Slide 1

CRYPTON: A New 128-bit Block Cipher

Submitter: Future Systems, Inc., Seoul, Korea

Algorithm Inventor: Chae Hoon Lim, Future Systems, Inc.

Slide 2

Contents

- High-level overview: Overall structure and main features
- Algorithm specification: component functions, en/decryption, key schedule
- Security analysis: differential/linear, key schedule cryptanalysis
- Implementation and efficiency: on various platforms.
- Conclusion

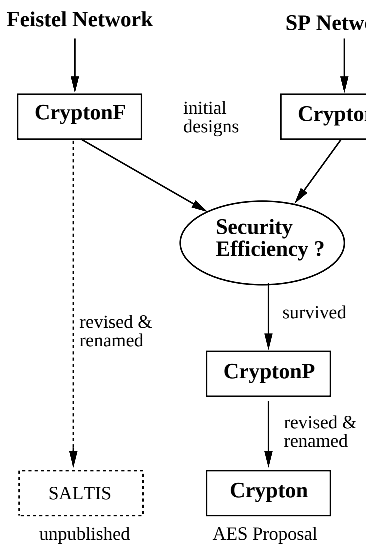
Slide 3

Design Goals

- Security against existing attacks.
- Efficiency in both SW and HW:
  - symmetry in encryption and decryption (identical process).
  - fast HW implementation for high speed applications.
- Efficient key schedule
  - efficiency in applications requiring frequent re-keying: hashing, MAC (e.g., for use in authentication protocols), smart card implementation, etc.
  - key setup within the time equivalent to a few blocks of encryption.

Slide 4

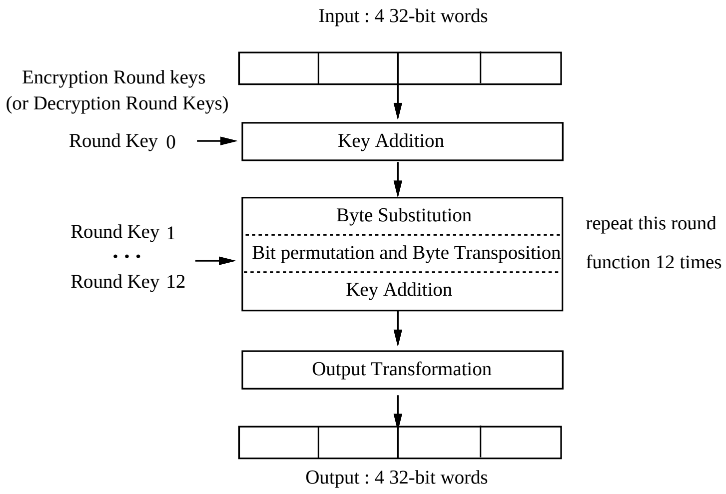
Design History



High-Level Structure of CRYPTON

Slide 5

- Overall structure: influenced by SQUARE (not Feistel-type).



Symbols and Notation

Slide 6

- Each component function acts on a 4×4 byte array.
  E.g., $A = (A[3], A[2], A[1], A[0])^t$:

$A[0]$	a_{03}	a_{02}	a_{01}	a_{00}
$A[1]$	a_{13}	a_{12}	a_{11}	a_{10}
$A[2]$	a_{23}	a_{22}	a_{21}	a_{20}
$A[3]$	a_{33}	a_{32}	a_{31}	a_{30}

- $X^{\ll n}$, $X \ll n$: $f \circ g$
- $\wedge, \vee, \oplus$; $\overline{X}$.

Slide 7

Main Features

- 12-round self-invertible cipher with block length 128 bits.
- Key length supported: $64 + 32k$ $(0 \leq k \leq 6)$ bits.
- Strong security against existing attacks
- Fine-grained design: byte-wise operations (AND, OR, XOR, table lookup) $\rightarrow$ word-wise operations.
- High parallelism for fast implementation in both SW and HW.
- Tradeoffs between speed and memory: none (HW); 48, 512 bytes (8-bit μP); 4K (32-bit μP); 12K, 512K (64-bit μP), etc.
- High speed in SW: 66/54 Mbps (Asm/C) on Pentium Pro 200 MHz (almost 2 times faster than DES).

