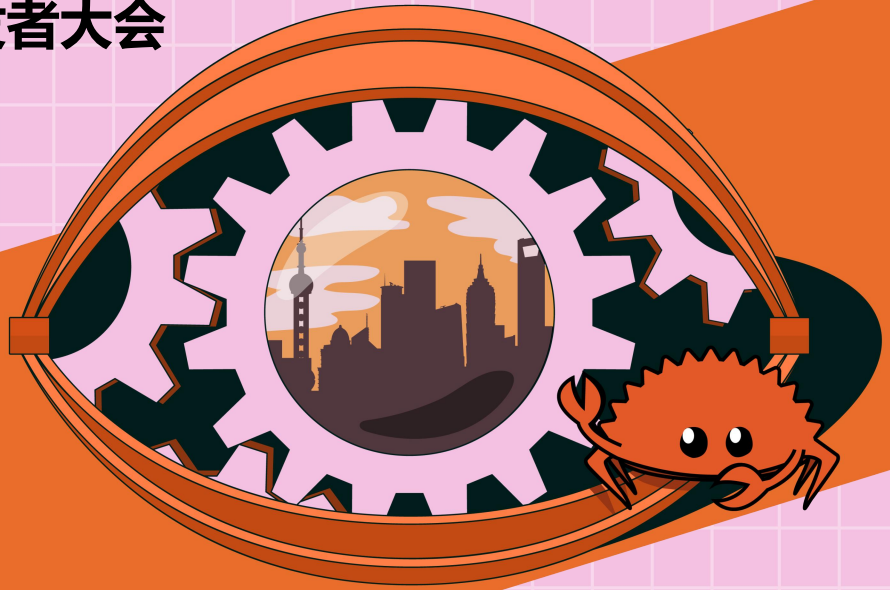
RUST CHINA CONF 2023

第三届中国Rust开发者大会



6.17-6.18 @Shanghai

Title

简谈 Rust 与国密 TLS

Introduction on Rust and SM TLS

王江桐
wangjiangtong@huawei.com
华为 公共开发部 嵌入式软件能力中心

Title

简谈 Rust 与国密 TLS

Introduction on Rust and Shangmi TLS

就职于华为，目前正在使用 Rust 开发密码相关模块。
Rustacean 在华为。

王江桐

wangjiangtong@huawei.com

华为 公共开发部 嵌入式软件能力中心

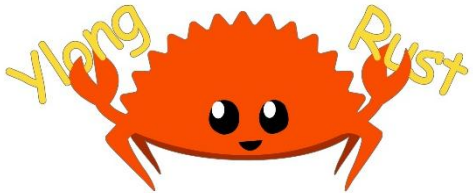


Table of Contents

目录

Section #1

国密算法总览

Overview to Shangmi Cryptography

- 密码算法安全目标
- 密码算法分类
- 国密套件总览

密码算法安全目标
Security Goals

- 通常来说，通过加密方式，对于信息的传输，我们希望达成以下五个目标：

机密性（Confidentiality）	保证信息私密性和保密性
真实性（Authentication）	确保信息来自正确身份的对象
完整性（Integrity）	信息没有被篡改
接入控制（Access control）	避免资源滥用
可获性（Availability）	资源可以被使用

- 不同的攻击方式可能针对于不同的目标进行攻击。比如DoS（拒绝服务Denial of Service）攻击就是针对可获性进行的攻击，使计算机或网络无法提供正常的服务。

密码算法分类与应用

Categories and Applications of Cryptography

国密概述
Introduction of Shangmi Cryptography

- 商用密码是中华人民共和国政府用于非国家机密信息保护所采用的一系列密码技术和密码产品的总称，其相关技术部分为国家秘密。商用密码的研发及使用由国家密码管理局统一管理。

- 根据国家密码管理局2007年4月23日公布的《商用密码产品使用管理规定》和《境外组织和个人在华使用密码产品管理办法》：

使用者	密码产品产地	限制使用情况
境外组织、个人	境内	可以使用
	境外	需要《使用境外生产的密码产品准用证》
中国法人、组织、公民	境内	可以使用
	境外	不得使用

Section #2

国密算法与协议介绍

Introduction to Shangmi Algorithms and Protocols

- 国密套件算法简介
- 国密 TLS 简介

来源：国密算法加密芯片，支持国密全套件等安全算法，http://www.bjlcs-tech.com/article/95.html

国密套件总览

List of Shangmi Cryptography

算法	算法标准	功能	类型	安全位数(bit)	对应算法	是否公开	应用
Sm1	/	分组加解密	对称加密	128	AES128	否，仅以 IP 核的形式存在于芯片中	智能 IC 卡、智能密码钥匙、加密卡、加密机等
Sm2	GB/T 32918-2016 ISO/IEC 10118-3:2018	ECC加解密，签名验签，密钥交换	非对称加密	128	ECC	是	TLCP、区块链等场景，用于签名验签等
Sm3	GM/T 0004-2012 ISO/IEC 10118-3:2018	计算密码杂凑	哈希	256	SHA256	是	TLCP、数字签名及验证、消息认证码生成及验证、随机数生成、密钥扩充
Sm4	GM/T 0002-2012 ISO/IEC WD1 18033-3/AMD2	分组加解密	分组加密	128	AES128，但是更多次轮询	是	TLCP、消息加解密，用于替代 DES/AES 等国际算法
Sm7	/	分组加解密	分组加密	128		否，仅以 IP 核的形式存在于芯片中	卡证类、票务类、支付与通卡类应用
Sm9	GM/T 0044-2016 ISO/IEC 10118-3:2018	标识密码算法：签名校验，密钥交换，密钥封装与加解密	非对称加密	128		是	TLCP，适用于新兴应用的安全保障（云、智能终端、物联网），系统可以提供身份标识
ZUC	GB/T 33133-2021 3GPP TS 35.221	对称加密算法	流加密	128	EEA3 & EIA3	是	国际组织 3GPP 推荐为 4G 无线通信的第三套国际加密和完整性的标准算法，为ISO/IEC 国际标准

SM1

Introduction of SM1

- SM1 是分组加密算法，实现对称加密，分组长度和密钥长度都为 128 位，对长消息进行加解密时，若消息长度过长，需要进行分组，如果消息长度不足，则要进行填充。
  - 保证数据机密性。

- 算法安全保密强度及相关软硬件实现性能与 AES 相当，该算法不公开，仅以 IP 核的形式存在于芯片中，调用该算法时，需要通过加密芯片的接口进行调用。

- 采用该算法已经研制了系列芯片、智能 IC 卡、智能密码钥匙、加密卡、加密机等安全产品，广泛应用于电子政务、电子商务及国民经济的各个应用领域（包括国家政务通、警务通等重要领域）。

SM2

Introduction of SM2

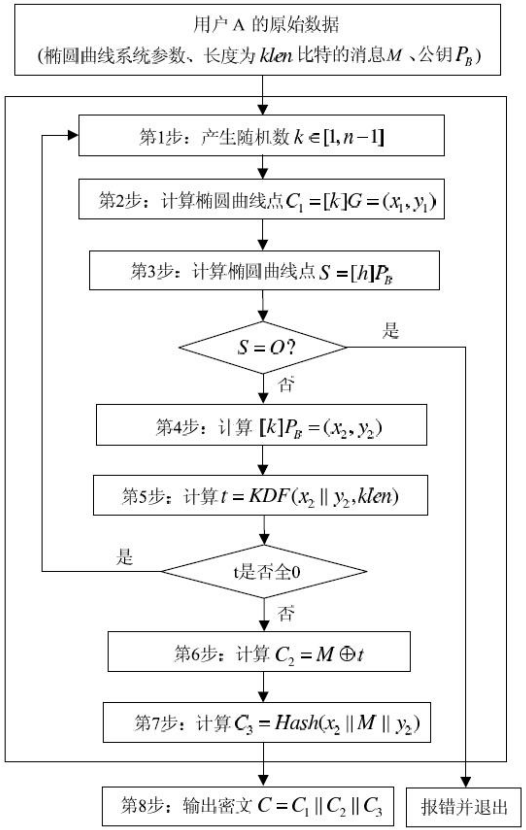
- SM2 为椭圆曲线（ECC）公钥加密算法，非对称加密，提供加解密、数字签名、证书生成、密钥交换功能。由于以上用例，也常用于区块链或网络安全密码协议，如SSL/TLS、VPN。
  - 保证数据机密性、真实性和完整性。

