Secure and Dynamic Route Navigation through RSU-based Authentication in IoV for Smart City

Abstract

1)One of the significant services provided by IOV is Vehicular Navigation.

2)A proper routing mechanism help drivers reach their destination to minimum time and with less fuel consumption

3)However such protocols often face security challenges

4)In this paper authors have proposed an authenticated navigation scheme with the help of pseudonym based asymmetric key cryptography

5)The architecture embodies GLP to find static route to a particular destination and message forwarding capabilities of RSUs to develop dynamic route after receiving feedback about traffic conditions

Key Features of this Protocol

1)Integrity

2)Anonymity

3)Unlinkability

4)robust protection from important security threats

5)minimal end to end delay

6)efficient real time routing

Vehicular ad hoc networks

Key INTRODUCTION Points

1)Amalgnation of Internet of Things to VANETS has elevated vehicular communication to the next level known as Internet of vehicles

2)Disadvantages of VANETS

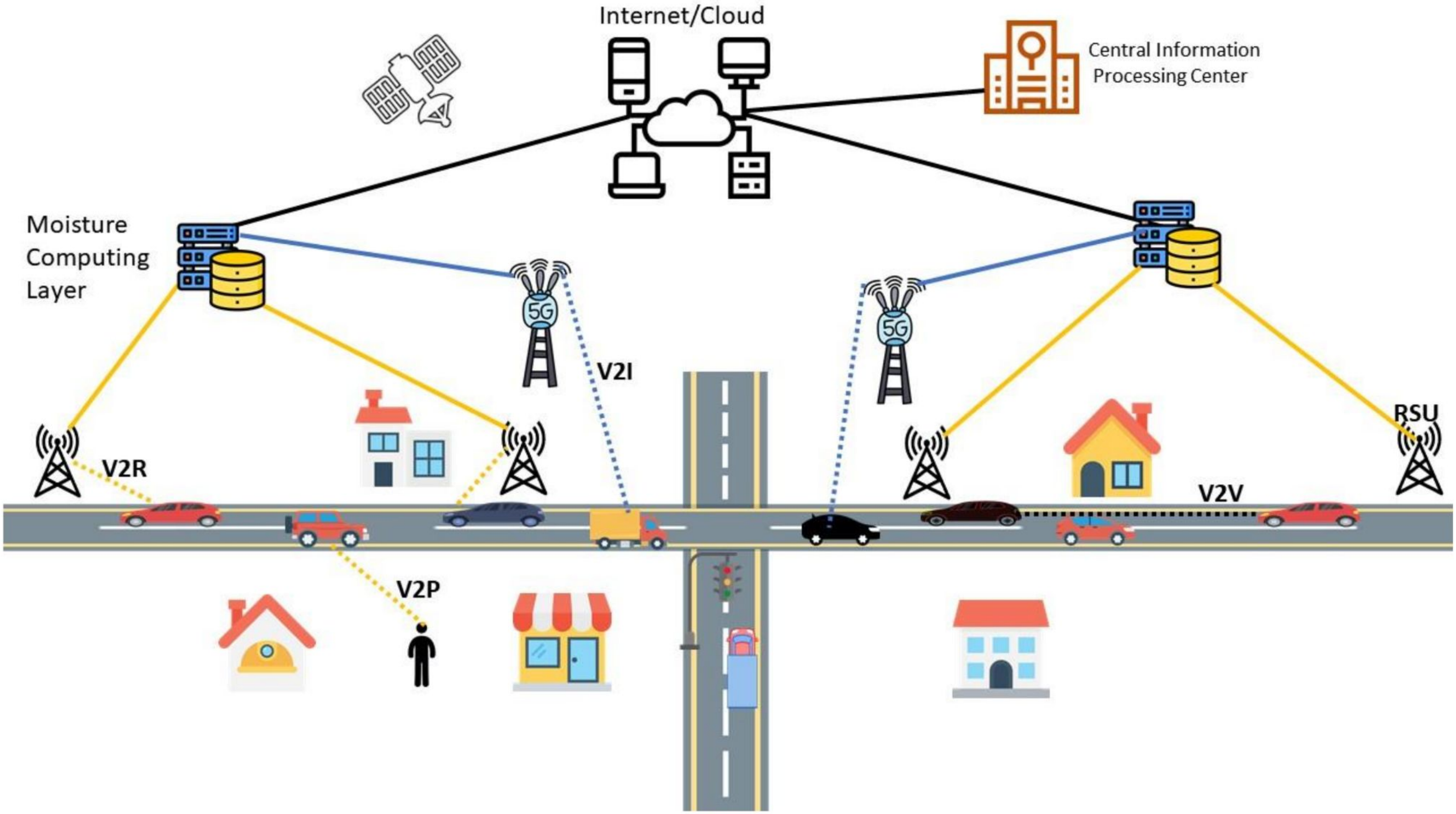
i)Low Network coverage area

ii)limited mobility

iii)restriction on number of vehicles

3)these will be rectified by internet of technologies hence this combo is powerful

INTERNET OF VEHICLES ARCHITECTURE


Internet/Cloud
Central Information Processing Center
Moisture Computing Layer
5G
V2I
5G
RSU
V2R
V2V
V2P

