Local Self-Organization with Strong Privacy Protection

Lucjan Hanzlik, Kamil Kluczniak, Mirosław Kutyłowski, Shlomi Dolev

Wrocław University of Science and Technology, Poland
Ben-Gurion University, Israel

IEEE Trustcom 2016, Tianjin, China
Applications

- Virtual brake lights
- Traffic information systems
- Virtual traffic lights
- and many more...
Authentication in VANET

Seclusiveness - sending fraudulent signals or forging on-board Units (Virtual Vehicle). Only a legal manufacturer can issue new On-Board Units.
Authentication in VANET

Seclusiveness - sending fraudulent signals or forging on-board Units (Virtual Vehicle). Only a legal manufacturer can issue new On-Board Units.

Unforgeability - impersonating another vehicle.
Authentication in VANET

Threats for Authentication to VANET

- **Seclusiveness** - sending fraudulent signals or forging on-board Units (Virtual Vehicle). Only a legal manufacturer can issue new On-Board Units.
- **Unforgeability** - impersonating another vehicle.
- **Privacy/Pseudonymity** - vehicles appear under different pseudonyms at each location/time.
Authentication in VANET

Threats for Authentication to VANET

- **Seclusiveness** - sending fraudulent signals or forging on-board Units (Virtual Vehicle). Only a legal manufacturer can issue new On-Board Units.
- **Unforgeability** - impersonating another vehicle.
- **Privacy/Pseudonymity** - vehicles appear under different pseudonyms at each location/time.
- **Accountability** - Deanonymization in case of misbehaviour and undeniability of ones actions.
Local Self -Organization (Virtual Traffic Lights)

Goal: Establish an ordering of vehicles.

- Participants should not have any advantage above others.
- Clone detection.
Local Self -Organization (Virtual Traffic Lights)

Goal: Establish an ordering of vehicles.

- Participants should not have any advantage above others.
- Clone detection.

Existing solutions (Leader election)

Run a leader election protocol → The leader decides the ordering → requires Honest Majority.
Consider \( n \) participating vehicles on a crossroad at location \( \text{location} \) \textit{at} time \( \text{time} \).
Consider $n$ participating vehicles on a crossroad at location location at time time.

- Each vehicle has a private key $sk$ and a certificate $cert$ on it.
Consider $n$ participating vehicles on a crossroad at location \textit{location} at \textit{time}.

- Each vehicle has a private key $sk$ and a certificate $cert$ on it.
- A vehicle broadcasts his pseudonym $nym \leftarrow (H(\text{location}) \cdot H(\text{time}))^{sk}$ - \textit{privacy}.
Consider \( n \) participating vehicles on a crossroad at location \( \text{location} \) at time \( \text{time} \).

- Each vehicle has a private key \( sk \) and a certificate \( cert \) on it.
- A vehicle broadcasts his pseudonym
  \[
  \text{nym} \leftarrow (H(\text{location}) \cdot H(\text{time}))^{sk} - \text{privacy}.
  \]
  - It is infeasible to link the pseudonyms with a particular user.
Consider $n$ participating vehicles on a crossroad at location $\text{location}$ at time $\text{time}$.

- Each vehicle has a private key $sk$ and a certificate $cert$ on it.
- A vehicle broadcasts his pseudonym $nym \leftarrow (H(\text{location}) \cdot H(\text{time}))^{sk}$ - privacy.
  - It is infeasible to link the pseudonyms with a particular user.
- A vehicle signs the location, time and additional data - accountability.
Consider $n$ participating vehicles on a crossroad at location \textit{location} at time \textit{time}.

- Each vehicle has a private key $sk$ and a certificate $cert$ on it.
- A vehicle broadcasts his pseudonym $nym \leftarrow (H(\text{location}) \cdot H(\text{time}))^{sk}$ - privacy.
  - It is infeasible to link the pseudonyms with a particular user.
- A vehicle signs the location, time and additional data - accountability.

