Floating Car Data from Smartphones: What Google And Waze Know About You and How Hackers Can Control Traffic

Black Hat | Europe

March 12-15, 2013

Tobias Jeske
Institute for Security in Distributed Applications
TU Hamburg-Harburg
tobias.jeske@tu-harburg.de

TUHH
Technische Universität Hamburg-Harburg

Agenda

- **Introduction**
- **Protocol Analysis**
  Google Protocol
  Waze Protocol
- **Evaluation**
  Privacy
  Authenticity / Attack
- **Solution**
  Requirements
  Zero-Knowledge Protocols
  Protocol
  Discussion
- **Conclusion**

Agenda

- **Introduction**
- **Protocol Analysis**
  Google Protocol
  Waze Protocol
- **Evaluation**
  Privacy
  Authenticity / Attack
- **Solution**
  Requirements
  Zero-Knowledge Protocols
  Protocol
  Discussion
- **Conclusion**

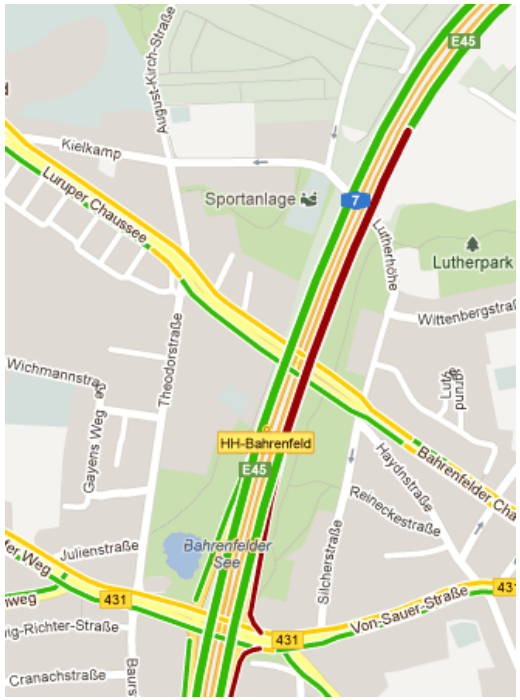
TMC

- Navigation devices receive traffic reports on the Traffic Message Channel (TMC)
- Sources → the police, traffic cameras, inductive loops, volunteers...

- Radio stations transmit TMC data in the non-audible range of the FM frequency band
- TMC is widespread, however,...
  - traffic reports are often out of date
  - low transfer rate
- In 2007 Andrea Barisani and Daniele Bianco showed how counterfeited TMC messages can be sent to navigation devices [3]
  - TMC data is not transmitted encrypted but...
  - it is necessary that the navigation devices are in the range of the transmitter

Google Live Traffic

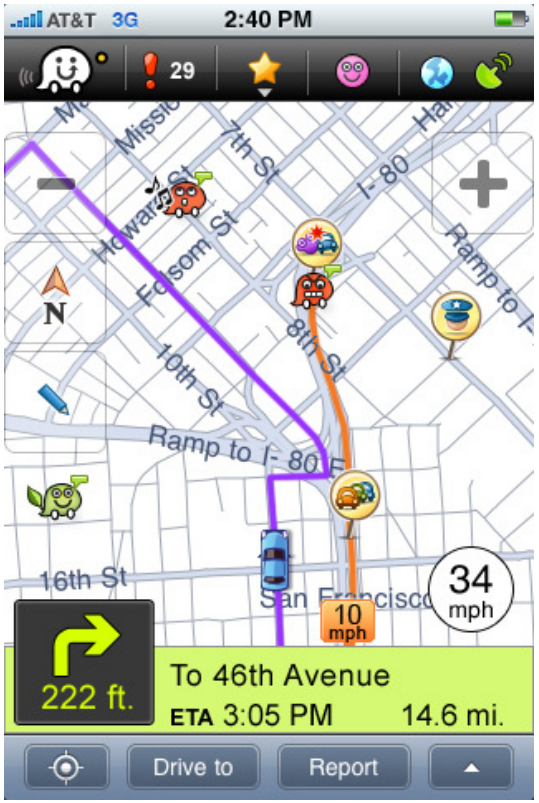
- In 2007, Google added Google Live Traffic to Google Maps [7]
- Google uses position data of smartphones with Android operating system [2]
  - Floating Car Data (FCD)
  - Real-time traffic information
- Since 2011, Google Live Traffic has been used to optimize route calculation in Google Navigation [9]
  $\rightarrow$ traffic jams avoidance!



Waze

- Free GPS application, which uses FCD of smartphones in order to generate traffic information in real-time [12]
- The application can be installed on Android, IOS, Windows Mobile, Symbian and BlackBerry
- In the iTunes Store top 20 of free apps

  $\rightarrow$ 36 million users end of 2012
- Users can add new roads, report accidents, traffic jams and speed traps directly via the Waze-App



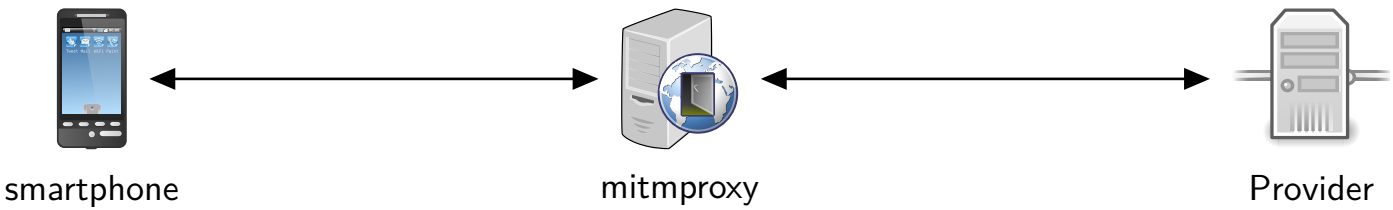
Agenda

- Introduction
- **Protocol Analysis**
  Google Protocol
  Waze Protocol
- Evaluation
  Privacy
  Authenticity / Attack
- Solution
  Requirements
  Zero-Knowledge Protocols
  Protocol
  Discussion
- Conclusion

Google Protocol

- Smartphone data is transmitted to `https://www.google.com/loc/m/api`

- Man-in-the-middle attack using mitmproxy [1]
  - Google Nexus S smartphone with Android 4
  - Install root certificate from mitmproxy
  - Configure a system-wide proxy server
  - Analyze packets and source code (only available for older Android versions, Apache License version 2.0)



