资源代价与安全算力

密码算力的思考与优化

炼石 CipherGateway 炼石网络创始人、CEO 白小勇

CONTENTS
目 录

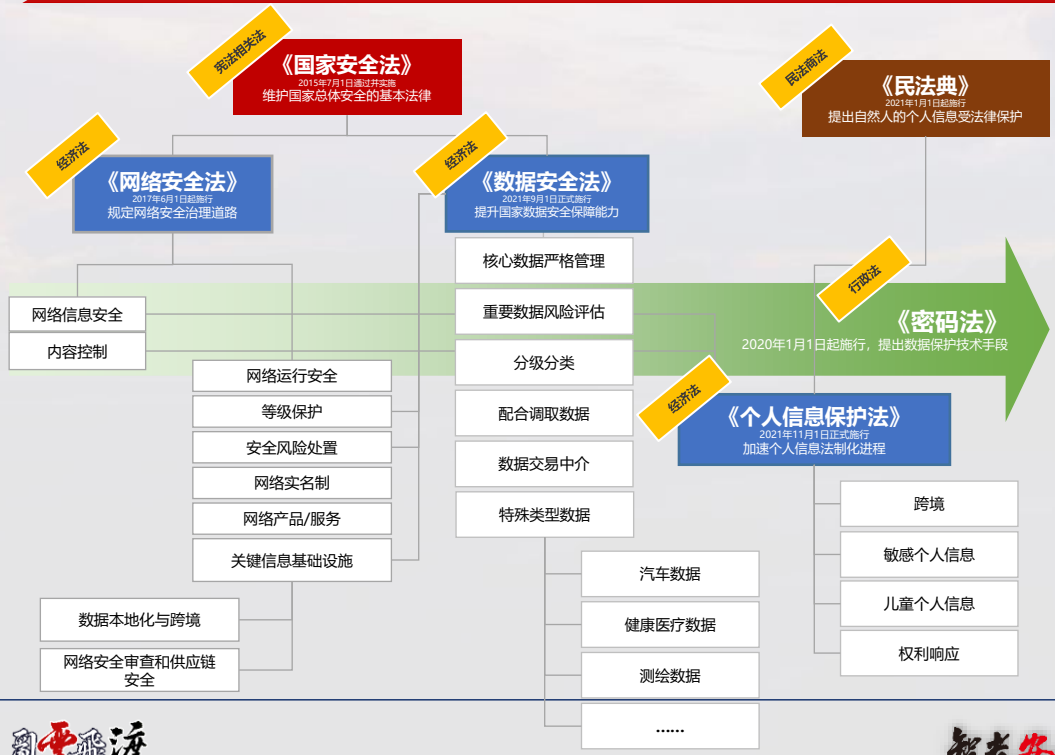
01 密码算力创造业务价值

面向“十四五”数字经济，强化安全建设

合规驱动

- **《中共中央国务院关于构建更加完善的要素市场化配置体制机制的意见》**明确要求数据安全
- **《中共中央关于制定国民经济和社会发展第十四个五年规划和二〇三五年远景目标的建议》**明确提出：保障国家数据安全，加强个人信息保护

五法一典



“十四五”数字经济发展规划

着力强化数字经济安全体系

- 加强网络安全等级保护和**密码应用安全性评估**。支持网络安全保护技术和产品研发应用，推广使用安全可靠的信息产品、服务和解决方案。
- 强化针对新技术、新应用的安全研究管理，为新产业新业态新模式健康发展提供保障。加快发展网络安全产业体系，**促进拟态防御、数据加密**等网络安全技术应用。
- **提升数据安全保障水平。**建立健全数据安全治理体系，**增强重点行业数据安全保障能力。进一步强化个人信息保护，**规范身份信息、隐私信息、生物特征信息的采集、传输和使用，加强对收集使用个人信息的安全监管能力。

业务伴生风险

收集

- **瑞智华胜 涉嫌非法窃取用户信息30亿条**
- 部署SD程序，从运营商服务器抓取网络用户数据

存储

- **Facebook 4.19亿用户的电话信息被泄露**
- 数据库缺失加密保护

使用

- **圆通 10亿条用户信息数据被出售**
- 黑客通过凭据注入攻击收集

加工

- **某运营商被外包人员 80万用户数据被删除**
- 误操作导致数据丢失

传输

- **速贷之家 贩卖个人信息被罚320万元**
- 未取得受害人同意，向下游公司传输数据、非法盈利

提供

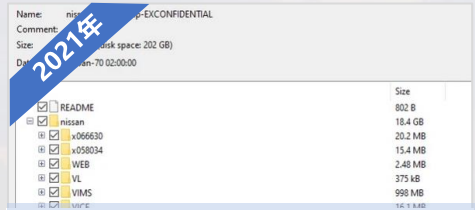
- **某股份制银行 泄露脱口秀演员信息被罚450万元**
- 银行未经授权，私自将其个人账户流水提供他人

公开

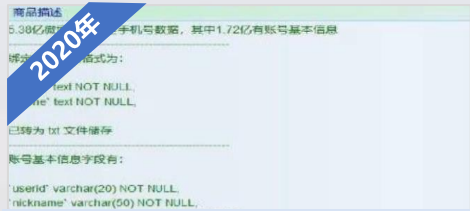
- **邯郸丛台区政府 泄露129位特困人员隐私**
- 敏感个人信息保护意识缺失，公开数据未加密处理

