Introduction

Solution Concept

Construction

Conclusions

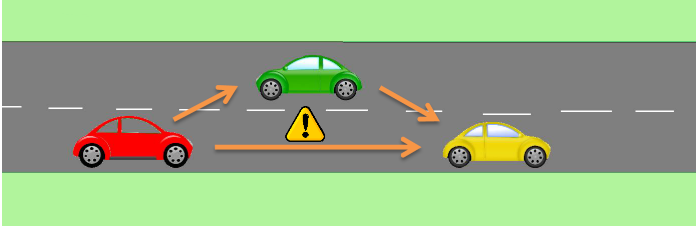
Local Self -Organization with Strong Privacy Protection

Lucjan Hanzlik, Kamil Kluczniak, Mirosław Kutyłowski, Shlomi Dolev

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

IEEE Trustcom 2016,
Tianjin, China

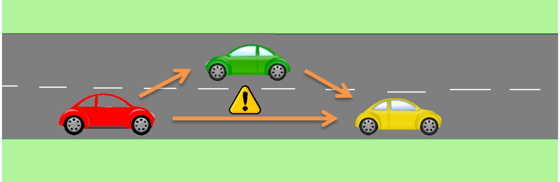
Vehicular Ad Hoc Networks (VANET)



Applications

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

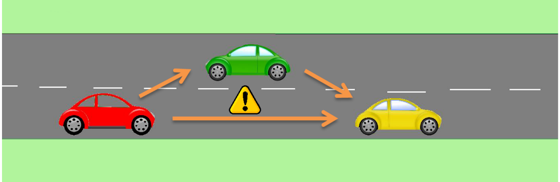
Authentication in VANET



Threats for Authentication to VANET

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

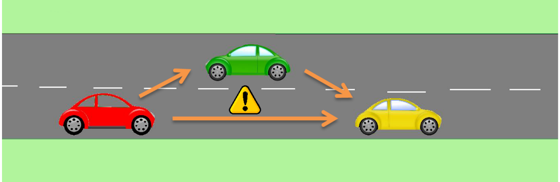
Authentication in VANET



Threats for Authentication to VANET

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

Authentication in VANET

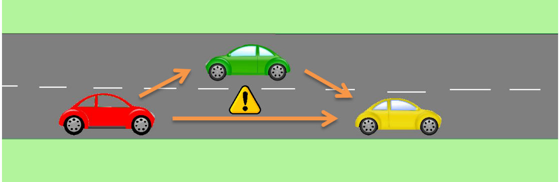
Lucjan
Hanzlik,
Kamil Kluczniak,
Mirosław
Kutyłowski,
Shlomi Dolev

Introduction

Solution
Concept

Construction

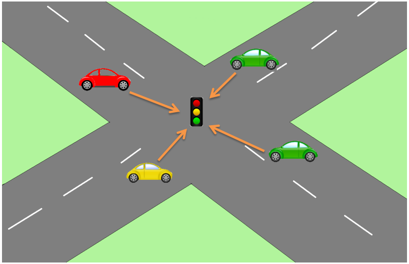
Conclusions



Threats for Authentication to VANET

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

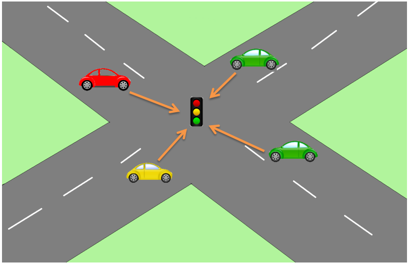
Local Self -Organization (Virtual Traffic Lights)



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.

2-D Traceable Domain Signature - Concept

Consider n participating vehicles on a crossroad at location `location` at time `time`.

2-D Traceable Domain Signature - Concept

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.

2-D Traceable Domain Signature - Concept

Consider n participating vehicles on a crossroad at location $\texttt{location}$ at time $\texttt{time}$.

- Each vehicle has a private key sk and a certificate $cert$ on it.
- A vehicle broadcasts his pseudonym $nym \leftarrow (H(\texttt{location}) \cdot H(\texttt{time}))^{sk}$ - privacy.

