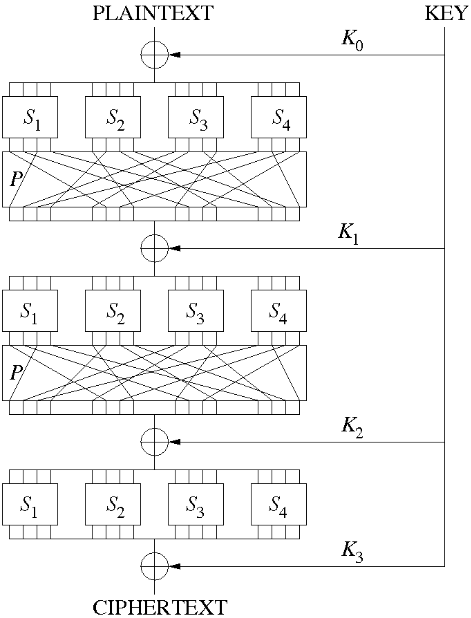
Feistel networks

In SPN Network S-boxes Invertible



SPN: PROS and CONS

PRO: With enough rounds secure.

CON: Hard to come up with invertible S-boxes.

Feistel Networks will not need invertible components but will be secure.

Feistel networks

1) Message length is ℓ. Just like SPN.

2) Key $k = k_1 \cdots k_r$ of length n. r rounds. Just like SPN.

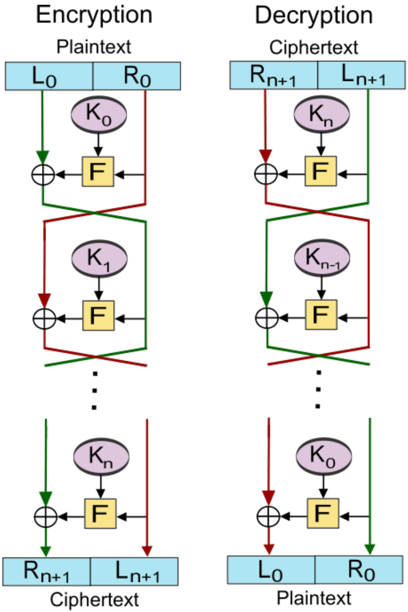
3) $|k_i| = n/r$. Need NOT be ℓ. Unlike SPN.

4) Use key k_i in ith round. Just like SPN.

5) Instead of S-boxes we have public functions $\hat{f}_i$. Need not be invertible! Unlike SPN. We derive $f_i(R) = \hat{f}_i(k_i, R)$ from them.

For 1-round:
Input: $L_0 R_0$, $|L_0| = |R_0| = \ell/2$.
Output: $L_1 R_1$ where $L_1 = R_0$, $R_1 = L_0 \oplus f_1(R_0)$
Invertible! The nature of $f_1(R)$ does not matter.
1) Input($L_1 R_1$)
2) $R_0 = L_1$.
3) Can compute $f_1(R_0)$ and hence $L_0 = R_1 \oplus f_1(R_0)$.

Feistel Network



r-round Feistel networks

1) Message length is ℓ. Just like SPN.

2) Key $k = k_1 \cdots k_r$ of length n. r rounds. Just like SPN.

3) $|k_i| = n/r$. Need NOT be ℓ. Unlike SPN.

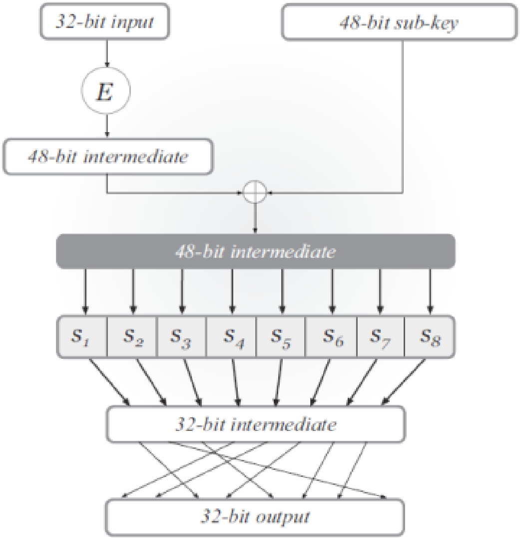
4) Use key k_i in ith round. Just like SPN.
5) Public functions $\hat{f}_i$. Need not be invertible! Unlike SPN.
$f_i(R) = \hat{f}_i(k_i, R)$ from

