Thursday, April 3 2025

Security Analysis of WeChat's MMTLS Encryption Protocol

Pellaeon Lin, Mona Wang

THECITIZENLAB
munk school OF GLOBAL AFFAIRS & PUBLIC POLICY
UNIVERSITY OF TORONTO

Agenda

Security Analysis of WeChat's MMTLS Encryption Protocol

- Introduction, motivation, methodologies
- WeChat network request lifecycle
- MMTLS encryption, Business-layer encryption
- Discussion, recommendations, future work

Pellaeon Lin

- Researcher at Citizen Lab, University of Toronto
- Security and privacy of mobile apps
- Past studies
  - TikTok vs Douyin - A Security and Privacy Analysis
  - Unmasked II: An Analysis of Indonesia and the Philippines' Government-launched COVID-19 Apps
  - Unmasked: COVID-KAYA and the Exposure of Healthcare Worker Data in the Philippines

THECITIZENLAB

Mona Wang

- Networking security researcher, PhD student at Princeton CITP
- OTF Information Controls Research Fellow at Citizen Lab
- Previously technologist at EFF
- Other work
  - Network measurement (CoNEXT 22)
  - Traffic fingerprinting resistance and censorship circumvention (PETS 22)
  - Threat modelling and security training for organizers (CSCW 22)

https://m0na.net

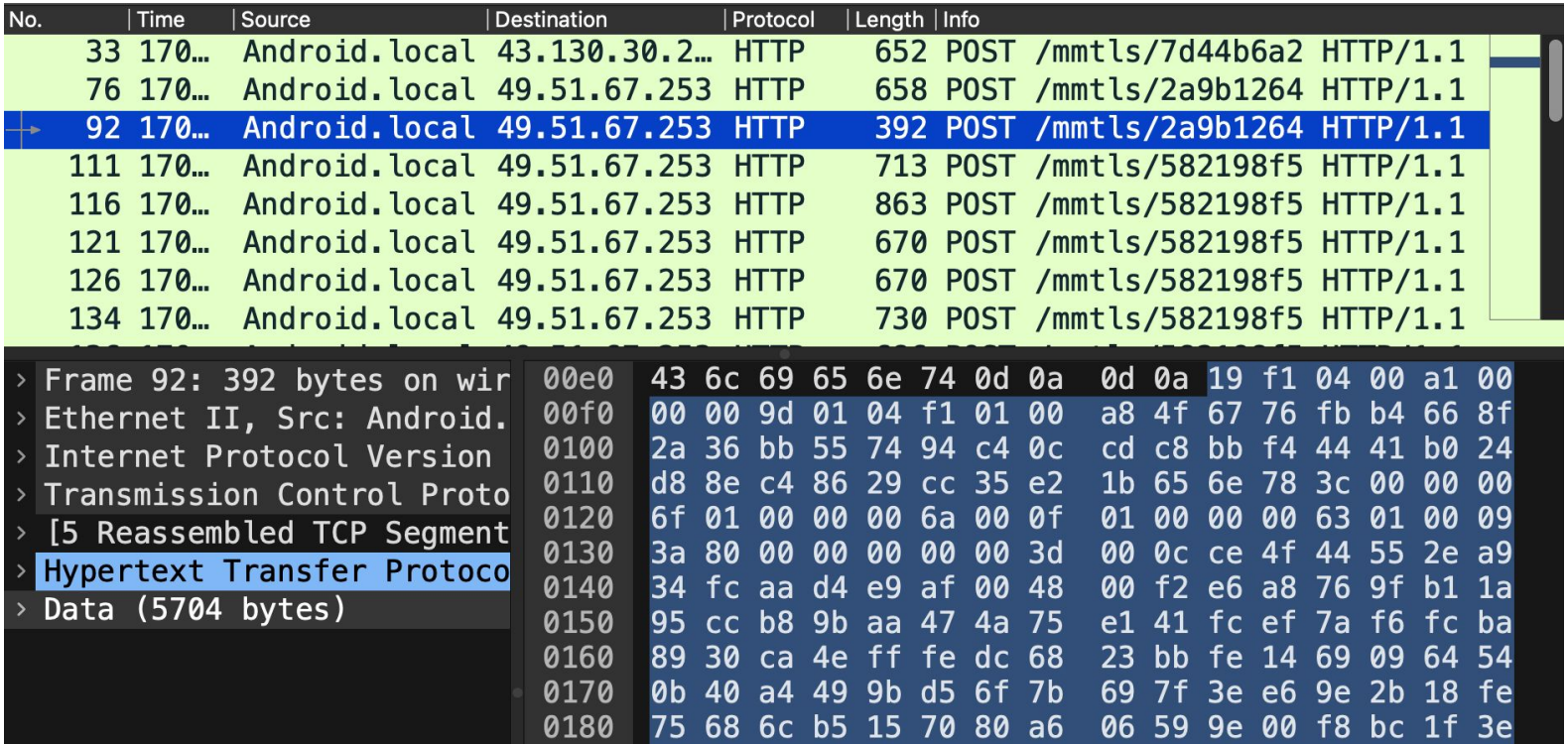
THECITIZENLAB

munk school
OF GLOBAL AFFAIRS & PUBLIC POLICY

UNIVERSITY OF
TORONTO

Motivation

What's being sent? **Is the encryption sound?** **Why custom encryption?**



No.	Time	Source	Destination	Protocol	Length	Info
33	170…	Android.local	43.130.30.2…	HTTP	652	POST /mmtls/7d44b6a2 HTTP/1.1
76	170…	Android.local	49.51.67.253	HTTP	658	POST /mmtls/2a9b1264 HTTP/1.1
92	170…	Android.local	49.51.67.253	HTTP	392	POST /mmtls/2a9b1264 HTTP/1.1
111	170…	Android.local	49.51.67.253	HTTP	713	POST /mmtls/582198f5 HTTP/1.1
116	170…	Android.local	49.51.67.253	HTTP	863	POST /mmtls/582198f5 HTTP/1.1
121	170…	Android.local	49.51.67.253	HTTP	670	POST /mmtls/582198f5 HTTP/1.1
126	170…	Android.local	49.51.67.253	HTTP	670	POST /mmtls/582198f5 HTTP/1.1
134	170…	Android.local	49.51.67.253	HTTP	730	POST /mmtls/582198f5 HTTP/1.1

THECITIZENLAB

Motivation

SSL/TLS

- Secures billions of users traffic
- 30+ years of development
- Open standard, lots of academic and public scrutiny

WeChat MMTLS

- Secures 1+ billion users traffic
- Deployed for ~8 years
- One public blog post

MMTLS deserves just as much scrutiny as TLS!!!