2-D Traceable Domain Signature - Concept

Consider n participating vehicles on a crossroad at location $\texttt{location}$ at time $\texttt{time}$.

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

2-D Traceable Domain Signature - Concept

Consider n participating vehicles on a crossroad at location $\texttt{location}$ at time $\texttt{time}$.

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

2-D Traceable Domain Signature - Concept

Consider *n* participating vehicles on a crossroad at location $\texttt{location}$ at time $\texttt{time}$.

- Each vehicle has a private key *sk* and a certificate *cert* on it.
- A vehicle broadcasts his pseudonym $nym \leftarrow (H(\texttt{location}) \cdot H(\texttt{time}))^{sk}$ - 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 knowns the secret key - unforgeability

2-D Traceable Domain Signature - Concept

Consider n participating vehicles on a crossroad at location $\texttt{location}$ at time $\texttt{time}$.

- Each vehicle has a private key *sk* and a certificate *cert* on it.
- A vehicle broadcasts his pseudonym $nym \leftarrow (H(\texttt{location}) \cdot H(\texttt{time}))^{sk}$ - 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 knowns the secret key - unforgeability
  - the secret key has a valid certificate - seclusiveness

Determining the ordering

Example

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

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.

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: $seed \leftarrow H(nym_0||nym_1||..nym_{n-1}||location||time)$.
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.

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

$(H(\texttt{location-1}) \cdot H(\texttt{time}))^{sk} = (h_1 \cdot H(\texttt{time}))^{sk}$ and
$(H(\texttt{location-2}) \cdot H(\texttt{time}))^{sk} = (h_2 \cdot H(\texttt{time}))^{sk}$

Adding Deanonymisation and Tracing Capabilities

Deanonymization/Opening

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

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.

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" $ID_p \leftarrow H(\texttt{time})^{sk}$.

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" $ID_p \leftarrow H(\mathtt{time})^{sk}$.

- 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" $ID_p \leftarrow H(\texttt{time})^{sk}$.

- 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.

Construction Background - Pointcheval-Sanders Signatures

- Setup(1^λ): Generate bilinear groups $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where q is the group order and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 pairing.

Construction Background - Pointcheval-Sanders Signatures

- Setup(1^λ): Generate bilinear groups $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where q is the group order and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 pairing.
- KeyGen(BG): Choose $\tilde{g} \leftarrow \mathbb{G}_2$ and $(x, y) \leftarrow \mathbb{Z}_q$ at random.

Construction Background - Pointcheval-Sanders Signatures

- Setup(1^λ): Generate bilinear groups $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where q is the group order and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 pairing.
- KeyGen(BG): Choose $\tilde{g} \leftarrow \mathbb{G}_2$ and $(x, y) \leftarrow \mathbb{Z}_q$ at random.
  Set the private key as $sk \leftarrow (x, y)$ and the public key $pk \leftarrow (\tilde{g}, \tilde{X}, \tilde{Y}) = (\tilde{g}, \tilde{g}^x, \tilde{g}^y)$

Construction Background - Pointcheval-Sanders Signatures

Lucjan Hanzlik, Kamil Kluczniak, Mirosław Kutyłowski, Shlomi Dolev

Introduction
Solution Concept
Construction
Conclusions

- Setup(1^λ): Generate bilinear groups $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where q is the group order and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 pairing.
- KeyGen(BG): Choose $\tilde{g} \leftarrow \mathbb{G}_2$ 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): $A \leftarrow \mathbb{G}_1$ and compute $B \leftarrow A^{x+y \cdot M}$.
  Output the signature $\sigma \leftarrow (A, B)$.

Construction Background - Pointcheval-Sanders Signatures

- Setup(1^λ): Generate bilinear groups $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where q is the group order and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 pairing.
- KeyGen(BG): Choose $\tilde{g} \leftarrow \mathbb{G}_2$ 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): $A \leftarrow \mathbb{G}_1$ and compute $B \leftarrow A^{x+y \cdot M}$.
  Output the signature $\sigma \leftarrow (A, B)$.
- Verify(pk, σ, 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:

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

Signatures of Knowledge

We use so called Signatures of Knowledge.
Example:

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

Schnorr signature

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

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

Signatures of Knowledge

We use so called Signatures of Knowledge.
Example:

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

Schnorr signature

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

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

- 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).