ISSUES IN THE IOV ARCHITECTURE

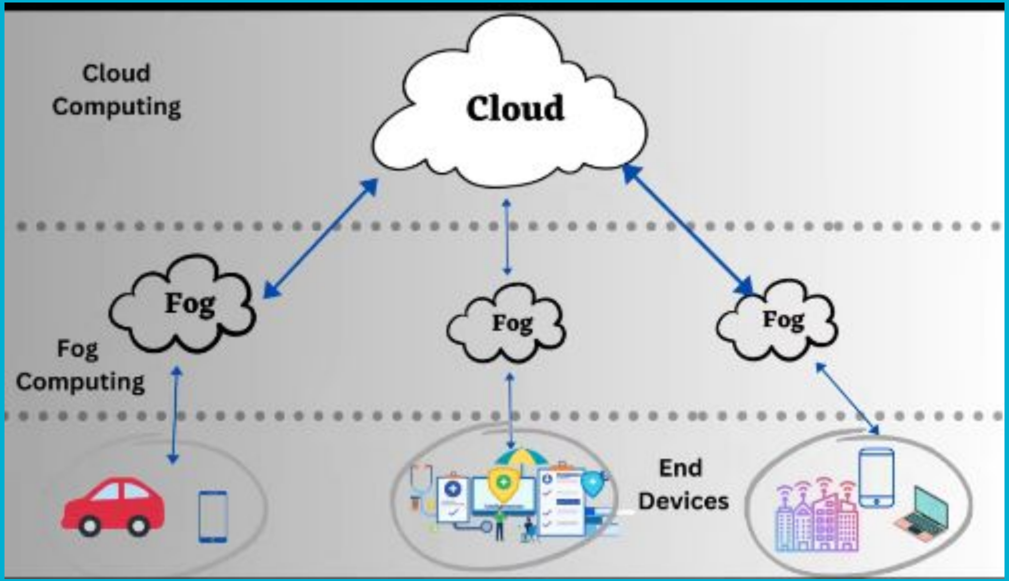
Three main issue in IOV Architecture

1)Since the cloud server is a centralized facility, it often encounters serious problems elaborated next. Firstly, it faces the issue of high computational latency due to the loads exerted by RSUs. Since all the RSUs are connected directly to the server, it often fails to provide real-time services to the drivers and passengers. It may lead to gross failure in fulfilling the objective of the IoV.

2)Secondly, breakdown of the cloud server leads to complete system failure unless there is an alternate arrangement

SOLUTION FOR THE ABOVE ISSUES IS FOG computing

FOG COMPUTING



Ref :- https://www.youtube.com/shorts/wlJZE_xKSZ4

Third Issue

The third issue is the security havoc in the centralized server, thus causing extensive damage to human lives

To address the Third issue it is crucial to separate server data security from message authentication

Key Feautures of The Proposed Framework

- A secure and dynamic route navigation framework through **RSU-based authentication** for IoV.
- **Geo-Location Provider (GLP)** and RSUs collaborate to function **without involving vehicles directly**, reducing delay and dependency.
- Uses **ElGamal encryption**, **digital signatures**, and **public key certificates** for secure communication.
- **Lightweight design**: avoids heavy operations like ECC point multiplication or bilinear pairings.