2亿条中国公民数据
在暗网被公开售卖

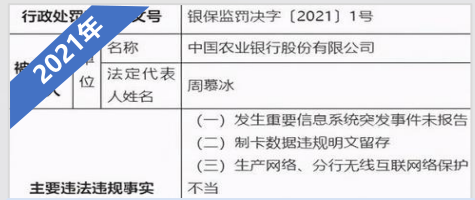
2元可买70张
明星素颜健康码照片被售卖



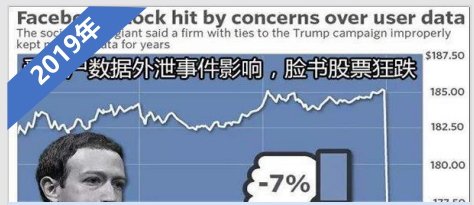
数据集约20GB
日产北美公司源代码泄露



5.38亿条数据，0.177 比特币
新浪微博数据泄露



农业银行被处罚420万
银保监会1号罚单



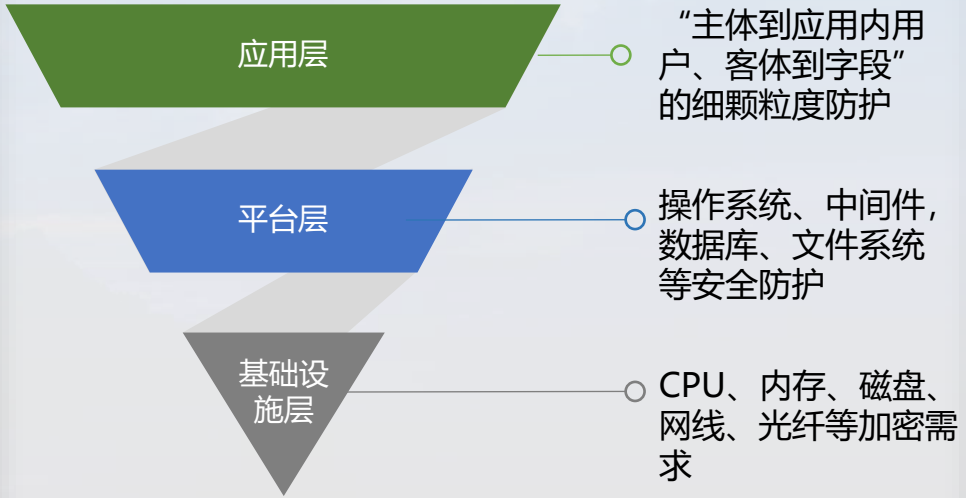
脸书被罚款50亿美元
股价暴跌

*上述案例来源于最高法院判决文书网站或权威网络

风险驱动密码

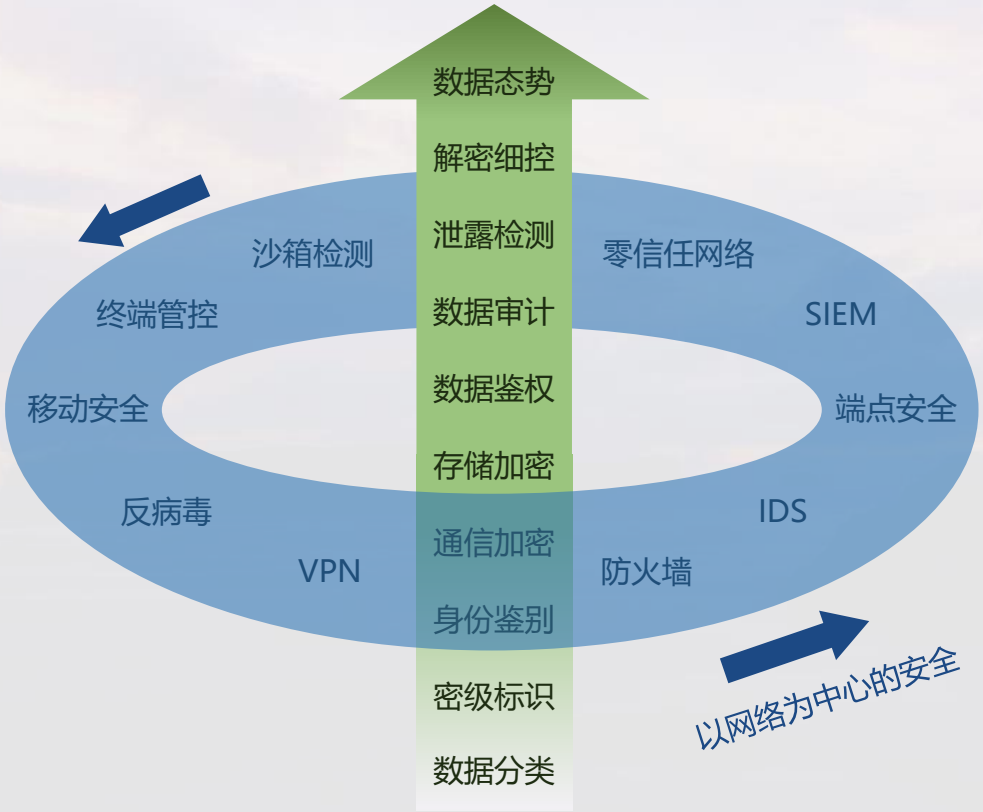
从时间维度，数据在全生命周期的密码需求

从空间维度，数据在不同层的不同密码防护颗粒度需求



密码融入业务

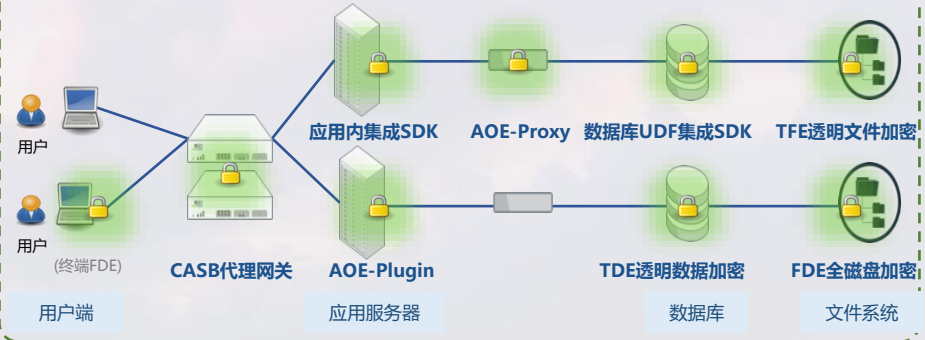
以数据为中心的安全



数据安全成为当前建设重点

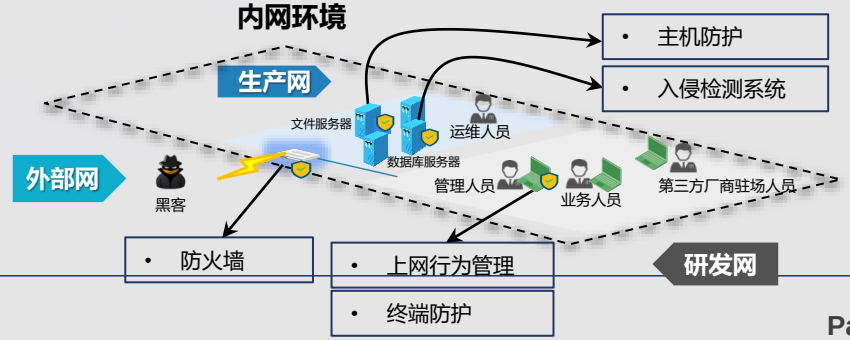
数据安全“追数据”

在**业务应用中**，面向全生命周期的数据，施加主动安全增强。



网络安全“按分区”

在**端点和网关**，面向网络元组和内容，施加攻击阻断和保护。

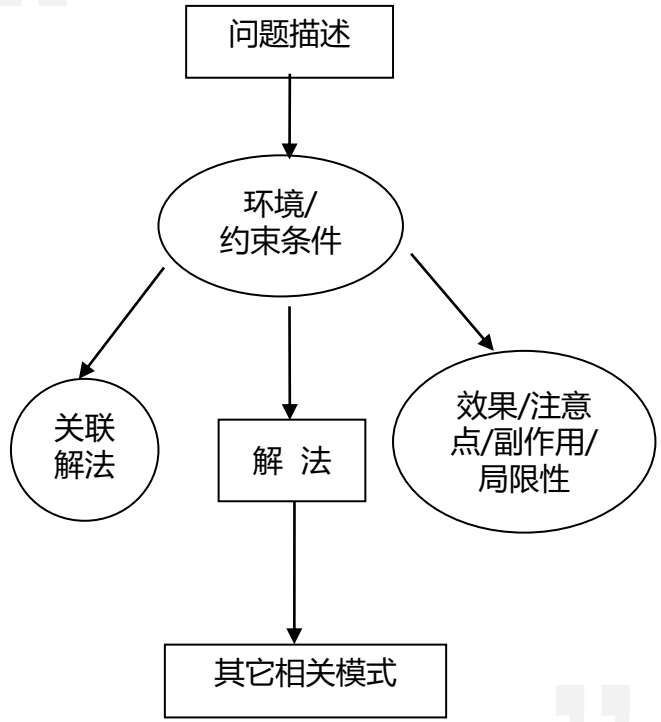


密码难点在于融合业务场景

密码应用模式让用户秒懂密码使用场景

- 模式是在特定环境中解决问题的一种方案。《设计模式：可复用面向对象软件的基础》描述了23种经典面向对象设计模式，创立了模式在软件设计中的地位。

- **模式经典定义**：每个模式都描述了一个在我们的环境中不断出现的问题，然后描述了该问题的解决方案的核心，通过这种方式，我们可以无数次地使用那些已有的解决方案，无需再重复相同的工作。
- **密码应用模式定义**：从信息化系统中，把密码应用设计提炼成可复用模式，并提供问题域解法的密码算法及协议清单，使用户通俗易懂，为甲方、密评单位、密码厂商、监管机构、ISV提供标准术语
- **密码应用模式的要素**
  - **威胁模型**
    - 威胁分析；环境/约束条件
    - 模式示例
  - **模式表述**
    - 解法；关联解法
    - 效果/注意点/副作用/局限性
    - 参考案例



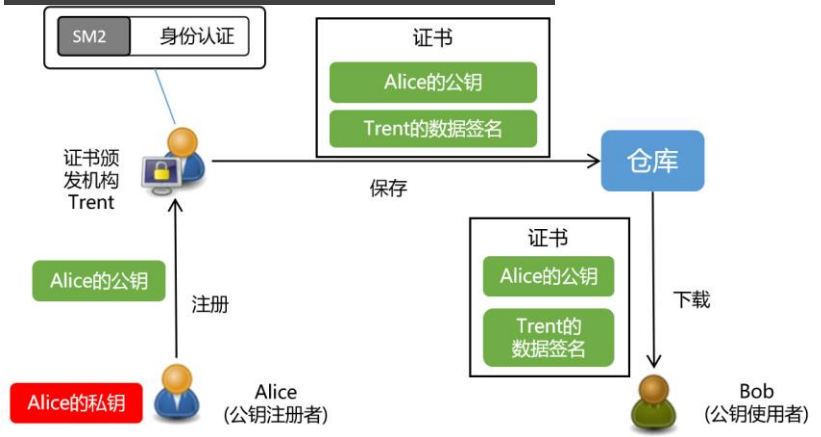
20种密码应用模式可覆盖主流场景

- 典型实现示例

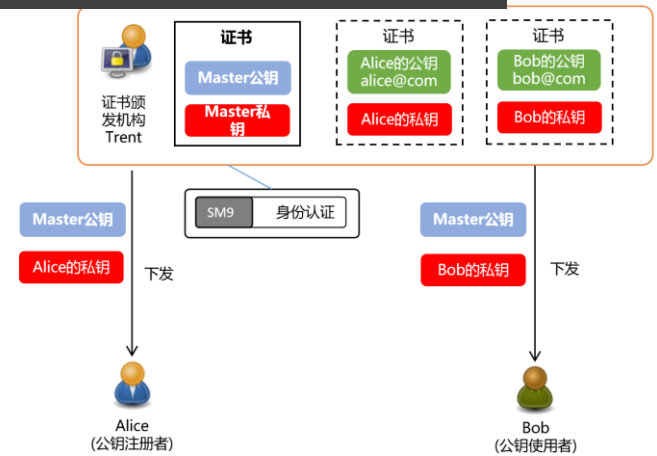
	身份鉴别及密钥管理	数据传输(通信安全)	数据存储(数据资产安全)	数据使用(数据共享与安全兼得)
应用层	③ 预共享密钥的身份鉴别 ④ 基于数字签名的身份鉴别 - 基于单一设备签名的身份鉴别 - 协同签名 -阈值签名	⑤ 离线通信消息加密 - PGP邮件加密 - S/MINE邮件加密 - Signal/OTR聊天加密 ⑥ 代理重加密受控分发消息	⑨ 应用内数据加密 - 应用内开发集成加密 - CASB代理网关加密 - AOE面向切面加密	⑫ 基于差分隐私的数据匿名化 ⑬ 基于属性加密的访问控制 ⑭ 锚点解密的防绕过数据安全 - TDF可控分享秘密信息 ⑮ 不可信环境中的数据运算 - FHE全同态加密 - MPC多方安全计算 - ZKP零知识证明、区块链隐私保护 ⑯ 可验证结果的计算外包 ⑰ 封装业务逻辑的可信运算环境 - 金融数据密码机
终端与基础设施层		⑦ 在线通信消息加密 - 基于SSL/TLS的HTTPS - VPN虚拟专用网络 - 链路密码机/网络密码机 ⑧ 可感知窃听的专线通信 - BB84量子密钥分发	⑩ 数据库存储加密 - DB-Proxy数据库代理加密 - 数据库UDF开发集成加密 - 数据库外挂加密 - TDE透明数据加密 ⑪ 文件存储加密 - TFE透明文件加密 - FDE全磁盘加密	⑱ 基于密码的数字水印追溯
基础密码产品	① PKI信任体系 - CA证书认证系统 - 安全认证网关 ② IBC信任体系			⑲ 基于密码校验的防篡改 - 电子签章 ⑳ 基于私钥签名的责任认定 - 签名验签服务器

