Breaking network crypto in popular Chinese mobile apps

Mona - CCC 2025

This talk is not just about surveillance in China

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."

Adversaries

- Anyone on your network
- Your ISP
- The server’s ISP
- Every network in between
- Every state actor in between

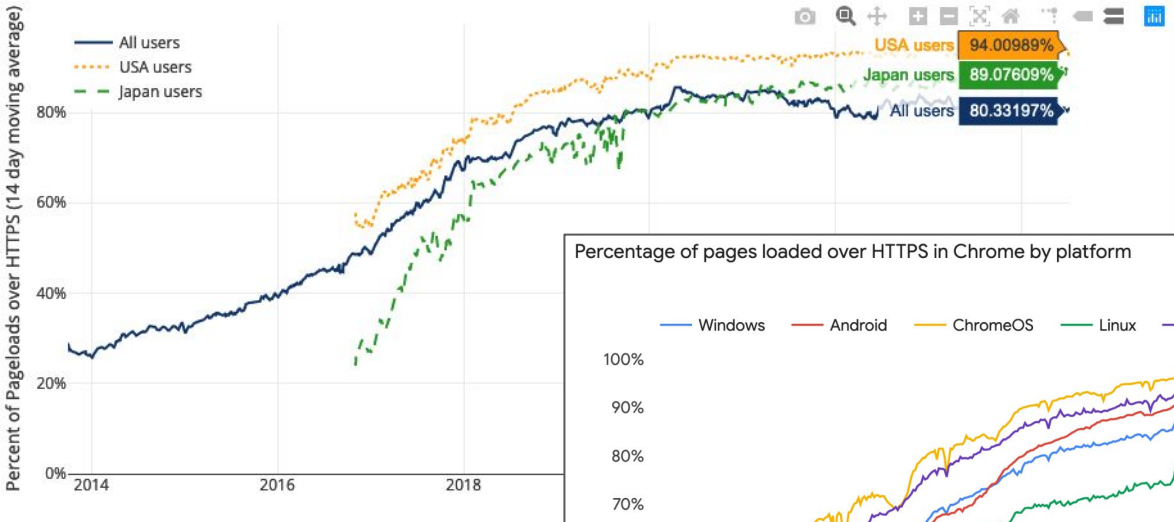
Encryption limits network surveillance

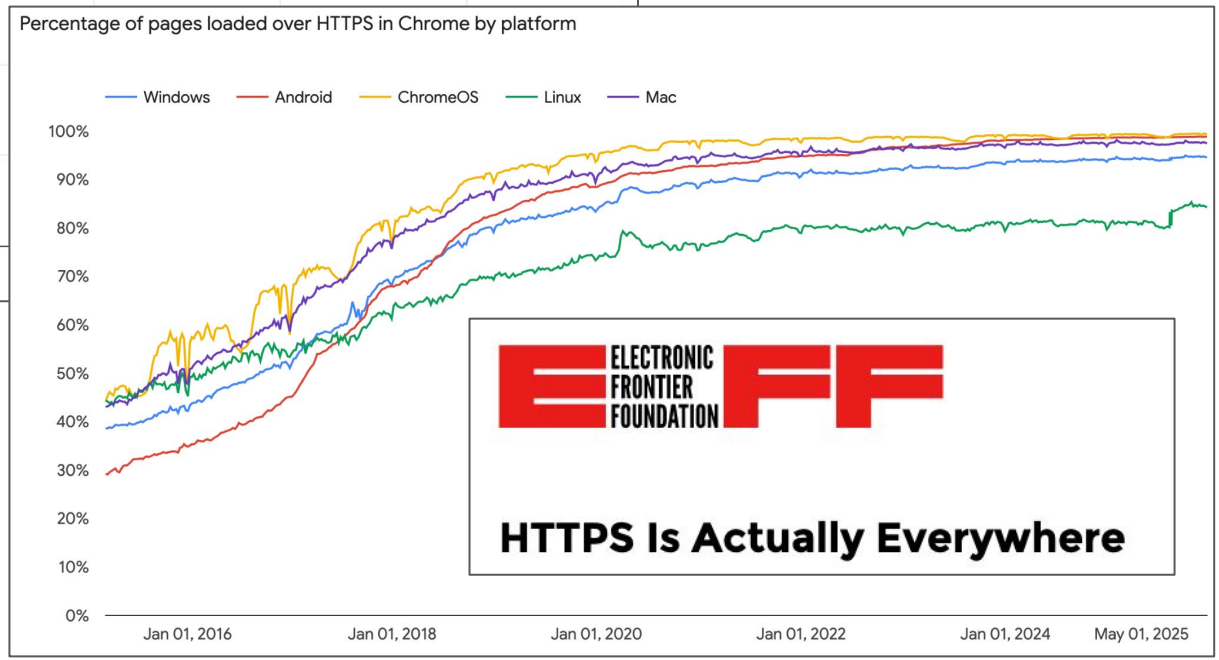
Today, TLS is the de-facto standard…

Use TLS! Don’t roll your own crypto!

Percentage of Web Pages Loaded by Firefox Using HTTPS

(14-day moving average, source: Firefox Telemetry)



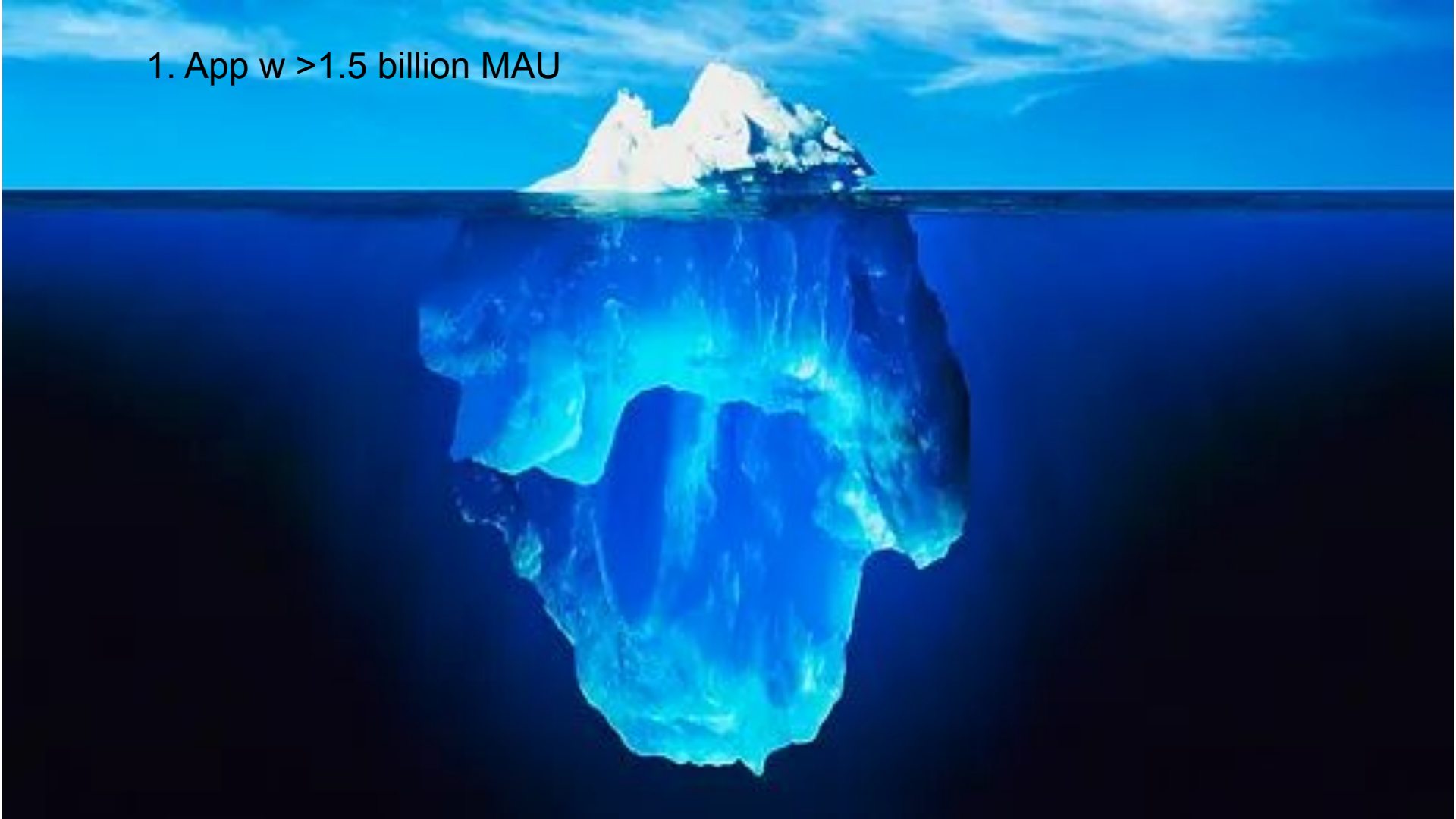


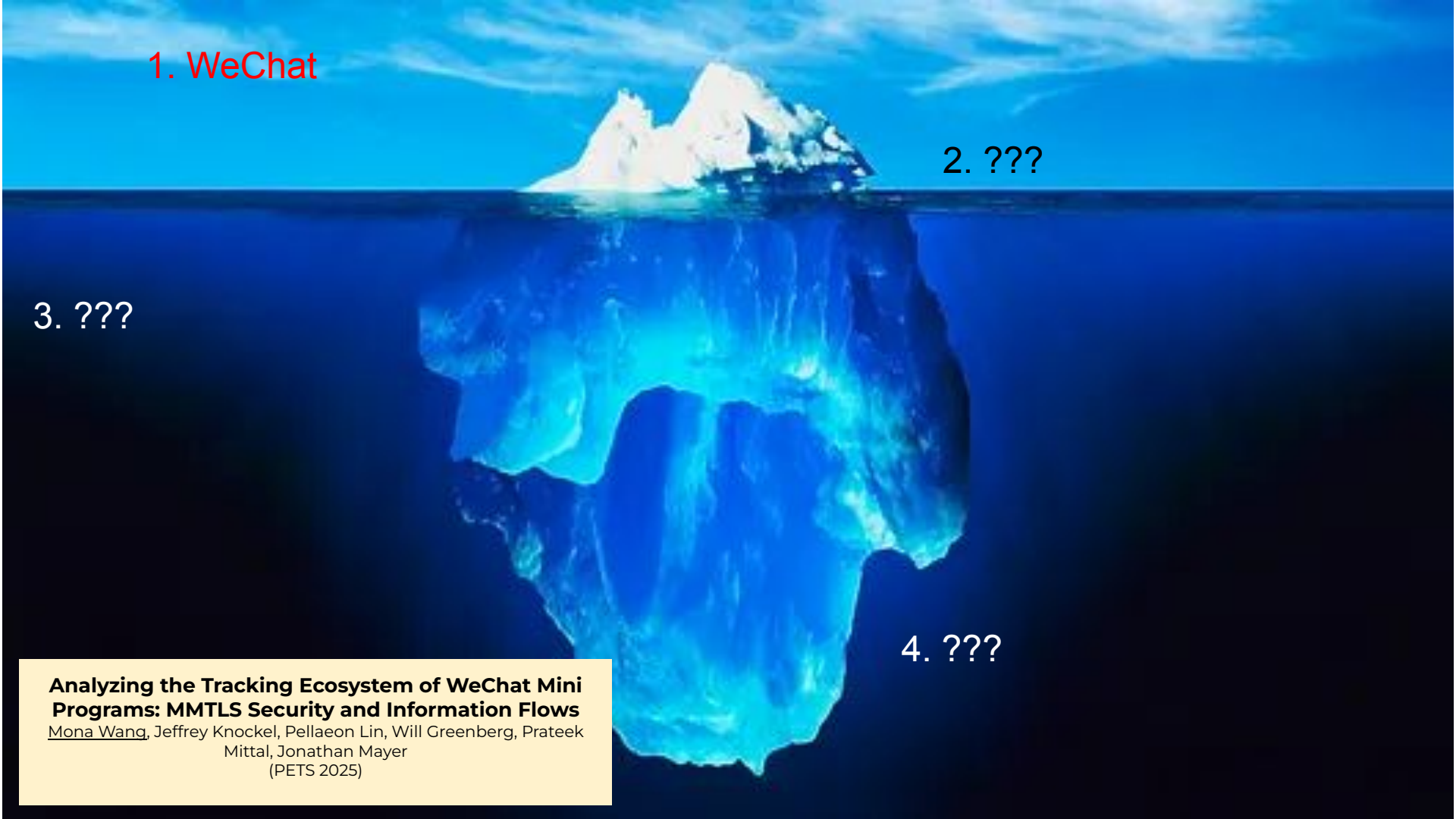
But what about mobile?

Today, mobile traffic surpasses web network traffic by ~2x globally.

Widespread assumption in the security community that TLS trends on web transfer to mobile… but do they?