Slide 8

Basic Building Blocks

- Components of Round Transformation ρ:
  1. S-box Transformation $\gamma : \gamma_o, \gamma_e$
  2. Bit Permutation $\pi : \pi_o, \pi_e$
  3. Byte Transposition τ
  4. Key Addition σ
- Round Transformation:
  - Odd rounds: $\rho_o = \sigma \circ \tau \circ \pi_o \circ \gamma_o$.
  - Even rounds: $\rho_e = \sigma \circ \tau \circ \pi_e \circ \gamma_e$.

Slide 9

S-box Transformation γ (1)

- Nonlinear Transform: byte-wise substitution using two 8×8 S-boxes S_0 and S_1 such that

$$S_0(S_1(x)) = S_1(S_0(x)) = x$$

for any $x \in [0, 256)$.

- Two nonlinear transforms, γ_o for odd rounds and γ_e for even rounds, such that

$$\gamma_o(\gamma_e(A)) = \gamma_o(\gamma_e(A)) = A$$

for any 4×4 byte array A.

Slide 10

S-box Transformation γ (2)

- Odd rounds (i.e., rounds 1, 3, etc.): $B = \gamma_o(A)$ defined by

$$B[0] \leftarrow S_1(a_{03}) \mid\mid S_0(a_{02}) \mid\mid S_1(a_{01}) \mid\mid S_0(a_{00}),$$
$$B[1] \leftarrow S_0(a_{13}) \mid\mid S_1(a_{12}) \mid\mid S_0(a_{11}) \mid\mid S_1(a_{10}),$$
$$B[2] \leftarrow S_1(a_{23}) \mid\mid S_0(a_{22}) \mid\mid S_1(a_{21}) \mid\mid S_0(a_{20}),$$
$$B[3] \leftarrow S_0(a_{33}) \mid\mid S_1(a_{32}) \mid\mid S_0(a_{31}) \mid\mid S_1(a_{30}).$$

- Even rounds (i.e., rounds 2, 4, etc.): $B = \gamma_e(A)$ defined by i

$$B[0] \leftarrow S_0(a_{03}) \mid\mid S_1(a_{02}) \mid\mid S_0(a_{01}) \mid\mid S_1(a_{00}),$$
$$B[1] \leftarrow S_1(a_{13}) \mid\mid S_0(a_{12}) \mid\mid S_1(a_{11}) \mid\mid S_0(a_{10}),$$
$$B[2] \leftarrow S_0(a_{23}) \mid\mid S_1(a_{22}) \mid\mid S_0(a_{21}) \mid\mid S_1(a_{20}),$$
$$B[3] \leftarrow S_1(a_{33}) \mid\mid S_0(a_{32}) \mid\mid S_1(a_{31}) \mid\mid S_0(a_{30}).$$

Slide 11

Bit Permutation π (1)

- Masking words:

$$\begin{aligned} M_0 &= m_3\|m_2\|m_1\|m_0 = \texttt{0x3fcff3fc}, \\ M_1 &= m_0\|m_3\|m_2\|m_1 = \texttt{0xfc3fcff3}, \\ M_2 &= m_1\|m_0\|m_3\|m_2 = \texttt{0xf3fc3fcf}, \\ M_3 &= m_2\|m_1\|m_0\|m_3 = \texttt{0xcff3fc3f}, \end{aligned}$$

  where $m_0 = \texttt{0xfc}, m_1 = \texttt{0xf3}, m_2 = \texttt{0xcf}, m_3 = \texttt{0x3f}$.

- Different versions of π, π_o for odd rounds and π_e for even rounds, such that

$$\pi_o^{-1} = \pi_o, \quad \pi_e^{-1} = \pi_e.$$

Slide 12

Bit Permutation π (2)

- Odd rounds: $B = \pi_o(A)$ defined by

$$\begin{aligned} B[0] &\leftarrow (A[0] \wedge M_0) \oplus (A[1] \wedge M_1) \oplus (A[2] \wedge M_2) \oplus (A[3] \wedge M_3) \\ B[1] &\leftarrow (A[0] \wedge M_1) \oplus (A[1] \wedge M_2) \oplus (A[2] \wedge M_3) \oplus (A[3] \wedge M_0) \\ B[2] &\leftarrow (A[0] \wedge M_2) \oplus (A[1] \wedge M_3) \oplus (A[2] \wedge M_0) \oplus (A[3] \wedge M_1) \\ B[3] &\leftarrow (A[0] \wedge M_3) \oplus (A[1] \wedge M_0) \oplus (A[2] \wedge M_1) \oplus (A[3] \wedge M_2) \end{aligned}$$

