CAST-256
Algorithm Specification

1. Algorithm Specification

1.1 CAST-128 Notation

The following notation from CAST-128 [A97b, A97c] is relevant to CAST-256.

- CAST-128 uses a pair of subkeys per round: a 5-bit quantity k_{r_i} is used as a "rotation" key for round *i* and a 32-bit quantity k_{m_i} is used as a "masking" key for round *i*.

- Three different round functions are used in CAST-128. The rounds are as follows (where D is the data input to the operation, I_a - I_d are the most significant byte through least significant byte of I, respectively, S_i is the i^{th} s-box (see following page for s-box definitions), and O is the output of the operation). Note that $+$ and - are addition and subtraction modulo 2^{32}, $\oplus$ is bitwise eXclusive-OR, and $\hookleftarrow$ is the circular left-shift operation.

  Type 1:
  $$I = ((k_{m_i} + D) \hookleftarrow k_{r_i})$$
  $$O = ((S_1[I_a] \oplus S_2[I_b]) - S_3[I_c]) + S_4[I_d]$$

  Type 2:
  $$I = ((k_{m_i} \oplus D) \hookleftarrow k_{r_i})$$
  $$O = ((S_1[I_a] - S_2[I_b]) + S_3[I_c]) \oplus S_4[I_d]$$

  Type 3:
  $$I = ((k_{m_i} - D) \hookleftarrow k_{r_i})$$
  $$O = ((S_1[I_a] + S_2[I_b]) \oplus S_3[I_c]) - S_4[I_d]$$

  Let f_1, f_2, f_3 be keyed round function operations of Types 1, 2, and 3 (respectively) above.

CAST-128 Notation (cont'd)

- CAST-128 uses four round function substitution boxes (s-boxes), S_1 - S_4. These are defined as follows (entries (written in hexadecimal notation) are to be read left-to-right, top-to-bottom).

S-Box S_1

30fb40d4	9fa0ff0b	6beccd2f	3f258c7a	1e213f2f	9c004dd3	6003e540	cf9fc949
bfd4af27	88bbbdb5	e2034090	98d09675	6e63a0e0	15c361d2	c2e7661d	22d4ff8e
28683b6f	c07fd059	ff2379c8	775f50e2	43c340d3	df2f8656	887ca41a	a2d2bd2d
a1c9e0d6	346c4819	61b76d87	22540f2f	2abe32e1	aa54166b	22568e3a	a2d341d0
66db40c8	a784392f	004dff2f	2db9d2de	97943fac	4a97c1d8	527644b7	b5f437a7
b82cbaef	d751d159	6ff7f0ed	5a097a1f	827b68d0	90ecf52e	22b0c054	bc8e5935
4b6d2f7f	50bb64a2	d2664910	bee5812d	b7332290	e93b159f	b48ee411	4bff345d
fd45c240	ad31973f	c4f6d02e	55fc8165	d5b1caad	a1ac2dae	a2d4b76d	c19b0c50
882240f2	0c6e4f38	a4e4bfd7	4f5ba272	564c1d2f	c59c5319	b949e354	b04669fe
b1b6ab8a	c71358dd	6385c545	110f935d	57538ad5	6a390493	e63d37e0	2a54f6b3
3a787d5f	6276a0b5	19a6fcdf	7a42206a	29f9d4d5	f61b1891	bb72275e	aa508167
38901091	c6b505eb	84c7cb8c	2ad75a0f	874a1427	a2d1936b	2ad286af	aa56d291
d7894360	425c750d	93b39e26	187184c9	6c00b32d	73e2bb14	a0bebc3c	54623779
64459eab	3f328b82	7718cf82	59a2cea6	04ee002e	89fe78e6	3fab0950	325ff6c2
81383f05	6963c5c8	76cb5ad6	d49974c9	ca180dcf	380782d5	c7fa5cf6	8ac31511
35e79e13	47da91d0	f40f9086	a7e2419e	31366241	051ef495	aa573b04	4a805d8d
548300d0	00322a3c	bf64cddf	ba57a68e	75c6372b	50afd341	a7c13275	915a0bf5
6b54bfab	2b0b1426	ab4cc9d7	449ccd82	f7fbf265	ab85c5f3	1b55db94	aad4e324
cfa4bd3f	2deaa3e2	9e204d02	c8bd25ac	eadf55b3	d5bd9e98	e31231b2	2ad5ad6c
954329de	adbe4528	d8710f69	aa51c90f	aa786bf6	22513f1e	aa51a79b	2ad344cc
7b5a41f0	d37cfbad	1b069505	41ece491	b4c332e6	032268d4	c9600acc	ce387e6d
bf6bb16c	6a70fb78	0d03d9c9	d4df39de	e01063da	4736f464	5ad328d8	b347cc96
75bb0fc3	98511bfb	4ffbcc35	b58bcf6a	e11f0abc	bfc5fe4a	a70aec10	ac39570a
3f04442f	6188b153	e0397a2e	5727cb79	9ceb418f	1cacd68d	2ad37c96	0175cb9d
c69dff09	c75b65f0	d9db40d8	ec0e7779	4744ead4	b11c3274	dd24cb9e	7e1c54bd
f01144f9	d2240eb1	9675b3fd	a3ac3755	d47c27af	51c85f4d	56907596	a5bb15e6
580304f0	ca042cf1	011a37ea	8dbfaadb	35ba3e4a	3526ffa0	c37b4d09	bc306ed9
98a52666	5648f725	ff5e569d	0ced63d0	7c63b2cf	700b45e1	d5ea50f1	85a92872
af1fbda7	d4234870	a7870bf3	2d3b4d79	42e04198	0cd0ede7	26470db8	f881814c
474d6ad7	7c0c5e5c	d1231959	381b7298	f5d2f4db	ab838653	6e2f1e23	83719c9e
bd91e046	9a56456e	dc39200c	20c8c571	962bda1c	e1e696ff	b141ab08	7cca89b9
1a69e783	02cc4843	a2f7c579	429ef47d	427b169c	5ac9f049	dd8f0f00	5c8165bf

S-Box S_2

1f201094	ef0ba75b	69e3cf7e	393f4380	fe61cf7a	eec5207a	55889c94	72fc0651
ada7ef79	4e1d7235	d55a63ce	de0436ba	99c430ef	5f0c0794	18dcdb7d	a1d6eff3
a0b52f7b	59e83605	ee15b094	e9ffd909	dc440086	ef944459	ba83ccb3	e0c3cdfb
d1da4181	3b092ab1	f997f1c1	a5e6cf7b	01420ddb	e4e7ef5b	25a1ff41	e180f806
1fc41080	179bee7a	d37ac6a9	fe5830a4	98de8b7f	77e83f4e	79929269	24fa9f7b
e113c85b	acc40083	d7503525	f7ea615f	62143154	0d554b63	5d681121	c866c359
3d63cf73	cee234c0	d4d87e87	5c672b21	071f6181	39f7627f	361e3084	e4eb573b
602f64a4	d63acd9c	1bbc4635	9e81032d	2701f50c	99847ab4	a0e3df79	ba6cf38c
10843094	2537a95e	f46f6ffe	a1ff3b1f	208cfb6a	8f458c74	d9e0a227	4ec73a34
fc884f69	3e4de8df	ef0e0088	3559648d	8a45388c	1d804366	721d9bfd	a58684bb
e8256333	844e8212	128d8098	fed33fb4	ce280ae1	27e19ba5	d5a6c252	e49754bd
c5d655dd	eb667064	77840b4d	a1b6a801	84db26a9	e0b56714	21f043b7	e5d05860
54f03084	066ff472	a31aa153	dadc4755	b5625dbf	68561be6	83ca6b94	2d6ed23b
eccf01db	a6d3d0ba	b6803d5c	af77a709	33b4a34c	397bc8d6	5ee22b95	5f0e5304
81ed6f61	20e74364	b45e1378	de18639b	881ca122	b96726d1	8049a7e8	22b7da7b
5e552d25	5272d237	79d2951c	c60d894c	488cb402	1ba4fe5b	a4b09f6b	1ca815cf
a20c3005	8871df63	b9de2fcb	0cc6c9e9	0beeff53	e3214517	b4542835	9f63293c
ee41e729	6e1d2d7c	50045286	1e6685f3	f33401c6	30a22c95	31a70850	60930f13

```
73f98417 a1269859 ec645c44 52c877a9 cdff33a6 a02b1741 7cbad9a2 2180036f
50d99c08 cb3f4861 c26bd765 64a3f6ab 80342676 25a75e7b e4e6d1fc 20c710e6
cdf0b680 17844d3b 31eef84d 7e0824e4 2ccb49eb 846a3bae 8ff77888 ee5d60f6
7af75673 2fdd5cdb a11631c1 30f66f43 b3faec54 157fd7fa ef8579cc d152de58
db2ffd5e 8f32ce19 306af97a 02f03ef8 99319ad5 c242fa0f a7e3ebb0 c68e4906
b8da230c 80823028 dcdef3c8 d35fb171 088a1bc8 bec0c560 61a3c9e8 bca8f54d
c72feffa 22822e99 82c570b4 d8d94e89 8b1c34bc 301e16e6 273be979 b0ffeaa6
61d9b8c6 00b24869 b7ffce3f 08dc283b 43daf65a f7e19798 7619b72f 8f1c9ba4
dc8637a0 16a7d3b1 9fc393b7 a7136eeb c6bcc63e 1a513742 ef6828bc 520365d6
2d6a77ab 3527ed4b 821fd216 095c6e2e db92f2fb 5eea29cb 145892f5 91584f7f
5483697b 2667a8cc 85196048 8c4bacea 833860d4 0d23e0f9 6c387e8a 0ae6d249
b284600c d835731d dcb1c647 ac4c56ea 3ebd81b3 230eabb0 6438bc87 f0b5b1fa
8f5ea2b3 fc184642 0a036b7a 4fb089bd 649da589 a345415e 5c038323 3e5d3bb9
43d79572 7e6dd07c 06dfdf1e 6c6cc4ef 7160a539 73bfbe70 83877605 4523ecf1
```

