

一种多阶段密钥交换协议的设计与实现

答辩人：史豪

指导老师：王立斌

华南师范大学

2025 年 4 月 28 日



目录

研究目标

主要工作内容

OW-ChCCA KEM 实现

TIMKE 协议实现

性能评估与分析

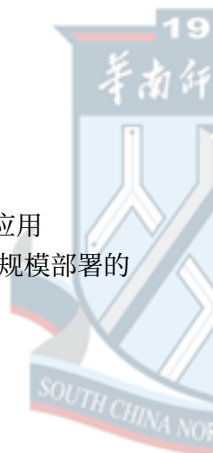
结论与展望

致谢

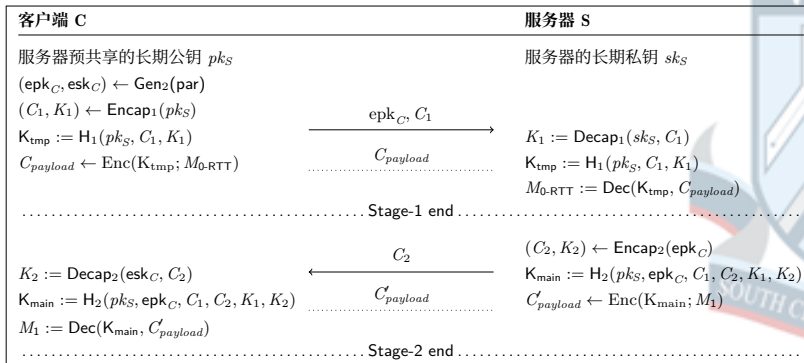


研究目标

- 量子计算威胁传统密码体系
- 多阶段密钥交换 (MSKE) 协议在现代网络通信的广泛应用
- TIMKE 协议提供紧致安全和后量子安全，有望解决大规模部署的安全问题



TIMKE 协议图



主要工作内容

- 实现一种基于格的 OW-ChCCA KEM
- 系统化实现紧致安全多阶段密钥交换协议 (TIMKE)
- 提供完整的演示系统与性能测试框架



开发语言——Golang

- 提供完善的密码学支持
- 静态类型系统和内存安全
- 跨平台兼容性优秀
- 编译速度快
- 拥有强大的并发处理能力



OW-ChCCA KEM 实现

- 基于格的 OW-ChCCA KEM 的数据结构设计、算法实现
- 性能优化（并行矩阵运算、优化内存管理等）



基于格的 OW-ChCCA KEM

Alg Setup(1^λ)

01 **return** $\text{par} := \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$

Alg Gen(par)

02 $b \xleftarrow{\$} \{0, 1\}$, $\mathbf{Z}_b \leftarrow D_{\mathbb{Z}, \alpha}^{m \times \lambda}$
 03 $\mathbf{U}_b := \mathbf{A} \mathbf{Z}_b$, $\mathbf{U}_{1-b} \xleftarrow{\$} \mathbb{Z}_q^{n \times \lambda}$
 04 $\text{pk} := (\mathbf{U}_0, \mathbf{U}_1)$, $\text{sk} := (\mathbf{Z}_b, b)$
 05 **return** (pk, sk)

Alg Encap(pk)

06 $R \xleftarrow{\$} \{0, 1\}^\lambda$, $(\mathbf{s}, \rho, \mathbf{h}_0, \mathbf{h}_1) := \mathbf{G}(R)$
 07 $\mathbf{e} := \text{SampleD}(m, \alpha'; \rho)$
 08 $\mathbf{x} := \mathbf{A}^t \mathbf{s} + \mathbf{e}$
 09 $\hat{\mathbf{h}}_0 := \mathbf{U}_0^t \mathbf{s} + \mathbf{h}_0 \lfloor q/2 \rfloor \in \mathbb{Z}_q^\lambda$
 10 $\hat{\mathbf{h}}_1 := \mathbf{U}_1^t \mathbf{s} + \mathbf{h}_1 \lfloor q/2 \rfloor \in \mathbb{Z}_q^\lambda$
 11 $\hat{K}_0 := \mathbf{H}(\mathbf{x}, \hat{\mathbf{h}}_0, \mathbf{h}_0)$, $C_0 := \hat{K}_0 \oplus R$
 12 $\hat{K}_1 := \mathbf{H}(\mathbf{x}, \hat{\mathbf{h}}_1, \mathbf{h}_1)$, $C_1 := \hat{K}_1 \oplus R$
 13 $\text{ct} := (C_0, C_1, \mathbf{x}, \hat{\mathbf{h}}_0, \hat{\mathbf{h}}_1)$
 14 **return** $(\text{ct}, K := R)$

