)}$.
- Dec: **Ex**

- **Theorem** (w/o proof): $F$-rCFB is CPA-secure if $F$ is a PRF.

▷ Same idea as rOFB: except for an input collision, a RFO will define RFO$(c^{(i-1)})$ as a truly random string; hence, also $c^{(i)} = m^{(i)} \oplus$ RFO$(c^{(i-1)})$ will be truly random string with negligible collision prob. So, we will get an OTP in the prefect world except for negligible probability.

## Randomized cipher feedback (rCFB) mode

- Schematic representation of OFB mode for $l(n) = l_{\text{in}}(n) = l_{\text{out}}(n)$



- To process the $i$-th block of data, we need to know $F_k(c^{(i-1)})$ for CFB; so CFB is an improvement over OFB, as we only need one PRF call to decrypt a single block – neglecting transmission errors.

▷ **Ex**: Is $F$-rCFB CCA-secure for $F$ a PRF?

- **Ex**: Again, adapt the definition of rCFB s.t. $\mathcal{M}_n = \{0,1\}^*$ and $|c| = |m| + l_{\text{out}}(n)$.

1. Lecture 8 – PRFs and Modes of Operations
2. Lecture 9 – PRPs, CBC mode, MACs
3. Lecture 10 – EtM, CCA-sec/AE, practical MACs
4. Lecture 11 – Conjectured pratical PRPs/PRFs: DES, Rijndael, *Keccak

## Definition of secure block cipher

- Recall: Our definition of a PRF was based on the observation that rCTR mode essentially uses a block cipher $F$ as PRG.
  - Would like to have that $F_k(x)$ is indistinguishable from a OTP for any input $x$.
  - I.e. $F$ instantiated on the secret random $k$ should be indistingushable from a (RFO).
- But if $F$ is a block cipher, then $F_k$ is invertible for every key $k$.

▷ The more values $F_k(x_i)$ we know, the less random the output of $F_k$ becomes as $F_k(x_{i+1}) \notin \{F_k(x_0), \ldots, F_k(x_i)\}$.

▷ So, we will now compare $F_k$ to random permutation oracle (RPO).

## Random permutation oracle

"random permutation oracle RPO"

```
Init: create an empty hashmap $T$ and empty set $S$
Query: on input $x \in \{0,1\}^{l(n)}$
if $T[x]$ is undefined: $T[x] := (y \overset{u}{\in} \{0,1\}^{l(n)} \setminus S)$
$S := S \cup \{T[x]\}$
return $T[x]$.
```

- So far an RPO does not answer preimage-queries (later).
- As in the case of RFO, one can imagine that a RPO
  - either chooses the random permutation once before the first query
  - or defines it on the fly; but now it has to ensure that the map is always injective.

  ▷ Note: Ratio permutations to functions over $\{0,1\}^l$ is negligible:

$$\frac{2^l!}{(2^l)^{2^l}} \approx \frac{\sqrt{2\pi}2^{l/2}(2^l/e)^{2^l}}{2^{l2^l}} = \sqrt{2\pi}2^{l/2}e^{-2^l}$$

## Detour: Random oracles

- Different kinds of random oracles, e.g.:
  - RFO: random function oracle with some fixed input and output length; implements a random function (on the fly).
  - RPO: random permutation oracle with some fixed block length; implements a random permutation (on the fly).

    Depending on the application, a RPO may also answer queries regarding the preimage of an element (see strong PRP).
  - RO: random oracle in its most generic form; implements a random function from $\{0,1\}^*$ to $\{0,1\}^\omega$ (from finite bit strings to infinite bit strings).

    Implicitly it is always assumed that the output of RO is truncated to the length required by the algorithm or protocoll.

▷ Note that we can have several independent (instances of) RFOs; then every RFO implements a independently chosen random function (same for RPO and RO).

## Pseudorandom permutations (PRP)

- **Definition**:

  Let $F$ be a keyed permutation ("'block cipher") of block length $l$.

  $F$ is a PRP if every PPT-alg. $\mathcal{D}$ has negl. adv. in the game $\textsc{IndPRP}$:

  1. Alice&Bob set up two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$:

| $\mathcal{O}_0$: "random permutation oracle RPO" | $\mathcal{O}_1$: replies using $F$ |
|---|---|
| Init: create an empty hashmap $T$ and empty set $S$ | Init: create $k \in \{0,1\}^n$ |
| Query: on input $x \in \{0,1\}^{l(n)}$ | Query: on input $x \in \{0,1\}^{l(n)}$ |
| if $T[x]$ is undefined: $T[x] := (y \overset{u}{\in} \{0,1\}^{l(n)} \setminus S)$ | |
| $S := S \cup \{T[x]\}$ | |
| return $T[x]$ | return $F_k(x)$. |

  They toss a fair coin $b \overset{u}{\in} \{0,1\}$ and pass $\mathcal{O}_b$ in a black box $\mathcal{O}$ to Eve.

  2. Eve runs $\mathcal{D}^{\mathcal{O}}(1^n)$ to obtain a reply $r$.

  ▷ Let $\mathrm{Win}_{n,F}^{\textsc{IndPRP}}(\mathcal{D})$ denoe the event that $r = b$.

- Definition of $(t, q, \varepsilon)$-PRP as for PRF.

## Pseudorandom permutations (PRP)

- **Definition**: strong PRP

  Definition of PRP but the oracles also can compute the inverse.

  - $\mathcal{O}_1$ answers normal queries using $F_k$, and “inverse” queries using $F_k^{-1}$.
  - $\mathcal{O}_0$ uses e.g. a second hash map $T_{\mathsf{inv}}$ with $T_{\mathsf{inv}}[y] = x$ iff $T[x] = y$.

▷ Strong PRPs required for the CCA-security of some ES.

▷ (Strong) PRPs can be constructed from PRFs using a construction underlying the DES-cipher, called Feistel network.

## Pseudorandom permutations (PRP)

- **Definition**:

  Let $f : \{0,1\}^* \to \{0,1\}^*$ be some function s.t. $|f(x)| = |x|$ for all $x \in \{0,1\}^*$.

  A single-round Feistel network $\mathsf{FN}_f$ is defined by

  $$\mathsf{FN}_f(x||y) := y||x \oplus f(y) \text{ for all } x, y \in \{0,1\}^* \text{ with } |x| = |y|\,.$$

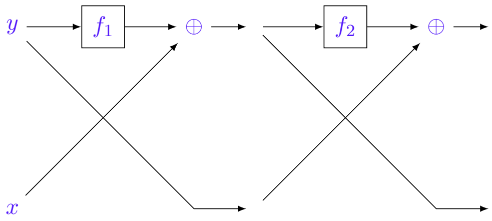
  Similarly, given functions $f_1, \ldots, f_j$ a $j$-round Feistel network is inductively defined by

  $$\mathsf{FN}_{f_1,f_2,\ldots,f_j}(x||y) := \mathsf{FN}_{f_j}(\mathsf{FN}_{f_1,f_2\ldots,f_{j-1}}(x||y))$$

- **Proposition**: (Ex.)

  Independent of the choice of $f_1, \ldots, f_j$ the function $\mathsf{FN}_{f_1,\ldots,f_j}$ is invertible when $f_1, \ldots, f_j$ are known.

## Pseudorandom permutations: Feistel networks



- **Theorem**: Luby/Rackoff (e.g. [12])

  Let $F(\cdot,\cdot)$ be a PRF and choose $a, b, c, d \overset{u}{\in} \{0,1\}^n$ independently and uniformly at random. Then:

  - $F^{(3)}_{a||b||c}(x) := \mathsf{FN}_{F_a,F_b,F_c}(x||y)$ is a PRP on $\{0,1\}^{2n}$ using $3n$-bit keys.
  - $F^{(4)}_{a||b||c||d}(x) := \mathsf{FN}_{F_a,F_b,F_c,F_d}(x||y)$ is a strong PRP on $\{0,1\}^{2n}$ using $4n$-bit keys.

## PRPs as PRFs

- For the modes s/rCTR and rOFB we do not need a PRP.
- In fact, we rather would like to have PRF because of our proofs.

▷ Can we treat a PRP as a PRF?

- Difference between the games $\textsc{IndPRF}$ and $\textsc{IndPRP}$:

| $\textsc{IndPRP}$: RPO | $\textsc{IndPRF}$: RFO |
|---|---|
| Init: create an empty hashmap $T$ and empty set $S$ | Init: create an empty hashmap $T$ |
| Query: on input $x \in \{0,1\}^{l(n)}$ | Query: on input $x \in \{0,1\}^{l(n)}$ |
| if $T[x]$ is undefined: $T[x] := (y \overset{u}{\in} \{0,1\}^{l(n)} \setminus S)$ | if $T[x]$ is undefined: $T[x] := (y \overset{u}{\in} \{0,1\}^{l(n)})$ |
| $S := S \cup \{T[x]\}$ | |
| return $T[x]$ | return $T[x]$. |

▷ All that matters to $\mathcal{D}$ is the answers it gets from its oracle $\mathcal{O}$.

▷ If RFO does not generate a collision, any $\mathcal{D}$ has to behave just as if it was given RPO:

$$\Pr\left[\mathcal{D}^{\mathsf{RFO}}(1^n) = 0 \mid \text{no collision}\right] = \Pr\left[\mathcal{D}^{\mathsf{RPO}}(1^n) = 0\right]$$

## PRPs as PRFs

- If $\mathcal{D}$ makes $q(n)$ oracle queries (birthday problem):

  $c_{q(n)} := \Pr[\text{collision}] = \theta \cdot \binom{q(n)}{2} 2^{-l(n)}$ for some $\theta \in [1/2, 1]$
- So with $\Pr\left[\mathcal{D}^{\mathsf{RFO}}(1^n) = 0 \mid \text{no collision}\right] = \Pr\left[\mathcal{D}^{\mathsf{RPO}}(1^n) = 0\right]$:

$$\begin{aligned} & \left|\Pr\left[\mathcal{D}^{\mathsf{RFO}}(1^n) = 0\right] - \Pr\left[\mathcal{D}^{\mathsf{RPO}}(1^n) = 0\right] (c_{q(n)} + 1 - c_{q(n)})\right| \\ = \; & c_{q(n)} \left|\Pr\left[\mathcal{D}^{\mathsf{RFO}}(1^n) = 0 \mid \text{col.}\right] - \Pr\left[\mathcal{D}^{\mathsf{RPO}}(1^n) = 0\right]\right| \\ \leq \; & 2c_{q(n)} \leq q(n)^2 2^{-l(n)-1} \end{aligned}$$

- I.e. RFO and RPO look the same to any $\mathcal{D}$ except for negl. prob.
- Hence: $\Pr\left[\mathsf{Win}_{n,F}^{\textsc{IndPRF}}(\mathcal{D})\right] \leq \Pr\left[\mathsf{Win}_{n,F}^{\textsc{IndPRP}}(\mathcal{D})\right] + q(n)^2 2^{-l(n)-2}$
- **Theorem**:

  If $F$ is a PRP (with $2^{-l(n)}$ negligible), then it is also a PRF.

## PRPs as PRFs

- **Ex**: Does also the other direction hold?

  That is, if $F$ is a PRF, then may we use it also a PRP?

  (Leaving aside the problem of how to compute the inverse of $F$.)

- **Ex**: Show that there is a PPT-adversary which distinguishes a PRP $F$ from RFO with a negligible, but nonzero advantage.

- **Remark**:

  For the CPA-security of $F$-sCTR, $F$-rCTR, $F$-rOFB we assumed that $F$ is a PRF.

  By the preceding result, when $F$ is instead only a PRP, the advantage of the constructed adversaries $\mathcal{D}$ increases by roughly $\frac{q(n)^2}{2^{l(n)+1}}$.

  So, also for $F$-sCTR the block length of $F$ matters, and the security bounds for $F$-sCTR and $F$-rCTR become roughly the same.

## Randomized cipher block chaining (rCBC) mode

▷ **Definition**: rCBC mode for a PRP $F$ with block length $l(n)$

- $\mathcal{K}_n = \{0,1\}^n$, $\mathcal{M}_n = \left(\{0,1\}^{l(n)}\right)^*$, $\mathcal{C}_n = \{0,1\}^{l(n)} \left(\{0,1\}^{l(n)}\right)^*$
- Gen: On input $1^n$, output $k \overset{u}{\in} \mathcal{K}_n$.
- Enc: On input $k \in \mathcal{K}_n$ and $m = m^{(1)}||\ldots||m^{(t)} \in \mathcal{M}_n$,

  choose $\mathrm{IV} \overset{u}{\in} \{0,1\}^{l(n)}$ and set $c^{(0)} = \mathrm{IV}$, and

  for $i = 1$ to $i = t$ compute $c^{(i)} := F_k(c^{(i-1)} \oplus m^{(i)})$;

  output $c = c^{(0)}||c^{(1)}||\ldots||c^{(t)}$.
- Dec: **Ex**

- **Theorem** (see e.g. here): $F$-rCBC is CPA-secure if $F$ is a PRP.

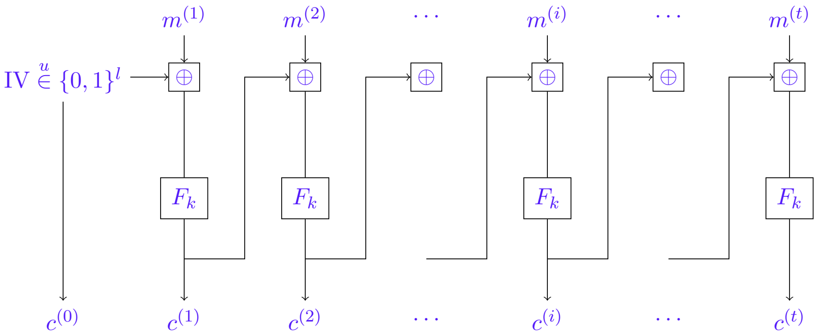
▷ Idea: Similar to rOFB mode:

- If $c^{(i-1)}$ is uniformly distributed, so is $c^{(i-1)} \oplus m^{(i)}$.

  ▷ If no collision, RPO$(c^{(i-1)} \oplus m^{(i)})$ is again uniformly distributed (RFO $\approx$ RPO).

