SIMON and SPECK

Block ciphers for the IoT

Ray Beaulieu · Doug Shors* · Jason Smith
Stefan Treatman-Clark · Bryan Weeks · Louis Wingers

National Security Agency
Trusted Systems Research Group

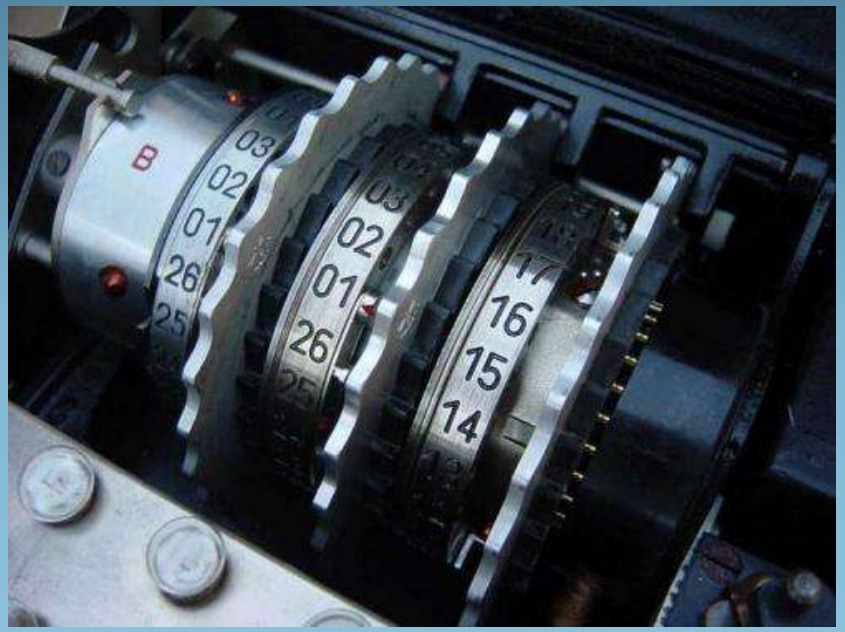
NIST Lightweight Cryptography Workshop
20-21 July 2015

Overview

- Introduction
- The algorithms
- Security
- Interlude: SIMON and SPECK vs. AES, a visual comparison
- Performance

Introduction

Cryptography is implemented on physical devices.



"Enigma-rotor-stack". Wikimedia Commons

Introduction

Much of the cryptography we use today was designed with desktop processors in mind.

"Apple II plus." Wikimedia Commons. Original uploader Hellis it.wikipedia.

Fast forward

We are entering a world of *ubiquitous computing.*

- tight constraints on resources (area, power, energy,…)
- highly interconnected heterogeneous networks
- serious security challenges

dreamstime.com

What should crypto look like to work well in this world?

NSA Trusted Systems Research Group

Specialists and generalists

The koala — a specialist; diet consists almost entirely of eucalyptus leaves.

The American crow — a highly adaptable generalist.

Specialists and generalists

Specialists require stable environments; generalists are better able to adapt to changing environments.

Question: What kind of *cryptographic environment* should we prepare for in the IoT era?

Specialists and generalists

Many of the cryptographic algorithms proposed for this world are *koalas*, highly optimized for particular platforms or use cases.

Performance in other use cases or on other platforms might be ignored, or may be treated as an afterthought. Often this *untargeted* performance is poor.

We believe this represents a misreading of future needs.

Specialists and generalists

Our proposals, SIMON and SPECK, are *crows*. They're designed with simplicity and flexibility in mind. Because of this, they offer high performance on a large range of existing platforms, and in many use cases.

We believe they are best positioned to provide high performance on *future* constrained platforms, whatever they may be.

Our view

In the rapidly evolving IoT world, the most useful cryptographic algorithms most likely will not be those that are tailored for specific current platforms or use cases!

In the words of Donald Knuth (taken out of context!),

Premature optimization is the root of all evil.

Our advice

Be skeptical!

- Question premises: Is x really what lightweight crypto is all about? How will algorithm y perform for *other* uses?
- Question comparisons: Are the authors comparing their algorithms to the *best* implementations of the competing algorithms?

SIMON and SPECK

We're now ready to discuss our algorithms.

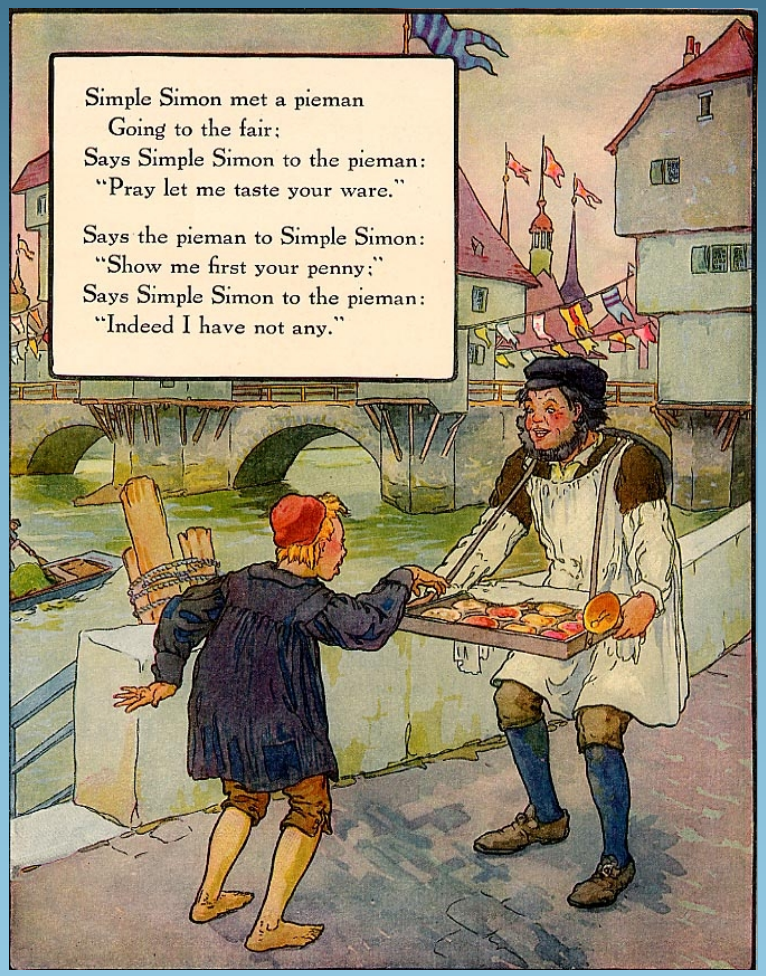
SIMON and SPECK

SIMON and SPECK are families of lightweight block ciphers, each with 10 block/key sizes, ranging from 32/64 to 128/256.

They have high performance on ASICs, FPGAs, microcontrollers, and microprocessors.

They support a range of implementations from very compact to very high throughput / low latency.

SIMON



Versatile in hardware and software.

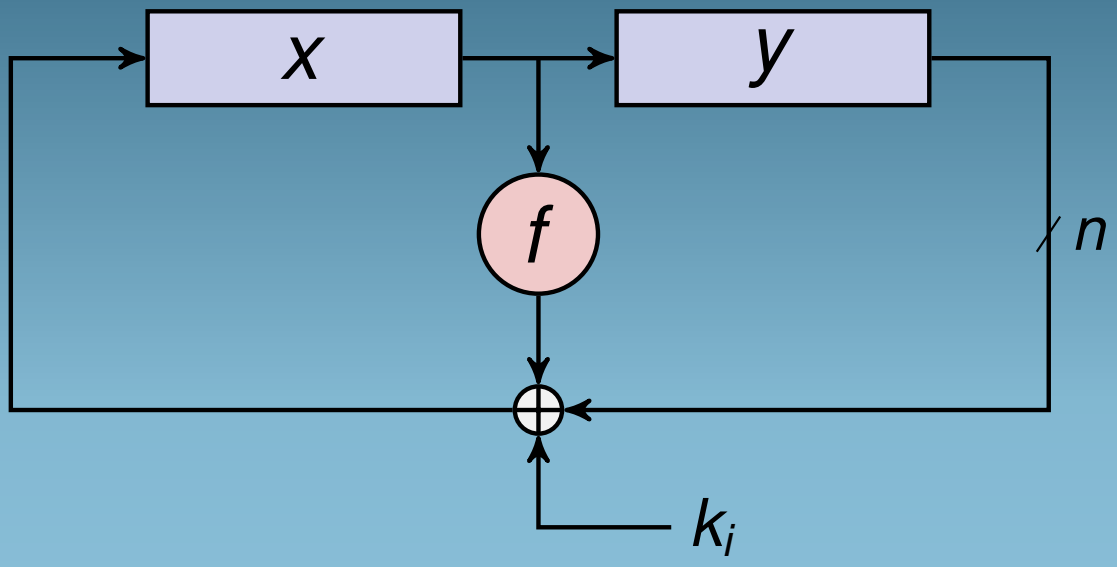
For pure hardware applications SIMON is best.