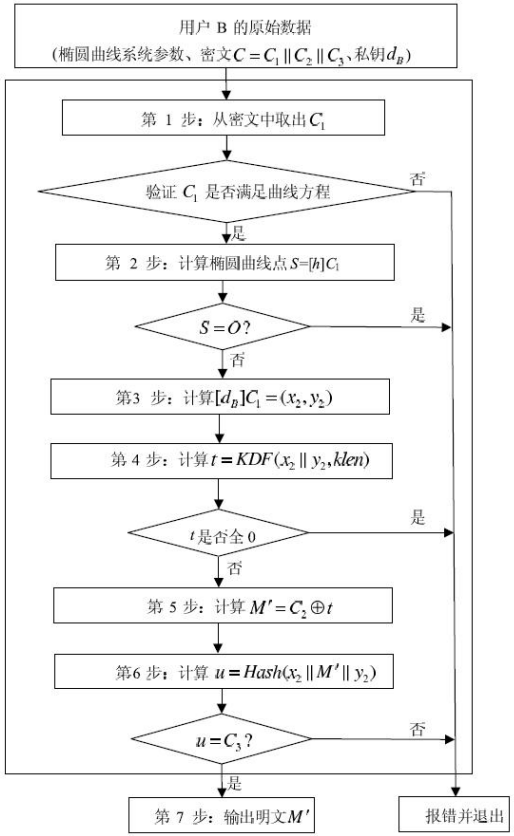
- SM2 算法和 RSA 算法都是公钥加密算法，SM2 算法是一种更先进安全的算法，其性能与安全性优于RSA，在我们国家商用密码体系中被用来替换 RSA 算法。
  - 椭圆曲线可使用更少的运算位数来达成与RSA相等的安全性
  - 椭圆曲线与RSA的安全性都依赖于离散对数问题的复杂程度
    - 离散对数问题：已知数A，B，且A = B^n，求数n

Bits of security	Symmetric key algorithms	FFC (e.g., DSA, D-H)	IFC (e.g., RSA)	ECC (e.g., ECDSA)
80	2TDEA[18]	$L = 1024$ $N = 160$	$k = 1024$	$f = 160\text{-}223$
112	3TDEA	$L = 2048$ $N = 224$	$k = 2048$	$f = 224\text{-}255$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256\text{-}383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384\text{-}511$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

SM2 加解密算法：流程图

Flowchart of SM2 Encrypt and Decrypt Algorithm





SM3

Introduction of SM3

- SM3 为密码杂凑算法，采用密码散列（hash）函数标准，用于替代 MD5/SHA-1/SHA-2 等国际算法，是在 SHA-256 基础上改进实现的一种算法，消息分组长度为 512 位，摘要值长度为 256 位，其中使用了异或、模、模加、移位、与、或、非运算，由填充、迭代过程、消息扩展和压缩函数所构成。
  - 保证信息的完整性。

- 在商用密码体系中，SM3 主要用于数字签名及验证、消息认证码生成及验证、随机数生成、密钥扩充等。据国家密码管理局表示，其安全性及效率要高于 MD5 算法和 SHA-1 算法，与 SHA-256 相当。

- SM3 将对长度为l(l < 2^64) 比特的消息 m，经过填充和迭代压缩，生成杂凑值，杂凑值长度为256比特。

SM3

Introduction of SM3

5.4.3 密钥派生函数

密钥派生函数的作用是从一个共享的秘密比特串中派生出密钥数据。在密钥协商过程中，密钥派生函数作用在密钥交换所获共享的秘密比特串上，从中产生所需的会话密钥或进一步加密所需的密钥数据。

密钥派生函数需要调用密码杂凑函数。

设密码杂凑函数为$H_v(\ )$，其输出是长度恰为v比特的杂凑值。

密钥派生函数$KDF(Z, klen)$：

输入：比特串Z，整数$klen$(表示要获得的密钥数据的比特长度，要求该值小于$(2^{32}-1)v$)。

输出：长度为$klen$的密钥数据比特串K。

a)初始化一个32比特构成的计数器ct=0x00000001；

b)对i从1到$\lceil klen/v \rceil$执行：

b.1)计算$Ha_i=H_v(Z \| ct)$；

b.2)ct++；

c)若$klen/v$是整数，令$Ha!_{\lceil klen/v \rceil}=Ha_{\lceil klen/v \rceil}$，否则令$Ha!_{\lceil klen/v \rceil}$为$Ha_{\lceil klen/v \rceil}$最左边的$(klen-(v\times\lfloor klen/v \rfloor))$比特；

d)令$K=Ha_1||Ha_2||\cdots||Ha_{\lceil klen/v \rceil-1}||Ha!_{\lceil klen/v \rceil}$。

SM4

Introduction of SM4

- SM4 为无线局域网标准，是分组加密算法，实现对称加密，用于替代 DES/AES 等国际算法，SM4 算法与 AES 算法具有相同的密钥长度和分组长度，均为 128 位。对长消息进行加解密时，若消息长度过长，需要进行分组，要消息长度不足，则要进行填充。

- 加密算法与密钥扩展算法都采用 32 轮非线性迭代结构，解密算法与加密算法的结构相同，只是轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。

	SM4	DES	AES
计算轮数	32	16（3DES 为 16*3）	10/12/14
密码部件	S 盒、非线性变换、线性变换、合成变换	标准算术和逻辑运算、先替换后置换，不含线性变换	S 盒、行移位变换、列混合变换、圈密钥加变换（AddRoundKey）

SM7

Introduction of SM7

- SM7 为分组加密算法，对称加密，分组长度为128比特，密钥长度为128比特。
  - 保证信息机密性。

- 该算法不公开，应用包括身份识别类应用（非接触式 IC 卡、门禁卡、工作证、参赛证等），票务类应用（大型赛事门票、展会门票等），支付与通卡类应用（积分消费卡、校园一卡通、企业一卡通等）。

SM9

Introduction of SM9

- SM9 为标识加密算法（Identity-Based Cryptography），非对称加密，标识加密将用户的标识（如微信号、邮件地址、手机号码、QQ 号等）作为公钥，省略了交换数字证书和公钥过程，使得安全系统变得易于部署和管理。提供签名校验，密钥交换，密钥封装与加解密功能。由于以上用例，可以用于网络安全密码协议，如SSL/TLS。
  - 保证数据机密性、真实性和完整性。

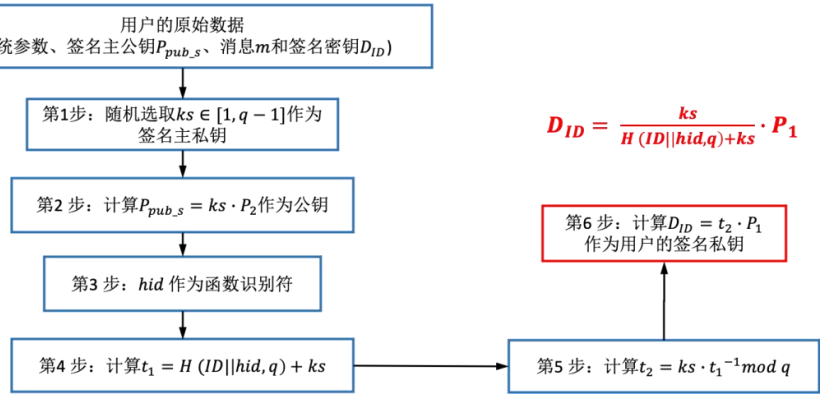
- 在商用密码体系中，SM9 主要用于用户的身份认证，据新华网公开报道，SM9 的加密强度等同于 3072 位密钥的 RSA 加密算法，也就是 128 位安全位数。

- 适用于互联网应用的各种新兴应用的安全保障，如基于云技术的密码服务、电子邮件安全、智能终端保护、物联网安全、云存储安全等等。这些安全应用可采用手机号码或邮件地址作为公钥，实现数据加密、身份认证、通话加密、通道加密等。