The signature proofs that:
  - the signer knows the secret key - unforgeability
Consider $n$ participating vehicles on a crossroad at location $\text{location}$ at time $\text{time}$.

- Each vehicle has a private key $sk$ and a certificate $\text{cert}$ on it.
- A vehicle broadcasts his pseudonym $nym \leftarrow (H(\text{location}) \cdot H(\text{time}))^{sk}$ - privacy.
  - It is infeasible to link the pseudonyms with a particular user.
- A vehicle signs the location, time and additional data - accountability.

The signature proofs that:

- the signer knows the secret key - unforgeability
- the secret key has a valid certificate - seclusiveness
Determining the ordering

Example

1. Sort the pseudonyms lexicographically and hash:

   \[ \text{seed} \leftarrow H(nym_0 | nym_1 | \ldots nym_{n-1} | location | time). \]
Determining the ordering

Example

1. Sort the pseudonyms lexicographically and hash:
   \[ seed \leftarrow H(nym_0 \| nym_1 \| \ldots nym_{n-1} \| location \| time) \].

2. For \( i = 0 \) to \( n \): the \( next \leftarrow i + seed \mod n \) goes first.
Determining the ordering

Example

1. Sort the pseudonyms lexicographically and hash:
   $seed \leftarrow H(nym_0|nym_1|..nym_{n-1}|location|time)$.

2. For $i = 0$ to $n$: the $next \leftarrow i + seed \mod n$ goes first.

Greedy Parties

The pseudonyms are deterministic - a user cannot derive a different pseudonym at a given time and location.
Determining the ordering

**Example**

1. Sort the pseudonyms lexicographically and hash: 
   
   seed ← $H(nym_0 || nym_1 || ... nym_{n-1} || location || time)$.

2. For $i = 0$ to $n$ the next ← $i + seed$ mod $n$ goes first.

**Greedy Parties**

The pseudonyms are deterministic - a user cannot derive a different pseudonym at a given time and location - he would break seclusiveness or unforgeability.
Determining the ordering

Example

1. Sort the pseudonyms lexicographically and hash:
   
   \[ seed \leftarrow H(nym_0\|nym_1\|..nym_{n-1}\|location\|time) \].
2. For \( i = 0 \) to \( n \): the \( next \leftarrow i + seed \mod n \) goes first.

Greedy Parties

The pseudonyms are deterministic - a user cannot derive a different pseudonym at a given time and location - he would break seclusiveness or unforgeability.

Unlinkability of pseudonyms - Decisional Diffie-Hellman

\[ (H(location-1) \cdot H(time))^{sk} = (h_1 \cdot H(time))^{sk} \quad \text{and} \quad (H(location-2) \cdot H(time))^{sk} = (h_2 \cdot H(time))^{sk} \]
Adding Deanonymisation and Tracing Capabilities

Deanonymization/Opening

- The signature contains also an encryption of the users identity: $ID \leftarrow \hat{h}^{sk}$.
Deanonymization/Opening

- The signature contains also an encryption of the users identity: $ID \leftarrow \hat{h}^{sk}$.
- An Opening Authority can decrypt the identity of a misbehaving vehicle.
Adding Deanonymisation and Tracing Capabilities

**Deanonymization/Opening**
- The signature contains also an encryption of the users identity: \( ID \leftarrow \hat{h}^{sk} \).
- An **Opening Authority** can decrypt the identity of a misbehaving vehicle.

**Tracing - Protection Against Cloning**
The signature contains another encryption of a “partial identity” \( ID_p \leftarrow H(\text{time})^{sk} \).
Adding Deanonymisation and Tracing Capabilities

Deanonymization/Opening
- The signature contains also an encryption of the users identity: $ID \leftarrow \hat{h}^{sk}$.
- An Opening Authority can decrypt the identity of a misbehaving vehicle.