1. App w >1.5 billion MAU



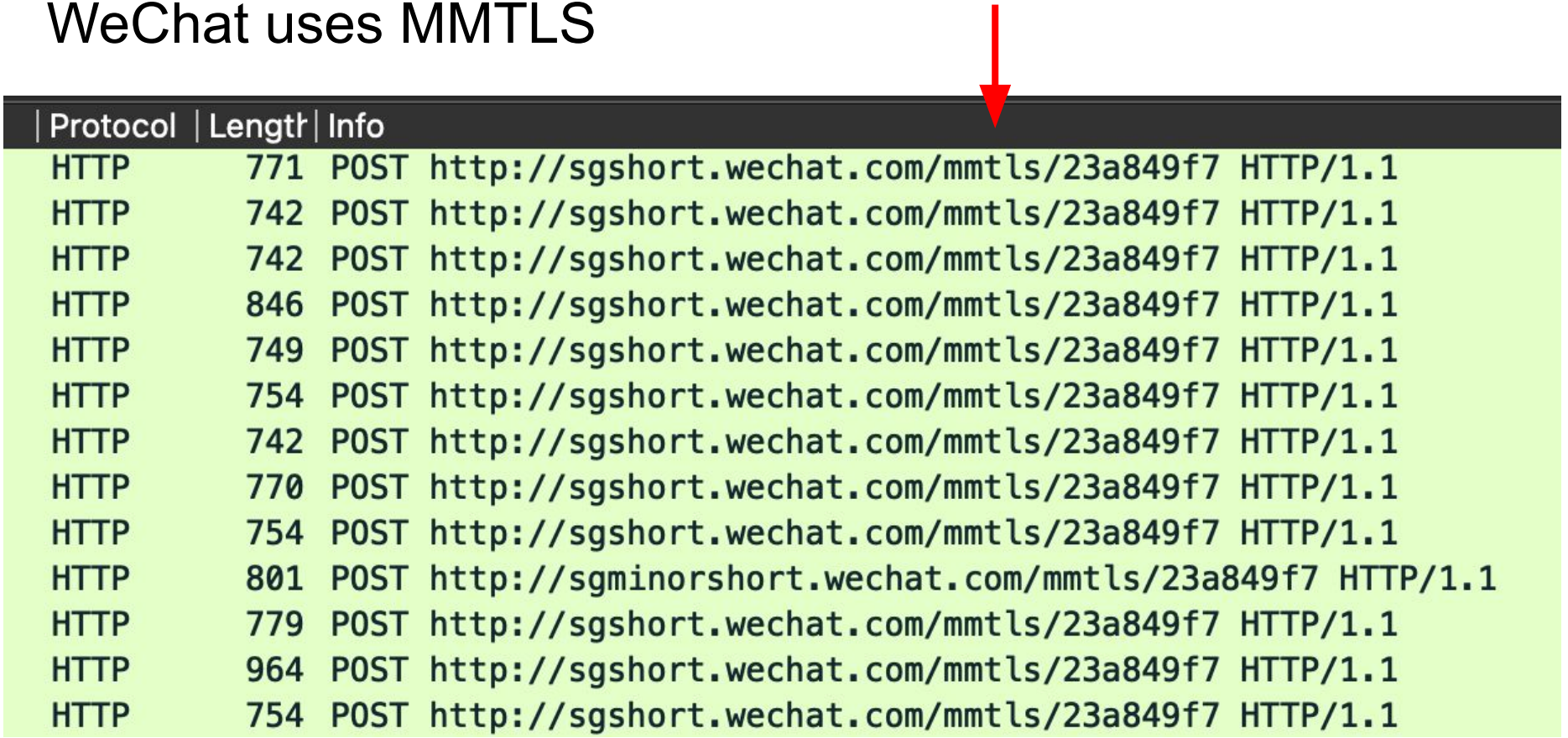


Analyzing the Tracking Ecosystem of WeChat Mini Programs: MMTLS Security and Information Flows

Mona Wang, Jeffrey Knockel, Pellaeon Lin, Will Greenberg, Prateek Mittal, Jonathan Mayer

(PETS 2025)

WeChat uses MMTLS



Protocol	Length	Info
HTTP	771	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	742	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	742	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	846	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	749	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	754	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	742	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	770	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	754	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	801	POST http://sgminorshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	779	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	964	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1
HTTP	754	POST http://sgshort.wechat.com/mmtls/23a849f7 HTTP/1.1

What is MMTLS?

- WeChat’s custom network encryption
- Responsible for encrypting over one billion users’ network traffic
- There’s only one document published by Tencent about it on Github[1]

[1] WeChat Mobile Development Team, 基于TLS1.3的微信安全通信协议mmtls介绍/“Introducing WeChat’s secure communication protocol mmtls based on TLS1.3”, Github

How does WeChat encrypt requests?

- HTTPS → QUIC are used for large file downloads from CDNs
- MMTLS is used for everything else
  - Sending/receiving messages
  - Advertisements
  - Search
  - Payments
  - Moments
  - Analytics during Mini Program usage

How does WeChat encrypt requests?

Business-layer

MMTLS

transport

How does WeChat encrypt requests?

Two transports: **Longlink** and **Shortlink**

Longlink

- TCP, port 8080
- Long-lived connection
- Multiple request-responses

- used for, e.g.
  - WeChat messages

Shortlink

- **HTTP POST**, port 80
- Short-lived connection
- Supports single request-response

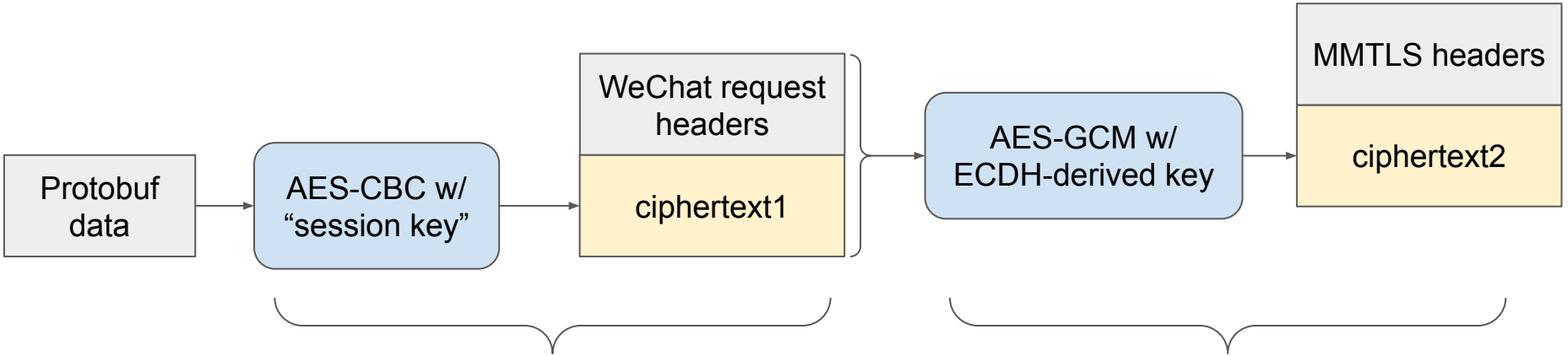
- used for, e.g.
  - Mini program, search analytics

How does WeChat encrypt requests?

	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?

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



"Business-layer" encryption
- Found and reported many issues
- Tencent responds and says they'd fix them???

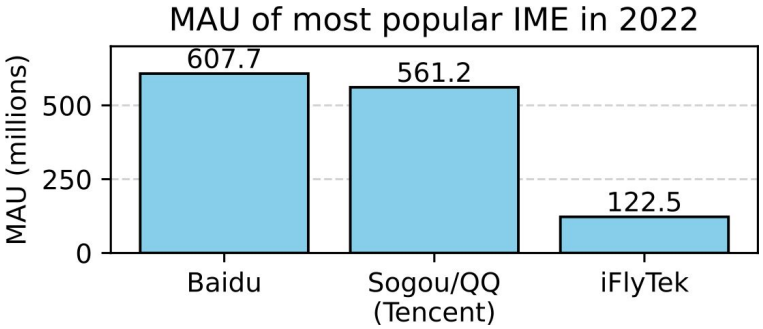
"MMTLS" encryption
- Added in 2016

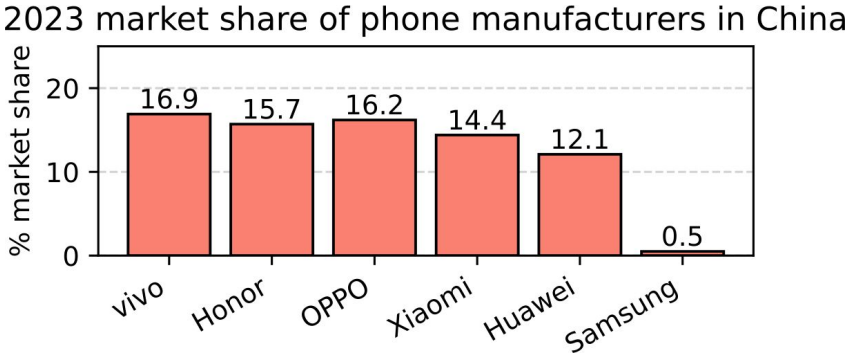
The Not-So-Silent Type: Vulnerabilities in Chinese IME Keyboards' Network Security Protocols

Jeffrey Knockel, Mona Wang, Zoë Reichert

ACM CCS 2024

Landscape of third-party Chinese keyboards (IMEs)





- IME: Input Method Editor
- Keyboard software
- Third-party keyboard software is essential for typing Chinese

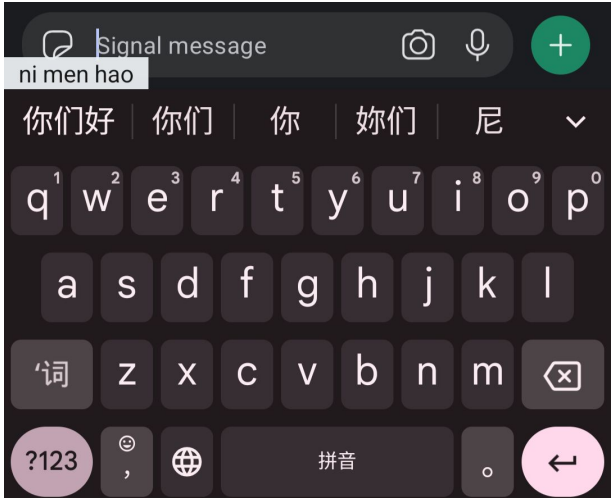
What’s a Chinese IME?

What’s a Chinese IME?

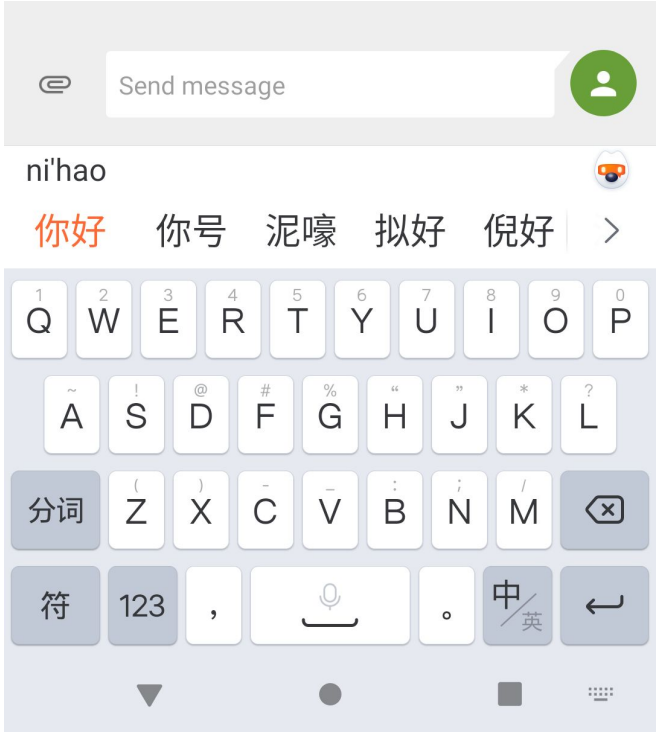
What’s a Chinese IME?

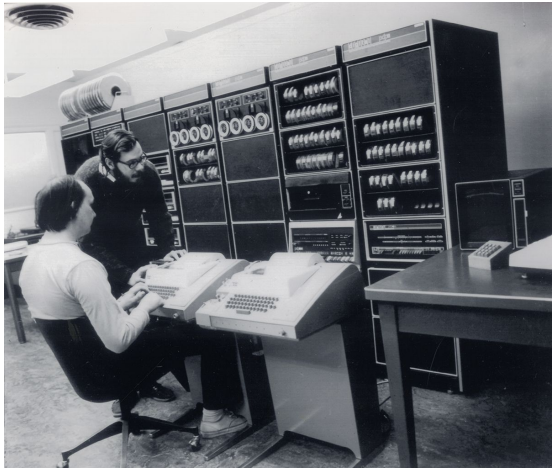
What's a Chinese IME?

What’s a Chinese IME?