S-Box S_3

```
8defc240 25fa5d9f eb903dbf e810c907 47607fff 369fe44b 8c1fc644 aececa90
beb1f9bf eefbcaea e8cf1950 51df07ae 920e8806 f0ad0548 e13c8d83 927010d5
11107d9f 07647db9 b2e3e4d4 3d4f285e b9afa820 fade82e0 a067268b 8272792e
553fb2c0 489ae22b d4ef9794 125e3fbc 21fffcee 825b1bfd 9255c5ed 1257a240
4e1a8302 bae07fff 528246e7 8e57140e 3373f7bf 8c9f8188 a6fc4ee8 c982b5a5
a8c01db7 579fc264 67094f31 f2bd3f5f 40fff7c1 1fb78dfc 8e6bd2c1 437be59b
99b03dbf b5dbc64b 638dc0e6 55819d99 a197c81c 4a012d6e c5884a28 ccc36f71
b843c213 6c0743f1 8309893c 0feddd5f 2f7fe850 d7c07f7e 02507fbf 5afb9a04
a747d2d0 1651192e af70bf3e 58c31380 5f98302e 727cc3c4 0a0fb402 0f7fef82
8c96fdad 5d2c2aae 8ee99a49 50da88b8 8427f4a0 1eac5790 796fb449 8252dc15
efbd7d9b a672597d ada840d8 45f54504 fa5d7403 e83ec305 4f91751a 925669c2
23efe941 a903f12e 60270df2 0276e4b6 94fd6574 927985b2 8276dbcb 02778176
f8af918d 4e48f79e 8f616ddf e29d840e 842f7d83 340ce5c8 96bbb682 93b4b148
ef303cab 984faf28 779faf9b 92dc560d 224d1e20 8437aa88 7d29dc96 2756d3dc
8b907cee b51fd240 e7c07ce3 e566b4a1 c3e9615e 3cf8209d 6094d1e3 cd9ca341
5c76460e 00ea983b d4d67881 fd47572c f76cedd9 bda8229c 127dadaa 438a074e
1f97c090 081bdb8a 93a07ebe b938ca15 97b03cff 3dc2c0f8 8d1ab2ec 64380e51
68cc7bfb d90f2788 12490181 5de5ffd4 dd7ef86a 76a2e214 b9a40368 925d958f
4b39fffa ba39aee9 a4ffd30b faf7933b 6d498623 193cbcfa 27627545 825cf47a
61bd8ba0 d11e42d1 cead04f4 127ea392 10428db7 8272a972 9270c4a8 127de50b
285ba1c8 3c62f44f 35c0eaa5 e805d231 428929fb b4fcdf82 4fb66a53 0e7dc15b
1f081fab 108618ae fcfd086d f9ff2889 694bcc11 236a5cae 12deca4d 2c3f8cc5
d2d02dfe f8ef5896 e4cf52da 95155b67 494a488c b9b6a80c 5c8f82bc 89d36b45
3a609437 ec00c9a9 44715253 0a874b49 d773bc40 7c34671c 02717ef6 4feb5536
a2d02fff d2bf60c4 d43f03c0 50b4ef6d 07478cd1 006e1888 a2e53f55 b9e6d4bc
a2048016 97573833 d7207d67 de0f8f3d 72f87b33 abcc4f33 7688c55d 7b00a6b0
947b0001 570075d2 f9bb88f8 8942019e 4264a5ff 856302e0 72dbd92b ee971b69
6ea22fde 5f08ae2b af7a616d e5c98767 cf1febd2 61efc8c2 f1ac2571 cc8239c2
67214cb8 b1e583d1 b7dc3e62 7f10bdce f90a5c38 0ff0443d 606e6dc6 60543a49
5727c148 2be98a1d 8ab41738 20e1be24 af96da0f 68458425 99833be5 600d457d
282f9350 8334b362 d91d1120 2b6d8da0 642b1e31 9c305a00 52bce688 1b03588a
f7baefd5 4142ed9c a4315c11 83323ec5 dfef4636 a133c501 e9d3531c ee353783
```

S-Box S_4

```
9db30420 1fb6e9de a7be7bef d273a298 4a4f7bdb 64ad8c57 85510443 fa020ed1
7e287aff e60fb663 095f35a1 79ebf120 fd059d43 6497b7b1 f3641f63 241e4adf
28147f5f 4fa2b8cd c9430040 0cc32220 fdd30b30 c0a5374f 1d2d00d9 24147b15
ee4d111a 0fca5167 71ff904c 2d195ffe 1a05645f 0c13fefe 081b08ca 05170121
80530100 e83e5efe ac9af4f8 7fe72701 d2b8ee5f 06df4261 bb9e9b8a 7293ea25
ce84ffdf f5718801 3dd64b04 a26f263b 7ed48400 547eebe6 446d4ca0 6cf3d6f5
2649abdf aea0c7f5 36338cc1 503f7e93 d3772061 11b638e1 72500e03 f80eb2bb
abe0502e ec8d77de 57971e81 e14f6746 c9335400 6920318f 081dbb99 ffc304a5
4d351805 7f3d5ce3 a6c866c6 5d5bcca9 daec6fea 9f926f91 9f46222f 3991467d
a5bf6d8e 1143c44f 43958302 d0214eeb 022083b8 3fb6180c 18f8931e 281658e6
26486e3e 8bd78a70 7477e4c1 b506e07c f32d0a25 79098b02 e4eabb81 28123b23
69dead38 1574ca16 df871b62 211c40b7 a51a9ef9 0014377b 041e8ac8 09114003
```

bd59e4d2 e3d156d5 4fe876d5 2f91a340 557be8de 00eae4a7 0ce5c2ec 4db4bba6
e756bdff dd3369ac ec17b035 06572327 99afc8b0 56c8c391 6b65811c 5e146119
6e85cb75 be07c002 c2325577 893ff4ec 5bbfc92d d0ec3b25 b7801ab7 8d6d3b24
20c763ef c366a5fc 9c382880 0ace3205 aac9548a eca1d7c7 041afa32 1d16625a
6701902c 9b757a54 31d477f7 9126b031 36cc6fdb c70b8b46 d9e66a48 56e55a79
026a4ceb 52437eff 2f8f76b4 0df980a5 8674cde3 edda04eb 17a9be04 2c18f4df
b7747f9d ab2af7b4 efc34d20 2e096b7c 1741a254 e5b6a035 213d42f6 2c1c7c26
61c2f50f 6552daf9 d2c231f8 25130f69 d8167fa2 0418f2c8 001a96a6 0d1526ab
63315c21 5e0a72ec 49bafefd 187908d9 8d0dbd86 311170a7 3e9b640c cc3e10d7
d5cad3b6 0caec388 f73001e1 6c728aff 71eae2a1 1f9af36e cfcbd12f c1de8417
ac07be6b cb44a1d8 8b9b0f56 013988c3 b1c52fca b4be31cd d8782806 12a3a4e2
6f7de532 58fd7eb6 d01ee900 24adffc2 f4990fc5 9711aac5 001d7b95 82e5e7d2
109873f6 00613096 c32d9521 ada121ff 29908415 7fbb977f af9eb3db 29c9ed2a
5ce2a465 a730f32c d0aa3fe8 8a5cc091 d49e2ce7 0ce454a9 d60acd86 015f1919
77079103 dea03af6 78a8565e dee356df 21f05cbe 8b75e387 b3c50651 b8a5c3ef
d8eeb6d2 e523be77 c2154529 2f69efdf afe67afb f470c4b2 f3e0eb5b d6cc9876
39e4460c 1fda8538 1987832f ca007367 a99144f8 296b299e 492fc295 9266beab
b5676e69 9bd3ddda df7e052f db25701c 1b5e51ee f65324e6 6afce36c 0316cc04
8644213e b7dc59d0 7965291f ccd6fd43 41823979 932bcdf6 b657c34d 4edfd282
7ae5290c 3cb9536b 851e20fe 9833557e 13ecf0b0 d3ffb372 3f85c5c1 0aef7ed2

1.2 CAST-256 Notation

The following notation is employed in the specification of CAST-256.

Let f_1, f_2, f_3 be as defined for CAST-128.

Let $\beta = (ABCD)$ be a 128-bit block where $A, B, C,$ and D are each 32 bits in length.