一、身份鉴别及密钥管理

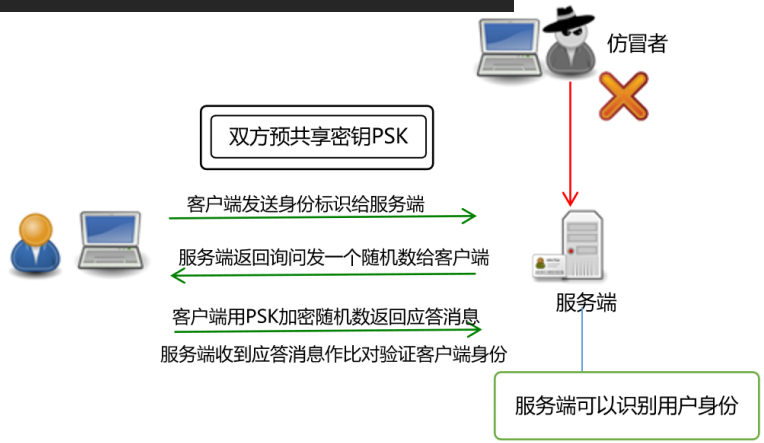
01-基于PKI的信任体系



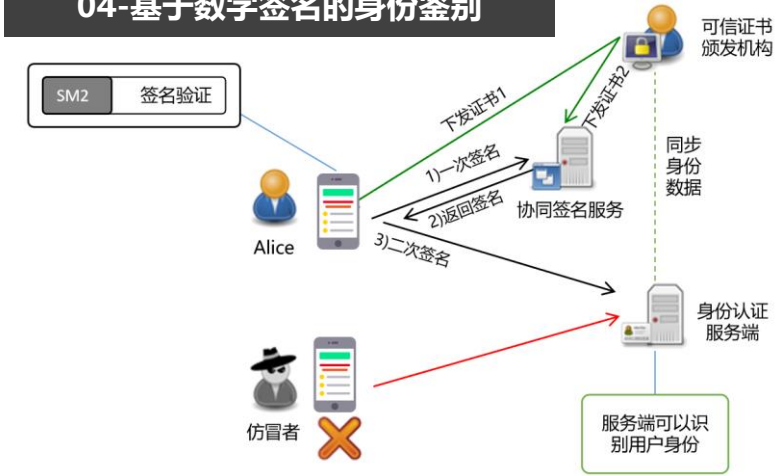
02-基于IBC的信任体系



03-预共享密钥的身份鉴别

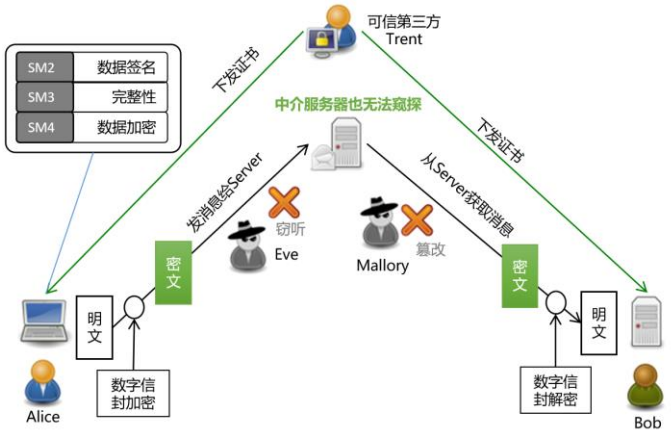


04-基于数字签名的身份鉴别

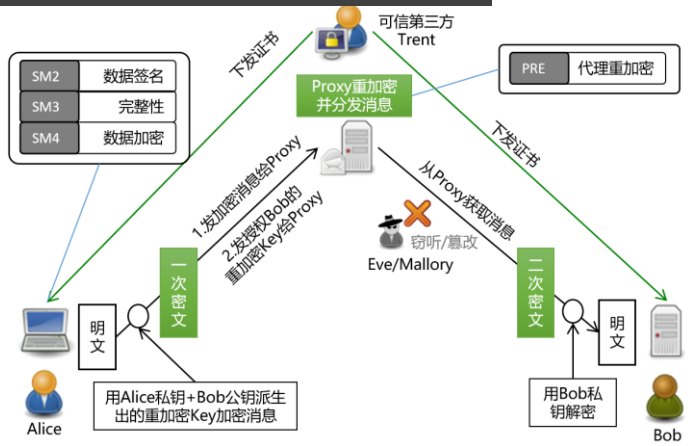


二、数据传输(通信安全)

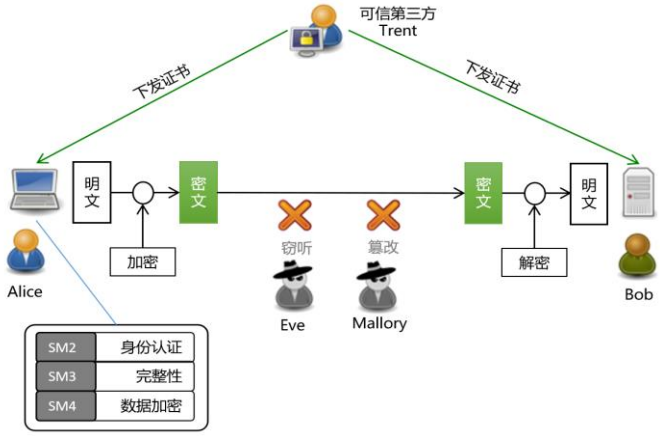
05-离线通信消息加密



06-代理重加密受控分发消息



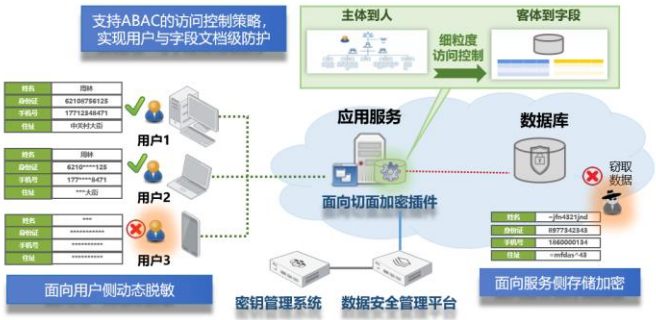
07-在线通信消息加密



08-可感知窃听的专线通信

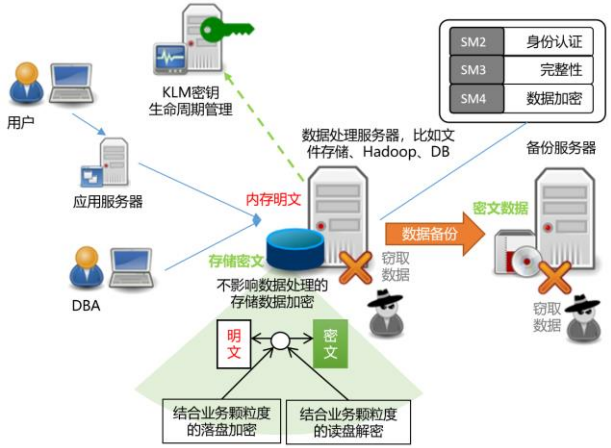
三、数据存储（数据资产安全）

09-AOE面向切面加密



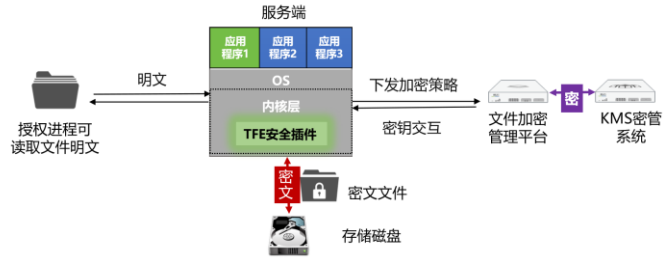
- 将安全组件与业务从技术上解耦、但又实现了能力融合交织，将密码应用从传统“外挂式”升级为“内嵌式”，有效提升内生安全能力
- 赋能企业风险识别、安全防御、安全检测、安全响应和安全恢复五维安全能力
- 有效防范“外部黑客”攻击、“内部高危”操作、“敏感数据”泄漏