SM9

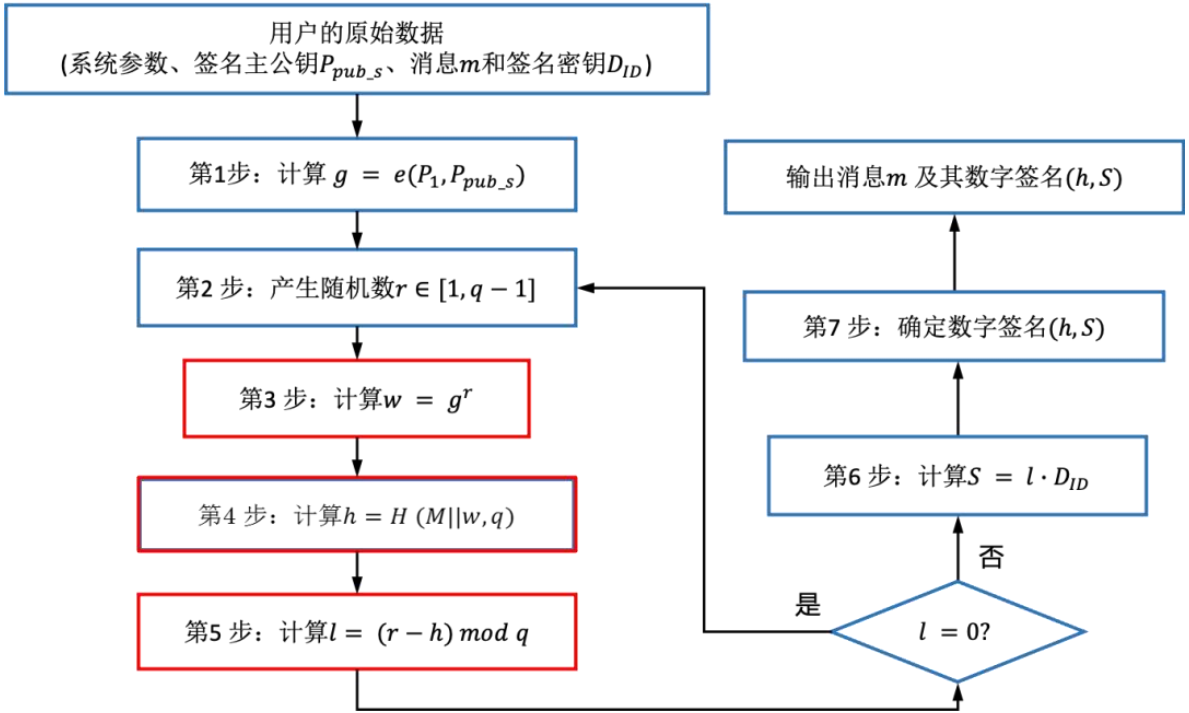
Introduction of SM9

- 引入标识密码的优势：
  - 实现基于身份的密码体制，也就是公钥与用户的身份信息即标识相关，从而比传统意义上的公钥密码体制有许多优点，省去了证书管理等。因此，使用SM9算法不需要申请数字证书，适用于互联网各种新兴应用的安全保障，应用可采用手机号码或邮件地址作为公钥，实现数据加密、身份认证、通话加密、通道加密等安全应用，并具有使用方便，易于部署的特点。
  - 不同于传统签名算法的由用户随机选择私钥然后计算得到公钥的方式，SM9 能够实现用户指定公钥（即身份标识），密钥生成中心通过公钥计算私钥。



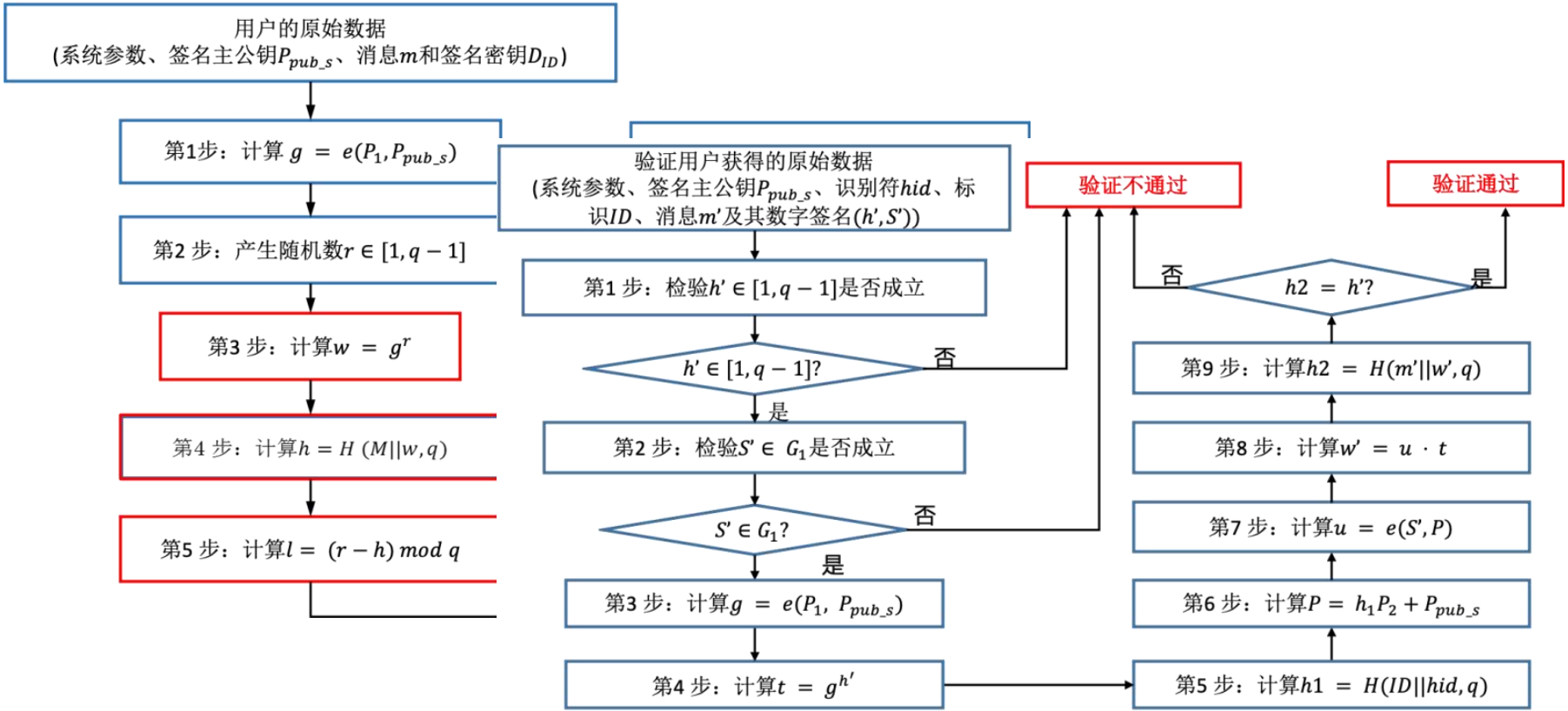
SM9

Introduction of SM9



SM9

Introduction of SM9



ZUC

Introduction of ZUC

- ZUC 为流密码算法，是一种对称加密加密，该机密性算法可适用于 3GPP LTE 通信中的加密和解密，该算法包括祖冲之算法（ZUC）、机密性算法（128-EEA3）和完整性算法（128-EIA3）三个部分。已经被国际组织 3GPP 推荐为 4G 无线通信的第三套国际加密和完整性的标准算法，并称为ISO/IEC 国际标准。
  - 128-EEA3：流加密算法，使用 ZUC
  - 128-EIA3：MAC
  - 保证数据机密性、真实性和完整性。

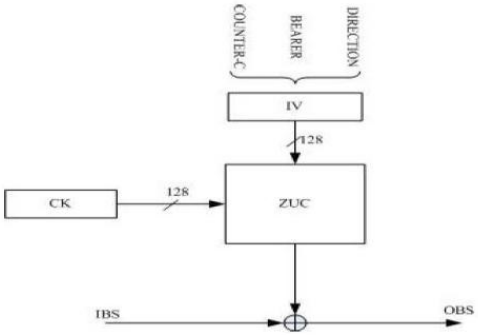


Fig. 1: Principles of the 128-EEA3 encryption operation

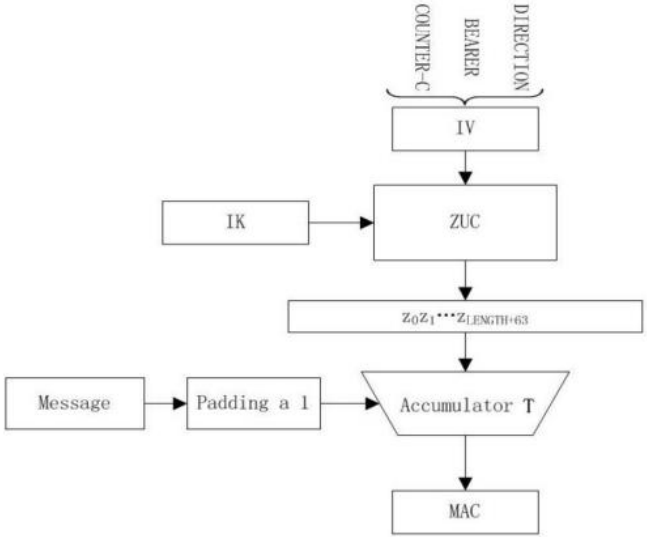


Fig. 4: EIA3 algorithm structure

ZUC

Introduction of ZUC

- ZUC 为流密码算法，是一种对称加密加密，该机密性算法可适用于 3GPP LTE 通信中的加密和解密，该算法包括祖冲之算法（ZUC）、机密性算法（128-EEA3）和完整性算法（128-EIA3）三个部分。已经被国际组织 3GPP 推荐为 4G 无线通信的第三套国际加密和完整性的标准算法，并称为ISO/IEC 国际标准。
  - 128-EEA3：流加密算法，使用 ZUC
  - 128-EIA3：MAC
  - 保证数据机密性、真实性和完整性。