Let "$\beta \leftarrow Q_i(\beta)$" be short-hand notation for the following:

$$
\begin{aligned}
C &= C \oplus f_1(D, k_{r_0}{}^{(i)}, k_{m_0}{}^{(i)}) \\
B &= B \oplus f_2(C, k_{r_1}{}^{(i)}, k_{m_1}{}^{(i)}) \\
A &= A \oplus f_3(B, k_{r_2}{}^{(i)}, k_{m_2}{}^{(i)}) \\
D &= D \oplus f_1(A, k_{r_3}{}^{(i)}, k_{m_3}{}^{(i)})
\end{aligned}
$$

Let "$\beta \leftarrow \overline{Q_i}(\beta)$" be short-hand notation for the following:

$$
\begin{aligned}
D &= D \oplus f_1(A, k_{r_3}{}^{(i)}, k_{m_3}{}^{(i)}) \\
A &= A \oplus f_3(B, k_{r_2}{}^{(i)}, k_{m_2}{}^{(i)}) \\
B &= B \oplus f_2(C, k_{r_1}{}^{(i)}, k_{m_1}{}^{(i)}) \\
C &= C \oplus f_1(D, k_{r_0}{}^{(i)}, k_{m_0}{}^{(i)})
\end{aligned}
$$

($Q(\cdot)$ is called a "forward quad-round" and $\overline{Q}(\cdot)$ is called a "reverse quad-round".)

Let $k_r{}^{(i)} = \{k_{r_0}{}^{(i)}, k_{r_1}{}^{(i)}, k_{r_2}{}^{(i)}, k_{r_3}{}^{(i)}\}$ be the set of rotation keys for the i^{th} quad-round, where $k_{r_j}{}^{(i)}$ is a 5-bit rotation key for $f_1, f_2,$ or f_3 (as specified above).

Let $k_m{}^{(i)} = \{k_{m_0}{}^{(i)}, k_{m_1}{}^{(i)}, k_{m_2}{}^{(i)}, k_{m_3}{}^{(i)}\}$ be the set of masking keys for the i^{th} quad-round, where $k_{m_j}{}^{(i)}$ is a 32-bit masking key for $f_1, f_2,$ or f_3 (as specified above).

CAST-256 Notation (cont'd)

Let $\kappa = (ABCDEFGH)$ be a 256-bit block where $A, B, \ldots, H$ are each 32 bits in length.

Let "$\kappa \leftarrow \omega_i(\kappa)$" be short-hand notation for the following:

$$
\begin{aligned}
G &= G \oplus f_1(H, t_{r_0}{}^{(i)}, t_{m_0}{}^{(i)}) \\
F &= F \oplus f_2(G, t_{r_1}{}^{(i)}, t_{m_1}{}^{(i)}) \\
E &= E \oplus f_3(F, t_{r_2}{}^{(i)}, t_{m_2}{}^{(i)}) \\
D &= D \oplus f_1(E, t_{r_3}{}^{(i)}, t_{m_3}{}^{(i)}) \\
C &= C \oplus f_2(D, t_{r_4}{}^{(i)}, t_{m_4}{}^{(i)}) \\
B &= B \oplus f_3(C, t_{r_5}{}^{(i)}, t_{m_5}{}^{(i)}) \\
A &= A \oplus f_1(B, t_{r_6}{}^{(i)}, t_{m_6}{}^{(i)}) \\
H &= H \oplus f_2(A, t_{r_7}{}^{(i)}, t_{m_7}{}^{(i)})
\end{aligned}
$$

($\omega(\cdot)$ is called a "forward octave".)

Let "$k_r{}^{(i)} \leftarrow \kappa$" be short-hand notation for the following:

$$k_{r_0}{}^{(i)} = 5LSB(A),\;\; k_{r_1}{}^{(i)} = 5LSB(C),\;\; k_{r_2}{}^{(i)} = 5LSB(E),\;\; k_{r_3}{}^{(i)} = 5LSB(G)$$

where $5LSB(x)$ denotes "the five least significant bits of x".

Let "$k_m{}^{(i)} \leftarrow \kappa$" be short-hand notation for the following:

$$k_{m_0}{}^{(i)} = H,\;\; k_{m_1}{}^{(i)} = F,\;\; k_{m_2}{}^{(i)} = D,\;\; k_{m_3}{}^{(i)} = B$$

1.3 The CAST-256 Cipher

$\beta =$ 128 bits of plaintext.

$$
\begin{aligned}
&for(i = 0;\ i < 6;\ i++) \\
&\qquad \beta \leftarrow Q_i(\beta) \\
&for(i = 6;\ i < 12;\ i++) \\
&\qquad \beta \leftarrow \overline{Q}_i(\beta)
\end{aligned}
$$

128 bits of ciphertext $= \beta$

Round Key Re-Ordering for Decryption

The cipher employs a 256-bit primary key K. Decryption is identical to encryption except that the sets of quad-round keys $k_r{}^{(i)}, k_m{}^{(i)}$ derived from K are used in reverse order as follows.

$$
\begin{aligned}
&for(i = 0; i < 12; i++)\{ \\
&\qquad k_{r_{new}}{}^{(i)} = k_r{}^{(11-i)} \\
&\qquad k_{m_{new}}{}^{(i)} = k_m{}^{(11-i)} \\
&\}
\end{aligned}
$$

1.4 The CAST-256 Key Schedule

Initialization:

$$c_m = 2^{30}\sqrt{2} = 5A827999_{16}$$
$$m_m = 2^{30}\sqrt{3} = 6ED9EBA1_{16}$$
$$c_r = 19$$
$$m_r = 17$$

$$
\begin{array}{l}
for(i = 0;\ i < 24;\ i++) \\
\quad for(\,j = 0;\ j < 8;\ j++)\{ \\
\quad\quad t_{m_j}{}^{(i)} = c_m \\
\quad\quad c_m = (c_m + m_m) \bmod 2^{32} \\
\quad\quad t_{r_j}{}^{(i)} = c_r \\
\quad\quad c_r = (c_r + m_r) \bmod 32 \\
\quad \}
\end{array}
$$

Key Schedule:

$\kappa = ABCDEFGH =$ 256 bits of primary key, K.

$$
\begin{array}{l}
for(i = 0;\ i < 12;\ i++)\{ \\
\quad \kappa \leftarrow \omega_{2i}(\kappa) \\
\quad \kappa \leftarrow \omega_{2i+1}(\kappa) \\
\quad k_r{}^{(i)} \leftarrow \kappa \\
\quad k_m{}^{(i)} \leftarrow \kappa \\
\}
\end{array}
$$

Note:

$$(|K| = 128) \Rightarrow (E = F = G = H = 0)$$
$$(|K| = 160) \Rightarrow (F = G = H = 0)$$
$$(|K| = 192) \Rightarrow (G = H = 0)$$
$$(|K| = 224) \Rightarrow (H = 0)$$

2. Design Rationale

2.1 Overall Structure

The fundamental mechanism for the expansion of a 64-bit block size to a larger block size is the generalization of the basic Feistel network (Schneier and Kelsey [SK96] have referred to the structure used here as an "incomplete" Feistel network). The motivation is as follows. In a traditional Feistel network (such as DES), rather than thinking of the exchange of left and right halves in each round as a "swap", it may be viewed as a circular right-shift of 32 bits. Such a view allows one to consider a cipher with a block size of 32*n* bits, which uses the same round function as the original cipher but requires *n* rounds (instead of 2) to input all bits of the block to the round function.

A picture may help to clarify the operation.

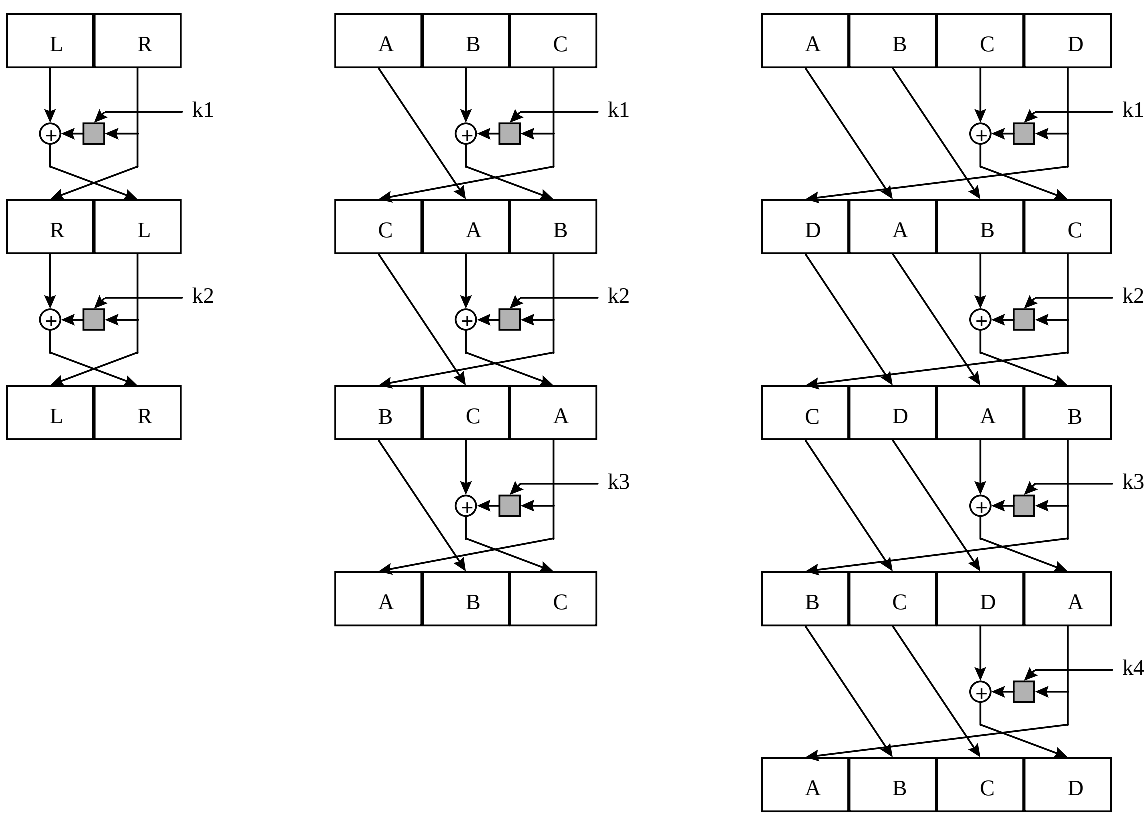

Figure 1

The left-most diagram is the "traditional" Feistel network. If this describes two rounds of DES, then *L* and *R* are each 32 bits in length and the cipher has a 64-bit block size. Continuing the illustration, the middle diagram describes an extended Feistel network for a cipher with a 96-bit block size, and the right-most diagram describes the structure of a cipher with a 128-bit block size. In each case, we may think of the number of rounds shown as a basic "unit" (in terms of submitting all input bits to the round function); the actual number of rounds chosen for the full cipher will be some multiple of this "unit" (e.g., for DES, the multiple is 8).