Small, fast, low energy and power. Record breaking performance on ASICs and FPGAs.

Excels on microcontrollers and microprocessors too.

NSA Trusted Systems Research Group

SIMON



$$f(y) = (Sy \& S^8 y) \oplus S^2 y,$$

where S is a left circular shift by 1 bit.

SPECK

"Südtiroler Speck g.g.A. mit Gütesiegel" by Fotograf Frieder Blickle, Wikimedia Commons

Versatile in software and hardware.

For pure software applications SPECK is best.

Small, fast, low energy and power. Record breaking performance on microcontrollers and microprocessors.

Excels on ASICs and FPGAs too.

SPECK

"Südtiroler Speck g.g.A. mit Gütesiegel" by Fotograf Frieder Blickle, Wikimedia Commons

Versatile in software and hardware.

For pure software applications SPECK is best.

Small, fast, low energy and power. Record breaking performance on microcontrollers and microprocessors.

Excels on ASICs and FPGAs too.

SPECK

Versatile in software and hardware.

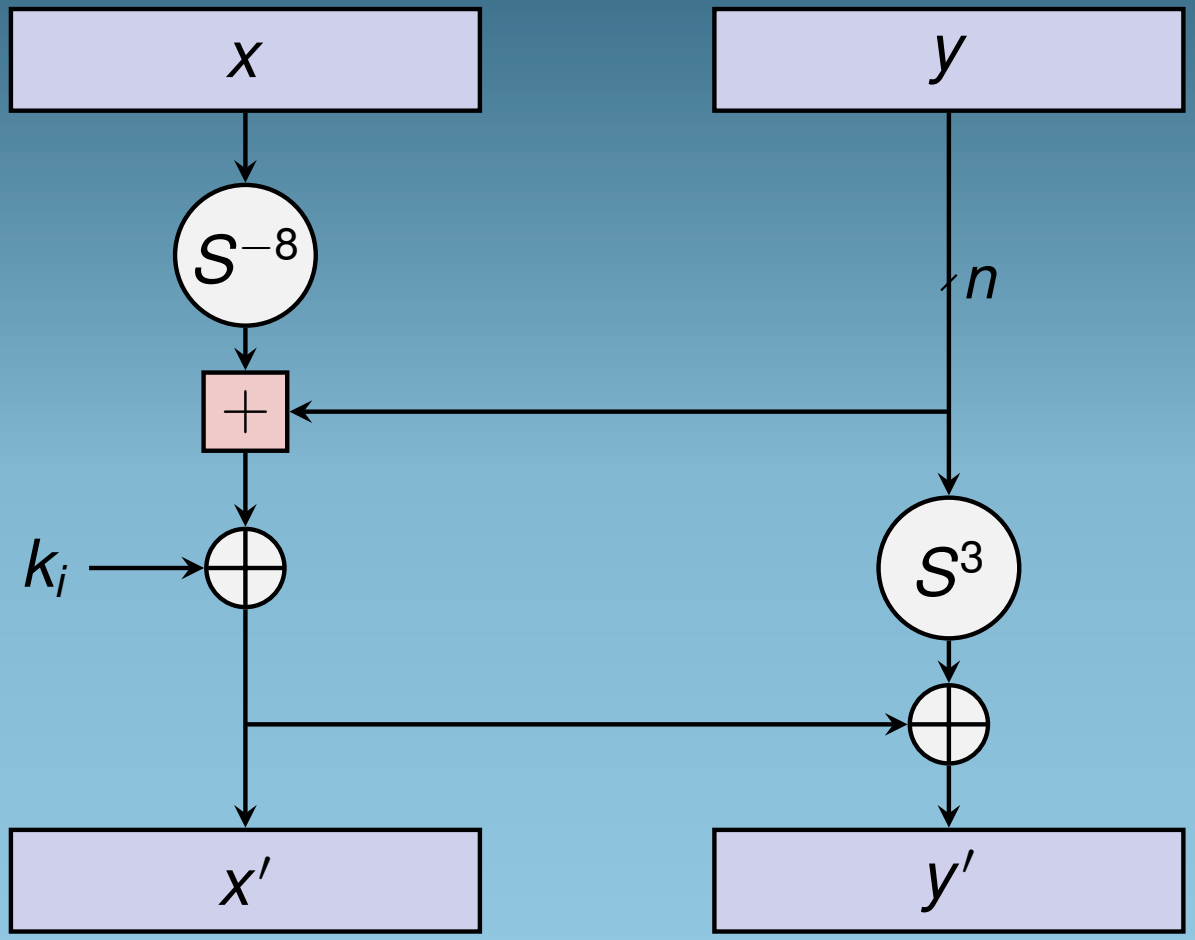
For pure software applications SPECK is best.

Small, fast, low energy and power. Record breaking performance on microcontrollers and microprocessors.

Excels on ASICs and FPGAs too.

NSA Trusted Systems Research Group

SPECK



NSA Trusted Systems Research Group

Security

size		SIMON rounds			SPECK rounds		
block	key	total	attacked		total	attacked	
32	64	32	23	(72%)	23	15	(65%)
48	72	36	24	(67%)	23	15	(65%)
48	96	36	25	(69%)	23	15	(65%)
64	96	42	30	(71%)	26	18	(69%)
64	128	44	31	(70%)	27	19	(70%)
96	96	52	37	(71%)	28	16	(57%)
96	144	54	38	(70%)	29	17	(59%)
128	128	68	49	(72%)	32	17	(53%)
128	192	69	51	(74%)	33	18	(55%)
128	256	72	53	(74%)	34	19	(56%)
AES-128		10	7	(70%)	10	7	(70%)
PRESENT		31	26	(84%)	31	26	(84%)

SIMON and SPECK have been analyzed by the world's leading cry-tologists; about 30 cryptanalysis papers to date. (See our paper for references.)

NSA Trusted Systems Research Group

Interlude: AES vs. SIMON and SPECK

We give visual depictions of the differences between these algorithms.

First — ASIC implementations (AES and SIMON). Then — fast C code (AES OpenSSL and SPECK).