10-数据库存储加密



- **解法：**结合文件存储、Hadoop或DB，对文件、块文件、表空间等进行加密
- **关联解法：**全磁盘加密：粒度粗，只能防拔盘；部署在应用与DB间的数据库加密网关：密文计算难以工业级交付
- **效果/注意点：**要结合目标文件系统、或数据库品牌版本等
- **参考案例：**TFE透明文件加密防商业文档泄露、Hadoop敏感数据块防泄露、数据库TDE防范库文件泄漏

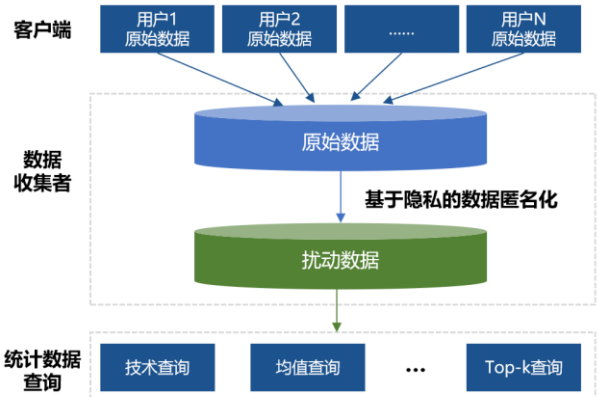
11-文件存储加密



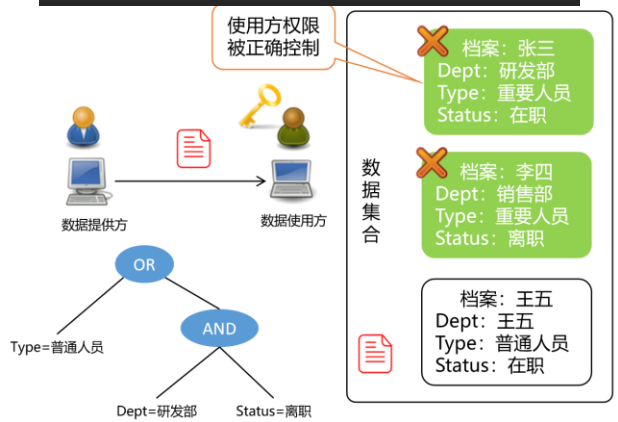
- 文件安全插件部署在文件服务器操作系统的驱动层，文件在被应用写入的过程中可对文件进行加密；在读取的过程中进行解密.
- 同时，文件进出存储磁盘的过程，都可以被插件记录，进行安全审计，进一步对审计记录进行综合分析，可以全方位获得敏感文件的被使用情况，及时发现存在的威胁。