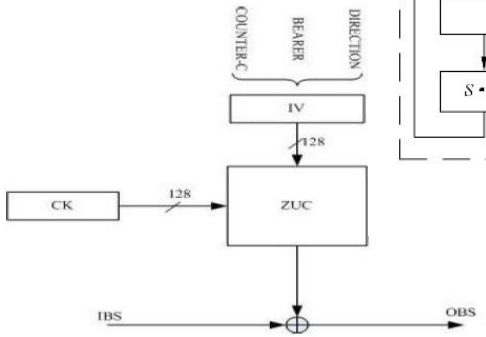


Fig. 1: Principles of the 128-EEA3 encryption operation

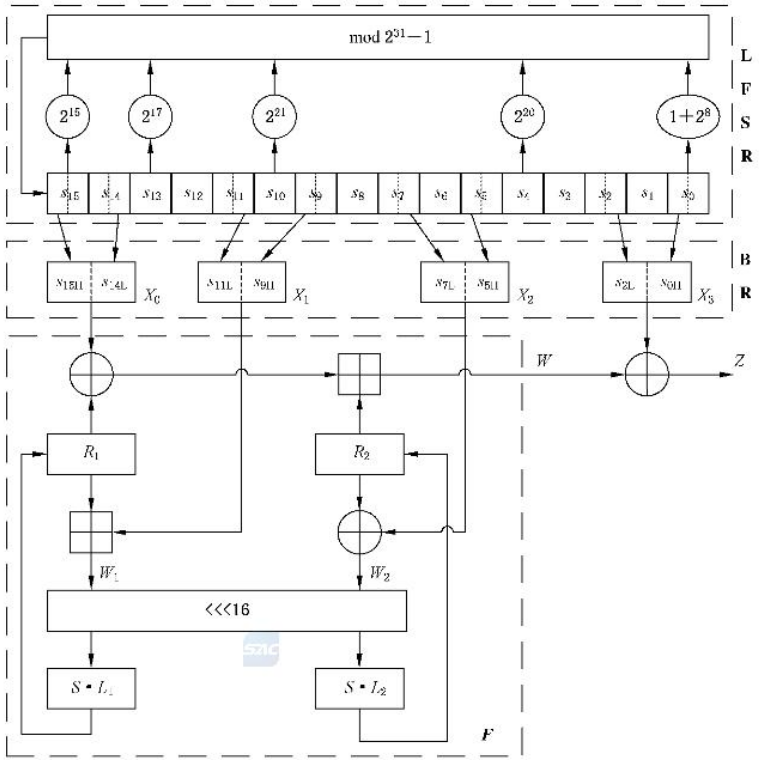


图 1　祖冲之算法结构图

Fig. 4: EIA3 algorithm structure

国密算法在安全协议中的应用

Application of Shangmi Cryptography in Protocols

	GM/T 0024-2014 SSL VPN	GB/T 38636-2020 TLCP	RFC 8998
协议	基于协议 TLS 1.1，但是版本号为 0x0101	基于协议 TLS 1.2，但是版本号为 0x0101	TLS 1.3 国密增强
对称加密算法	SM1_CBC，SM4_CBC	SM4_CBC，SM4_GCM	SM4_GCM，SM4_CCM
签名算法	RSA_SHA1，RSA_SM3，ECC_SM3，IBS_SM3	ECC_SM3，IBC，RSA_SHA256	SM2_SM3
密钥交换	ECDHE，IBSDH，ECC，IBC，RSA	ECDHE，ECC，IBSDH，RSA	SM2ECDHE
密钥派生	PRF（HMAC SHA1，SM3）	PRF（HMAC SHA1，SM3）	HKDF（HMAC SM3）
哈希算法	SHA1，SM3	SHA256，SM3	SM3

国密 TLS

Introduction of Shangmi TLS

- 传输层安全性协议（Transport Layer Security，TLS）是一种密码协议，主要目的是在两个或多个通信计算机应用程序之间提供加密，包括隐私（机密性），完整性和使用证书的真实性。TLS 协议广泛用于电子邮件、即时消息和 IP 语音等应用程序，但它在 HTTPS 方面的使用仍然是最常见的。未通过 TLS 保护的 HTTP 链接通常使用端口 80，HTTPS 则使用端口 443；

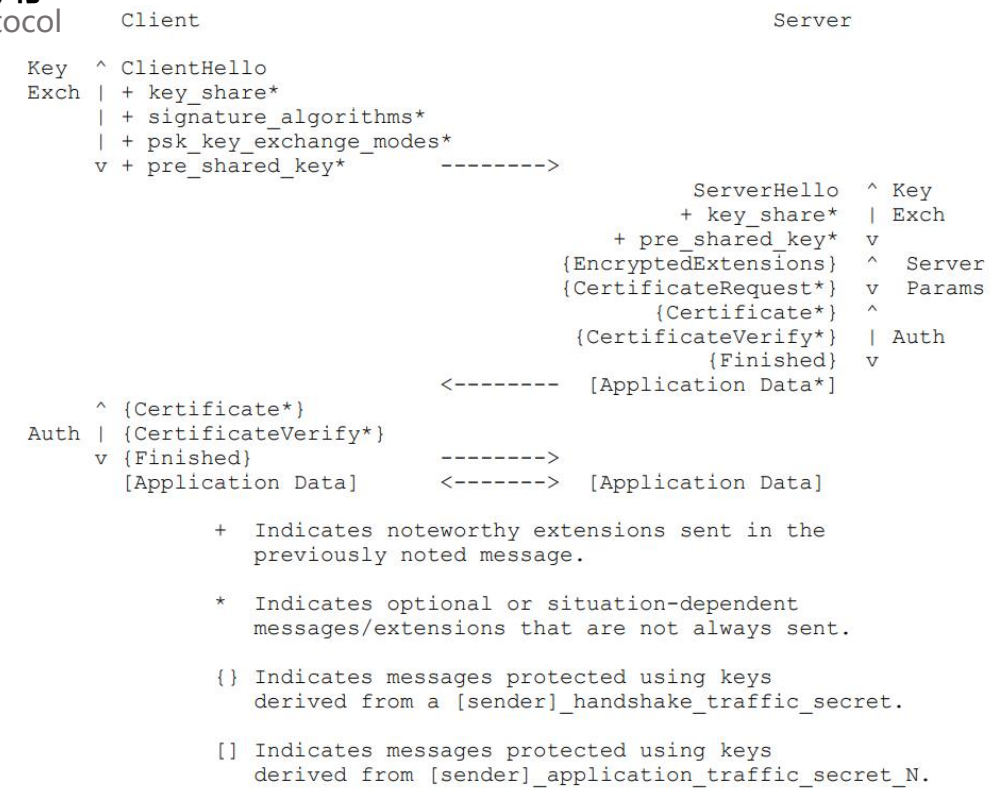
- 国密 TLS 指使用国密套件的 TLS 协议，包含如下规范：
  - GM/T 0024-2014 SSL VPN技术规范：国密 SSL 协议，参考了 TLS 1.0 规范，整个协议握手与加密流程基本与其一致，但和 TLS 1.0 并不兼容；
  - GB/T 38636-2020 信息安全技术 传输层密码协议（TLCP）：TLCP 协议，参考 TLS 1.2 规范，基本兼容 GM/T 0024-2014 且废弃此版本，对于密码算法进行了更新，使用更安全的密码算法；
  - RFC 8998：基于 RFC 8446，扩展通用 TLS 1.3，增加国密套件支持，声明 SM4_GCM_SM3 、SM4_CCM_SM3 以及 SM2 单证书机制在 TLS 1.3 中的使用；