WeChat network request lifecycle

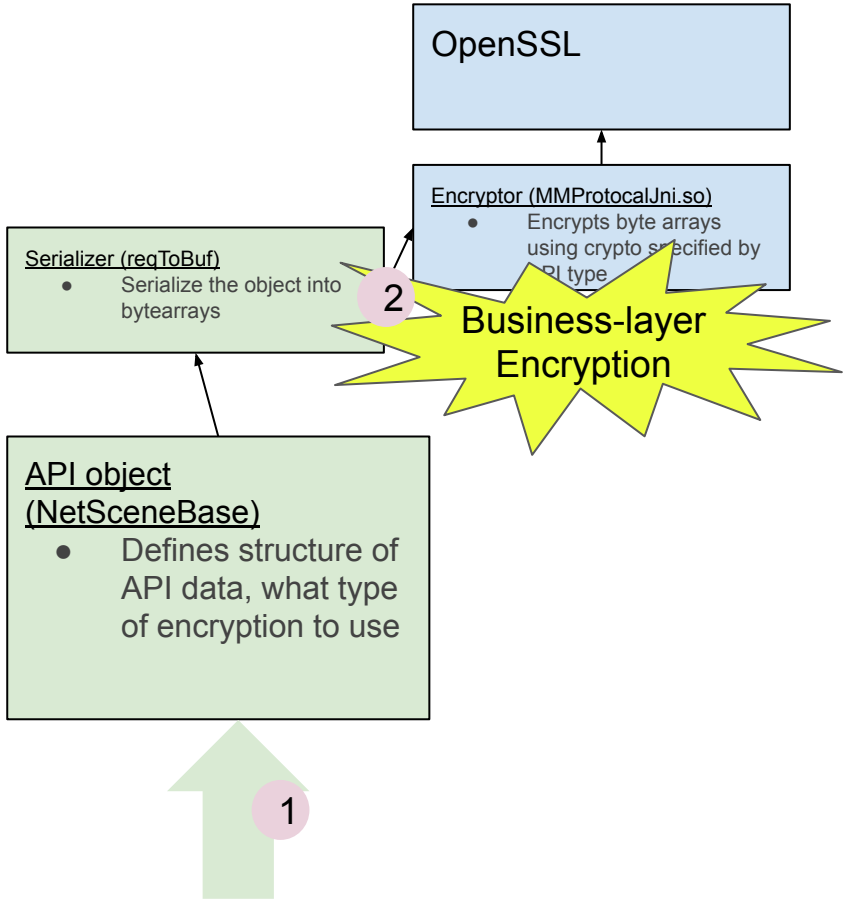
Anatomy of a Wechat network request

```
/* renamed from: com.tencent.mm.model.bm */
/* loaded from: classes3.dex */
23 public final class MMReqRespReg2 extends ReqRespBase {

    /* renamed from: pcn */
    private final MMReg2.Req reqobj = new MMReg2.Req();

    /* renamed from: pco */
    private final MMReg2.Resp respobj = new MMReg2.Resp();

    @Override // com.tencent.p486mm.network.MMTLSConnection
    public final int getType() {
        return 126;
    }

    @Override // com.tencent.p486mm.network.MMTLSConnection
    public final String getUri() {
        return "/cgi-bin/micromsg-bin/newreg";
    }
```

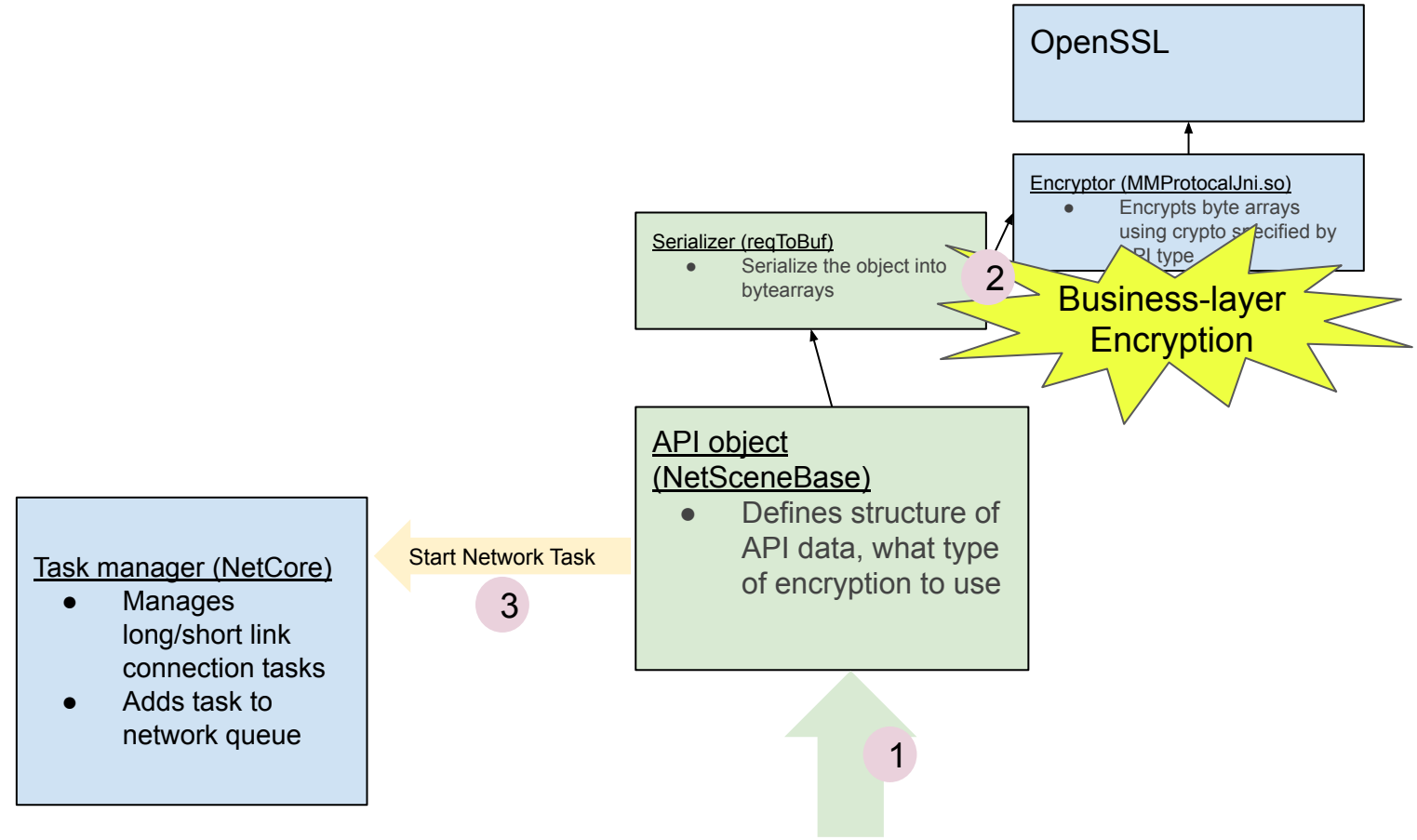
- API endpoint is referred to as “Scene”, has unique “type” number and URI