四、数据使用（数据共享与安全兼得）

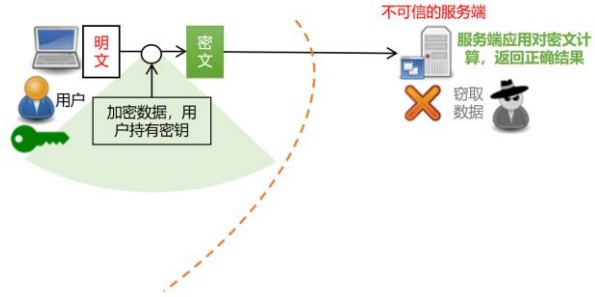
12-基于差分隐私的数据匿名化



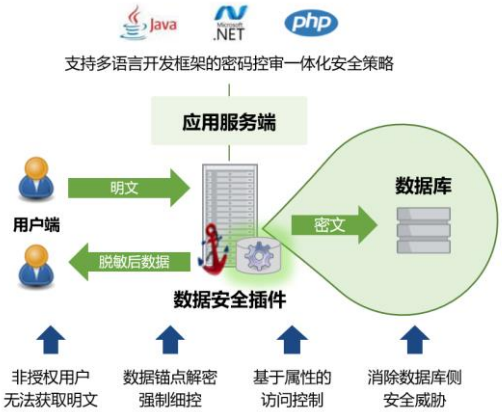
13 -基于属性加密的访问控制



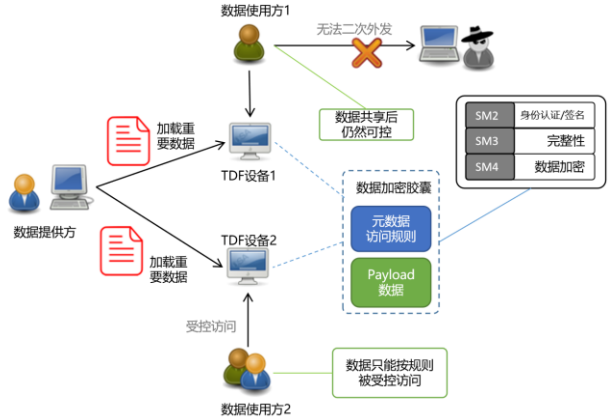
15-不可信环境中的数据运算



14-1 锚点解密的防绕过数据安全

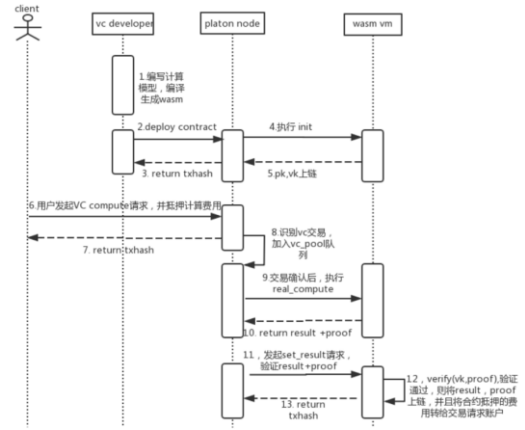


14-2 TDF可控分享秘密信息

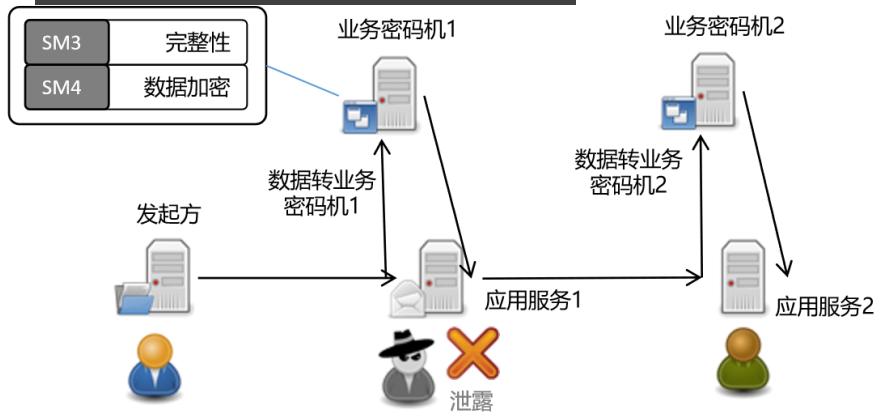


四、数据使用（数据共享与安全兼得）

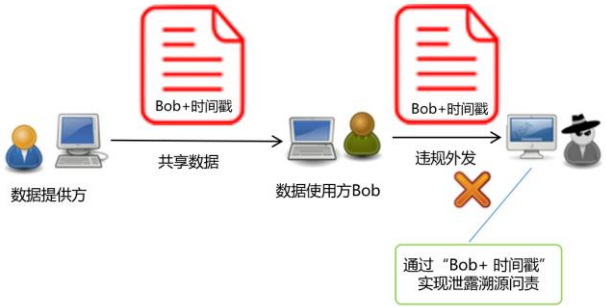
16-可验证结果的计算外包



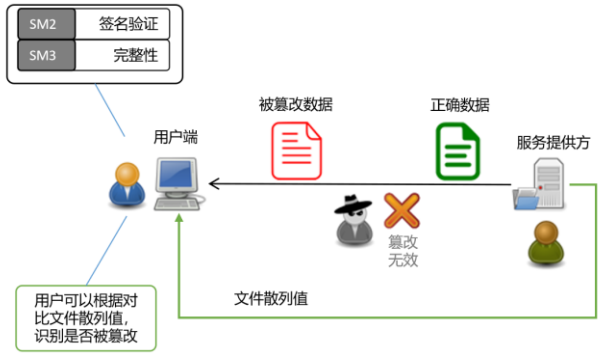
17-封装业务逻辑的可信运算环境



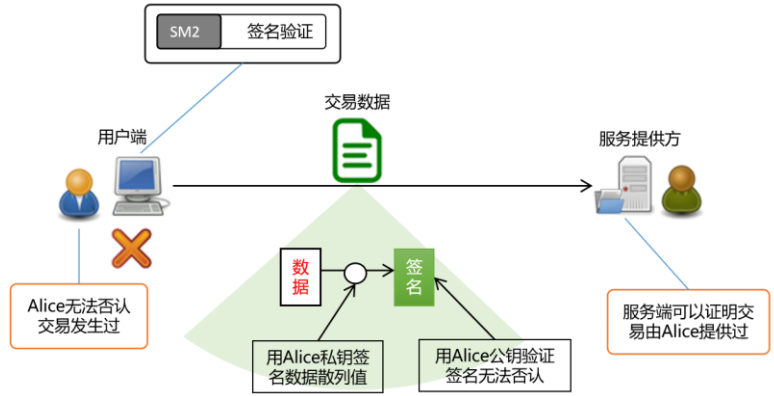
18-基于密码的数字水印追溯



19-基于密码校验的防篡改



20-基于私钥签名的责任认定



密码算力创造业务价值

静态数据（或信息）的基础属性模型

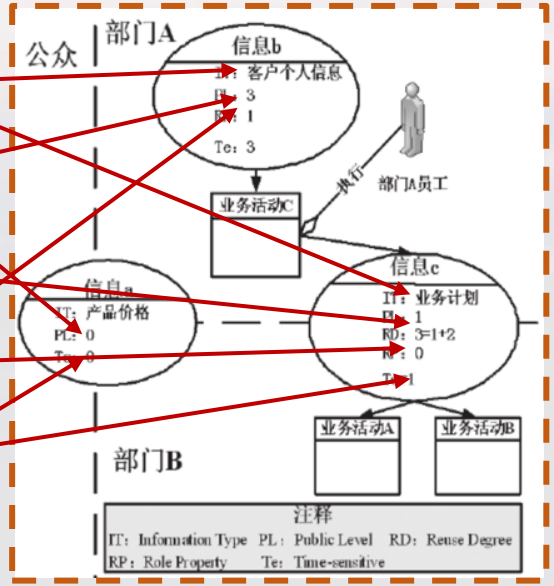
保密性和价值 — 信息类别

多大范围内公开及访问权限级别 — 公开程度

保密性和完整性 — 复用程度

责任追究以及隐私 — 人员属性

保密性、价值、时间 — 时间敏感度



动态数据（或信息）的流程属性模型

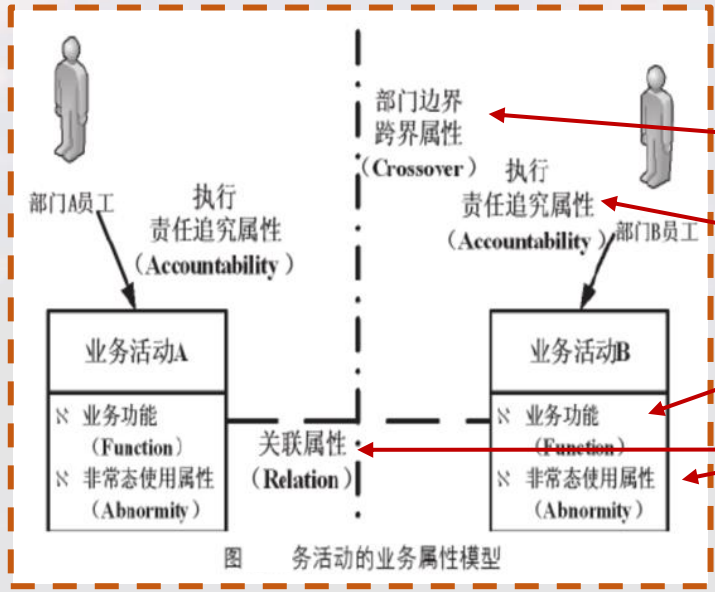


图　务活动的业务属性模型

可追溯、定责

不可抵赖性

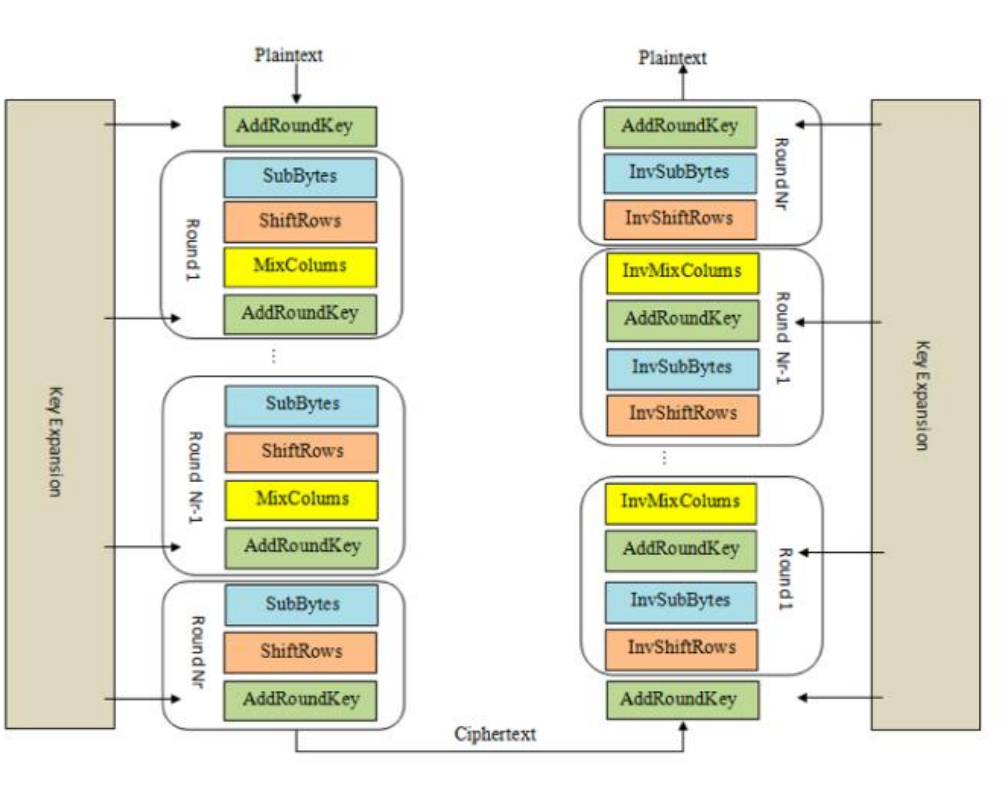
重要性、保密性

可用性

02 国密算法优化实践探索

AES算法介绍

AES算法



AES轮函数

基于AES-NI的AES算法实现效率极高

- Intel Advanced Encryption Standard New Instructions
- Extension to the x86 Instruction Set Architecture
- Proposed by Intel in 2008, now in Intel, AMD and ARM CPUs
- The magic part of the super speed of AES

- Six instructions
  - _mm_aesenc_si128
  - _mm_aesenclast_si128
  - _mm_aesdec_si128
  - _mm_aesdeclast_si128
  - _mm_aesimc_si128
  - _mm_aeskeygenassist_si128

Operation

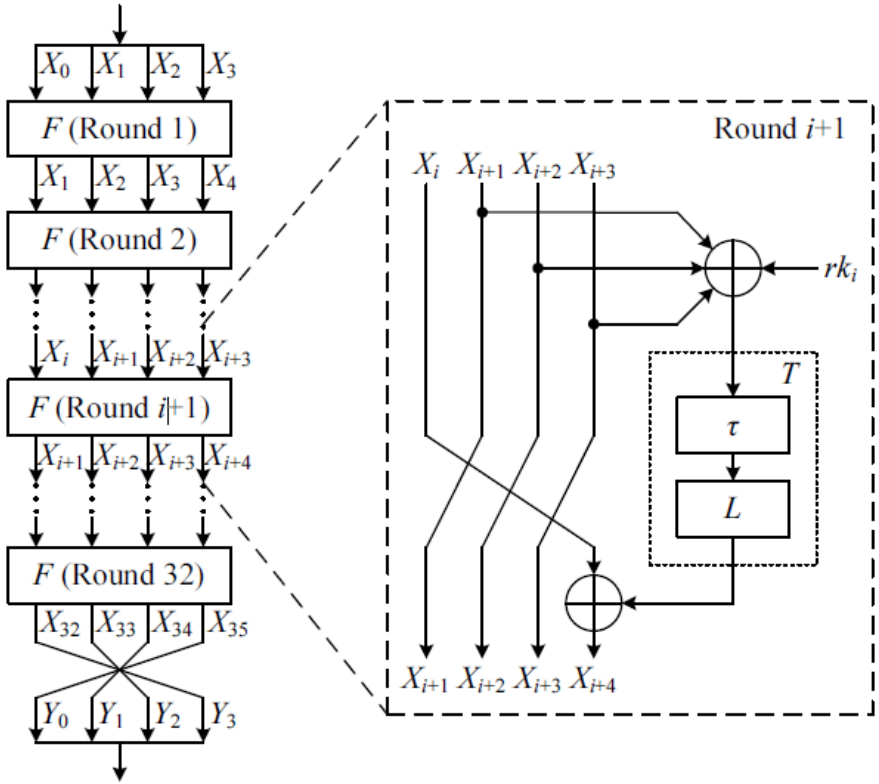
```
state := a
a[127:0] := ShiftRows(a[127:0])
a[127:0] := SubBytes(a[127:0])
dst[127:0] := a[127:0] XOR RoundKey[127:0]
```

Performance

Architecture	Latency	Throughput (CPI)
Skylake	4	1
Broadwell	7	1
Haswell	7	1
Ivy Bridge	8	1

How Fast is AES-NI?