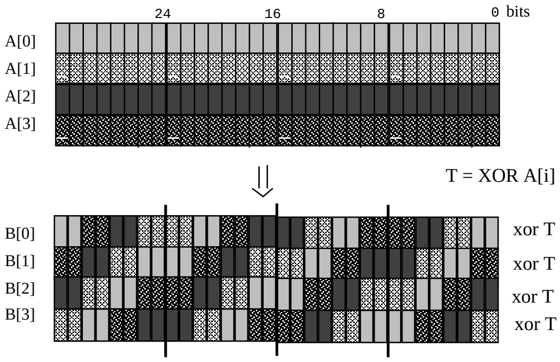
- Even rounds: $B = \pi_e(A)$ defined by

$$\begin{aligned} B[0] &\leftarrow (A[0] \wedge M_1) \oplus (A[1] \wedge M_2) \oplus (A[2] \wedge M_3) \oplus (A[3] \wedge M_0) \\ B[1] &\leftarrow (A[0] \wedge M_2) \oplus (A[1] \wedge M_3) \oplus (A[2] \wedge M_0) \oplus (A[3] \wedge M_1) \\ B[2] &\leftarrow (A[0] \wedge M_3) \oplus (A[1] \wedge M_0) \oplus (A[2] \wedge M_1) \oplus (A[3] \wedge M_2) \\ B[3] &\leftarrow (A[0] \wedge M_0) \oplus (A[1] \wedge M_1) \oplus (A[2] \wedge M_2) \oplus (A[3] \wedge M_3) \end{aligned}$$

Slide 13

Bit Permutation π (3)

- Graphical view of π_o:



Slide 14

Bit Permutation π (4)

- Shifting properties of π:

1. A^i: i-th byte column of A, i.e., $A^i = (a_{3i}, a_{2i}, a_{1i}, a_{0i})^t$.
   π_i: π restricted to the i-th byte column A^i.

$$\pi_o(A) = (\pi_3(A^3), \pi_2(A^2), \pi_1(A^1), \pi_0(A^0))^t,$$
$$\pi_e(A) = (\pi_0(A^3), \pi_3(A^2), \pi_2(A^1), \pi_1(A^0))^t.$$

2. If $\pi_0([d,c,b,a]^t) = [h,g,f,e]^t$, then

$$\begin{aligned}
\pi_1([d,c,b,a]^t) &= [e,h,g,f]^t, \\
\pi_2([d,c,b,a]^t) &= [f,e,h,g]^t, \\
\pi_3([d,c,b,a]^t) &= [g,f,e,h]^t.
\end{aligned}$$

Slide 15

Byte Transposition τ

- $B = \tau(A)$:

$$\begin{array}{c|c|c|c|c|} \cline{2-5} A[0] & a_{03} & a_{02} & a_{01} & a_{00} \\ \cline{2-5} A[1] & a_{13} & a_{12} & a_{11} & a_{10} \\ \cline{2-5} A[2] & a_{23} & a_{22} & a_{21} & a_{20} \\ \cline{2-5} A[3] & a_{33} & a_{32} & a_{31} & a_{30} \\ \cline{2-5} \end{array} \quad \overset{\tau}{\Rightarrow} \quad \begin{array}{c|c|c|c|c|} \cline{2-5} B[0] & a_{30} & a_{20} & a_{10} & a_{00} \\ \cline{2-5} B[1] & a_{31} & a_{21} & a_{11} & a_{01} \\ \cline{2-5} B[2] & a_{32} & a_{22} & a_{12} & a_{02} \\ \cline{2-5} B[3] & a_{33} & a_{23} & a_{13} & a_{33} \\ \cline{2-5} \end{array}$$

- $\tau^{-1} = \tau$.