Tracing - Protection Against Cloning
The signature contains another encryption of a “partial identity” $ID_p \leftarrow H(\text{time})^{sk}$.
- The Tracing Authority can decrypt $ID_p$ from a signature.
Adding Deanonymisation and Tracing Capabilities

Deanonymization/Opening
- The signature contains also an encryption of the users identity: $ID \leftarrow \hat{h}^{sk}$.
- An Opening Authority can decrypt the identity of a misbehaving vehicle.

Tracing - Protection Against Cloning
- The signature contains another encryption of a “partial identity” $ID_p \leftarrow H(time)^{sk}$.
- The Tracing Authority can decrypt $ID_p$ from a signature.
- The Tracing Authority will know if a vehicle appears in different locations at the same time.
Setup($1^\lambda$): Generate bilinear groups $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where $q$ is the group order and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a type-3 pairing.
Construction Background - Pointcheval-Sanders Signatures

- **Setup(1^λ)**: Generate bilinear groups $BG = (q, G_1, G_2, G_T, e)$, where $q$ is the group order and $e : G_1 \times G_2 \rightarrow G_T$ is a type-3 pairing.

- **KeyGen(BG)**: Choose $\tilde{g} \leftarrow G_2$ and $(x, y) \leftarrow \mathbb{Z}_q$ at random.
Construction Background - Pointcheval-Sanders Signatures

- Setup($1^\lambda$): Generate bilinear groups $BG = (q, G_1, G_2, G_T, e)$, where $q$ is the group order and $e: G_1 \times G_2 \rightarrow G_T$ is a type-3 pairing.

- KeyGen($BG$): Choose $\tilde{g} \leftarrow G_2$ and $(x, y) \leftarrow \mathbb{Z}_q$ at random.
Set the private key as $sk \leftarrow (x, y)$ and the public key $pk \leftarrow (\tilde{g}, \tilde{X}, \tilde{Y}) = (\tilde{g}, \tilde{g}^x, \tilde{g}^y)$
Setup($1^λ$): Generate bilinear groups

\[ BG = (q, G_1, G_2, G_T, e) \]

where \( q \) is the group order and

\[ e : G_1 \times G_2 \rightarrow G_T \]

is a type-3 pairing.

KeyGen($BG$): Choose \( \tilde{g} \leftarrow G_2 \) and \((x, y) \leftarrow \mathbb{Z}_q\)

at random.

Set the private key as \( sk \leftarrow (x, y) \) and the public key

\[ pk \leftarrow (\tilde{g}, \tilde{X}, \tilde{Y}) = (\tilde{g}, \tilde{g}^x, \tilde{g}^y) \]

Sign($pk, sk, M$): \( A \leftarrow G_1 \) and compute \( B \leftarrow A^{x+y} \cdot M \).

Output the signature \( \sigma \leftarrow (A, B) \).
Construction Background - Pointcheval-Sanders Signatures

- **Setup($1^\lambda$)**: Generate bilinear groups $BG = (q, G_1, G_2, G_T, e)$, where $q$ is the group order and $e : G_1 \times G_2 \rightarrow G_T$ is a type-3 pairing.

- **KeyGen(BG)**: Choose $\tilde{g} \leftarrow G_2$ and $(x, y) \leftarrow \mathbb{Z}_q$ at random.
  Set the private key as $sk \leftarrow (x, y)$ and the public key $pk \leftarrow (\tilde{g}, \tilde{X}, \tilde{Y}) = (\tilde{g}, \tilde{g}^x, \tilde{g}^y)$

- **Sign(pk, sk, M)**: $A \leftarrow G_1$ and compute $B \leftarrow A^{x+y \cdot M}$. Output the signature $\sigma \leftarrow (A, B)$.