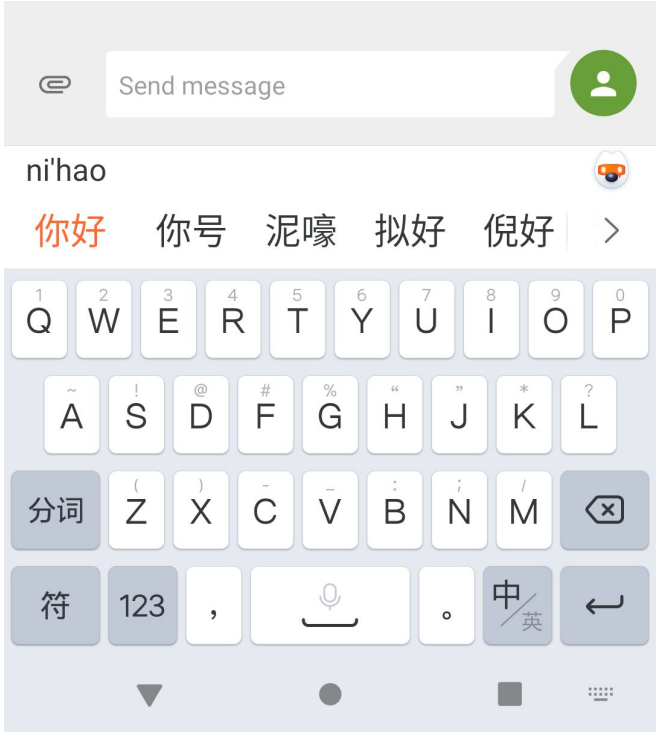


The advent of “cloud-based” prediction





The advent of “cloud-based” prediction



Google Faces Plagiarism Questions Over Chinese Software

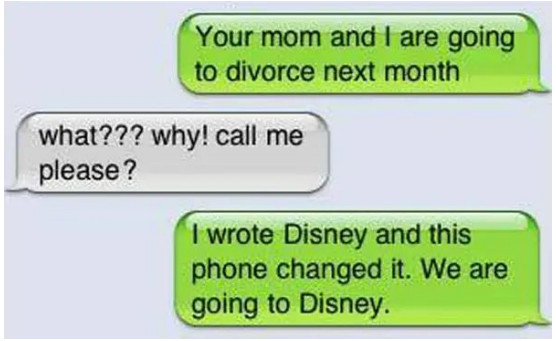
187

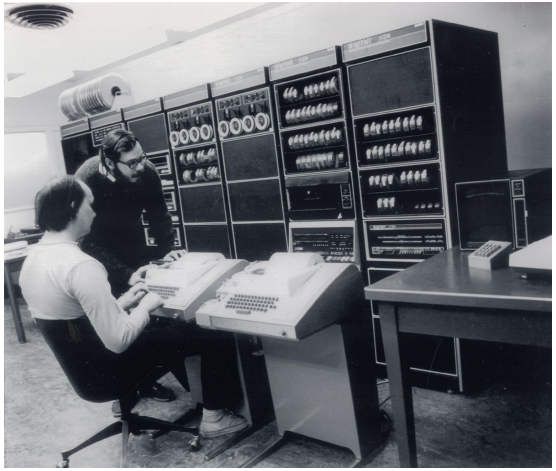
Posted by Zonk on Sunday April 08, 2007 @03:03PM from the i'll-just-take-a-

yaohua2000 writes

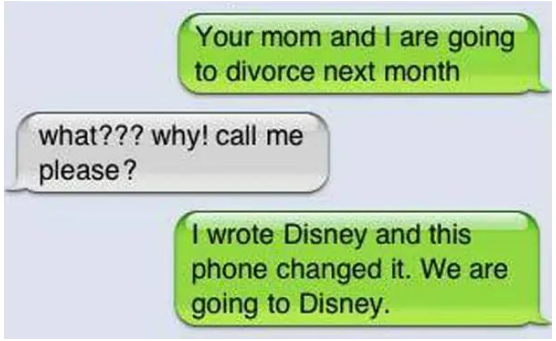
> "Google's laboratory in China has launched its first product, a Pinyin Input Method Editor. The software allows the romanized characters to be translated to more traditional Chinese symbols , via entering on a QWERTY keyboard. Users soon discovered that the data Google used for the product was unusually similar to the data used by a Chinese rival, Sogou. Google has evaded the question about software similarities, reports PC World. 'The similarities, which included an error involving the name of a celebrity, were noted on a Google Labs discussion board about its Pinyin IME. Users noted that entering the Pinyin pinggong into the Google IME incorrectly produced the name of Feng Gong, an actor and comedian.'"

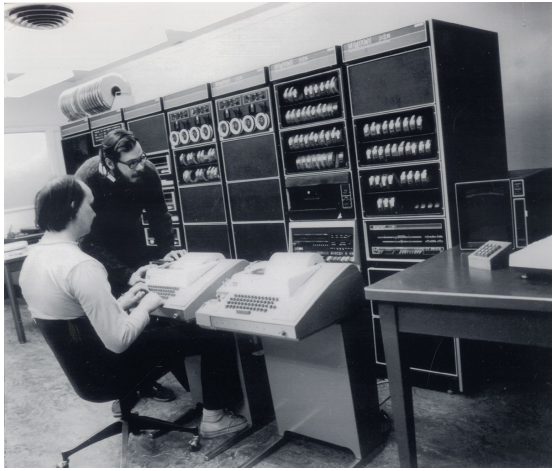
The advent of “cloud-based” prediction





The advent of “cloud-based” prediction





Google Faces Plagiarism Questions Over Chinese Software

187

Posted by Zonk on Sunday April 08, 2007 @03:03PM from the i'll-just-take-a-

yaohua2000 writes

> "Google's laboratory in China has launched its first product, a Pinyin Input Method Editor. The software allows the romanized characters to be translated to more traditional Chinese symbols , via entering on a QWERTY keyboard. Users soon discovered that the data Google used for the product was unusually similar to the data used by a Chinese rival, Sogou. Google has evaded the question about software similarities, reports PC World. 'The similarities, which included an error involving the name of a celebrity, were noted on a Google Labs discussion board about its Pinyin IME. Users noted that entering the Pinyin pinggong into the Google IME incorrectly produced the name of Feng Gong, an actor and comedian.'"

They are keyloggers

You Shouldn’t Collect My Secrets: Thwarting Sensitive Keystroke Leakage in Mobile IME Apps
Jin Chen, Haibo Chen, Erick Bauman, Zhiqiang Lin, Binyu Zang, Haibing Guan
Usenix Security 2015

They are keyloggers

You Shouldn’t Collect My Secrets: Thwarting Sensitive Keystroke Leakage in Mobile IME Apps
Jin Chen, Haibo Chen, Erick Bauman, Zhiqiang Lin, Binyu Zang, Haibing Guan
Usenix Security 2015

…this talk is **not** about how they are keyloggers

Legend	
✘✘	working exploit created to decrypt transmitted keystrokes for both **active and passive** eavesdroppers
✘	working exploit created to decrypt transmitted keystrokes for an **active** eavesdropper
!	weaknesses present in cryptography implementation
✔	no known issues
N/A	product not offered or not present on device analyzed

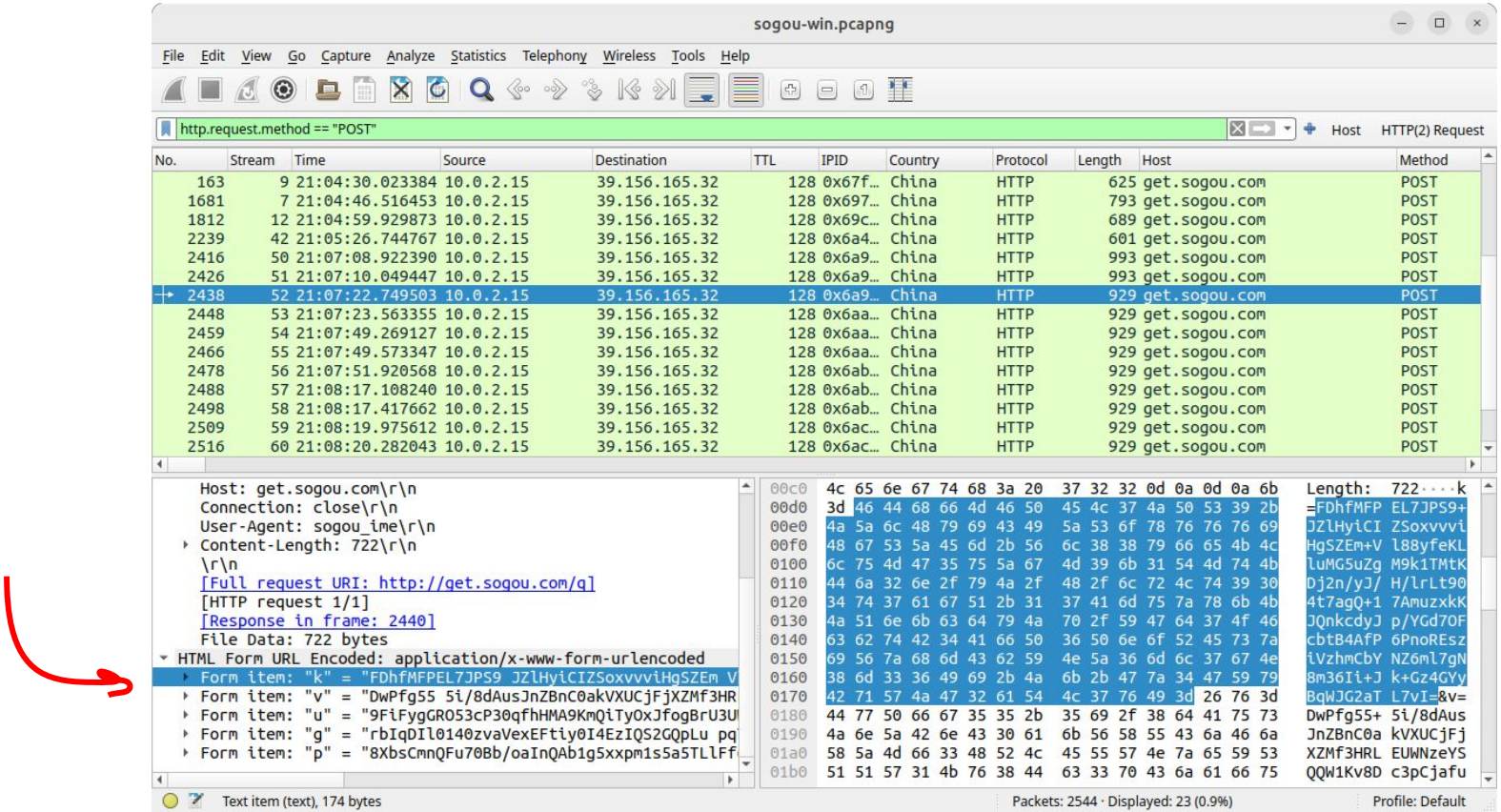
Keyboard developer	Android	iOS	Windows
Tencent	✘	✔	✘
Baidu	!	!	✘✘
iFlytek	✘✘	✔	✔

Legend	
✘✘	working exploit created to decrypt transmitted keystrokes for both **active and passive** eavesdroppers
✘	working exploit created to decrypt transmitted keystrokes for an **active** eavesdropper
!	weaknesses present in cryptography implementation
✔	no known issues
N/A	product not offered or not present on device analyzed

Device manufacturer	Own	Sogou	Baidu	iFlytek	iOS	Windows
Samsung	✘✘	✔*	✘✘	N/A	N/A	N/A
Huawei	✔*	✔	N/A	N/A	N/A	N/A
Xiaomi	N/A	✘*	✘✘	✘✘	N/A	N/A
OPPO	N/A	✘	✘✘*	N/A	N/A	N/A
Vivo	✔*	✘	N/A	N/A	N/A	N/A
Honor	N/A	N/A	✘✘*	N/A	N/A	N/A

