Strong forward security in signcryption schemes

Madeline González Muñiz

What is Signcryption?

- Sign-then-Encrypt for authenticity and confidentiality
- Combine both and achieve efficiency with signcryption primitive
- Goal: Achieve confidentiality, unforgeability, and non-repudiation
- Setup has sender and receiver
  - Let Alice be the sender
  - Let Bob be the receiver
  - Let Eve be an adversary

Scheme Setup

- Signcryption scheme **SC**={K, SE, VD}, **U**={SK_U, PK_U}
- Correctness $VD_B(c)=m$ for $c=SE_A(m)$
  - Alice computes $SE_A(m)$ using SK_A, PK_B
  - Bob computes $VD_B(c)$ using SK_B, PK_A

Outsider Security

- Eve has access to the public keys of Alice and Bob
- EUF-CMA
  - Eve has to come up with a valid signcryption of new message m that Bob designcrypts successfully
- IND-CCA
  - Eve chooses two messages, one of which is signcrypted and Eve must guess which one

Insider Security

- Induced signature scheme from **SC**
  - Signing key is SK_A, PK_B
  - Verification key is SK_B, PK_A
    - Make SK_B public
- Induced encryption scheme from **SC**
  - Encryption Key is SK_A, PK_B
    - Make SK_A public
  - Decryption Key is SK_B, PK_A
- Apply EUF-CMA, IND-CCA security definitions to induced schemes

Bilinear Map

- Let G_1 (additive) and G_2 (multiplicative) be two groups of prime order q and e: $G_1 X\ G_1 \rightarrow G_2$ a bilinear map:

1. Bilinear: for all P, Q in G_1, for all a, b in Z_q^*, $e(aP, bQ)= e(P, Q)^{ab}$
2. Non-degenerate: for any P in G_1, $e(P, Q)=1$ for all Q in G_1 iff $P=0$
3. Computable: there exists an efficient algorithm to compute $e(P, Q)$ for all P, Q in G_1

Bilinear Diffie-Hellman Problem

- For G_1, G_2, q, and e as described in the previous slide, and a generator P of G_1, the BDHP is to computer $e(P, P)^{abc}$ given (P, aP, bP, cP).
- The decisional bilinear Diffie-Hellman Problem is to decide whether $h = e(P, P)^{abc}$ for given (P, aP, bP, cP) and h in G_2.

Identity-Based Cryptography

- User identifier information serves as a public key (such as email address)
- Trusted third party is the Private Key Generator
- Using the public key of the PKG and Bob’s identity, Alice can encrypt to Bob
- Using the public key of the PKG and Alice’s identity, Bob can verify signatures from Alice

Example: Signcryption Scheme

Let $H_1: \{0, 1\}^* \rightarrow G_1$, $H_2: G_2 \rightarrow \{0, 1\}^n$, $H_3: \{0, 1\}^* \times G_2 \rightarrow Z_q$

$P_{pub}=sP$, $Q_{ID}= H_1(ID)$, $d_{ID}=sQ_{ID}$

Signcrypt : to send a message m to Bob, Alice follows the steps below	**Unsigncrypt :** when receiving $\sigma = (c, r, S)$, Bob performs the following tasks
1. Compute $Q_{ID_B} = H_1(ID_B) \in \mathbb{G}_1$. 2. Choose $x \leftarrow_R \mathbb{F}_q^*$, and compute $k_1 = \hat{e}(P, P_{pub})^x$ and $k_2 = H_2(\hat{e}(P_{pub}, Q_{ID_B})^x)$. 3. Compute $c = E_{k_2}(m)$, $r = H_3(c, k_1)$ $S = xP_{pub} - rd_{ID_A} \in \mathbb{G}_1$. The ciphertext is $\sigma = (c, r, S)$.	1. Compute $Q_{ID_A} = H_1(ID_A) \in \mathbb{G}_1$ 2. Compute $k_1 = \hat{e}(P, S)\hat{e}(P_{pub}, Q_{ID_A})^r$ 3. Compute $\tau = \hat{e}(S, Q_{ID_B})\hat{e}(Q_{ID_A}, d_{ID_B})^r$ and $k_2 = H_2(\tau)$. 4. Recover $m = D_{k_2}(c)$ and accept σ if and only if $r = H_3(c, k_1)$.

