EE817/IS 893

Cryptography Engineering and Cryptocurrency

Yongdae Kim

한국과학기술원

Admin Stuff

- Schedule
  - Make up class: May 28, 30, 9:30 AM - 12:00 PM?

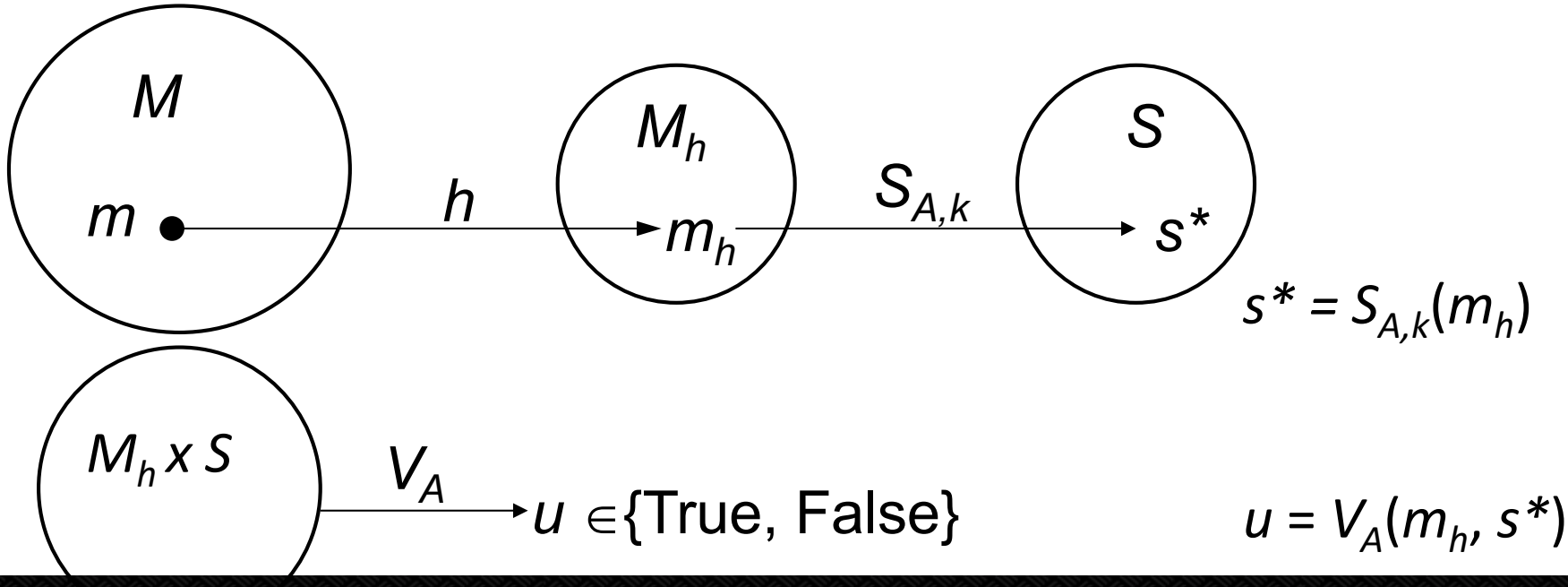
Digital Signature

Digital Signature

- Integrity
- Authentication
- Non-repudiation

Digital Signature with Appendix

- Schemes with appendix
  - Requires the message as input to verification algorithm
  - Rely on cryptographic hash functions rather than customized redundancy functions
  - DSA, ElGamal, Schnorr etc.



Desirable Properties

- For each $k \in R$, $S_{A,k}$ should be efficient to compute
- V_A should be efficient to compute
- It should be computationally infeasible for an entity other than the *signer* to find an $m \in M$ and an $s^* \in S$ such that $V_A(m', s^*) = true$, where $m' = h(m)$

Types of Attacks

- Key-only: adversary knows only the public key

- Message attacks
  - known-message attack: adversary has signatures for a set of messages which are known to the adversary but not chosen by him
  - chosen-message attack: adversary obtains valid signatures from a chosen list of his choice (non adaptive)
  - Adaptive chosen-message attack: adversary can use the signer as an oracle

RSA Signature

- Key generation n, p, q, e, d
- Sign
  - compute $s = h(m)^d \bmod n$
  - Signature: (m, s)
- Verify
  - Obtain authentic public key (n, e)
  - verify $h(m) = s^e \bmod n$

DSA (US Standard)

- DSA Algorithm : key generation
  1. Select a prime q of 160 bits
  2. 1024 bit p with $q|p-1 \Rightarrow p = kq+1$
  3. Select g' in Z_p^*, and $g = g'^k = g'^{(p-1)/q} \bmod p$, $g \neq 1$
  4. Select $1 \leq x \leq q-1$, compute $y = g^x \bmod p$
  5. public key (p, q, g, y), private key x

DSA (cont)

