Certificateless Public Key Cryptography

Julia Miller

Public Key Encryption (PKE) Overview

- Each user has public and private key
  - Public key (A) is used to encrypt message for A; verify A’s signature
  - Private key (A) is used to decrypt messages received by A; sign messages by A

- Traditional Public Key Infrastructure (PKI)
  - Uses certificates to establish relationship between public keys and users.
  - Certificates are issued and signed by a trusted Certificate Authority (CA)
    - Generates key pairs, issues digital certificates, publishes certificates, verifies certificates, revokes certificates (Certificate Revocation List)
  - Registration Authority (RA)
    - authentication
PKE Overview

Negatives of PKI

- Certificate management
  - Revocation (CRLs)
- Certificate verification is costly
Identity-Based Public Key Cryptography (ID-PKC)

- Private Key Generator (PKG) generates all private keys.
- Public key derived from aspects of identity
- Eliminates certificates
- Can automatically revoke keys
- Must put a lot of trust in PKG:
  - PKG can decrypt any ciphertext in an identity-based public key encryption scheme
  - PKG could forge any entity’s signatures in an identity-based signature scheme
  - So, compromise of PKG master key results in disaster
CL-PKC

- Falls in between traditional PKI and ID-PKE
- Uses key generating center (KGC) instead of PKG
  - does not have access to entities’ private keys
  - Provides A with a partial private key $D_A$, which the KGC computes from an identifier $ID_A$ and a master key
  - A combines $D_A$ with secret information to create full private key $S_A$. In a similar way the public key $P_A$ is created.
- To encrypt a message to A or verify a signature from A, entity B makes use of $P_A$ and $ID_A$, no certificates required
7 Randomized Algorithms

- **Setup (KGC)**
  - Input: security parameter $k$
  - Output: system parameters $params$ and $master-key$ (only known to KGC)

- **Partial-Private-Key-Extract (KGC)**
  - Input: $params$, $master-key$, $ID_A \in \{0, 1\}^*$
  - Output: partial private key $D_A$

- **Set-Secret-Value (A)**
  - Input: $params$, $ID_A$
  - Output: A’s secret value $x_A$, chosen randomly from suitable and large set

- **Set-Private-Key (A)**
  - Input: $params$, $D_A$, $x_A$
  - Output: $S_A$, computed from $x_A$ and $D_A$
7 Randomized Algorithms Cont.

- **Set-Public-Key (A)**
  - Input: $\textit{params}$, $x_A$
  - Output: public key $P_A$, computed using only $\textit{params}$ and $x_A$

- **Encrypt**
  - Input: $\textit{params}$, message $M \in \mathcal{M}$, $P_A \text{ ID}_A$
  - Output: if $P_A$ does not have correct form, $\perp$. Otherwise, ciphertext $C \in \mathcal{C}$

- **Decrypt**
  - Input: $\textit{params}$, $C \in \mathcal{C}$, private key $S_A$
  - Output: if decryption fails, $\perp$. Otherwise, $M \in \mathcal{M}$
Security

- Based on bilinear maps
- Bilinear Diffie-Hellman Problem and Generalized Bilinear Diffie-Hellman Problem are hard
  - no polynomial-time algorithm for solving BDHP or GBDHP
  - Uses map derived from either the Weil or Tate pairing on an elliptic curve over a finite field
- IND-CCA Secure
  - Proof can be found in paper
Summary

- No need to place all trust in KGC as you do in PKG
- Security vulnerability of KGC is about the same as certificates - not losing anything here
- No certificates - less costly
- Secure: Generalized Bilinear Diffie-Hellman Problem (GBDHP) is hard
- NOTE: Secure channel is needed between KGC and A