- **Verify(pk, \sigma, M)**: Check that $e(A, \tilde{X} \cdot \tilde{Y}^m) = e(B, \tilde{g})$. 
Signatures of Knowledge

We use so called Signatures of Knowledge. Example:

\[ \text{SoK}\{(\alpha, \beta) : X = g^\alpha \land Y = g^\beta \cdot h^\alpha\}(M) \]
Signatures of Knowledge

We use so called Signatures of Knowledge. Example:

\[ \text{SoK}\{(\alpha, \beta) : X = g^\alpha \land Y = g^\beta \cdot h^\alpha\}(M) \]

Schnorr signature

Public key \( X \in \mathbb{G} \) and secret key \( x \in \mathbb{Z}_q \) st. \( X = g^x \).

\[ \text{Sok}\{(\alpha) : X = g^\alpha\}(M) \]
Signatures of Knowledge

We use so called Signatures of Knowledge.

Example:

\[ \text{SoK}\{(\alpha, \beta) : X = g^\alpha \land Y = g^\beta \cdot h^\alpha \}(M) \]

Schnorr signature

Public key \( X \in \mathbb{G} \) and secret key \( x \in \mathbb{Z}_q \) st. \( X = g^x \).

\[ \text{Sok}\{(\alpha) : X = g^\alpha \}(M) \]

- Sign: Choose \( t \leftarrow \mathbb{Z}_q \), compute \( T \leftarrow g^t \), compute \( c \leftarrow H(T||M) \), compute \( s \leftarrow t + c \cdot x \). The signature on \( M \) is \( (c, s) \).
Signatures of Knowledge

We use so called Signatures of Knowledge. Example:

$$\text{SoK}\{ (\alpha, \beta) : X = g^\alpha \land Y = g^\beta \cdot h^\alpha \}(M)$$

Schnorr signature

Public key $X \in \mathbb{G}$ and secret key $x \in \mathbb{Z}_q$ st. $X = g^x$.

$$\text{Sok}\{ (\alpha) : X = g^\alpha \}(M)$$

- **Sign:** Choose $t \leftarrow \mathbb{Z}_q$, compute $T \leftarrow g^t$, compute $c \leftarrow H(T||M)$, compute $s \leftarrow t + c \cdot x$. The signature on is $\{c, s\}$.

- **Verify:** Compute $\tilde{T} \leftarrow g^s \cdot X^{-c}$, check whether $c = H(\tilde{T}||M)$.
Putting Things Together

Setup

1. Run $BG = (q, G_1, G_2, G_T, e) \leftarrow \text{Setup}_{RS}$.

\footnote{For example Cramer-Shoup or ElGamal cryptosystem}
Putting Things Together

- Setup
  1. Run $BG = (q, G_1, G_2, G_T, e) \leftarrow \text{Setup}_{RS}$,
  2. Choose $h \leftarrow G_1$.

---

1 For example Cramer-Shoup or ElGamal cryptosystem
Putting Things Together

Setup

1. Run $BG = (q, G_1, G_2, G_T, e) \leftarrow \text{Setup}_{RS}$.
2. Choose $\hat{h} \leftarrow G_1$.
3. $(sk_{RS}, pk_{RS}) = ((x, y), (\tilde{g}, \tilde{X}, \tilde{Y})) \leftarrow \text{KeyGen}_{RS}(BG)$.

\footnote{For example Cramer-Shoup or ElGamal cryptosystem}
Putting Things Together

Setup

1. Run $BG = (q, G_1, G_2, G_T, e) \leftarrow \text{Setup}_{RS}$,
2. Choose $\hat{h} \leftarrow G_1$.
3. $(sk_{RS}, pk_{RS}) = ((x, y), (\tilde{g}, \tilde{X}, \tilde{Y})) \leftarrow \text{KeyGen}_{RS}(BG)$.
4. $(sk_{CS}^{trace}, pk_{CS}^{trace}) \leftarrow \text{KeyGen}_{Enc}(BG)^1$.