* Default keyboard on device

Sogou IME


sogou-win.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http.request.method == "POST"
Host HTTP(2) Request
No. Stream Time Source Destination TTL IPID Country Protocol Length Host Method
163 9 21:04:30.023384 10.0.2.15 39.156.165.32 128 0x67f… China HTTP 625 get.sogou.com POST
1681 7 21:04:46.516453 10.0.2.15 39.156.165.32 128 0x697… China HTTP 793 get.sogou.com POST
1812 12 21:04:59.929873 10.0.2.15 39.156.165.32 128 0x69c… China HTTP 689 get.sogou.com POST
2239 42 21:05:26.744767 10.0.2.15 39.156.165.32 128 0x6a4… China HTTP 601 get.sogou.com POST
2416 50 21:07:08.922390 10.0.2.15 39.156.165.32 128 0x6a9… China HTTP 993 get.sogou.com POST
2426 51 21:07:10.049447 10.0.2.15 39.156.165.32 128 0x6a9… China HTTP 993 get.sogou.com POST
2438 52 21:07:22.749503 10.0.2.15 39.156.165.32 128 0x6a9… China HTTP 929 get.sogou.com POST
2448 53 21:07:23.563355 10.0.2.15 39.156.165.32 128 0x6aa… China HTTP 929 get.sogou.com POST
2459 54 21:07:49.269127 10.0.2.15 39.156.165.32 128 0x6aa… China HTTP 929 get.sogou.com POST
2466 55 21:07:49.573347 10.0.2.15 39.156.165.32 128 0x6aa… China HTTP 929 get.sogou.com POST
2478 56 21:07:51.920568 10.0.2.15 39.156.165.32 128 0x6ab… China HTTP 929 get.sogou.com POST
2488 57 21:08:17.108240 10.0.2.15 39.156.165.32 128 0x6ab… China HTTP 929 get.sogou.com POST
2498 58 21:08:17.417662 10.0.2.15 39.156.165.32 128 0x6ab… China HTTP 929 get.sogou.com POST
2509 59 21:08:19.975612 10.0.2.15 39.156.165.32 128 0x6ac… China HTTP 929 get.sogou.com POST
2516 60 21:08:20.282043 10.0.2.15 39.156.165.32 128 0x6ac… China HTTP 929 get.sogou.com POST
Host: get.sogou.com\r\n
Connection: close\r\n
User-Agent: sogou_ime\r\n
Content-Length: 722\r\n
\r\n
[Full request URI: http://get.sogou.com/q]
[HTTP request 1/1]
[Response in frame: 2440]
File Data: 722 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "k" = "FDhfMFPEL7JPS9 JZlHyiCIZSoxvvviHgSZEm V
Form item: "v" = "DwPfg55 5i/8dAusJnZBnC0akVXUCjFjXZMf3HR
Form item: "u" = "9FiFygGRO53cP30qfhHMA9KmQiTyOxJfogBrU3U
Form item: "g" = "rbIqDIl0140zvaVexEFtiy0I4EzIQS2GQpLu pq
Form item: "p" = "8XbsCmnQFu70Bb/oaInQAb1g5xxpm1s5a5TLlFf
00c0 4c 65 6e 67 74 68 3a 20 37 32 32 0d 0a 0d 0a 6b Length: 722····k
00d0 3d 46 44 68 66 4d 46 50 45 4c 37 4a 50 53 39 2b =FDhfMFP EL7JPS9+
00e0 4a 5a 6c 48 79 69 43 49 5a 53 6f 78 76 76 76 69 JZlHyiCI ZSoxvvvi
00f0 48 67 53 5a 45 6d 2b 56 6c 38 38 79 66 65 4b 4c HgSZEm+V l88yfeKL
0100 6c 75 4d 47 35 75 5a 67 4d 39 6b 31 54 4d 74 4b luMG5uZg M9k1TMtK
0110 44 6a 32 6e 2f 79 4a 2f 48 2f 6c 72 4c 74 39 30 Dj2n/yJ/ H/lrLt90
0120 34 74 37 61 67 51 2b 31 37 41 6d 75 7a 78 6b 4b 4t7agQ+1 7AmuzxkK
0130 4a 51 6e 6b 63 64 79 4a 70 2f 59 47 64 37 4f 46 JQnkcdyJ p/YGd7OF
0140 63 62 74 42 34 41 66 50 36 50 6e 6f 52 45 73 7a cbtB4AfP 6PnoREsz
0150 69 56 7a 68 6d 43 62 59 4e 5a 36 6d 6c 37 67 4e iVzhmCbY NZ6ml7gN
0160 38 6d 33 36 49 69 2b 4a 6b 2b 47 7a 34 47 59 79 8m36Ii+J k+Gz4GYy
0170 42 71 57 4a 47 32 61 54 4c 37 76 49 3d 26 76 3d BqWJG2aT L7vI=&v=
0180 44 77 50 66 67 35 35 2b 35 69 2f 38 64 41 75 73 DwPfg55+ 5i/8dAus
0190 4a 6e 5a 42 6e 43 30 61 6b 56 58 55 43 6a 46 6a JnZBnC0a kVXUCjFj
01a0 58 5a 4d 66 33 48 52 4c 45 55 57 4e 7a 65 59 53 XZMf3HRL EUWNzeYS
01b0 51 51 57 31 4b 76 38 44 63 33 70 43 6a 61 66 75 QQW1Kv8D c3pCjafu
Text item (text), 174 bytes
Packets: 2544 · Displayed: 23 (0.9%)
Profile: Default

Sogou IME

- K – AES key (encrypted with 1024-bit public RSA key)
- V – IV (encrypted with 1024-bit public RSA key)

RSA-bootstrapped AES…

Sogou IME

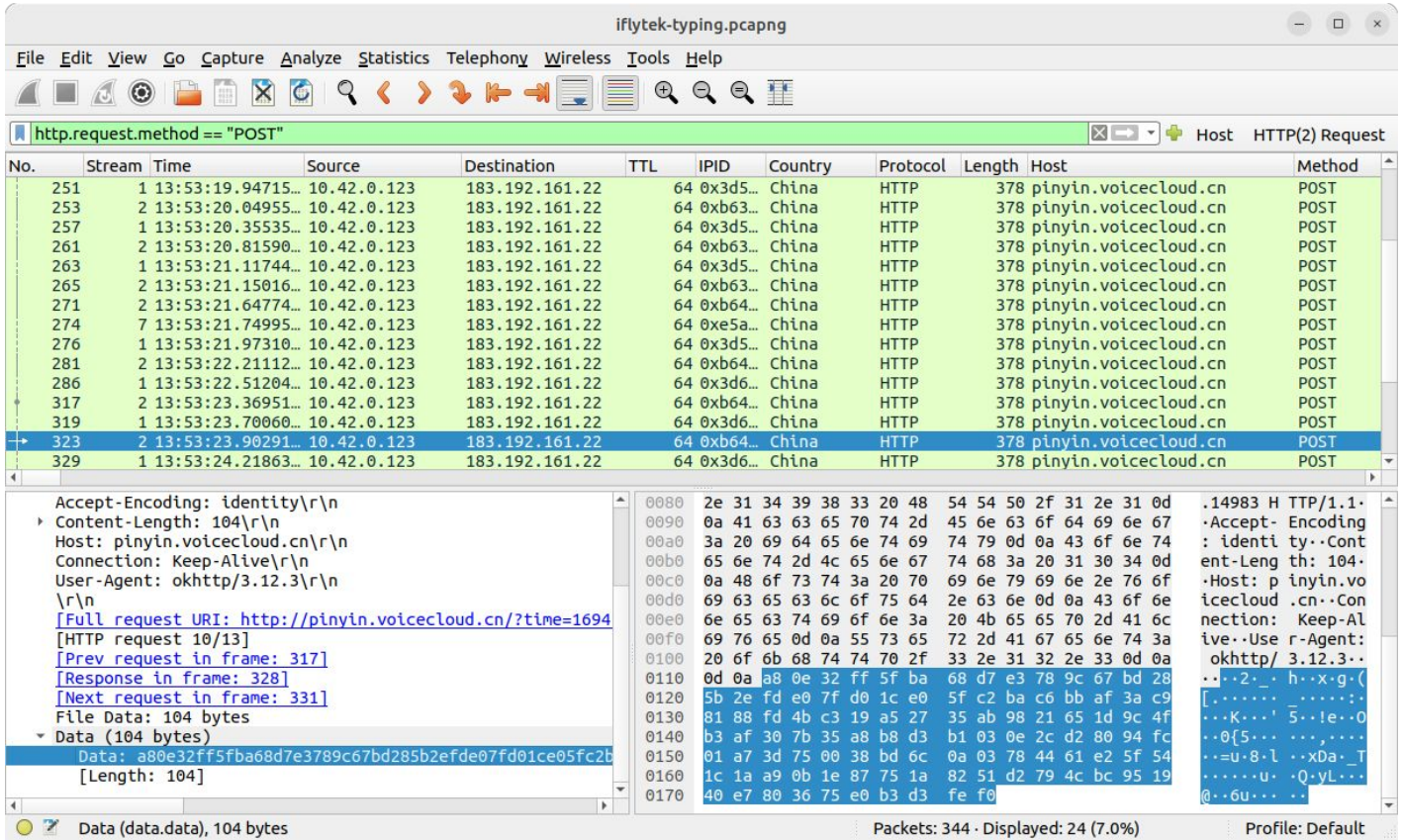
- K – AES key (encrypted with 1024-bit public RSA key)
- V – IV (encrypted with 1024-bit public RSA key)

RSA-bootstrapped AES…

Susceptible to a variant of a CBC padding oracle attack

```
1 {
  1: 1
  2 {
    1 {
      2: "1111_sogou_pinyin_guanwang_13.4e_1111"
      3: "13.4.0.7561"
      5: 3
      7: 1
      8: "13.4.0.7561"
    }
    7: "nihaohaohaohaohaohaozdaasdfffaahellocanyoureadthis"
    16: 11
    17 {
      3 {
        1: 2
        2: 1
      }
      9: 1
      10: 1
    }
    19 {
      4: "0"
    }
[...]
```

iFlytek-Android



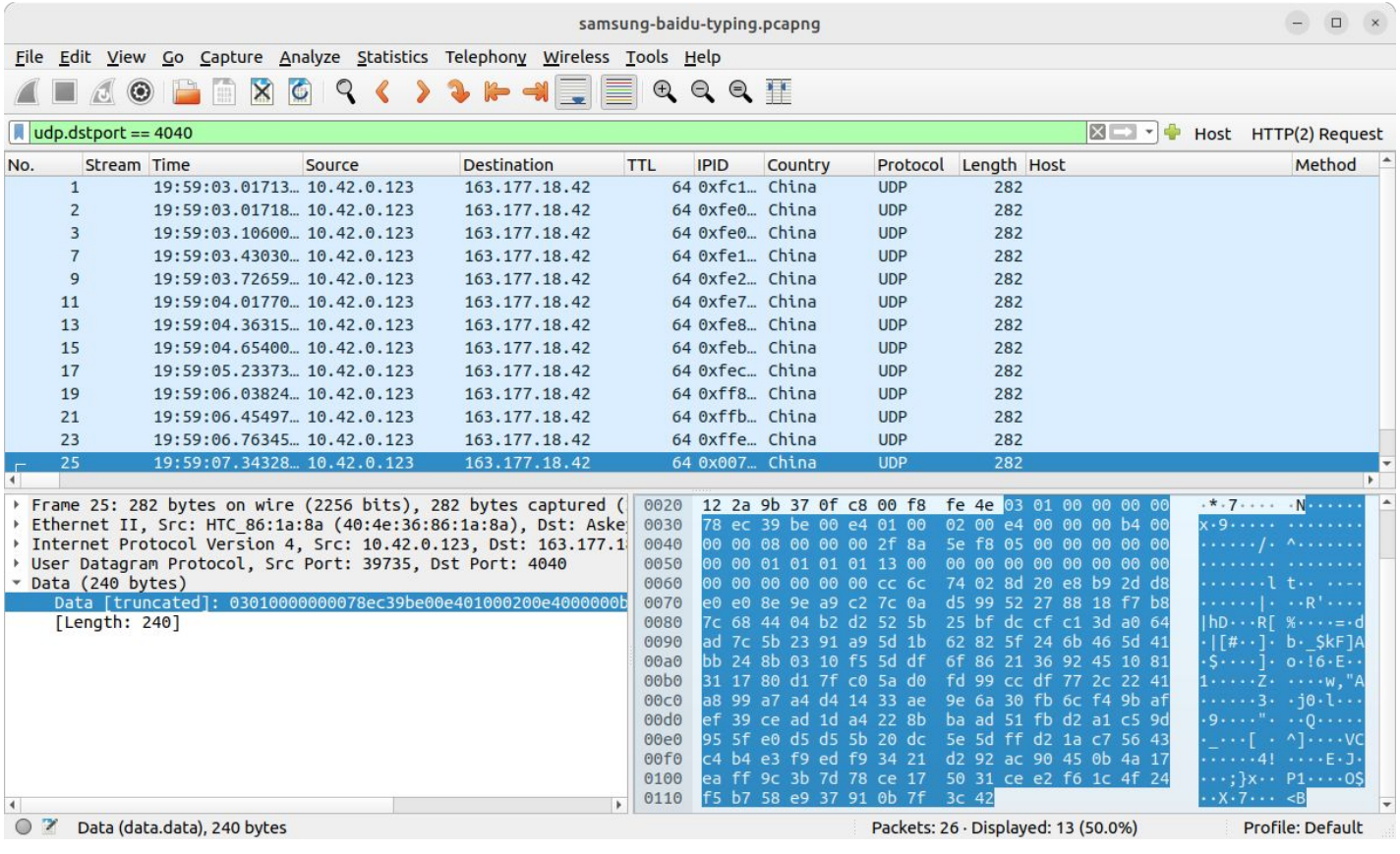
iFlytek-Android

```
__int64 generate_key_seed()
{
  return now_nanoseconds() / 1000000;
}
```

[...]

```
  if ( ptr && length )
  {
    v17 = (_DWORD)length + 8;
    output = malloc((int)length + 8);
    memset(output, 0, v17);
    sprintf((char *)key, "%08llu", (key_seed % 0x5F5E100) ^ 0x1001111);
    v19 = DES_ECB_Encrypt(ptr, (__int64)output, key, (unsigned int)length);
    free(ptr);
```