Public Ciphertext Verifiability

- Why not public message verifiability?
- In the IND-CCA game, Eve selects messages m_1 and m_2.
- Eve gets a signcryption of m_1 or m_2
- With public message verifiability, Eve can check whether m_1 or m_2 was signcrypted, thus distinguishing
- For non-repudiation of message m, use zero-knowledge proof

Forward Security

- Forward security in encryption schemes
  - Compromise of the current secret key does not allow an adversary to decrypt in the past
- Forward security in signature schemes
  - Compromise of the current secret key does not allow an adversary to forge in the past
- Strong forward security in signcryption achieves both

A Forward-Secure Public-Key Encryption Scheme, Canetti et al. 2007
A Forward-Secure Digital Signature Scheme, Bellare and Miner 1999

Example: Signcryption Scheme with Encryption Forward Security

- Note that only Bob can recover $e(R, d_B)$ since Alice cannot recover r from R.

Authenticrypt : to send a message m to Bob, Alice follows the steps below	**Authentidecrypt :** when receiving $\sigma = (c, R, S)$, Bob performs these tasks
1. Compute $Q_{ID_B} = H_1(ID_B) \in \mathbb{G}_1$. 2. Choose $x \leftarrow_R \mathbb{F}_q^*$, and compute $(k_1, k_2) = H_2(\hat{e}(P_{pub}, Q_{ID_B})^x)$. 3. Compute $c = E_{k_2}(m)$, $r = H_3(c, k_1)$, $S = xP_{pub} - rd_{ID_A} \in \mathbb{G}_1$ and $R = rQ_{ID_A}$. The ciphertext is $\sigma = (c, R, S)$.	1. Compute $Q_{ID_A} = H_1(ID_A) \in \mathbb{G}_1$. 2. Compute $\tau = \hat{e}(S, Q_{ID_B})\hat{e}(R, d_{ID_B})$ and $(k_1, k_2) = H_2(\tau)$. 3. Recover $m = D_{k_2}(c)$. 4. Compute $r = H_3(c, k_1)$ and rQ_{ID_A}. 5. Accept σ if and only if $R = rQ_{ID_A}$.

A Forward-Secure Public-Key Encryption Scheme, Canetti et al. 2007
A Forward-Secure Digital Signature Scheme, Bellare and Miner 1999

Non-Interactive Key Exchange

- Using pairings Alice and Bob can non-interactively compute a secret key in the identity-based setting:
  - Alice computes $e(d_A, Q_B)$
  - Bob computes $e(Q_A, d_B)$
  - $e(d_A, Q_B)=e(sQ_A, Q_B)=e(Q_A, Q_B)^s=e(Q_A, sQ_B)=e(Q_A, d_B)$
- If the state of either Alice or Bob is compromised, the shared secret key is also compromised

Partial Forward Secret Key Exchange

- Alice does the following:
  - Computes Q_B
  - Randomly selects x in Z_q^*
  - $k_t = H_2(e(P_{pub}, Q_B)^x)$
  - Sets $r = H_2(t, k_t)$, $R = rQ_A$, and $S = xP_{pub} - rd_A$
  - Sends Bob (t, R, S)
- If the sender's state becomes compromised, still secure because Alice cannot recover r given R

Partial Forward Secret Key Exchange

- Bob can do the same for randomly selected x' in Z_q^* and (t, R', S') where $R'=r'Q_B$
- Then the agreed key becomes $k_t=e(rd_A, R')=e(R, r'd_B)$
- If either the sender or receiver's state becomes compromised, still secure because Alice cannot recover r given R *and Bob cannot recover* r' given R'
- What if both states are compromised?

Partial Forward Secret Key Exchange

- What if both states are compromised?
- Alice needs to encrypt to Bob in a way that he cannot decrypt past messages, and vice versa
- We need forward-secure encryption

Forward-Secure Encryption

- Recall that compromise of the current secret key does not allow an adversary to decrypt in the past
- Decryption keys are generated via one-way functions in such a way that even the rightful user cannot decrypt in the past
- Seed used to generate the decryption keys may be stored in a separate/secure device to recover decryption keys from past periods
- Canetti et al. make use of binary trees to create a forward-secure encryption scheme

Strong Forward-Secure Signcryption

- Use the concept of a binary tree so that each user updates their secret key
- Alice and Bob can signcrypt messages to each other in such a way that an adversary cannot decrypt nor forge in the past

Küsimused?