## Randomized cipher block chaining (rCBC) mode

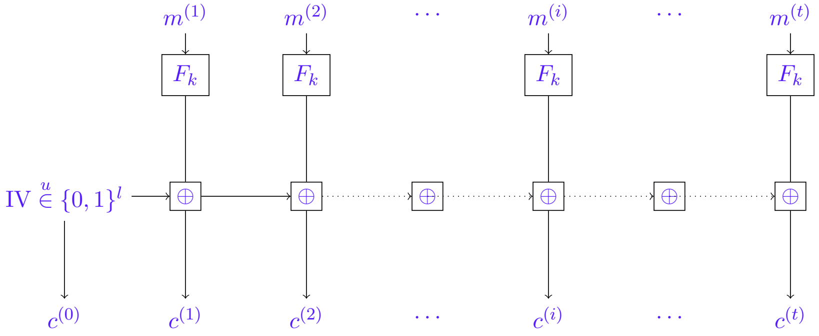
- Schematical representation of rCBC mode:



- Main drawback: no parallel encryption as data blocks depend on each other.
- **Ex**: Is $F$-rCBC CCA-secure for $F$ a PRP?

## Randomized cipher block chaining (rCBC) mode

- **Ex**: (Question 2016) Is the following variant of rCBC secure? (XOR has been moved after the block cipher call.)



## Randomized cipher block chaining (rCBC) mode

- Stateful variant: CBC with chained $\mathrm{IV}$ (CBC-CIV)
  - Used e.g. in the binary packet protocol of SSH 2.0 and in TLS prior 1.1 (BEAST attack).
- $\mathrm{IV}$ becomes the state

▷ Initially: $\mathrm{IV} \overset{u}{\in} \{0,1\}^{l(n)}$.

▷ After each encryption: $\mathrm{IV}$ becomes the last block of the just computed ciphertext.

▷ I.e. it can be seen as rCBC-mode where all messages are concatenated into a single large message (stream) ... almost.

- **Ex**: Show that CBC-CIV is not CPA-secure.

▷ See also [4].

## Message origin authentication for CCA-security

- Recall: game $\textsc{IndCCA}$

  ❶ Alice&Bob generate a random key $k := \mathsf{Gen}(1^n)$ and give Eve's attack $\mathcal{A}$ oracle access to $\mathsf{Enc}_k$ and $\mathsf{Dec}_k$.

  ❷ Eve runs $\mathcal{A}^{\mathsf{Enc}_k,\mathsf{Dec}_k}(1^n)$ to obtain two sequences $\vec{m}_0, \vec{m}_1$ with $\vec{m}_b = (m_b^{(1)}, \ldots, m_b^{(q)})$, and $\left|{m_0}^{(j)}\right| = \left|{m_1}^{(j)}\right|$, and $m_b^{(i)} \in \mathcal{M}_n$.

  ❸ Alice&Bob choose $b \in \{0,1\}$ by tossing a fair coin, compute $c^{(i)} = \mathsf{Enc}_k(m_b^{(i)})$ from left $i = 1$ to right $i = q$, and send $\vec{c} = (c^{(1)}, \ldots, c^{(q)})$ to Eve.

  ❹ Eve runs $\mathcal{A}^{\mathsf{Enc}_k,\mathsf{Dec}_k}(1^n, \vec{c})$ to obtain her reply $r \in \{0,1\}$ where $\mathcal{A}^{\mathsf{Enc}_k,\mathsf{Dec}_k}$ is not allowed to query $\mathsf{Dec}_k$ for any $c^{(i)}$.

  ▷ Let $\mathsf{Win}_{n,\mathcal{E}}^{\textsc{IndCCA}}$ be the event that $b = r$.

- Idea: want to reuse CPA-results, so make the $\mathsf{Dec}_k$-oracle useless

▷ Extend ciphertext by a message authentication code (MAC): $c||t$

▷ Goal: $\mathsf{Dec}_k$ can use $t$ to detect and reject $c$ originating from $\mathcal{A}$.

## Message authentication code (MAC)

- **Definition**: A MAC scheme consists of

| Algorithm | Type | Input | Output |
|---|---|---|---|
| Gen | PPT | $1^n$ | $k \overset{r}{\in} \mathcal{K}_n$ with $\lvert k \rvert \geq n$ |
| Mac | PPT | $k \in \mathcal{K}_n$, $m \in \mathcal{M}_n$ | $t \in \mathcal{T}_n$ |
| Vrf | DPT | $k \in \mathcal{K}_n$, $m \in \mathcal{M}_n$, $t \in \mathcal{T}_n$ | $r \in \{0,1\}$ |

where for all $k \in \mathcal{K}_n$: $\mathsf{Vrf}_k(m, \mathsf{Mac}_k(m)) = 1$ iff $m \in \mathcal{M}_n$.

- $\mathcal{K}_n/\mathcal{M}_n/\mathcal{T}_n$: key/message/tag space
- Gen/Mac/Vrf: key generator/mac generator/mac verifier
- Alice sends $m||\mathsf{Mac}_k(m)$ to Bob.

A MAC scheme is

- of fixed-length $l(\cdot)$ if $\mathcal{M}_n = \{0,1\}^{l(n)}$.
- deterministic if Mac is a DPT-algorithm.
- stateful if $\mathsf{Mac}_k$ saves some state (e.g. message counter) between two runs; otherwise it is stateless.

## Message authentication code (MAC)

- What should a MAC achieve?

▷ Only the knowledge of the secret $k_{\mathsf{Mac}}$ should enable a person to create, for some message $m$, a tag $t$ which is valid for $m$ w.r.t. $k_{\mathsf{Mac}}$:

  - I.e. $\mathsf{Vrf}_{k_{\mathsf{Mac}}}(m, t) = 1$.

▷ Without the knowledge of $k_{\mathsf{Mac}}$, an adversary should succeed only with negl. prob. in forging a tag for a message $m$.

  - Which messages $m$?

  ▷ Let the adversary choose $m$ to make the definition as general as possible.

  ▷ So, we do not need to care what particular subset of $\mathcal{M}_n$ our application is using.

  - Any message chosen by the adversary?

  ▷ The adversary knows valid tags for all eavesdropped messages.

▷ Secure MAC: Forge a valid tag for a message not seen so far.

## Security of MACs

- **Definition**: Game FrgMAC:

  1. Alice& Bob generate $k \overset{r}{:=} \mathsf{Gen}(1^n)$, and give Eve oracle access to $\mathsf{Mac}_k$.

     Alice&Bob keep a list $\mathcal{Q}$ of Eve's oracle queries.

  2. Eve runs $\mathcal{A}^{\mathsf{Mac}_k}(1^n)$ to obtain $(m, t)$.

  ▷ Let $\mathsf{Win}^{\mathsf{FrgMAC}}_{n,\mathcal{S}}(\mathcal{A})$ be the event that (i) $\mathsf{Vrf}_k(m, t) = 1$ and (ii) $m \notin \mathcal{Q}$.

  A MAC scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrf})$ is existentially unforgeable under an adaptive chosen-message attack (short: secure) if for every PPT-adversary $\mathcal{A}$ the prob. $\Pr\left[\mathsf{Win}^{\mathsf{FrgMAC}}_{n,\mathcal{S}}(\mathcal{A})\right]$ is negligible in $n$:

- A secure MAC is not required to provide message privacy!

▷ The MAC tag $t$ is simply appended to the message $m$: $m||t$.

- **Remark**: Above definition assumes that Eve can compute $\mathsf{Vrf}_k$ herself. This is the case for all MACs discussed here (**Ex.**). In general, the definition would Eve also give oracle access to $\mathsf{Vrf}_k$.

## Message integrity/origin authentication

- Note: If $\mathsf{Vrf}_k(m,t)=1$, then $t$ will always be a valid mac tag for $m$

  That is: Eve can *replay* $m||t$ at some later point;

  if Bob is a server he must be prevented to not behave the same way as when he first saw that message, although the tag is valid.

▷ To do so, a state needs to be introduced into the communication protocol which guarantees that a message $m$ can be sent at most once:

▷ Append to each message a nonce (number used once)

  - E.g. (a combination of) time-stamps, (pseudo)random numbers, message counters.

▷ Then send $m||\lfloor\text{nonce}\rceil||\mathsf{Mac}_k(m||\lfloor\text{nonce}\rceil)$ instead of $m||\mathsf{Mac}_k(m)$.

- Several technical problems need to be overcome e.g. synchronization of message counters or of clocks.

## MACs from encryption schemes?

- **Example**: $B$-r/sCTR as MAC?

  For simplicity, assume a block cipher of block length $l = 104$.

  Set $\mathsf{Mac}_k(m) := \mathrm{ctr}||m^{(1)} \oplus F_k(\lfloor \mathrm{ctr}+1\rceil)||\dots||m^{(t)} \oplus F_k(\lfloor \mathrm{ctr}+t\rceil)$.

  Let $m$ be the 8bit-ASCII encoding of "send 2 pizzas", i.e. $|m| = 104$.

  Then $(m,t)$ is a $39$ byte string, e.g.,

  $$\underbrace{\text{send 2 pizzas}}_{=m}\underbrace{\text{pk2njM\%f-9231}}_{=\lfloor \mathrm{ctr}\rceil}\underbrace{\text{r-9wfvwaXR,\#bP}}_{=m\oplus F_k(\lfloor \mathrm{ctr}\rceil+1)}$$

  Eve can turn $2 \hat{=} 00110010$ into $6 \hat{=} 00110110$ by flipping a single bit.

  To make the tag valid again, simply flip the corresponding bit in $m \oplus F_k(\lfloor \mathrm{ctr}+1\rceil)$.

- Even if we only send the encryption $\lfloor \mathrm{ctr}\rceil||m \oplus F_k(\lfloor \mathrm{ctr}\rceil)$, Eve might still manipulate the message by flipping randomly chosen bits.

▷ If orders have a fixed format, then Eve knows which bits to attack.

## MACs from encryption schemes?

- **Ex**: Does rOFB mode yield a secure MAC? Does the OTP?
- **Ex**: Let $F$ be a PRF with $l(n) = l_{\mathsf{in}}(n) = l_{\mathsf{out}}(n)$. Does $\mathsf{Mac}_k(m) = F_k(m^{(1)})||F_k(m^{(2)})$ yield a secure MAC (with $\mathcal{M}_n = \{0,1\}^{2l(n)}$)?
- **Ex**: Show that $F$-rCBC does not yield a secure MAC.

  Hint: By modifying $\mathrm{IV}$, Eve can modify also the first block of an intercepted message.

## MACs from encryption schemes?

- In general, even CPA-secure ES cannot be used (directly) as secure MACs.
- Message privacy and message origin authentication should be treated as different goals.
- Common to both is that we may use the same cryptographic primitive to achieve them: pseudorandom functions

  An adversary, not knowing $k$, cannot predict the value $F_k(x)$ for a "fresh" $x$ if $F$ is a PRF; $F_k(x)$ simply looks like a truly random string (random oracle) to him.

▷ Use a PRF directly as MAC.

## Secure MACs from PRFs

- **Definition**: $F$-MAC

  Let $F$ by a PRF with input length $l_{\text{in}}(n)$ and output length $l_{\text{out}}(n)$.

  Then $F$-MAC is defined by

  - $\mathcal{K}_n = \{0,1\}^n, \mathcal{M}_n = \{0,1\}^{l_{\text{in}}(n)}, \mathcal{T} = \{0,1\}^{l_{\text{out}}(n)}$.
  - Gen: on input $1^n$, output $k \overset{u}{\in} \{0,1\}^n$.
  - Mac: on input $k \in \mathcal{K}_n$ and $m \in \mathcal{M}_n$, output $F_k(m)$.
  - Vrf: on input $k \in \mathcal{K}_n, m \in \mathcal{M}_n, t \in \mathcal{T}_n$, output $1$ iff $F_k(m) = t$.

- **Theorem**: $F$-MAC is a secure MAC if $F$ is a PRF.

## Secure MACs from PRFs: Proof

| Alice&Bob | $\mathcal{D}$ | $\mathcal{A}$ |
|---|---|---|
| $b \overset{u}{\in} \{0,1\}$ | | |
| if $b=0$: $\mathcal{O} := \text{RFO}$ | | |
| if $b=1$: $\mathcal{O} := F_k$ | | |
| run $\mathcal{D}^{\mathcal{O}}(1^n)$ | | |
| | emulate $\text{Mac}_k$ using $\mathcal{O}$ | |
| | remember the queries by $\mathcal{A}$ in $\mathcal{Q}$ | |
| | run $\mathcal{A}^{\text{Mac}_k}(1^n)$ | |
| | | return $(m,t)$ |
| | return $1$ if $m \notin \mathcal{Q} \wedge \mathcal{O}(m) = t$ else $0$ | |

$$
\begin{aligned}
& \Pr\left[\text{Win}^{\text{IndPRF}}_{n,F}(\mathcal{D})\right] \\
= \;& 1/2 \cdot \underbrace{\Pr_{b=0}\left[\text{Win}^{\text{IndPRF}}_{n,F}(\mathcal{D})\right]}_{=1-\underbrace{\Pr\left[\text{Win}^{\text{FrgMAC}}_{n,\text{RFO-MAC}}(\mathcal{A})\right]}_{\leq 2^{-l_{\text{out}}(n)}}} + 1/2 \cdot \underbrace{\Pr_{b=1}\left[\text{Win}^{\text{IndPRF}}_{n,F}(\mathcal{D})\right]}_{=\Pr\left[\text{Win}^{\text{FrgMAC}}_{n,F\text{-MAC}}(\mathcal{A})\right]} \\
\geq \;& 1/2 + 1/2\left(\Pr\left[\text{Win}^{\text{FrgMAC}}_{n,F\text{-MAC}}(\mathcal{A})\right] - 2^{-l_{\text{out}}(n)}\right)
\end{aligned}
$$

1. Lecture 8 – PRFs and Modes of Operations
2. Lecture 9 – PRPs, CBC mode, MACs
3. Lecture 10 – EtM, CCA-sec/AE, practical MACs
4. Lecture 11 – Conjectured pratical PRPs/PRFs: DES, Rijndael, *Keccak

## Extending the domain of a PRF

- As seen: PRFs give us immediately secure MACs.

▷ Hence, would like to have a PRF with "arbitrary" large $l_{\mathsf{in}}(n)$

and $l_{\mathsf{out}}(n)$ just long enough to make the prob. of guessing a correct tag negligible.

