Goals

- Authenticate without revealing credentials
  - Consider two groups $G_1$ and $G_2$
  - Two parties $A \in G_1$ and $B \in G_2$. $A$ and $B$ wants to authenticate each other.
  - If $G_1 \neq G_2$: $A$ and $B$ only know they are not in the same group.
  - If $G_1 = G_2$: $A$ and $B$ can authenticate to each other.
  - A third party learns nothing by observing conversations between $A$ and $B$.

Preliminaries: Pairing-based Cryptography

- Bilinear Maps:
  - Two cyclic groups of large prime order $q$: $G_1$ and $G_2$
  - $\hat{\cdot}: G_1 \times G_1 \to G_2$ is a bilinear map if
    \[ \forall a, b \in \mathbb{Z}_q; P, Q \in G_1; \hat{\cdot}(aP, bQ) = \hat{\cdot}(P, Q)^{ab} \]
  - $\hat{\cdot}$ should be computable, non-degenerate and satisfies Bilinear Diffie-Hellman assumption, i.e., given $P, aP, bP, cP$, it is hard to compute $\hat{\cdot}(P, P)^{abc}$.
Protocol Sketch

- Equipped with bilinear map $\hat{e}$ and one-way hash function $H_1$
- CA has a master key $t$.
- Assume a drivers and cops scenario.

Driver's Licence: "p65748392a", $T_A = tH_1("p65748392a-driver")$

Traffic cop credential: "xy6542678d", $T_B = tH_1("xy6542678d-cop")$

Driver's licence, please. Please show me your pseudonym. $xy6542678d, p65748392a$ (driver-cop)

Caution: This guy is not a cop.

Protocol Sketch – Attacker Igor

Driver's Licence: "p65748392a", $T_A$

Obtains Bob's pseudonym "xy6542678d"
Secret-Handshake Scheme (SHS)

- **SHS.CreateGroup(G)**: executed by an administrator, generates the group secret GroupSecret for G.
- **SHS.AddUser(U,G,GroupSecret):** creates user secret UserSecret for new user U.
- **SHS.HandShake(A,B):** Users A and B authenticates each other. B discovers A \( A \in G \) if and only if A discovers B \( B \in G \).
- **SHS.TraceUser:** Administrator tells the user from a transcript \( T \) generated during conversation between A and B.
- **SHS.RemoveUser:** Administrator revokes user U.

Pairing-Based Handshake (PBH)

- **PBH.CreateGroup:** Administrator sets GroupSecret_i as a random number \( s_i \in \mathbb{Z}_q \).
- **PBH.AddUser:** Administrator generates pseudonyms for users:
  \( \{id_1, \ldots, id_n\} \)
  and then generates the corresponding secret points:
  \( \{\text{priv}_{id_1}, \ldots, \text{priv}_{id_n}\} \)
  where
  \( \text{priv}_{id_i} = s_i H_i(id_{id_i}) \)
  \( H_i \) is a one-way hash function.

Pairing-Based Handshake (PBH)

- **PBH.Handshake:**
  - \[ A \xrightarrow{id_1,n_1} B \]
  - \[ A \xrightarrow{id_2,n_2} B \]
  - \[ A \xleftarrow{V_i} B \]
  \[ V_0 = H_1(\hat{g}(H_1(id_2), \text{priv}_2)) | id_1 | n_1 | n_2 | 0 \]
  \[ V_1 = H_1(\hat{g}(\text{priv}_1,H_1(id_1))) | id_1 | id_2 | n_1 | n_2 | 1 \]
  \[ S = H_1(\hat{g}(\text{priv}_1,H_1(id_2)) | id_1 | id_1 | id_1 | n_1 | n_2 | 12 \]
  \[ = H_1(\hat{g}(H_1(id_1), \text{priv}_2)) | id_1 | id_1 | n_1 | n_2 | 12 \]
Pairing-Based Handshake (PBH)

- **PBH.TraceUser**: Since the conversations of handshaking include the pseudonyms, administrator can easily figure out the users.

- **PBH.RemoveUser**: Administrator removes user U by broadcasting its pseudonyms to all the other users, so that other users won’t accept pseudonyms of U.

Computational Diffie-Hellman Instead of Bilinear Diffie-Hellman

- **CreateGroup**: Administrator picks \((p, q, g)\). p and q are primes, g is a generator of a subgroup in \(\mathbb{Z}_p^*\) of order q. Also, picks up a private key \(x\), and computes the public key \(y = g^x \mod p\).

- **AddUser**: For user U, administrator generates id_U, then generates a pair so that \(id_U, w, t\) will be given to the user.

  - How to generate the pair \((w, t)\)?
  - Randomly pick \(r\), compute
    
    \[
    w = g^r \mod p \\
    t = r + xH(w, ID)
    \]
Computational Diffie-Hellman Instead of Bilinear Diffie-Hellman

- **Handshake:** Assume user $A$ has $(\text{id}_A, w_A, t_A)$ and user $B$ has $(\text{id}_B, w_B, t_B)$. Define several marks (ElGamal Encryption):
  
  - Recover $(y, \text{id}, w) = PK^y H(\text{id}) \mod p$
  - $\text{Enc}_{w_B}(m) = [c_1, c_2]$  
    
    $$= [g^m \mod p, m \oplus \text{H}^t(PK^t \mod p)]$$
  - $\text{Dec}_B([c_1, c_2]) = m \oplus \text{H}^t(c_1 \mod p)$

Intuition

- If $A$ and $B$ are in the same group, each of them can decrypt the random number ($r_A$ and $r_B$).

- If not, neither of them can get any information about $r_A$ or $r_B$. 