- DSA signature generation
  - Select a random integer k, $0 < k < q$
  - compute $r=(g^k \bmod p) \bmod q$
  - compute $k^{-1} \bmod q$
  - compute $s = k^{-1} * (h(m) + xr) \bmod q$
  - signature = (r, s)
- DSA signature verification
  - verify $0<r<q$ and $0<s<q$, if not, invalid
  - compute $w= s^{-1} \bmod q$ and h(m)
  - compute $u_1=w*h(m) \bmod q$, $u_2=r*w \bmod q$
  - compute $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$
  - valid iff v=r

DSA (cont)

- $H(m) = -xr + ks \pmod q$
- $w\ h(m) + xrw = k \bmod q$
- $u_1 + x\ u_2 = k \bmod q$
- $(g^{u1}\ y^{u2} \bmod p) \bmod q = (g^k \bmod p) \bmod q$

- Security of DSA
  - two distinct DL problems: Z_p^*, cyclic subgroup order q
- Parameters:
  - q~160bits, p 768~1Kb, p, q, g can be system wide

DSA (cont)

- Performance
  - Signature Generation
    - One modular exponentiation
    - Several 160-bit operations (if p is 1024 bits)
    - The exponentiation can be precomputed
  - Verification
    - Two modular exponentiations

comparison: RSA vs. DSA

- Speed
  - Signature generation
    - RSA
    - DSA
  - Signature verification
    - RSA
    - DSA
- Memory
  - RSA
  - DSA
- Which one do you want to use?

Blind signature scheme

- chaum for Electronic cash

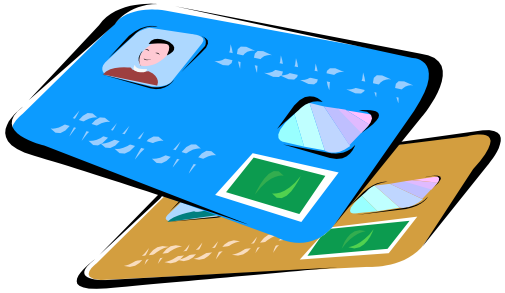
- Sender A; Signer B
- B's RSA public and private key are as usual. k is a random secret integer chosen by A, satisfying $0 \leq k < n$
- Protocol actions
  - (blinding) A: comp $m^* = mk^e \bmod n$, to B

    Note: $(mk^e)^d = m^d k$
  - (signing) B comp $s^* = (m^*)^d \bmod n$, to A
  - (unblinding) A: computes $s = k^{-1}s^* \bmod n$

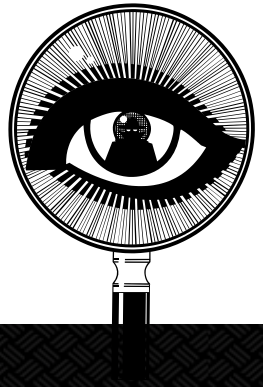
Identification

Basis of identification

- Something *known* - passwords, PINs, keys…
  - a^*ehk3&(dAs
- Something *possessed* - cards, handhelds…