---

1 For example Cramer-Shoup or ElGamal cryptosystem
Putting Things Together

- **Setup**
  1. Run $BG = (q, G_1, G_2, G_T, e) \leftarrow \text{Setup}_{RS}$
  2. Choose $\hat{h} \leftarrow G_1$
  3. $(sk_{RS}, pk_{RS}) = ((x, y), (\tilde{g}, \tilde{X}, \tilde{Y})) \leftarrow \text{KeyGen}_{RS}(BG)$
  4. $(sk_{CS}^{\text{trace}}, pk_{CS}^{\text{trace}}) \leftarrow \text{KeyGen}_{Enc}(BG)^1$
  5. $(sk_{CS}^{\text{open}}, pk_{CS}^{\text{open}}) \leftarrow \text{KeyGen}_{Enc}(BG)^1$

- **Issue:**
  - The user obtains $usk = (u, \sigma) = (u, (\sigma_1, \sigma_1^{x+y\cdot u}))$.

---

1. For example Cramer-Shoup or ElGamal cryptosystem
Putting Things Together

Setup

1. Run $BG = (q, G_1, G_2, G_T, e) \leftarrow Setup_{RS}$,
2. Choose $\hat{h} \leftarrow G_1$.
3. $(sk_{RS}, pk_{RS}) = ((x, y), (\tilde{g}, \tilde{X}, \tilde{Y})) \leftarrow KeyGen_{RS}(BG)$.
4. $(sk^{trace}_{CS}, pk^{trace}_{CS}) \leftarrow KeyGen_{Enc}(BG)^1$.
5. $(sk^{open}_{CS}, pk^{open}_{CS}) \leftarrow KeyGen_{Enc}(BG)^1$.

Issue:

- The user obtains $usk = (u, \sigma) = (u, (\sigma_1, \sigma_1^{x+y\cdot u}))$.
- The issuer obtains $ID = \hat{h}^u$.

---

1 For example Cramer-Shoup or ElGamal cryptosystem
Putting Things Together

- **Setup**
  1. Run $\text{BG} = (q, G_1, G_2, G_T, e) \leftarrow \text{Setup}_{RS}$,
  2. Choose $\hat{h} \leftarrow G_1$.
  3. $(sk_{RS}, pk_{RS}) = ((x, y), (\tilde{g}, \tilde{X}, \tilde{Y})) \leftarrow \text{KeyGen}_{RS}(BG)$.
  4. $(sk_{CS}^{\text{trace}}, pk_{CS}^{\text{trace}}) \leftarrow \text{KeyGen}_{Enc}(BG)^1$.
  5. $(sk_{CS}^{\text{open}}, pk_{CS}^{\text{open}}) \leftarrow \text{KeyGen}_{Enc}(BG)^1$.

- **Issue:**
  - The user obtains $usk = (u, \sigma) = (u, (\sigma_1, \sigma_1^{x+y\cdot u}))$.
  - The issuer obtains $ID = \hat{h}^u$.

  The issue protocol does not reveal $u$, to the Issuer.

---

1 For example Cramer-Shoup or ElGamal cryptosystem
Putting Things Together

\[ u sk = (u, \sigma) = (u, (\sigma_1, \sigma_1^{x+y}u)) \]

- NymGen(\(usk, location, time\)):
  1. output \(nym \leftarrow (H_1(location) \cdot H_2(time))^u\).
Putting Things Together

\[ usk = (u, \sigma) = (u, (\sigma_1, \sigma_1^x + y \cdot u)) \]

- NymGen(usk, location, time):
  1. output nym ← \((H_1(\text{location}) \cdot H_2(\text{time}))^u\).

- Sign(usk, nym, M):
  1. \( C_1 \leftarrow \text{Enc}_{CS}(pk_{tsk}^{cs}, H(\text{time}||\text{tracing})^u) \) and
     \( C_2 \leftarrow \text{Enc}_{CS}(pk_{osk}^{cs}, \hat{h}^u) \).
Putting Things Together

\[ usk = (u, \sigma) = (u, (\sigma_1, \sigma_1^{x+y} \cdot u)) \]

- **NymGen(usk, location, time):**
  1. Output \( nym \leftarrow (H_1(location) \cdot H_2(time))^u \).