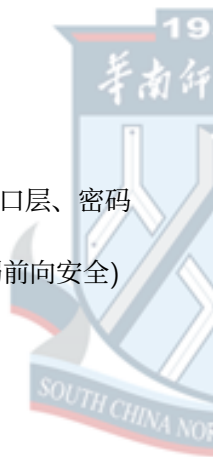
Alg Decap(sk, ct)

15 **let** $\text{ct} = (C_0, C_1, \mathbf{x}, \hat{\mathbf{h}}_0, \hat{\mathbf{h}}_1)$
 16 **let** $\text{sk} = (\mathbf{Z}_b, b)$
 17 $\mathbf{h}'_b := \text{Round}(\hat{\mathbf{h}}_b - \mathbf{Z}_b^t \mathbf{x}) \in \{0, 1\}^\lambda$
 18 $\hat{K}_b := \mathbf{H}(\mathbf{x}, \hat{\mathbf{h}}_b, \mathbf{h}'_b)$
 19 $R := C_b \oplus \hat{K}_b$
 20 $(\mathbf{s}, \rho, \mathbf{h}_0, \mathbf{h}_1) := \mathbf{G}(R)$
 21 $\mathbf{e} := \text{SampleD}(m, \alpha'; \rho)$
 22 $\hat{\mathbf{h}}'_{1-b} := \mathbf{U}_{1-b}^t \mathbf{s} + \mathbf{h}_{1-b} \lfloor q/2 \rfloor$
 23 $\hat{K}_{1-b} := \mathbf{H}(\mathbf{x}, \hat{\mathbf{h}}'_{1-b}, \mathbf{h}_{1-b})$
 24 **if** $\mathbf{x} \neq \mathbf{A}^t \mathbf{s} + \mathbf{e}$ **return** \perp
 25 **if** $\hat{K}_{1-b} \oplus R \neq C_{1-b}$ **return** \perp
 26 **if** $\mathbf{h}'_b \neq \mathbf{h}_b$ **return** \perp
 27 **if** $\hat{\mathbf{h}}'_{1-b} \neq \hat{\mathbf{h}}_{1-b}$ **return** \perp
 28 **return** $K := R$

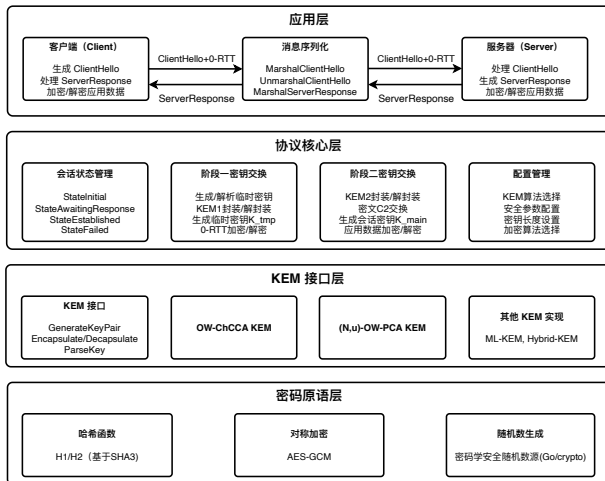
Figure 4: The key encapsulation mechanism $\text{KEM}_{\text{LWE}} = (\text{Setup}, \text{Gen}, \text{Encap}, \text{Decap})$, where $\mathbf{H}: \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ and $\mathbf{G}: \{0, 1\}^* \rightarrow [-\eta, \eta]^n \times \{0, 1\}^\lambda \times \{0, 1\}^\lambda \times \{0, 1\}^\lambda$ are random oracles. To save space, one could set $\mathbf{A} := \mathbf{H}^*(0)$ for a random oracle $\mathbf{H}^*: \{0, 1\}^* \rightarrow \mathbb{Z}_q^{n \times m}$.

TIMKE 协议实现

- 协议架构：应用层、序列化层、协议核心层、KEM 接口层、密码原语层
- 协议流程：预共享、第一阶段 (0-RTT)、第二阶段 (弱前向安全)