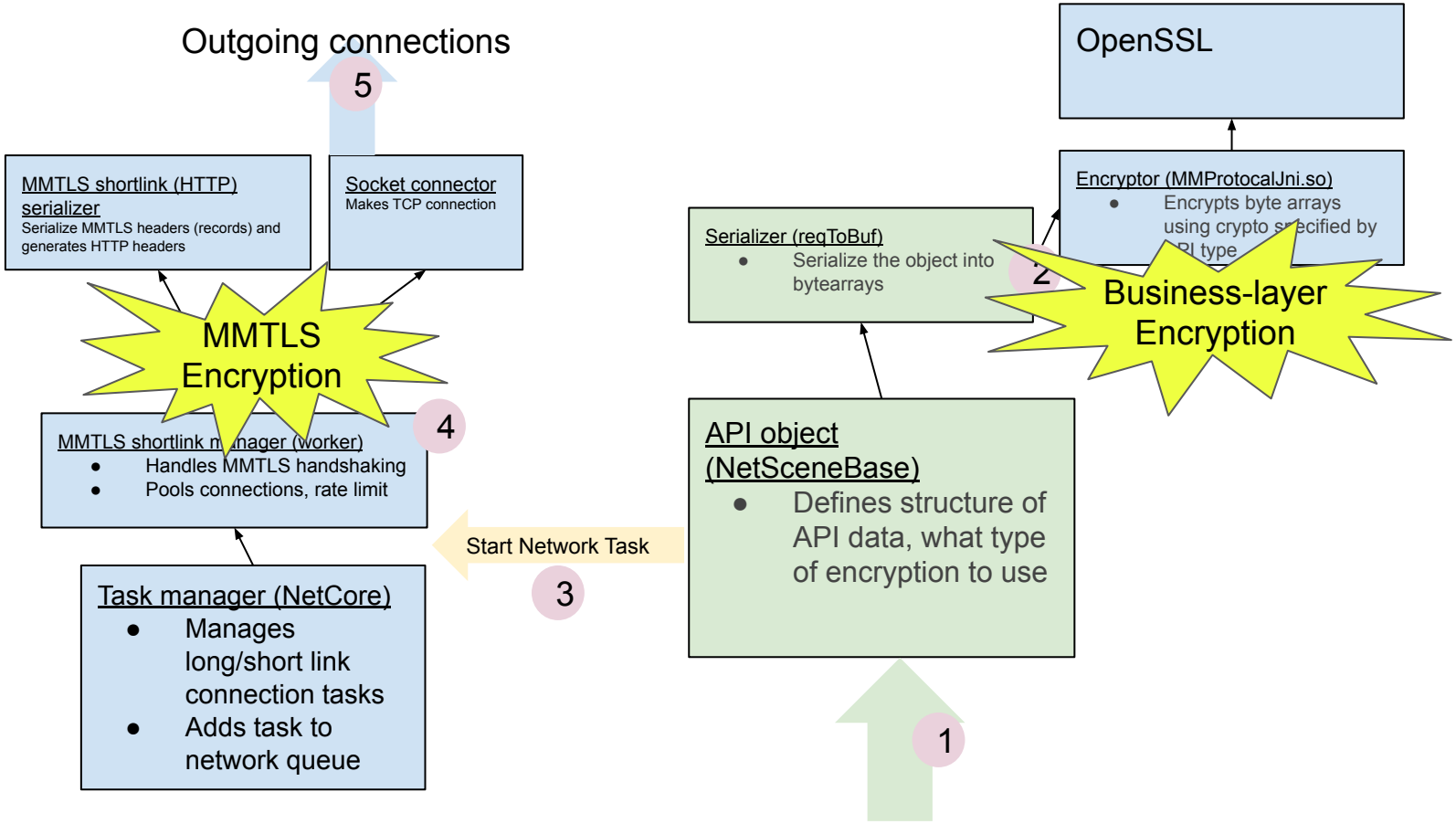
```java
/* renamed from: com.tencent.mm.protocal.protobuf.dvx */
/* loaded from: classes10.dex */
public final class MMReg2ReqPB extends BaseReqPBBuilder {

    /* renamed from: Qgf */
    public String mobiledetail3;
    public String UserName;

    /* renamed from: abHA */
    public String email;

    /* renamed from: abHB */
    public String mobile;

    /* renamed from: abHz */
    public int bindqq;

    /* renamed from: abJe */
    public String appid;
    public int abPi;
    public String abPj;
    public GenericBufferPB abPm;
    public GenericBufferPB abPn;
    public String abPo;

    /* renamed from: abPp */
    public SignKeyPB signkey;

    /* renamed from: abPv */
    public PubECDHKeyPbContainer pubkey;
```

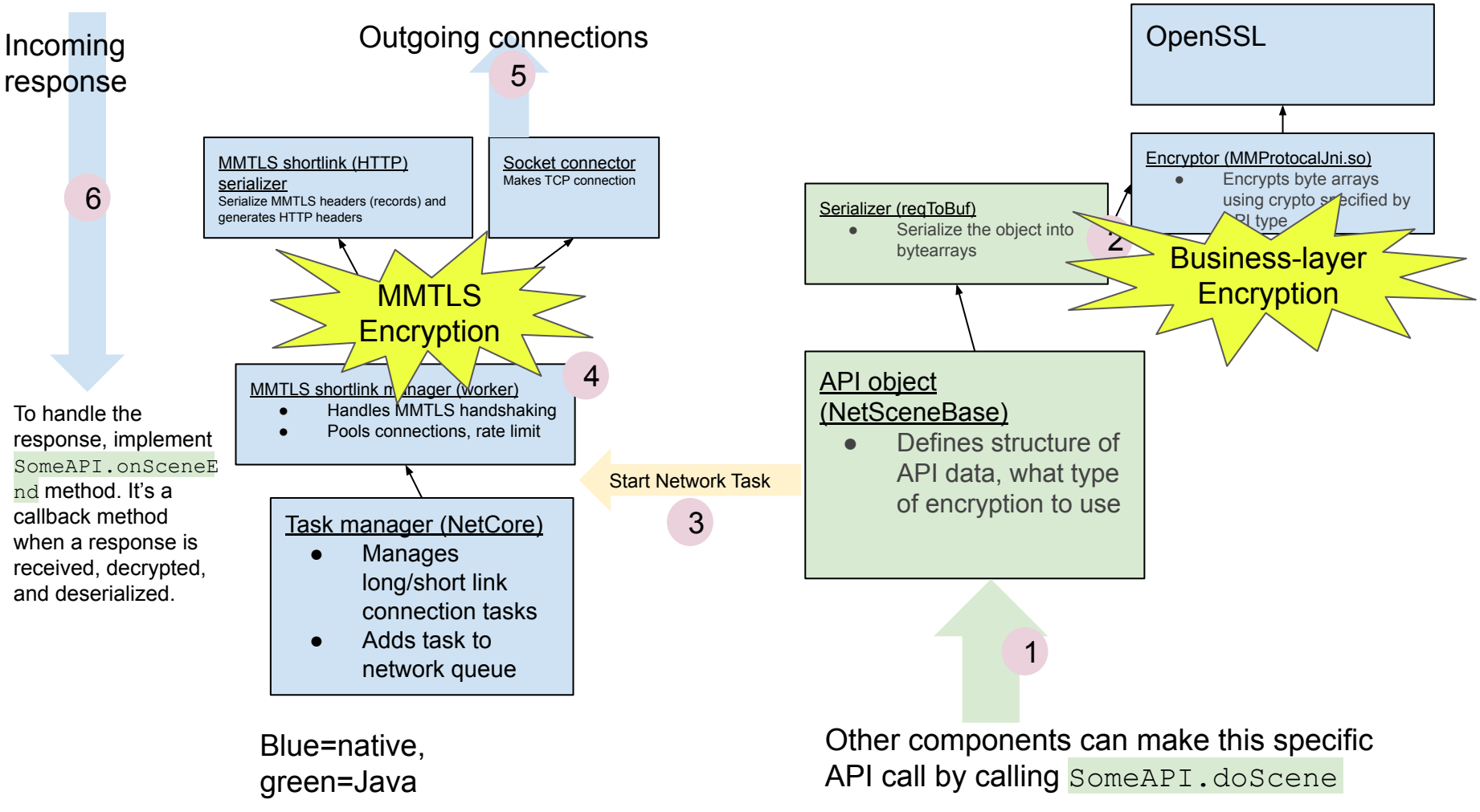
- Request and response formats are defined using Protobuf
- Screenshot shows a portion of the request Protobuf fields