Security & Performance Highlights

- Ensures **driver anonymity**, **session key secrecy**, and **unlinkability**.
- Defends against **collusion**, **Sybil**, **MITM**, **replay**, and **masquerading attacks**.
- **GLP acts as a trusted intermediary** between navigation requesters and RSU responders.
- Verified using **Scyther tool**, confirming **strong security guarantees** and **efficient performance**.

Mathematical Preliminaries-Cryptographic Foundations

- Our protocol uses:
  - **ElGamal-based encryption, decryption, and digital signature**
  - **Cryptographic hash functions**
- Operations are performed over a **multiplicative group G** with generator **g**
- Common operations: **Multiplication** and **Exponentiation**

The ElGamal Cryptosystem

Key Generation (Alice – message receiver)

Select p	p is a very large prime number
Find a primitive root of p: g	
Choose a random integer **a** as her private key	$1<a<p-1$
Compute e: $e=g^a \bmod p$	
Public key: (p,g,e)	

Encryption (Bob– message sender)

Plaintext: m	$m<p$
Select a random integer: **b**	$1<b<p-1$
Compute two values C_1 and C_2, where	
$C_1=g^b \bmod p$	
$C_2=m*e^b \bmod p$	
Ciphertext: (C_1, C_2)	

Decryption (Alice – message receiver)

Ciphertext: (C_1, C_2)
Plaintext:
$x=(C_1)^a \bmod p$
$m=C_2*x^{(p-2)} \bmod p$

The ElGamal Cryptosystem

Three steps with math formulas

Receiver — **Alice**

a: $1<a<p-1$

1 Key Generation

p, g, a

$e = g^a \bmod p$

public key (p, g, e)

3 Decryption

$x = (C_1)^a \bmod p$

$m = C_2 * x^{(p-2)} \bmod p$

Public

public key (p, g, e) →

← (C_1, C_2)

Sender — **Bob**

b: $1<b<p-1$

Bob wants to send m(plaintext) to Alice.

2 Encryption

Where $m<p$

Select a random integer b

(p, g, e)

$C_1 = g^b \bmod p$

$C_2 = m * e^b \bmod p$

cipertext: (C_1, C_2)