Baidu-v3-1

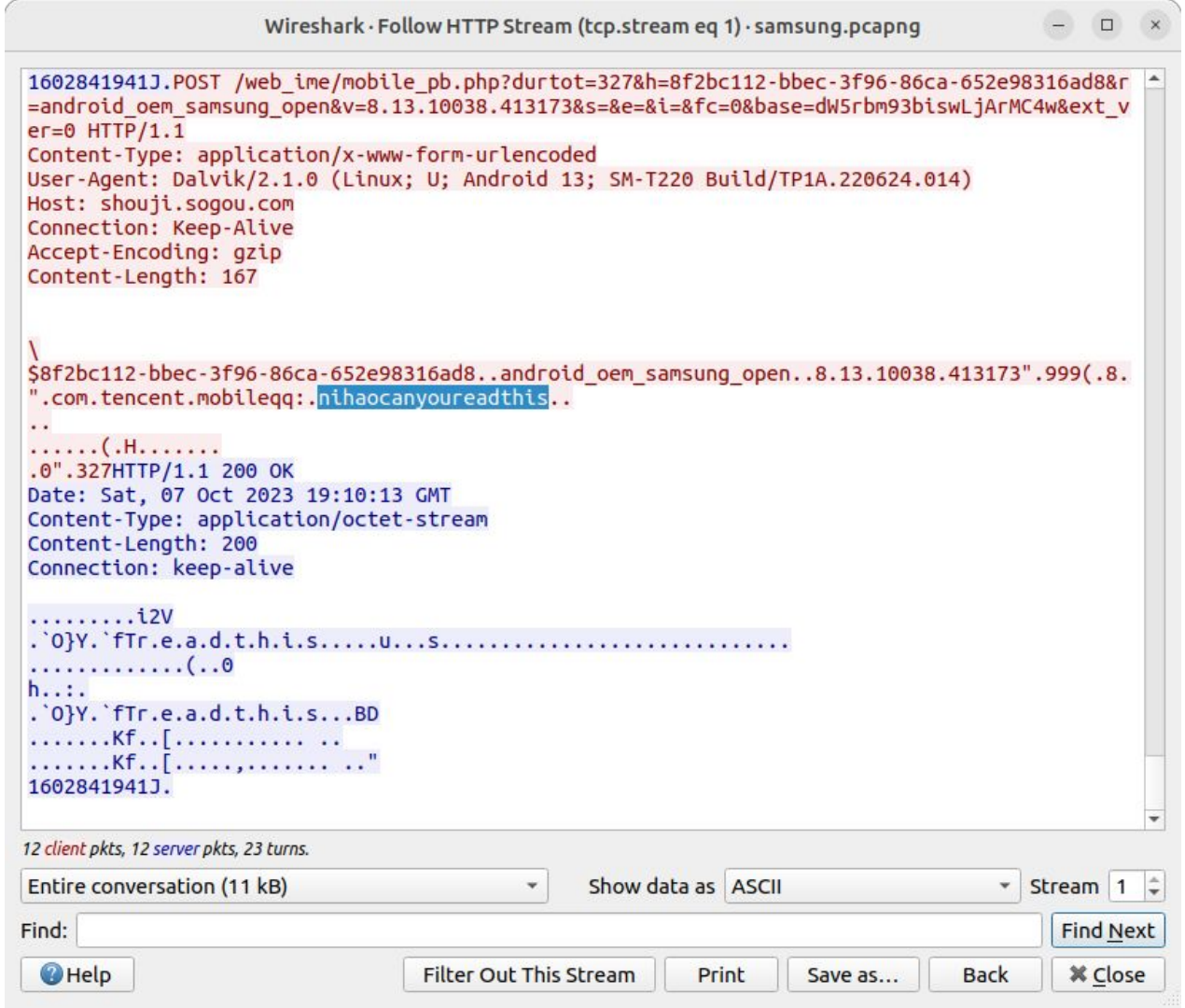

samsung-baidu-typing.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.dstport == 4040
Host
HTTP(2) Request
No. Stream Time Source Destination TTL IPID Country Protocol Length Host Method
1 19:59:03.01713… 10.42.0.123 163.177.18.42 64 0xfc1… China UDP 282
2 19:59:03.01718… 10.42.0.123 163.177.18.42 64 0xfe0… China UDP 282
3 19:59:03.10600… 10.42.0.123 163.177.18.42 64 0xfe0… China UDP 282
7 19:59:03.43030… 10.42.0.123 163.177.18.42 64 0xfe1… China UDP 282
9 19:59:03.72659… 10.42.0.123 163.177.18.42 64 0xfe2… China UDP 282
11 19:59:04.01770… 10.42.0.123 163.177.18.42 64 0xfe7… China UDP 282
13 19:59:04.36315… 10.42.0.123 163.177.18.42 64 0xfe8… China UDP 282
15 19:59:04.65400… 10.42.0.123 163.177.18.42 64 0xfeb… China UDP 282
17 19:59:05.23373… 10.42.0.123 163.177.18.42 64 0xfec… China UDP 282
19 19:59:06.03824… 10.42.0.123 163.177.18.42 64 0xff8… China UDP 282
21 19:59:06.45497… 10.42.0.123 163.177.18.42 64 0xffb… China UDP 282
23 19:59:06.76345… 10.42.0.123 163.177.18.42 64 0xffe… China UDP 282
25 19:59:07.34328… 10.42.0.123 163.177.18.42 64 0x007… China UDP 282
Frame 25: 282 bytes on wire (2256 bits), 282 bytes captured (
Ethernet II, Src: HTC_86:1a:8a (40:4e:36:86:1a:8a), Dst: Aske
Internet Protocol Version 4, Src: 10.42.0.123, Dst: 163.177.1
User Datagram Protocol, Src Port: 39735, Dst Port: 4040
Data (240 bytes)
Data [truncated]: 03010000000078ec39be00e401000200e4000000b
[Length: 240]
0020 12 2a 9b 37 0f c8 00 f8 fe 4e 03 01 00 00 00 00
0030 78 ec 39 be 00 e4 01 00 02 00 e4 00 00 00 b4 00
0040 00 00 08 00 00 00 2f 8a 5e f8 05 00 00 00 00 00
0050 00 00 01 01 01 01 13 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 cc 6c 74 02 8d 20 e8 b9 2d d8
0070 e0 e0 8e 9e a9 c2 7c 0a d5 99 52 27 88 18 f7 b8
0080 7c 68 44 04 b2 d2 52 5b 25 bf dc cf c1 3d a0 64
0090 ad 7c 5b 23 91 a9 5d 1b 62 82 5f 24 6b 46 5d 41
00a0 bb 24 8b 03 10 f5 5d df 6f 86 21 36 92 45 10 81
00b0 31 17 80 d1 7f c0 5a d0 fd 99 cc df 77 2c 22 41
00c0 a8 99 a7 a4 d4 14 33 ae 9e 6a 30 fb 6c f4 9b af
00d0 ef 39 ce ad 1d a4 22 8b ba ad 51 fb d2 a1 c5 9d
00e0 95 5f e0 d5 d5 5b 20 dc 5e 5d ff d2 1a c7 56 43
00f0 c4 b4 e3 f9 ed f9 34 21 d2 92 ac 90 45 0b 4a 17
0100 ea ff 9c 3b 7d 78 ce 17 50 31 ce e2 f6 1c 4f 24
0110 f5 b7 58 e9 37 91 0b 7f 3c 42
Data (data.data), 240 bytes
Packets: 26 · Displayed: 13 (50.0%)
Profile: Default

Baidu-v3-1

```
def derive_key():
  key = []
  x = 0
  for i in range(16):
    b = ~i ^ ((i + 11) * (x >> (i & 3)))
    key.append(b & 0xff)
    x += 1937
  return bytes(key)
```

Samsung IME

Samsung IME


Wireshark · Follow HTTP Stream (tcp.stream eq 1) · samsung.pcapng
1602841941J.POST /web_ime/mobile_pb.php?durtot=327&h=8f2bc112-bbec-3f96-86ca-652e98316ad8&r
=android_oem_samsung_open&v=8.13.10038.413173&s=&e=&i=&fc=0&base=dW5rbm93biswLjArMC4w&ext_v
er=0 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 13; SM-T220 Build/TP1A.220624.014)
Host: shouji.sogou.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 167
\
$8f2bc112-bbec-3f96-86ca-652e98316ad8..android_oem_samsung_open..8.13.10038.413173".999(.8.
".com.tencent.mobileqq:.nihaocanyoureadthis..
..
......(.H.......
.0".327HTTP/1.1 200 OK
Date: Sat, 07 Oct 2023 19:10:13 GMT
Content-Type: application/octet-stream
Content-Length: 200
Connection: keep-alive
.........i2V
.`O}Y.`fTr.e.a.d.t.h.i.s.....u...s.............................
.............(..0
h..:.
.`O}Y.`fTr.e.a.d.t.h.i.s...BD
.......Kf..[........... ..
.......Kf..[.....,....... .."
1602841941J.
12 client pkts, 12 server pkts, 23 turns.
Entire conversation (11 kB)
Show data as
ASCII
Stream
1
Find:
Find Next
Help
Filter Out This Stream
Print
Save as...
Back
Close