Slide 16

Key Addition σ

- Key addition $\sigma_K : B = \sigma_K(A)$ defined by

$$\begin{aligned} B[0] &\leftarrow A[0] \oplus K[0] \\ B[1] &\leftarrow A[1] \oplus K[1] \\ B[2] &\leftarrow A[2] \oplus K[2] \\ B[3] &\leftarrow A[3] \oplus K[3] \end{aligned}$$

- $\sigma_K^{-1} = \sigma_K$

Slide 17

Encryption/Decryption (1)

- Note the round transformations under the round key K:

$$\begin{aligned}\rho_{oK}(A) &= (\sigma_K \circ \tau \circ \pi_o \circ \gamma_o)(A) \ \text{ for } r = 1, 3, \cdots \text{ etc.},\\ \rho_{eK}(A) &= (\sigma_K \circ \tau \circ \pi_e \circ \gamma_e)(A) \ \text{ for } r = 2, 4, \cdots \text{ etc.},\end{aligned}$$

- K_e^i : the i-th encryption round key consisting of 4 words.
- $\phi_e = \tau \circ \pi_e \circ \tau$: the output transformation.
- Encryption (r rounds):

$$E_K = \phi_e \circ \rho_{eK_e^r} \circ \rho_{oK_e^{r-1}} \circ \cdots \circ \rho_{eK_e^2} \circ \rho_{oK_e^1} \circ \sigma_{K_e^0}$$

Slide 18

Encryption/Decryption (2)

- Combining the last round with ϕ_e: $\phi_e \circ \rho_{eK_e^r} = \sigma_{K_d^0} \circ \tau \circ \gamma_e$.
- K_d^i : the i-th decryption round key given by

$$K_d^{r-i} = \begin{cases} \phi_e(K_e^i) & \text{for } i = 0, 2, 4, \cdots, \\ \phi_o(K_e^i) & \text{for } i = 1, 3, 5, \cdots. \end{cases}$$

- Decryption (r rounds):

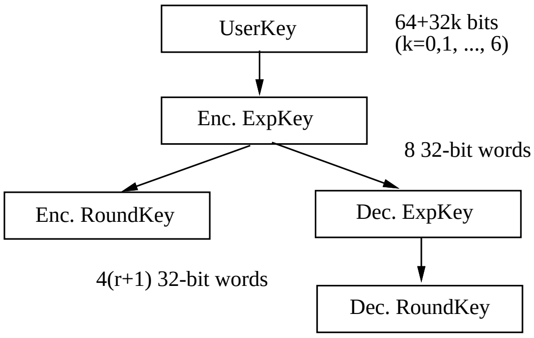
$$D_K = \phi_e \circ \rho_{eK_d^r} \circ \rho_{oK_d^{r-1}} \circ \cdots \circ \rho_{eK_d^2} \circ \rho_{oK_d^1} \circ \sigma_{K_d^0}$$

$\rightarrow$ the same process as encryption.

Key Scheduling (1)

Slide 19

- Different key schedules for encryption and decryption
- Two step generation of round keys:



Key Scheduling (2)

Slide 20

- Generation of Expanded Keys for encryption:
  1. 256-bit User Key: $K = k_{15} \cdots k_1 k_0$ (zero-extended).
  2. $U[i] = k_{4i+3}k_{4i+2}k_{4i+1}k_{4i}$ $(0 \le i \le 7)$.
  3. 8 Expanded Keys $E_e[i]$ $(0 \le i \le 7)$:

$$
\begin{aligned}
(V_e[3], V_e[2], V_e[1], V_e[0])^t &= (\tau \circ \gamma_o \circ \sigma_P \circ \pi_o)((U[6], U[4], U[2], U[0])^t), \\
(V_e[7], V_e[6], V_e[5], V_e[4])^t &= (\tau \circ \gamma_e \circ \sigma_Q \circ \pi_e)((U[7], U[5], U[3], U[1])^t), \\
T_0 &= V_e[0] \oplus V_e[1] \oplus V_e[2] \oplus V_e[3], \\
T_1 &= V_e[4] \oplus V_e[5] \oplus V_e[6] \oplus V_e[7], \\
E_e[i] &= V_e[i] \oplus T_1 \quad \text{for } i = 0, 1, 2, 3, \\
E_e[i] &= V_e[i] \oplus T_0 \quad \text{for } i = 4, 5, 6, 7,
\end{aligned}
$$

$(P = (P_3, P_2, P_1, P_0)^t,\ Q = (Q_3, Q_2, Q_1, Q_0)^t$: constants)

Slide 21

Key Scheduling (3)

- Generation of encryption round keys:
  $K_e^i = (K_e[4i+3], K_e[4i+2], K_e[4i+1], K_e[4i])^t$