mitmproxy

```
2012-04-28 16:01:21  POST https://www.google.com/loc/m/api
2012-04-28 16:01:21    <- 200 application/binary, 78B
Request                                    Response
Keep-Alive:          300
Connection:          Keep-Alive
Host:                www.google.com
User-Agent:          GoogleMobile/1.0 (crespo IML74K); gzip
Content-Type:        application/binary
Content-Length:      297


0000000000 00 02 00 00 1f 6c 6f 63 61 74 69 6f 6e 2c 31 31   .....location,11
0000000010 31 36 2c 61 6e 64 72 6f 69 64 2c 67 6d 6d 2c 65   16,android,gmm,e
0000000020 6e 5f 55 53 e8 10 d7 4c 2e 38 c7 31 00 01 67 00   n_US...L.8.1..g.
0000000030 00 00 f3 00 01 01 00 06 00 08 67 3a 6c 6f 63 2f   ..........g:loc/
0000000040 75 6c 00 00 00 04 50 4f 53 54 6d 72 00 00 00 04   ul....POSTmr....
0000000050 52 4f 4f 54 00 00 00 00 cb 00 01 67 1f 8b 08 00   ROOT.......g....
0000000060 00 00 00 00 00 00 e3 8a e7 62 31 34 34 34 13 72   .........b1444.r
0000000070 48 cc 4b 29 ca cf 4c d1 4f cf cf 4f cf 49 d5 2f   H.K)..L.O..O.I./
0000000080 ce cf 2a d5 4f 2e 4a 2d 2e c8 b7 32 d1 33 d0 33   ..*.O.J-...2.3.3
0000000090 d1 f7 f4 f5 35 37 73 d1 37 b2 b4 b4 30 b1 b4 2a   ....57s.7...0..*
00000000a0 2d 4e 2d d2 2f 4a cd 49 4d 2c 4e d5 cd 4e ad 2c   -N-./J.IM,N..N.,
00000000b0 d6 62 4d 49 8d 77 71 35 e2 e3 60 15 62 2d ce cc   .bMI.wq5..`.b-..
00000000c0 ad cc 57 60 d6 68 63 52 8a e0 92 e6 12 e2 38 bd   ..W`.hcR......8.
00000000d0 83 49 a0 ff db 7b 3b 09 66 85 36 26 0d f6 00 56   .I...{;.f.6&..V
00000000e0 81 ed 7f 9f 9e 7c af 26 64 c2 31 ed ff 33 10 43   .....|.&d.1..3.C
00000000f0 9b 8b 41 48 a4 30 2d 55 21 20 bf a8 24 31 29 27   ..AH.0-U! ..$1)'
0000000100 55 c1 23 bf 04 68 7b 89 c2 f2 ff 50 c0 e8 b0 e9   U.#..h{....P....
0000000110 e8 c4 73 6f 5b b4 66 b1 31 71 30 59 88 03 00 aa   ..so[.f.1q0Y....
0000000120 51 3f e0 bd 00 00 00 00 00                       Q?.......

[1]                                                  [m:hex][*:8080]
```

MASF header is marked red

compressed protobuf payload is marked blue

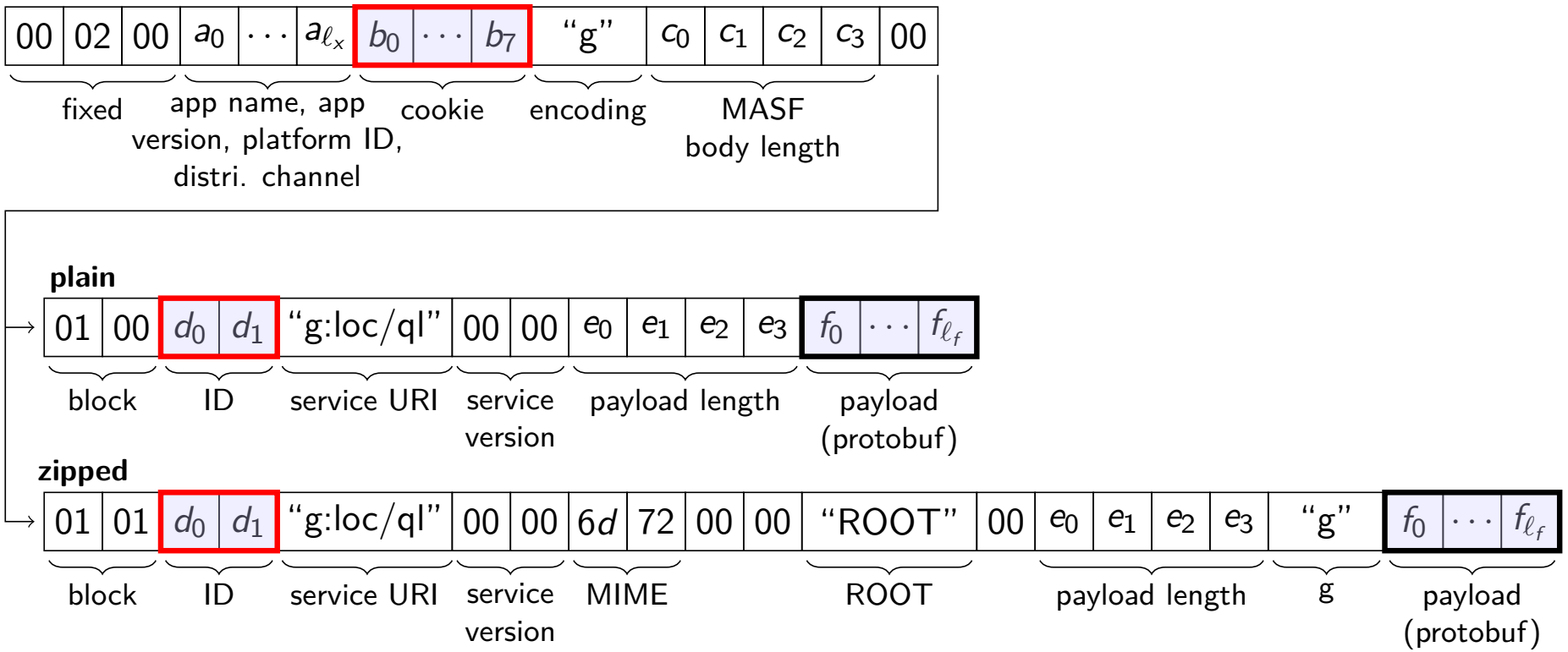
Google Protocol

- The protocol is a request/response protocol and based on MASF (Mobile Application Sensing Framework)

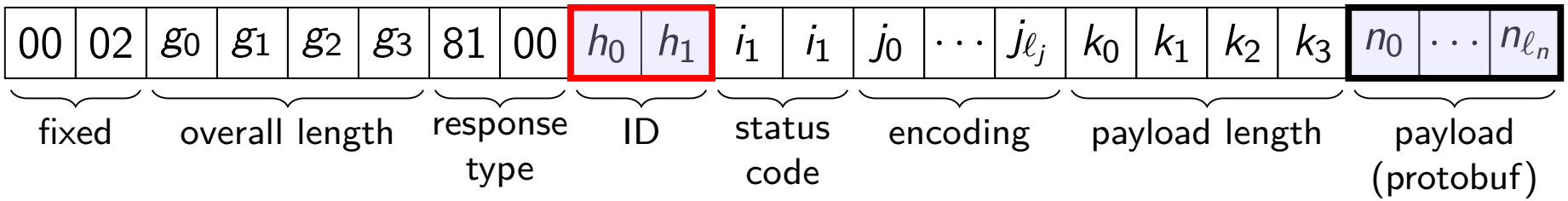
The Google Protocol in a nutshell:

- Smartphone sends Google status information of the GPS, wireless and mobile unit
  - → data amount depends on the units activated and the system configuration
- Google responds with the (approximate) location of the phone
  - → speeds up later location determination

MASF Request Message



MASF Response Message



Protocol Buffers Payload

- Protocol Buffers [8] to encode payload
  - Data format developed by Google to serialize data structures
  - Binary format → high processing speed and data density
  - Open source since July 2008
- **Request:**
  - Request element contains zero, one or more profiles
    → `Cellular`, `Wifi` and `Location` (GPS)
  - Platform profile
    Platform →
    `android/google/soju/crespo:4.0.4/IMM76D/`
    `299849:user/release-keys`
    Platform Key → **pseudonym to track smartphone**
- **Response:**
  - Current position (if possible)
  - Location of individual Wi-Fi AP and radio towers
  - New Platform Key (optional)