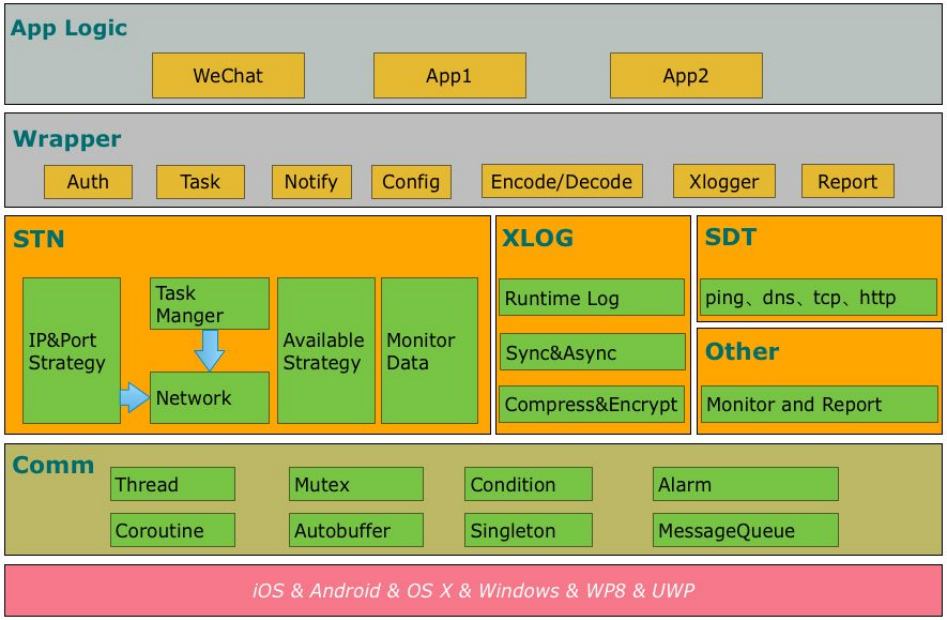
Other components can make this specific API call by calling `SomeAPI.doScene`


OpenSSL
Encryptor (MMProtocalJni.so)
Encrypts byte arrays using crypto specified by API type
Serializer (reqToBuf)
Serialize the object into bytearrays
2
Business-layer Encryption
API object (NetSceneBase)
Defines structure of API data, what type of encryption to use
Start Network Task
3
Task manager (NetCore)
Manages long/short link connection tasks
Adds task to network queue
1
Blue=native, green=Java
Other components can make this specific API call by calling SomeAPI.doScene


Outgoing connections
5
OpenSSL
MMTLS shortlink (HTTP) serializer
Serialize MMTLS headers (records) and generates HTTP headers
Socket connector
Makes TCP connection
Encryptor (MMProtocalJni.so)
Encrypts byte arrays using crypto specified by API type
Serializer (reqToBuf)
Serialize the object into bytearrays
2
Business-layer Encryption
MMTLS Encryption
MMTLS shortlink manager (worker)
Handles MMTLS handshaking
Pools connections, rate limit
4
API object (NetSceneBase)
Defines structure of API data, what type of encryption to use
Start Network Task
3
Task manager (NetCore)
Manages long/short link connection tasks
Adds task to network queue
1
Blue=native, green=Java
Other components can make this specific API call by calling SomeAPI.doScene

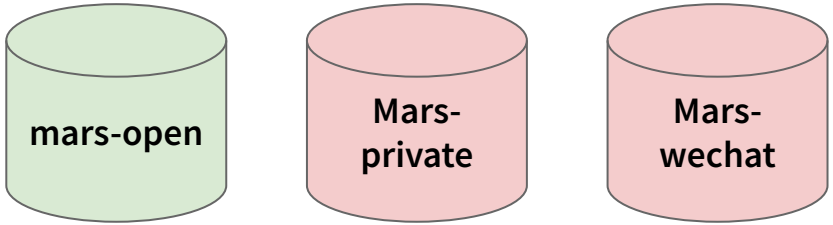

Incoming response
6
To handle the response, implement SomeAPI.onSceneEnd method. It's a callback method when a response is received, decrypted, and deserialized.
Outgoing connections
5
MMTLS shortlink (HTTP) serializer
Serialize MMTLS headers (records) and generates HTTP headers
Socket connector
Makes TCP connection
MMTLS Encryption
MMTLS shortlink manager (worker)
Handles MMTLS handshaking
Pools connections, rate limit
4
Task manager (NetCore)
Manages long/short link connection tasks
Adds task to network queue
Start Network Task
3
API object (NetSceneBase)
Defines structure of API data, what type of encryption to use
1
Other components can make this specific API call by calling SomeAPI.doScene
Serializer (reqToBuf)
Serialize the object into bytearrays
2
OpenSSL
Encryptor (MMProtocalJni.so)
Encrypts byte arrays using crypto specified by API type
Business-layer Encryption
Blue=native, green=Java

One more thing...