AES S-box

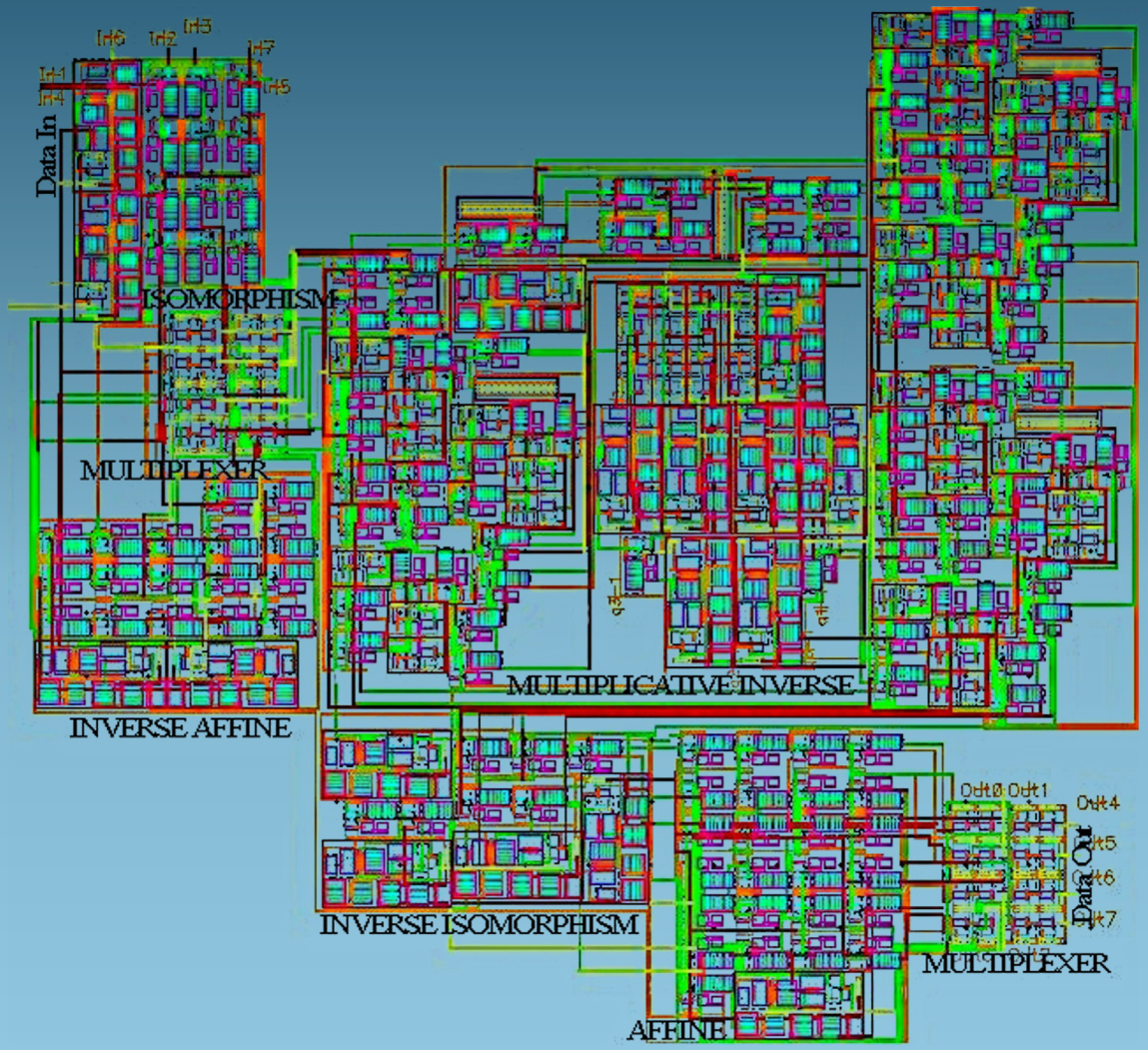


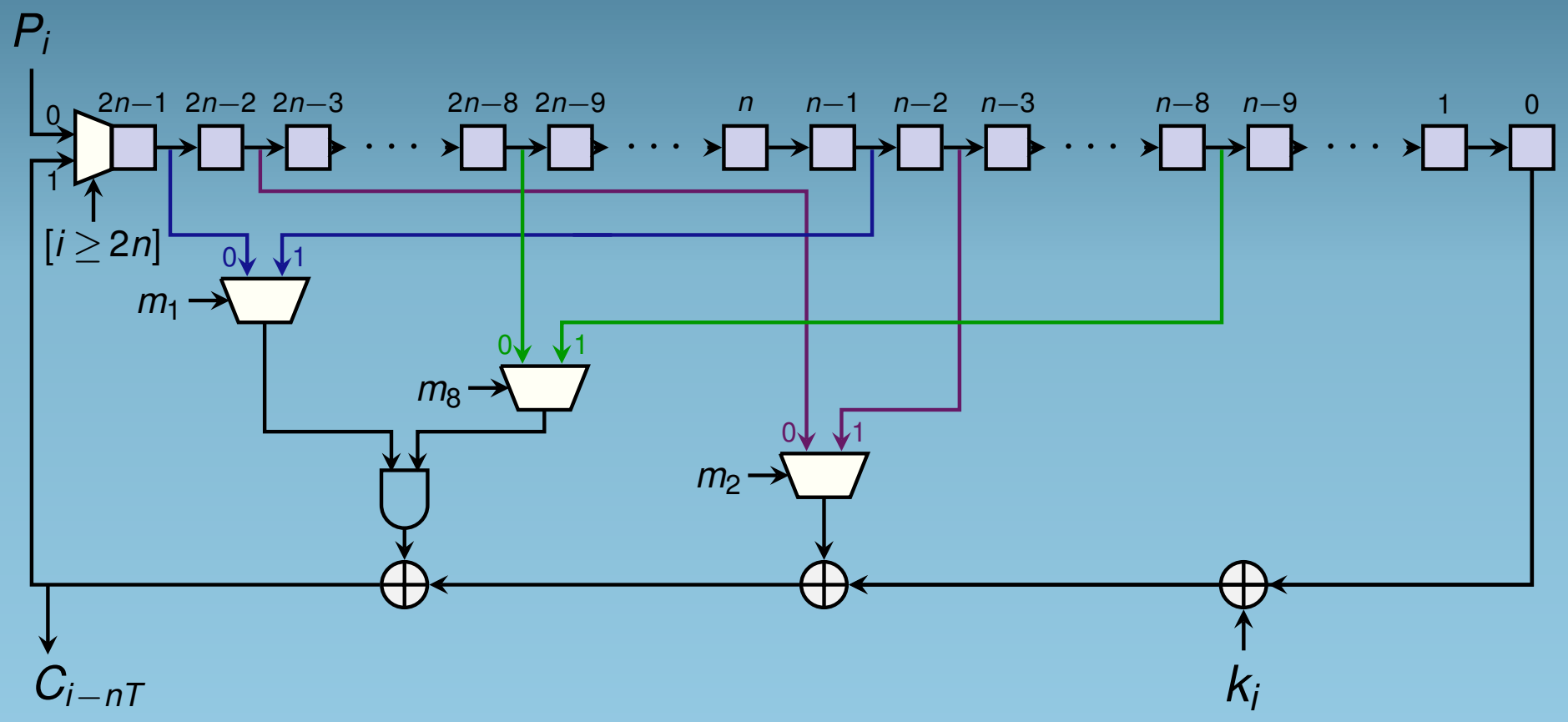
Fig. 13. Complete layout of the S-Box/Inv S-Box.

From Ahmad and Hasan, Low-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate. In INTEGRATION, the VLSI journal, vol. 46 (2013), pp. 333-344.