- Candidates for PRFs in practice?
    - Block ciphers of fixed block length $l = l_{\mathsf{in}} = l_{\mathsf{out}} > |k|$.
    - Compression functions used within some hash functions with $l_{\mathsf{in}} > l_{\mathsf{out}} = |k|$.

      (In cryptography, a compression function is in general not invertible.)

▷ So the problem becomes:

Given $F_k$ with domain $\{0,1\}^{l_{\mathsf{in}}(n)}$ and codomain $\{0,1\}^{l_{\mathsf{out}}(n)}$

construct $F_k^{\mathsf{ext}}$ with domain "$\left(\{0,1\}^{l_{\mathsf{in}}(n)}\right)^*$" and codomain $\{0,1\}^{l_{\mathsf{out}}(n)}$.

## Extending the domain of a PRF

- Idea: output $F_k(x)$ is just as good as a truly random string vs. an adversary that does not know $k$ and has not seen $F_k(x)$ so far.

▷ $F$ only takes two inputs, so given $m = m^{(1)}||\dots||m^{(d)}$:

 ▷ Either use $F_k(x)$ to randomly permute the next block as in CBC mode:

 Set $x^{(0)} := \mathrm{IV} := 0^{l_{\mathsf{in}}(n)}$.

 For $i = 1$ to $i = d$: set $x^{(i)} := F_k(m^{(i)} \oplus x^{(i-1)})$.

 Output $F_k^{\mathsf{CBC}}(m) := x^{(d)}$.

 Suited for $F$ a block cipher with $l_{\mathsf{in}} = l_{\mathsf{out}} > |k|$.

 ▷ or use $F_k(x)$ as a new key for processing the next messag block:

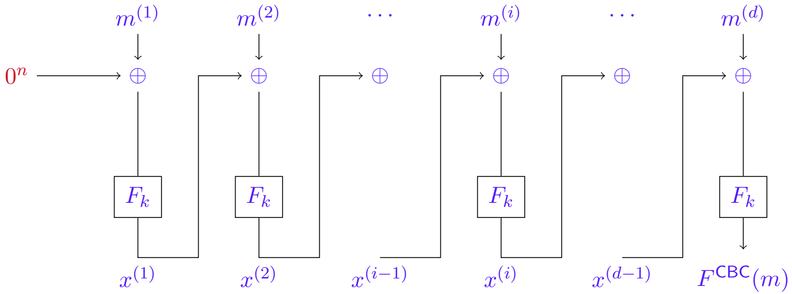
 Set $k^{(0)} := k$.

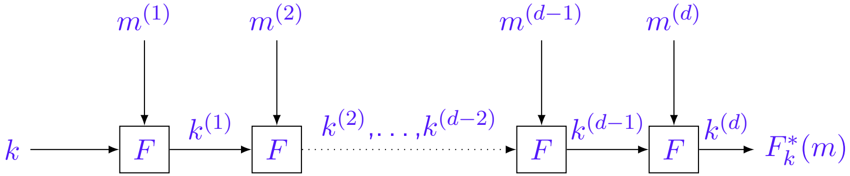
 For $i = 1$ to $i = d$: set $k^{(i)} := F_{k^{(i-1)}}(m^{(i)})$.

 Output $F_k^*(m) := k^{(d)}$.

 Suited for $F$ a compression function with $l_{\mathsf{in}} > l_{\mathsf{out}} = |k|$,

## Extending the domain of a PRF





## Extending the domain of a PRF

- $F^*$ is called Merkle-Damgård (MD) or cascaded construction.
- In the following: $F^{\mathsf{ext}}$ denotes either $F^{\mathsf{CBC}}$ or $F^*$.
- Let $q_1, \ldots, q_l$ be all queries made by $\mathcal{D}$ to $\mathcal{O}$.

  The queries are prefix-free if there are no $i \neq j$ s.t. $q_i$ is a prefix of $q_j$.
- **Theorem** [6, 21, 2]:

  If $F$ is a PRF, then $F^{\mathsf{CBC}}$ and $F^*$ are PRFs under the restriction that a PPT-adversary $\mathcal{D}$ may only use prefix-free oracle queries.

## Extending the domain of a PRF

- Problem of prefix-queries w.r.t. CBC resp. MD:

  Allows an adversary to obtain the intermediate values $x^{(i)}$ resp. $k^{(i)}$ which allows him to distinguish $F^{\mathsf{CBC}}$ resp. $F^*$ from a RFO.
- **Ex**: Let $F$ with $l_{\mathsf{in}}(n) = l_{\mathsf{out}}(n) = n$ and $\mathcal{D}^{\mathcal{O}}$ with
  - Compute $y := \mathcal{O}(0^n)$ and $z := \mathcal{O}(0^n||1^n)$
  - In case of $F^*$: return $z \stackrel{?}{=} F_y(1^n)$
  - In case of $F^{\mathsf{CBC}}$: return $z \stackrel{?}{=} \mathcal{O}((y \oplus 1^n) \oplus \mathrm{IV})$

  Determine the advantage of $\mathcal{D}$ in both cases.
- Several options exist which can be added on top of $F^{\mathsf{ext}}$ to prevent the adversary from obtaining the intermediate values.

## Extending the domain of a PRF

- Let $F$ be a "small" PRF of input length $l_{\text{in}}(n)$, and $\text{pad}\colon \{0,1\}^+ \to (\{0,1\}^n)^+$ be some injective padding function,
  - **Ex**: Why can't we simply pad by appending $0\ldots0$?
  - E.g.: $\text{pad}_{\text{CBC}}(m) = \lfloor|m|\rceil||m||0^p$, or $\text{pad}_{\text{MD-0}}(m) = m||0^p||\lfloor|m|\rceil$

    with $\lfloor|m|\rceil$ encoded using $l_{\text{in}}(n)$ bits, and $p$ minimal.
- Some options to turn $F^{\text{ext}}$ into a PRF [3, 1, 2]:
  - Length-dependent keys, e.g. $F_k^{\text{padext}}(m) := F^{\text{ext}}_{F_k(\lfloor|m|\rceil)}(\text{pad}(m))$.
  - Prefix-free padding, e.g.: $F^{\text{padext}}(m) := F^{\text{ext}}(\text{pad}_{\text{CBC}}(m))$.
  - Additional "outer encryption", e.g. $F^{\text{padext}}_{k_o,k_i}(m) := F_{k_o}(F^{\text{ext}}_{k_i}(\text{pad}(m)))$.

    In practice: pseudorandom keys, e.g. $k_o = F_k(\lfloor 1\rceil), k_i = F_k(\lfloor 2\rceil)$.
  - Truncation of the output, i.e. making only e.g. the first half of $F^{\text{ext}}(\text{pad}(m))$ public.
  - Envelop construction: $F_k^{\text{ext}}(\text{pad}(\text{pad}(m)||k))$ – key and data bits must not be mixed in a common block.

## Extending the domain of a PRF

- **Ex**: Show that $\mathsf{pad}_{\mathsf{CBC}}$ has prefix-free output.
- **Ex**: Let $F$ be a PRF. Let $F'$ be some truncation of $F$ (e.g. to only the first half). Show that $F'$ is a PRF.
- **Ex***: Let $F$ be a PRF and $G$ a PRG of stretch $l(n) = 2n$. Split the output of $G$ into half: $G(k) = G_0(k)||G_1(k)$ with $G_0(k), G_1(k) \in \{0,1\}^n$.

  Show that for any PPT-adversary $\mathcal{D}$ having access to two oracles the following prob. is negligible in $n$:

  1. $\left|\Pr_{k,k' \overset{u}{\in} \{0,1\}^n}\left[\mathcal{D}^{F_k, F_{k'}}(1^n) = 1\right] - \Pr\left[\mathcal{D}^{\mathsf{RFO},\mathsf{RFO}'}(1^n) = 1\right]\right|$

     Hint: Use a hybrid argument to decide which of the missing oracles to simulate. How might this be generalized to a polynomial number of keys?

  2. $\left|\Pr_{k,k' \overset{u}{\in} \{0,1\}^n}\left[\mathcal{D}^{F_k, F_{k'}}(1^n) = 1\right] - \Pr_{k \overset{u}{\in} \{0,1\}^n}\left[\mathcal{D}^{F_{G_0(k)}, F_{G_1(k)}}(1^n) = 1\right]\right|$
  3. $\left|\Pr\left[\mathcal{D}^{\mathsf{RO},\mathsf{RO}'}(1^n) = 1\right] - \Pr_{k \overset{u}{\in} \{0,1\}^n}\left[\mathcal{D}^{F_{G_0(k)}, F_{G_1(k)}}(1^n) = 1\right]\right|$

  where the two keys $k, k'$ are chosen independently of each other, and also $\mathsf{RFO}, \mathsf{RFO}'$ are two independent random function oracles.

## CBC-MAC

- **Definition**: CBC-MAC with prepended message length

  Let $F$ be a PRF with $l_{\text{in}}(n) = l_{\text{out}}(n)$:

  - $\mathcal{K}_n = \{0,1\}^n$, $\mathcal{M}_n = \left(\{0,1\}^{l_{\text{in}}(n)}\right)^*$, $\mathcal{T}_n = \{0,1\}^{l_{\text{out}}(n)}$
  - Gen: on input $1^n$, output $k \overset{u}{\in} \{0,1\}^n$.
  - Mac: on input $k \in \mathcal{K}_n, m \in \mathcal{M}_n$, output $F^{\text{CBC}}(\text{pad}_{\text{CBC}}(m))$

    with $\text{pad}_{\text{CBC}}(m) := |m| \,||m||0^p$.

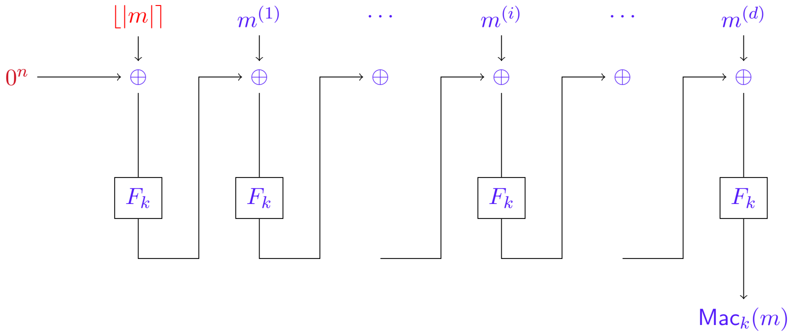
    **Reminder**: The $\mathrm{IV}$ is fixed and publicly known, in general $\mathrm{IV} = 0^{l_{\text{in}}(n)}$.
  - Vrf: on input $k \in \mathcal{K}_n, m \in \mathcal{M}_n, t \in \mathcal{T}_n$, output $1$ iff $\text{Mac}_k(m) = t$.

- **Theorem** [3, 13]:

  $F$-CBC-MAC (with $\text{pad}_{\text{CBC}}$) is a secure MAC for $F$ a PRF.

- See [3] for further discussion of how to turn $F^{\text{CBC}}$ into a secure MAC for messages of arbitrary length.

## CBC-MAC



- **Ex**: Show that $F^{\mathsf{CBC}}(\mathsf{pad}_{\mathsf{MD\text{-}0}}(m))$ does not yield a secure MAC.

## CMAC

- Uses the CBC-construction but avoids unnecessary block-cipher calls:

▷ It derives from $k$ two pseudorandom keys $k_p$ and $k_m$

▷ If the message is a multiple of the block length,

it XORs $k_m$ to the last block of the message,

and uses the CBC-construction with $k$ to obtain the output.

▷ If the message has to be padded to the block length,

it appends to the input $10\ldots0$,

then XORs $k_p$ to the last block of the message,

and uses the CBC-construction with $k$ to obtain the output.

- **Ex**: Can we choose $k_p = k_m$?

## Nested MAC (NMAC)

- **Defintion**: $F$-NMAC

  Let $F$ be a PRF with $l_{\mathsf{in}}(n) \geq l_{\mathsf{out}}(n) = n$ and $\mathsf{pad}_{\mathsf{MD}}$ either $\mathsf{pad}_{\mathsf{MD\text{-}0}} := m||0^p||\lfloor|m|\rceil$ or $\mathsf{pad}_{\mathsf{MD\text{-}1}} := m||10^p||\lfloor|m|\rceil$.

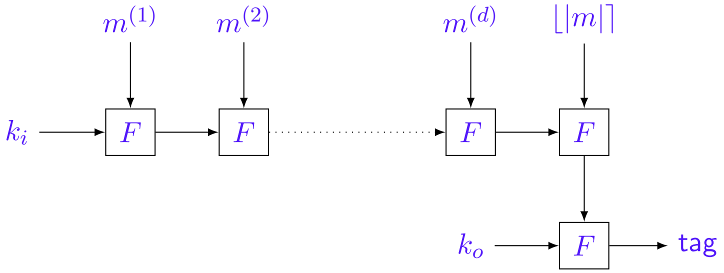
  - $\mathcal{K}_n = \{0,1\}^{2n}$, $\mathcal{M}_n = \{m \in \{0,1\}^* \mid |m| < 2^{l_{\mathsf{in}}(n)}$, $\mathcal{C}_n = \{0,1\}^n$
  - Gen: on input $1^n$, output $(k_o, k_i)$ with $k_o, k_i \overset{u}{\in} \{0,1\}^{2n}$.

    $k_o$: outer key; $k_i$: inner key.
  - Mac: on input $(k_o, k_i) \in \mathcal{K}_n, m \in \mathcal{M}_n$, output $F_{k_o}(F^*_{k_i}(\mathsf{pad}_{\mathsf{MD}}(m)))$.
  - Vrf: on input $(k_o, k_i), m \in \mathcal{M}_n, t \in \mathcal{C}_n$, output $1$ iff $t = \mathsf{Mac}_k(m)$.

- **Theorem**: ([1]) $F$-NMAC is a secure MAC for any PRF $F$.

## Nested MAC (NMAC)

- Schematic representation of $F$-NMAC.



- Keys are assumed to be the input coming from the left.
- Not shown: padding for the outer part if $l_{\text{in}}(n) > n$.
- **Ex**: Assume $F$ is a PRP with $n = l(n)$.