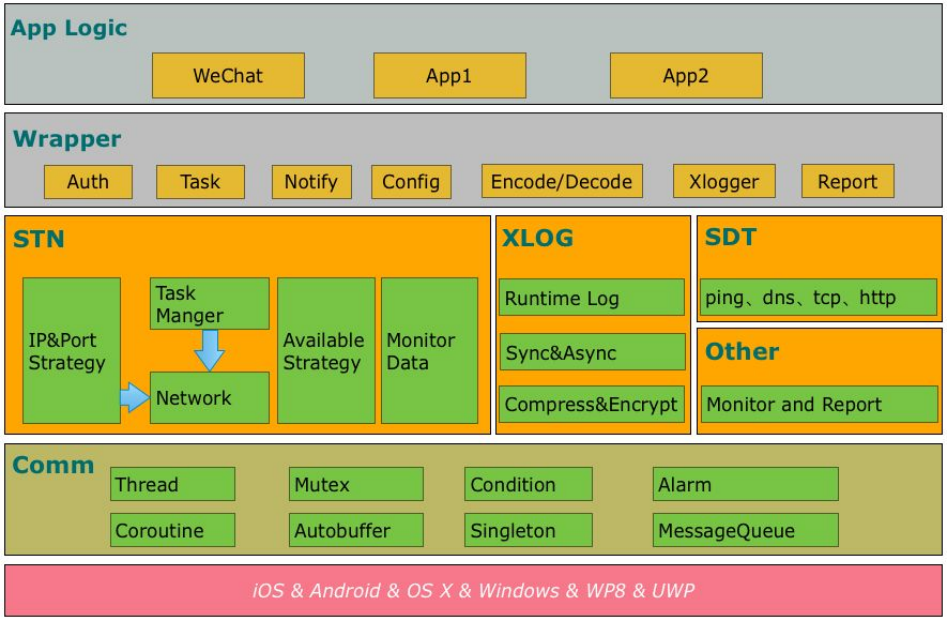


"Mars"

- Mars is Tencent's cross-platform infrastructure component, written in C++
- Network requests are handled by submodule "*STN*"
- Mars is partially open source
  - *mars-open* is the open source part
  - *mars-private* : "potentially open sourced"
  - *mars-wechat* : wechat-specific code, including MMTLS encryption



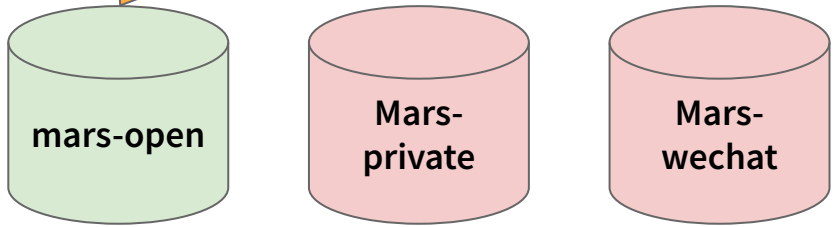
One more thing...



"Mars"

- Mars is Tencent's cross-platform infrastructure component, written in C++
- Network requests are handled by submodule "*STN*"
- Mars is partially open source
  - *mars-open* is the open source part
  - *mars-private* : "potentially open sourced"

Mars-open helps us reverse engineer other closed-source parts ;-)



WeChat network encryption

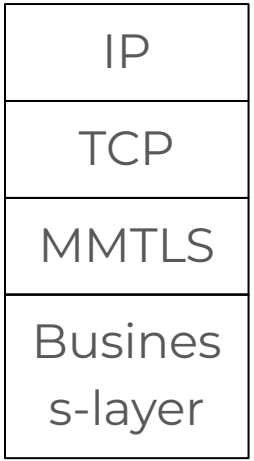
How does WeChat encrypt requests?

How does WeChat encrypt requests?

How does WeChat encrypt requests?

Two transport protocols: **Longlink** and **Shortlink**

Longlink

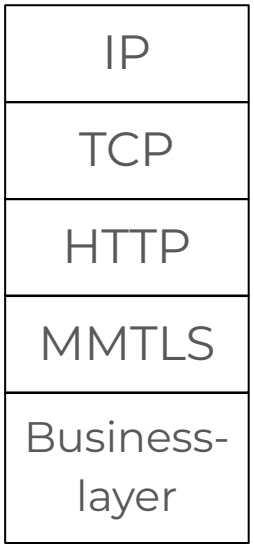


- TCP, port 8080
- Long-lived connection
- Supports multiple request-response cycle
- Likely used for server-initiated transmissions

How does WeChat encrypt requests?

Two transport protocols: **Longlink** and **Shortlink**

Shortlink



- HTTP POST, port 80
- Short-lived connection
- Supports single request-response cycle
- Used for most client-initiated transmissions

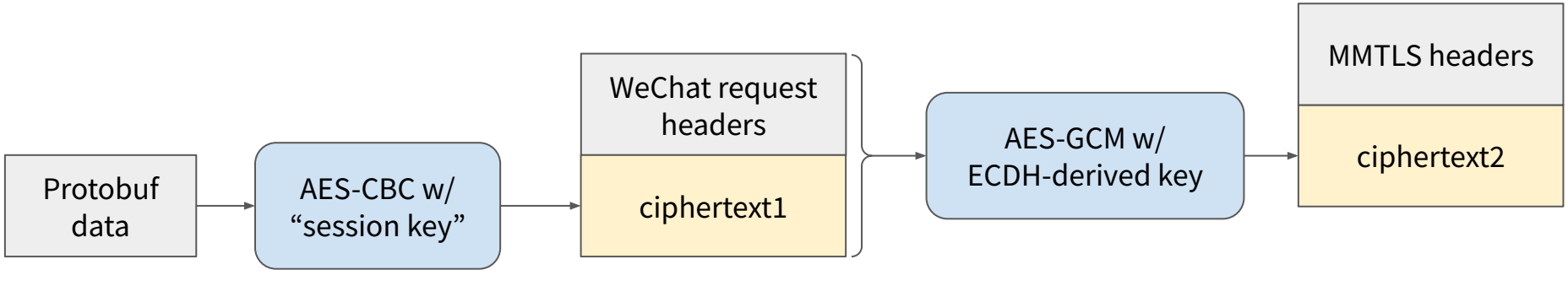
How does WeChat encrypt requests?

Encrypted **twice** (and also differently if you're logged-out)

	Key derivation	Encryption	Library
MMTLS layer	DH with resumption	AES-GCM with tag	`libwechatnetwork.so`
Business-layer, logged-out	Static DH	AES-GCM with tag	`libwechatmm.so`
Business-layer, logged-in	Fixed key from server	AES-CBC with checksum	`libMMProtocalJNI.so`

How does WeChat encrypt requests?