SIMON S-box

SIMON S-box

SIMON round function, bit serial implementation



NSA Trusted Systems Research Group

AES 128 C code (1)

```
static const u32 Te0[256] = {
    0xc66363a5U, 0xf87c7c84U, 0xee777799U, 0xf67b7b8dU, 0xfff2f20dU, 0xd66b6bbdU, 0xde6f6fb1U, 0x91c5c554U,
    0x60303050U, 0x02010103U, 0xce6767a9U, 0x562b2b7dU, 0xe7fefe19U, 0xb5d7d762U, 0x4dababe6U, 0xec76769aU,
    0x8fcaca45U, 0x1f82829dU, 0x89c9c940U, 0xfa7d7d87U, 0xeffafa15U, 0xb25959ebU, 0x8e4747c9U, 0xfbf0f00bU,
    0x41adadecU, 0xb3d4d467U, 0x5fa2a2fdU, 0x45afafeaU, 0x239c9cbfU, 0x53a4a4f7U, 0xe4727296U, 0x9bc0c05bU,
    0x75b7b7c2U, 0xe1fdfd1cU, 0x3d9393aeU, 0x4c26266aU, 0x6c36365aU, 0x7e3f3f41U, 0xf5f7f702U, 0x83cccc4fU,
    0x6834345cU, 0x51a5a5f4U, 0xd1e5e534U, 0xf9f1f108U, 0xe2717193U, 0xabd8d873U, 0x62313153U, 0x2a15153fU,
    0x0804040cU, 0x95c7c752U, 0x46232365U, 0x9dc3c35eU, 0x30181828U, 0x379696a1U, 0x0a05050fU, 0x2f9a9ab5U,
    0x0e070709U, 0x24121236U, 0x1b80809bU, 0xdfe2e23dU, 0xcdebeb26U, 0x4e272769U, 0x7fb2b2cdU, 0xea75759fU,
    0x1209091bU, 0x1d83839eU, 0x582c2c74U, 0x341a1a2eU, 0x361b1b2dU, 0xdc6e6eb2U, 0xb45a5aeeU, 0x5ba0a0fbU,
    0xa45252f6U, 0x763b3b4dU, 0xb7d6d661U, 0x7db3b3ceU, 0x5229297bU, 0xdde3e33eU, 0x5e2f2f71U, 0x13848497U,
    0xa65353f5U, 0xb9d1d168U, 0x00000000U, 0xc1eded2cU, 0x40202060U, 0xe3fcfc1fU, 0x79b1b1c8U, 0xb65b5bedU,
    0xd46a6abeU, 0x8dcbcb46U, 0x67bebed9U, 0x7239394bU, 0x944a4adeU, 0x984c4cd4U, 0xb05858e8U, 0x85cfcf4aU,
    0xbbd0d06bU, 0xc5efef2aU, 0x4faaaae5U, 0xedfbfb16U, 0x864343c5U, 0x9a4d4dd7U, 0x66333355U, 0x11858594U,
    0x8a4545cfU, 0xe9f9f910U, 0x04020206U, 0xfe7f7f81U, 0xa05050f0U, 0x783c3c44U, 0x259f9fbaU, 0x4ba8a8e3U,
    0xa25151f3U, 0x5da3a3feU, 0x804040c0U, 0x058f8f8aU, 0x3f9292adU, 0x219d9dbcU, 0x70383848U, 0xf1f5f504U,
    0x63bcbcdfU, 0x77b6b6c1U, 0xafdada75U, 0x42212163U, 0x20101030U, 0xe5ffff1aU, 0xfdf3f30eU, 0xbfd2d26dU,
    0x81cdcd4cU, 0x180c0c14U, 0x26131335U, 0xc3ecec2fU, 0xbe5f5fe1U, 0x359797a2U, 0x884444ccU, 0x2e171739U,
    0x93c4c457U, 0x55a7a7f2U, 0xfc7e7e82U, 0x7a3d3d47U, 0xc86464acU, 0xba5d5de7U, 0x3219192bU, 0xe6737395U,
    0xc06060a0U, 0x19818198U, 0x9e4f4fd1U, 0xa3dcdc7fU, 0x44222266U, 0x542a2a7eU, 0x3b9090abU, 0x0b888883U,
    0x8c4646caU, 0xc7eeee29U, 0x6bb8b8d3U, 0x2814143cU, 0xa7dede79U, 0xbc5e5ee2U, 0x160b0b1dU, 0xaddbdb76U,
    0xdbe0e03bU, 0x64323256U, 0x743a3a4eU, 0x140a0a1eU, 0x924949dbU, 0x0c06060aU, 0x4824246cU, 0xb85c5ce4U,
    0x9fc2c25dU, 0xbdd3d36eU, 0x43acacefU, 0xc46262a6U, 0x399191a8U, 0x319595a4U, 0xd3e4e437U, 0xf279798bU,
    0xd5e7e732U, 0x8bc8c843U, 0x6e373759U, 0xda6d6db7U, 0x018d8d8cU, 0xb1d5d564U, 0x9c4e4ed2U, 0x49a9a9e0U,
    0xd86c6cb4U, 0xac5656faU, 0xf3f4f407U, 0xcfeaea25U, 0xca6565afU, 0xf47a7a8eU, 0x47aeaee9U, 0x10080818U,
    0x6fbabad5U, 0xf0787888U, 0x4a25256fU, 0x5c2e2e72U, 0x381c1c24U, 0x57a6a6f1U, 0x73b4b4c7U, 0x97c6c651U,
    0xcbe8e823U, 0xa1dddd7cU, 0xe874749cU, 0x3e1f1f21U, 0x964b4bddU, 0x61bdbddcU, 0x0d8b8b86U, 0x0f8a8a85U,
    0xe0707090U, 0x7c3e3e42U, 0x71b5b5c4U, 0xcc6666aaU, 0x904848d8U, 0x06030305U, 0xf7f6f601U, 0x1c0e0e12U,
    0xc26161a3U, 0x6a35355fU, 0xae5757f9U, 0x69b9b9d0U, 0x17868691U, 0x99c1c158U, 0x3a1d1d27U, 0x279e9eb9U,
    0xd9e1e138U, 0xebf8f813U, 0x2b9898b3U, 0x22111133U, 0xd26969bbU, 0xa9d9d970U, 0x078e8e89U, 0x339494a7U,
    0x2d9b9bb6U, 0x3c1e1e22U, 0x15878792U, 0xc9e9e920U, 0x87cece49U, 0xaa5555ffU, 0x50282878U, 0xa5dfdf7aU,
    0x038c8c8fU, 0x59a1a1f8U, 0x09898980U, 0x1a0d0d17U, 0x65bfbfdaU, 0xd7e6e631U, 0x844242c6U, 0xd06868b8U,
    0x824141c3U, 0x299999b0U, 0x5a2d2d77U, 0x1e0f0f11U, 0x7bb0b0cbU, 0xa85454fcU, 0x6dbbbbd6U, 0x2c16163aU};
```