2.2 Decryption Considerations

The disadvantage of the generalized structure given above is that it requires a separate structure for decryption (since data must be left-shifted, rather than right-shifted, in each round in order to go backwards through the rounds). By contrast, with the "traditional" Feistel network decryption and encryption are identical except for a change in the ordering of the round keys so no separate structure is needed. Clearly, in constrained environments (such as hardware or firmware implementations that are very resource-limited) requiring two structures is unattractive.

A simple solution to the above concern is to design the structure such that if there are *r* rounds in the full cipher, the first *r*/2 rounds use right-shifting (as shown in the diagram above) and the last *r*/2 rounds use left-shifting. In this way, the desirable feature of "traditional" Feistel networks with respect to decryption (i.e., that decryption is identical to encryption, requiring only a reversal of the round keys) is preserved. This simplifies implementation and operation of the cipher and helps to make its use feasible in resource-limited environments.

2.3 Choice of Round Function

One of the very attractive features of the generalized structure given above is that it enables direct re-use of the round function from the "traditional" Feistel network. Within the class of DES-like ciphers, it is well known that increasing the size of the round function typically involves increasing the size of its component substitution boxes (s-boxes); it is also well known that increasing s-box size is generally difficult. For those ciphers that already employ large s-boxes, size increases can be a monumental task. [As a particular example, doubling the input and output sizes of a carefully-constructed 8×32 s-box may require a work factor of roughly 2^{64} steps (more than is necessary to break DES by exhaustive search!), aside from the fact that the resulting s-box grows from 4 Kbytes to more than half a million bytes of memory.] Being able to re-use the original round function is therefore very desirable. The important technical decision, however, is which "traditional" Feistel network round function to use in the generalized network.

The CAST-128 set of round functions has a number of appealing features.

- Firstly, the component bent-function-based s-boxes are designed according to a mathematical procedure which produces substitution boxes with several important cryptographic properties (such as high nonlinearity, low XOR difference distribution table values, good higher-order Strict Avalanche Criterion, and good higher-order (Output) Bit Independence Criterion) [A97b].

- Secondly, the use of both a “masking” key and a “rotation” key ensures that the key entropy is higher than the data entropy in each round (following the recommendation of [RPD97]) and appears to make the construction of iterative statistical attacks such as linear and differential cryptanalysis significantly more difficult (or impossible) [A97b].

- Thirdly, the mixing of operations from different algebraic groups (addition modulo 2 and addition / subtraction modulo 2^{32}) appears to be effective not only in reducing the probability of the round differential [AM97, O’C98], but in reducing the possibility of higher-order differential attacks as well [MSK98].

- Finally, mixing the order of the group operations (i.e., by varying the order of round functions throughout the cipher, as is done in CAST-128) appears to frustrate the practical construction of iterative characteristics.

In summary, then, the extensive analysis done on the CAST design procedure (including focused attention within several master’s- and doctoral-level theses on symmetric cipher design and analysis) lends confidence to its choice as the round function for this generalized Feistel network.

[See the attached document *CAST-256: Algorithm Analysis* for a partial list of published work which discusses or analyzes various aspects of the CAST design procedure. For one significant example of unpublished work that has been done on CAST, the Communications Security Establishment, after extensive analysis, has determined and will formally state that the CAST-128 algorithm is suitable for the protection of all levels of Designated information within the Government of Canada. Please see the attached letter dated June 5th, 1998, and note that “CAST5” is the name used for “CAST-128” when specific key lengths are explicitly intended (see [A97c], Section 2.5.]

2.4 Number of Rounds

Given that the basic unit (see "Overall Structure" above) in DES is a "double round" and that a multiple of 8 is used to give the full 16-round cipher, it is reasonable to conclude that a 128-bit block size, with a "quad-round" as the basic unit, should consist of at least 32 rounds for the full cipher. It is important to note, however, that a cipher being constructed as a candidate for AES consideration must support not only twice the block size of CAST-128, but twice the key size as well. A deeper security analysis (see attached document, *CAST-256: Algorithm Analysis*) suggests that 48 rounds (i.e., 12 "quad rounds") provides security protection commensurate with the parameters of the desired cipher.

2.5 Key Schedule

Key scheduling (deriving a set of round keys from an initial key) is an extremely important aspect of cipher design since sub-optimal key schedules can lead to exploitable weaknesses in the cipher (including weak keys, equivalent keys, complementation properties, and susceptibility to related-key attacks), and overly-complicated key schedules can lead to prohibitively-long set-up times (limiting the use of the cipher in some environments).

The design philosophy chosen for the CAST-256 key schedule is identical to that chosen for the CAST-256 cipher itself: the key schedule essentially describes a generalized Feistel network with a 256-bit block size. A simple (but fixed) set of round keys is used to key this network and the CAST-256 initial key is used as the plaintext input. Some of the output bits of selected rounds during this "encryption" define the actual round keys for the CAST-256 cipher. Important features of this key scheduling approach include the following.

- The inherent strength of the generalized Feistel network is used in the key schedule to create round keys, increasing confidence that the set of key values (comprised of the generated round keys and the CAST-256 initial key) will appear to be pair-wise independent to any statistical analysis.

- If an attack can be mounted that derives four or more full round keys (i.e., full masking keys and the corresponding rotation keys) from the CAST-256 cipher, it still appears to require a computational effort of at least $2^{256 - (4 * 32) - (4 * 5)} = 2^{108}$ guesses to derive the CAST-256 initial key from this information.

- Since the key schedule describes a generalized Feistel network, it is extremely unlikely that key collisions can occur. The key schedule defines a cipher with a fixed key (i.e., a permutation over the input space) so for two different CAST-256 initial keys to produce identical sets of round keys, the different cipher inputs would have to map to round function outputs (in every relevant round) that differed only in the 108 bits *not* used to produce round key bits. The probability of this occurring in each octave that produces round keys is $2^{108}/2^{256} = 2^{-148}$, so the probability that this occurs over the full set of round keys is $2^{-148*12} = 2^{-1776}$ (essentially zero, since there are only 2^{256} possible initial keys).

- The key scheduling operation requires the equivalent of four CAST-256 encryption operations to produce a full set of round keys. This ratio is not prohibitive for most environments and compares favorably with many current implementations of DES.

The key schedule chosen for CAST-256 appears to have a number of desirable cryptographic features and takes into account much of the research into key schedule design and analysis over the past two decades (see, for example, [A94] and the references included in [A97]).

2.6 Conclusions

A number of alternatives exist for doubling the block size of a cipher from 64 bits to 128 bits, including the following.

- Feistel network. In such a design, the round function of the Feistel network is the original 64-bit cipher, which may itself be a Feistel network (this is a simple extension of ideas presented in, for example, [LR88, L96]).

- Substitution-Permutation (SP) network [F73]. In such a design, two parallel implementations of the original cipher are used as the substitution layers; these are interspersed with an extended permutation layer (i.e., a permutation which is the width of the desired block size).

- “Fenced” Construction [R96]. In such a design, two parallel implementations of the original cipher are surrounded by specially-constructed mixing operations, which in turn are surrounded by a layer of substitution boxes.

However, it was felt that all the alternatives considered had one or more drawbacks which made them somewhat less attractive as AES submission candidates. For example, the Feistel network suffers significant security degradation if one or two rounds may be "peeled off" by some attack (not an uncommon situation) since the entire outer network would likely consist of only four or six rounds (for performance reasons). The SP network may be subject to poor encryption / decryption performance since even two substitution layers with a permutation layer in between (the minimum possible configuration) halves the speed of the original cipher; a larger number of layers decreases performance significantly beyond this. Finally, the Fenced construction has non-trivial design and implementation impacts with the need for solid theoretical justification for the particular mixing operations used and the need for sufficient processing time and memory for the pseudo-random generation and storage of the necessary s-boxes.

The approach taken in this proposal to achieve block size doubling (i.e., the use of a generalized Feistel network) appears to be the simplest and most elegant of the various alternatives. It has none of the drawbacks listed above, is straightforward to understand and to analyze, and builds on the confidence gained from the extensive literature on ciphers based on Feistel networks. Furthermore, it allows unmodified re-use of a round function with a number of attractive cryptographic features, and suggests an intuitive architecture for the associated key scheduling algorithm.

We conclude that the rationale for CAST-256 is solid, resting on firm theoretical results and immediately appealing, defensible, concepts for every aspect of the cipher design. The resulting algorithm has good performance, reasonable code and memory size, and high security (according to all analysis conducted to date); it thus appears to meet all the requirements for an AES submission candidate.