SM4算法介绍



- Published in 2006，Initially used in WAPI
- GM/T002-2012，GB/T 32907-2016
- 128-bit block，128-bit key
- Unbalanced Feistel，32 rounds，32-bit round key
- xor, rotation, sbox

- Round Function $F(\cdot)$

$$F(X_0, X_1, X_2, X_3, rk) = X_0 \oplus T(X_1 \oplus X_2 \oplus X_3 \oplus rk)$$
$$X_0, X_1, X_2, X_3, rk \in Z_2^{32}$$

- Composed Transformation $T(\cdot)$

$$T(\cdot) = L(\tau(\cdot))$$

- Non-Linear Transformation $\tau(\cdot)$

$$B = \tau(A) = \left(Sbox(x_0), Sbox(x_1), Sbox(x_2), Sbox(x_3)\right)$$
$$A = (x_0, x_1, x_2, x_3), B = (y_0, y_1, y_2, y_3),$$
$$x_0, x_1, x_2, x_3 \in Z_2^8, y_0, y_1, y_2, y_3 \in Z_2^8$$

- Linear Transformation $L(\cdot)$

$$L(B) = B \oplus (B \lll 2) \oplus (B \lll 10) \oplus (B \lll 18) \oplus (B \lll 24)$$
$$L'(B) = B \oplus (B \lll 13) \oplus (B \lll 22)$$

SM4软件实现优化初步思路

- Eliminate Loops: unroll loops with macros of C/C++
- Avoid Moving Data Inside Memory
  - Branch rotation costs nothing
- Know Your CPU: Cache System, Intrinsic Instructions, …
- Try Different Compilers with Different Optimization Options
- Parallel at Different Levels, adapted according to different service context
- Equivalent Transformation of SM4 Algorithm
- Learning From Hardware Implementation
- Security Side: Constant Time

SM4基础实现

```
#include <climits>
#define bitsof(x) (CHAR_BIT * sizeof(x))
#define rol(x, n) (((x) << (n)) | ((x) >> ((bitsof(x) - (n)))))
#define ror(x, n) (((x) >> (n)) | ((x) << ((bitsof(x) - (n)))))


#define SM4_CORE_4R(rk0, rk1, rk2, rk3)                    \
 do {                                                      \
            tmp = m[1] ^ m[2] ^ m[3] ^ rk0, m[0] ^= sm4_t_sub(tmp); \
            tmp = m[2] ^ m[3] ^ m[0] ^ rk1, m[1] ^= sm4_t_sub(tmp); \
            tmp = m[3] ^ m[0] ^ m[1] ^ rk2, m[2] ^= sm4_t_sub(tmp); \
            tmp = m[0] ^ m[1] ^ m[2] ^ rk3, m[3] ^= sm4_t_sub(tmp); \
 } while (0)
```

```
void sm4_enc_core(u32t *m, const u32t *rk) {
        u32t tmp;
        SM4_CORE_4R(rk[0], rk[1], rk[2], rk[3]);
        SM4_CORE_4R(rk[4], rk[5], rk[6], rk[7]);
        SM4_CORE_4R(rk[8], rk[9], rk[10], rk[11]);
        SM4_CORE_4R(rk[12], rk[13], rk[14], rk[15]);
        SM4_CORE_4R(rk[16], rk[17], rk[18], rk[19]);
        SM4_CORE_4R(rk[20], rk[21], rk[22], rk[23]);
        SM4_CORE_4R(rk[24], rk[25], rk[26], rk[27]);
        SM4_CORE_4R(rk[28], rk[29], rk[30], rk[31]);
        tmp = m[0], m[0] = m[3], m[3] = tmp;
        tmp = m[1], m[1] = m[2], m[2] = tmp;
}
```

用并行改进SM4基础实现

- We are working on 64-bit system, we have 64-bit register, why just working with 32-bit variables?
- Processing two blocks in parallel with 64-bit register

- Expecting a 2x speed up?
- Usually NOT that Lucky
  - Formatting input and output messages
  - SBox substitution has to be done in a serialized way, e.g. one by one
- But we do gain speed improvement:

  30.7 cycles/byte → 23.5 cycles/byte

- What if we have 128-bit, 256-bit or even 512-bit register?

基于T-Table的SM4实现改进

- Composed Transformation: $T(\cdot) = L(\tau(\cdot))$
- SBox is followed by linear transformation
- Fold the linear transformation into SBox
  - Incurs (almost) no extra cost on SBox Operation
  - Eliminate the linear transformation: saves 4 rotations per round

$$L(B) = B \oplus (B \lll 2) \oplus (B \lll 10) \oplus (B \lll 18) \oplus (B \lll 24)$$

- Of Course, SBox is now different
  - One 8x8 SBox to Four 8x32 SBox (4KB in total)
  - $L(\tau(\cdot)) \rightarrow Sbox_0(x_0) \oplus Sbox_1(x_1) \oplus Sbox_2(x_2) \oplus Sbox_3(x_3)$

基于T-Table和并行的SM4实现改进

- Composed Transformation: $T(\cdot) = L(\tau(\cdot))$
- We are working on 64-bit system
- We have 128-bit, 256-bit registers
- Processing 2/4/8 blocks in parallel

- Expecting a 2x, 4x, 8x speed up?
- NOT that Lucky
  - Formatting input and output messages
  - SBox substitution has to be done in a serialized way, e.g. one by one
- We do gain speed improvement:

32 → 64 → 128 → 256

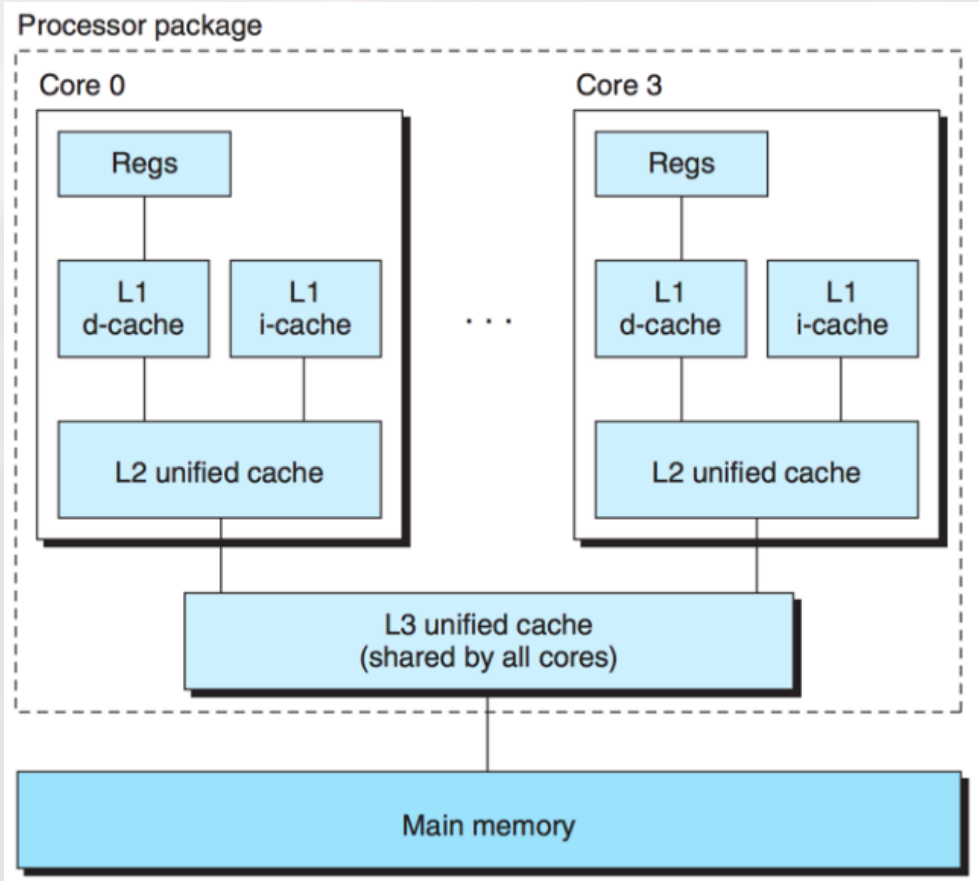
19.9 →13.8 → 11.2 → 9.6

基于Cache的侧信道攻击

- Single Instruction Multiple Data
- Similar technique has been applied to AES software impl.
- Which is then introduced into OpenSSL
- Big table lead to non-constant encryption time due to cache miss …

Daniel J. Bernstein (DJB) in 2005: demonstrated complete AES key recovery from known-plaintext timings of a network server on another computer