AES 128 C code (2)

```
static const u32 Te1[256] = {
    0xa5c66363U, 0x84f87c7cU, 0x99ee7777U, 0x8df67b7bU, 0x0dfff2f2U, 0xbdd66b6bU, 0xb1de6f6fU, 0x5491c5c5U,
    0x50603030U, 0x03020101U, 0xa9ce6767U, 0x7d562b2bU, 0x19e7fefeU, 0x62b5d7d7U, 0xe64dababU, 0x9aec7676U,
    0x458fcacaU, 0x9d1f8282U, 0x4089c9c9U, 0x87fa7d7dU, 0x15effafaU, 0xebb25959U, 0xc98e4747U, 0x0bfbf0f0U,
    0xec41adadU, 0x67b3d4d4U, 0xfd5fa2a2U, 0xea45afafU, 0xbf239c9cU, 0xf753a4a4U, 0x96e47272U, 0x5b9bc0c0U,
    0xc275b7b7U, 0x1ce1fdfdU, 0xae3d9393U, 0x6a4c2626U, 0x5a6c3636U, 0x417e3f3fU, 0x02f5f7f7U, 0x4f83ccccU,
    0x5c683434U, 0xf451a5a5U, 0x34d1e5e5U, 0x08f9f1f1U, 0x93e27171U, 0x73abd8d8U, 0x53623131U, 0x3f2a1515U,
    0x0c080404U, 0x5295c7c7U, 0x65462323U, 0x5e9dc3c3U, 0x28301818U, 0xa1379696U, 0x0f0a0505U, 0xb52f9a9aU,
    0x090e0707U, 0x36241212U, 0x9b1b8080U, 0x3ddfe2e2U, 0x26cdebebU, 0x694e2727U, 0xcd7fb2b2U, 0x9fea7575U,
    0x1b120909U, 0x9e1d8383U, 0x74582c2cU, 0x2e341a1aU, 0x2d361b1bU, 0xb2dc6e6eU, 0xeeb45a5aU, 0xfb5ba0a0U,
    0xf6a45252U, 0x4d763b3bU, 0x61b7d6d6U, 0xce7db3b3U, 0x7b522929U, 0x3edde3e3U, 0x715e2f2fU, 0x97138484U,
    0xf5a65353U, 0x68b9d1d1U, 0x00000000U, 0x2cc1ededU, 0x60402020U, 0x1fe3fcfcU, 0xc879b1b1U, 0xedb65b5bU,
    0xbed46a6aU, 0x468dcbcbU, 0xd967bebeU, 0x4b723939U, 0xde944a4aU, 0xd4984c4cU, 0xe8b05858U, 0x4a85cfcfU,
    0x6bbbd0d0U, 0x2ac5efefU, 0xe54faaaaU, 0x16edfbfbU, 0xc5864343U, 0xd79a4d4dU, 0x55663333U, 0x94118585U,
    0xcf8a4545U, 0x10e9f9f9U, 0x06040202U, 0x81fe7f7fU, 0xf0a05050U, 0x44783c3cU, 0xba259f9fU, 0xe34ba8a8U,
    0xf3a25151U, 0xfe5da3a3U, 0xc0804040U, 0x8a058f8fU, 0xad3f9292U, 0xbc219d9dU, 0x48703838U, 0x04f1f5f5U,
    0xdf63bcbcU, 0xc177b6b6U, 0x75afdadaU, 0x63422121U, 0x30201010U, 0x1ae5ffffU, 0x0efdf3f3U, 0x6dbfd2d2U,
    0x4c81cdcdU, 0x14180c0cU, 0x35261313U, 0x2fc3ececU, 0xe1be5f5fU, 0xa2359797U, 0xcc884444U, 0x392e1717U,
    0x5793c4c4U, 0xf255a7a7U, 0x82fc7e7eU, 0x477a3d3dU, 0xacc86464U, 0xe7ba5d5dU, 0x2b321919U, 0x95e67373U,
    0xa0c06060U, 0x98198181U, 0xd19e4f4fU, 0x7fa3dcdcU, 0x66442222U, 0x7e542a2aU, 0xab3b9090U, 0x830b8888U,
    0xca8c4646U, 0x29c7eeeeU, 0xd36bb8b8U, 0x3c281414U, 0x79a7dedeU, 0xe2bc5e5eU, 0x1d160b0bU, 0x76addbdbU,
    0x3bdbe0e0U, 0x56643232U, 0x4e743a3aU, 0x1e140a0aU, 0xdb924949U, 0x0a0c0606U, 0x6c482424U, 0xe4b85c5cU,
    0x5d9fc2c2U, 0x6ebdd3d3U, 0xef43acacU, 0xa6c46262U, 0xa8399191U, 0xa4319595U, 0x37d3e4e4U, 0x8bf27979U,
    0x32d5e7e7U, 0x438bc8c8U, 0x596e3737U, 0xb7da6d6dU, 0x8c018d8dU, 0x64b1d5d5U, 0xd29c4e4eU, 0xe049a9a9U,
    0xb4d86c6cU, 0xfaac5656U, 0x07f3f4f4U, 0x25cfeaeaU, 0xafca6565U, 0x8ef47a7aU, 0xe947aeaeU, 0x18100808U,
    0xd56fbabaU, 0x88f07878U, 0x6f4a2525U, 0x725c2e2eU, 0x24381c1cU, 0xf157a6a6U, 0xc773b4b4U, 0x5197c6c6U,
    0x23cbe8e8U, 0x7ca1ddddU, 0x9ce87474U, 0x213e1f1fU, 0xdd964b4bU, 0xdc61bdbdU, 0x860d8b8bU, 0x850f8a8aU,
    0x90e07070U, 0x427c3e3eU, 0xc471b5b5U, 0xaacc6666U, 0xd8904848U, 0x05060303U, 0x01f7f6f6U, 0x121c0e0eU,
    0xa3c26161U, 0x5f6a3535U, 0xf9ae5757U, 0xd069b9b9U, 0x91178686U, 0x5899c1c1U, 0x273a1d1dU, 0xb9279e9eU,
    0x38d9e1e1U, 0x13ebf8f8U, 0xb32b9898U, 0x33221111U, 0xbbd26969U, 0x70a9d9d9U, 0x89078e8eU, 0xa7339494U,
    0xb62d9b9bU, 0x223c1e1eU, 0x92158787U, 0x20c9e9e9U, 0x4987ceceU, 0xffaa5555U, 0x78502828U, 0x7aa5dfdfU,
    0x8f038c8cU, 0xf859a1a1U, 0x80098989U, 0x171a0d0dU, 0xda65bfbfU, 0x31d7e6e6U, 0xc6844242U, 0xb8d06868U,
    0xc3824141U, 0xb0299999U, 0x775a2d2dU, 0x111e0f0fU, 0xcb7bb0b0U, 0xfca85454U, 0xd66dbbbbU, 0x3a2c1616U};
```

AES 128 C code (3)

```
static const u32 Te2[256] = {
    0x63a5c663U, 0x7c84f87cU, 0x7799ee77U, 0x7b8df67bU, 0xf20dfff2U, 0x6bbdd66bU, 0x6fb1de6fU, 0xc55491c5U,
    0x30506030U, 0x01030201U, 0x67a9ce67U, 0x2b7d562bU, 0xfe19e7feU, 0xd762b5d7U, 0xabe64dabU, 0x769aec76U,
    0xca458fcaU, 0x829d1f82U, 0xc94089c9U, 0x7d87fa7dU, 0xfa15effaU, 0x59ebb259U, 0x47c98e47U, 0xf00bfbf0U,
    0xadec41adU, 0xd467b3d4U, 0xa2fd5fa2U, 0xafea45afU, 0x9cbf239cU, 0xa4f753a4U, 0x7296e472U, 0xc05b9bc0U,
    0xb7c275b7U, 0xfd1ce1fdU, 0x93ae3d93U, 0x266a4c26U, 0x365a6c36U, 0x3f417e3fU, 0xf702f5f7U, 0xcc4f83ccU,
    0x345c6834U, 0xa5f451a5U, 0xe534d1e5U, 0xf108f9f1U, 0x7193e271U, 0xd873abd8U, 0x31536231U, 0x153f2a15U,
    0x040c0804U, 0xc75295c7U, 0x23654623U, 0xc35e9dc3U, 0x18283018U, 0x96a13796U, 0x050f0a05U, 0x9ab52f9aU,
    0x07090e07U, 0x12362412U, 0x809b1b80U, 0xe23ddfe2U, 0xeb26cdebU, 0x27694e27U, 0xb2cd7fb2U, 0x759fea75U,
    0x091b1209U, 0x839e1d83U, 0x2c74582cU, 0x1a2e341aU, 0x1b2d361bU, 0x6eb2dc6eU, 0x5aeeb45aU, 0xa0fb5ba0U,
    0x52f6a452U, 0x3b4d763bU, 0xd661b7d6U, 0xb3ce7db3U, 0x297b5229U, 0xe33edde3U, 0x2f715e2fU, 0x84971384U,
    0x53f5a653U, 0xd168b9d1U, 0x00000000U, 0xed2cc1edU, 0x20604020U, 0xfc1fe3fcU, 0xb1c879b1U, 0x5bedb65bU,
    0x6abed46aU, 0xcb468dcbU, 0xbed967beU, 0x394b7239U, 0x4ade944aU, 0x4cd4984cU, 0x58e8b058U, 0xcf4a85cfU,
    0xd06bbbd0U, 0xef2ac5efU, 0xaae54faaU, 0xfb16edfbU, 0x43c58643U, 0x4dd79a4dU, 0x33556633U, 0x85941185U,
    0x45cf8a45U, 0xf910e9f9U, 0x02060402U, 0x7f81fe7fU, 0x50f0a050U, 0x3c44783cU, 0x9fba259fU, 0xa8e34ba8U,
    0x51f3a251U, 0xa3fe5da3U, 0x40c08040U, 0x8f8a058fU, 0x92ad3f92U, 0x9dbc219dU, 0x38487038U, 0xf504f1f5U,
    0xbcdf63bcU, 0xb6c177b6U, 0xda75afdaU, 0x21634221U, 0x10302010U, 0xff1ae5ffU, 0xf30efdf3U, 0xd26dbfd2U,
    0xcd4c81cdU, 0x0c14180cU, 0x13352613U, 0xec2fc3ecU, 0x5fe1be5fU, 0x97a23597U, 0x44cc8844U, 0x17392e17U,
    0xc45793c4U, 0xa7f255a7U, 0x7e82fc7eU, 0x3d477a3dU, 0x64acc864U, 0x5de7ba5dU, 0x192b3219U, 0x7395e673U,
    0x60a0c060U, 0x81981981U, 0x4fd19e4fU, 0xdc7fa3dcU, 0x22664422U, 0x2a7e542aU, 0x90ab3b90U, 0x88830b88U,
    0x46ca8c46U, 0xee29c7eeU, 0xb8d36bb8U, 0x143c2814U, 0xde79a7deU, 0x5ee2bc5eU, 0x0b1d160bU, 0xdb76addbU,
    0xe03bdbe0U, 0x32566432U, 0x3a4e743aU, 0x0a1e140aU, 0x49db9249U, 0x060a0c06U, 0x246c4824U, 0x5ce4b85cU,
    0xc25d9fc2U, 0xd36ebdd3U, 0xacef43acU, 0x62a6c462U, 0x91a83991U, 0x95a43195U, 0xe437d3e4U, 0x798bf279U,
    0xe732d5e7U, 0xc8438bc8U, 0x37596e37U, 0x6db7da6dU, 0x8d8c018dU, 0xd564b1d5U, 0x4ed29c4eU, 0xa9e049a9U,
    0x6cb4d86cU, 0x56faac56U, 0xf407f3f4U, 0xea25cfeaU, 0x65afca65U, 0x7a8ef47aU, 0xaee947aeU, 0x08181008U,
    0xbad56fbaU, 0x7888f078U, 0x256f4a25U, 0x2e725c2eU, 0x1c24381cU, 0xa6f157a6U, 0xb4c773b4U, 0xc65197c6U,
    0xe823cbe8U, 0xdd7ca1ddU, 0x749ce874U, 0x1f213e1fU, 0x4bdd964bU, 0xbddc61bdU, 0x8b860d8bU, 0x8a850f8aU,
    0x7090e070U, 0x3e427c3eU, 0xb5c471b5U, 0x66aacc66U, 0x48d89048U, 0x03050603U, 0xf601f7f6U, 0x0e121c0eU,
    0x61a3c261U, 0x355f6a35U, 0x57f9ae57U, 0xb9d069b9U, 0x86911786U, 0xc15899c1U, 0x1d273a1dU, 0x9eb9279eU,
    0xe138d9e1U, 0xf813ebf8U, 0x98b32b98U, 0x11332211U, 0x69bbd269U, 0xd970a9d9U, 0x8e89078eU, 0x94a73394U,
    0x9bb62d9bU, 0x1e223c1eU, 0x87921587U, 0xe920c9e9U, 0xce4987ceU, 0x55ffaa55U, 0x28785028U, 0xdf7aa5dfU,
    0x8c8f038cU, 0xa1f859a1U, 0x89800989U, 0x0d171a0dU, 0xbfda65bfU, 0xe631d7e6U, 0x42c68442U, 0x68b8d068U,
    0x41c38241U, 0x99b02999U, 0x2d775a2dU, 0x0f111e0fU, 0xb0cb7bb0U, 0x54fca854U, 0xbbd66dbbU, 0x163a2c16U};
```