  1. $K_e^0 = (E_e[3], E_e[2].E_e[1], E_e[0])^t$,
     $K_e^1 = (E_e[7], E_e[6].E_e[5], E_e[4])^t$.
  2. $K_e^{2i+2} = \mathrm{RoundKeyUpdate0}(K_e^{2i})$,
     $K_e^{2i+3} = \mathrm{RoundKeyUpdate1}(K_e^{2i+1})$ for $i \geq 0$, where

$$\mathrm{RoundKeyUpdate0}(D, C, B, A; n, m, RC) = (D \oplus RC, C^{\lll m}, B \oplus RC, A^{\lll n}),$$

$$\mathrm{RoundKeyUpdate1}(D, C, B, A; n, m, RC) = (D^{\lll m}, C \oplus RC, B^{\lll n}, A \oplus RC).$$

Slide 22

Key Scheduling (4)

- Rotation amounts and constants for updating enc. round keys

	$K_e^{2i} \to K_e^{2i+2}$				$K_e^{2i+1} \to K_e^{2i+3}$			
i	3	2	1	0	3	2	1	0
0	RC_0	16	RC_0	8	24	RC_0	16	RC_0
1	8	RC_1	24	RC_1	RC_1	16	RC_1	8
2	RC_2	24	RC_2	16	8	RC_2	24	RC_2
3	16	RC_3	8	RC_3	RC_3	24	RC_3	16
4	RC_4	8	RC_4	24	16	RC_4	8	RC_4
5	24	RC_5	16	RC_5				

- $RC_0 = \texttt{0x01010101}$, $RC_i = RC_{i-1} \lll 1$ for $i \geq 1$.

Slide 23

Key Scheduling (5)

- Generation of decryption round keys:

  1. convert E_e into E_d using ϕ_i:

     $$\{E_d[3], E_d[2], E_d[1], E_d[0]\} = \{\phi_0(K_e[51]), \phi_3(K_e[50]), \phi_2(K_e[49]), \phi_1(K_e[48])\},$$

     $$\{E_d[7], E_d[6], E_d[5], E_d[4]\} = \{\phi_3(K_e[47]), \phi_2(K_e[46]), \phi_1(K_e[45]), \phi_0(K_e[44])\}.$$

  2. The remaining steps are similar to those of encryption key schedule.

Slide 24

Diffusion Property of π

- Diffusion order of $\pi_i = 4$.
- Minimal number of possible active bytes in successive rounds

Class	1	2	3	4	5	6	7	8	sum
Class 1	starting with 1 active bytes								
	1	3	9	3	1	3	9	3	32
Class 2	starting with 2 active bytes								
	2	2	6	6	2	2	6	6	32
Class 3	starting with 3 active bytes								
	3	1	3	9	3	1	3	9	32

Slide 25

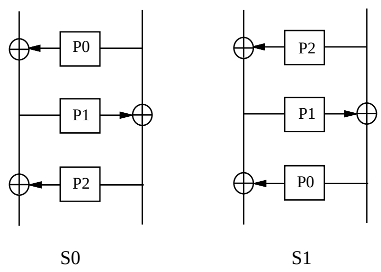
S-box Design History

- Initial : 8×8 S-box from an inverse poly. over GF(p).
  - self-invertible.
  - good DC/LC char.: $p_d = p_l = 2^{-6}$.
- Motivation to change
  - 16 parallel accesses to S-boxes in Crypton.
  - table look-up: sequential $\rightarrow$ high memory access delay.
  - fast and efficient implementation in HW ?
- Revision: 8×8 S-box from a 3-round Feistel cipher.
  - implementable in parallel HW logic.
  - a little worse DC/LC char. $\rightarrow$ No. rounds increased.

Slide 26

DC/LC Property of S-boxes (1)

- S-box construction from 4-bit S-boxes P_i.



- P_i's were selected so that S_i's have good differential/linear char. and that they can be implemented efficiently in hardware (107 nand gates of depth 20).