3. Bit Naming / Numbering Convention Provided

True (needed only in section *1.1 CAST-128 Notation* above, where most- to least-significant bytes of a 32-bit word are specified).

4. No Parity Bits Specified in the Key Definition

True.

5. References

[A94] C. Adams, "Simple and Effective Key Scheduling for Symmetric Ciphers", in *Workshop Record* of the Workshop on Selected Areas in Cryptography (SAC '94), Kingston, Canada, May 1994, pp.129-133.

[A97] C. Adams, "DES-80", in *Workshop Record* of the Workshop on Selected Areas in Cryptography (SAC '97), Ottawa, Canada, August 1997, pp.160-171.

[A97b] C. Adams, "Constructing Symmetric Ciphers Using the CAST Design Procedure", in *Selected Areas in Cryptography*, E. Kranakis and P. Van Oorschot (ed.), Kluwer Academic Publishers, 1997, pp.71-104 (reprinted from *Designs, Codes and Cryptography*, vol. 12, no. 3, November 1997).

[A97c] C. Adams, "The CAST-128 Encryption Algorithm", RFC 2144, May 1997.

[AM97] C. Adams and S. Mister, Preliminary experimental results concerning the mixing of operations from different algebraic groups and the contents of the resulting XOR difference distribution table (unpublished).

[F73] H. Feistel, "Cryptography and Computer Privacy", *Scientific American*, vol. 228, no. 5, 1973, pp.15-23.

[L96] S. Lucks, "Faster Luby-Rackoff Ciphers", in *Proceedings of the Third International Workshop on Fast Software Encryption*, Cambridge, UK, February 1996, Springer, LNCS 1039, pp.189-203.

[LR88] M. Luby and C. Rackoff, "How to Construct Pseudorandom Permutations From Pseudorandom Functions", *SIAM Journal of Computing*, vol. 17, no. 2, April 1988, pp.373-386.

[MSK98] S. Moriai, T. Shimoyama, and T. Kaneko, "Higher Order Differential Attack of a CAST Cipher", *Proceedings of the Fifth International Workshop on Fast Software Encryption*, Paris, France, March 1998, LNCS 1372, Springer, pp.17-31.

[O'C98] L. O'Connor, Preliminary analytical results concerning the mixing of operations from different algebraic groups and the maximum value of the resulting XOR difference distribution table (unpublished).

[R96] T. Ritter, "The Fenced DES Cipher: Stronger Than DES But Made From DES", *http://www.io.com/~ritter/FENCED.HTM*, November 10, 1996.

[RPD97] V. Rijmen, B. Preneel, and E. De Win, "On Weaknesses of Non-surjective Round Functions", in *Selected Areas in Cryptography*, E. Kranakis and P. Van Oorschot (ed.), Kluwer Academic Publishers, 1997, pp.41-54.

[SK96] B. Schneier and J. Kelsey, "Unbalanced Feistel Networks and Block Cipher Design", in *Proceedings of the Third International Workshop on Fast Software Encryption*, Cambridge, UK, February 1996, Springer, LNCS 1039, pp.121-144.

Communications Security Establishment
P.O. Box 9703
Terminal
Ottawa, Canada
K1G 3Z4

Centre de la sécurité des télécommunications
C.P. 9703
Terminus
Ottawa, Canada
K1G 3Z4

05 June 1998

Mr. Brian O'Higgins
Executive Vice President and
Chief Technology Officer
Entrust Technologies
750 Heron Road
Suite 800
Ottawa, Ontario
K1V 1A7

Dear Mr. O'Higgins,

I am very pleased to advise you that CSE has completed its evaluation of the CAST5 algorithm (80 and 128 bit versions). We have determined that CAST5 is suitable for the protection of all levels of Designated information within the GOC. A formal statement of this approval will be promulgated to Government of Canada departments and agencies in the very near future.

On behalf of the Communications Security Establishment please accept my congratulations.

David McKerrow
Communications Security Establishment
Director General Information Technology Security

CAST-256
Minimum Acceptability Requirements

1. The Algorithm Implements Symmetric (Secret) Key Cryptography

True.

2. The Algorithm is a Block Cipher

True.

3. The Algorithm Supports the Required Key-Block Size Combinations

128-128:	True.
192-128:	True.
256-128:	True.

3.1 Other Supported Key-Block Size Combinations

160-128:	True.
224-128:	True.

CAST-256
Statement of Expected Strength

1. Key Size = 128; Block Size = 128

1.1 Expected Strength

128 bits (i.e., a computational effort on the order of 2^{128} encryptions, due to the need for exhaustive search over the entire key space).

1.2 Rationale

The most powerful / successful attacks known against this cipher appear to require more plaintext than can be generated at this block size (i.e., more than 2^{128} plaintext values), and so cannot be mounted. See the *CAST-256: Algorithm Analysis* section of this submission package for details.

2. Key Size = 192; Block Size = 128

2.1 Expected Strength

192 bits (i.e., a computational effort on the order of 2^{192} encryptions, due to the need for exhaustive search over the entire key space).

2.2 Rationale

The most powerful / successful attacks known against this cipher appear to require more plaintext than can be generated at this block size (i.e., more than 2^{128} plaintext values), and so cannot be mounted. See the *CAST-256: Algorithm Analysis* section of this submission package for details.

3. Key Size = 256; Block Size = 128

3.1 Expected Strength

256 bits (i.e., a computational effort on the order of 2^{256} encryptions, due to the need for exhaustive search over the entire key space).

3.2 Rationale

The most powerful / successful attacks known against this cipher appear to require more plaintext than can be generated at this block size (i.e., more than 2^{128} plaintext values), and so cannot be mounted. See the *CAST-256: Algorithm Analysis* section of this submission package for details.

4. Key Size = 160; Block Size = 128

4.1 Expected Strength

160 bits (i.e., a computational effort on the order of 2^{160} encryptions, due to the need for exhaustive search over the entire key space).

4.2 Rationale

The most powerful / successful attacks known against this cipher appear to require more plaintext than can be generated at this block size (i.e., more than 2^{128} plaintext values), and so cannot be mounted. See the *CAST-256: Algorithm Analysis* section of this submission package for details.

5. Key Size = 224; Block Size = 128

5.1 Expected Strength

224 bits (i.e., a computational effort on the order of 2^{224} encryptions, due to the need for exhaustive search over the entire key space).

5.2 Rationale

The most powerful / successful attacks known against this cipher appear to require more plaintext than can be generated at this block size (i.e., more than 2^{128} plaintext values), and so cannot be mounted. See the *CAST-256: Algorithm Analysis* section of this submission package for details.

6. Additional Notes Regarding Expected Strength

As noted in the *CAST-256: Algorithm Analysis* section of this submission package (item *1.1 Ciphertext Only Attack*), an attack based upon the birthday paradox can be mounted against any block cipher when used in CBC mode. This means that a cipher with a 128-bit block size should not be used to encrypt more than 2^{64} blocks of data with a single key. Note, however, that this attack recovers a relationship between a given pair of plaintext blocks; it does not recover all plaintext and does not recover any key bits.

Furthermore, Biham [B96] has described so-called *key-collision attacks* and shown that the strength of a cipher cannot exceed the square root of the size of the key space. Although this is strictly a theoretical attack and would be highly impractical to mount in a real-world situation for the key sizes required for AES, it does serve to put a bound on the expected theoretical strength of any block cipher.

[B96] E. Biham, "How to Forge DES-Encrypted Messages in 2^{28} Steps", *Technical Report CS 884*, Department of Computer Science, Technion, Haifa, Israel, August 1996 (see also http://www.cs.technion.ac.il/~biham/publications.html)

CAST-256
Algorithm Analysis

1. Analysis With Respect to Known Attacks

The classical attacks on ciphers are as follows: *ciphertext only*; *known plaintext*; and *chosen plaintext*. The advent of public-key cryptography added utility to the concept of a *chosen ciphertext* attack, but this appears to be of little added value in the analysis of symmetric ciphers. Research in the past decade or so has also introduced the notions of *chosen key* and *related key* attacks, which have enjoyed some success in the cryptanalysis of specific symmetric ciphers. Within the iterated symmetric ciphers (the class of algorithms to which CAST-256 belongs), the techniques known as *linear cryptanalysis* and *differential cryptanalysis* (along with their combinations and higher-orders) currently represent the most general and powerful instances of *known plaintext* and *chosen plaintext* attacks, respectively.

This section of the submission package examines the CAST-256 algorithm with respect to the families of cryptanalytic attack listed above.

1.1 Ciphertext Only Attack

No techniques are currently known that will allow an attacker to infer or derive information about the plaintext, the primary key, or any subset of round keys from any collection of ciphertext blocks. The one (unavoidable) exception to this is the technique applicable to all n-bit-block ciphers when used in Cipher-Block-Chaining (CBC) mode: once $2^{n/2}$ blocks have been encrypted, with probability roughly ½ (rapidly increasing as more blocks are encrypted) an XOR relationship between a particular pair of plaintexts will be known.