AES 128 C code (4)

```
static const u32 Te3[256] = {
    0x6363a5c6U, 0x7c7c84f8U, 0x777799eeU, 0x7b7b8df6U, 0xf2f20dffU, 0x6b6bbdd6U, 0x6f6fb1deU, 0xc5c55491U,
    0x30305060U, 0x01010302U, 0x6767a9ceU, 0x2b2b7d56U, 0xfefe19e7U, 0xd7d762b5U, 0xababe64dU, 0x76769aecU,
    0xcaca458fU, 0x82829d1fU, 0xc9c94089U, 0x7d7d87faU, 0xfafa15efU, 0x5959ebb2U, 0x4747c98eU, 0xf0f00bfbU,
    0xadadec41U, 0xd4d467b3U, 0xa2a2fd5fU, 0xafafea45U, 0x9c9cbf23U, 0xa4a4f753U, 0x727296e4U, 0xc0c05b9bU,
    0xb7b7c275U, 0xfdfd1ce1U, 0x9393ae3dU, 0x26266a4cU, 0x36365a6cU, 0x3f3f417eU, 0xf7f702f5U, 0xcccc4f83U,
    0x34345c68U, 0xa5a5f451U, 0xe5e534d1U, 0xf1f108f9U, 0x717193e2U, 0xd8d873abU, 0x31315362U, 0x15153f2aU,
    0x04040c08U, 0xc7c75295U, 0x23236546U, 0xc3c35e9dU, 0x18182830U, 0x9696a137U, 0x05050f0aU, 0x9a9ab52fU,
    0x0707090eU, 0x12123624U, 0x80809b1bU, 0xe2e23ddfU, 0xebeb26cdU, 0x2727694eU, 0xb2b2cd7fU, 0x75759feaU,
    0x09091b12U, 0x83839e1dU, 0x2c2c7458U, 0x1a1a2e34U, 0x1b1b2d36U, 0x6e6eb2dcU, 0x5a5aeeb4U, 0xa0a0fb5bU,
    0x5252f6a4U, 0x3b3b4d76U, 0xd6d661b7U, 0xb3b3ce7dU, 0x29297b52U, 0xe3e33eddU, 0x2f2f715eU, 0x84849713U,
    0x5353f5a6U, 0xd1d168b9U, 0x00000000U, 0xeded2cc1U, 0x20206040U, 0xfcfc1fe3U, 0xb1b1c879U, 0x5b5bedb6U,
    0x6a6abed4U, 0xcbcb468dU, 0xbebed967U, 0x39394b72U, 0x4a4ade94U, 0x4c4cd498U, 0x5858e8b0U, 0xcfcf4a85U,
    0xd0d06bbbU, 0xefef2ac5U, 0xaaaae54fU, 0xfbfb16edU, 0x4343c586U, 0x4d4dd79aU, 0x33335566U, 0x85859411U,
    0x4545cf8aU, 0xf9f910e9U, 0x02020604U, 0x7f7f81feU, 0x5050f0a0U, 0x3c3c4478U, 0x9f9fba25U, 0xa8a8e34bU,
    0x5151f3a2U, 0xa3a3fe5dU, 0x4040c080U, 0x8f8f8a05U, 0x9292ad3fU, 0x9d9dbc21U, 0x38384870U, 0xf5f504f1U,
    0xbcbcdf63U, 0xb6b6c177U, 0xdada75afU, 0x21216342U, 0x10103020U, 0xffff1ae5U, 0xf3f30efdU, 0xd2d26dbfU,
    0xcdcd4c81U, 0x0c0c1418U, 0x13133526U, 0xecec2fc3U, 0x5f5fe1beU, 0x9797a235U, 0x4444cc88U, 0x1717392eU,
    0xc4c45793U, 0xa7a7f255U, 0x7e7e82fcU, 0x3d3d477aU, 0x6464acc8U, 0x5d5de7baU, 0x19192b32U, 0x737395e6U,
    0x6060a0c0U, 0x81819819U, 0x4f4fd19eU, 0xdcdc7fa3U, 0x22226644U, 0x2a2a7e54U, 0x9090ab3bU, 0x8888830bU,
    0x4646ca8cU, 0xeeee29c7U, 0xb8b8d36bU, 0x14143c28U, 0xdede79a7U, 0x5e5ee2bcU, 0x0b0b1d16U, 0xdbdb76adU,
    0xe0e03bdbU, 0x32325664U, 0x3a3a4e74U, 0x0a0a1e14U, 0x4949db92U, 0x06060a0cU, 0x24246c48U, 0x5c5ce4b8U,
    0xc2c25d9fU, 0xd3d36ebdU, 0xacacef43U, 0x6262a6c4U, 0x9191a839U, 0x9595a431U, 0xe4e437d3U, 0x79798bf2U,
    0xe7e732d5U, 0xc8c8438bU, 0x3737596eU, 0x6d6db7daU, 0x8d8d8c01U, 0xd5d564b1U, 0x4e4ed29cU, 0xa9a9e049U,
    0x6c6cb4d8U, 0x5656faacU, 0xf4f407f3U, 0xeaea25cfU, 0x6565afcaU, 0x7a7a8ef4U, 0xaeaee947U, 0x08081810U,
    0xbabad56fU, 0x787888f0U, 0x25256f4aU, 0x2e2e725cU, 0x1c1c2438U, 0xa6a6f157U, 0xb4b4c773U, 0xc6c65197U,
    0xe8e823cbU, 0xdddd7ca1U, 0x74749ce8U, 0x1f1f213eU, 0x4b4bdd96U, 0xbdbddc61U, 0x8b8b860dU, 0x8a8a850fU,
    0x707090e0U, 0x3e3e427cU, 0xb5b5c471U, 0x6666aaccU, 0x4848d890U, 0x03030506U, 0xf6f601f7U, 0x0e0e121cU,
    0x6161a3c2U, 0x35355f6aU, 0x5757f9aeU, 0xb9b9d069U, 0x86869117U, 0xc1c15899U, 0x1d1d273aU, 0x9e9eb927U,
    0xe1e138d9U, 0xf8f813ebU, 0x9898b32bU, 0x11113322U, 0x6969bbd2U, 0xd9d970a9U, 0x8e8e8907U, 0x9494a733U,
    0x9b9bb62dU, 0x1e1e223cU, 0x87879215U, 0xe9e920c9U, 0xcece4987U, 0x5555ffaaU, 0x28287850U, 0xdfdf7aa5U,
    0x8c8c8f03U, 0xa1a1f859U, 0x89898009U, 0x0d0d171aU, 0xbfbfda65U, 0xe6e631d7U, 0x4242c684U, 0x6868b8d0U,
    0x4141c382U, 0x9999b029U, 0x2d2d775aU, 0x0f0f111eU, 0xb0b0cb7bU, 0x5454fca8U, 0xbbbbd66dU, 0x16163a2cU};
```