Logged-in example of network request encryption:



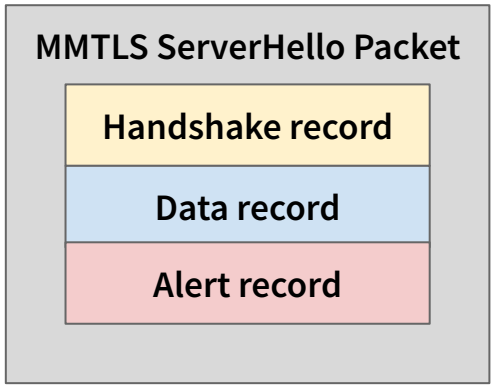
“Business-layer” encryption

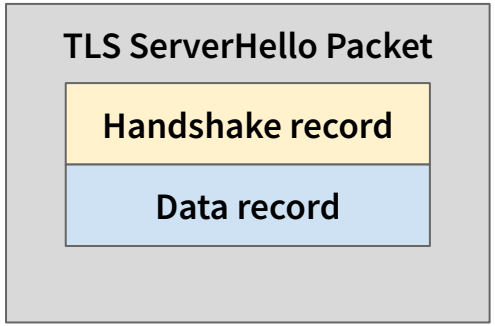
- Found and reported many issues

“MMTLS” encryption

- Added in 2016

MMTLS “records”





MMTLS

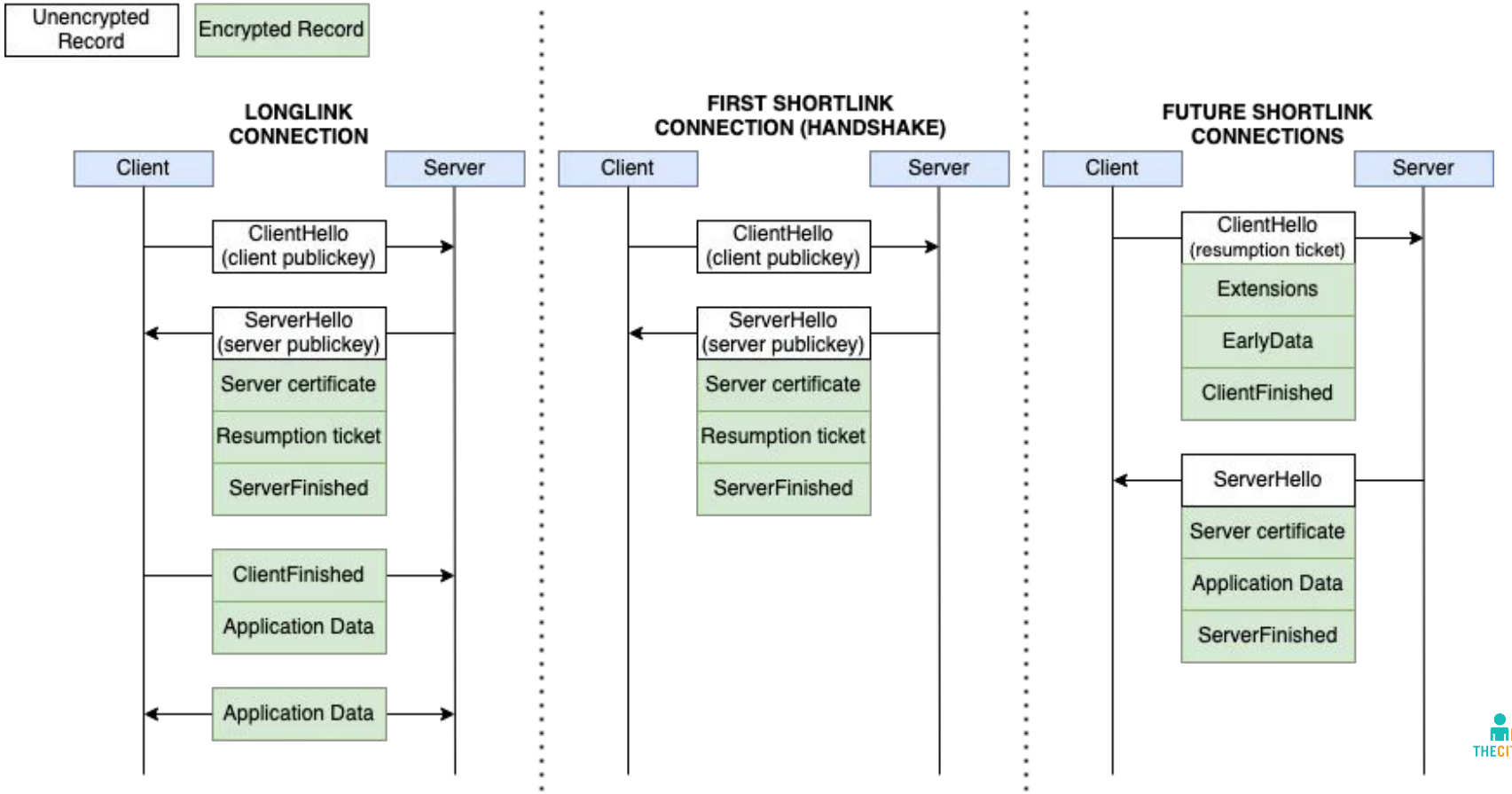
MMTLS record headers

`19 f1 04`	Handshake resumption
`16 f1 04`	Handshake
`17 f1 04`	Data
`15 f1 04`	Alert