1.2 Known Plaintext Attack: Linear Cryptanalysis

Linear cryptanalysis [M94] attempts to exploit any high-probability occurrences of linear expressions of input, output, and round key bits in the round function of an iterated cipher. It has been approximated [M94] that the best linear expression for r-rounds of a cipher has a probability of being satisfied that is bounded as follows:

$$\left| p_L - \tfrac{1}{2} \right| \leq 2^{\alpha-1} \cdot \left| p_\beta - \tfrac{1}{2} \right|^{\alpha}$$

where p_L represents the probability that the linear expression holds, p_β represents the probability of the best linear approximation, and α represents the number of s-boxes involved in that linear approximation. This expression is based on the assumption of independent round keys such that the linear approximations of the s-boxes are independent. In an analogous way to "differentials" and "characteristics" in differential cryptanalysis, provable immunity in linear cryptanalysis relies on bounding the likelihood of an overall linear expression (sometimes referred to as the "linear hull") rather than any particular linear "characteristic". However, for many ciphers (including CAST-256) this is a difficult analytical task. What are typically considered, therefore, are the building blocks of an overall linear expression: the sequence of approximations of the round functions which result in the overall linear expression.

A basic linear attack typically uses a sequence of linear approximations of the rounds to create an overall linear expression involving subsets of plaintext and ciphertext bits. From this it is possible to derive the equivalent of one key bit represented as the XOR of a number of round key bits. In this case, it is shown [M94] that the number of known plaintexts required is approximately

$$N_L = \left| p_L - \tfrac{1}{2} \right|^{-2}.$$

It can be shown that the best linear approximation has a probability given by

$$\left| p_\beta - \tfrac{1}{2} \right| = \frac{2^{m-1} - NL_{\min}}{2^m}$$

where m is the number of input bits to the s-box and $NL_{\min}$ is the nonlinearity of the s-box [LHT97]. For the s-boxes of CAST-256, $m = 8$ and $NL_{\min} = 74$. Furthermore, for the CAST-256 cipher, the best linear approximation appears to involve 4 s-boxes every 4 rounds such that the linear approximation of the round function for every 4th round involves no output bits. That is, the linear expression used is $X_{i_1} \oplus X_{i_2} \oplus \ldots \oplus X_{i_t}$, where X_{i_j} represents an input bit to the s-box. Hence, for an r-round linear approximation, $\alpha = r$. The number of known plaintexts required for a 48-round linear approximation of CAST-256, then, is approximately 2^{122}. Note that this is almost equal to the total number of plaintexts available (2^{128}) and argues against the practicality of a linear attack on this cipher.

Furthermore, Youssef, *et al*, have proposed [YCT97] that a more accurate bound on the number of plaintexts required for linear cryptanalysis of a CAST cipher can be obtained by considering the combination of s-boxes in the round function rather than the individual s-boxes. In particular, they compute the value for NL_S, the nonlinearity of the composite 32×32 s-box when the individual 8×32 s-boxes are combined using XOR. Using this in place of $NL_{\min}$ in the equations above and setting $m = 32$ and $\alpha = r/2$ (since an r-round linear approximation must involve at least as many 32×32 s-boxes as $r/2$ iterations of the best 2-round approximation) yields a number of known plaintexts required for a 48-round linear approximation at more than 2^{174} (far beyond the number of plaintexts available). Note that experimental evidence suggests that combining s-boxes using mixed operations may increase the nonlinearity of the composite s-box even further.

It therefore appears that CAST-256 is immune to a linear cryptanalysis attack.

1.3 Chosen Plaintext Attack: Differential Cryptanalysis

Differential cryptanalysis [BS93] attempts to exploit any high-probability output differences resulting from particular input differences in the round function of an iterated cipher. A block cipher can be proved to be resistant to differential cryptanalysis if it can be shown that no high-probability differentials [LMM91] exist, where an *i*-round differential is defined to be the XOR of two outputs after *i* rounds, where the outputs correspond to two plaintexts with a given XOR.

In a good cipher the probability of all differentials should approach 2^{-N}, where N is the block size. Strictly speaking, differential cryptanalysis requires only the existence of a highly-probable differential to succeed. However, differentials can be viewed to be comprised of a number of possible characteristics, where a characteristic specifies the exact sequence of input and output XORs for each round to achieve the overall differential input and output XOR.

It is typically difficult to derive the probability of any particular differential and, in practice, it would be hard for a cryptanalyst to determine the existence of a highly-probable differential without searching for highly-probable characteristics. Although it is often the case that an upper bound on the probability of a differential cannot be stated for a particular cipher (that is, resistance to a differential cryptanalytic attack cannot be proved), the probabilities of the most likely characteristics can be determined. These probabilities can then be used as a measure of the cipher's resistance to differential cryptanalysis.

As is common in the literature, the analysis here is based on the assumption that all round keys are independent (although this assumption is not always necessary; see [C97]) and that the occurrence of output XORs given particular input XORs is independent for different rounds. Under such conditions, the probability of an *r*-round characteristic is given by

$$p_{\Omega_r} = \prod_{i=1}^{r} p_i$$

where p_i represents the probability of the output XOR given the input XOR in round *i*. The best characteristics that can be constructed are typically iterative in nature. For the CAST-256 cipher with *R* rounds, the following appears to be the best possible *r*-round characteristic, where *r* is a multiple of 4. (Note that the notation (*W*,*X*,*Y*,*Z*) represents XOR vectors for the four 32-bit sub-blocks in a CAST-256 round function input.)

$(0,0,0,\Delta)$	[input XOR to round 1]
$0 \leftarrow \Delta$ with probability p	[round 1]
$0 \leftarrow 0$ with probability 1	[round 2]
$0 \leftarrow 0$ with probability 1	[round 3]
$0 \leftarrow 0$ with probability 1	[round 4]
...	repeat up to *R*/2 rounds
$(0,\Delta,0,0)$, or some variation	[input XOR to round (*R*/2 + 1)]
$0 \leftarrow 0$ with probability 1	[round (*R*/2 + 1)]
$0 \leftarrow 0$ with probability 1	[round (*R*/2 + 2)]
$0 \leftarrow \Delta$ with probability p	[round (*R*/2 + 3)]
$0 \leftarrow 0$ with probability 1	[round (*R*/2 + 4)]
...	repeat up to *r* rounds for *r*-round char.

The input XOR to round (*R*/2 +1) will be a vector in which one of the sub-blocks is non-zero and the other three sub-blocks are zero (the precise variation which applies for a given cipher depends upon the value of *R*). Without loss of generality, the example $(0,\Delta,0,0)$ is shown above.

As per the analysis and rationale given in [LHT97], the input-output XOR pair for a simplified CAST round function (i.e., one which does not include the key-dependent rotation, and for which the only s-box combining operation used is XOR) can be assumed to have a probability of $p \le 2^{-14}$. This is based on the fact that all four s-boxes in the CAST round function are injective and the format of the XOR pair has the output XOR being equal to 0. This leads to the conclusion that the best *r*-round iterated characteristic as shown above has a probability given by

$$p_{\Omega_r} \le (2^{-14})^{r/4}$$

In particular, a 40-round characteristic must have a probability less than or equal to 2^{-140} according to the assumptions of the analysis. This implies that the number of chosen plaintexts required for this attack would be greater than 2^{140} for the 48-round cipher (substantially greater than the number of plaintexts available for a 128-bit block size).

It therefore appears that CAST-256 is immune to a differential cryptanalysis attack.

1.4 Chosen Key Attack

CAST-256 appears to be secure with respect to this attack. The use of a cipher (built around the CAST-128 set of round functions) as a key schedule gives confidence that no exploitable statistical correlation exists between the primary key and the set of generated round keys. Thus, allowing an attacker to choose a particular primary key difference appears to yield no exploitable similarities in the corresponding sets of round keys compared with the victim encrypting with two randomly-chosen primary keys.

1.5 Related Key Attack

CAST-256 appears to be secure with respect to this attack. The use of a cipher (built around the CAST-128 set of round functions) as a key schedule gives confidence that no exploitable statistical correlations exist within the set of generated round keys. Thus, this attack, which depends upon the use of a simple derivation algorithm for a round key from previous round keys, appears not to be applicable to CAST-256.

1.6 Enhancements to the Above Statistical Attacks: Combinations and Higher-Orders

The analysis given above for both linear and differential cryptanalysis applies to a greatly simplified version of the CAST-256 cipher. The actual cipher, which includes key-dependent rotation and mixed operations in the round function (both for data masking and for s-box combination), appears to be much more difficult / impossible to attack using the methods as described in [M94] and [BS93] (see [A97] for some discussion of this). In particular, experiments in which two CAST-256 s-boxes are combined using addition or subtraction modulo 2^{32} show that the maximum value in the XOR difference distribution table is approximately 10% of the maximum that occurs when the s-boxes are combined using XOR. Experiments on combinations of three CAST-256 s-boxes are on-going, but thus far show similar results. This lends confidence that combinations of four s-boxes using mixed operations (as is done in the CAST-256 round function) are effective in increasing resistance to differential cryptanalysis.

The above experimental work [AM97] is supported by a new analytical result [O'C98], which shows that for a random n-bit permutation, the probability that the maximum entry in a differential table based on XOR differences is greater than a bound B_n approaches 1 as n grows, whereas the probability that the maximum entry in a table based on non-XOR differences (e.g., modular addition or multiplication) is greater than that same bound approaches 0. Furthermore, the bound is accurate for the 8-bit case. Thus, although the details of the analyzed structure differ slightly from the internals of the CAST-256 round function as used in the above experiments, the conclusion is the same: using operations from different algebraic groups appears to be helpful in increasing resistance to differential cryptanalysis (by lowering the differential probability of a single round).

1.6.1 Combination Attacks

CAST-256 appears to be immune to both linear and differential cryptanalysis (requiring more plaintext than is available from the 128-bit block size) and appears to be immune to both chosen and related key attacks (due to the absence of exploitable statistical correlations among its generated keys). Given this, it seems highly unlikely that various combination attacks (such as *linear-differential*, or *differential-related-key*) can have any measure of success.