Input: $L_0 R_0$, $|L_0| = |R_0| = \ell/2$.
Output or Round 1: $L_1 R_1$ where $L_1 = R_0$, $R_1 = L_0 \oplus f_1(R_0)$
Output or Round 2: $L_2 R_2$ where $L_2 = R_1$, $R_2 = L_1 \oplus f_2(R_1)$
$\vdots \quad \vdots \quad \vdots$
Output or Round r: $L_r R_r$ where $L_r = R_{r-1}$, $R_r = L_{r-1} \oplus f_r(R_{r-1})$

Data Encryption Standard (DES)

- Standardized in 1977
- 56-bit keys, 64-bit block length
- 16-round Feistel network
  - Same round function in all rounds (but different sub-keys)
  - Basically an SPN design! But easier to build.

DES mangler function is $\hat{f}_i$



Avalanche effect – Like SPN!

- Consider 1-bit difference in left half of input
  - After 1 round, 1-bit difference in right half
  - S-boxes cause 2-bit difference, implying a 3-bit difference overall after 2 rounds
  - Mixing permutation spreads differences into different S-boxes
  - ...

Security of DES

PRO: DES is extremely well-designed

Security of DES

PRO: DES is extremely well-designed

PRO: Known attacks brute force or need lots of Plaintext.

Security of DES

PRO: DES is extremely well-designed

PRO: Known attacks brute force or need lots of Plaintext.

BIG CON: Parameters are too small! Brute-force search is feasible

56-bit key length

- A concern as soon as DES was released.
- Released in 1975, but that was then, this is now.

- Brute-force search over 2^{56} keys is possible
  - 1997: 1000s of computers, 96 days
  - 1998: distributed.net, 41 days
  - 1999: Deep Crack ($250,000), 56 hours
  - 2018: 48 FPGAs, 1 day
  - 2019: Will do as Classroom demo when teach this course in Fall of 2019.

Increasing key length?

- DES has a key that is too short
- How to fix?
  - Design new cipher. HARD!
  - Tweak DES so that it takes a larger key. HARD!
  - Build a new cipher using DES as a black box. EASY?

Double encryption

- Let $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$
  - (i.e. n=56, ℓ=64 for DES)
- Define $F^2 : \{0,1\}^{2n} \times \{0,1\}^\ell \to \{0,1\}^\ell$ as follows:

$$F^2_{k_1,k_2}(x) = F_{k_1}(F_{k_2}(x))$$

  (still invertible)

- If best known attack on F takes time 2^n, is it reasonable to assume that the best known attack on F^2 takes time 2^{2n}? Vote! YES, NO, UNKNOWN TO SCIENCE

Double encryption

- Let $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$
  - (i.e. n=56, ℓ=64 for DES)
- Define $F^2 : \{0,1\}^{2n} \times \{0,1\}^\ell \to \{0,1\}^\ell$ as follows:

$$F^2_{k_1,k_2}(x) = F_{k_1}(F_{k_2}(x))$$

  (still invertible)

- If best known attack on F takes time 2^n, is it reasonable to assume that the best known attack on F^2 takes time 2^{2n}?
  Vote! YES, NO, UNKNOWN TO SCIENCE
  NO

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk

N0 Its just Shift!

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

4) Vig: if Vig twice, does sec increase? Vote: Yes, No, Unk

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

4) Vig: if Vig twice, does sec increase? Vote: Yes, No, Unk
YES Key size is LCM(k_1, k_2).