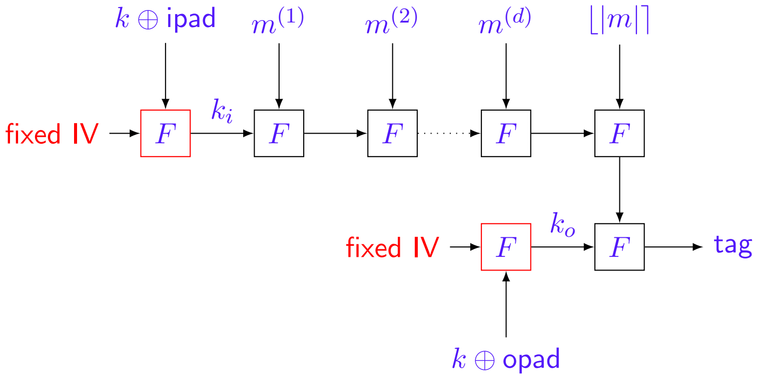
  Is it secure to replace $k_o$ by $F_{0^n}(k_o)$ and $k_i$ by $F_{0^n}(k_i)$?

## Nested MAC (NMAC)

- The inner part of $F$-NMAC consists of computing $F^*(\text{pad}_{\text{MD}}(m))$.
- Many hash functions use essentially the same construction internally.
  - Merkle-Damgard construction (later)
  - For hash functions, $F$ satisfies $l_{\text{in}}(n) > l_{\text{out}}(n) = n$ and is called a compression function.
  - E.g. SHA-1 $l_{\text{in}} = 512, l_{\text{out}} = 160$.
- In practice, NMAC is thus usually instantiated using a hash function.
  - [1]: Security of NMAC can then be understood as implicitly assuming that the compression function used within the hash function is a PRF.
  - ▷ In fact, the block ciphers SHACAL-1/2 are built from SHA-1/2 based on this assumption.
  - When working with a concrete implementation of a hash function it might not be possible to set $k_i$ as "initialization vector".
  - ▷ For this reason, a variant of NMAC, called HMAC is used in practice.

## Hash-based MAC (HMAC)

- Schematic representation of $F$-HMAC.



- Keys are assumed to be the input coming from the left.
- Not shown: padding for the outer part if $l_{\mathsf{in}}(n) > n$.

## Hash-based MAC (HMAC)

- ▷ **Definition**: Let $F$ be a PRF of block length $n$ and $F\text{-NMAC} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrf})$.

  Define opad by repeating the bit pattern $0x36$ till $n$ bits are generated. Similarly, generate ipad from $0x5c$.

  Set $\mathsf{out}(k)(k) := F(\mathsf{IV}||(k \oplus \mathsf{opad}))$ and $\mathsf{in}(k) := F(\mathsf{IV}||(k \oplus \mathsf{ipad}))$.

  Then $F\text{-HMAC} = (\mathsf{Gen}, \mathsf{Mac}', \mathsf{Vrf})$ with $\mathsf{Mac}'_k(m) := \mathsf{Mac}_{\mathsf{out}(k),\mathsf{in}(k)}(m)$.

- **Theorem**: [13, 1] For any PRF $F$, if $G(k) := \mathsf{out}(k)\mathsf{in}(k)$ is a PRG, then $F$-HMAC is a secure MAC.

- In practice: Given a hash function $H\colon \{0,1\}^* \to \{0,1\}^l$,

  set $\mathsf{Mac}_k(m) := H((k \oplus \mathsf{opad})||H((k \oplus \mathsf{ipad})||m))$.

- Alternative to HMAC: envelop construction [15]

  $\mathsf{Mac}_k(m) := F^*_{\mathrm{IV}}(k0\ldots0||m0\ldots0||k10\ldots0)$ where the number of zeros is chosen in such a way that no bits of $k$ and $m$ end up in a common block.

## Hash-based MAC (HMAC)

- Before NMAC and HMAC, several ad-hoc solutions for constructing MACs were used.

  For instance, given a (hash) function $H\colon \{0,1\}^* \to \{0,1\}^l$, the tag was defined to be $\mathsf{Mac}_k(m) := H(k||m)$, i.e. the outer encryption used in NMAC and HMAC is missing.

- **Ex**: Assume a PRF $F$ with, for simplicity, $n = l_{\mathsf{in}}(n) = l_{\mathsf{out}}(n)$.

  Using the padding function $\mathsf{pad} := m||10^p||\lfloor|m|\rceil$,

  set $\mathsf{Mac}_k(m) := H(k||m) := F_k^*(\mathsf{pad}(m))$

  for $k \in \{0,1\}^n$.

  Show that $\mathsf{Mac}_k(m)$ is not secure.

  Hint: Recall that the outer encryption used by NMAC and HMAC is to restrict the adversary to prefix-free queries.

## Birthday (collision) attacks on MACs

- Deterministic MACs built from some "small" PRF/PRP using e.g. CBC or Merkle-Damgård can be attacked using a Birthday attack:
- Generic Birthday problem: Prob. that within $q$ values $z_1, \ldots, z_q$, all of which chosen uniformly at random from a set $S$ and independently of each other, there is a pair ("collision") $z_i = z_j$ for $i \neq j$.
  - ▷ Particular instance: $q$ persons at a party, $z_i$ birthday of person $i$.
  - It can be shown: $\Pr\left[\bigvee_{i\neq j} z_i = z_j\right] \approx \frac{q^2}{|S|}$
  - We have seen that upper bound in the analysis of rCTR.
- Enumerating sufficiently many messages can be used to produce a collision in the intermediate values.

▷ Consider e.g. $F_k^{\mathsf{cbc}}$ for $k \overset{u}{\in} \{0,1\}^n$ and $F_k \colon \{0,1\}^{l(n)} \to \{0,1\}^{l(n)}$ a "secure block cipher" (PRP).

## Birthday (collision) attacks on MACs

- Choose $q$ input $x_1, x_2, \ldots, x_q$ of the form $x_i^{(0)}||x_i^{(1)}||0^{l(n)}$ for $x_i^{(0)}, x_i^{(1)} \overset{u}{\in} \{0,1\}^{l(n)}$.
- Let $y_i := F_k^{\mathsf{cbc}}(\mathsf{pad}_{\mathsf{cbc}}(x_i)) = F_k^{\mathsf{cbc}}(\lfloor 3l(n) \rceil || x_i^{(0)} || x_i^{(1)} || 0^{l(n)})$.
- Recall, the output of $F_k^{\mathsf{CBC}}$ is indistinguishable from that of a RFO – for an adversary who does not know $k$.

▷ Prob. for a output collision ($y_i = y_j$ for $i \neq j$) is $\approx q^2 \cdot 2^{-l(n)}$.

- As all inputs end on $0^{l(n)}$, if $y_i = y_j$, then we have already found a collision in the intermediate values:

$$F_k^{\mathsf{cbc}}(\lfloor 3l(n) \rceil || x_i^{(0)} || x_i^{(1)}) = F_k^{\mathsf{cbc}}(\lfloor 3l(n) \rceil || x_j^{(0)} || x_j^{(1)})$$

  (Here we use that $F_k$ is a permutation; for this reason, we also need to consider messages consisting of three blocks.)

- Thus, also $x_i^{(0)}||x_i^{(1)}||1^n$ and $x_i^{(0)}||x_i^{(1)}||1^n$ yield the same output.

▷ Possible solution: output only $l(n)/2$ bits, $2^{-l(n)/2}$ should still be negl.

## Other MAC schemes

- Lots of variants of CBC-MAC: CMAC, XCBC

  Most significant changes to “CBC-MAC + prefix-free padding”

  - From a single truly random key $k$ two or more pseudorandom keys are obtained by using $F_k$ as a PRG.
  - Depending on whether the given message is a multiple of the block length or not a different pseudorandom key is used for permuting the last message block.

- PMAC (parallelizable)
- Carter-Wegman MACs [10], e.g., UMAC (parallelizable)
  - Based on the idea of universal hashing. See the following exercise.

## Carter-Wegman MACs

- **Ex**: Let $A, B$ be finite sets and $\mathcal{H}$ some finite set of functions from $A$ to $B$. $\mathcal{H}$ is called strongly 2-universal if for every two distinct $a_1, a_2 \in A$ and any two (not necessarily distinct) $b_1, b_2$ the number of functions $f \in \mathcal{H}$ with $f(a_1) = b_1 \wedge f(a_2) = b_2$ is exactly $|\mathcal{H}|/|B|^2$.

  1. Let $p$ be a prime. $\mathbb{Z}_p$ is then a field w.r.t. addition and multiplication modulo $p$ (simply think of $\mathbb{R}$).

     For $c, d \in \mathbb{Z}_p$, let $l_{c,d}(x) := (c \cdot x + d) \bmod p$ be the line with slope $c$ and y-intercept $d$. Set $A = B = \mathbb{Z}_p$ and $\mathcal{H} = \{l_{c,d} \mid c, d \in \mathbb{Z}_p\}$.

     Show that $\mathcal{H}$ is strongly 2-universal.

  2. Consider the following experiment:

     - Alice&Bob choose a prime $p$ and then $c, d \overset{u}{\in} \mathbb{Z}_p$.
       They pass $p$ directly to Eve and grant her a single oracle query $m$ to $l_{c,d}$.
     - Eve returns at some point $m'$ and $y'$.

     ▷ Eve wins iff $m \neq m'$ and $l_{c,d}(m') = y'$.

     What is Eve's probability to win?

## Combining ES and MAC

- Possible ways of combining an ES with a MAC:

  Enc-and-Mac (E&M): $\mathsf{Enc}_k(m)||\mathsf{Mac}_k(m)$

  Mac-then-Enc (MtE): $\mathsf{Enc}_k(m||\mathsf{Mac}_k(m))$

  Enc-then-Mac (EtM): $c||\mathsf{Mac}_k(c)$ with $c \overset{r}{:=} \mathsf{Enc}_k(m)$

▷ Except for the Enc-then-Mac approach (see also [5]), one can construct (sometimes contrived) counter-examples so that the resulting ES is not CCA-secure ES.

▷ Still for specific combinations also EaM and MtE can be used:

- **Example**:
  - SSH binary packet protocol uses Enc-and-Mac.
  - TLS uses Mac-then-Enc.

▷ But in general EtM is the least error-prone way.

## Encrypt then authenticate

- **Defintion**: Enc-then-Mac (EtM)

  Let $\mathcal{E} = (\mathsf{Gen}^{\mathcal{E}}, \mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure ES, and $\mathcal{S} = (\mathsf{Gen}^{\mathcal{S}}, \mathsf{Mac}, \mathsf{Vrf})$ a secure deterministic MAC with $\mathcal{M}_n^{\mathcal{E}} \subseteq \mathcal{M}_n^{\mathcal{S}}$.

  Define $\mathcal{ES} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

  - $\mathcal{K}_n' = \mathcal{K}_n^{\mathcal{E}} \times \mathcal{K}_n^{\mathcal{S}}$, $\mathcal{M}_n' = \mathcal{M}_n^{\mathcal{E}}$, $\mathcal{C}_n' = \mathcal{C}_n^{\mathcal{E}} \times \mathcal{T}_n^{\mathcal{S}}$.
  - $\mathsf{Gen}'$: on input $1^n$, compute $k^{\mathcal{E}} :\overset{r}{=} \mathsf{Gen}_{\mathcal{E}}(1^n)$ and $k^{\mathcal{S}} :\overset{r}{=} \mathsf{Gen}_{\mathcal{S}}(1^n)$, output $k = (k^{\mathcal{E}}, k^{\mathcal{S}})$.
  - $\mathsf{Enc}'$: on input $m$ and $(k^{\mathcal{E}}, k^{\mathcal{S}})$, compute $c :\overset{r}{=} \mathsf{Enc}_{k^{\mathcal{E}}}(m)$, then $t :\overset{r}{=} \mathsf{Mac}_{k^{\mathcal{S}}}(c)$, output $c||t$.
  - $\mathsf{Dec}'$: on input $c||t$ and $(k^{\mathcal{E}}, k^{\mathcal{S}})$, check if $\mathsf{Vrf}_{k^{\mathcal{S}}}(c, t) \overset{?}{=} 1$; if yes, output $\mathsf{Dec}_{k_E}(c)$; otherwise output an exception $\perp$.

- **Theorem**: (see [13], p. 144)

  $\mathcal{ES}$ is a CCA-secure ES if $\mathcal{E}$ is a CPA-secure ES and $\mathcal{S}$ is a secure MAC.

- Proof idea:
- ▷ We may assume that an adversary $\mathcal{A}$ queries its decryption oracle only for ciphertexts which $\mathcal{A}$ has not obtained from its encryption oracle.
- Event $F$: "$\mathcal{A}$ succeeds at least once at forging a ciphertext."

  $$\Pr[F] \leq q_{\mathsf{Dec}_k}(n)\Pr\left[\mathsf{Win}_{n,\mathcal{S}}^{\mathsf{FrgMAC}}(\mathcal{B})\right] \leq \mathsf{negl}_{\mathcal{S}}(n)$$

  where $\mathcal{B}$ simply selects a random $\mathsf{Dec}_k$-query of $\mathcal{A}$.
- Event $\neg F$: "$\mathcal{A}$ does not succeed at forging a single ciphertext."

  The decryption oracle then always detects the forged ciphertexts, and, thus, can be simulated in game $\textsc{IndCPA}$ by always replying $\perp$.

  $$\Pr\left[\mathsf{Win}_{n,\mathcal{ES}}^{\textsc{IndCCA}}(\mathcal{A}) \mid \neg F\right] \leq \Pr\left[\mathsf{Win}_{n,\mathcal{E}}^{\textsc{IndCPA}}(\mathcal{A})\right] \leq \frac{1}{2} + \mathsf{negl}_{\mathcal{E}}(n)$$

## Remark: Restricted forms of decryption oracles

- In EtM, an error message ($\perp$) is returned if verification of the mac tag fails.
- It is indeed important that this error message does not leak any information.

▷ Already telling the attacker if a message was correctly padded might be problematic:

  - The TLS (up to 1.2) allows i.a. to use Mac-then-Enc (MtE).
  - The way TLS 1.1/1.2 uses MtE can be shown to be secure.
  - But TLS prior to 1.1 returned two different error message (mac invalid vs. padding invalid) which could be used to break TLS.

  ▷ Thus, starting with TLS 1.1 the same error message is always returned.

  (Sill, in case of an invalid padding, TLS 1.1+ leaks some information via the time needed to process a dummy mac tag [Lucky 13 attack].)

## Remark: Other ways of achieving CCA-security

- **Ex***: Let $P$ be a strong PRP of block length $2n$ and key length $n$.