Protocol Buffers Payload (Request)

```
message LatLngMsg
    required fixed32 Lat = 1;
    required fixed32 Lng = 2;

message LocationProfileMsg
    optional LatLngMsg LatLng = 1;
    optional int32 Accuracy = 3;
    optional int64 Timestamp = 6;
    optional int32 LocType = 8;
    optional int32 Altitude = 10;
    optional fixed32 Speed = 16;
    optional bool PluggedIn = 17;

message CellMsg
    required int32 Lac = 1;
    required int32 Cellid = 2;
    optional int32 Mnc = 3;
    optional int32 Mcc = 4;
    optional int32 Rssi = 5;
    optional int32 RadioType = 10;

message WifiDeviceMsg
    required string MAC = 1;
    optional string SSID = 2;
    optional int32 Rssi = 4;
```

```
message RequestMsg
    message PlatformProfileMsg
        required string Version = 1;
        optional string Platform = 2;
        optional string PlatformKey = 3;
        optional string Locale = 5;

        message CellularPlatformProfileMsg
            optional int32 RadioType = 1;
            optional string Carrier = 2;
            optional int32 HomeMnc = 4;
            optional int32 HomeMcc = 5;

        optional CellularPlatformProfileMsg CellPlatformProfile = 6;

    required PlatformProfileMsg PlatformProfile = 1;
    message RequestElementsMsg

        message CellularProfileMsg
            required CellMsg PrimaryCell = 1;
            required int64 Timestamp = 2;

        optional CellularProfileMsg CellularProfile = 1;

        message WifiProfileMsg
            required int64 Timestamp = 1;
            repeated WifiDeviceMsg WifiDevice = 2;

        optional WifiProfileMsg WifiProfile = 2;

        optional LocationProfileMsg LocationProfile = 3;

    repeated RequestElementsMsg RequestElements = 4;
```

Protocol Buffers Payload (Response)

```
message LatLngMsg
    required fixed32 Lat = 1;
    required fixed32 Lng = 2;

message LocationProfileMsg
    optional LatLngMsg LatLng = 1;
    optional int32 Accuracy = 3;
    optional int64 Timestamp = 6;
    optional int32 LocType = 8;
    optional int32 Altitude = 10;
    optional fixed32 Speed = 16;
    optional bool PluggedIn = 17;

message CellMsg
    required int32 Lac = 1;
    required int32 Cellid = 2;
    optional int32 Mnc = 3;
    optional int32 Mcc = 4;
    optional int32 Rssi = 5;
    optional int32 RadioType = 10;

message WifiDeviceMsg
    required string MAC = 1;
    optional string SSID = 2;
    optional int32 Rssi = 4;
```

```
message ResponseMsg
    required int32 Status = 1;
    message LocReplyElementMsg
        required int32 Status = 1;
        optional LocationProfileMsg Location = 2;
        message DeviceLocationMsg
            optional LocationProfileMsg Location = 1;
            optional CellMsg Cell = 2;
            optional WifiDeviceMsg WifiDevice = 3;
        repeated DeviceLocationMsg DeviceLocation = 3;
    repeated LocReplyElementMsg LocReplyElement = 2;
    optional string PlatformKey = 3;
```

Agenda

- Introduction
- **Protocol Analysis**
  Google Protocol
  Waze Protocol
- Evaluation
  Privacy
  Authenticity / Attack
- Solution
  Requirements
  Zero-Knowledge Protocols
  Protocol
  Discussion
- Conclusion

Waze Protocol

- Simple request/response protocol
- Complete source code is released under the GNU General Public License v2
- Position data is transmitted in the clear
- TLS for login
- Use mitmproxy to record packets
- Transmitted data is encoded as an ASCII string
- User usually registers himself before using the app
  - User gets a server ID and a cookie from the server
  - All subsequent messages contain the ID and the cookie

Waze Request Message

UID, 628311428 (server ID), 2Dyqtmg7r0HCZPFw (server cookie)

SeeMe, 2 (visability), 2 (visability report), T (download Wazers), T (download reports), T (download traffic), 1 (allow ping), -1 (events radius)

SetMood, 34 (mood)

Location, 9.946943 (longitude), 53.569241 (latitude)

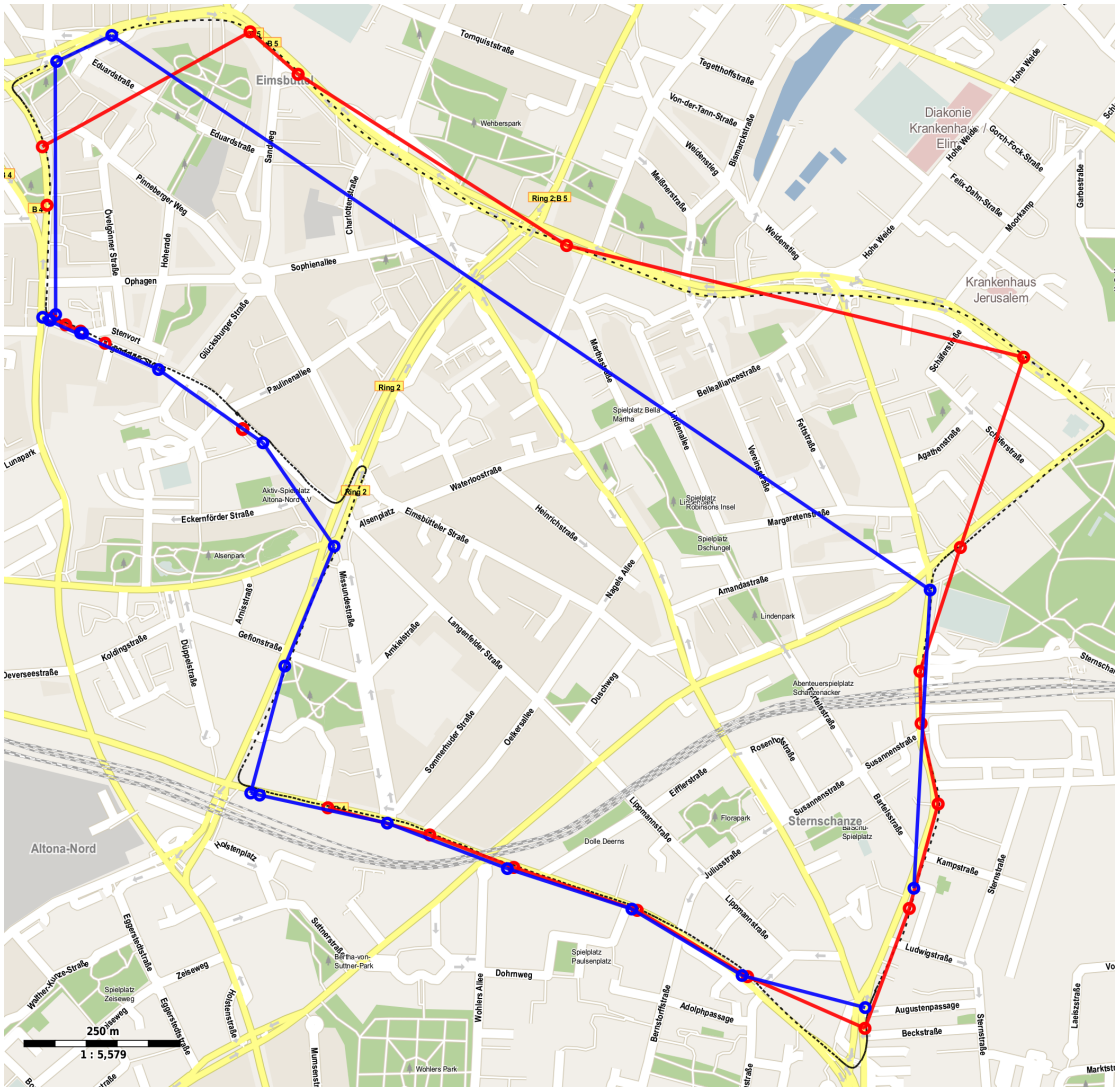
At, 9.951823 (longitude), 53.561904 (latitude), 0.000068 (altitude), -76 (steering), 17 (speed), 85068217 (from node), 85067935 (to node), T (refresh users)

GPSPath, 1334275968 (GPS time), 3 (count * 3), 9.965820 (longitude), 53.569185 (latitude), 57 (altitude), 0 (seconds gap)

Agenda

- Introduction
- Protocol Analysis
  - Google Protocol
  - Waze Protocol
- **Evaluation**
  - Privacy
  - Authenticity / Attack
- Solution
  - Requirements
  - Zero-Knowledge Protocols
  - Protocol
  - Discussion
- Conclusion