AES 128 C code (5)

```
static const u32 Te4[256] = {
    0x63636363U, 0x7c7c7c7cU, 0x77777777U, 0x7b7b7b7bU, 0xf2f2f2f2U, 0x6b6b6b6bU, 0x6f6f6f6fU, 0xc5c5c5c5U,
    0x30303030U, 0x01010101U, 0x67676767U, 0x2b2b2b2bU, 0xfefefefeU, 0xd7d7d7d7U, 0xabababababU, 0x76767676U,
    0xcacacacaU, 0x82828282U, 0xc9c9c9c9U, 0x7d7d7d7dU, 0xfafafafaU, 0x59595959U, 0x47474747U, 0xf0f0f0f0U,
    0xadadadadU, 0xd4d4d4d4U, 0xa2a2a2a2U, 0xafafafafU, 0x9c9c9c9cU, 0xa4a4a4a4U, 0x72727272U, 0xc0c0c0c0U,
    0xb7b7b7b7U, 0xfdfdfdfdU, 0x93939393U, 0x26262626U, 0x36363636U, 0x3f3f3f3fU, 0xf7f7f7f7U, 0xccccccccU,
    0x34343434U, 0xa5a5a5a5U, 0xe5e5e5e5U, 0xf1f1f1f1U, 0x71717171U, 0xd8d8d8d8U, 0x31313131U, 0x15151515U,
    0x04040404U, 0xc7c7c7c7U, 0x23232323U, 0xc3c3c3c3U, 0x18181818U, 0x96969696U, 0x05050505U, 0x9a9a9a9aU,
    0x07070707U, 0x12121212U, 0x80808080U, 0xe2e2e2e2U, 0xebebebebU, 0x27272727U, 0xb2b2b2b2U, 0x75757575U,
    0x09090909U, 0x83838383U, 0x2c2c2c2cU, 0x1a1a1a1aU, 0x1b1b1b1bU, 0x6e6e6e6eU, 0x5a5a5a5aU, 0xa0a0a0a0U,
    0x52525252U, 0x3b3b3b3bU, 0xd6d6d6d6U, 0xb3b3b3b3U, 0x29292929U, 0xe3e3e3e3U, 0x2f2f2f2fU, 0x84848484U,
    0x53535353U, 0xd1d1d1d1U, 0x00000000U, 0xededededU, 0x20202020U, 0xfcfcfcfcU, 0xb1b1b1b1U, 0x5b5b5b5bU,
    0x6a6a6a6aU, 0xcbcbcbcbU, 0xbebebebeU, 0x39393939U, 0x4a4a4a4aU, 0x4c4c4c4cU, 0x58585858U, 0xcfcfcfcfU,
    0xd0d0d0d0U, 0xefefefefU, 0xaaaaaaaaU, 0xfbfbfbfbU, 0x43434343U, 0x4d4d4d4dU, 0x33333333U, 0x85858585U,
    0x45454545U, 0xf9f9f9f9U, 0x02020202U, 0x7f7f7f7fU, 0x50505050U, 0x3c3c3c3cU, 0x9f9f9f9fU, 0xa8a8a8a8U,
    0x51515151U, 0xa3a3a3a3U, 0x40404040U, 0x8f8f8f8fU, 0x92929292U, 0x9d9d9d9dU, 0x38383838U, 0xf5f5f5f5U,
    0xbcbcbcbcU, 0xb6b6b6b6U, 0xdadadadaU, 0x21212121U, 0x10101010U, 0xffffffffU, 0xf3f3f3f3U, 0xd2d2d2d2U,
    0xcdcdcdcdU, 0x0c0c0c0cU, 0x13131313U, 0xeceecececU, 0x5f5f5f5fU, 0x97979797U, 0x44444444U, 0x17171717U,
    0xc4c4c4c4U, 0xa7a7a7a7U, 0x7e7e7e7eU, 0x3d3d3d3dU, 0x64646464U, 0x5d5d5d5dU, 0x19191919U, 0x73737373U,
    0x60606060U, 0x81818181U, 0x4f4f4f4fU, 0xdcdcdcdcU, 0x22222222U, 0x2a2a2a2aU, 0x90909090U, 0x88888888U,
    0x46464646U, 0xeeeeeeeeU, 0xb8b8b8b8U, 0x14141414U, 0xdedededeU, 0x5e5e5e5eU, 0x0b0b0b0bU, 0xdbdbdbdbU,
    0xe0e0e0e0U, 0x32323232U, 0x3a3a3a3aU, 0x0a0a0a0aU, 0x49494949U, 0x06060606U, 0x24242424U, 0x5c5c5c5cU,
    0xc2c2c2c2U, 0xd3d3d3d3U, 0xacacacacU, 0x62626262U, 0x91919191U, 0x95959595U, 0xe4e4e4e4U, 0x79797979U,
    0xe7e7e7e7U, 0xc8c8c8c8U, 0x37373737U, 0x6d6d6d6dU, 0x8d8d8d8dU, 0xd5d5d5d5U, 0x4e4e4e4eU, 0xa9a9a9a9U,
    0x6c6c6c6cU, 0x56565656U, 0xf4f4f4f4U, 0xeaeaeaeaU, 0x65656565U, 0x7a7a7a7aU, 0xaeaeaeaeU, 0x08080808U,
    0xbababababaU, 0x78787878U, 0x25252525U, 0x2e2e2e2eU, 0x1c1c1c1cU, 0xa6a6a6a6U, 0xb4b4b4b4U, 0xc6c6c6c6U,
    0xe8e8e8e8U, 0xdddddddddU, 0x74747474U, 0x1f1f1f1fU, 0x4b4b4b4bU, 0xbdbdbdbdU, 0x8b8b8b8bU, 0x8a8a8a8aU,
    0x70707070U, 0x3e3e3e3eU, 0xb5b5b5b5U, 0x66666666U, 0x48484848U, 0x03030303U, 0xf6f6f6f6U, 0x0e0e0e0eU,
    0x61616161U, 0x35353535U, 0x57575757U, 0xb9b9b9b9U, 0x86868686U, 0xc1c1c1c1U, 0x1d1d1d1dU, 0x9e9e9e9eU,
    0xe1e1e1e1U, 0xf8f8f8f8U, 0x98989898U, 0x11111111U, 0x69696969U, 0xd9d9d9d9U, 0x8e8e8e8eU, 0x94949494U,
    0x9b9b9b9bU, 0x1e1e1e1eU, 0x87878787U, 0xe9e9e9e9U, 0xcecececeU, 0x55555555U, 0x28282828U, 0xdfdfdfdfU,
    0x8c8c8c8cU, 0xa1a1a1a1U, 0x89898989U, 0x0d0d0d0dU, 0xbfbfbfbfU, 0xe6e6e6e6U, 0x42424242U, 0x68686868U,
    0x41414141U, 0x99999999U, 0x2d2d2d2dU, 0x0f0f0f0fU, 0xb0b0b0b0U, 0x54545454U, 0xbbbbbbbbU, 0x16161616U};
```