Slide 27

DC/LC Property of S-boxes (2)

- Boolean expressions for $y = P_i(x)$:

P_0 (16)	$y_0 = (x_2 \wedge x_3) \vee (\overline{x_1} \wedge \overline{x_3})$
	$y_1 = (x_0 \wedge x_3) \vee (\overline{x_0} \wedge \overline{x_2})$
	$y_2 = (x_1 \wedge x_2) \vee (\overline{x_0} \wedge \overline{x_2})$
	$y_3 = (x_0 \wedge x_1) \vee (\overline{x_1} \wedge \overline{x_3})$
P_1 (27)	$y_0 = (x_3 \wedge (\overline{x_1} \vee (\overline{x_0} \wedge x_2))) \vee ((\overline{\overline{x_0} \wedge \overline{x_2}}) \wedge ((\overline{x_0} \wedge \overline{x_1}) \vee (\overline{x_2} \wedge \overline{x_3})))$
	$y_1 = (x_0 \wedge (\overline{x_3} \vee (\overline{x_1} \wedge \overline{x_2}))) \vee ((\overline{x_1 \wedge \overline{x_2}}) \wedge ((\overline{x_0} \wedge x_1) \vee (\overline{x_2} \wedge \overline{x_3})))$
	$y_2 = (x_1 \wedge (\overline{x_3} \vee (\overline{x_0} \wedge \overline{x_2}))) \vee ((\overline{\overline{x_0} \wedge \overline{x_2}}) \wedge ((\overline{x_0} \wedge \overline{x_1}) \vee (\overline{x_2} \wedge \overline{x_3})))$
	$y_3 = (x_3 \wedge (\overline{x_0} \vee (x_1 \wedge \overline{x_2}))) \vee ((\overline{x_1 \wedge \overline{x_2}}) \wedge ((\overline{x_0} \wedge x_1) \vee (\overline{x_2} \wedge \overline{x_3})))$
P_2 (16)	$y_0 = (x_0 \wedge x_2) \vee (\overline{x_2} \wedge x_3)$
	$y_1 = (x_1 \wedge x_3) \vee (\overline{x_1} \wedge x_2)$
	$y_2 = (x_1 \wedge x_3) \vee (x_0 \wedge \overline{x_3})$
	$y_3 = (x_0 \wedge x_2) \vee (\overline{x_0} \wedge x_1)$

Slide 28

DC/LC Property of S-boxes (3)

- Complexity for DC/LC: $\sim 1/DP_{S_i}(1/LP_{S_i})$

$$DP_S = \max_{\Delta x \neq 0, \Delta y} \frac{\#\{x \in X | S(x) \oplus S(x \oplus \Delta x) = \Delta y\}}{2^n},$$

$$LP_S = \max_{\Gamma x, \Gamma y \neq 0} \left(\frac{\#\{x \in X | x \bullet \Gamma x = S(x) \bullet \Gamma y\} - 2^{n-1}}{2^{n-1}} \right)^2,$$

- DP/LP for selected S-boxes:

$$p_d \stackrel{\text{def}}{=} DP_{S_i} = \frac{8}{256} = 2^{-5},$$

$$p_l \stackrel{\text{def}}{=} LP_{S_i} = (\frac{2 \cdot 32}{256})^2 = 2^{-4},$$

Slide 29

Strength of CRYPTON against DC/LC

- Best possible 8-round characteristic prob.:

$$p_{C_8} < p_d^{32} = 2^{-160}.$$

- Best possible 8-round linear approx. prob.:

$$p_{L_8} < p_l^{32} = 2^{-128}.$$

- Conjecture:
  The complexity of 10-round CRYPTON against DC/LC $> 2^{128}$.

Slide 30

Strength against Other Possible Attacks

- believed secure against other variants of DC/LC (truncated/higher-order differential, multiple linear approx., etc.), algebraic attacks such as interpolation attack, etc.
- Key schedule cryptanalysis: no weak/semi-weak keys, no equivalent keys, no simple relations, no related keys.
  $\rightarrow$ No serious weakness found, but the key schedule will be strengthened in the near future.
- Strong resistance to timing attacks and differential power analysis due to uniform processing of simple boolean operations.