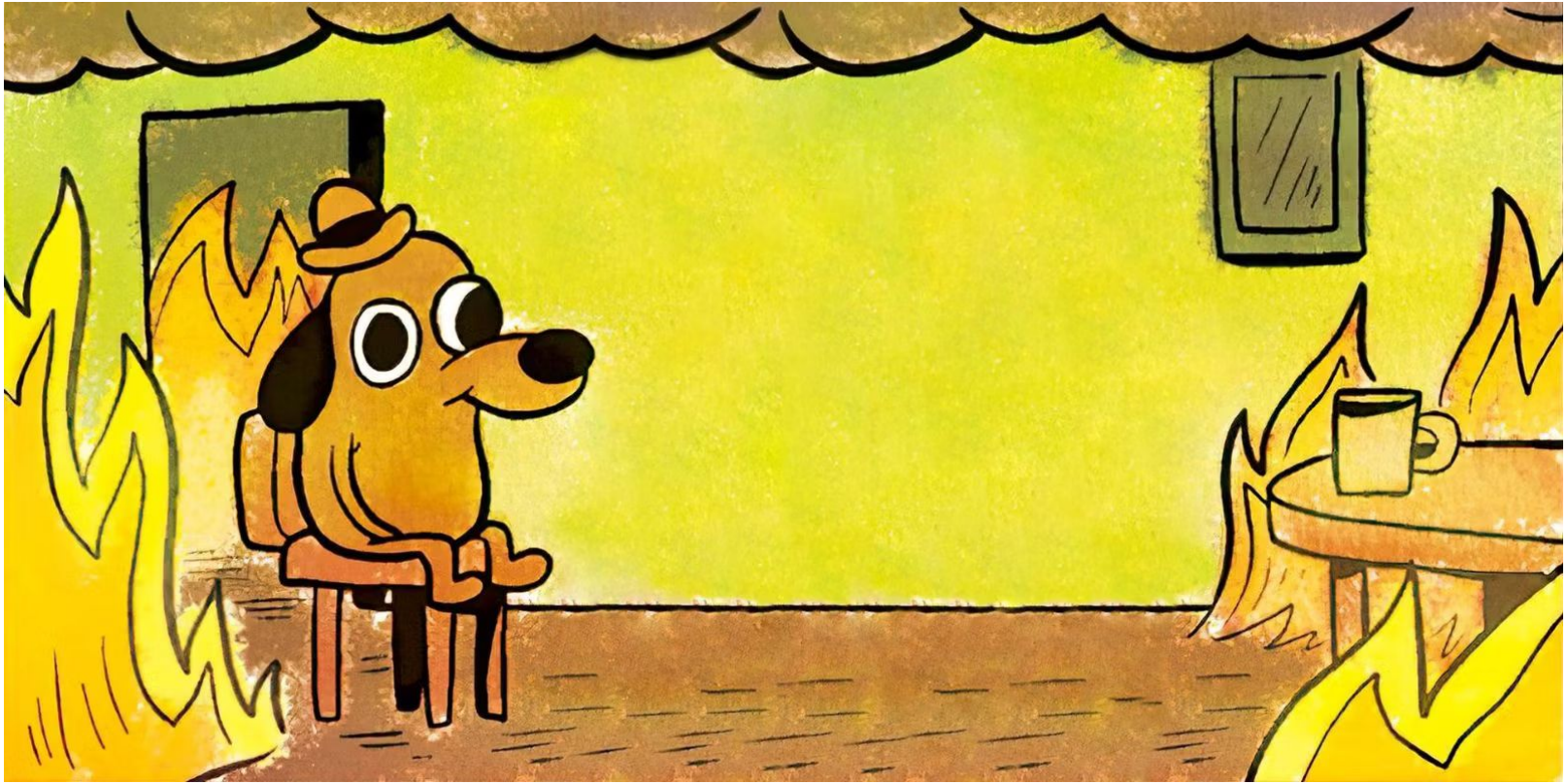
IME keyboard protocols

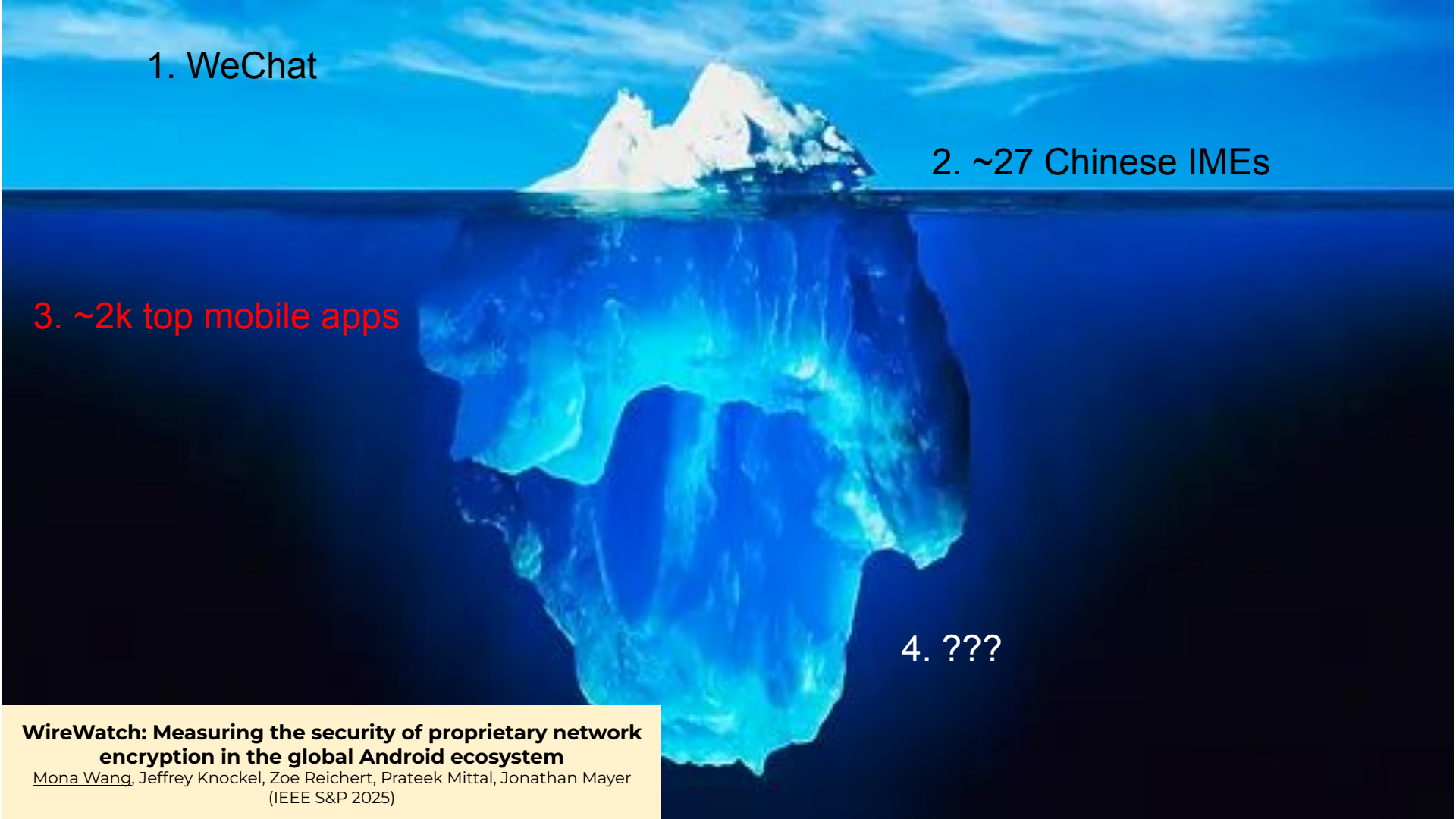
Protocol	Status	Core vulnerability	Mode	Variation
EncryptWall-And	Decryptable	CBC padding oracle	AES-CBC	Splits key into two, uses fixed IV
EncryptWall-Win	Decryptable	CBC padding oracle	AES-CBC	
BAIDUv3-1	Passively decryptable	Fixed key	AES-ECB	Additional permutations each AES round
BAIDUv3-2	Passively decryptable	Fixed key	AES-ECB	Missing AES round
BAIDUv4	Not CPA-secure	IV and key re-use	AES-BCTR	Uses home-rolled CTR mode
iFlytek-And	Passively decryptable	Key derived from timestamp	DES-ECB	

IME keyboard protocols

Vendor	Program name	File/package name	Version analyzed	Platform	Protocol used	As of 2024-04-01
Tencent	Sogou IME	SogouInput_11.20_android_sweb.apk	11.20	Android	*EncryptWall-And*	TLS
Tencent	Sogou IME	com.sogou.sogouinput	11.21	iOS	TLS	TLS
Tencent	Sogou IME	sogou_pinyin_guanwang_13.4e_1111.exe	13.4	Windows	*EncryptWall-Win*	TLS
Tencent	QQ IME	com.tencent.qqpinyin	8.6.3	Android	*EncryptWall-And*	*EncryptWall-And*
Tencent	QQ IME	QQPinyin_Setup_6.6.6304.400.exe	6.6.6304.400	Windows	*EncryptWall-Win*	*EncryptWall-Win*
Baidu	Baidu IME	com.baidu.input	11.7.19.9	Android	*BAIDUv4*	*BAIDUv4*
Baidu	Baidu IME	com.baidu.inputMethod	11.7.20	iOS	*BAIDUv4*	*BAIDUv4*
Baidu	Baidu IME	BaiduPinyinSetup_6.0.3.44.exe	6.0.3.44	Windows	*BAIDUv3-2*	*BAIDUv4*
iFlytek	iFlytek IME	com.iflytek.inputmethod	12.1.10	Android	*iFlytek-And*	TLS
iFlytek	iFlytek IME	com.iflytek.inputime	12.1.3338	iOS	TLS	TLS
iFlytek	iFlytek IME	iFlyIME_Setup_3.0.1734.exe	3.0.1734	Windows	TLS	TLS
Honor	Baidu IME Honor Version	com.baidu.input_hihonor	8.2.501.1	Android	*BAIDUv3-2*	*BAIDUv3-2*
Huawei	Celia IME	com.huawei.ohos.inputmethod	1.0.19.333	Android	TLS	TLS
Huawei	Sogou IME	com.sohu.inputmethod.sogou	11.31	Android	TLS*	TLS
OPPO	Sogou IME Custom Version	com.sohu.inputmethod.sogouoem	8.32.0322.2305171502	Android	*EncryptWall-And*	TLS
OPPO	Baidu IME Custom Version	com.baidu.input_oppo	8.5.30.503	Android	*BAIDUv3-2*	*BAIDUv4*
Samsung	Samsung Keyboard	com.samsung.android.honeyboard	5.6.10.26	Android	No encryption	TLS
Samsung	Sogou IME Samsung Version	com.sohu.inputmethod.sogou.samsung	10.32.38.202307281642	Android	TLS*	TLS
Samsung	Baidu IME	com.baidu.input	8.5.20.4	Android	*BAIDUv3-1*	*BAIDUv4*
Vivo	Jovi IME	com.vivo.ai.ime	2.6.1.2305231	Android	TLS	TLS
Vivo	Sogou IME Custom Version	com.sohu.inputmethod.sogou.vivo	10.32.13023.2305191843	Android	*EncryptWall-And*	TLS
Xiaomi	Sogou IME Xiaomi Version	com.sohu.inputmethod.sogou.xiaomi	10.6.120.480	Android	*EncryptWall-And*	TLS
Xiaomi	Baidu IME Xiaomi Version	com.baidu.input_mi	10.6.120.480	Android	*BAIDUv3-2*	*BAIDUv4*
Xiaomi	iFlytek IME Xiaomi Version	com.iflytek.inputmethod.miui	8.1.8014	Android	*iFlytek-And*	TLS

* Tested after our initial disclosure with Tencent Sogou, but may have been previously using *EncryptWall-And*.



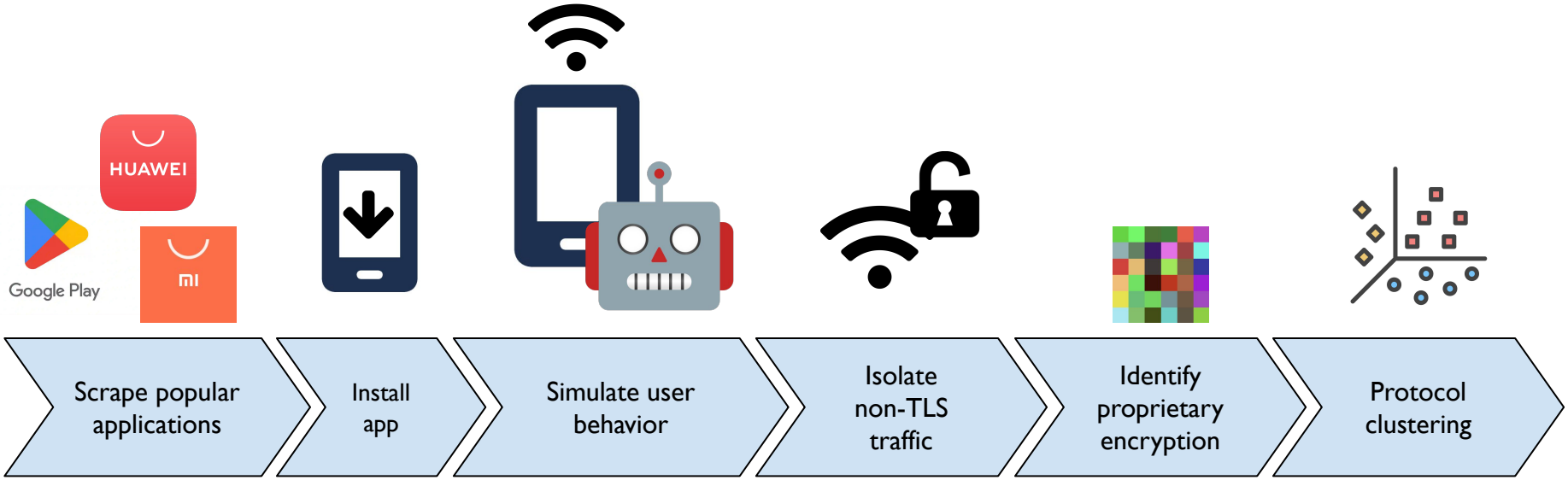


WireWatch: Measuring the security of proprietary network encryption in the global Android ecosystem

Mona Wang, Jeffrey Knockel, Zoe Reichert, Prateek Mittal, Jonathan Mayer (IEEE S&P 2025)

Research Questions

1. **How common is proprietary encryption as compared to TLS?**
2. How secure are these proprietary protocols?

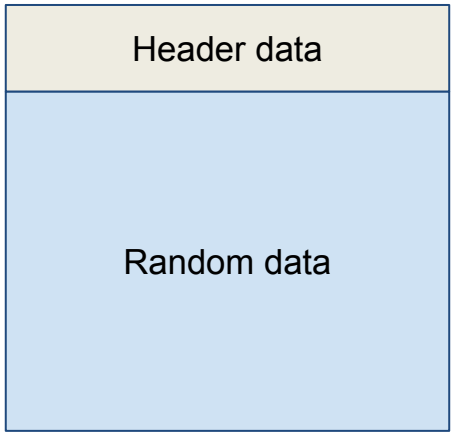

HUAWEI
Google Play
Scrape popular applications
Install app
Simulate user behavior
Isolate non-TLS traffic
Identify proprietary encryption
Protocol clustering

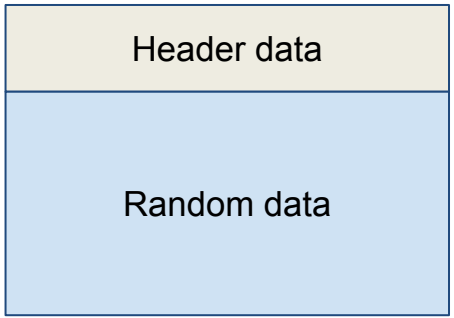
How to identify proprietary encryption?

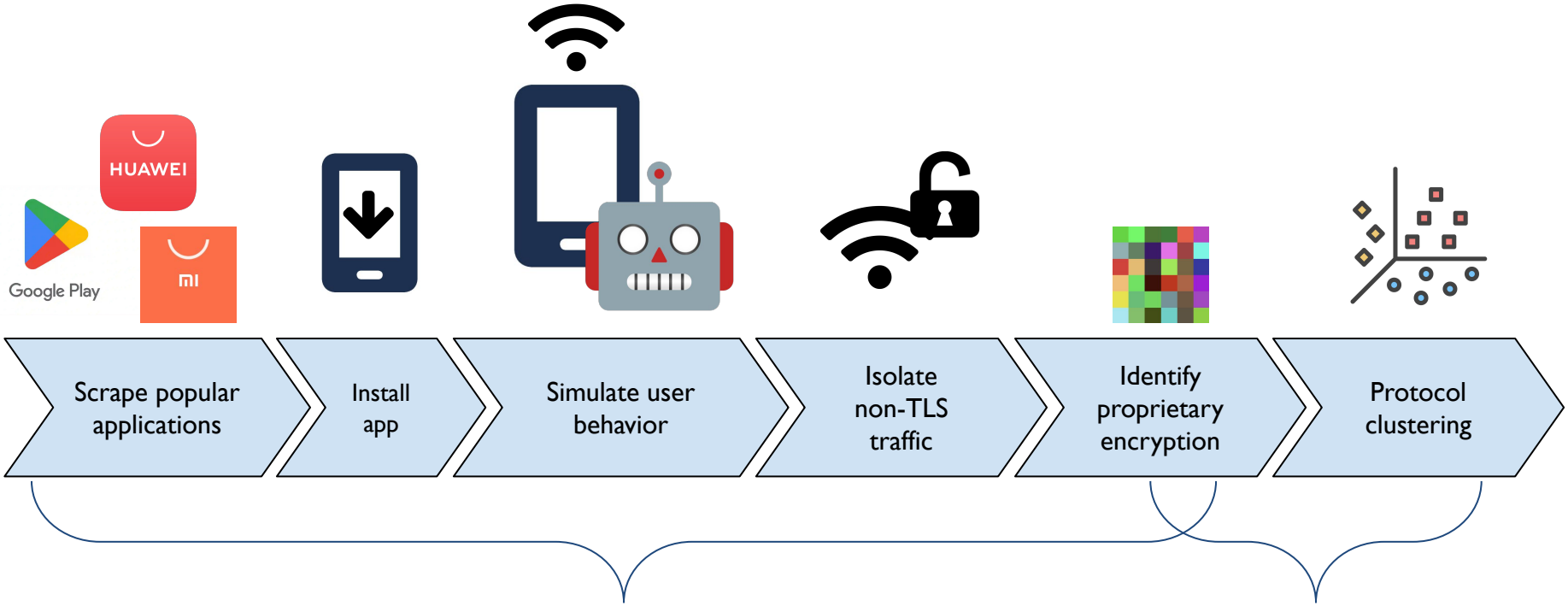
1. It doesn’t conform with existing standardized encryption (e.g. TLS, QUIC-TLS)
2. It looks like **encryption** → (e.g. looks like **random data**)

How to cluster “similar” encryption?