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

4) Vig: if Vig twice, does sec increase? Vote: Yes, No, Unk
YES Key size is LCM(k_1, k_2).

5) Matrix: if Matrix twice, does sec increase? Vote: Yes, No, Unk

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

4) Vig: if Vig twice, does sec increase? Vote: Yes, No, Unk
YES Key size is LCM(k_1, k_2).

5) Matrix: if Matrix twice, does sec increase? Vote: Yes, No, Unk
NO Its just Matrix!

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

4) Vig: if Vig twice, does sec increase? Vote: Yes, No, Unk
YES Key size is LCM(k_1, k_2).

5) Matrix: if Matrix twice, does sec increase? Vote: Yes, No, Unk
NO Its just Matrix!

6) OTP: if OTP twice, does security increase? Vote: Yes, No, Unk

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

4) Vig: if Vig twice, does sec increase? Vote: Yes, No, Unk
YES Key size is LCM(k_1, k_2).

5) Matrix: if Matrix twice, does sec increase? Vote: Yes, No, Unk
NO Its just Matrix!

6) OTP: if OTP twice, does security increase? Vote: Yes, No, Unk
NO Its just OTP!

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

4) Vig: if Vig twice, does sec increase? Vote: Yes, No, Unk
YES Key size is LCM(k_1, k_2).

5) Matrix: if Matrix twice, does sec increase? Vote: Yes, No, Unk
NO Its just Matrix!

6) OTP: if OTP twice, does security increase? Vote: Yes, No, Unk
NO Its just OTP!

7) RSA: if RSA twice, does security increase? Vote: Yes, No, Unk

Encrypt Twice – Lets look at Past Ciphers

1) Shift: if Shift twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Shift!

2) Affine: if Affine twice, does sec increase? Vote: Yes, No, Unk
N0 Its just Affine!

3) Cubic: if Cubic twice, does sec increase? Vote: Yes, No, Unk
YES Higher Deg poly!

4) Vig: if Vig twice, does sec increase? Vote: Yes, No, Unk
YES Key size is $\mathrm{LCM}(k_1, k_2)$.

5) Matrix: if Matrix twice, does sec increase? Vote: Yes, No, Unk
NO Its just Matrix!

6) OTP: if OTP twice, does security increase? Vote: Yes, No, Unk
NO Its just OTP!

7) RSA: if RSA twice, does security increase? Vote: Yes, No, Unk
NO $(m^e)^e = m^{e^2}$.

Encrypt Twice Sometimes Gives the Same Exact Cipher

Encrypting Twice:

Shift, Affine, Matrix: Give same cipher, NO increase in key length.

Cubic: Gave diff cipher.

Vig: Gave same cipher but longer key length. So Still crackable?

DES:
Is double-DES really DES with a longer key? Vote: Yes, No, Unk.

Encrypt Twice Sometimes Gives the Same Exact Cipher

Encrypting Twice:

Shift, Affine, Matrix: Give same cipher, NO increase in key length.

Cubic: Gave diff cipher.

Vig: Gave same cipher but longer key length. So Still crackable?

DES:
Is double-DES really DES with a longer key? Vote: Yes, No, Unk.
No, for technical reasons I don't want to get into.

Encrypt Twice Sometimes Gives the Same Exact Cipher

Encrypting Twice:

Shift, Affine, Matrix: Give same cipher, NO increase in key length.

Cubic: Gave diff cipher.

Vig: Gave same cipher but longer key length. So Still crackable?

DES:
Is double-DES really DES with a longer key? Vote: Yes, No, Unk.
No, for technical reasons I don't want to get into.
Is double-DES harder than DES to crack?? Vote: Yes, No, Unk.

Encrypt Twice Sometimes Gives the Same Exact Cipher

Encrypting Twice:

Shift, Affine, Matrix: Give same cipher, NO increase in key length.

Cubic: Gave diff cipher.

Vig: Gave same cipher but longer key length. So Still crackable?