- 当前 360 安全浏览器、奇安信可信浏览器等产品已支持 TLCP 协议，部分银行或金融产品仅允许通过 TLCP 协议进行通信。

国密算法在 TLS 安全协议中的应用

Application of Shangmi Cryptography in TLS Protocol

TLCP / TLS 1.2 握手流程



TLS 1.3 握手流程

Section #2

Rust 实现国密密码与安全协议的优势与现状

Use Rust in Cryptographic Systems

- 密码系统实现的潜在问题
- Rust 的优势
- 国密实现生态

来源：https://developer.okta.com/blog/2022/03/18/programming-security-and-why-

密码系统实现的潜在问题
Potential Problems of Cryptography Systems

- Empirical Study： “You Really Shouldn’ t Roll Your Own Crypto: An Empirical Study of Vulnerabilities in Cryptographic Libraries”，MIT；

- 对于8个大型、通用、开源的 C 与 C++ 密码库进行调研；

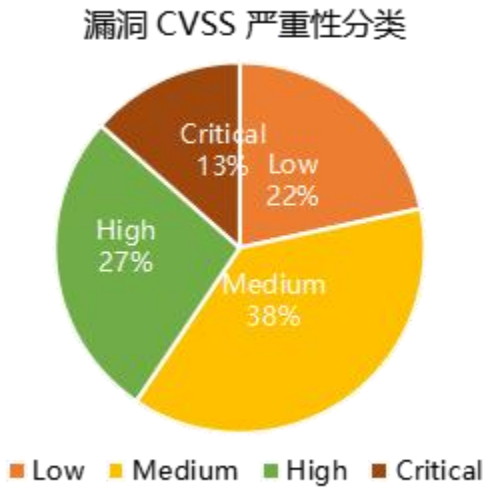
- 现有的问题在于：
  - 密码库导致的一些错误，除了本身包含的一些算法错误以及内存错误，其他的一些问题出在用户的使用错误，即文档、API、等相关说明的缺失；
  - 37.2%的漏洞在于实现时的系统内存错误，其中19.4%是buffer问题，17.7%是资源管理问题；
  - 对于CVSS评分为 7.0 - 10.0 的严重错误中，只有 3.57% - 11.11% 的漏洞是密码学相关，意味着其他的漏洞更多出自系统内存错误以及其他分类；
  - 密码系统问题发现时间长，中位数为4.18年；
  - 大型的C/C++项目很难保证代码安全性；
    - 以 OpenSSL 为例，平均每1000行代码就会引入一个攻击点，具有安全漏洞。

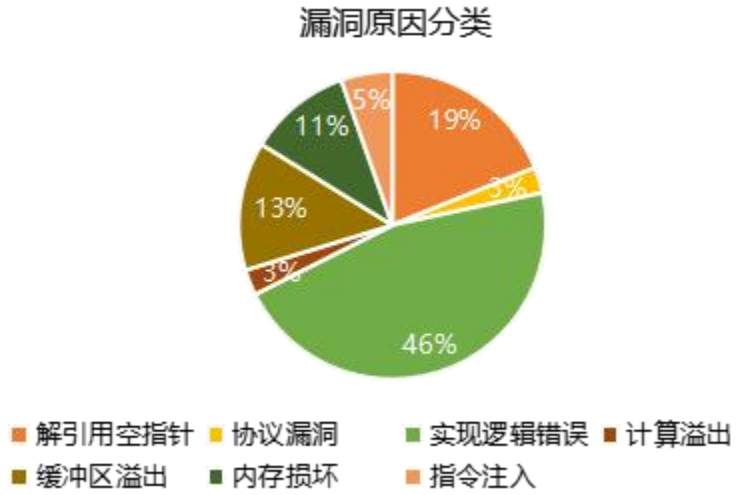
密码系统实现的潜在问题
Potential Problems of Cryptography Systems

- 数据来源于 OpenSSL 页面 Vulnerabilities 以及 CVE Details，截止至2023.6.3，统计 2020 – 2023 近四年的 OpenSSL 相关安全漏洞：

	Low	Medium	High	Critical	Total
解引用空指针	0	6	1	0	7
协议漏洞	1	0	0	0	1
实现逻辑错误	7	5	4	1	17
计算溢出	0	0	1	0	1
缓冲区溢出	0	1	3	1	5
内存损坏	0	2	1	1	4
指令注入	0	0	0	2	2
Total	8	14	10	5	37

- 内存安全问题包括解引用空指针、缓冲区溢出、内存损坏，占总问题的 43.2%，High 及 Critical 问题的 46.7%。

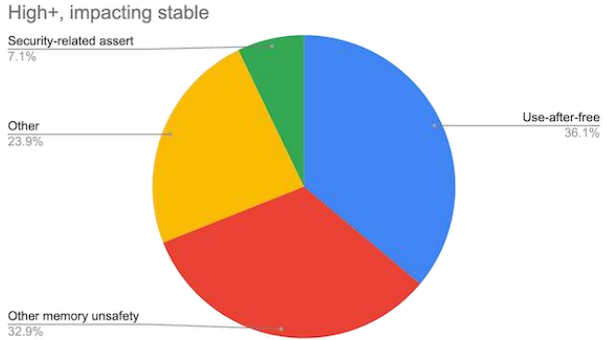




密码系统实现的潜在问题
Potential Problems of Cryptography Systems

- 谷歌等多个公司级项目调研：
  - 内存问题居多
    - 谷歌： Chromium项目中，70%的安全问题是内存安全问题，非安全的bug根错误也与此相同；90%的安卓漏洞是内存安全问题；
    - 苹果：iOS和macOS中60-70%的漏洞是内存安全漏洞；
    - 总体来说，80%被利用的漏洞是内存安全问题相关的漏洞；
  - 密码系统问题发现时间长
    - 谷歌： Chromium项目中，超过50%的安全问题发现时间超过1年，约25%超过3年；

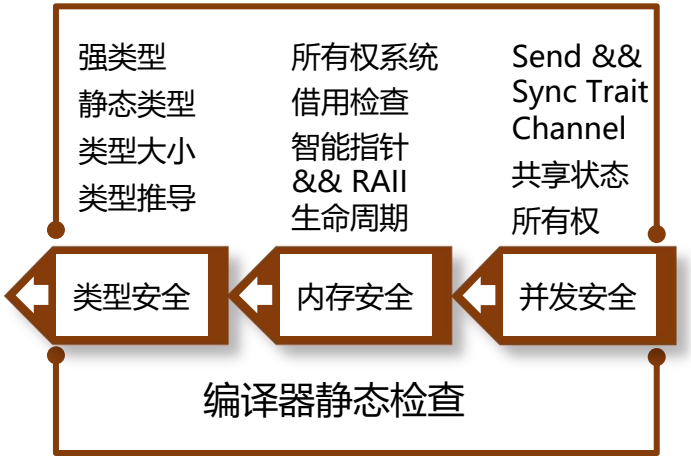
- Rust：
  - 内存安全；
  - 性能与 C 持平。



Memory Safety，Google，Analysis Based on 912 High or Critical Severity Security Bugs since 2015, Affecting the Stable Channel

Rust：解决内存安全问题

Advantages in Programming in Rust



Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92

Rust性能基本和C、C++持平，适用于系统级编程领域

- ✓ 无GC、无Runtime、无解释器
- ✓ 零成本抽象
- ✓ 后端LLVM优化
- ✓ 支持C-ABI的FFI方式
- ✓ 支持自定义内存分配器

- ✓ 强大编译器
- ✓ 全开源方式运作
- ✓ Cargo
- ✓ Crates.io
- ✓ Docs.rs
- ✓ 自带测试框架
- ✓ 支持跨平台
- ✓ 多编程范式
- ✓ 丰富的文档手册

高可靠 1 — **高性能** 2 — **高生产力** 3

国密实现生态

Overview of Shangmi Cryptography Implementation

		RustCrypto	Ring	libsm	OpenSSL	GmSSL	BabaSSL	TASSL	Crypto++	GMSM	Pyca / cryptography	Java Cryptography Archtecture
语言		Rust				C				Go	Python	Java
SM2	加解密			√	√	√	√	√		√		
	ECDSA	√		√	√	√	√	√		√		
	ECDHE-IEEE					√				√		
	ECDHE-GBT			√		√	√	√		√		
SM3		√		√	√	√	√	√	√	√	√	
SM4		√		√	√	√	√	√	√	√	√	
SM9	加解密					√				√		
	签名与验签					√				√		
	密钥交换					√				√		
ZUC						√	√			√		
TLCP						√	√	√		√		
TLS 1.2国密支持						√						
TLS 1.3 国密支持						√	√	√				