It therefore appears that this cipher is immune to the combination attacks currently known in the literature.

1.6.2 Higher-Order Attacks

The concept of *higher-order differentials* has been introduced [L94, K95] and used to successfully cryptanalyze ciphers proved secure against ordinary differential cryptanalysis [JK97]. A simplified version of the CAST-128 cipher (one which uses XOR for all operations in the round function) has been examined with respect to the higher-order differential attack [MSK98]. It has been shown that this attack is successful up to 5 rounds, but cannot be extended to higher numbers of rounds. Furthermore, the introduction of the key-dependent rotation operation is effective in increasing the computational complexity of this attack. Finally, the use of operations from different algebraic groups "makes the degree too high to cryptanalyze by the higher-order differential attack" [MSK98], so that the attack cannot even be mounted on a 5-round version of the cipher.

It therefore appears that CAST-256 (which has 48 rounds and uses the CAST-128 round functions) is immune to a higher-order differential attack.

2. Statements Regarding Properties of Keys

This section provides statements regarding the following properties of keys with respect to CAST-256: *weak keys*, *semi-weak keys*, *fixed points of a key*, *equivalent keys*, and *restrictions on key selection*. It also includes a statement on *complementation properties* since this is sometimes related to the way that round keys are used within a DES-like cipher.

2.1 Weak Keys

None known. In the CAST-256 cipher, all keys appear to be of equivalent strength and are usable for double encryption (i.e., no key appears to be its own inverse).

2.2 Semi-Weak Keys

None known. In the CAST-256 cipher, there appear to be no pairs of keys which cannot be used for double encryption (i.e., there do not appear to be pairs of keys k_i and k_j such that k_j is the inverse of k_i).

2.3 Fixed Points of a key K

None known. From all evidence available thus far in the open literature, fixed points have only been easily found (i.e., requiring a level of effort for an *n*-bit block cipher of roughly $2^{n/2}$ operations rather than 2^n operations) in DES-like ciphers for weak and semi-weak keys. It therefore appears that CAST-256 has no easily-found fixed points for any key.

2.4 Equivalent Keys

None known. The key schedule defines a cipher with a fixed key (i.e., a permutation over the input space) so for two different CAST-256 initial keys to produce identical sets of round keys, the different cipher inputs would have to map to round function outputs (in every relevant round) that differed only in the 108 bits *not* used to produce round key bits. The probability of this occurring in each octave that produces round keys is $2^{108}/2^{256} = 2^{-148}$, so the probability that this occurs over the full set of round keys is $2^{-148*12} = 2^{-1776}$ (essentially zero, since there are only 2^{256} possible initial keys).

2.5 Restrictions on Key Selection

None known. The key scheduling algorithm defines a symmetric block cipher with a fixed key where the CAST-256 primary key is used as the plaintext input. Because in this symmetric block cipher there are no restrictions on the input space (i.e., any plaintext can be encrypted), it follows that no restrictions are placed upon selection of CAST-256 primary keys.

2.6 Complementation Properties

None known. There appear to be no complementations of combinations of plaintext, key, and ciphertext that lead to identities. This is due to the complexity of the key scheduling operation (so that complementing the primary key leads to random-looking changes to all round keys) and also to the use of multiple operations to combine data, key, and s-boxes within the round functions (XOR, rotation, and addition and subtraction modulo 2^{32}).

3. Statement Regarding Trap-Doors

None known. There are several reasons to feel confident that there are no trap-doors in this cipher.

- CAST-256 uses the four round function s-boxes in CAST-128. The design criteria and construction procedure for these s-boxes have been published [A97, MA96] and the specific s-boxes themselves have been examined by a number of researchers.
- CAST-256 uses the three round functions in CAST-128. The design criteria for these round functions have been published [A97] and the specific round functions themselves have been examined by a number of researchers. Furthermore, the complexity introduced by the mixed operations in the round functions would seem to make it difficult to insert a trap-door of any kind.
- CAST-256 uses 48 rounds. Inserting a non-obvious trap-door that will carry through 48 rounds of the cipher would seem to be a formidable task.
- CAST-256 uses a significantly more complex key scheduling algorithm than DES. A trap-door in the final round that allows the attacker (i.e., the one knowing this trap-door) to recover information about the final round key will be of little help in deriving either other round keys or the primary key. This contrasts with DES in which knowledge of any round key gives knowledge of the primary key with only a brute-force search over 8 bits of key.

4. Publications Discussing or Analyzing Aspects of the CAST Design Procedure

C. Adams and S. Tavares, "The Use of Bent Sequences to Achieve Higher-Order Strict Avalanche Criterion in S-Box Design", *Technical Report TR 90-013*, Department of Electrical Engineering, Queen's University, Kingston, Ontario, January 1990.

C. Adams, "A Formal and Practical Design Procedure for Substitution-Permutation network Cryptosystems", *Ph.D. Thesis*, Dept. of Electrical Engineering, Queen's University, Kingston, Ontario, Canada, September, 1990.

C. Adams and S. Tavares, "The Structured Design of Cryptographically Good S-Boxes", *Journal of Cryptology*, vol. 3, no. 1, 1990, pp.27-41.

C. Adams and S. Tavares, "Designing S-Boxes for Ciphers Resistant to Differential Cryptanalysis", *Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography*, Rome, Italy, 1993, pp.181-190.

C. Adams, "Simple and Effective Key Scheduling for Symmetric Ciphers", Workshop on Selected Areas in Cryptography, SAC' 94, *Workshop Record*, 1994, pp.129-133.

C. Adams, "Designing DES-like Ciphers with Guaranteed Resistance to Differential and Linear Attacks", Workshop on Selected Areas in Cryptography, SAC' 95, *Workshop Record*, 1995, pp.133-144.

C. Adams, "Constructing Symmetric Ciphers Using the CAST Design Procedure", *Designs, Codes, and Cryptography*, vol. 12, no. 3, 1997, pp.283-316.

H. Heys and S. Tavares, "On the Security of the CAST Encryption Algorithm", *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, Halifax, NS, Canada, September 1994, pp.332-335.

J. Lee, "An investigation of some security aspects of the CAST encryption algorithm", *M.Sc. Thesis*, Dept. of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada, 1995.

J. Lee, H. Heys, and S. Tavares, "Resistance of a CAST-like Encryption Algorithm to Linear and Differential Cryptanalysis", *Designs, Codes and Cryptography*, vol. 12, no. 3, 1997, pp.267-282.

A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997, p.281.

S. Mister and C. Adams, “Practical S-Box Design”, Workshop on Selected Areas in Cryptography, SAC ’96, *Workshop Record*, 1996, pp.61-76.

S. Moriai, T. Shimoyama, and T. Kaneko, “Higher Order Differential Attack of a CAST Cipher”, *Proceedings of the Fifth International Workshop on Fast Software Encryption*, Paris, France, March 1998, LNCS 1372, Springer, pp.17-31.

V. Rijmen, B. Preneel and E. De Win “On weaknesses of non-surjective round functions”, *Designs, Codes and Cryptography*, vol. 12, no. 3, 1997, pp.253-266.

B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C (2nd edition)*, John Wiley & Sons, 1996, pp.334-335.

W. Stallings, *Cryptography and Network Security: Principles and Practice, 2nd Edition*, Prentice Hall, 1998 (to appear).

A. Youssef, S. Tavares, S. Mister and C. Adams, “Linear approximation of Injective S-boxes”, *IEE Electronics Letters*, vol.31, no. 25, 1995, pp.2168-2169.

A. Youssef, Z. Chen and S. Tavares, “Construction of Highly Nonlinear Injective S-boxes with Application to CAST-like Encryption Algorithm”, *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE’ 97)*, St. John’s, NF, Canada, May 1997, pp.330-333.

A. Youssef, “Analysis and Design of Block Ciphers”, *Ph.D. Thesis*, Dept. of Electrical and Computer Engineering, Queen’s University, Kingston, Canada, 1997.

X. Zhu, “A New Class of Unbalanced CAST Ciphers and Its Security Analysis”, *M.Sc. Thesis*, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, 1997.

X. Zhu and H. Heys, “The Analysis of a New Class of Unbalanced CAST Ciphers”, *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE’ 97)*, St. John’s, NF, Canada, May 1997, pp.326-329.

[Please note that a number of the above papers are available at the following location:
http://adonis.ee.queensu.ca:8000/cast/]

5. References

[A97] C. Adams, “Constructing Symmetric Ciphers Using the CAST Design Procedure”, in *Selected Areas in Cryptography*, E. Kranakis and P. Van Oorschot (ed.), Kluwer Academic Publishers, 1997, pp.71-104.

[AM97] C. Adams and S. Mister, Preliminary experimental results concerning the mixing of operations from different algebraic groups and the contents of the resulting XOR difference distribution table (unpublished).

[BS93] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer, 1993.

[C97] A. Canteaut, “Differential Cryptanalysis of Feistel Ciphers and Differentially δ-uniform Mappings”, Workshop on Selected Areas in Cryptography, SAC ’97, *Workshop Record*, 1997, pp.172-184.

[JK97] T. Jakobsen and L. Knudsen, “The Interpolation Attack on Block Ciphers”, *Proceedings of the Fourth International Workshop on Fast Software Encryption*, Haifa, Israel, January 1997, LNCS 1267, Springer, pp.28-40.