  Let $\mathsf{Enc}_k(m) = P_k(\rho||m)$ for $m \in \{0,1\}^n$ and $\rho \overset{u}{\in} \{0,1\}^n$.

  - Define the decryption.
  - Show that this ES is CCA-secure.

- AEAD modes

  - AEAD: authenticated encryption with associated data
  - Authenticated encryption (AE): CCA-secure + secure message origin authentication
  - Associated data: public data whose origin only needs to be authenticated (like headers in a protocoll)

  ▷ EtMac achieves AE, but associated data is also encrypted.

## Authenticated encryption

- Several specific constructions for AEAD exist:

  OCB, GCM (TLS 1.2), CCM (TLS 1.2), CWC, Keccak in duplex mode

▷ Often specified only for a specific block length like $128$bits (e.g. OCB,GCM,CCM).

▷ But more efficient like e.g. rCTR-then-CBC-MAC, as they allow for parallelization, require less block cipher calls e.g. by only authenticating associated data, etc.

## Authenticated encryptions

- Enc-Then-Mac:

  CPA-secure ES + secure MAC yields CCA-secure ES with authenticated encryptions (AE)
- Disadvantage of Enc-Then-Mac: The data has to be processed twice.

## Tweakable block ciphers

- Recall: CBC mode $t^{(i)} = F_k(m^{(i)} \oplus t^{(i-1)})$

▷ Alternative view:

CBC transforms $F$ into $F_k[t](m) := F_k(m \oplus t)$ where $t$ is called the tweak.

- If $F$ is a PRP (secure block cipher),

then "$F$-rCBC-then-$F$-CBC-MAC" is CCA-secure.

▷ Ideally, we would like to reuse the similarities in $F$-rCBC and $F$-CBC-MAC to speed up the whole computation, but:

 ▷ $F$-rCBC: the IV has to be random, all tweaks $t^{(i)}$ have to be made public.

 ▷ $F$-CBC-MAC: the IV has to be fixed, the intermediate tweaks $t^{(i)}$ have to be kept secret.

 ▷ We first need to compute $\mathsf{Enc}_{k_{\mathsf{Enc}}}(m)$, only then we may compute $\mathsf{Mac}_{k_{\mathsf{Mac}}}(\mathsf{Enc}_{k_{\mathsf{Enc}}}(m))$.

## Tweakable block ciphers

- **Definition**: A tweakable block cipher (TBC) $F$ of block-length $l$ defines for any secret $n$-bit key $k$ and any public tweak $t \in T$ a permutation $F_k[t]$ on $\{0,1\}^l$.

▷ In order to unify and simplifiy the treatment of ESs and MACs, security should only depend on the secrecy of $k$.

That is, for any two distinct tweaks $t, t'$, an efficient adversary should not be able to distinguish the permutations $F_k[t]$ and $F_k[t']$ (where $k \overset{u}{\in} \{0,1\}^l$ is secret) from independently chosen random permutations.

## Tweakable block ciphers

- **Definition**: Let $F$ be a TBC with tweak space $T$.

  ❶ Alice&Bob set up two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$:

| $\mathcal{O}_0$: "tweaked random permutation oracle TRPO" | $\mathcal{O}_1$: replies using $F$ |
|---|---|
| Init: create an empty hashmap $P\colon T \to$ RPO | Init: create $k \in \{0,1\}^n$ |
| Query: on input $(t, x)$ | Query: on input $(t, x)$ |
| if $P[t]$ is undefined: $P[t] :=$ new RPO | |
| return $P[t] \to$ RPO$(x)$ | return $F_k[t](x)$. |

  They toss a fair coin $b \overset{u}{\in} \{0,1\}$ and pass $\mathcal{O}_b$ in a black box $\mathcal{O}$ to Eve.

  ❷ Eve runs $\mathcal{D}^{\mathcal{O}}(1^n)$ to obtain a reply $r$.

  ▷ $\mathcal{D}$ wins iff $r = b$.

  $F$ is a secure TBC if any efficient distinguisher $\mathcal{D}$ has only negligible advantage in above game (efficient: either PPT or $(t, q, \varepsilon)$ formalism).

▷ **Example**: Set $F[t](x) := F_k(x \oplus t)$.

  Consider then $\mathcal{D}$ which replies $1$ iff $\mathcal{O}(0^n, 0^n) = \mathcal{O}(1^n, 1^n)$.

## Tweakable block ciphers

- **Remark**: As "secure TBC" generalizes PRP, "strong TBC" generalizes strong PRP, i.e. in case of a strong TBC, the oracle also computes the inverse $F_k[t]^{-1}(x)$ for any $(t, x)$ chosen by $\mathcal{D}$.
- The notion of TBCs and their security was first introduced by Liskov, Rivest, and Wagner in [16] who observed that many existing construction had already been using this idea implicitly.
- **Theorem** [16]: If $F$ is a $(t, q, \varepsilon)$-PRP,

  then $F_k(t \oplus F_k(m))$ is a $(t, q, \varepsilon + \Theta(q^2/2^l))$-secure TBC.
  - [16] also shows how to turn a strong $(t, q, \varepsilon)$-PRP into a strong TBC.
- Disadvantage: double encryption of every message block.

## Tweakable block ciphers

- **Ex****: Is $F_k[t](m) := F_{F_k(t)}(m)$ a secure TBC if $F$ is a PRP?
- **Ex**: Let $F$ be a PRF of key and block lenght $n$. Show that none of following is a secure MAC for messages of fixed length $2n$:
  - $\mathsf{Mac}_k(m) := F_k(m^{(1)} \oplus m^{(2)})$
  - $\mathsf{Mac}_k(m) := F_k(m^{(1)}) \oplus F_k(m^{(2)})$
- **Ex***: Let $F$ be a secure TBC. Show that the following is NOT a secure MAC for messages of fixed length $2n$:
  $\mathsf{Mac}_k(m) := F_k[\lfloor 1 \rceil](m^{(1)}) \oplus F_k[\lfloor 2 \rceil](m^{(2)})$.
  Could we fix it by using a static counter variable $c$ and process the $c$-th block by $F_k[c](m^{(c)})$?
- **Ex***: Let $F$ be a strongly secure TBC with "tweak space", key space, and domain $\{0,1\}^n$. Which of the following yields a CPA-secure ES?
  - $\mathsf{Enc}_k(m^{(1)}||\dots||m^{(s)}) :=$
    $\rho||m^{(1)} \oplus F_k[\rho+1](0^n)||\dots||m^{(s)} \oplus F_k[\rho+s](0^n)$ with $\rho \overset{u}{\in} \{0,1\}^n$.
  - $\mathsf{Enc}_k(m^{(1)}||\dots||m^{(s)}) := \rho||F_k[\rho+1](m^{(1)})||\dots||F_k[\rho+s](m^{(s)})$ with $\rho \overset{u}{\in} \{0,1\}^n$.

## Tweakable block ciphers

- A more practical construction of a strong TBC is the XEX-construction (see [22]) used e.g. by TrueCrypt and OCB/PMAC:

▷ The data $m$ is uniquely partitioned into blocks of data $m_{N,i}$:

$N$: data unit index; $i$: block index within the data unit.

▷ For $F$ a block cipher $F$ of block length $l$ set:

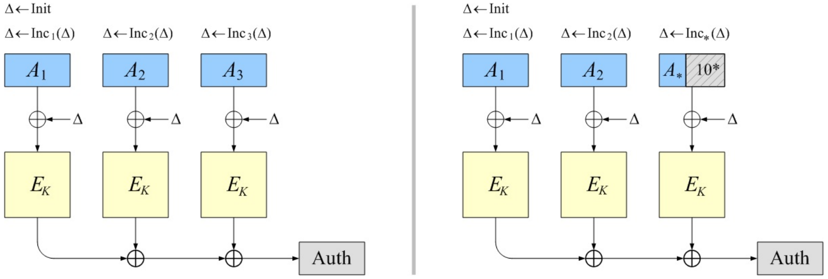
$$F_k[N,i](x) := \Delta \oplus F_k(\Delta \oplus x) \text{ where } \Delta = F_k(N) \cdot 2^i.$$

▷ The product $F_k(N) \cdot 2^i$ is calculated in GF$(2^l)$ assuming $2 \hat{=} X$ a primitive root (see appendix).

▷ If $F_k[N,i]$ is a secure TBC, then each block is essentially processed by its own RPO which is independent of all other blocks.

- Note that $\Delta$ is independent of $m$ and can thus be precomputed for each tweak $(N,i)$.

## TBC example: OCB-mode



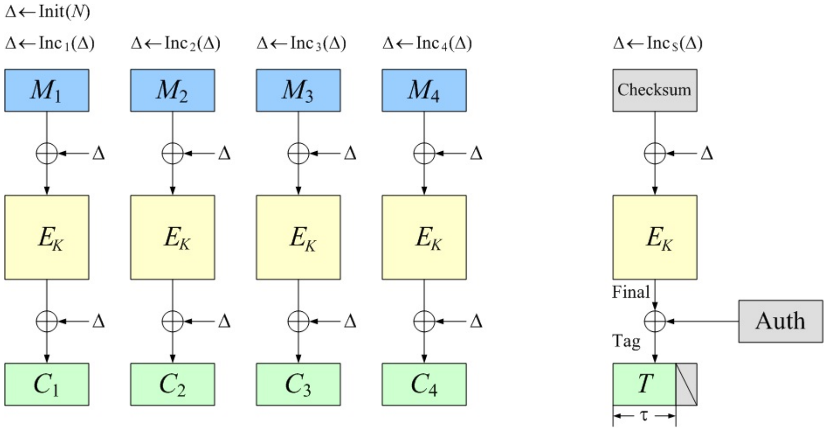
Taken from http://www.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm

- Check above link for the most current revision.
- Designed for block length $128$bit.
- Allows to also authenticate "associated data $A$" along the actual message $m$.

▷ If $A = \varepsilon$, then $\text{Auth} = 0^{128}$.

- Left (right): length of $A$ is (not) a multiple of block length.

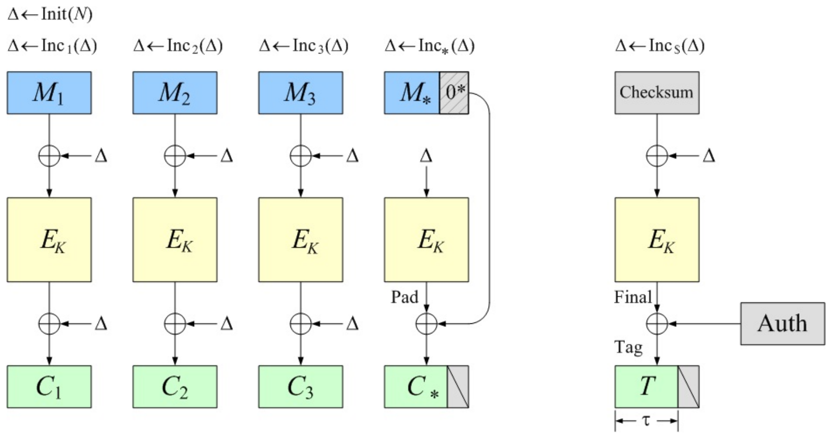
## TBC example: OCB-mode



Taken from http://www.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm

- OCB requires a nonce $N$, e.g. a global message counter.
- Encryption if $m$ is a multiple of the block length.
- Checksum $:= M_1 \oplus M_2 \oplus M_3 \oplus (M_4\,\mathsf{Pad}[|M_4|\,, l])$ for block length $l$.

## TBC example: OCB-mode



Taken from http://www.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm

- Encryption if $m$ is not a multiple of the block length.

## Data encryption standard (DES)

- DES $=$ data encryption standard
- In 1973, NBS (now NIST) on behalve of the NSA solicited proposals for a cipher based on Kerckhoff's principle that the security was solely based on the secrecy of the key.
- All initial submissions were deemed unsuitable which led to a second round in 1974.
- This time IBM submitted a design based on Feistel's Lucifer cipher.
- Lucifer eventually turned into what we now call DES.

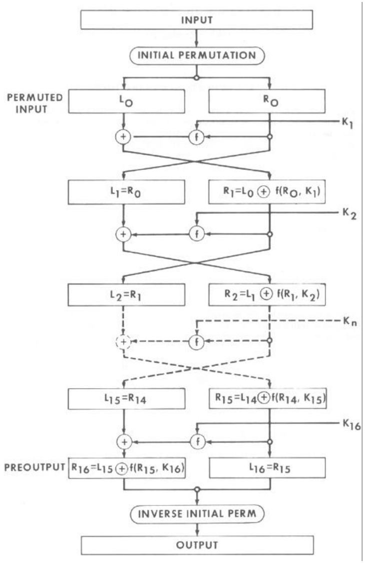
## Data encryption standard (DES)

- Depending on who you ask:
  - “We developed the DES algorithm entirely within IBM using IBMers. The NSA did not dictate a single wire!”
  - “NSA worked closely with IBM to strengthen the algorithm against all except brute force attacks and to strengthen substitution tables, called S-boxes. Conversely, NSA tried to convince IBM to reduce the length of the key from 64 to 48 bits. Ultimately they compromised on a 56-bit key.”
- See the links on wikipedia Engl. and Ger..
- But nobody seems to believe that the NSA has introduced any “backdoor” into DES.

## DES: Feistel network

- DES is basically $16$-round Feistel network $\mathsf{FN}_{f^{(1)},\dots,f^{(16)}}$ with input/output length $64$bits except for the following:
    1. An initial permutation IP is applied to the input before the FN.
    2. The two halves of the output of the FN are swapped afterwards.
    3. Finally, $\mathsf{IP}^{-1}$ is applied.
- For each round a 48bit "round key" $k_i$ is generated deterministically from $k$ by duplicating bits of $k$.
    - Think of it as a PRG which stretches $k$ to a $16 \cdot 48$-bit string.
    - We won't discuss the details of the generation of $k_i$.
- In every round the DES-mangler function $\hat{f}$ instantiated with $k_i$ is applied.
    - Recall: A PRF can be turned into a PRP using a Feistel network. $\hat{f}$ therefore is/was a candidate for a PRF.
- See the NIST specification [20] for implementation details.