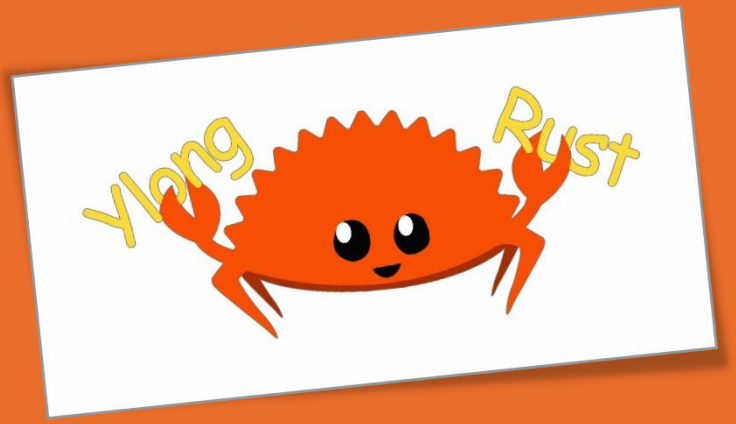
国密实现生态

Overview of Shangmi Cryptography Implementation

		RustCrypto	Ring	libsm	OpenSSL	GmSSL	BabaSSL	TASSL	Crypto++	GMSM	Pyca / cryptography	Java Cryptography Archtecture
语言		Rust				C				Go	Python	Java
SM2	加解密			√	√	√	√	√		√		
	ECDSA	√		√	√	√	√	√		√		
	ECDHE-IEEE					√				√		
	ECDHE-GBT			√		√	√	√		√		
SM9												
ZUC						√	√			√		
TLCP						√	√	√		√		
TLS 1.2国密支持						√						
TLS 1.3 国密支持						√	√	√				

- Rust 社区中对于国密的支持较弱；
- C 社区中 GmSSL 等库提供完整国密能力支持，提供对应优化；Rust 社区中，对于下载量超过 1W 且半年内有更新、在维护期的国密套件库，仅有 RustCrypto 和 libsm，未审核，且缺少安全协议功能支持，在性能上也可以进一步优化。

Section #4



华为 Ylong Rust 密码库

Huawei Ylong Rust Cryptographic Framework

- 使用 Rust 实现国密框架
- 国密算法在安全协议中的应用

使用 Rust 实现国密框架

Use Rust to Implement Shangmi Cryptography and Protocol Framework

- 相较于其他语言：
  - Rust 实现内存安全，并且性能比肩 C 语言，框架具有一定竞争力；
- 相较于 Rust 社区其他库：
  - 社区中国密支持较弱，当前框架未经过审计，提供统一实现可以解决这一问题；
- 期望：实现具有统一管理、标准并且通过审核保证规范性的Rust密码库：
  - 更好地实现社区暂时缺少支持的国密算法以及国密 TLS 协议，补充生态完整性；
  - 完成公司审计，保证规范性和安全性。

使用 Rust 实现国密算法

Use Rust to Implement Shangmi Algorithms

		ylong	Rust Crypto	Ring	libsm	OpenSSL	GmSSL	BabaSSL	TASSL	Crypto++	GMSM	Pyca / cryptography	Java Cryptography Archtecture
语言		Rust						C			Go	Python	Java
SM2	加解密	√			√	√	√	√	√		√		
	ECDSA	√	√		√	√	√	√	√		√		
	ECDHE-IEEE	√					√				√		
	ECDHE-GBT	√			√		√	√	√		√		
SM3		√	√		√	√	√	√	√	√	√	√	
SM4		√	√		√	√	√	√	√	√	√	√	
SM9	加解密	已规划					√				√		
	签名与验签	已规划					√				√		
	密钥交换	已规划					√				√		
ZUC		√					√	√			√		
TLCP		√					√	√	√		√		
TLS 1.2国密支持		√					√						
TLS 1.3 国密支持		√					√	√	√				

- Ylong Rust 计划提供全套国密支持，并且提供统一管理、审核、与优化。

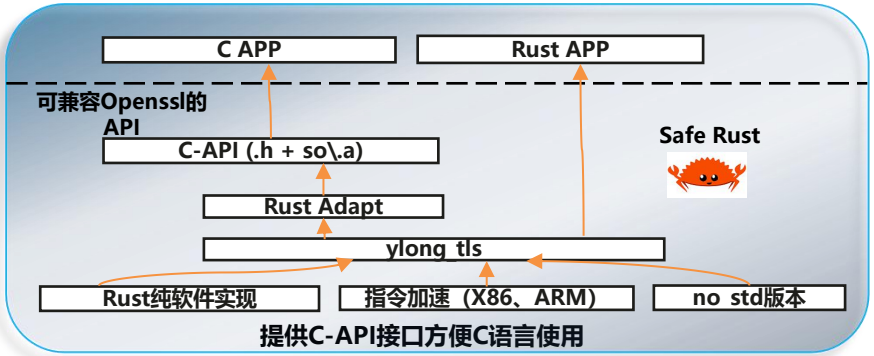
使用 Rust 实现国密 TLS

Use Rust to Implement Shangmi Protocols

TLCP GB/T 38636-2020	TLS 1.2 RFC 5246 GB/T 38636-2020	TLS 1.3 RFC 8446 RFC 8998
• Resumption机制 • SM4GCMSM3 • Sm2国密双证书 • Sm2 GMT ECDHE • Sm2ECC	• 国际通用密码套 • 单证书机制 • Resumption + Ticket机制 • SM4GCMSM3 • Sm2国密单/双证书 • Sm2 IEEE ECDHE • Sm2ECC	• 国际通用密码套 • 单证书机制 • Resumption + Ticket机制 • SM4GCMSM3 • Sm2国密单证书 • Sm2 IEEE ECDHE

1 2 3

ylong_tls 支持版本以及特性



密码算法：

1. **对称加密**：AES 、Chacha20 、SM4
2. **加密模式**：GCM、XTS、CBC、CFB、CTR、OFB
3. **非对称加密**：SM2、RSA
4. **签名算法**：ECDSA(Nist、Brainpool)、SM2、ED25519/448 、RSA
5. **密钥交换**：ECDHE(Nist、Brainpool)、SM2、X25519/448
6. **哈希算法**：SHA1/2/512/3、SM3
7. **MAC**：HMAC、Poly1305
8. **安全随机数**：DRBG