Intel Core i7 Cache Hierarchy

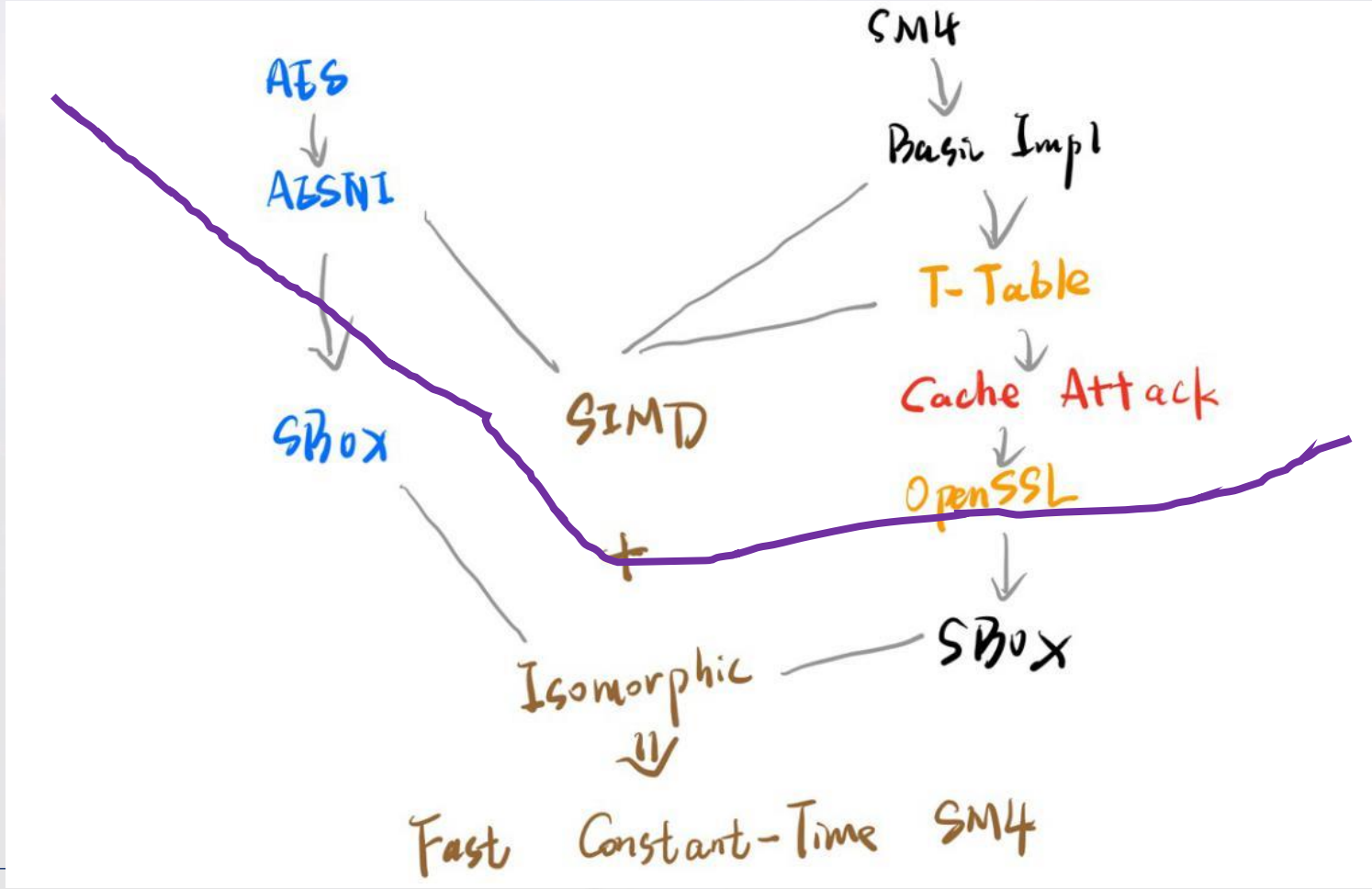


OpenSSL提供了一种保护思路

```
109  static ossl_inline uint32_t SM4_T_slow(uint32_t X)
110  {
111      uint32_t t = 0;
112
113      t |= ((uint32_t)SM4_S[(uint8_t)(X >> 24)]) << 24;
114      t |= ((uint32_t)SM4_S[(uint8_t)(X >> 16)]) << 16;
115      t |= ((uint32_t)SM4_S[(uint8_t)(X >> 8)]) << 8;
116      t |= SM4_S[(uint8_t)X];
117
118      /*
119       * L linear transform
120       */
121      return t ^ rotl(t, 2) ^ rotl(t, 10) ^ rotl(t, 18) ^ rotl(t, 24);
122  }
123
124  static ossl_inline uint32_t SM4_T(uint32_t X)
125  {
126      return SM4_SBOX_T[(uint8_t)(X >> 24)] ^
127             rotl(SM4_SBOX_T[(uint8_t)(X >> 16)], 24) ^
128             rotl(SM4_SBOX_T[(uint8_t)(X >> 8)], 16) ^
129             rotl(SM4_SBOX_T[(uint8_t)X], 8);
130  }
```

```
194      /*
195       * Uses byte-wise sbox in the first and last rounds to provide some
196       * protection from cache based side channels.
197       */
198      SM4_RNDS( 0,  1,  2,  3, SM4_T_slow);
199      SM4_RNDS( 4,  5,  6,  7, SM4_T);
200      SM4_RNDS( 8,  9, 10, 11, SM4_T);
201      SM4_RNDS(12, 13, 14, 15, SM4_T);
202      SM4_RNDS(16, 17, 18, 19, SM4_T);
203      SM4_RNDS(20, 21, 22, 23, SM4_T);
204      SM4_RNDS(24, 25, 26, 27, SM4_T);
205      SM4_RNDS(28, 29, 30, 31, SM4_T_slow);
206
207      store_u32_be(B3, out);
208      store_u32_be(B2, out + 4);
209      store_u32_be(B1, out + 8);
210      store_u32_be(B0, out + 12);
```

SIMD加速SM4实现的路线图



分析AES SBox

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 7. S-box: substitution values for the byte xy (in hexadecimal format).

AES Sbox内部数学结构

- Single Instruction Multiple Data
- $SBox_{AES}(x) = A \cdot x^{-1} + 0x63,$
- $x \in GF(2^8), f(x) = x^8 + x^4 + x^3 + x + 1$

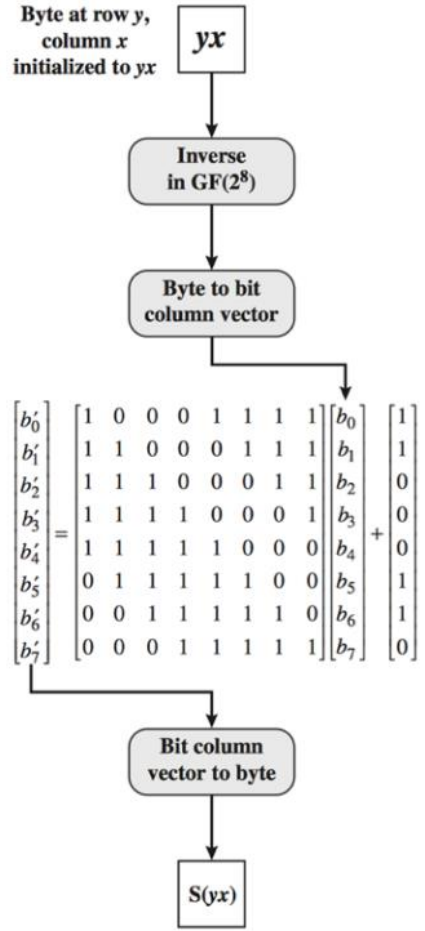
- Gen AES SBox with Sage

```
#generate aes's sbox
for v in range(0, 0x100):
  v_inv = aes_field_inv(v)

  v_inv_list = [int(d) for d in format(v_inv, "08b")]
  v_inv_list.reverse() # lower bit comes first
  v_inv_vec = matrix(GF(2),8,1, v_inv_list)

  res_matrix = A * v_inv_vec + C
  res = ZZ(res_matrix.list(), base=2)

  print "%02x" % res,
  if (v+1) % 16 == 0:
    print
```



分析SM4 SBox

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	D6	90	E9	FE	CC	E1	3D	B7	16	B6	14	C2	28	FB	2C	05
1	2B	67	9A	76	2A	BE	04	C3	AA	44	13	26	49	86	06	99
2	9C	42	50	F4	91	EF	98	7A	33	54	0B	43	ED	CF	AC	62
3	E4	B3	1C	A9	C9	08	E8	95	80	DF	94	FA	75	8F	3F	A6
4	47	07	A7	FC	F3	73	17	BA	83	59	3C	19	E6	85	4F	A8
5	68	6B	81	B2	71	64	DA	8B	F8	EB	0F	4B	70	56	9D	35
6	1E	24	0E	5E	63	58	D1	A2	25	22	7C	3B	01	21	78	87
7	D4	00	46	57	9F	D3	27	52	4C	36	02	E7	A0	C4	C8	9E
8	EA	BF	8A	D2	40	C7	38	B5	A3	F7	F2	CE	F9	61	15	A1
9	E0	AE	5D	A4	9B	34	1A	55	AD	93	32	30	F5	8C	B1	E3
A	1D	F6	E2	2E	82	66	CA	60	C0	29	23	AB	0D	53	4E	6F
B	D5	DB	37	45	DE	FD	8E	2F	03	FF	6A	72	6D	6C	5B	51
C	8D	1B	AF	92	BB	DD	BC	7F	11	D9	5C	41	1F	10	5A	D8
D	0A	C1	31	88	A5	CD	7B	BD	2D	74	D0	12	B8	E5	B4	B0
E	89	69	97	4A	0C	96	77	7E	65	B9	F1	09	C5	6E	C6	84
F	18	F0	7D	EC	3A	DC	4D	20	79	EE	5F	3E	D7	CB	39	48

注：输入'EF'，则经S盒后的值为表中第E行和第F列的值，Sbox(EF)=84。

SM4 Sbox内部数学结构

- Specification contains no info. how the SBox is construted
- Internal structure of the SBox was reverse engineered later
  - Analysis of the SMS4 Block Cipher @ ACISP' 07 by Liu et al.