TLS record headers

`16 03 04`	Handshake
`17 03 04`	Data
`15 03 04`	Alert

MMTLS handshake



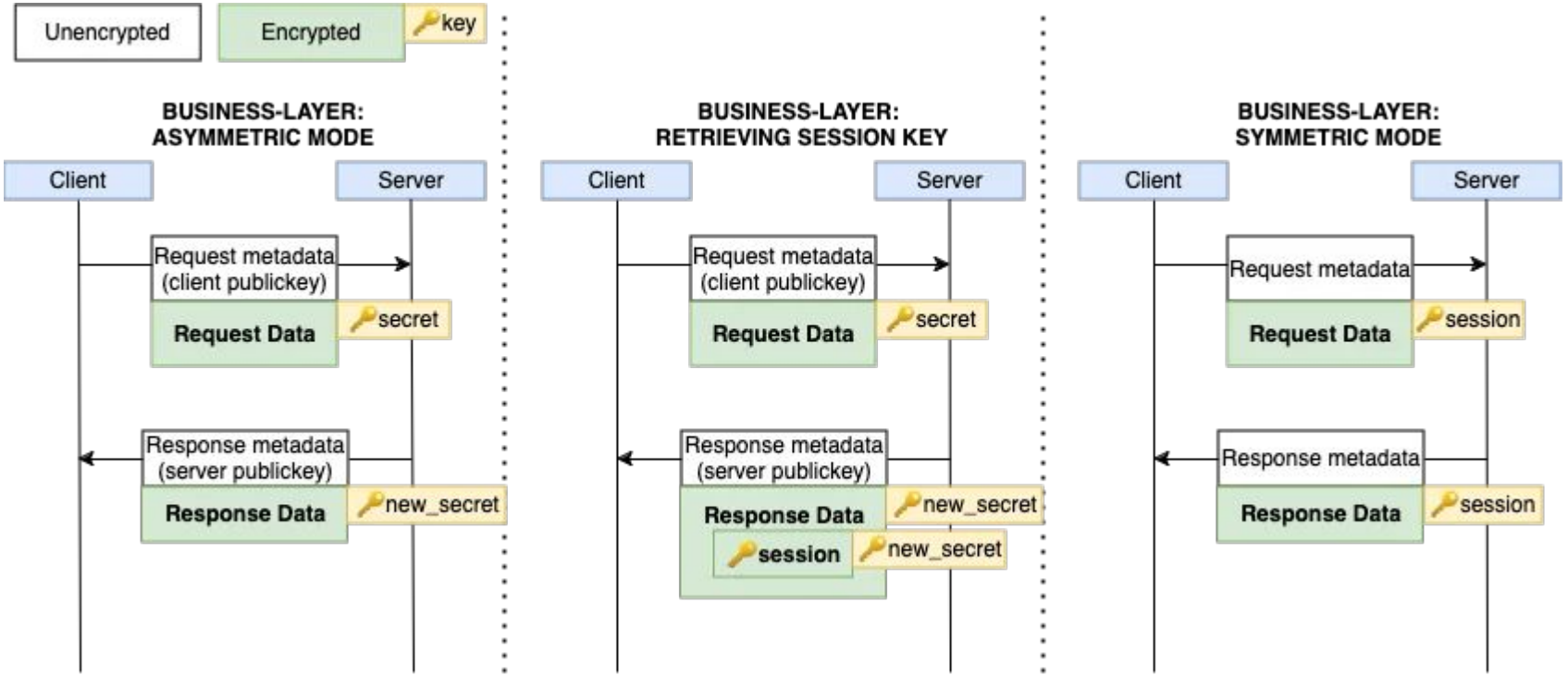
THECITIZENLAB

MMTLS Layer

- Modifications from TLS 1.3:
  - Limited ciphersuite selection, pinned keys and certificate (since WeChat controls both client and server)
- AES-GCM + tag for encryption, authenticity
- Public documentation on Github

- **Public flaws:** lack of forward secrecy, heavy use of session resumption implies no replay resistance

Business-Layer



Logged out

LoggING in

Logged in

THECITIZENLAB

Business-Layer (Logged out)

- Static Diffie-Hellman
  - static public server key + newly generated client key to generate session key
  - No forward secrecy (e.g. if static private server key is compromised, all session can be compromised)
- AES-GCM + tag for encryption, authenticity

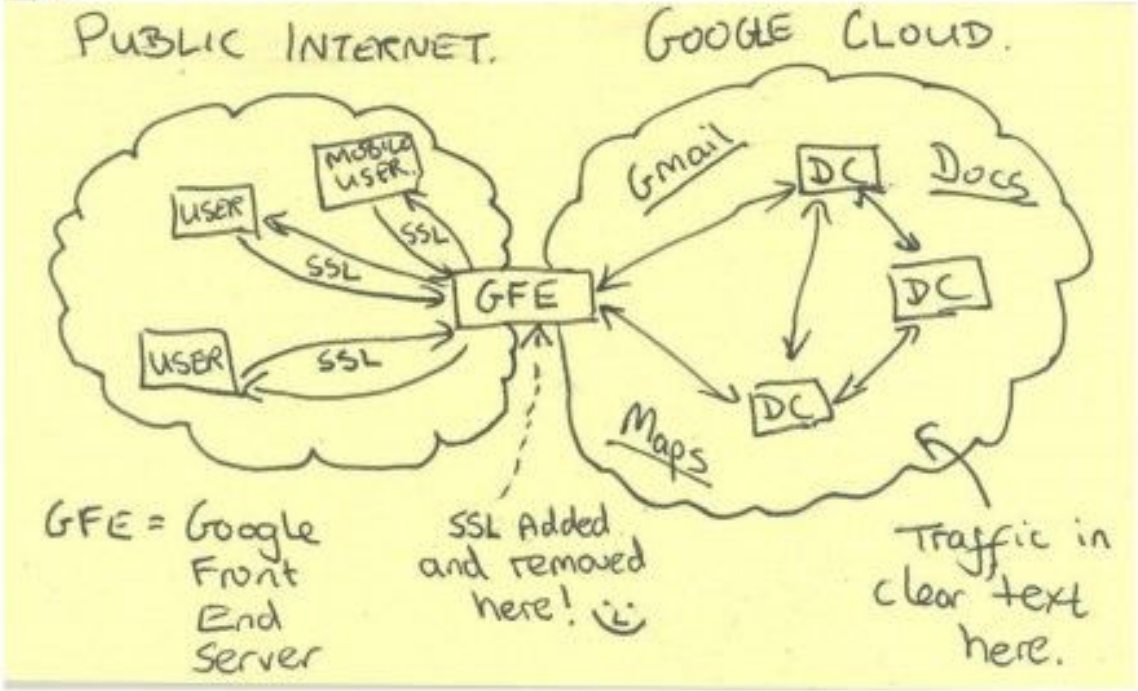
Business-Layer (Logged in)

- Uses key given by server
  - Server sends key to client encrypted with “logged-out” encryption– highly unusual!
- AES-CBC + checksum
  - Checksum is forgeable and provides no cryptographic guarantees
- Prior to 2016, this was the only layer of encryption…
  - But it leaks metadata such as user ID and request URI
  - Acknowledged by Tencent to be one reason to develop MMTLS Encryption

Disclosure

- We reported to Tencent, suggested to switch to QUIC/TLS1.3 or remove Business-layer encryption altogether
- They replied saying they would **upgrade** Business-layer encryption to use AES-GCM instead of AES-CBC
- ???
- Possibly, **Business-layer encryption** is the *only* layer of encryption within WeChat internal networks
    - This is also bad: means WeChat data could be subject to surveillance 

Current Efforts - Google



Discussion

It’s not just WeChat…

Source: https://www.businessofapps.com/data/most-popular-apps/ (1/30/2025)

THECITIZENLAB

Which always use HTTPS/TLS?

❌	WeChat	1,012
❌	Alipay	901
❌	Taobao	795
✅	TikTok	773
✅	Instagram	759
❌	Pinduoduo	728

✅	Douyin	695
❌	QQ	583
✅	Facebook	571
✅	WhatsApp	527
❌	Baidu	491
❌	Kuaishou	480

*but they're also **not not** encrypting…

they are often **using proprietary cryptography**

THECITIZENLAB

HTTPS adoption on mobile?

Google Play

12.9% of top 1k apps sent plaintext traffic.

3.5% of top 1k apps used proprietary cryptography.

Xiaomi Store (Chinese version)

65.4% of top 1k apps sent plaintext traffic.

47.6% of top 1k used proprietary cryptography!

Is the proprietary cryptography secure?

We manually analyzed the **9** most popular proprietary protocols globally…

8 contained severe vulnerabilities where we broke the encryption!

The remaining one was **MMTLS… !**

Why does this matter?