Google / Waze, GPS on / WiFi on



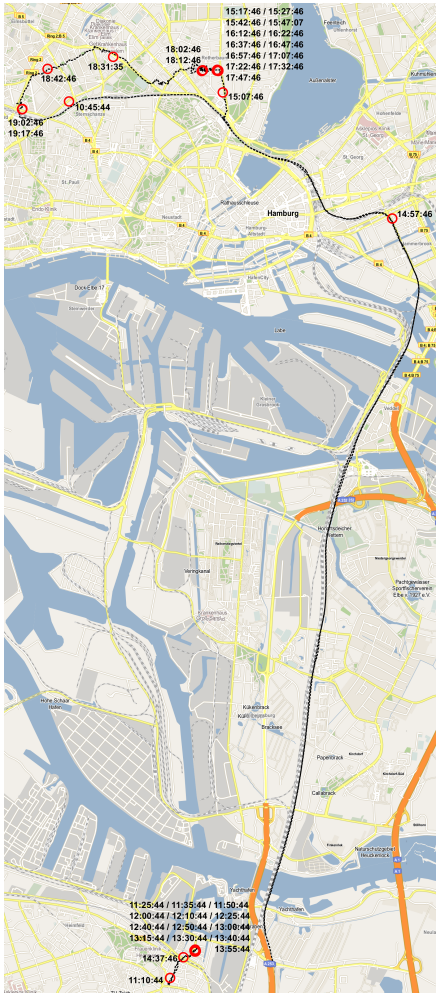
GPS track / data sent to Waze

Google, route 1

Google, route 2

measurement point

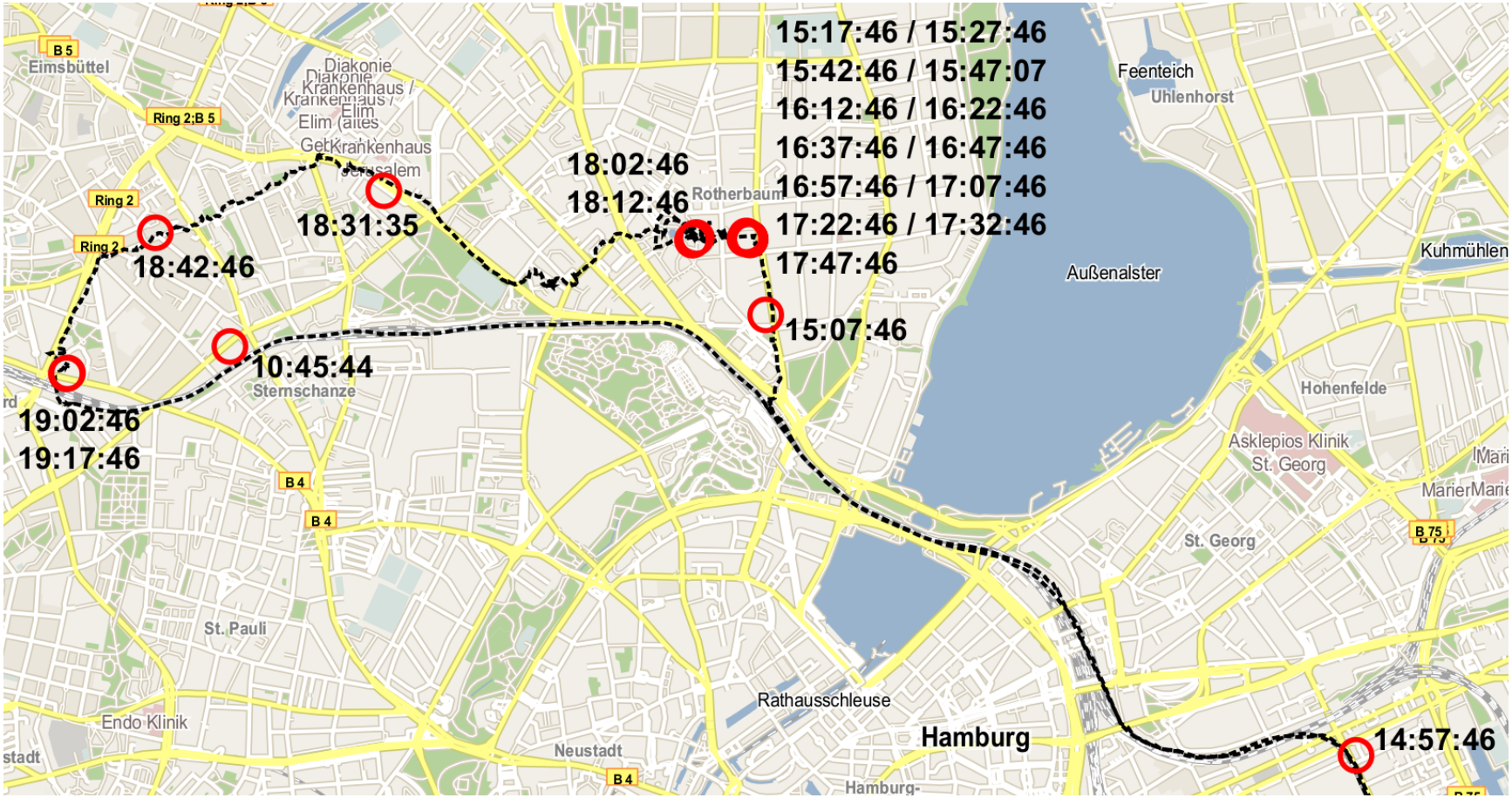
Google, GPS off / WiFi on (1)



..... GPS track / data sent to Waze

measurement point

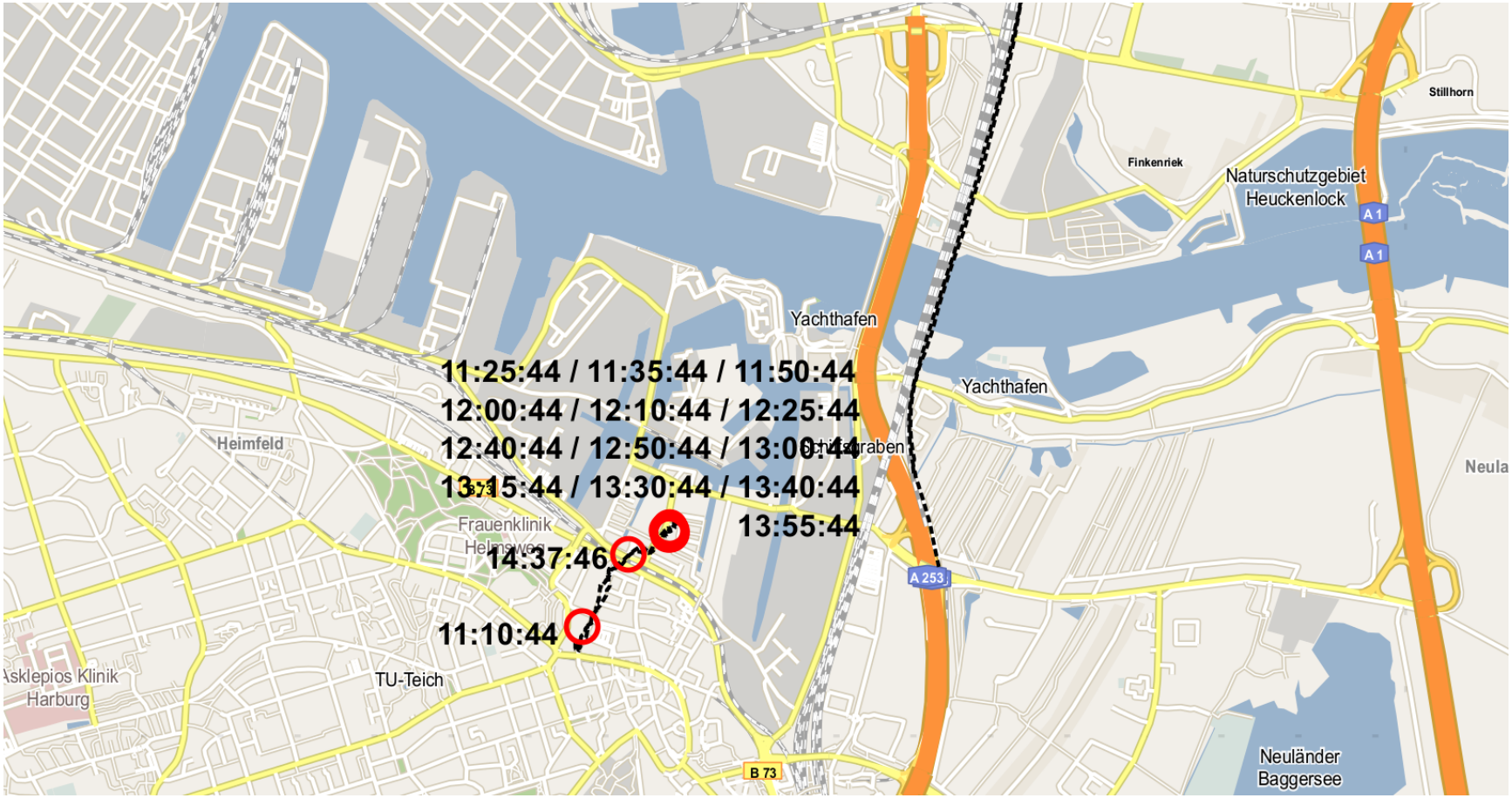
Google, GPS off / WiFi on (2)