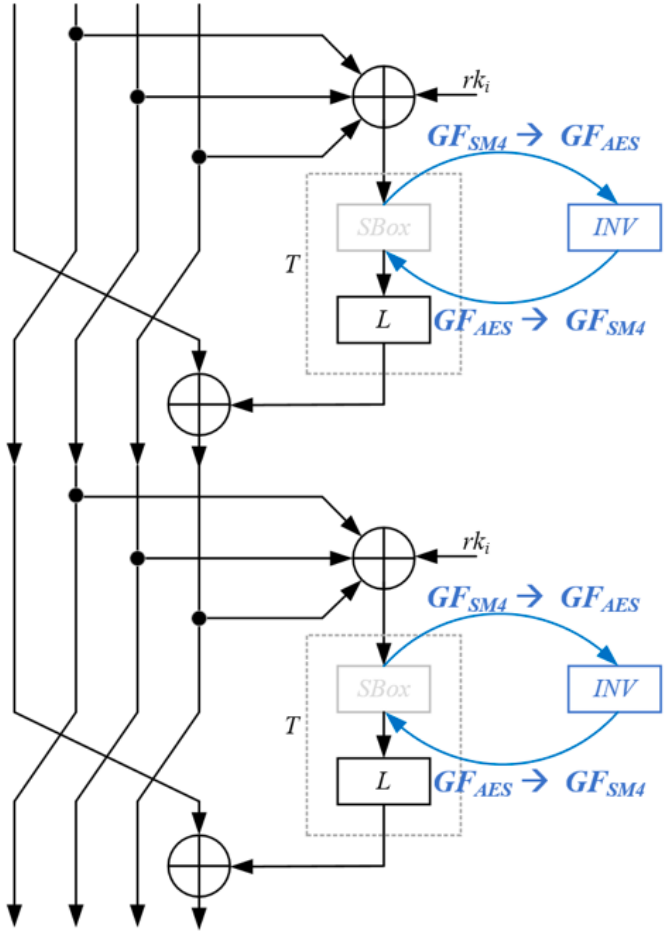
$$SBox_{SM4}(x) = (x \cdot A_1 + C_1)^{-1} \cdot A_2 + C_2$$

  - Algebraic Cryptanalysis of SMS4: Grobner Basis Attack and SAT Attack Compared @ ICISC' 09 by Erickson et al.

$$SBox_{SM4}(x) = (x \cdot A + C)^{-1} \cdot A + C$$

- $SBox_{SM4}(x) = \boldsymbol{Inv}(x \cdot A_1 + C_1) \cdot A_2 + C_2,$
- $x \in GF(2^8), g(x) = x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$

这种优化思路，被Intel远见布局



US 20150341168A1

(19) **United States**

(12) **Patent Application Publication**

GUERON

(10) **Pub. No.: US 2015/0341168 A1**

(43) **Pub. Date: Nov. 26, 2015**

(54) **TECHNOLOGIES FOR MODIFYING A FIRST CRYPTOGRAPHIC CIPHER WITH OPERATIONS OF A SECOND CRYPTOGRAPHIC CIPHER**

(71) Applicant: **Intel Corporation**, Santa Clara, CA (US)

(72) Inventor: **SHAY GUERON**, Haifa (IL)

(73) Assignee: **Intel Corporation**, Santa Clara, CA (US)

(21) Appl. No.: **14/283,955**

(22) Filed: **May 21, 2014**

Publication Classification

(51) **Int. Cl.**

H04L 9/08	(2006.01)
H04L 9/14	(2006.01)

(52) **U.S. Cl.**

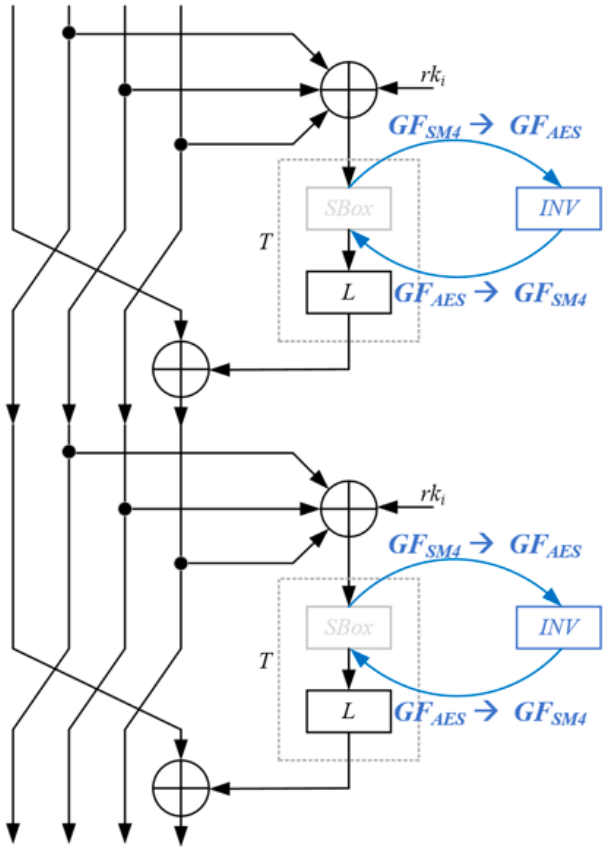
CPC ***H04L 9/0838*** (2013.01); ***H04L 9/14*** (2013.01); *H04L 2209/24* (2013.01)

(57) **ABSTRACT**

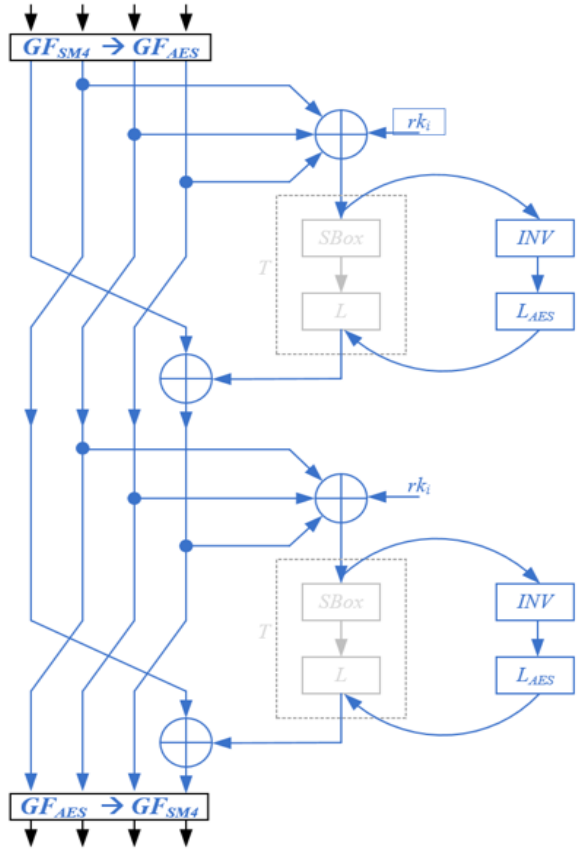
Generally, the present disclosure provides technology modifying a first cryptographic cipher with one or more operations of a second cryptographic cipher. In some embodiments the technology leverages a mathematical relationship between representations of data used in the first and second ciphers to enable the substitution of one or more operations of the first cipher with one or more operations of the second cipher. The resulting modified cipher may in some instances exhibit improved performance and or security, relative to the unmodified first cipher. Methods, computer readable media, and apparatus including or utilizing the technologies are also described.

炼石显著改进Intel专利安全性和效率

Intel专利



炼石PCT专利



炼石SM4加解密可能是目前业界最快实现，对业务不影响

基于PKI的数字信封加密

协同签名身份鉴别

密钥派生接口

EVP国密接口

FPE格式保留加密

Java

C#

python

Ruby

Programming

C++

GO

SM2

SM3

SM4

SM9

ZUC

Windows 10

Linux

Mac OS

android

iOS

基于SM4的保留格式加密（FPE）

加密10亿条手机号，仅耗时20秒

2018年保密技术交流大会暨产品博览会
Information Security Technology Conference & Expo 2018

炼石
CipherGateway

天下密码 唯快不破

炼石独有PCT专利
打破Intel垄断

单颗CPU上SM4加解密
突破 140Gbps!
相当于每秒可加密6部3GB大小的高清电影

移动端SM4加解密
突破1.7Gbps!
相当于每秒可加密100多张2MB大小的高清照片

谢谢大家

安天冬训营 wtc.antiy.cn