TIMKE 协议实现 - 架构图 P.22



TIMKE 协议实现 - 演示系统



图: TIMKE 协议演示系统界面

TIMKE 协议实现 - 演示系统

```

    TIMKE

    Tightly secure Multi-stage Key Exchange - Server

    Available KEM algorithms:
    - OWChCCA-64
    - ML-KEM-512
    - ML-KEM-768
    - ML-KEM-1024
    - OWChCCA-16
    - OWChCCA-32

    Generating new ML-KEM-768 key pair...
    Key pair generated and saved to /Users/shihao/Downloads/book/MingLuo-TIMKE/code/TIMKE/.temp/server-key.pem.pub
    Server public key (for client use): bb7c3d436b9887d54e9db4bad1169c94ab27b446261e64818fc791253bcad6
    Long Term Key algorithm: ML-KEM-768

    Server keys generated:
    Private key: /Users/shihao/Downloads/book/MingLuo-TIMKE/code/TIMKE/.temp/server-key.pem
    Public key: /Users/shihao/Downloads/book/MingLuo-TIMKE/code/TIMKE/.temp/server-key.pem.pub

    Press Enter to continue...
    
```

图: TIMKE 服务端长期密钥生成

TIMKE 协议实现 - 演示系统

```
Project: main.go demo.bash x
Terminal: demo.bash (2) x +
----- Protocol Stage 1 -----
KEM1 Type: ML-KEM-768
KEM2 Type: ML-KEM-768
Ephemeral public key length: 1184 bytes
Ciphertext1 length: 1088 bytes
0-RTT data: Hello from TIMKE client! This is 0-RTT data.
Encrypted payload length: 72 bytes
len EphPubKey: 1184
len Ciphertext1: 1088
len EncryptedPayload: 72
len KEM1Type: 10
len KEM2Type: 10
Sent ClientHello (2384 bytes)
Sent 0-RTT data: "Hello from TIMKE client! This is 0-RTT data."
Waiting for server response...
Received server response (1188 bytes)
----- Protocol Stage 2 -----
Ciphertext2 length: 1088 bytes
Encrypted payload length: 84 bytes
Session established! Protocol completed in 1.153208ms
Session key (first 8 bytes): 86f8acc489927a19
Server data: Hello from TIMKE server! This is stage-2 protected data.
TIMKE protocol completed successfully!
Press Enter to continue...
```

图: TIMKE 协议演示系统界面

性能评估与分析

- 评估 OW-ChCCA KEM 理论参数的实用性挑战
- OW-ChCCA KEM 与 ML-KEM 性能对比 (计算时间、内存占用、密钥大小)
- 不同 KEM 组合的 TIMKE 协议性能评估

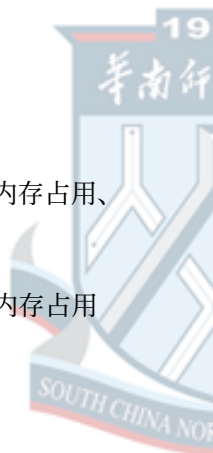


| KEM 组合 | 阶段 1 | 阶段 2 | 总时间 | 内存 (KB) |
|----------------------------|--------|--------|--------|-----------|
| OWChCCA-mini + ML-KEM-512 | 322.18 | 118.53 | 440.71 | 2,206,372 |
| OWChCCA-mini + ML-KEM-768 | 324.47 | 117.61 | 442.07 | 2,206,250 |
| OWChCCA-mini + ML-KEM-1024 | 374.76 | 127.47 | 502.23 | 2,204,554 |

表: 混合 KEM 实现的 TIMKE 协议性能 (单位: 毫秒)

性能评估与分析

- 评估 OW-ChCCA KEM 理论参数的实用性挑战
- OW-ChCCA KEM 与 ML-KEM 性能对比 (计算时间、内存占用、密钥大小)
- 不同 KEM 组合的 TIMKE 协议性能评估
- ML-KEM 替代方案的优异性能：毫秒级执行时间、低内存占用
- 结论：理论构造与实用性能的权衡



结论与展望

- 主要成果：
 - 实现基于格的 OW-ChCCA KEM
 - 成功实现紧致安全后量子协议并验证其可行性
- 未来工作方向：
 - 结构化格上的 OW-ChCCA KEM 优化
 - 协议功能扩展
 - 实际应用集成



额外工作

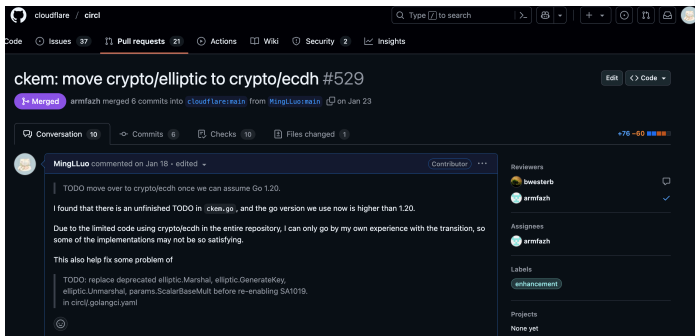


图: Cloudflare CIRCL 库的开源贡献

致谢

感谢评审老师们的耐心倾听！