## DES: Feistel network



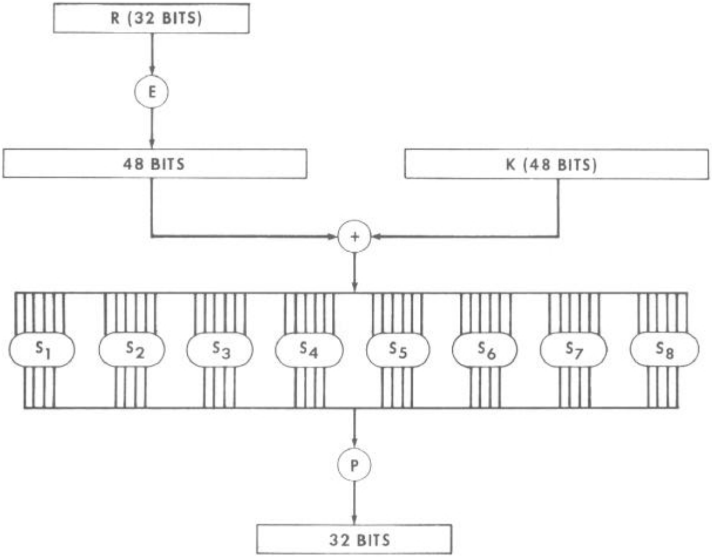
Taken from [20]

## DES: Mangler function

- Input:
  - $R$ the "right half" of the current Feistel round ($32$bits),
  - $K$ the current round key ($48$bits).
- Steps:
  1. Expansion of $R$ to $48$bits: denoted by $E(R)$; $E(\cdot)$ simply duplicates half of the bits of $R$.
  2. Compute $T := E(R) \oplus K$.
  3. Partition $T$ into $8$ blocks of $6$ consecutive bits,

     i.e., $T = T^{(1)} \dots T^{(8)}$ with $\left|T^{(i)}\right| = 6$.
  4. Feed $T^{(i)}$ to the $i$-th S-boxes $U^{(i)} := S_i(T^{(i)})$ with $\left|U^{(i)}\right| = 4$.
  5. Apply a final a permutation $P$ to $U = U^{(1)} \dots U^{(8)}$.

## DES: Mangler function



Taken from [20]

## DES: Mangler function

- From a theoretical point of view, the mangler funtion should be a candiate for a PRF.

▷ I.e. it should be indistinguishable from a RFO within some concrete security bound (w.r.t. to hardware from the 1970s).

- Recall: If we ask a RFO $\mathcal{O}$ for the value of two fresh values $x$ and $x'$, then $y := \mathcal{O}(x)$ and $y' := \mathcal{O}(x')$ will be chosen uniformly at random from all possible outputs independently of each other.

▷ That is, even if $x$ and $x'$ differ only in a single bit, $y$ and $y'$ will differ in $n/2$ bits on the average.

▷ So, also any candidate for a PRF, like the mangler function, should amplify a single bit difference in the inputs to a approx. $n/2$ difference in the outputs.

▷ Design principle: Confusion-diffusion paradigm

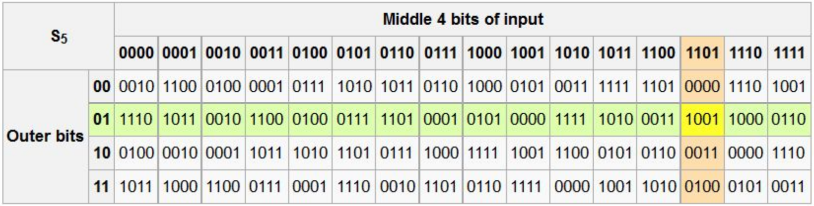
## Confusion-diffusion paradigm

- Shannon [23]:
  - Confusion: "The method of confusion is to make the relation between the simple statistics of [the intermediate ciphertext] $c$ and the simple description of $k$ a very complex and involved one."
  - ▷ achieved by functions called S(ubstitution)-boxes which only act on a very small part of the actual input; specifically designed to resist all known attacks, pass all statistical tests when compared to a randomly chosen function.
  - Diffusion: "In the method of diffusion the statistical structure of $m$ which leads to its redundancy is dissipated into long range statistics[...]."
  - ▷ achieved by a specifically designed permutation over the whole data; combines the output of the S-boxes so that it again can pass at a randomly chosen function, e.g. moves differences from one S-box to another.
- Their combination is called Substitution-permutation network.

## DES: Substitution-boxes (S-boxes)

- Take 6bit as input, output 4bits.
- The security of the DES crucially depends on the choice of the Substitution-boxes (S-boxes).
- It was shown that small changes to them or the random choice of them yields a less secure block cipher.
- Designed to make differential cryptanalysis useless (later).
- S-boxes are usually implemented via look-up $2^2 \times 2^4$ tables:
  - The first and the last input bit determine the row,
  - while the inner bits (2–5) determine the column.
  - Every cell contains a $4$bit output.

## DES: Substitution-boxes (S-boxes)



| $S_5$ | | Middle 4 bits of input | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0000** | **0001** | **0010** | **0011** | **0100** | **0101** | **0110** | **0111** | **1000** | **1001** | **1010** | **1011** | **1100** | **1101** | **1110** | **1111** |
| **Outer bits** | **00** | 0010 | 1100 | 0100 | 0001 | 0111 | 1010 | 1011 | 0110 | 1000 | 0101 | 0011 | 1111 | 1101 | 0000 | 1110 | 1001 |
| | **01** | 1110 | 1011 | 0010 | 1100 | 0100 | 0111 | 1101 | 0001 | 0101 | 0000 | 1111 | 1010 | 0011 | 1001 | 1000 | 0110 |
| | **10** | 0100 | 0010 | 0001 | 1011 | 1010 | 1101 | 0111 | 1000 | 1111 | 1001 | 1100 | 0101 | 0110 | 0011 | 0000 | 1110 |
| | **11** | 1011 | 1000 | 1100 | 0111 | 0001 | 1110 | 0010 | 1101 | 0110 | 1111 | 0000 | 1001 | 1010 | 0100 | 0101 | 0011 |

taken from here

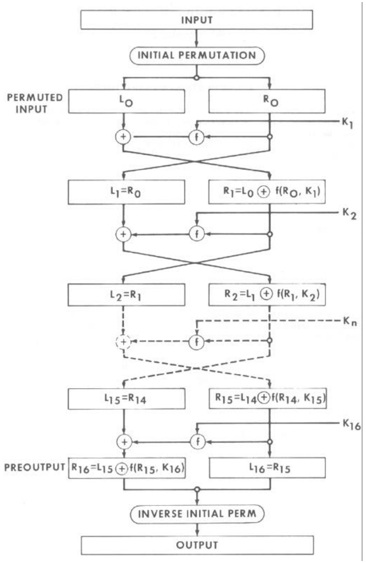
- S-boxes are defined in such a way that:
  1. each row in the table is a permutation of $\{0,1\}^4$.
     - ▷ hence, exactly four inputs are mapped on the same output ($2^6/2^4 = 4$).
  2. Changing one input bit always changes at least two output bits. (Confusion by substitution)

▷ as one would expect of a randomly chosen function/RFO.

## DES: Permutation and Avalanche effect

- ▷ The permutation $P$ spreads the differences introduced by the S-boxes into at least two other 6-bit blocks processed by different S-boxes in the next round. (Diffusion by permutation)
- ▷ Subsequent rounds guarantee that small changes to the original plaintext influence all output bits eventually. (DES avalanche effect):
- $(L_i, R_i)$: left/right halves fed to the Feistel network in the $i$-th round.
- $(L'_i, R'_i)$: for comparison second computation using the same key.
- ▷ Initially $L_0 = L'_0$ but $R_0$ and $R'_0$ differ in exactly one bit.

## DES: Feistel network



Taken from [20]

## DES: Permutation and Avalanche effect

- First round: mangler function is applied to $R_0$, resp. $R_0'$.
  - "Worst case": still 1-bit difference after exension function.
  - Hence: $T := E(R_0) \oplus K$ and $T' := E(R_0') \oplus K$ only differ in 1 bit only.
  - W.l.o.g. $T$ resp. $T'$ differ in the first block.
  - By its design, $S_1(T)$ and $S_1(T')$ differ in (at least) two bits.
  - $P$ is designed s.t. these two differences in the first block are spread e.g. into the second and the third 4-bit block.
  - After the first round, $(L_1, R_1)$ and $(L_1', R_1')$ differ in at least three bits as $L_1 := R_0$ and $L_1' := R_0'$.
- A single bit difference is expected to influence all $64$bits in the worst case after roughly eight rounds. [13]
- As in the case of the S-boxes, it was shown that if $P$ is replaced by a randomly chosen permutation, then (with high prob.) the avalance effect is much weaker.

## DES: Security

- Attacks on reduced-round versions:
  - When less than $16$ rounds of the FN are used, attacks exist which not only obtain the plaintext, but also the key.
  - See [13] for attacks on 1/2/3-rounds.
  - There exists an attack on 9-round DES using $2^{15.8}$ plain-/ciphertext pairs (same key) which only needs to consider $2^{29.2}$ keys.
- Today the key space of DES is considered too small:
  - Already in 1998 specialized hardware (cost USD250.000) solved the DES Challenge II-2 in 56h.
  - Using cheaper reconfigurable hardware enumerating all keys took 34h/68h in 2009.
- Also the short block length can be a problem:
  - E.g., recall rCTR and the problem of a collision in the counter value.
  - Probability for a collision within $q$ encryptions: $\Theta(q^2/2^l)$

  ▷ The "effective" block length is halved.

## DES: Security

- **Ex**: As both the expansion and the round-key generation only duplicate input bits, one can show that

  $$\mathsf{DES}_{\overline{k}}(\overline{m}) = \overline{\mathsf{DES}_k(m)}$$

  reducing the key space to $2^{55}$ under a chosen-plaintext attack.
- In general, given two keys $k_1, k_2$ one cannot find a third key $k_3$

  s.t. $\mathsf{DES}_{k_1}(\mathsf{DES}_{k_2}(\cdot)) = \mathsf{DES}_{k_3}(\cdot)$.
- For DES, the following keys are considered "weak":
  - Keys $k$ which produce sixteen identical round keys s.t. $\mathsf{DES}_k(\mathsf{DES}_k(m)) = m$.
  - Keys which produce only two distinct round keys each of the two used eight times.

    These keys form pairs $(k_1, k_2)$ s.t. $\mathsf{DES}_{k_1}(\mathsf{DES}_{k_2}(m)) = m.$
  - See wikipedia for a list of all (semi-)weak keys.

## DES: Differential cryptanalysis

- Attributed to Biham and Shamir [7], although claimed to be known to the DES designers and the NSA already in the 1970's.
- Basic idea for block cipher $F$:
  1. Choose $\delta_i, \delta_o \in \{0,1\}^n$ (with $n$ the output length).
  2. Define $p(\delta_i, \delta_o)$ by $\Pr_{m,k}[F_k(m \oplus \delta_i) = F_k(m) \oplus \delta_o]$.

  ▷ I.e., $p(\delta_i, \delta_o)$ is the prob. that a random plaintext pair of difference $\delta_i$ yields a ciphertext pair of difference $\delta_o$ when encrypted using a random key.
- Usually, $p(\delta_i, \delta_o)$ is not precomputed using brute-force; instead, e.g., the S-boxes are analyzed.
- For badly designed S-boxes, some difference pairs appear with higher prob. than other pairs.
- Using a sufficient number of chosen plaintexts, these small differences can be sufficiently amplified in order to break a cipher, e.g. FEAL.

## DES: Differential cryptanalysis

- In the case of DES, $2^{47}$ chosen-plaintexts are needed, which makes the attack less practical than brute-force key search.
- See [7] for more details:
  - If one chooses all round keys at random, i.e., one uses a random key of $16 \cdot 48$bits, differential cryptanalysis allows to break DES using "only" $2^{61}$ chosen-plaintext pairs, i.e., a significantly larger key space of $768$bit keys only yields a small increase in security compared to the $56$bit keys used by DES w.r.t. differential cryptanalysis.
  - The S-boxes can be reordered in such a way that DES can be broken within $2^{46}$ chosen-plaintext pairs.
  - "Corollary": Never assume that a "simple" change to a good cryptographic algorithm/protocol yields again a good one.

## DES: Linear cryptanalysis

- Attributed to Matsui, published in 1992 [17]
- Also used to attack the FEAL cipher.
- Basic idea for block cipher $F$:
  - Study “linear” relations between particular input, output, and key bits.
  - I.e., choose subsets $I_i$, $I_o$, $I_k$ of input/output/key bits and compute the “bias” $p(I_i, I_o, I_k)$:

$$\Pr_{m,k,c:=F_k(m)}\left[\bigoplus_{i\in I_i} m_i \oplus \bigoplus_{I_o} c_i \oplus \bigoplus_{i\in I_k} k_i = 0\right]$$

- Linear cryptanalysis tries to approximate the nonlinear behavior of S-boxes using linear functions.

## DES: Linear cryptanalysis

- Matsui shows how to obtain the key from a sufficient set of known plain-/ciphertext pairs encrypted using the same key.
  - Differential cryptanalysis requires the attacker to be able to choose the plaintexts.
- In case of DES, the attack still requires $2^{43}$ known pairs.
- See [17] for more details.

## DES: Multiple encryptions

- Because of the absence of any practical attacks against DES (better than exhaustive search), the short key length is considered the only real shortcoming of DES.
  - At least when its block length does not matter.
  - ▷ Recall that for MAC schemes and some modes of operations the block length determines the feasibility of a birthday attack.
- 3DES is an attempt to fix this using three keys $k_1, k_2, k_3$ to define the block ciper:

$$\mathsf{3DES}_{k_1,k_2,k_3}(x) := \mathsf{DES}_{k_3}(\mathsf{DES}_{k_2}^{-1}(\mathsf{DES}_{k_1}(x)))$$

## DES: Multiple encryptions

- Keying options:

❶ $k_1, k_2, k_3$ are chosen at random and independently of each other.

This option is susceptible to a "a meet-in-the-middle attack" reducing the effictive keylength to $2^{2n}$ at the cost of a constant number of known[1] plain-/ciphertext pairs and $\mathcal{O}(2^{2n})$ memory:

▷ Given a plain-/ciphertext-pair $(m, c)$, (i) precompute $r(k_1, k_2) := \mathsf{DES}^{-1}_{k_2}(\mathsf{DES}_{k_1}(m_i))$ , (ii) precompute $l(k_3) := \mathsf{DES}^{-1}_{k_3}(c_i)$, (iii) search for $r(k_1, k_2) = l(k_3)$.