····· GPS track / data sent to Waze

o measurement point

Google, GPS off / WiFi on (3)



····· GPS track / data sent to Waze

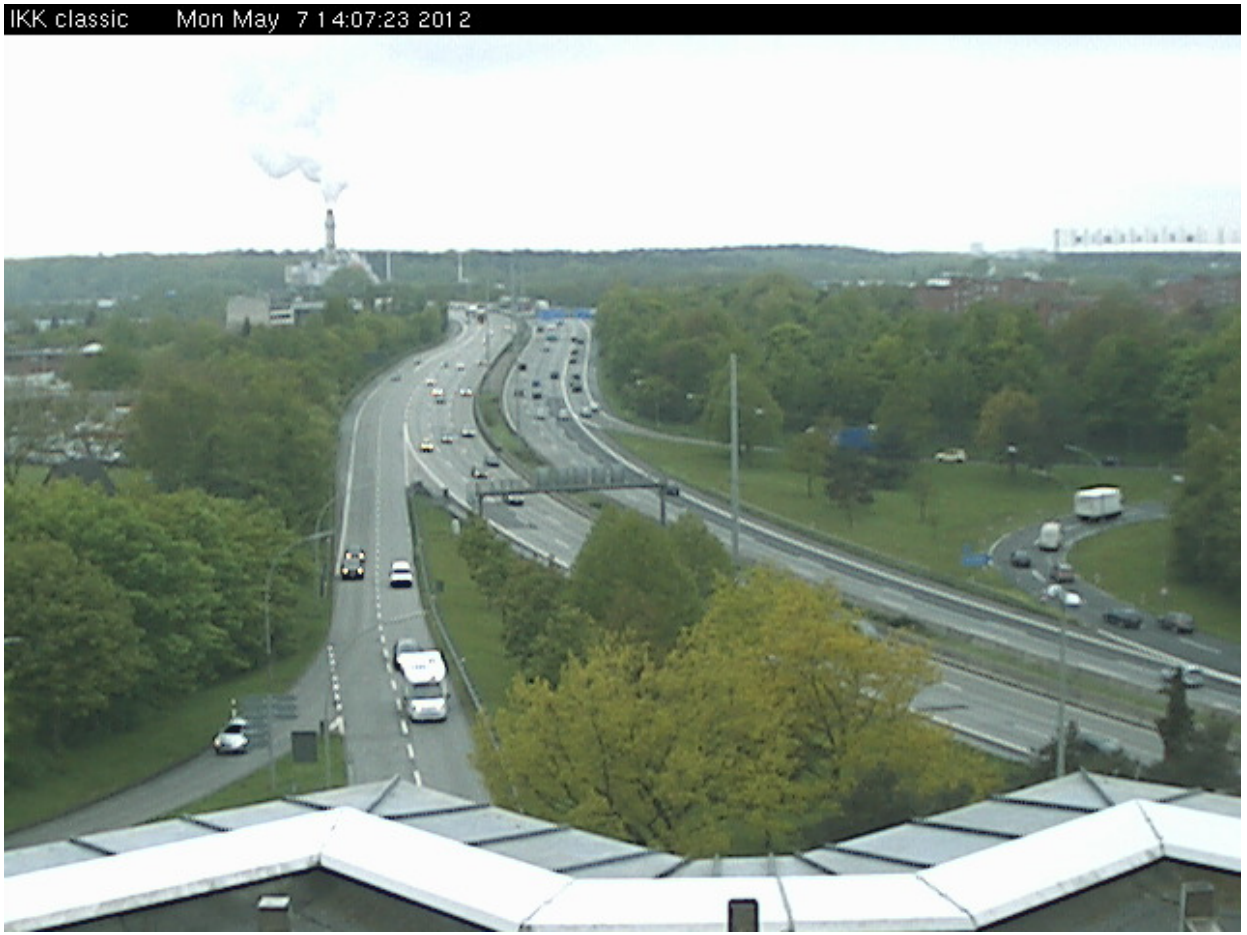
o measurement point

Privacy

- Platform key is generated by Google after first start but, uniquely identify the phone and never change (even after reboot)
- Android OS sends location data to Google even if Google Maps is not active (can be turned off at the cost of "user experience")
- Every MASF message has a sequence number
  $\rightarrow$ sequential ordering
- The Waze app periodically sends bunch of position data to the Waze server ($\rightarrow$ `GPSPath`)
- The Waze app sends Waze the current position from time to time ($\rightarrow$ `location`)
- Each message sent to Waze contains the unique server cookie and a server ID

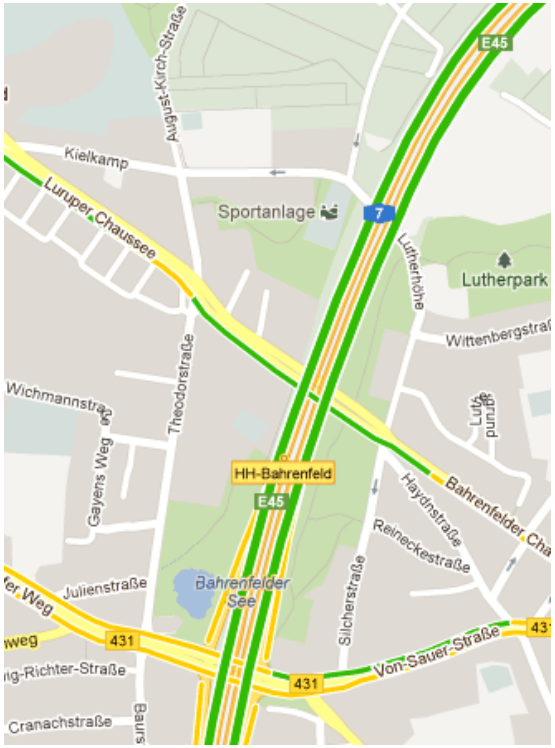
Authenticity / Attack

roads are clear...