- **Sign(usk, nym, M):**
  1. \( C_1 \leftarrow \text{Enc}_{CS}(pk^{tsk}_{cs}, H(time||tracing)^u) \) and \( C_2 \leftarrow \text{Enc}_{CS}(pk^{osk}_{cs}, \hat{h}^u) \).
  2. Compute the following Signature of Knowledge:
     \[
     \pi \leftarrow \text{SoK}\{(\alpha, \beta, \gamma) : \begin{align*}
     & C_1 = \text{Enc}_{CS}(pk^{tsk}_{cs}, H_2(time||tracing)^\alpha) \land \\
     & C_2 = \text{Enc}_{CS}(pk^{osk}_{cs}, \hat{h}^\alpha) \land \\
     & nym = (H_1(location) \cdot H_2(time))^\alpha \land \\
     & e(\beta, \tilde{X} \cdot \tilde{Y}^\alpha) = e(\gamma, \tilde{g})(M)
     \end{align*}\}
     \]
  3. Verify the signature of knowledge \( \pi \).
**Putting Things Together**

\[ usk = (u, \sigma) = (u, (\sigma_1, \sigma_1^x + y \cdot u)) \]

- **NymGen**(usk, location, time):
  1. output \( nym \leftarrow (H_1(\text{location}) \cdot H_2(\text{time}))^u \).

- **Sign**(usk, nym, M):
  1. \( C_1 \leftarrow \text{Enc}_{CS}(pk_{cs}^{tsk}, H(\text{time}||\text{tracing})^u) \) and \( C_2 \leftarrow \text{Enc}_{CS}(pk_{cs}^{osk}, \hat{h}^u) \).
  2. Compute the following Signature of Knowledge:

\[
\pi \leftarrow \text{SoK}\{(\alpha, \beta, \gamma) : \quad C_1 = \text{Enc}_{CS}(pk_{cs}^{tsk}, H_2(\text{time}||\text{tracing})^\alpha) \land \\
C_2 = \text{Enc}_{CS}(pk_{cs}^{osk}, \hat{h}^\alpha) \land \\
nym = (H_1(\text{location}) \cdot H_2(\text{time}))^\alpha \land \\
e(\beta, \tilde{X} \cdot \tilde{Y}^\alpha) = e(\gamma, \tilde{g})\} (M)\]

- **Verify**
  1. Verify the signature of knowledge \( \pi \).
Tracing and Opening

Tracing

Given signatures \((C_1, C_2, nym, \pi)\) and \((C_1', C_2', nym', \pi')\):

1. The tracer decrypts
   \[ u \leftarrow \text{Dec}(sk_{tskCS}, C_1) \]
   and
   \[ u' \leftarrow \text{Dec}(sk_{tskCS}, C_1') \]

2. Check whether
   \[ H(\text{time}||\text{tracing})u = H(\text{time}||\text{tracing})u' \]

Note that if \(\text{time}\) is different for both ciphertext, the identifiers are unlinkable.
Tracing and Opening

### Tracing

Given signatures \((C_1, C_2, nym, \pi)\) and \((C'_1, C'_2, nym', \pi')\):

1. The tracer decrypts

\[
\begin{align*}
\text{H(t)} & \text{ime||tracing}^u \leftarrow \text{Dec}(sk_{CS}^{tsk}, C_1) \text{ and} \\
\text{H(t)} & \text{ime||tracing}^{u'} \leftarrow \text{Dec}(sk_{CS}^{tsk}, C'_1)
\end{align*}
\]
Tracing and Opening

### Tracing

Given signatures \((C_1, C_2, nym, \pi)\) and \((C'_1, C'_2, nym', \pi')\):