Digital Signature

Digital signature is equivalent to an hand written signature

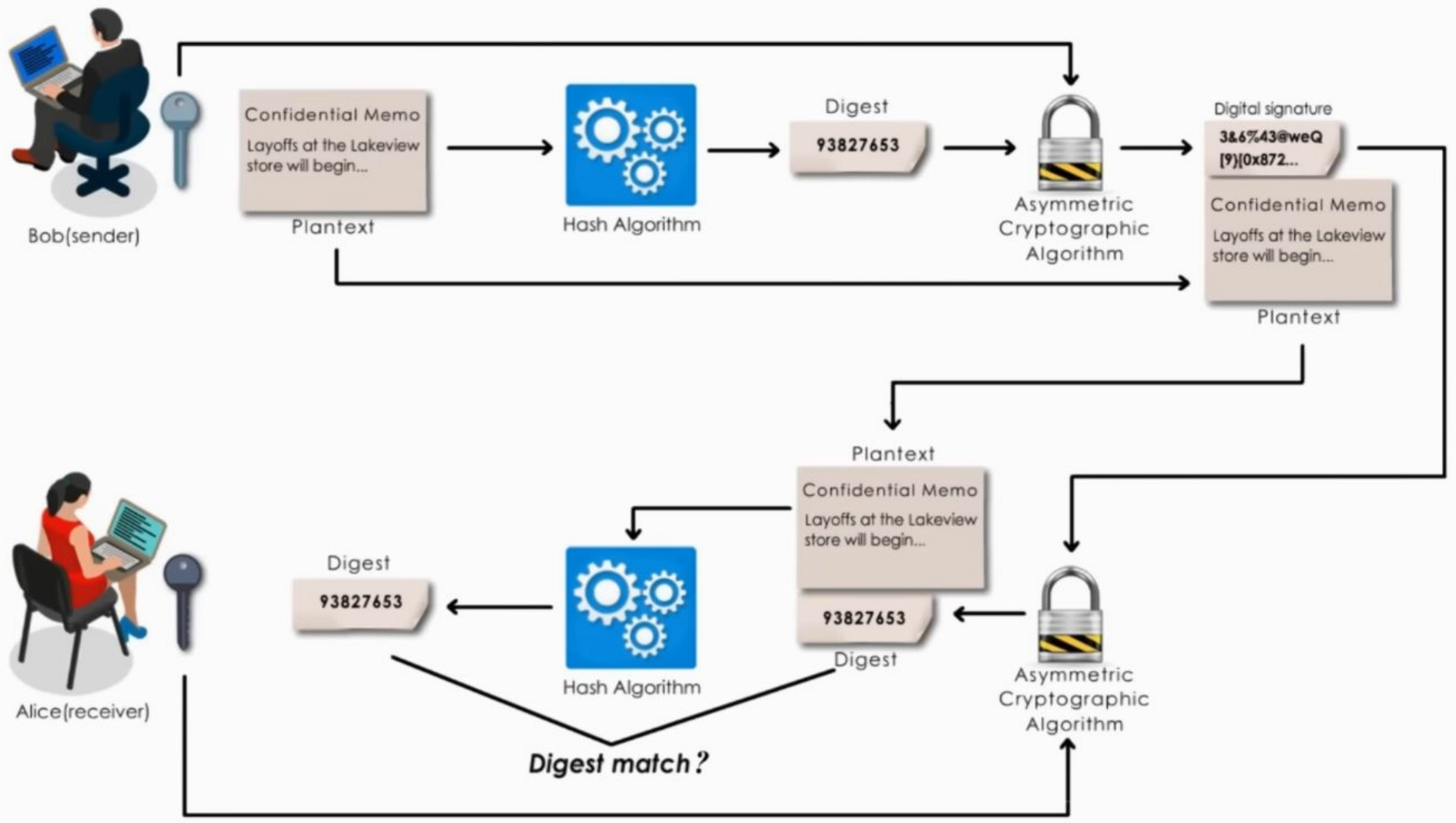
A Digital signature basically serves Three purposes

1)Authentication-A DS gives the receiver a reason to believe the message was created and sent by the claimed sender

2)Non repudiation-The receiver cant deny he didnt receive a message later on

3)Integrity-a DS assure no altering of the message is occurred

A digital signatures uses asymmetric key cryptography -public key cryptosystem


Bob(sender)
Confidential Memo
Layoffs at the Lakeview store will begin...
Plantext
Hash Algorithm
Digest
93827653
Asymmetric Cryptographic Algorithm
Digital signature
3&6%43@weQ [9)[0x872...
Confidential Memo
Layoffs at the Lakeview store will begin...
Plantext
Plantext
Confidential Memo
Layoffs at the Lakeview store will begin...
93827653
Digest
Asymmetric Cryptographic Algorithm
Hash Algorithm
Digest
93827653
Alice(receiver)
Digest match?