- Encrypted data may differ, but…
  - *Header data* might be similar, and contain similar constants, features, etc.
  - *Encoding* of data consistent (e.g. Base64? Hex? Raw data?)






HUAWEI
Google Play
Scrape popular applications
Install app
Simulate user behavior
Isolate non-TLS traffic
Identify proprietary encryption
Protocol clustering
95.6% accuracy in identifying proprietary encryption
94.7% accuracy in clustering protocols

WireWatch results

12.9% of top 1k apps sent plaintext traffic.

3.5% of top 1k apps used proprietary cryptography.

WireWatch results

12.9% of top 1k apps sent plaintext traffic.

3.5% of top 1k apps used proprietary cryptography.

65.4% of top 1k apps sent plaintext traffic.

47.6% of top 1k used proprietary cryptography!

WireWatch results

Independently confirms concurrent work! [1]

App Store	# apps analyzed	Plaintext traffic	Proprietary crypto	TLS not validated
Google Play	882	12.9%	3.5%	2.2%
Mi Store	817	65.4%	47.6%	49.1%
Total	1,699	38.1%	24.2%	24.1%

TABLE 1. SUMMARY OF RESULTS FROM OUR MEASUREMENT PIPELINE.

[1] **Racing for TLS Certificate Validation: A Hijacker's Guide to the Android TLS Galaxy**
Sajjad Pourali, Xiufen Yu, Lianying Zhao, Mohammad Mannan, Amr Youssef
Usenix Security 2024

Research Questions

1. How common is proprietary encryption?
2. ***How secure are these protocols?***

WireWatch results

- 177 protocol clusters
  - 94 used by only one application.
  - 83 were used across many different applications!
- Sorted by “popularity”, e.g. sum of downloads of all apps using each protocol
- Reverse-engineered top 18 protocols
  - Some protocols belonged to the same cryptosystem, so we grouped them into “protocol families”
  - **9 protocol families in total**

We manually analyzed the 9 most popular proprietary protocol families...

? contained severe vulnerabilities where we broke the encryption!

We manually analyzed the 9 most popular proprietary protocol families...

8 contained severe vulnerabilities where we broke the encryption!

We manually analyzed the **9** most popular proprietary protocol families...

8 contained severe vulnerabilities where we broke the encryption!

130 billion cumulative downloads of applications affected by the issues we found.

WireWatch results

Protocol family	# apps	Cumulative downloads	Most downloaded	MAU	Decryptable	Fixed?	Decrypted request contents
Kuaishou SDK	76	35.10B	Kuaishou	692 mill	YES	YES	Device metadata
MobSDK	82	30.30B	RedNote	312 mill	YES	YES	Device metadata
Alibaba mPaaS	15	25.43B	Taobao	921 mill	YES	No :(	Browsing data
DNSPod	11	18.10B	Pinduoduo	695 mill	YES	Wontfix	DNS requests
WUP	7	17.62B	QQ Browser	571 mill	YES	YES	Browsing data
iQIYI	3	11.28B	iQIYI	429 mill	YES	YES	Network metadata
iShumei	37	10.34B	RedNote	312 mill	YES	Mostly	Security config*
MMTLS	1	9.23B	WeChat	1.3 bill	NO	-	-
Beizi SDK	38	9.02B	Baidu Netdisk	107 mill	YES	No :(	Device metadata

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

Leaked browsing data

500 million+ active users

400 million+ active users

Data leaked from insecure cryptography includes **pages of visited URLs!**

Case study: Alibaba mPaaS (UC browser, Taobao)

- Heavily obfuscated!
- `libsgmain.so` is secretly a JAR file???

```
public byte[] staticBinarySafeEncrypt(int i, String str, byte[] bArr, String str2) throws Se
    if (str == null || str.length() <= 0 || i < 3 || i >= 19 || bArr == null || bArr.length
        throw new SecException("", 301);
    }
    return m94(1, i, 1, str, bArr, str2);
}

public byte[] staticBinarySafeEncryptNoB64(int i, String str, byte[] bArr, String str2) thro
    if (str == null || str.length() <= 0 || i < 3 || i >= 19 || bArr == null || bArr.length
        throw new SecException("", 301);
    }
    return m96(i, str, bArr, str2);
}
```

```
package com.taobao.wireless.security.adapter;

/* loaded from: classes.dex */
public class JNICLibrary {
    public static native Object doCommandNative(int i, Object... objArr);
}
```

Calls into `libsgmain<version>.so`

Case study: Alibaba mPaaS (UC browser, Taobao)

`libsgmain<version>.so` (main encryption library) is obfuscated :(

Case study: Alibaba mPaaS (UC browser, Taobao)

`libsgmain<version>.so` (main encryption library) is obfuscated :(

- String literals and constants were stored encrypted in the data section
  - had to dump `.bss` section at runtime

Case study: Alibaba mPaaS (UC browser, Taobao)

`libsgmain<version>.so` (main encryption library) is obfuscated :(

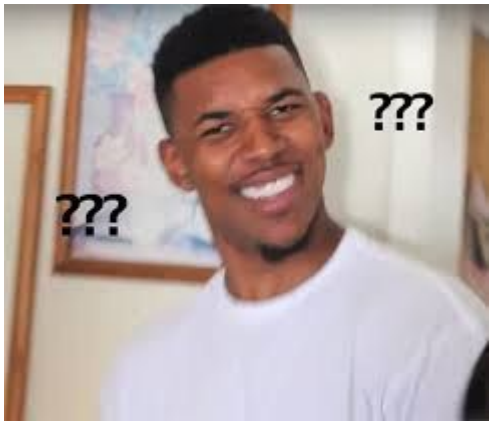
- String literals and constants were stored encrypted in the data section
  - had to dump `.bss` section at runtime
- Artificially introduces jumps to the result of a bunch of arithmetic, e.g. the first jump in `JNI_OnLoad`:

```
mov    w9,#0x2b              ; w9 = 0x2b
str    w9,[sp, #local_3c]    ; store w9 at offset #local_3c on stack
adr    x5, 0x128eec          ; x5 = 0x128eec
ldrsw  x27, [0x128f08]       ; x27 = 0xFFFFFFFA (value at 0x128f08)
mvn    x27, x27              ; x27 = bitwise NOT of x27 (= 0x5)
ldrsw  x6, [x10]=>local_3c   ; x6 = 0x2b (read from stack at #local_3c)
add    x27, x27, x6          ; x27 = 0x5 + 0x2b = 0x30
add    x5,x5,x27             ; x5 += x27
br     x5                    ; jump to address [0x128eec + ^0xFFFFFFFA + 0x2b]
```

  - Patched indirect branches with corresponding direct, PC-relative branches

Case study: Alibaba mPaaS (UC browser, Taobao)

Loads **static encryption keys** from a file called `res/drawable/yw_1222.jpg`



Case study: Alibaba mPaaS (UC browser, Taobao)

Loads **static encryption keys** from a file called `res/drawable/yw_1222.jpg`

- in the JPEG file contents, keys are stored encrypted with public data (the APK's public RSA signing key)
- googled "yw_1222.jpg" and found this public documentation:

> **Generate the encryption image (Apsara Stack configuration file)**
>
> When some components of the mPaaS Plug-in get access to the network, the contents must be encrypted to ensure security.
>
> - The image named as `yw_1222.jpg` provides a secret key for encryption and decryption. The components of mPaaS Plugin automatically use this image for encryption and decryption.

Case study: WUP (QQ, Sogou browsers)

Found two critical vulnerabilities in this protocol.

1. A standard AES-CBC padding oracle. Can retrieve encrypted plaintexts.
2. **Vulnerability in RSA construction. Can retrieve encryption key for 18.3% of keys.**

Either method can retrieve the underlying encrypted data.

Case study: WUP (QQ, Sogou browsers)

Used RSA without OAEP padding to perform key exchange, e.g.:

```
C = RSA_encrypt(pubkey, m)
```

Where `C` is sent to servers, and `m` is used as an AES key for future encryption.

- If the server decrypted `C`, and `m` was over 2^{128}, it would return a custom error:

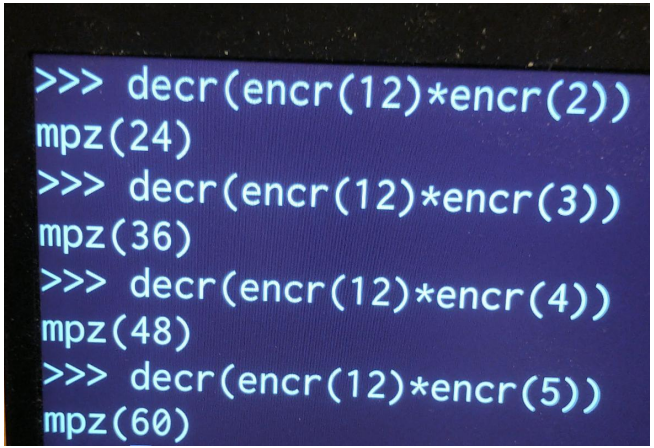
  ```
  error num:-3
  ```

- If the server decrypted `C`, and `m` was under 2^{128}, and AES decryption with `m` failed, it gave a different error:

  ```
  error num:-2
  ```

Case study: WUP (QQ, Sogou browsers)

Recall that RSA is **malleable** under multiplication:

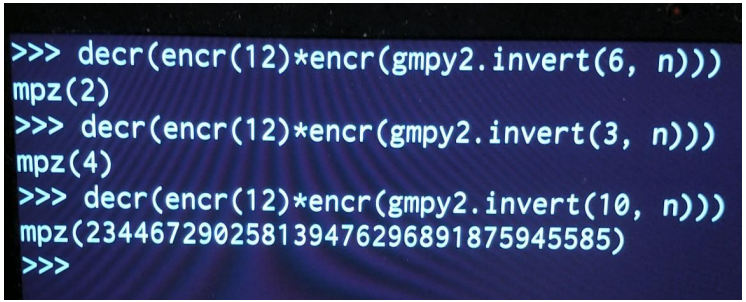


$$c = m^e \pmod N$$

$$k^e c \pmod N = k^e m^e \pmod N = (km)^e \pmod N$$

Case study: WUP (QQ, Sogou browsers)

This also applies to **division** (e.g. multiplying by the multiplicative inverse under modulus N):

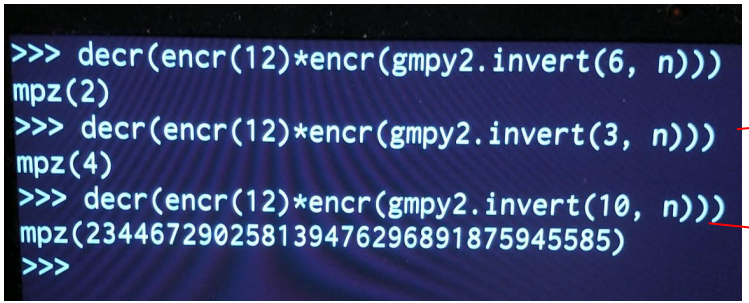


$$\text{If } m = 0 \,(mod\, k), \text{ then } k^{-1} m \,(mod\, N) = m/k < 2^{128}$$

$$\text{If } m \neq 0 \,(mod\, k), \text{ then } k^{-1} m \,(mod\, N) \geq 2^{128}$$

Case study: WUP (QQ, Sogou browsers)

This also applies to **division** (e.g. multiplying by the multiplicative inverse under modulus N):



$$\text{If } m = 0 \,(mod\, k), \text{ then } k^{-1} m \,(mod\, N) = m/k < 2^{128}$$

$$\text{If } m \neq 0 \,(mod\, k), \text{ then } k^{-1} m \,(mod\, N) \geq 2^{128}$$

We have an **oracle** for whether the AES key ***m*** is divisible by any factor ***k***!