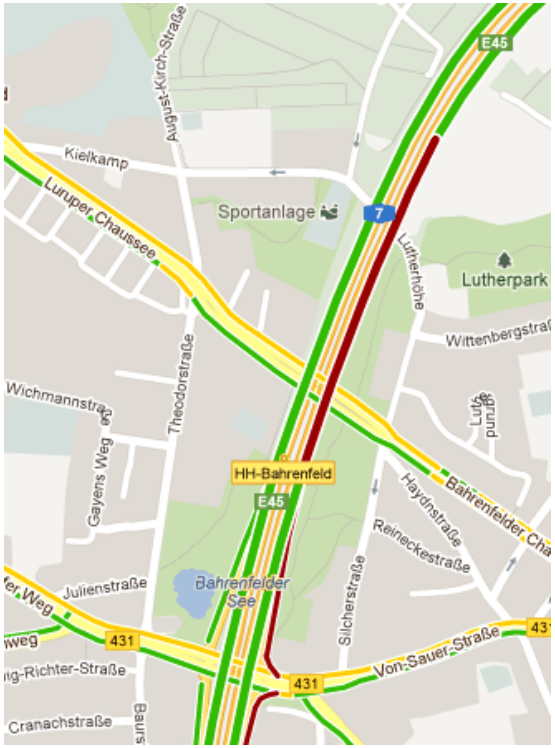


but...

Authenticity / Attack



(a) Before the attack



(b) Attack with wrong traffic data

Highway ramp A7 - Hamburg-Bahrenfeld, map data © Google

Authenticity / Attack

- TLS tunnel ensures data integrity but what if the attacker controls the beginning of the TLS tunnel?
  - Attacker randomly selects cookie and ID in the MASF header
  - Platform Key is generated by Google

  $\Rightarrow$ Attacker stays anonymous
- Attacker drives a route, collects data packets and replay them later with changed time stamps
- Attack can be intensified by carrying out several delayed transmissions with different cookies and platform keys to simulate multiple cars
  $\rightarrow$ adding noise to the measured values, use different IP addresses $\rightarrow$ distinction between real and fake location information is no longer possible

Authenticity / Attack

- An attacker can make people drive into traffic jams or keep roads clear if traffic data is used for navigation
- Important difference to the TMC attack
  - Low cost
  - Manipulation of traffic data worldwide

- Attack scenario can also be applied to Waze
- Attack becomes more difficult because the position data is associated with a user account
  - $\rightarrow$ however, position data can be transferred to Waze if user authentication is not performed
  - $\rightarrow$ attacker remains anonymous

Network Location Provider Protocol

- We use the Geolocation API in Google Gears to visualize the data points (mapping signal strengths from Wi-Fi AP to geographic coordinates)

- In 2011, Samy Kamkar found out that AP can be located worldwide by using the Geolocation API [10]
- Google changed its system → request containing Wi-Fi must at least have two MAC addresses of nearby AP
- ~~Still possible to locate a single AP → send Google two MAC addresses, one of the requested AP and one of an unknown AP~~ bug has been fixed by Google

Agenda

- Introduction
- Protocol Analysis
  - Google Protocol
  - Waze Protocol
- Evaluation
  - Privacy
  - Authenticity / Attack
- **Solution**
  - Requirements
  - Zero-Knowledge Protocols
  - Protocol
  - Discussion
- Conclusion

Requirements

Privacy:

- Smartphone owners are interested in a high degree of privacy
- User tracking by providers such as Google or Waze is generally considered as undesirable by the user

Authenticity:

- The provider is interested in the correctness of the data
- "Malicious smartphones" should be excluded from the calculation of the traffic flow
- Incorrect traffic data influences the user's navigation
  $\rightarrow$ hackers can affect the traffic flow

Zero-Knowledge Proof of Knowledge

Proof of Knowledge:

- Proof between prover & verifier that a mathematical statement is true
- An honest prover can always convince a verifier
- A dishonest prover will fail to convince a verifier with overwhelming probability

Zero-Knowledge Proof of Knowledge:

- Proof of knowledge where a verifier obtains no further information from the prover other than the fact that the prover knows the solution of the underlying mathematically hard problem
- Protocol runs are unlinkable
- Applications ($\rightarrow$ Authentication protocols, electronic cash, smart metering...)

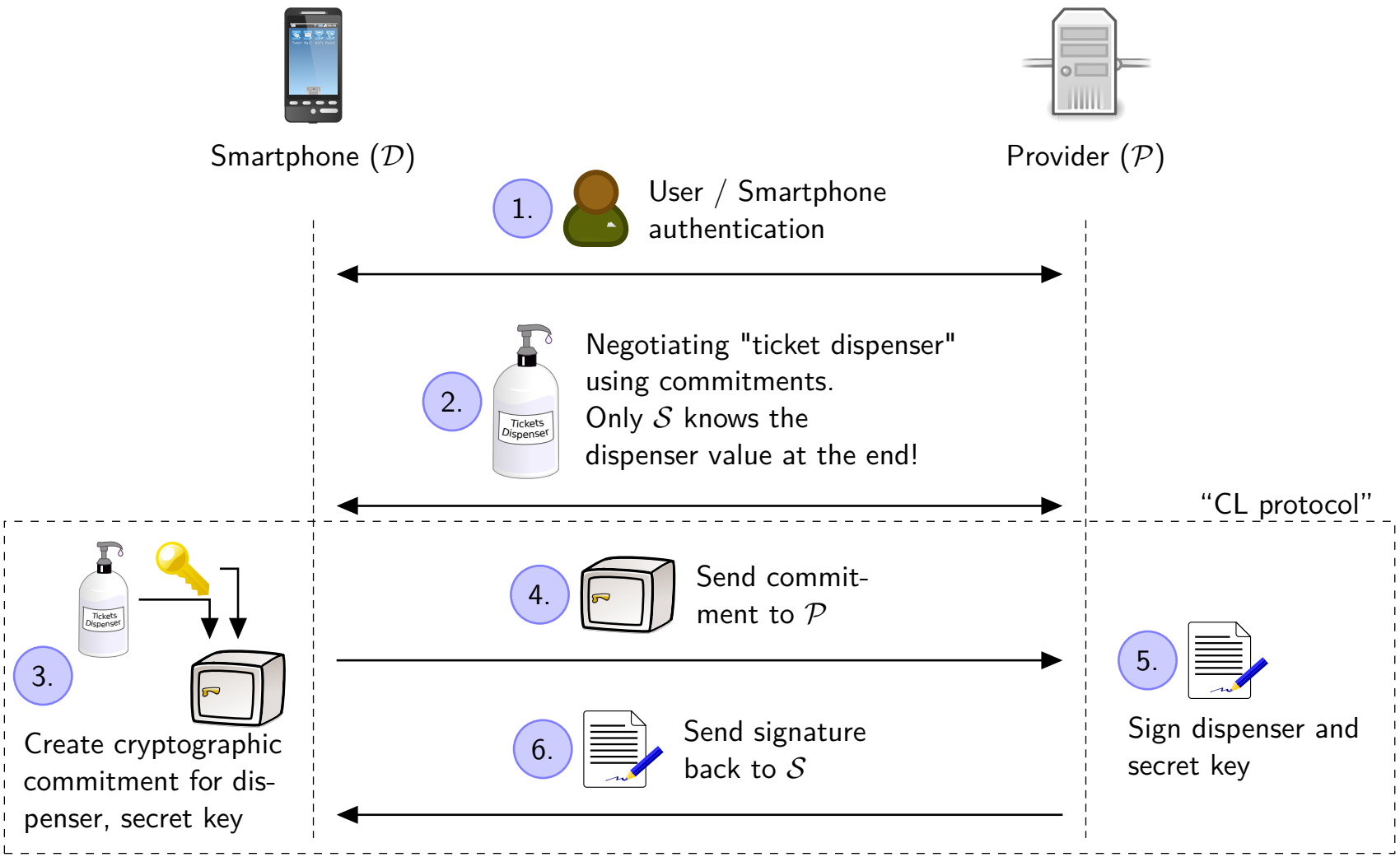
Protocol

Idea: Linking location information with tickets [4]

Protocol:

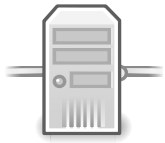
- *"Get Dispenser"-Protocol*
  - $\mathcal{D}$ (device / smartphone) authenticates itself to $\mathcal{P}$ (provider, e.g. Google) once and receives a "ticket dispenser"
- *"Submission"-Protocol*
  - With the help of the dispenser, $\mathcal{D}$ generates tickets in order to send authenticated position data to $\mathcal{P}$
  - $\mathcal{P}$ is able to check the validity of the tickets, but can't link tickets to a specific device due to the zero-knowledge techniques used
  - Each ticket has a time stamp limiting its validity in a fixed time slot (e.g. every 15 minutes). This restricts the maximum number of data packets per time slot and device

"Get Dispenser"-Protocol (simplified)



“Submission”-Protocol (simplified)

Smartphone ($\mathcal{D}$)



Provider ($\mathcal{P}$)

1. 