1. The tracer decrypts
   
   \[
   H(\text{time} \parallel \text{tracing})^u \leftarrow \text{Dec}(sk^{tsk}_{CS}, C_1) \quad \text{and} \quad H(\text{time} \parallel \text{tracing})^{u'} \leftarrow \text{Dec}(sk^{tsk}_{CS}, C'_1)
   \]

2. Check whether
   
   \[
   H(\text{time} \parallel \text{tracing})^u = H(\text{time} \parallel \text{tracing})^{u'}.
   \]
Tracing and Opening

**Tracing**

Given signatures \((C_1, C_2, nym, \pi)\) and \((C'_1, C'_2, nym', \pi')\):

1. The tracer decrypts
   \[
   H(\text{time}||\text{tracing})^u \leftarrow \text{Dec}(sk_{tsk}^{CS}, C_1) \quad \text{and} \\
   H(\text{time}||\text{tracing})^{u'} \leftarrow \text{Dec}(sk_{tsk}^{CS}, C'_1)
   \]

2. Check whether
   \[
   H(\text{time}||\text{tracing})^u = H(\text{time}||\text{tracing})^{u'}. \]

Note that if \(\text{time}\) is different for both ciphertext the identifiers are unlinkable.
Tracing and Opening

**Tracing**

Given signatures \((C_1, C_2, nym, \pi)\) and \((C'_1, C'_2, nym', \pi')\):

1. The tracer decrypts
   \[
   H(\text{time} \| \text{tracing})^u \leftarrow \text{Dec}(sk^{tsk}_{CS}, C_1) \quad \text{and} \quad H(\text{time} \| \text{tracing})^{u'} \leftarrow \text{Dec}(sk^{tsk}_{CS}, C'_1)
   \]

2. Check whether
   \[
   H(\text{time} \| \text{tracing})^u = H(\text{time} \| \text{tracing})^{u'}.
   \]

Note that if \(\text{time}\) is different for both ciphertext the identifiers are unlinkable.

**Opening**

Given a signature \((C_1, C_2, nym, \pi)\):

1. Decrypt the identity
   \[
   ID = \hat{h}^u = \text{Dec}(sk^{osk}_{CS}, C_2)
   \]
Conclusions

- We introduced 2D-Traceable Domain Signatures
Conclusions

- We introduced 2D-Traceable Domain Signatures
- It is a solution for VANET authentication:
  - Privacy
  - Accountability/Unforgeability
  - Seclusiveness
  - Clone detection
Conclusions

- We introduced 2D-Traceable Domain Signatures
- It is a solution for VANET authentication:
  - Privacy
  - Accountability/Unforgeability
  - Seclusiveness
  - Clone detection
- Pseudonyms are deterministic and a user cannot change his pseudonym at will.
Conclusions

- We introduced 2D-Traceable Domain Signatures
- It is a solution for VANET authentication:
  - Privacy
  - Accountability/Unforgeability
  - Seclusiveness
  - Clone detection
- Pseudonyms are deterministic and a user cannot change his pseudonym at will.
- Solution for Virtual Traffic Lights - **honest majority is not required.**
Conclusions

- We introduced 2D-Traceable Domain Signatures
- It is a solution for VANET authentication:
  - Privacy
  - Accountability/Unforgeability
  - Seclusiveness
  - Clone detection
- Pseudonyms are deterministic and a user cannot change his pseudonym at will.
- Solution for Virtual Traffic Lights - *honest majority is not required*.
- No need to build an expensive PKI infrastructure.
Conclusions

- We introduced 2D-Traceable Domain Signatures
- It is a solution for VANET authentication:
  - Privacy
  - Accountability/Unforgeability
  - Seclusiveness
  - Clone detection
- Pseudonyms are deterministic and a user cannot change his pseudonym at will.
- Solution for Virtual Traffic Lights - **honest majority is not required**.
- No need to build an expensive PKI infrastructure.
- A vehicle needs to store only single key to produce multiple pseudonyms.
Questions?