DES:
Is double-DES really DES with a longer key? Vote: Yes, No, Unk.
No, for technical reasons I don't want to get into.
Is double-DES harder than DES to crack?? Vote: Yes, No, Unk.
No

Encrypt Twice Sometimes Gives the Same Exact Cipher

Encrypting Twice:

Shift, Affine, Matrix: Give same cipher, NO increase in key length.

Cubic: Gave diff cipher.

Vig: Gave same cipher but longer key length. So Still crackable?

DES:
Is double-DES really DES with a longer key? Vote: Yes, No, Unk.
No, for technical reasons I don't want to get into.
Is double-DES harder than DES to crack?? Vote: Yes, No, Unk.
No
Next slide is Meet-in-the-Middle attack.

Encrypt Twice

We show that Encrypting twice does not help much in general. Let $\Pi = (Gen, Enc, Dec)$ be an encryption scheme. Let n be a security parameter which will be the length of the key.
Dr. Birdz has the following idea:

1) Alice and Bob share two keys k_1, k_2.
2) To encode m: send $Enc(k_1, Enc(k_2, m))$
3) To decode c: $Dec(Dec(k_1, c), k_2)$

Hope: Eve needs k_1 and k_2, $2n$ bits, twice as hard to crack.

Encrypt Twice

We show that Encrypting twice does not help much in general. Let $\Pi = (Gen, Enc, Dec)$ be an encryption scheme. Let n be a security parameter which will be the length of the key.
Dr. Birdz has the following idea:

1) Alice and Bob share two keys k_1, k_2.

2) To encode m: send $Enc(k_1, Enc(k_2, m))$

3) To decode c: $Dec(Dec(k_1, c), k_2)$

Hope: Eve needs k_1 and k_2, $2n$ bits, twice as hard to crack.
We Dash That Hope: We show that Eve can crack in $\sim 2^n$ steps.
Caveat: Eve needs LOTS of space.

Meet-in-the-middle attack

1) Alice and Bob share two keys k_1, k_2.

2) To encode m: send $Enc(k_1, Enc(k_2, m))$

3) To decode c: $Dec(Dec(k_1, c), k_2)$

Note: $m = Dec(Dec(k_1, c), k_2)$, so $Enc(m, k_2) = Dec(c, k_1)$

Meet-in-the-middle attack

Note: $m = Dec(Dec(k_1, c), k_2)$, so $Enc(m, k_2) = Dec(c, k_1)$
Assume Eve has one (m, c) pair.

1) $(\forall k \in \{0,1\}^n)$ Eve comp. $Enc(m, k)$. Sort 2^n $(Enc(m, k), k)$.
2) $(\forall k \in \{0,1\}^n)$ Eve comp. $Dec(c, k)$. Sort 2^n $(Dec(c, k), k)$.
3) Find pairs from each list that agree on 1st comp m.
4) Have $(m, k_2) = (m, k_1)$ so have k_1, k_2.

Time: $2 \times (2^{n+1} + n2^n) = 2^{n+2} + n2^{n+1}$.
Can do better: Can avoid Sorting (HW).
Upshot: Double Encryption did NOT double the exponent for Eve.

Triple encryption

- Define $F^3 : \{0,1\}^{3n} \times \{0,1\}^{\ell} \to \{0,1\}^{\ell}$ as follows:

$$F^3_{k_1,k_2,k_3}(x) = F_{k_1}(F_{k_2}(F_{k_3}(x)))$$

- Can do meet-in-the-middle but would be 2^{2n}.
- No better attack known.

Two-key triple encryption

- Define $F^3 : \{0,1\}^{2n} \times \{0,1\}^{\ell} \to \{0,1\}^{\ell}$ as follows:

$$F^3_{k_1,k_2}(x) = F_{k_1}(F_{k_2}(F_{k_1}(x)))$$

- Best attacks take time 2^{2n} — optimal given the key length!
- Sames on key length.
- Good for some backward-compatibiliy issues