Proof in ZK that CL signature of dispenser and secret key is correct

2.

Create new ticket for current time intervall c

3. 

Proof in ZK that ticket was correctly created

4. 

Send ticket, c and position data*

5.

Store ticket & c on blacklist
If ticket and timestamp was not used before (and proofs are correct)
$\rightarrow$ accept data

*Proofs "sign" these values!

Benchmark Results[2]

	Get Dispenser $(\mathcal{D})$	**Get Dispenser** $(\mathcal{P})$
Nexus S	112 ms	73 ms
Intel Xeon X3460	5 ms	3 ms

	Submission[1] $(\mathcal{D})$	**Submission** $(\mathcal{P})$
Nexus S	318 ms	154 ms
Intel Xeon X3460	14 ms	7 ms

$\Rightarrow$ **Results show that the protocol is already practically today!**

[1] Most of the calculations can be pre-calculated in the background!

[2] security level $\approx$ 1024 bit RSA

Discussion

Possible that data packets can be linked by their IP address, but:

- Mobile data connection is often disconnected (especially the case if the phone is moved)
- Several providers (at least in Germany) automatically disconnect the connection after a few hours
- Use anonymity networks such as Tor [11] for transmitting location data

The protocol can be extended:

- Identify misbehaving users [4]
- Limit the validity of the ticket dispenser
  $\rightarrow$ Device is forced to re-authenticate itself, e.g. every week
  $\rightarrow$ Chance to remove devices from the database

Agenda

- **Introduction**
- **Protocol Analysis**
  - Google Protocol
  - Waze Protocol
- **Evaluation**
  - Privacy
  - Authenticity / Attack
- **Solution**
  - Requirements
  - Zero-Knowledge Protocols
  - Protocol
  - Discussion
- **Conclusion**

Conclusion

- Evaluation of the Google and Waze protocol regarding privacy and authenticity
- Anonymity of the user is not assured
  $\rightarrow$ user tracking is possible
- Attackers can anonymously manipulate the traffic analysis and actively influence the navigation software
- There is a solution which increases the user's privacy and at the same time makes attacks manipulating the traffic analysis more difficult
- **Results of this research can be transferred to every other navigation system which uses real-time FCD!**

Demo

Thank you for your attention!

Any questions?

Please make sure you fill out the Black Hat Evaluation Form!

Contact

Tobias Jeske

TU Hamburg-Harburg
Institute for Security in
Distributed Applications
Harburger Schloßstraße 20
21079 Hamburg

Tel.: +49 (0)40/42878-3540
Fax: +49 (0)40/42878-2471
eMail: tobias.jeske@tu-harburg.de

revised version (if available) at:
https://www.sva.tuhh.de/

Literature I

Aldo Cortesi. *Mitmproxy 0.7 - an SSL-capable man-in-the-middle proxy*. 2012. URL: http://www.mitmproxy.org/.

Julia Angwin and Jennifer Valentino-Devries. *Apple, Google Collect User Data*. Apr. 2011. URL: http://online.wsj.com/article/SB10001424052748703983704576277101723453610.html.

Andrea Barisani and Daniele Bianco. "Injecting RDS-TMC Traffic Information Signals". In: *TELEMOBILITY 2007*. Nov. 2007.

Literature II

J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. "How to win the clone wars: Efficient periodic n-times anonymous authentication". In: *ACM Conference on Computer and Communications Security. ACM*. 2006.

J. Camenisch and M. Stadler. *Proof Systems for General Statements about Discrete Logarithms*. Tech. rep. 260. Institute for Theoretical Computer Science, ETH Zürich, 1997.

Literature III

Jan Camenisch and Anna Lysyanskaya. "A Signature Scheme with Efficient Protocols". In: *Security in Communication Networks*. Ed. by Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi. Vol. 2576. Lecture Notes in Computer Science. 10.1007/3-540-36413-7_20. Springer Berlin / Heidelberg, 2003, pp. 268–289. URL: http://dx.doi.org/10.1007/3-540-36413-7%5C_20.

David Wang. *Stuck in traffic?* Feb. 2007. URL: http://googleblog.blogspot.de/2007/02/stuck-in-traffic.html.

Google. *Protocol Buffers - Google's data interchange format*. URL: http://code.google.com/p/protobuf/.

Literature IV


Roy Williams. *Youve got better things to do than wait in traffic*. Mar. 2011. URL: http://googlemobile.blogspot.de/2011/03/youve-got-better-things-to-do-than-wait.html.


Samy Kamkar. *android map*. 2011. URL: http://samy.pl/androidmap/index.php.


The Tor Project. *Tor: anonymity online*. 2011. URL: https://www.torproject.org/.


Waze. *Waze - Outsmarting Traffic, Together*. URL: http://www.waze.com/.

Description ZPKs

Camenisch/Stadler Notation [5]

$$\text{ZPK}\left[(\omega) : \underbrace{x = g^{\omega}}_{predicate_i}\right]$$

- ZPK that the prover knows the secret w with $x = g^w$ ($x, g, w \in \mathbb{Z}_p$, p is prim), a homomorphism from w to x
- Secrets are denoted with Greek letters
- x and g are public values

Implementation: Schnorr's Identification Scheme

$\mathcal{P}[x, g, w]$		$\mathcal{V}[x, g]$
$k \in_R \mathcal{G}, r := g^k$		
	$\xrightarrow{r}$	
		$c \in_R \mathcal{C}$
	$\xleftarrow{c}$	
$s := k + cw$		
	$\xrightarrow{s}$	
		$g^s x^{-c} \stackrel{?}{=} r$
		$\Leftrightarrow g^{k+cw} x^{-c} = g^k$
		$\Leftrightarrow g^k x^c x^{-c} = g^k$

w: secret value, $x = g^w$: public value,

r: commitment, c: challenge, s: response

Implementation: Schnorr's Identification Scheme (non-interactive)

$\mathcal{P}[x, g, w]$	$\mathcal{V}[x, g]$
$k \in_R \mathcal{G}, r := g^k$	
$c = H(r)$	
$s := k + cw$	
$\xrightarrow{s,\ c}$	
	$H(g^s x^{-c}) \stackrel{?}{=} c$
	$\Leftrightarrow H(g^{k+cw} x^{-c}) = H(g^k)$
	$\Leftrightarrow H(g^k x^c x^{-c}) = H(g^k)$

w: secret value, $x = g^w$: public value,

r: commitment, c: challenge, s: response, H: hash function

More complicated proofs

- AND/OR proofs:

$$\mathrm{ZPK}\left[(\omega_1, \ldots, \omega_n) : \bigvee \bigwedge Pred_i(\omega_i)\right]$$

- Multiplicative proofs:

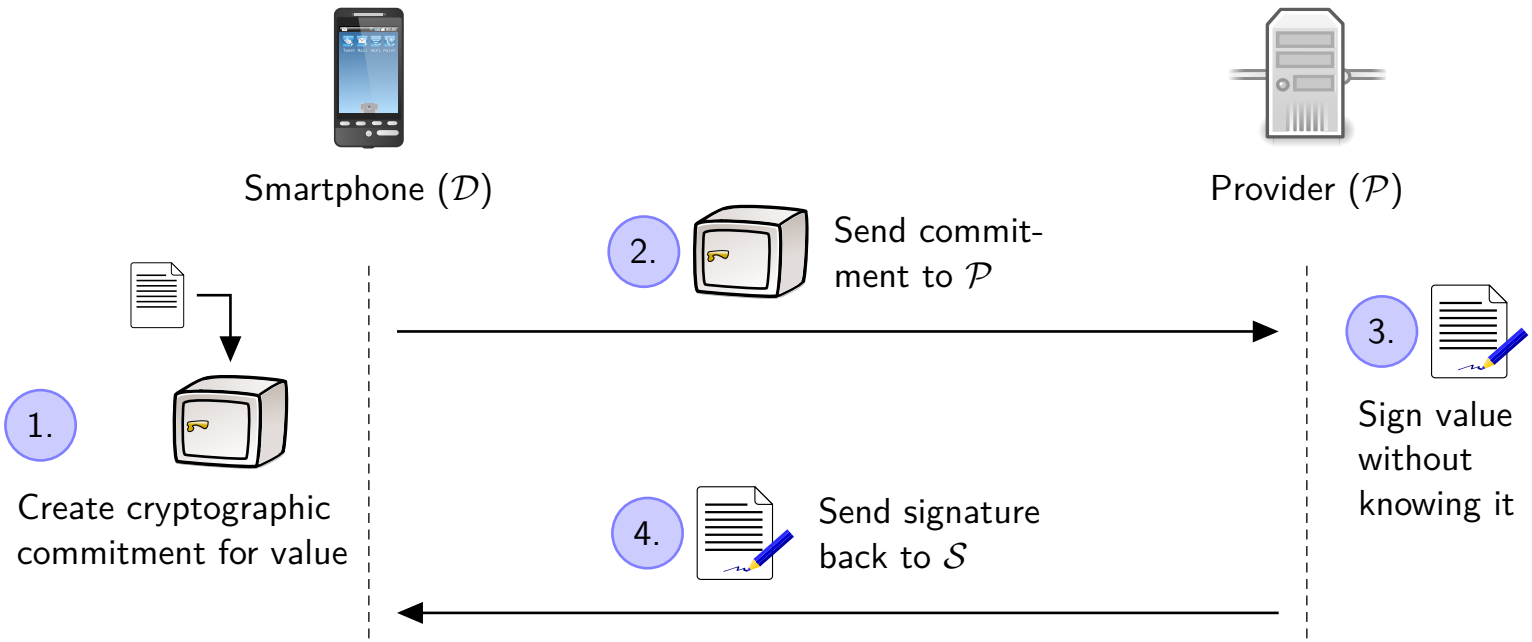
$$\mathrm{ZPK}\left[(\omega_1, \omega_2, \omega_3) : x = g^{\omega_1} h^{\omega_2} b^{\omega_3} \wedge \omega_3 = \omega_1 \cdot \omega_2\right]$$

- Interval proofs:

$$\mathrm{ZPK}\left[(\omega_1, \omega_2) : x = g^{\omega_1} h^{\omega_2} \wedge \omega_1 \in [a, b]\right]$$

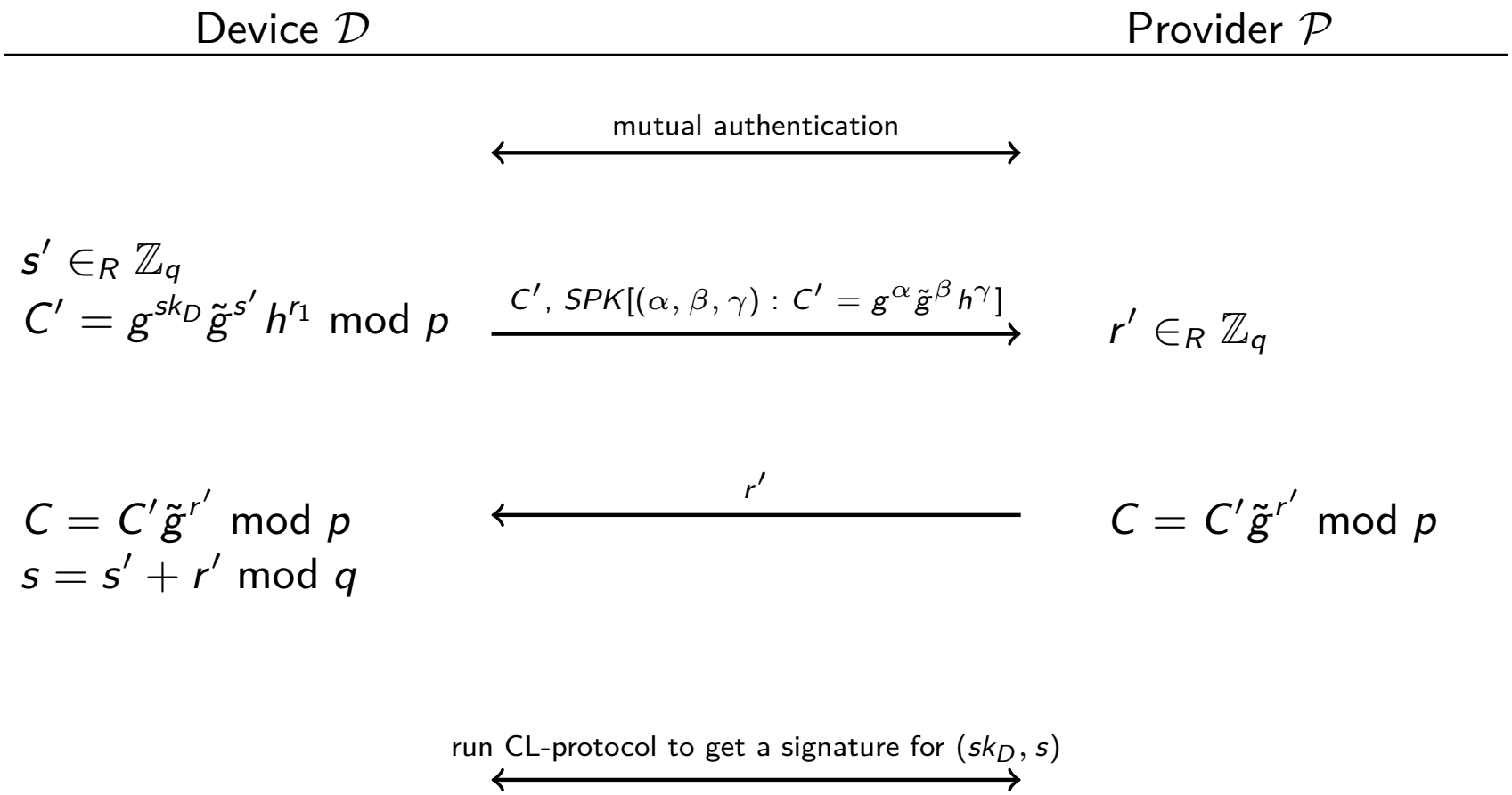
“CL-Protocol” [6]

- Protocol for issuing a signature on committed value(s)
  $\rightarrow$ Signer gets no information about the signed value(s)!



- $\mathcal{D}$ can later proof in zero-knowledge that it has a valid signature of the committed value(s)!

"Get Dispenser"-Protocol (simplified)



“Submission”-Protocol (simplified)

Device $\mathcal{D}$	Provider $\mathcal{P}$

$S = F_{g,s}(c) = g^{\frac{1}{s+c}} \bmod p$
$C_D = g^{sk_D} h^{r_1} \bmod p$
$C_s = g^{s} h^{r_2} \bmod p$

$\mathcal{D}$ proves to have a valid CL signature of (sk_D, s) from $\mathcal{P}$ $\longrightarrow$

$SPK[(\alpha, \beta, \gamma, \delta) : S = g^{\alpha} \wedge g = (C_s g^{c})^{\alpha} h^{\beta} \wedge C_u = g^{\gamma} h^{\delta}](m),\ C_D,\ C_s,\ S,\ c,\ m$ $\longrightarrow$

If proofs are correct and c corresponds to the current time, accept data and store (S, c) in database.