❷ $k_1, k_2$ are chosen at random and independently of each other with $k_3 := k_1$.

Currently, only an (inefficient) chosen-plaintext attack using $2^n$ pairs and reducing the key space to $2^n$ is known. (See here.)

❸ $k_1$ is chosen at random with $k_2, k_3 := k_1$.

This option is used for backward compatability and explains the inner decryption step as $\mathsf{3DES}_{k,k,k}(x) = \mathsf{DES}_k(x)$.

[1] I.e., Alice and Bob choose the pairs, not Eve.

## DES: Multiple encryptions

- 3DES is still considered a very strong block cipher and widely used today.
- The main drawbacks are the short block size (just as DES) and that it is quite slow.
    - Faster and still conjectured secure w.r.t. the key space [14]:

      $\mathsf{DESX}_{k_o,k_i,k}(m) := k_o \oplus \mathsf{DES}_k(m \oplus k_i)$ with $k_i, k_o \in \{0,1\}^{64}$.

    ▷ Still, this does not fix the block length.
- Main motivation for AES: fast block cipher with larger block length.

## AES: advanced encryption standard

- Based on the Rijndael cipher by Joan Daemen and Vincent Rijmen.
- Winner of a three round contest held by the NIST (1997-2000).
- All submissions were open to the public, checked by competing proposers, and the winner was selected from five finalists:

  Rijndael (Daemen, Rijmen), Serpent (Anderson, Biham, Knudsen), Twofish (Schneier, Ferguson, et al.), RC6 (Rivest, Robshaw, et al.), MARS (IBM/Coppersmith)

## AES: advanced encryption standard

- Requirement by the NIST (see here)

  "Algorithms will be judged on the following factors:

  - Actual security of the algorithm compared to other submitted algorithms (at the same key and block size).
  - The extent to which the algorithm output is indistinguishable from a random permutation on the input block.
  - Soundness of the mathematical basis for the algorithm's security.
  - Other security factors raised by the public during the evaluation process, including any attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter."

## AES/Rijndael: Outline

- Rijndael is a substitution-permutation network.
  - Recall the DES-mangler function is a single-round s/p network.
- The specification of Rijndael [11] allows to set the block and key length independently of each other to $128/192/256$ bits.
  - AES only differs from Rijndael by fixing the block length to $128$ bits. (See also the specification of AES [19].)
  - The number of rounds the s/p-network is repeated depends on the chosen key and block lengths and is defined in the specification.
  - The number of rounds were chosen based on the analysis of known attacks like differential/linear cryptanalysis.
  - In case of the AES combinations, the number of rounds are $10/12/14$.

    (There is some criticism regarding the number of rounds for key lengths $192/256$.)
  - The specification also discusses possible extensions to arbitrary block and key lengths (see section 12.1 of [11]).

## AES/Rijndael: Outline

- The basic structure of Rijndael is a substitution-permutation network similar to the DES mangler function:

▷ The intermediate ciphertext is again obtained by XOR'ing a round key to the given input. (AddRoundKey)

  - The round keys are obtained from the secret key by essentially using the S-box (below) as a PRG, i.e., w/o a secret key.
  - See the specification for further details on AddRoundKey.

▷ A single 8bit S-box is then used for achieving confusion. (SubBytes)

▷ The permutation for achieving diffusion is split into two permutations. (ShiftRows,MixColumns)

## AES/Rijndael: Outline

- The complete s/p network for $\mathrm{N}$ rounds[2]:

$$(\mathrm{AddRoundKey} \rightarrow \mathrm{SubBytes} \rightarrow \mathsf{ShiftRow} \rightarrow \mathsf{MixColumn})^{\mathrm{N}-1}$$
$$\rightarrow \mathrm{AddRoundKey} \rightarrow \mathrm{SubBytes} \rightarrow \mathsf{ShiftRow} \rightarrow \mathsf{AddRoundKey}.$$

▷ The final XOR'ing of a round key is required in order to prevent an adversary from easily reverting $\mathrm{SubBytes} \rightarrow \mathsf{ShiftRow}$.

[2]In the AES/Rijndael specification a "round" consists of $\mathrm{SubBytes} \rightarrow \mathsf{ShiftRow} \rightarrow \mathsf{MixColumn} \rightarrow \mathrm{AddRoundKey}$.

## AES/Rijndael: SubBytes

- In contrast to DES, AES does not use a Feistel network:

  In AES, the single S-box SubBytes is a permutation over $\{0,1\}^8$.
- The S-box interprets a 8-bit strings
  - once as an element of the praticular representation of the field $\mathsf{GF}(2^8)$,
  - once as a bit-vector of dimension 8.

  which leads to a very short algebraic definition:
  - Input: $x \in \{0,1\}^8$
  - Treat $x$ as an element of $\mathsf{GF}(2^8)$ and compute $y := x^{-1}$

    where $\mathsf{GF}(2^8)$ is defined w.r.t. the polynomial $1 + X + X^3 + X^4 + X^8$ (see next slide).
  - Treat $y \in \{0,1\}^8$ as vector of dimension 8 and output $z := A \cdot y + c$

    where $A \in \{0,1\}^{8\times 8}$ is an invertible matrix and $c \in \{0,1\}^8$

## AES/Rijndael: Crash course GF$(2^8)$

- Carrier: polynomials $\sum_{i=0}^{7} a_i X^i$ with coefficients in $\mathbb{Z}_2$.

  Succinct representation: Either as bit string, e.g. $a_0 a_1 \ldots a_7$, or as natural number $\sum_{i=0}^{7} a_i 2^i$.

  Example: $X + X^4 \hat{=} 01001000 \hat{=} 18$ and $1 + X^2 + X^4 \hat{=} 10101000 \hat{=} 21$.

- Addition: First add polynomials as usual, then reduce modulo $2$

  Example: $18 + 21 = 7$ in GF$(2^8)$.

  That is, simply bit-wise xor $a_0 a_1 \ldots a_7$ and $b_0 b_1 \ldots b_7$.

- Multiplication: First multiply polynomials as usual, then reduce modulo $2$ and $m(X) = 1 + X + X^3 + X^4 + X^8$.

  Example: $18 \cdot 21 = 97$ and $3^2 = 5$ in GF$(2^8)$.

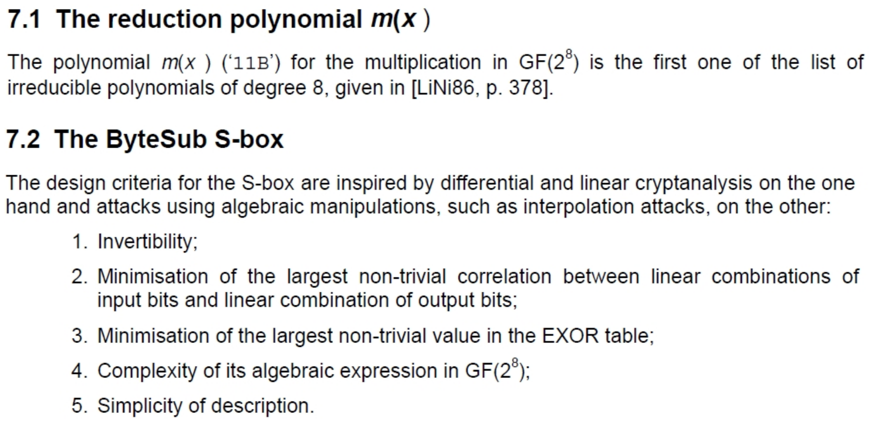
- Multiplicative inverse: Can be computed using the extendend Euclidean algorithm.

  Example: $18^{-1} = 170$ in GF$(2^8)$

## AES/Rijndael: SubBytes

- The basic goal of the computation in GF$(2^8)$ is to introduce a nonlinear operation which e.g. complicates differential and linear cryptanalysis.
- The re-interpretation of the 8-bit string in the second step tries to make the description of the combination of both steps difficult over GF$(2^8)$.

## From the Rijndael specifications:



### 7.1 The reduction polynomial *m*(*x*)

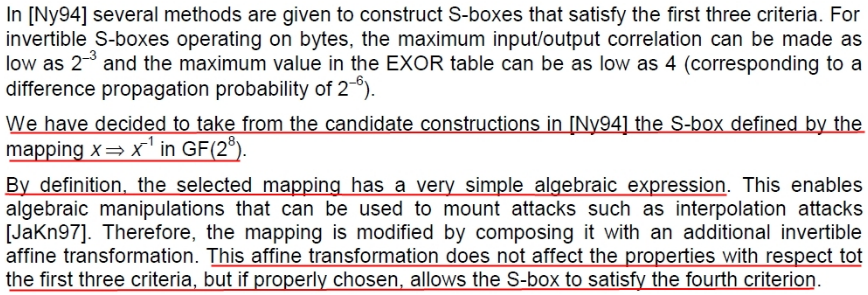
The polynomial *m*(*x*) (‘`11B`’) for the multiplication in GF($2^8$) is the first one of the list of irreducible polynomials of degree 8, given in [LiNi86, p. 378].

### 7.2 The ByteSub S-box

The design criteria for the S-box are inspired by differential and linear cryptanalysis on the one hand and attacks using algebraic manipulations, such as interpolation attacks, on the other:

1. Invertibility;
2. Minimisation of the largest non-trivial correlation between linear combinations of input bits and linear combination of output bits;
3. Minimisation of the largest non-trivial value in the EXOR table;
4. Complexity of its algebraic expression in GF($2^8$);
5. Simplicity of description.

## From the Rijndael specifications:



In [Ny94] several methods are given to construct S-boxes that satisfy the first three criteria. For invertible S-boxes operating on bytes, the maximum input/output correlation can be made as low as $2^{-3}$ and the maximum value in the EXOR table can be as low as 4 (corresponding to a difference propagation probability of $2^{-6}$).

We have decided to take from the candidate constructions in [Ny94] the S-box defined by the mapping $x \Rightarrow x^{-1}$ in GF($2^8$).

By definition, the selected mapping has a very simple algebraic expression. This enables algebraic manipulations that can be used to mount attacks such as interpolation attacks [JaKn97]. Therefore, the mapping is modified by composing it with an additional invertible affine transformation. This affine transformation does not affect the properties with respect tot the first three criteria, but if properly chosen, allows the S-box to satisfy the fourth criterion.

- Nyberg discusses in [Ny94] how to design S-boxes which, amongst others, make differential and linear cryptanalysis difficult.
- See also S-Box Design: A Literature Survey.

## AES/Rijndael: ShiftRows and MixColumns

- Data representation used for ShiftRows and MixColumns:

▷ By definition, the block length equals $32 \cdot \mathrm{Nb}$ for $\mathrm{Nb} \in \{4, 6, 8\}$.

▷ The $\mathrm{Nb} \cdot 32$bit string manipulated in every round is called the $\mathrm{state}$.

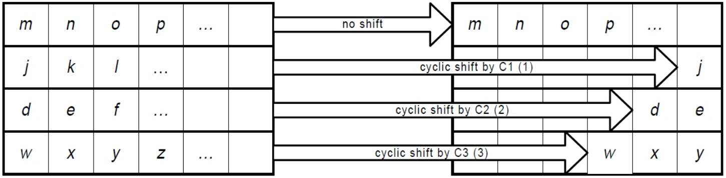
▷ AES/Rijndael interprets the $\mathrm{state}$ as a $4 \times \mathrm{Nb}$ matrix with $8$bit entries.

▷ For simplicity, let $\mathrm{Nb} = 4$ (as in AES). Then:

$$
\begin{aligned}
b_0 b_1 b_2 \ldots b_{126} b_{127} \quad &\hat{=} \quad \begin{pmatrix} (b_0 \ldots b_7) & \ldots & (b_{96} \ldots b_{103}) \\ \vdots & & \vdots \\ (b_{24} \ldots b_{31}) & \ldots & (b_{120} \ldots b_{127}) \end{pmatrix} \\
&=: \quad \begin{pmatrix} b^{(0,0)} & b^{(0,1)} & b^{(0,2)} & b^{(0,3)} \\ b^{(1,0)} & b^{(1,1)} & b^{(1,2)} & b^{(1,3)} \\ b^{(2,0)} & b^{(2,1)} & b^{(2,2)} & b^{(2,3)} \\ b^{(3,0)} & b^{(3,1)} & b^{(3,2)} & b^{(3,3)} \end{pmatrix}
\end{aligned}
$$

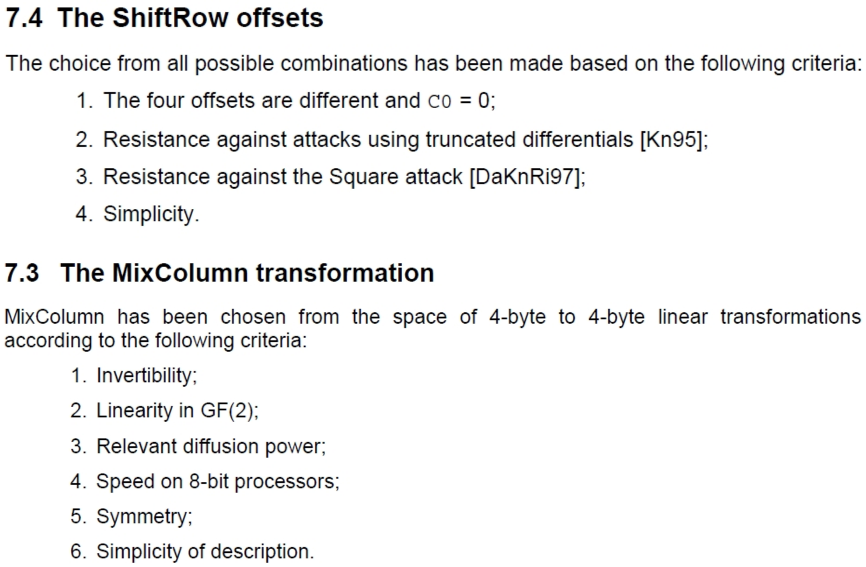
## AES/Rijndael: ShiftRows and MixColumns

- ShiftRows:
  - Consists of a predefined cyclic shift applied to each row ([11]):



  - The choice of the shifts makes Rijndael resistant against specific attacks [11].
- MixColumns:
  - Every column is interpreted as a polynomial of degree $4$ in $\mathsf{GF}(2^8)[X]$, and then multiplied by a predifined polynomial $c(X) \in \mathsf{GF}(2^8)[X]/X^4+1$.
  - Besides diffusion, also efficient implementation influenced the design.
- Within two rounds full diffusion is achieved [11].