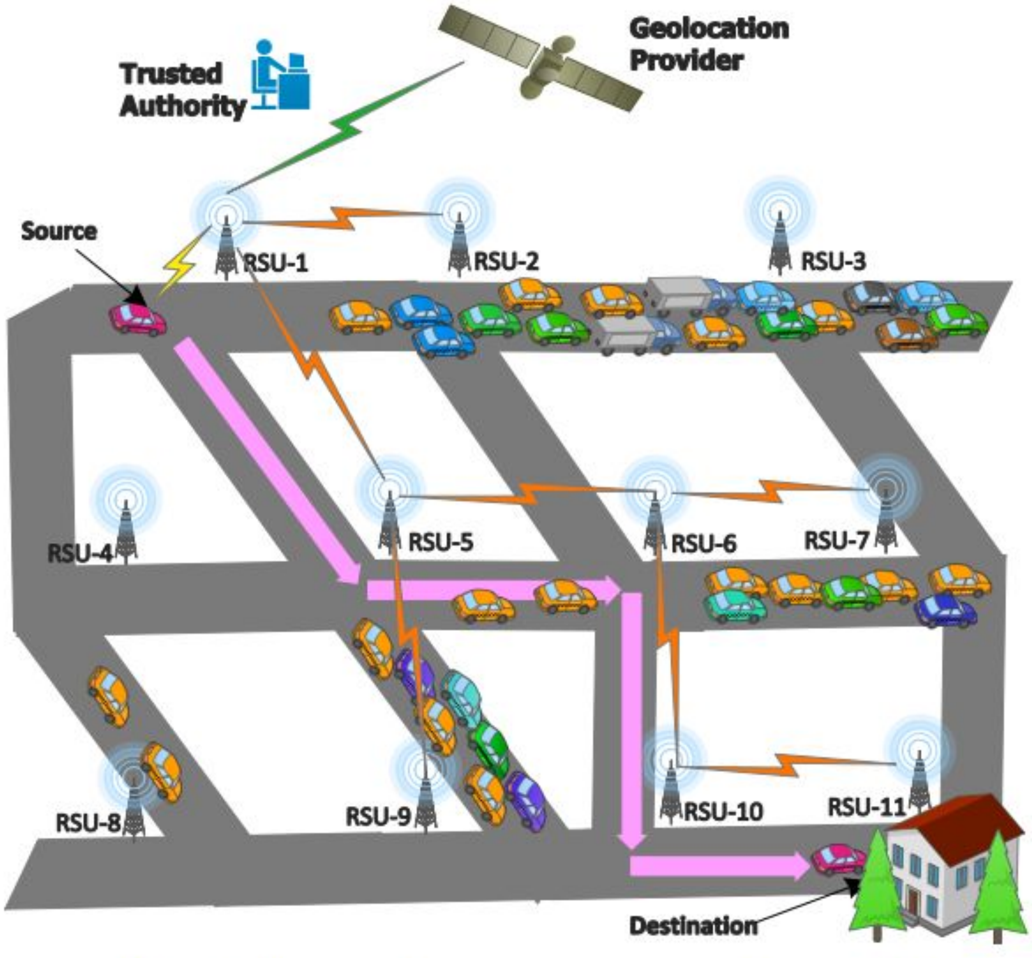


Fig. 2: Route selection in our navigation model.

TABLE I: Notations used in our proposed protocol

Notations	Meaning
GLP, V_i	Geolocation Provider, Vehicle i
R_j, TA	Road Side Unit j, Trusted Authority
ID_{V_i}, ID_{R_j}	Identity of V_i, Identity of R_j
PID_{V_i}, $PRID_j$	Pseudo-identity of V_i, Pseudo-identity of R_j
SND	Session-ID created by RSU
SID	Session-ID created by GLP
s, l, v	Random numbers
PR_x, PU_x	Private key of entity x, Public key of entity x
g, G	g is a generator of group G
$h(.)$	One-way hash function $h : \{0,1\}^* -> G$
$ENC_{PU_x}(M_k)$	Encryption of message M_k with public key PU_x of entity x
$SIG_{PR_x}(M_k)$	Signature on message M_k with private key PR_x of entity x
$Cert_x$, T	Public key certificate of entity x, Timestamps

Working of The Protocol

there are mainly 5 steps discussed in the Research paper IN this section

1)System initialisation

2)Geolocation Provider Registration

3)RSU registration

4)Vehicle Registration

5)Navigation Scheme

System Initialisation

Performed by: Trusted Authority (TA)

- **Step 1:** Select a multiplicative group **G** of order **q**, with generator **g**
- **Step 2:** Choose a master secret key **s ∈ Z*_q**
- **Step 3:** Compute public key:
  ```
  PUTA = gˢ mod q
  ```
- **Step 4:** Choose a one-way **hash function h(.)**
- **Step 5:** Create a **digital certificate**:
  ```
  CertTA = <TA, PUTA, SIGPRTA(TA, PUTA)>
  ```
- **Step 6:** Publish system parameters:
  ```
  <G, q, g, h(.), CertTA>
  ```

✅ These parameters are then shared with RSUs and vehicles for secure communication and authentication.