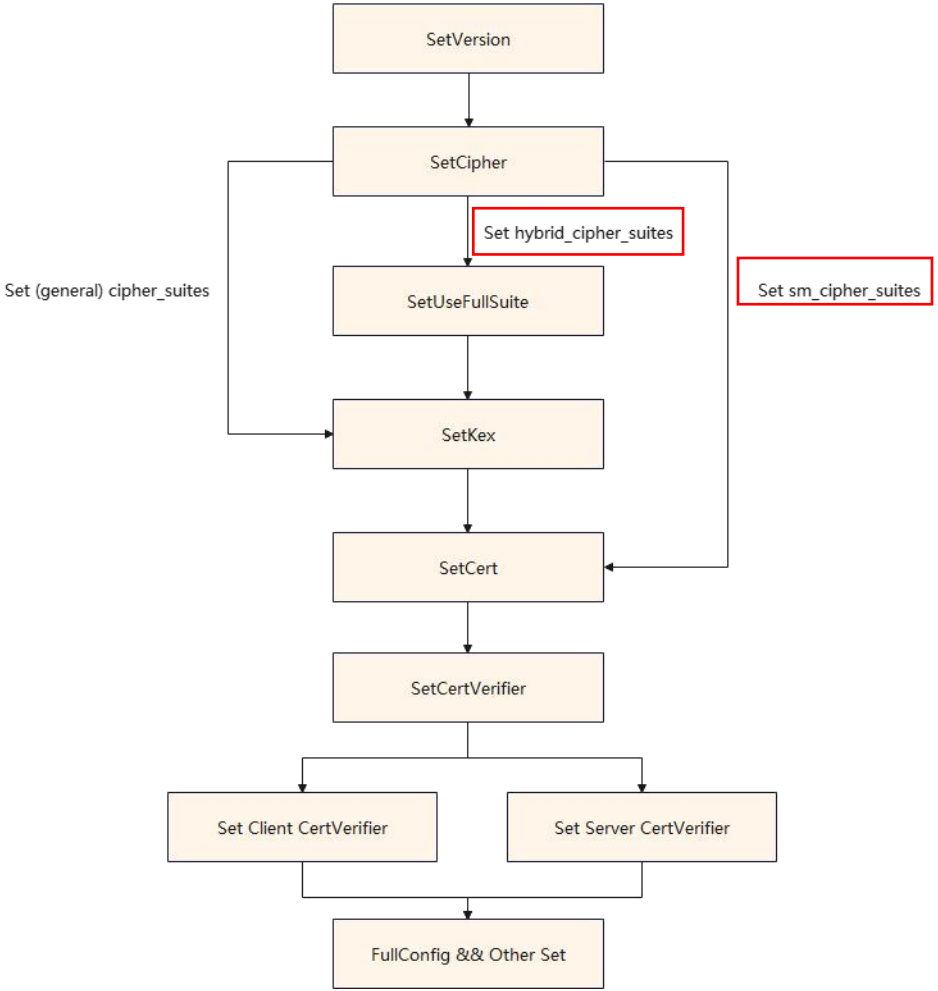
安全协议：

1. **协议版本**： TLCP & TLS1.2 & TLS1.3
2. **签名算法**：ECDSA、EDDSA、RSA
3. **密钥交换算法**：ECC、ECDHE
4. **加密算法**：AES128-GCM、AES256-GCM 、ChaCha20-Poly1305、SM4-GCM
5. **摘要算法**：SHA256、SHA384、SM3
6. **扩展**：ALPN、SNI
7. **Resumption**：SessionID（TLS1.2/TLCP）、Ticket（TLS1.2/TLS1.3）
8. **同步 API 以及基于可替换 Runtime 的异步 API**
9. **支持单国密模式，单通用模式，或混合模式 TLS 使用**

使用 Rust 实现国密 TLS

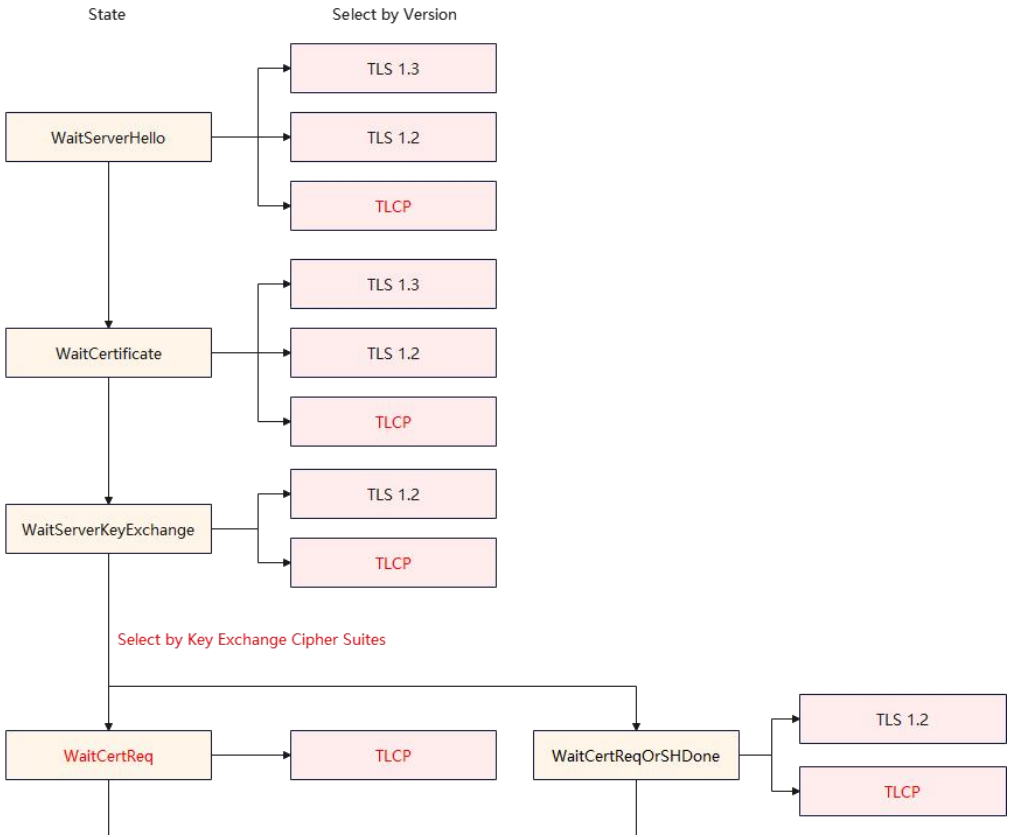
Use Rust to Implement Shangmi Protocols

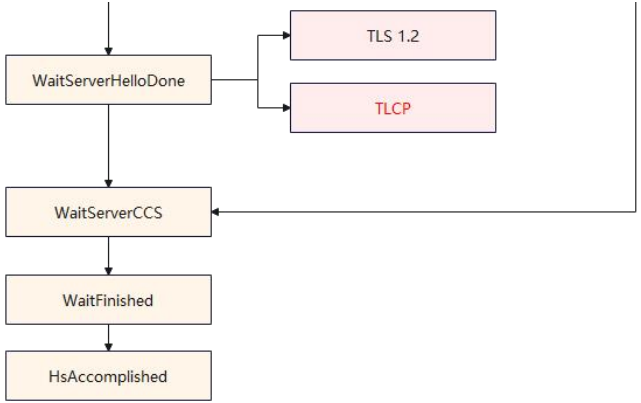
- 提供通用 / 国密 / 混合 TLS 实现；
- TLS 配置结构图，从 SetCipher 开始，增加国密/通用/混合设置接口，并校验已有配置，预先告警，避免错误。