Signatures of Knowledge

We use so called Signatures of Knowledge.
Example:

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

Schnorr signature

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

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

- 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 $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathrm{Setup}_{RS}$,

[1] For example Cramer-Shoup or ElGamal cryptosystem

Putting Things Together

- Setup
  1. Run $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathrm{Setup}_{RS}$,
  2. Choose $\hat{h} \leftarrow \mathbb{G}_1$.

[1] For example Cramer-Shoup or ElGamal cryptosystem

Putting Things Together

- Setup
  1. Run $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathsf{Setup}_{RS}$,
  2. Choose $\hat{h} \leftarrow \mathbb{G}_1$.
  3. $(sk_{RS}, pk_{RS}) = ((x, y), (\tilde{g}, \tilde{X}, \tilde{Y})) \leftarrow \mathsf{KeyGen}_{RS}(BG)$.

[1] For example Cramer-Shoup or ElGamal cryptosystem

Putting Things Together

- Setup
  1. Run $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \text{Setup}_{RS}$,
  2. Choose $\hat{h} \leftarrow \mathbb{G}_1$.
  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 $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \text{Setup}_{RS}$,
  2. Choose $\hat{h} \leftarrow \mathbb{G}_1$.
  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].
  5. $(sk_{CS}^{open}, pk_{CS}^{open}) \leftarrow \text{KeyGen}_{Enc}(BG)$[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 $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \text{Setup}_{RS}$,
  2. Choose $\hat{h} \leftarrow \mathbb{G}_1$.
  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].
  5. $(sk_{CS}^{open}, pk_{CS}^{open}) \leftarrow \text{KeyGen}_{Enc}(BG)$[1].
- Issue:
  - The user obtains $usk = (u, \sigma) = (u, (\sigma_1, \sigma_1^{x+y \cdot u}))$.
  - The issuer obtains $ID = \hat{h}^u$.

[1] For example Cramer-Shoup or ElGamal cryptosystem

Putting Things Together

- Setup
  1. Run $BG = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \text{Setup}_{RS}$,
  2. Choose $\hat{h} \leftarrow \mathbb{G}_1$.
  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].
  5. $(sk_{CS}^{open}, pk_{CS}^{open}) \leftarrow \text{KeyGen}_{Enc}(BG)$[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

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

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

Putting Things Together

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

- NymGen(usk, $\texttt{location}$, $\texttt{time}$):
  1. output $nym \leftarrow (H_1(\texttt{location}) \cdot H_2(\texttt{time}))^u$.
- Sign(usk, nym, M):
  1. $C_1 \leftarrow \mathrm{Enc}_{CS}(pk_{cs}^{tsk}, \mathrm{H}(\texttt{time}||tracing)^u)$ and $C_2 \leftarrow \mathrm{Enc}_{CS}(pk_{cs}^{osk}, \hat{h}^u)$.

Putting Things Together

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

- NymGen(usk, $\texttt{location}$, $\texttt{time}$)):
  1. output $nym \leftarrow (H_1(\texttt{location}) \cdot H_2(\texttt{time}))^u$.
- Sign(usk, nym, M):
  1. $C_1 \leftarrow \mathrm{Enc}_{CS}(pk_{cs}^{tsk}, \mathrm{H}(\texttt{time}||tracing)^u)$ and $C_2 \leftarrow \mathrm{Enc}_{CS}(pk_{cs}^{osk}, \hat{h}^u)$.
  2. Compute the following Signature of Knowledge:

$$\pi \leftarrow SoK\{(\alpha, \beta, \gamma) :$$
$$C_1 = \mathrm{Enc}_{CS}(pk_{cs}^{tsk}, \mathrm{H}_2(\texttt{time}||tracing)^{\alpha}) \wedge$$
$$C_2 = \mathrm{Enc}_{CS}(pk_{cs}^{osk}, \hat{h}^{\alpha}) \wedge$$
$$nym = (H_1(\texttt{location}) \cdot H_2(\texttt{time}))^{\alpha} \wedge$$
$$e(\beta, \tilde{X} \cdot \tilde{Y}^{\alpha}) = e(\gamma, \tilde{g})\}(M)$$