Let's assume:

- TA ID: `"TA"`
- Public Key (from earlier): `PUTA = 6`
- Hash function: `h(.)` (e.g., SHA-256 or similar)

Digital Certificate Construction:

makefile

Copy Edit

```makefile
CertTA = <"TA", PUTA, SIGPRTA("TA", 6)>
        = <"TA", 6, Signature>
```

Published System Parameters:

php-template

Copy Edit

```php-template
<G = Z*23, q = 23, g = 5, h(.), CertTA>
```

📌 These values are distributed securely to:

- RSUs (for registration and route verification)
- Vehicles (for pseudonym-based authentication)

GLP REGISTRATION

Goal: Register the **Geo-Location Provider (GLP)** securely with the **Trusted Authority (TA)**

Steps:

1. **GLP → TA:** Sends identity proof over a **secure channel** (e.g., TLS)
2. **TA → GLP:**
   - Generates a **pseudo-identity**:

     `PIDGLP = h(s || IDGLP)`

   - Selects a random number **l ∈ Z*_q**
   - Computes **private key**:

     `PRGLP = g^l`

   - Computes **public key**:

     `PUGLP = PRGLP^s`

3. **GLP stores:**

   `PIDGLP`, `PRGLP`, `PUGLP` for future use

- Prime order: `q = 23`
- Generator: `g = 5`
- Master secret (from TA): `s = 6`
- GLP ID = `"GLP1"` → Hash: `h(s || IDGLP) = 14`
- Random number selected by TA: `l = 3`

Calculations:

- `PRGLP = g^l mod q = 5^3 mod 23 = 125 mod 23 = 10`
- `PUGLP = PRGLP^s mod q = 10^6 mod 23 = 1,000,000 mod 23 = 6`

Goal: Securely register a **Roadside Unit (RSU)** with the **Trusted Authority (TA)**

Steps:

1. **RSU → TA:** Sends:
    - **Identity proof (IDRSUj)**
    - **Location code (LCj)**
      (Over a **secure channel**)
2. **TA computes:**
    - **Pseudo-ID:** `PRIDj = h(s || IDRSUj || LCj)`
    - **Private Key:** `PRRSUj` (randomly chosen)
    - **Public Key:** `P URSUj = g^PRRSUj`
3. **TA creates a digital certificate:**

```ini
CertRSUj = <PRIDj, LCj, PURSUj, SIGPRTA(PURSUj, PRIDj, LCj)>
```

4. **TA sends to RSU:**

```
<CertRSUj, CertGLP, PRRSUj>
```

5. **RSU stores** in a **tamper-proof device** and **publishes** its public key.

- Prime order: `q = 23`, Generator: `g = 5`
- TA's master secret: `s = 6`
- RSU ID = `"RSU1"`
- Location Code = `"ZoneA"`
- Hash → `h(s || IDRSUj || LCj) = 17`

TA assigns:

- Pseudo-ID: `PRIDj = 17`
- Private Key (chosen randomly): `PRRSUj = 4`
- Public Key: `P URSUj = g^PRRSUj mod q = 5^4 mod 23 = 625 mod 23 = 4`

Certificate:

```ini
CertRSUj = <17, "ZoneA", 4, SIGPRTA(4, 17, "ZoneA")>
```

Sent Securely to RSU:

```
<CertRSUj, CertGLP, PRRSUj = 4>
```

📌 RSU stores all this in its **tamper-proof device** and **publishes** its public key for others to verify.

✅ Slide 1: Vehicle Registration – Explanation

Overview of the Registration Process:

- Each vehicle submits **ID proof** securely to the **Trusted Authority (TA)**.
- After one-time verification, TA issues a **Tamper-Proof Device (TPD)** (or **Hardware Security Module**).
- TA generates:
  - **Pseudo-identity**: `P_IDVi = h(s || IDVi)`
  - **TPD Activation Password**: `PWVi`
  - **Random number**: `v ∈ Z\*_p`
  - **Private Key**: `PRVi = g^v`
  - **Public Key**: `PUVi = PRVi^s = g^{vs}`
  - **Public Key Certificate**:

    `CertVi = <P_IDVi, PUVi, SIGPRTA(T || P_IDVi || PUVi)>`

Final Output (Sent to Vehicle securely):

`<P_IDVi, PWVi, PUVi, PRVi, CertVi>` along with the TPD.