Slide 31

Implementation and Efficiency (1)

- Implementation on 32-bit microprocessors: use extended S-boxes of 4 Kbytes constructed by, for $0 \leq j \leq 255$,

$$\begin{aligned} SS_0[j] &= S_0[j] \wedge m_3 \parallel S_0[j] \wedge m_2 \parallel S_0[j] \wedge m_1 \parallel S_0[j] \wedge m_0, \\ SS_1[j] &= S_1[j] \wedge m_0 \parallel S_1[j] \wedge m_3 \parallel S_1[j] \wedge m_2 \parallel S_1[j] \wedge m_1, \\ SS_2[j] &= S_0[j] \wedge m_1 \parallel S_0[j] \wedge m_0 \parallel S_0[j] \wedge m_3 \parallel S_0[j] \wedge m_2, \\ SS_3[j] &= S_1[j] \wedge m_2 \parallel S_1[j] \wedge m_1 \parallel S_1[j] \wedge m_0 \parallel S_1[j] \wedge m_3. \end{aligned}$$

Slide 32

Implementation and Efficiency (2)

- Implementation on 32-bit microprocessors: $B = \rho_{oK}(A)$

$$\begin{aligned} B_0 &= SS_0[a_{00}] \oplus SS_1[a_{10}] \oplus SS_2[a_{20}] \oplus SS_3[a_{30}] \oplus K[0], \\ B_1 &= SS_1[a_{01}] \oplus SS_2[a_{11}] \oplus SS_3[a_{21}] \oplus SS_0[a_{31}] \oplus K[1], \\ B_2 &= SS_2[a_{02}] \oplus SS_3[a_{12}] \oplus SS_0[a_{22}] \oplus SS_1[a_{32}] \oplus K[2], \\ B_3 &= SS_3[a_{03}] \oplus SS_0[a_{13}] \oplus SS_1[a_{23}] \oplus SS_2[a_{33}] \oplus K[3]. \end{aligned}$$

Slide 33

Implementation and Efficiency (3)

- Speed on Pentium Pro (200 MHz) / UltraSparc (167 MHz):

Language\Clocks	Alg	Key (E/D)	Enc/Dec
Asm (PP)	N/A	N/A	370 / 370 (66 Mbps)
MSVC 5.0 (PP)	9740	325 / 360	450 / 450 (54 Mbps)
GNU C (US)	11460	450 / 500	600 / 600 (34 Mbps)

Slide 34

Implementation and Efficiency (4)

- Speed comparison with existing ciphers on various architectures (SARM $\rightarrow$ TMS320C6x: increasing parallelism):

cipher\Mbps	SARM (233MHz)	Pentium Pro (200MHz)		TMS320C6x (200MHz)
DES	17/17	29/29	36/36	33/33
CAST	29/28	36/36	54/55	40/41
Blowfish	33/33	49/49	58/58	53/50
RC5 (16)	46/48	84/100	85/101	61/55
Crypton	25/25	54/54	66/66	79/79

Slide 35

Implementation and Efficiency (5)

- Speed estimation on an 8-bit μP (Load-Store Arch.):

RAM	20 (data) + 208 (round keys) = 228 bytes		
EEPROM	512 bytes for S-boxes		
	# of instr	# of cycles	1 instr = 3 cycles
Enc/Dec	2544	7632	Round keys in RAM
Enc Key Setup	1096	3288	
Dec Key Setup	1246	3738	

RAM	20 (data) + 32 (UserKey) = 52 bytes		
EEPROM	512 bytes for S-boxes		
	# of instr	# of cycles	1 instr = 3 cycles
Encryption	3752	11256	In-line Key Schedule
Decryption	4168	12504	In-line Key Schedule

Slide 36

Implementation and Efficiency (6)

- Estimated gate count for VLSI implementation:
  - counted the number of 2-input nand gates.
  - 1 xor gate = 4 nand gates.

	# of nand gates	depth	register
Enc/Dec	38464	312	256 bits
Key Schedule	10688	$\ll 312$	288 bits
Total	49152	312	544 bits

Slide 37

Advantages / Limitations

- Simplicity and Ease of analysis/implementation.
- Identical process for encryption and decryption.
- Strong guaranteed security against DC/LC.
- Efficiency in both SW and HW: fine-grained design, high parallelism, hardware-efficient.
- Key scheduling: fast, easy to extend.
- No limitation compared to other block ciphers.

Slide 38

Revision and Future Work

- S-box transform revised (available on the CRYPTON home page):

$$\begin{aligned} &S_0, & S_1(x) &= S_0(x)^{\lll 6} \text{ for } x \in [0, 256), \\ S_2 &= S_0^{-1}, & S_3 &= S_1^{-1}. \end{aligned}$$

- Key schedule will be revised (to be posted on the web soon).
- Will continue to do further analysis.