- Bad encryption enables ***mass surveillance and MITM.***
- If apps use bad encryption, users of those apps are more vulnerable to mass surveillance by **all governments and attackers**.
- Not just Chinese people are affected!
  - Chinese apps have sizable international user base
    - E.G. RedNote/XiaoHongShu: https://citizenlab.ca/2025/02/network-security-issues-in-rednote/
  - Non-Chinese apps may still use Chinese SDK

THECITIZENLAB

TOP SECRET STRAP1

Response to improving security

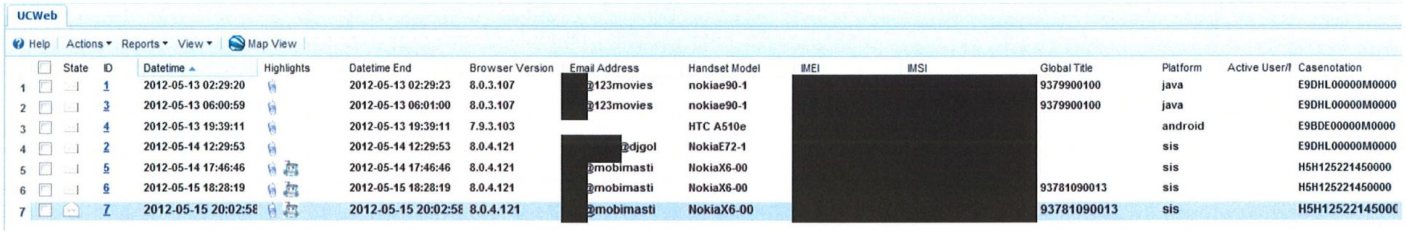
- For the past decade, NSA has lead an aggressive, multi-pronged effort to break widely used Internet encryption technologies
- Cryptanalytic capabilities are now coming on line
- Vast amounts of encrypted Internet data which have up till now been discarded are now exploitable
- Major new processing systems, SIGDEV efforts and tasking must be put in place to capitalize on this opportunity

PTD "We penetrate targets' defences."

Success Stories

* UCWeb mobile browser identification
  * Discovered by GCHQ analyst during DSD workshop
  * Chinese mobile web browser – leaks IMSI, MSISDN, IMEI and device characteristics

UCWeb – XKS Microplugin



UCWeb

Help | Actions | Reports | View | Map View

	State	ID	Datetime	Highlights	Datetime End	Browser Version	Email Address	Handset Model	IMEI	IMSI	Global Title	Platform	Active User/I	Casenotation
1		1	2012-05-13 02:29:20		2012-05-13 02:29:23	8.0.3.107	[redacted]@123movies	nokiae90-1	[redacted]		9379900100	java		E9DHL00000M0000
2		3	2012-05-13 06:00:59		2012-05-13 06:01:00	8.0.3.107	[redacted]@123movies	nokiae90-1			9379900100	java		E9DHL00000M0000
3		4	2012-05-13 19:39:11		2012-05-13 19:39:11	7.9.3.103		HTC A510e				android		E9BDE00000M0000
4		2	2012-05-14 12:29:53		2012-05-14 12:29:53	8.0.4.121	[redacted]@djgol	NokiaE72-1				sis		E9DHL00000M0000
5		5	2012-05-14 17:46:46		2012-05-14 17:46:46	8.0.4.121	[redacted]@mobimasti	NokiaX6-00				sis		H5H125221450000
6		6	2012-05-15 18:28:19		2012-05-15 18:28:19	8.0.4.121	[redacted]@mobimasti	NokiaX6-00			93781090013	sis		H5H125221450000
7		7	2012-05-15 20:02:58		2012-05-15 20:02:58	8.0.4.121	[redacted]@mobimasti	NokiaX6-00			93781090013	sis		H5H12522145000

Discussion:
Why do Chinese apps prefer custom cryptography?

- Distrust in TLS?
  - Early concerns on TLS Certificate Authority ecosystem circa 2011
- Obfuscation mechanism?
  - Why not just use commercial packers?
- OS performance / compatibility ?
  - Fragmented Android OS and app store ecosystem increases the need for dirty patches and workarounds
- Network filtering?
  - Prevalent ISP filtering and traffic poisoning prompts the need for dirty patches
- “Not invented here” problem?
- Technical debt / inertia

Discussion:
Why do Chinese apps prefer custom cryptography?

- Distrust in TLS?
  - Early concerns on TLS Certificate Authority ecosystem circa 2011
- Obfuscation mechanism?
  - Why not just use commercial packers?
- OS performance / compatibility ?
  - Fragmented Android OS and app store ecosystem, patches and workarounds
- Network filtering?
  - Prevalent ISP filtering and traffic poison
- “Not invented here” problem?
- Technical debt / inertia

Discussion:
How can we improve security in ~~Chinese~~ ALL apps?

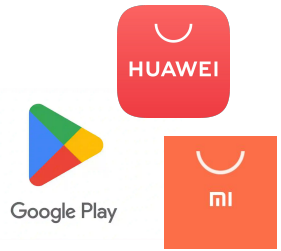
- Continued study of privacy and security of consumer apps?
- Researchers should engage more with Global South developers and security engineers?
- App store reviews/attestation of network security?
- OS vendors should provide better documentation, easy-to-use development tools?

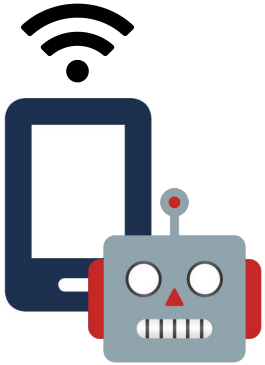
Thank you! Questions?

pellaeon@citizenlab.ca
monaw@princeton.edu

Link to full report

Appendix





Scrape popular applications

Install app

Simulate user behavior

Isolate non-TLS traffic

Entropy analysis

Protocol clustering

THECITIZENLAB

Is the proprietary cryptography secure?

Protocol family	# apps	Cumulative downloads	Most downloaded app	MAU	Decryptable	Fixed?	Decrypted request contents
Kuaishou SDK	76	35.10B	Kuaishou	692 mill	YES	YES	Device metadata
[redacted]	82	30.30B	[redacted]	[redacted]	YES	Tried to	Device metadata
[redacted]	15	25.43B	[redacted]	[redacted]	YES	NO	Browsing data
[redacted]	11	18.10B	[redacted]	[redacted]	YES	NO	DNS requests
[redacted]	7	17.62B	[redacted]	[redacted]	YES	NO	Browsing data
iQIYI	3	11.28B	iQIYI	429 mill	YES	YES	Network metadata
[redacted]	37	10.34B	[redacted]	[redacted]	YES	NO	Security config*
MMTLS	**1**	**9.23B**	**WeChat**	**1.3 bill**	**NO**	**-**	**-**
[redacted]	38	9.02B	[redacted]	[redacted]	YES	NO	Device metadata

*contained vuln s.t. network attackers can read file contents on users phones

THECITIZENLAB