AES 128 C code (6)

```
int AES_set_encrypt_key(const unsigned char *userKey, const int bits,AES_KEY *key)
{

   u32 *rk, temp;
   int i = 0;

   if (!userKey || !key) return -1;
   rk = key->rd_key;  key->rounds = 10;
   rk[0] = GETU32(userKey     );  rk[1] = GETU32(userKey +  4);
   rk[2] = GETU32(userKey +  8);  rk[3] = GETU32(userKey + 12);
   for (;;) {
     temp  = rk[3];
     rk[4] = rk[0] ^ (Te4[(temp >> 16) & 0xff] & 0xff000000) ^
     (Te4[(temp >>  8) & 0xff] & 0x00ff0000) ^
     (Te4[(temp      ) & 0xff] & 0x0000ff00) ^
     (Te4[(temp >> 24)       ] & 0x000000ff) ^ rcon[i];
     rk[5] = rk[1] ^ rk[4];  rk[6] = rk[2] ^ rk[5];  rk[7] = rk[3] ^ rk[6];
     if (++i == 10) return 0;
     rk += 4;
   }

   rk[4] = GETU32(userKey + 16);
   rk[5] = GETU32(userKey + 20);
   rk[6] = GETU32(userKey + 24);
   rk[7] = GETU32(userKey + 28);

   return 0;
}
```

AES 128 C code (7)

```
void AES_encrypt(const unsigned char *in, unsigned char *out,const AES_KEY *key) {

  const u32 *rk;
  u32 s0, s1, s2, s3, t0, t1, t2, t3;
  int r;

  assert(in && out && key); rk = key->rd_key;

  s0 = GETU32(in     ) ^ rk[0];
  s1 = GETU32(in +  4) ^ rk[1];
  s2 = GETU32(in +  8) ^ rk[2];
  s3 = GETU32(in + 12) ^ rk[3];

  r = key->rounds >> 1;

  for (;;) {
    t0 = Te0[(s0 >> 24)       ] ^ Te1[(s1 >> 16) & 0xff] ^
         Te2[(s2 >>  8) & 0xff] ^ Te3[(s3      ) & 0xff] ^ rk[4];
    t1 = Te0[(s1 >> 24)       ] ^ Te1[(s2 >> 16) & 0xff] ^
         Te2[(s3 >>  8) & 0xff] ^ Te3[(s0      ) & 0xff] ^ rk[5];
    t2 = Te0[(s2 >> 24)       ] ^ Te1[(s3 >> 16) & 0xff] ^
         Te2[(s0 >>  8) & 0xff] ^ Te3[(s1      ) & 0xff] ^ rk[6];
    t3 = Te0[(s3 >> 24)       ] ^ Te1[(s0 >> 16) & 0xff] ^
         Te2[(s1 >>  8) & 0xff] ^ Te3[(s2      ) & 0xff] ^ rk[7];

    rk += 8;  if (--r == 0) break;
```

AES 128 C code (8)

```
    s0 = Te0[(t0 >> 24)       ] ^ Te1[(t1 >> 16) & 0xff] ^
         Te2[(t2 >>  8) & 0xff] ^ Te3[(t3      ) & 0xff] ^ rk[0];
    s1 = Te0[(t1 >> 24)       ] ^ Te1[(t2 >> 16) & 0xff] ^
         Te2[(t3 >>  8) & 0xff] ^ Te3[(t0      ) & 0xff] ^ rk[1];
    s2 = Te0[(t2 >> 24)       ] ^ Te1[(t3 >> 16) & 0xff] ^
         Te2[(t0 >>  8) & 0xff] ^ Te3[(t1      ) & 0xff] ^ rk[2];
    s3 = Te0[(t3 >> 24)       ] ^ Te1[(t0 >> 16) & 0xff] ^
         Te2[(t1 >>  8) & 0xff] ^ Te3[(t2      ) & 0xff] ^ rk[3];
  }

  s0 = (Te4[(t0 >> 24)       ] & 0xff000000) ^ (Te4[(t1 >> 16) & 0xff] & 0xff0000)
     ^ (Te4[(t2 >>  8) & 0xff] & 0xff00) ^ (Te4[(t3      ) & 0xff] & 0xff)
     ^ rk[0];  PUTU32(out     , s0);

  s1 = (Te4[(t1 >> 24)       ] & 0xff000000) ^ (Te4[(t2 >> 16) & 0xff] & 0xff0000)
     ^ (Te4[(t3 >>  8) & 0xff] & 0xff00) ^ (Te4[(t0      ) & 0xff] & 0xff)
     ^ rk[1];  PUTU32(out +  4, s1);

  s2 = (Te4[(t2 >> 24)       ] & 0xff000000) ^ (Te4[(t3 >> 16) & 0xff] & 0xff0000)
     ^ (Te4[(t0 >>  8) & 0xff] & 0xff00) ^ (Te4[(t1      ) & 0xff] & 0xff)
     ^ rk[2];  PUTU32(out +  8, s2);

  s3 = (Te4[(t3 >> 24)       ] & 0xff000000) ^ (Te4[(t0 >> 16) & 0xff] & 0xff0000)
     ^ (Te4[(t1 >>  8) & 0xff] & 0xff00) ^ (Te4[(t2      ) & 0xff] & 0xff)
     ^ rk[3]; PUTU32(out + 12, s3);  //132 characters for this part, just under one tweet!
}
```

NSA Trusted Systems Research Group

SPECK 128/128 in one tweet

```
#define R(x,y,k)x=x>>8|x<<56,x+=y,x^=k,y=y<<3|y>>61,y^=x
void E(uint64_t *T,uint64_t *K){for(int i=0;i<32;){R(T[1],*T,*K);
                                                    R(K[1],*K,i++);}}
```

Adapted from code by Bo Zhu, https://gist.github.com/zhuzhuor

SIMON and SPECK performance

For most platforms and constraints SIMON, SPECK, or both outperform existing block ciphers.

- ASIC/FPGA area
- ASIC/FPGA efficiency (throughput/area)
- Latency
- Ease of side-channel protection
- Power and energy efficiency
- Software performance (size, speed, energy) on 8-, 16-, 32-, and 64-bit processors

We don't have the time here to discuss all the various implementation results. Please see our workshop paper for more.

We briefly mention *latency* on the next slide.

Latency

Low latency $\neq$ single-cycle encryption.

algorithm	area (GE)	latency (ns)	clock (MHz)	cycles
PRINCE	8679*	28.17	35.5	1
	9522*	22.9	43.7	1
SIMON 64/128	6576	27.56	399.1	11
	9516	22.88	393.3	9
SPECK 64/128	6377	52.36	191.0	10

* registers not counted

The SIMON and SPECK Team

Ray Beaulieu	rayb@ccrwest.org
Doug Shors	djshors@tycho.ncsc.mil
Jason Smith	jksmit3@tycho.ncsc.mil
Stefan Treatman-Clark	sgtreat@tycho.ncsc.mil
Bryan Weeks	beweeks@tycho.ncsc.mil
Louis Wingers	lrwinge@tycho.ncsc.mil

NSA Trusted Systems Research Group

Thanks for listening!

Thanks for listening!

NSA Trusted Systems Research Group