Case study: WUP (QQ, Sogou browsers)

1. Use division oracle to identify all factors of ***m*** under 2^{24} (~1 million queries)
   a. Let ***F*** = product of all factors of m under 2^{24}
   b. We need to find remaining factor product ***R*** such that ***m = F * R***

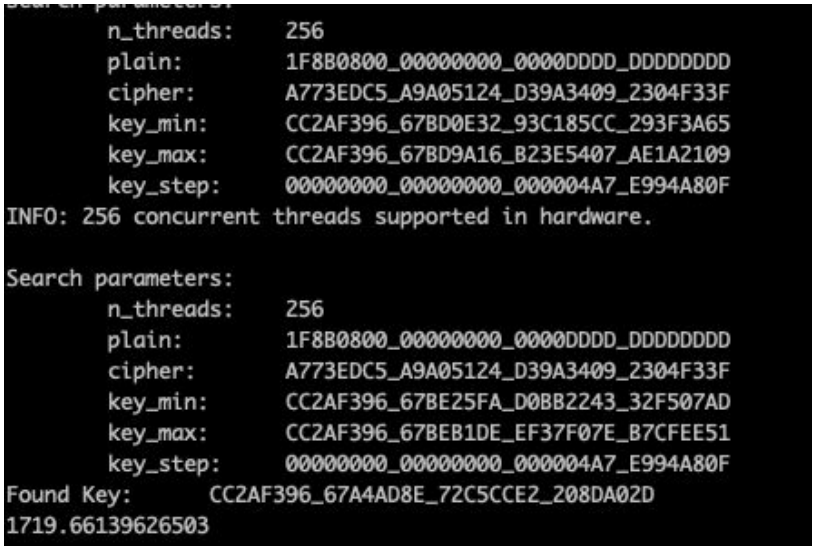
2. Narrow search space for ***R***:
   a. Develop a similar oracle to tell whether $\boldsymbol{kR >= 2^{128}}$
   b. Binary-search ***k*** such that: $\boldsymbol{kR < 2^{128}}$ and $\boldsymbol{(k+1)R >= 2^{128}}$ (max 128 queries)

3. With the bound $[2^{128}/(k+1), 2^{128}/k]$ for ***R***, can we brute-force it?

Case study: WUP (QQ, Sogou browsers)

- With the bound $[2^{128}/(k+1), 2^{128}/k]$ for ***R***, can we brute-force it?
  - Yes!

- From running attack simulations for 100k randomly chosen keys ***m***, we are able to find ***m*** for about 1 in 5 (~18.3%) keys!
  - E.g. the search space for ***R*** is under 2^{48} (brute-forceable within a day) for 18.3% of keys.

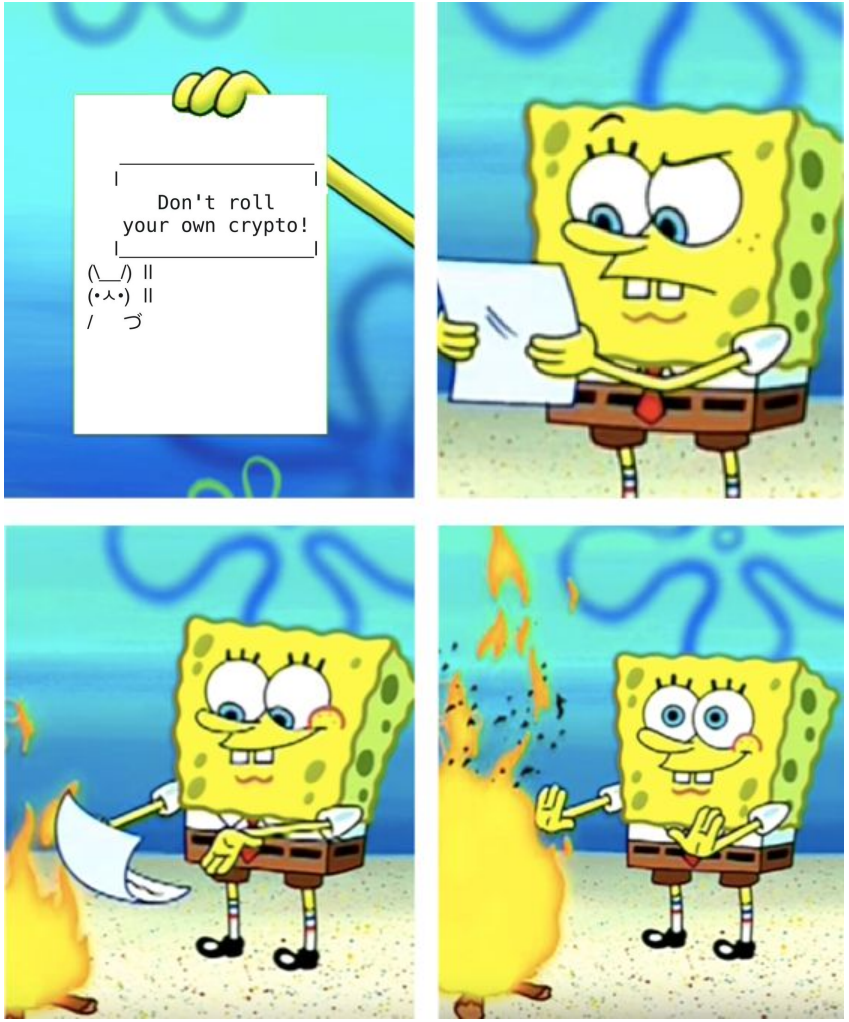


Case study: WUP (QQ, Sogou browsers)

- What kind of data is protected with this encryption protocol?
  - WUP is used by browsers.
    - The full URL of each page visited in the browser
    - Network metadata (WiFi access points, name of connected network)
    - Device metadata (screen dimensions in pixels, OS data)

Overall…

- Basic, but critical, flaws, including:
  - Static key use with symmetric cryptography
    - Relying on code obfuscation to hide secrets
  - Key seeded from timestamp
  - Using RSA without OAEP
  - Susceptible to AES-CBC padding oracle
  - Not validating TLS certificates
  - Running untrusted code
- None of the schemes tried to provide cryptographic integrity or authenticity, except for MMTLS


Don't roll
your own crypto!
(__/) ||
(•ㅅ•) ||
/ づ

This is still a systemic problem

Xiaomi Store (2024)

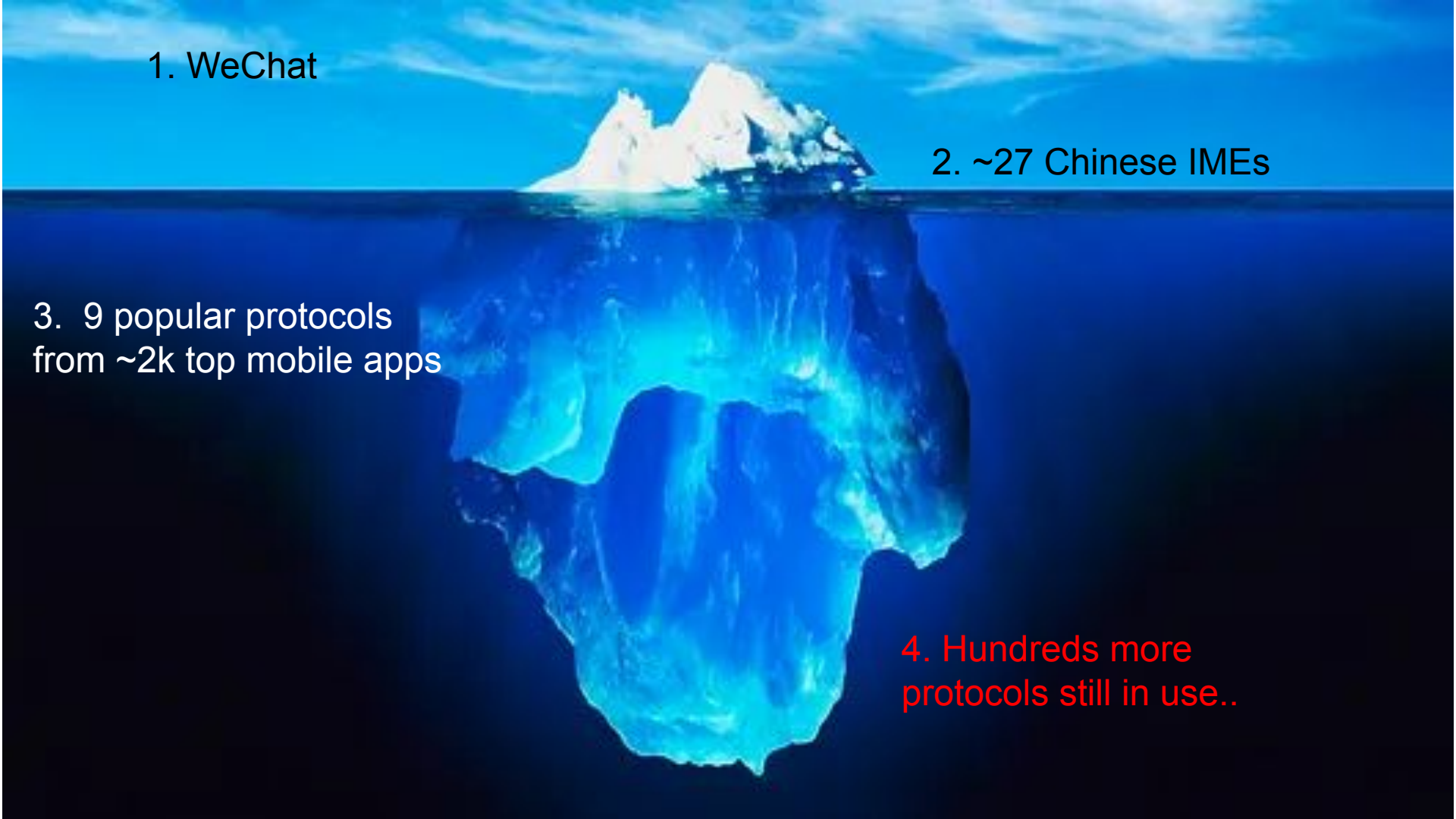
65.4% of top 1k apps sent plaintext traffic.

47.6% of top 1k used proprietary cryptography!

Xiaomi Store (2025)

56.2% of top 1k apps sent plaintext traffic.

38.8% of top 1k used proprietary cryptography!


1. WeChat
2. ~27 Chinese IMEs
3. 9 popular protocols
from ~2k top mobile apps
4. Hundreds more
protocols still in use..

Discussion

- Are these backdoors?

Discussion

- Are these backdoors?
  - **No**

Discussion

- Are these backdoors?
  - **No**

- No selective access
- Companies that did respond, pushed fixes very quickly
- CNCERT/CC is actively interested in improving transport security
- Data is already within China jurisdiction
- Similar vulnerabilities were actively exploited by Five Eyes to surveil Chinese users abroad

Where is X-KEYSCORE?

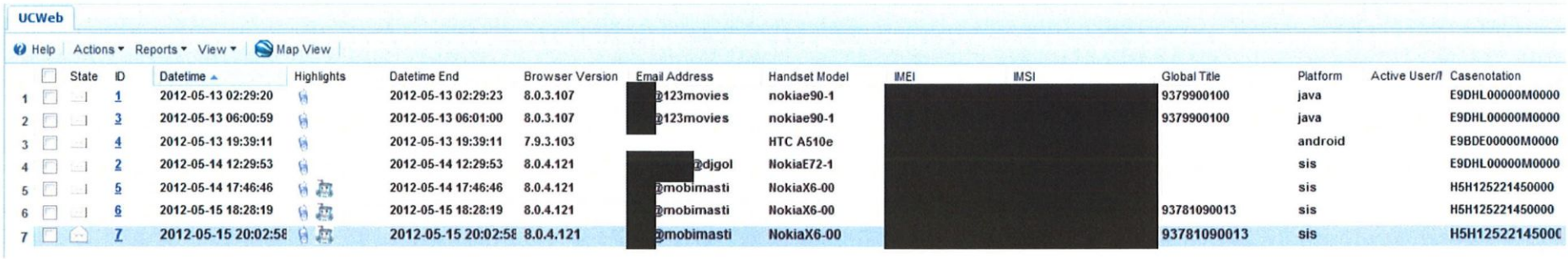
Approximately 150 sites

Over 700 servers

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

Why is this happening?

Why is this happening?

- Many of these applications became massively popular around the early 2010s– before TLS was de-facto standard
  - \+ Inertia
- Anti-scraping/competition (mistaking obfuscation for security)
- 996 → 007 working culture

- ??????

How do we fix it?

1. Find the problems
   - Security researchers should pay more attention to these **massively popular** but understudied apps
   - Any researcher that looked at this traffic in Wireshark would know there is a problem

How do we fix it?

2. Report the problems

- Many did switch to TLS when we reported severe vulns, some did not
- We need to better engage with these companies and put pressure on them to design better software

How do we fix it?

3. Prevent future problems?

- Can platforms, app store enforcement, etc. impose restrictions on the nature of app's network access?
- "Don't roll your own crypto" – how do we spread this message?

Acknowledgements!

Collaborators: Jeffrey Knockel, Zoë Reichert, Pellaeon Lin, Will Greenberg

Blog post review and disclosure coordination: Adam Senft, Ron Deibert

Reviewing cryptographic attacks: Keegan Ryan, Seth Schoen

Ph.D. advisors: Jonathan Mayer, Prateek Mittal

Learn more!

- WeChat MMTLS: https://citizenlab.ca/2024/10/should-we-chat (PETS 25)
- IMEs: https://citizenlab.ca/2024/04/vulnerabilities-across-keyboard (CCS 24)
- WireWatch: https://m0na.net/papers/wirewatch.pdf (S&P 25, blog post incoming)
- RedNote: https://citizenlab.ca/2025/02/network-security-issues-in-rednote/
  - RedNote used some of these protocols, **and also** retrieves videos/image over plaintext HTTP

Questions?

monaw@berkeley.edu