🔢 Slide 2: Vehicle Registration – Numerical Example

Assume:

- `IDVi = "KA01AB1234"`
- `s = 9`, `v = 4`, `g = 2`
- `h()` is a hash function (e.g., SHA-256 simplified for illustration)

Step-by-step:

1. `P_IDVi = h(9 || "KA01AB1234") = h("9KA01AB1234") → e.g., "a4c1..."`
2. `PRVi = g^v = 2^4 = 16`
3. `PUVi = PRVi^s = 16^9 = 68719476736`
4. `CertVi = <"a4c1...", 68719476736, SIGPRTA(...)>`

Delivered securely:

```
<"a4c1...", PWVi, 68719476736, 16, CertVi> + TPD
```

Navigation Scheme

- Real-time route discovery is challenging in urban traffic due to dynamic conditions.
- AI/ML-based sub-optimal solutions exist, but secure communication is critical.
- Goal: Find safe, optimal routes with minimal delay, fuel, and risk.
- RSUs & GLP assist vehicles by:
  - Authenticating queries
  - Finding static + dynamic routes
  - Ensuring privacy and robustness

RSU and GLP Role in Navigation

1. 🚗 Vehicle → RSU: Sends (source, destination) query.

2. ✅ RSU authenticates vehicle & forwards query to GLP.

3. 📡 GLP:

   - Uses Dijkstra's algorithm to compute route options.

   - Maps zones to RSUs.

   - Sends ACK to original RSU (route not revealed).

4. 📍 GLP → Neighboring RSUs: Sends route sketch to start dynamic route discovery.

5. 📶 Each RSU:

   - Caches route

   - Forwards to next RSU if traffic/weather is favorable

   - Drops otherwise (session time-out triggers retry)

Route Finalization & Vehicle Reply

6. 🔄 RSUs forward query hop-by-hop until reaching destination RSU.
7. 🎯 Destination RSU:
   - Sends reply backward through upstream RSUs
   - Reaches GLP
8. 🧠 GLP selects reply with lowest hop count
9. 🚦 GLP → Original RSU → Vehicle:
   - Sends final route
   - If δT timeout occurs → Vehicle retries via same/new RSU

✅ Ensures secure, traffic-aware, efficient navigation

Authentication and Navigation

Vehicle – RSU Communication

- Step 1:

- Vehicle Vi receives beacon from RSU1 (R1)
- Sends message M1 to R1:

M1 = <PRID1, CertVi, T, NQ, SIGPRVi(h(PRID1 || NQ || T))>

- PRID1: RSU pseudonym

- CertVi: Vehicle certificate issued by TA

- T: Timestamp

- NQ: Navigation query (initially null)

- Step 2:

- RSU1 verifies CertVi using TA’s public key
- If valid, uses PUVi (from CertVi) to verify SIGPRVi
- On success: identifies a navigation query request from Vi

RSU1 Responds to Vehicle

- Step 3:

- RSU1 generates a session-ID (SND)
- Encrypts (SND, CertGLP) using PUVi

- Sends Message M2:

M2 = <PRID1, PIDVi, CertR1, T,

SIGPRR1(PRID1, PIDVi, T),

ENCPUVi(SND, CertGLP)>

- Step 4:

- Vehicle verifies SIGPRR1 and timestamp T
- Decrypts encrypted part to get SND and CertGLP
- Prepares navigation query NQ = <S, D>

- Sends Message M3:

M3 = <PIDVi, PRID1, T, ENCPUGLP(PIDVi, PRID1, SND, NQ),

SIGPRVi(h(PIDVi, PRID1, T))>

RSU1 → GLP Communication

- Step 1:

- RSU1 verifies SIGPRVi (can't decrypt NQ)
- Forwards encrypted NQ to GLP

- Message M4:

M4 = <PRID1, CertR1, PIDGLP, T,

ENCPUGLP(PIDVi, PRID1, SND, NQ),

SIGPRR1(h(PRID1, PIDGLP, T))>

- Step 2:

- GLP verifies CertR1, checks timestamp freshness
- Decrypts with its private key to get <PIDVi, PRID1, SND, NQ>
- Sends acknowledgment to RSU1

GLP ⤐ RSUs (Path Discovery

- GLP finds possible routes for NQ = <S, D>:

RT1 = R1→R2→R6→R7→R11

RT2 = R1→R5→R6→R10→R11

RT3 = R1→R5→R9→R10→R11

RT4 = R1→R5→R6→R7→R11

- Sends to next-hop RSUs:

- To R2:

ENCPUR2(NQ, RT1, T, SID, CertRSU1), SIGPRGLP(h(NQ, RT1, T, SID))

- To R5:

ENCPUR5(NQ, RT2, RT3, RT4, T, SID, CertGLP), SIGPRGLP(h(...))

- GLP stores map(SND, SID), NQ, vehicle & RSU IDs

RSU5 ⤀ RSU6 & RSU9

- R2 detects traffic jam → remains silent

- R5 is active:

• Sends to R6:

ENCPUR6(NQ, RT2, RT4, T, SID, CertRSU5),

SIGPRR5(h(NQ, RT2, RT4, T, SID))

• Sends to R9:

ENCPUR9(NQ, RT3, T, SID, CertRSU5),

SIGPRR5(h(NQ, RT3, T, SID))

- R5 stores <NQ, SID, R6, R9>

RSU6 ↣ RSU7 & RSU10

- R9 has traffic jam → silent

- R6 forwards:

• To R7:

ENCPUR7(NQ, RT4, T, SID, CertRSU6),

SIGPRR6(h(...))

• To R10:

ENCPUR10(NQ, RT2, T, SID, CertRSU6),

SIGPRR6(h(...))

- Stores <NQ, SID, R7, R10>

RSU10 ⤐ RSU11 and Backpropagation Begins

- R7 has traffic jam → silent

- R10 sends to R11:

ENCPUR11(NQ, RT2, T, SID, CertR10),

SIGPRR10(h(...))

- R11 detects it's the destination:

- Sends back to R10:

<ENCPUR10(T, SID, R11), h(T||SID||R11)>

Route Reply (Backtracking Path)

- R10 → R6:

ENCPUR6(T, SID, R10, R11), h(...)

- R6 → R5:

ENCPUR5(T, SID, R6, R10, R11), h(...)

- R5 → GLP:

ENCPUGLP(T, SID, R5, R6, R10, R11), h(...)

Final Route Response to Vehicle

- GLP matches <NQ, SID> and retrieves old SND
- Constructs final navigation reply:

NR = <R1, R5, R6, R10, R11>

NRE = ENCPUVi(SND, NQ, NR)

- Sends:

<PRID1, PIDVi, CertGLP, NRE>,

SIGPRGLP(PRID1, PIDVi, SND)

RSU1 ➳ Vehicle (Final Delivery)

- 🔹 RSU1 verifies signature, matches SND
- 🔹 Sends to Vi:

RM = <PRID1, PIDVi, SND, NRE>,

SIGPRR1(h(PRID1, PIDVi, SND, NRE))

- 🔹 Vehicle:

- Matches SND
- Decrypts NRE with its private key
- Extracts NQ and NR (Navigation Route)

TABLE II: Performance Comparison

Features	**VSPN[11]**	**SPNS[13]**	**PiSim[14]**	**EPNS[16]**	**Ours**
Computation Cost	High	High	High	Low	Low
Cryptography	ECC, Bilinear Pairing	AES, ElGamal, Bilinear Pairing	Bilinear Pairing	Secure Multiparty Computation	ElGamal
Group signature	Yes	Yes	Yes	No	No
Crowdsourcing	No	Yes	Yes	No	No
Fog computing	No	Yes	No	No	No
Message Forwarding	point-to-point	broadcast	broadcast	broadcast	point-to-point
Network Traffic	Low	High	High	High	Low

Thank you