Putting Things Together

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

- NymGen(usk, $\texttt{location}$, $\texttt{time}$)):
  1. output $nym \leftarrow (H_1(\texttt{location}) \cdot H_2(\texttt{time}))^u$.
- Sign(usk, nym, M):
  1. $C_1 \leftarrow \mathrm{Enc}_{CS}(pk_{cs}^{tsk}, \mathrm{H}(\texttt{time}||tracing)^u)$ and $C_2 \leftarrow \mathrm{Enc}_{CS}(pk_{cs}^{osk}, \hat{h}^u)$.
  2. Compute the following Signature of Knowledge:

$$\pi \leftarrow SoK\{(\alpha, \beta, \gamma) :$$
$$C_1 = \mathrm{Enc}_{CS}(pk_{cs}^{tsk}, \mathrm{H}_2(\texttt{time}||tracing)^{\alpha}) \wedge$$
$$C_2 = \mathrm{Enc}_{CS}(pk_{cs}^{osk}, \hat{h}^{\alpha}) \wedge$$
$$nym = (H_1(\texttt{location}) \cdot H_2(\texttt{time}))^{\alpha} \wedge$$
$$e(\beta, \tilde{X} \cdot \tilde{Y}^{\alpha}) = e(\gamma, \tilde{g})\}(M)$$

- Verify
  1. Verify the signature of knowledge π.

Tracing and Opening

Tracing

Given signatures (C_1, C_2, nym, π) and (C_1', C_2', nym', π'):

Tracing and Opening

Tracing

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

1. The tracer decrypts
   $\mathrm{H}(\mathtt{time}||tracing)^u \leftarrow Dec(sk_{CS}^{tsk}, C_1)$ and
   $\mathrm{H}(\mathtt{time}||tracing)^{u'} \leftarrow Dec(sk_{CS}^{tsk}, C'_1)$

Tracing and Opening

Tracing

Given signatures (C_1, C_2, nym, π) and (C_1', C_2', nym', π'):

1. The tracer decrypts $H(\mathtt{time}||tracing)^u \leftarrow Dec(sk_{CS}^{tsk}, C_1)$ and $H(\mathtt{time}||tracing)^{u'} \leftarrow Dec(sk_{CS}^{tsk}, C_1')$
2. Check whether $H(\mathtt{time}||tracing)^u = H(\mathtt{time}||tracing)^{u'}$.

Tracing and Opening

Tracing

Given signatures (C_1, C_2, nym, π) and (C_1', C_2', nym', π'):

1. The tracer decrypts $\mathsf{H}(\mathtt{time}||tracing)^u \leftarrow Dec(sk_{CS}^{tsk}, C_1)$ and $\mathsf{H}(\mathtt{time}||tracing)^{u'} \leftarrow Dec(sk_{CS}^{tsk}, C_1')$
2. Check whether $\mathsf{H}(\mathtt{time}||tracing)^u = \mathsf{H}(\mathtt{time}||tracing)^{u'}$.

Note that if `time` is different for both ciphertext the identifiers are unlinkable.

Tracing and Opening

Tracing

Given signatures (C_1, C_2, nym, π) and (C_1', C_2', nym', π'):

1. The tracer decrypts $\mathsf{H}(\mathtt{time}||tracing)^u \leftarrow Dec(sk_{CS}^{tsk}, C_1)$ and $\mathsf{H}(\mathtt{time}||tracing)^{u'} \leftarrow Dec(sk_{CS}^{tsk}, C_1')$
2. Check whether $\mathsf{H}(\mathtt{time}||tracing)^u = \mathsf{H}(\mathtt{time}||tracing)^{u'}$.

Note that if `time` is different for both ciphertext the identifiers are unlinkable.

Opening

Given a signature (C_1, C_2, nym, π):

1. Decrypt the identity $ID = \hat{h}^u = Dec(sk_{CS}^{osk}, C_2)$

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?