使用 Rust 实现国密 TLS

Use Rust to Implement Shangmi Protocols

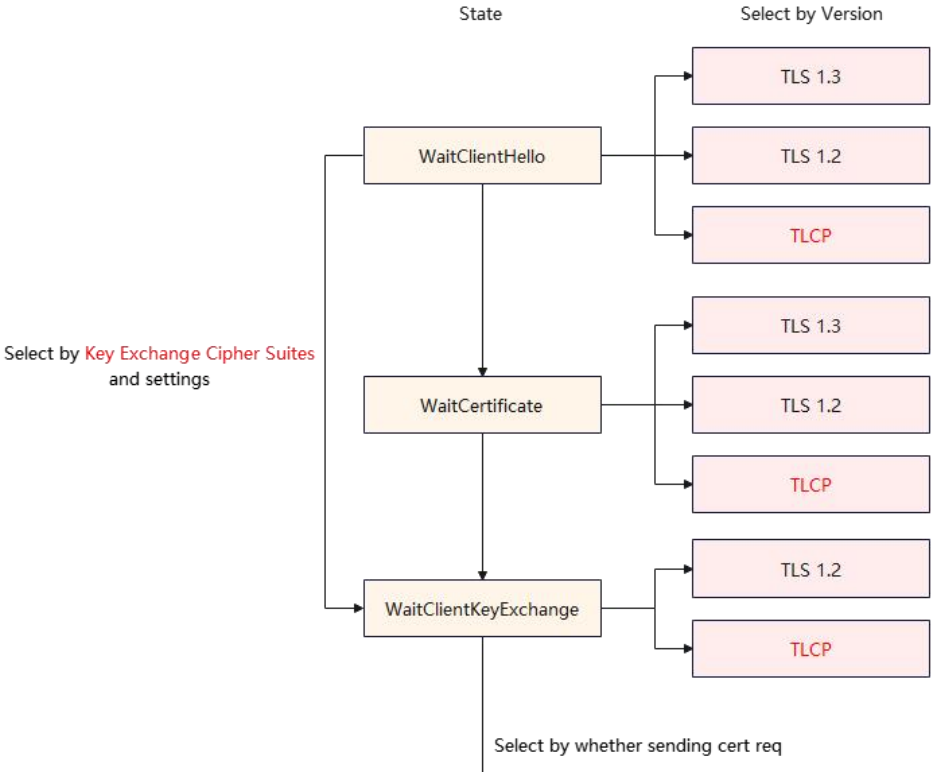


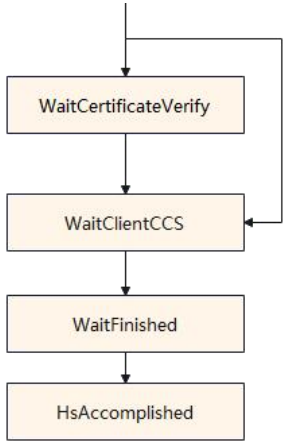


客户端握手状态机与各 TLS 版本差异消息

使用 Rust 实现国密 TLS

Use Rust to Implement Shangmi Protocols



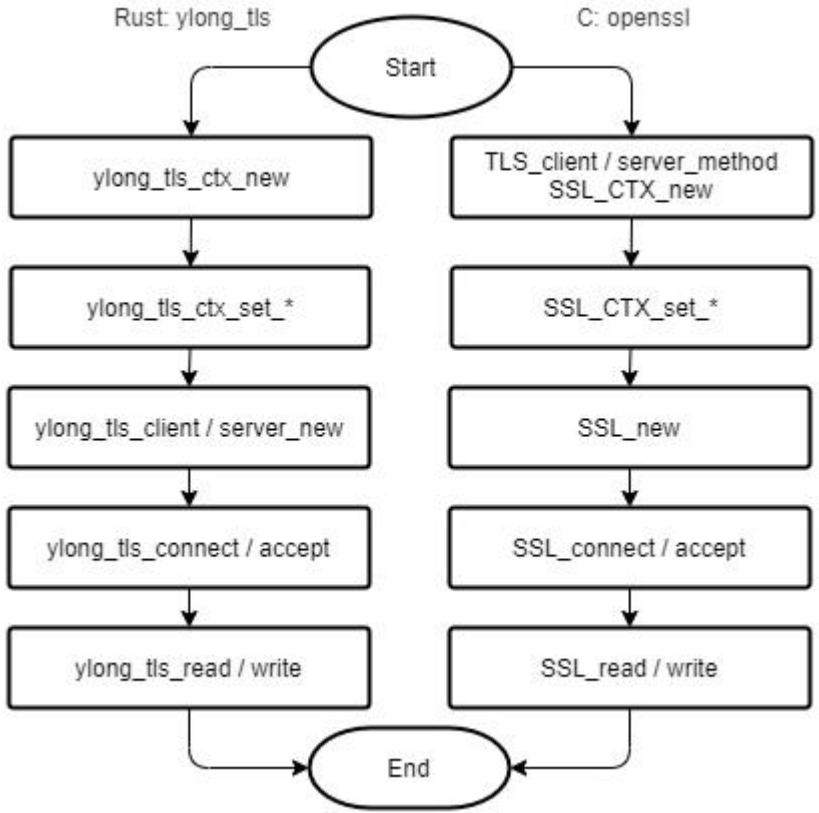


服务端握手状态机与各 TLS 版本差异消息

使用 Rust 实现国密 TLS

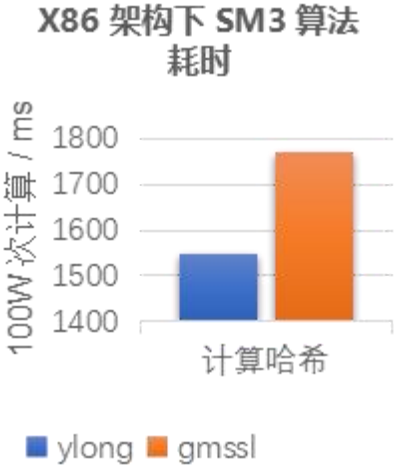
Use Rust to Implement Shangmi Protocols

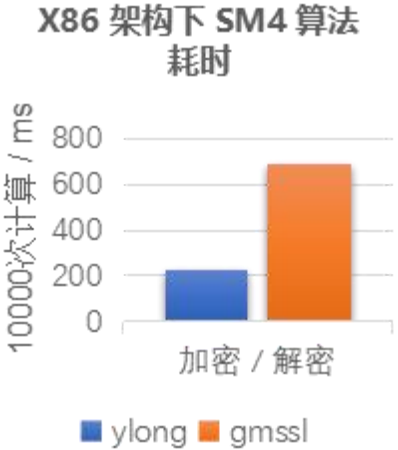
- 接口与 OpenSSL 对比：兼容 OpenSSL，便与 C 与其他语言组件切换迁移使用

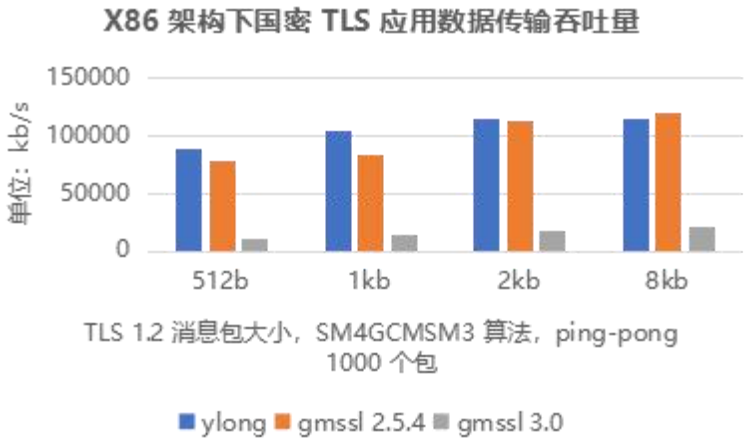


Ylong Rust 国密算法性能对比

Performance Comparison of Ylong Rust Shangmi Cryptography Framework



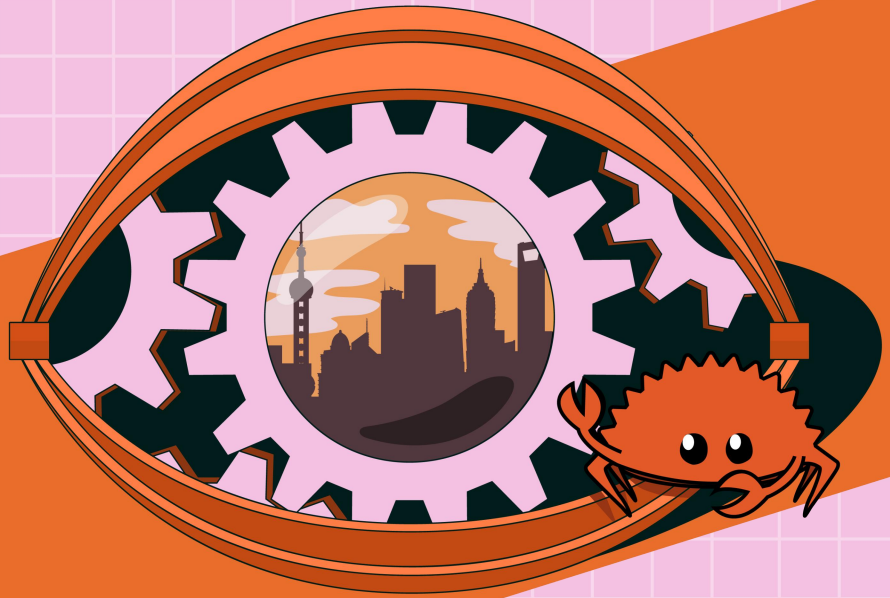




- Ylong Rust 算法实现在 x86架构下国密 SM3 / 4 性能优于 GmSSL；
- TLS 由于与 GmSSL 支持的部分算法不同，仅对比了 TLS 1.2 应用数据传输性能，在 x86 架构下优于或持平 GmSSL ；
- 未来仍会持续优化，或计划开源。

测试环境：
操作系统：Ubuntu 18.04 x86_64
CPU：Intel(R) Xeon(R) Gold 6278C CPU @ 2.60GHz
内存：16G

Thank you!



参考
References

- Transport Layer Security，https://en.wikipedia.org/wiki/Transport_Layer_Security
- «You Really Shouldn’ t Roll Your Own Crypto: An Empirical Study of Vulnerabilities in Cryptographic Libraries»，Blessing Jenny，Specter Michael A.，Weitzner Danieal J.，MIT，arXiv:2107.04940v1 [cs.CR]，https://arxiv.org/abs/2107.04940，2021年6月11日
- GM/T 0024-2014 SSL VPN技术规范：https://github.com/guanzhi/GM-Standards/blob/master/GMT%E6%AD%A3%E5%BC%8F%E6%A0%87%E5%87%86/GMT%200024-2014%20SSL%20VPN%20%E6%8A%80%E6%9C%AF%E8%A7%84%E8%8C%83.PDF
- GB/T 38636-2020 信息安全技术 传输层密码协议（TLCP）：https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=778097598DA2761E94A5FF3F77BD66DA
- RFC 8998：https://www.ietf.org/rfc/rfc8998.txt