[K95] L. Knudsen, “Truncated and Higher Order Differentials”, *Proceedings of the Second International Workshop on Fast Software Encryption*, Leuven, Belgium, 1995, LNCS 1008, Springer, pp.196-211.

[L94] X. Lai, “Higher Order Derivatives and Differential Cryptanalysis”, in *Proceedings of the Symposium on Communication, Coding and Cryptography, in honor of James L. Massey on the occasion of his 60th birthday*, Feb. 10-13, 1994, Monte-Verita, Ascona, Switzerland, 1994.

[LMM91] X. Lai, J. Massey, and S. Murphy, “Markov Ciphers and Differential Cryptanalysis”, *Advances in Cryptology: Proceedings of Eurocrypt ’91*, Springer, 1991, pp.17-38.

[LHT97] J. Lee, H. Heys, and S. Tavares, “Resistance of a CAST-like Encryption Algorithm to Linear and Differential Cryptanalysis”, *Designs, Codes and Cryptography*, vol. 12, no. 3, 1997, pp.267-282.

[M94] M. Matsui, “Linear Cryptanalysis Method for DES Cipher”, *Advances in Cryptology: Proceedings of Eurocrypt ’93*, Springer, 1994, pp.386-397.

[MA96] S. Mister and C. Adams, “Practical S-Box Design”, Workshop on Selected Areas in Cryptography, SAC ’96, *Workshop Record*, 1996, pp.61-76.

[MSK98] S. Moriai, T. Shimoyama, and T. Kaneko, "Higher Order Differential Attack of a CAST Cipher", *Proceedings of the Fifth International Workshop on Fast Software Encryption*, Paris, France, March 1998, LNCS 1372, Springer, pp.17-31.

[O'C98] L. O'Connor, Preliminary analytical results concerning the mixing of operations from different algebraic groups and the maximum value of the resulting XOR difference distribution table (unpublished).

[YCT97] A. Youssef, Z. Chen and S. Tavares, "Construction of Highly Nonlinear Injective S-boxes with Application to CAST-like Encryption Algorithm", *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE' 97)*, St. John's, NF, Canada, May 1997, pp.330-333.

CAST-256
Computational Efficiency

1. Efficiency Estimates for the NIST AES Analysis Platform

1.1 Platform Description

IBM-compatible PC, with an Intel Pentium Pro Processor, 200MHz clock speed, 64MB RAM, running Windows95.

1.2 Speed Estimates (in clock cycles)

Operation	128/128	192/128	256/128
Encrypt one data block:	1790	1790	1790
Decrypt one data block:	1790	1790	1790
Key setup:	9090	9090	9090
Algorithm setup:	0	0	0
Key change:	9090	9090	9090

1.3 Tradeoffs Between Speed and Memory

For environments in which memory is not a scarce resource, s-boxes S_1 and S_2 can be combined into three 16×32 s-boxes (one corresponding to $S_1 \oplus S_2$, one corresponding to S_1-S_2, and one corresponding to S_1+S_2, for each of the three round function types). This saves one memory lookup and combining operation per round, which will result in a modest performance increase.

2. Efficiency Estimates for 8-Bit Processors

2.1 Platform Description

Motorola 6811 microprocessor, 2MHz clock speed, assembly language implementation.

2.2 Speed Estimates (in clock cycles)

Operation	128/128	192/128	256/128
Encrypt one data block:	26000	26000	26000
Decrypt one data block:	26000	26000	26000
Key setup:	110000	110000	110000
Algorithm setup:	0 ms	0 ms	0 ms
Key change:	110000	110000	110000

2.3 Tradeoffs Between Speed and Memory

None known.

3. Efficiency Estimates for Other Platforms

3.1 Platform Description

IBM-compatible PC, with an Intel Pentium II Processor, 300MHz clock speed, 128MB RAM, running Windows NT 4.0, assembly language implementation.

3.2 Speed Estimates (in clock cycles)

Operation	128/128	192/128	256/128
Encrypt one data block:	815	815	815
Decrypt one data block:	815	815	815
Key setup:	4130	4130	4130
Algorithm setup:	0	0	0
Key change:	4130	4130	4130

3.3 Tradeoffs Between Speed and Memory

For environments in which memory is not a scarce resource, s-boxes S_1 and S_2 can be combined into three 16×32 s-boxes (one corresponding to $S_1 \oplus S_2$, one corresponding to $S_1 - S_2$, and one corresponding to $S_1 + S_2$, for each of the three round function types). This saves one memory lookup and combining operation per round, which will result in a modest performance increase.

4. Efficiency Estimates for Other Platforms

4.1 Platform Description

Sun UltraSparc 1, 167MHz clock speed, 124MB RAM, running Solaris 2.5.

4.2 Speed Estimates (in clock cycles)

Operation	128/128	192/128	256/128
Encrypt one data block:	1180	1180	1180
Decrypt one data block:	1180	1180	1180
Key setup:	5830	5830	5830
Algorithm setup:	0	0	0
Key change:	5830	5830	5830

4.3 Tradeoffs Between Speed and Memory

For environments in which memory is not a scarce resource, s-boxes S_1 and S_2 can be combined into three 16×32 s-boxes (one corresponding to $S_1 \oplus S_2$, one corresponding to S_1-S_2, and one corresponding to S_1+S_2, for each of the three round function types). This saves one memory lookup and combining operation per round, which will result in a modest performance increase.

5. General Efficiency Comments

As will be noted in the tables given above, CAST-256 has the following features:

- it requires no algorithm setup time (e.g., there is no need to generate s-boxes or other tables, and no need to pre-compute values);
- decryption performance is identical to encryption performance;
- key change time is identical to key setup time;
- there is no penalty for key size differences (i.e., encryption / decryption performance and key setup performance are unaffected by whether the primary key is 128 bits, 256 bits, or a value in between).

CAST-256
Algorithm Advantages and Limitations

1. Advantages

The CAST-256 cipher has a number of advantages compared with other algorithms found in the open literature, including the following.

- Speed: the cipher has very good encryption / decryption performance and an acceptable key set-up time for most environments.
- History: the s-boxes and round functions have been examined in detail by a number of cryptographers and cryptanalysts in the context of CAST-128.
- Simplicity: the operations used in the cipher (XOR, addition, subtraction, rotation) are all simple, available, and fast on typical computing platforms.
- Identical Operation: encryption and decryption are identical operations, requiring a simple reversal in the order of the round keys.
- Fixed Speed: the encryption / decryption speed is unaffected by a change in key size (from 128 bits to 256 bits).
- Secure: quite conservative analysis indicates that the cipher is as strong as its key size.

2. Limitations

For some specific environments, the following may be seen as limitations of the CAST-256 cipher.

- Memory: the 4 Kilobytes of total storage required for the CAST-256 s-boxes may be too high for some environments with very constrained resources.
- Key Set-Up Time: the time to generate the set of round keys from the primary key (four times the time required to encrypt a single block of data), although comparable to DES, may be too slow for some very high-speed environments that need to change keys very frequently.
- Rotation Operation: the key-dependent rotation operation in the CAST-256 round function may be too slow for some environments that cannot do a multi-bit rotation in a single machine instruction.

3. Implementation for Various Purposes

3.1 Stream Cipher

No limitations known (when run in any of the various approved stream cipher modes of operation, such as OFB-*n* and CFB-*n*). This may, however, be slower than an algorithm designed specifically as a stream cipher (rather than a block cipher).

3.2 MAC Generator

No limitations known (when run in any of the various approved modes of operation for Message Authentication Codes). It is possible, however, that any block cipher used as a MAC generator may be less secure than an algorithm designed specifically for MAC generation (such as HMAC).

3.3 Pseudo-Random Number Generator

No limitations known (when run in any of the various approved PRNG modes of operation, such as a feedback mode). It is noted, however, that it is typically very difficult to prove anything formally about the security of the resulting PRNG (in contrast, for example, with some of the literature on provably-secure pseudo-random number generators).

3.4 Hashing Algorithm

No limitations known (when run in any of the various approved hashing modes of operation). This will, however, typically be slower than an algorithm designed specifically as a hashing algorithm, such as SHA-1 or RIPEMD-160.

4. Implementation in Various Environments

4.1 Smart Cards, 8-Bit Processors

The size of the s-boxes (4 Kilobytes in total) may be too large for some very constrained current environments. However, this problem is expected to diminish/disappear in the next generation or two of these processors as Moore's Law continues to operate.

4.2 ATM

No limitations known.

4.3 HDTV

No limitations known.

4.4 B-ISDN

No limitations known.

4.5 Voice Applications

No limitations known.

4.6 Satellite Applications

No limitations known.

5. Other Key and Block Sizes

Aside from the required 128-bit block size and the 128-, 192-, and 256-bit key sizes, the CAST-256 cipher supports key sizes of 160 bits and 224 bits.

Furthermore (although this is not specified explicitly in this submission), the framework of the cipher is flexible and can support other block sizes in multiples of 32. For example, it would be possible to specify parameters (e.g., number of rounds and appropriate key sizes) for block sizes of 96, 160, 192, 224, and 256 bits.

6. Other Advantageous Features

The method used to double the block size of CAST-128 (the extended Feistel framework) is intuitively appealing. It is a clear and simple extension of the basic Feistel network with which most researchers in the field are familiar. This network — popularized by, and intensely studied as, the foundation of the Data Encryption Standard — has come to be trusted by many as a secure structure and forms the basis for a number of cryptographic algorithms in the open literature. The extension builds upon this trust in a natural and readily-analyzable way.