## From the Rijndael specifications:



### 7.4 The ShiftRow offsets

The choice from all possible combinations has been made based on the following criteria:

1. The four offsets are different and `C0` = 0;
2. Resistance against attacks using truncated differentials [Kn95];
3. Resistance against the Square attack [DaKnRi97];
4. Simplicity.

### 7.3 The MixColumn transformation

MixColumn has been chosen from the space of 4-byte to 4-byte linear transformations according to the following criteria:

1. Invertibility;
2. Linearity in GF(2);
3. Relevant diffusion power;
4. Speed on 8-bit processors;
5. Symmetry;
6. Simplicity of description.

## AES/Rijndael: Security

- Recently, an attack on full-round AES has been found which is roughly four times faster than brute force enumeration of all keys, e.g. in case of AES-128 instead of $2^{128}$ only $2^{124}$ computations are needed. [9]
- For AES-192/256 several related-key attacks are known:
  - Related secret keys: keys unknown to the attacker, but he knows that keys are somehow related to each other (e.g. key one is the negation of key two), and the attacker knows or even may choose this relation.
  - Using a related-four-key attack, where the relation is chosen by the attacker, AES-256 can be broken in $2^{99.5}$ steps.
  - ▷ This is not considered to be a practical attack, but it is taken as a hint that number of rounds might be too small for key lengths $192/256$.

## AES/Rijndael: Security

- Another concern is its simple algebraic formulation:
  - The S-box can be completely described as quadratic equation system on GF$(2^8)$ by embedding AES in a block cipher of block length $1024$bits [18].
  - This in turn yields a quadratic equation system for the complete cipher whose solution yields the key [18, 8].
- Currently, only side-channel attacks on concrete implementations of AES pose the greatest risk.

  See e.g. here for an overview, or here for a specific attack.

## Quantum exhaustive key search

- Exhaustive key search: Given $x, y$ find $k$ s.t. $F_k(x) = y$

▷ I.e. search the unordered list $(k, F_k(x))_{k \in \{0,1\}^n}$ for $(k, y)$.

  - No need to store the list explicitly.

- Searching an unordered list of $N$ elements takes
  - on a classical computer $N$ time and constant space.
  - on a quantum computer using Grover's algorithm $\sqrt{N}$ time and $\log_2 N$ space (qubits).

- Exhaustive key search would be reduced from $2^{128}$ to $2^{64}$ in case of AES-128 – which is at least on the verge of practical attacks.

▷ See wikipedia for the development of quantum computers.

Currently (2018) Google claims to have a $72$ qubit computer.

## Keccak

- On October 2, 2012, the NIST chose Keccak as the new secure hash algorithm SHA-3. ([link])
  - ▷ The final specifications have yet to be fixed.
- Similarly to the choice of Rijndael for AES, Keccak was the winner of a three-round contest:
  - ▷ Submission deadline was October 31, 2008.
  - ▷ Finalists:
    - BLAKE
    - Grøstel i.a. Knudsen (Serpent/AES finalist) based on MD
    - JH
    - Keccak i.a. Daemen (Rijndael/AES)
    - Skein i.a. Schneier (Twofish/AES finalist), Bellare based on TBCs
  - ▷ For a summary see [here].

## Keccak

- NIST initiated the contest because of attacks on MD5 and (to some extent) on SHA-1.
    - Official announcement of the SHA-3 competition: [PDF]
    - ▷ Note that NIST still conjectures SHA-2 to be secure.
- ▷ Both MD5, SHA-1, and SHA-2 are using the MD construction (later).
- ▷ NIST wanted to have an alternative to SHA-2 which does not use the MD construction.
- ▷ Keccak is based on the so-called sponge construction.

## Keccak

- (Possible) advantage of the SHA-3 finalists:

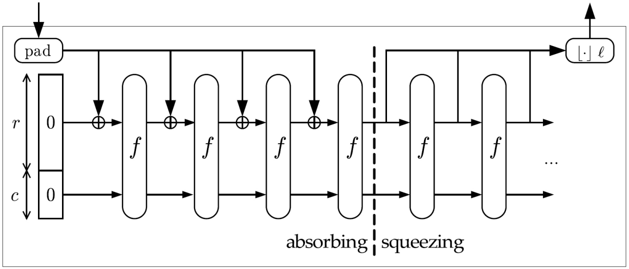
▷ NSA is stated as the author of SHA-2.

▷ The finalist have undergone the same public review process as the AES finalists.

▷ Several (e.g. Skein and Keccak) come with proofs of security.

▷ NIST explicitly encouraged to take typical cryptographic applications of hash functions into account for the design like MACs, PRFs, PRGs, and so forth.

## Sponge construction

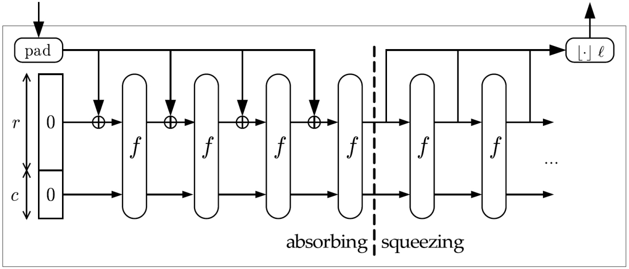


Taken from http://sponge.noekeon.org/

- $r, c \in \mathbb{N}$.
- $\mathsf{pad}(x) := x||10^p1$ with $p$ minimal such that $|\mathsf{pad}(x)| \in r\mathbb{Z}$.
- $f\colon \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ (not required to be a permutation).

▷ Short: "'sponge-$f$" for sponge with a specific $f$.

# Sponge construction



Taken from http://sponge.noekeon.org/

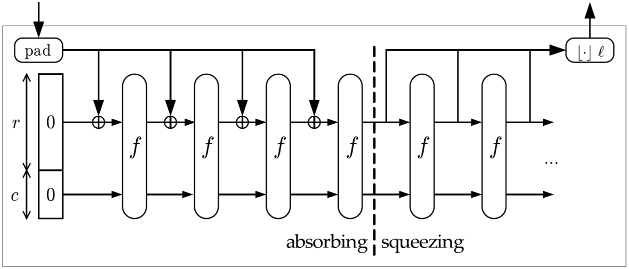
- "'Absorption" of input $x$:

▷ Let $x^{(1)}||\ldots||x^{(t)} = \mathsf{pad}(x) \in (\{0,1\}^r)^*$.

▷ Initially: $R||C := 0^{r+c}$.

▷ For $i$ from $1$ to $t$ do: $R||C := f(R \oplus x^{(i)}||C)$.

## Sponge construction



Taken from http://sponge.noekeon.org/

- "'Squeezing" out (for simplicty) $k \cdot r$ output bits:

▷ From absoprtion of $x$: $R||C$.

▷ Init: $y := R$.

▷ For $i$ from $2$ to $k$ do: $R||C := f(R||C)$; $y := y||R$;

  - In general: truncate output to a prefix of required length.

## Keccak: Choosing the parameters

- Output length $n$ influences the prob. of an output collision:

  I.e. the prob. that two different inputs yield the same output $\sim 2^{-n/2}$

- Capacity $c$ influences the prob. of an inner collision:

  I.e. the prob. that two different inputs yield the same internal state $\sim 2^{-c/2}$

- The $f$ used by Keccak is defined for $r + c = 25 \cdot 2^l$ bits with $l = 0, \ldots, 6$.

- For "‘$256$bit security”

  - i.e. $2^{-n/2} \leq 2^{-256}$ and $2^{-c/2} \leq 2^{-256}$.

  take (at least) $c = 512$ and $n = 512$.

- The larger $r$ the faster the input is "absorbed".

  For the maximal $l = 6$ we get $r = 1088$.

- Similar to AES and Rijndael, SHA3 only uses a subset of all the possible parameter values of Keccak.

▷ All SHA3 versions use a the same permutation on $1600$ bits ($l = 6$), they only differ in the output length $n$ and the capacity $c$.

- In one of the first proposals, for SHA3 capacity was set to $c = 2n$, so that the (conjectured) (2nd) preimage resistance was higher than the collision resistance,

- In a later proposal, this was changed to $c = n$ so that collision and (2nd) preimage resistance matched, and at the same time absorption became faster.

▷ This led to some criticism, so that in the most recent draft of the SHA3 standard $c = 2n$ is used again.

## Keccak: SHA3

- Official standards for using KECCAK with $r + c = 1600$:
  - For fixed-length output:

    SHA3-224$(m)$ := Keccak$[c = 448](M||01)[0 : 224]$

    SHA3-256$(m)$ := Keccak$[c = 512](M||01)[0 : 256]$

    SHA3-384$(m)$ := Keccak$[c = 786](M||01)[0 : 384]$

    SHA3-512$(m)$ := Keccak$[c = 1024](M||01)[0 : 512]$
  - For variable-length output:

    SHAKE128$(m, d)$ := Keccak$[c = 256](M||1111)[0 : d]$

    SHAKE256$(m, d)$ := Keccak$[c = 512](M||1111)[0 : d]$
  - In all cases Keccak is the sponge construction appliead to a fixed permutation $f$ on $25 \cdot 2^l$ bits.

## Keccak-f

- Design of the "Keccak's $f$" similar to the design of the permutations/functions used in block ciphers like AES.

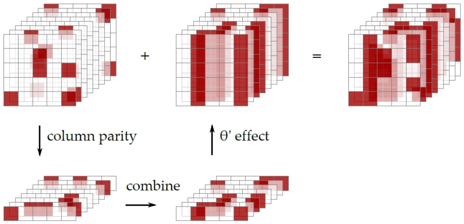
▷ Keccak's $f$ is a permutation of variables block length $5 \times 5 \times 2^l$.

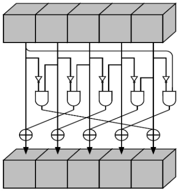
  - Bits are interpreted as a $5 \times 5 \times 2^l$ cube.
  - Think of it as "'three dimensional" version of AES' SubBytes-ShiftRows-MixCols permutation.

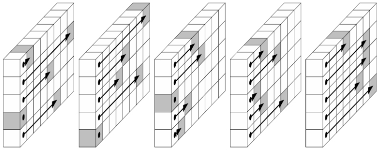
  ▷ Recall that ShiftRows-MixCols organizes bits as a matrix.

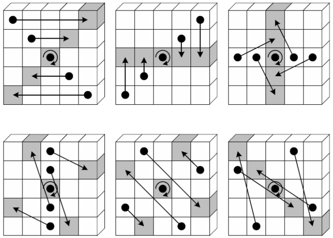
  - The number of times these operations have to applied in ordet to get the final output of $f$ is also fixed by the SHA3 standard.

## Keccak-f







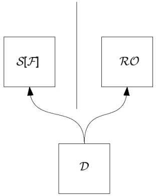


Taken from http://csrc.nist.gov/groups/ST/hash/sha-3/documents/Keccak-slides-at-NIST.pdf.

## Sponge construction: security properties

- The properties of the sponge, of course, depend strongly on the properties of $f$:
    - **Ex**: If Id is simply the identity, sponge-Id is not collision resistant.
- To assess the properties of the sponge construction independent of the concrete choice of $f$, only generic attacks are considered:
    - I.e. attacks that do not exploit properties of $f$ but only properties of the sponge construction.
    - ▷ To this end, consider attacks on a random sponge where $f$ is replaced by a RFO or a RPO.
    - ▷ That is, the adversary treats $f$ as a random function or permutation in his attack on "sponge-$f$".
    - ▷ Note that this is an idealization.

## Random sponge: Indistinguishability from RO



Taken from http://sponge.noekeon.org/

- Adversary needs to decide whether black box contains
  - either $\mathcal{S}[\mathcal{F}]$: random sponge (with $\mathcal{F}$ either a RPO or a RFO)
  - or a RO – a random orcale that can produce input of arbitrary length so that the adversary can "'squeeze" the sponge as much as he likes (resp. can given the time bound).
- Advantage of the adversary in both cases ($\mathcal{F}$ either a RPO or a RFO) upper bounded by $\sim \frac{N(N+1)}{2^{c+1}}$. (see here)
  - $N \ll 2^c$ is the total number of calls to $\mathcal{F}$.

## Random sponge: Indistinguishability from RO

- Problem of indistinguishability setting:

  In the real world, an attacker has access to $f$, so he should also have access to $\mathcal{F}$, and not only to $\mathcal{S}[\mathcal{F}]$.

- Indifferentiability setting by Maurer/Renner/Holenstein:

  Adversary again has to decide whether a black box contains

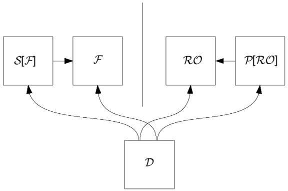
  - either $\mathcal{S}[\mathcal{F}]$ and $\mathcal{F}$
  - or RO and $\mathcal{P}[\text{RO}]$.

  ▷ $\mathcal{P}$ is a PPT-algorithm which has oracle access to RO and tries to simulate $\mathcal{F}$.

  ▷ $\mathcal{P}$ may depend on the specific adversary/attack.

  But he can query both $\mathcal{S}[\mathcal{F}]$ and $\mathcal{F}$ resp. RO and $\mathcal{P}[\text{RO}]$.

## Random sponge: Indifferentiability from RO



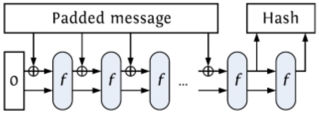
Taken from http://sponge.noekeon.org/

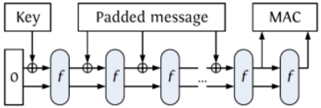
- Advantage of the adversary does not change (see here):

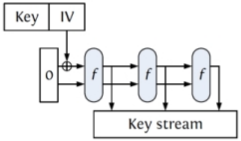
  In both cases a DPT-simulator exists such that the advantage is upper bounded by $\sim \frac{N(N+1)}{2^{c+1}}$.

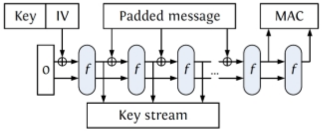
  - $N \ll 2^c$ is the total number of calls to $\mathcal{F}$.

# Keccak: some use cases









Taken from http://csrc.nist.gov/groups/ST/hash/sha-3/documents/Keccak-slides-at-NIST.pdf.