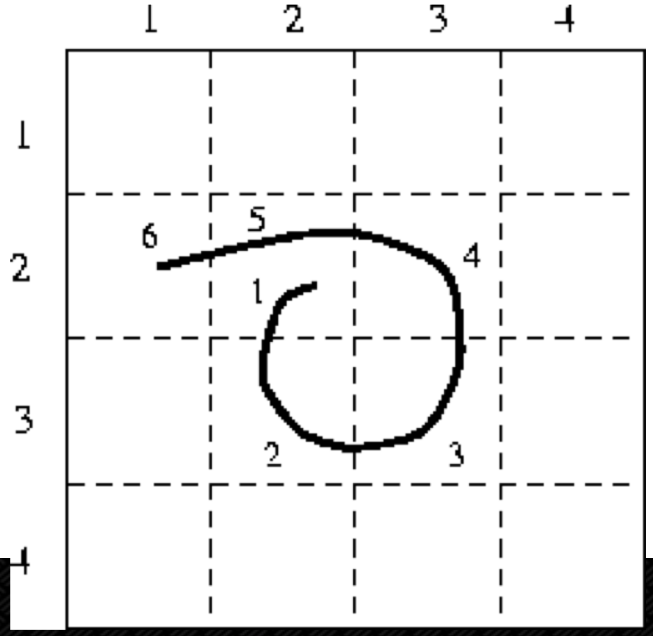
- Something *inherent* - biometrics

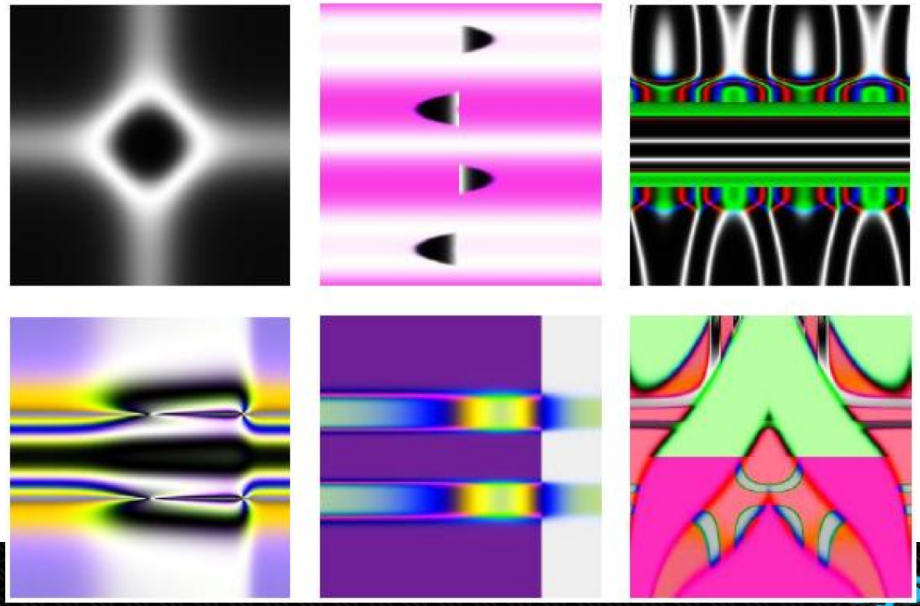


PINs and keys

- Long key on physical device (card), short PIN to remember
- PIN unlocks long key
- Need possession of both card and PIN
- Provides ***two-level*** security (or two-factor authentication)

Other password: graphical





Lamport's One Time Passwords

- User has a secret w
  - Using a OWF h, create the password sequence:

    $$w, h(w), h(h(w)), \cdots, h^t(w)$$

  - Bob knows only $h^t(w)$
  - Password for i-th identification is: $w_i = h^{t-i}(w)$

- Attacks
  - ***Pre-play attack*** - Eve intercepts an unused password and uses it later
  - Make sure you're giving password to the right party
  - Bob must be *authenticated*

Another one-time password

- Stores actual passwords on system side
- Alice and Bob share a password P
- Alice: generate r, send to Bob: $(r, h(r, P))$
- check: Bob computes $h(r, P)$, from given r, and local copy of P.
- Security
  - works only if r is something that will only be accepted once (else replay attack!)
  - Any other?

challenge-response authentication

- Alice is identified by a *secret* she possesses
  - *Bob* needs to know that Alice does indeed possess this secret
  - *Alice* provides ***response*** to a time-variant ***challenge***
  - Response depends on ***both*** secret and challenge

- using
  - Symmetric encryption
  - One way functions
  - Public key encryption
  - Digital signatures

Challenge Response using SKE

- Alice and Bob share a key K
- Taxonomy
  - ***unidirectional*** authentication using ***timestamps***
  - ***unidirectional*** authentication using ***random numbers***
  - ***Mutual*** authentication using ***random numbers***
- Unilateral authentication using timestamps
  - Alice $\longrightarrow$ Bob: $E_K(t_A, B)$
  - Bob decrypts and verified that timestamp is OK
  - Parameter B prevents replay of same message in B $\longrightarrow$ A direction

Challenge Response using SKE

- Unilateral authentication using random numbers
  - Bob $\rightarrow$ Alice: r_b
  - Alice $\rightarrow$ Bob: $E_k(r_b, B)$
  - Bob checks to see if r_b is the one it sent out
    - Also checks "B" - prevents reflection attack
  - r_b must be ***non-repeating***
- Mutual authentication using random numbers
  - Bob $\rightarrow$ Alice: r_b
  - Alice $\rightarrow$ Bob: $E_k(r_a, r_b, B)$
  - Bob $\rightarrow$ Alice: $E_k(r_a, r_b)$
  - Alice checks that r_a, r_b are the ones used earlier

challenge-response using OWF

- Instead of encryption, used keyed MAc h_k
- check: compute MAc from *known quantities*, and check with message
- SKID3
  - Bob $\longrightarrow$ Alice: r_b
  - Alice $\longrightarrow$ Bob: $r_a, h_k(r_a, r_b, B)$
  - Bob $\longrightarrow$ Alice: $h_k(r_a, r_b, A)$

Challenge-response using PKE

- Mutual Authentication based on Pk decryption
  - Alice $\rightarrow$ Bob: $P_B(r_A, B)$
  - Bob $\rightarrow$ Alice: $P_A(r_A, r_B)$
  - Alice $\rightarrow$ Bob: r_B

Challenge-response using DS

- Timestamp-based
  - Alice $\rightarrow$ Bob: $cert_A, t_A, B, S_A(t_A, B)$
  - Bob checks:
    - Timestamp ok
    - Identifier "B" is its own
    - Signature is valid (after getting public key of Alice using certificate)
- Mutual Authentication using Signatures
  - Bob $\rightarrow$ Alice: r_B
  - Alice $\rightarrow$ Bob: $cert_A, r_A, B, S_A(r_A, r_B, B)$
  - Bob $\rightarrow$ Alice: $cert_B, A, S_B(r_A, r_B, A)$

Key Establishment

classification and concepts

- use of trusted servers
  - key establishment involve a centralized or trusted party, for either or both initial system setup and on-line actions
  - trusted third party, authentication server, key distribution/ translation center, certification authority
- secure key establishment
  - each party should be able to determine the true identity of the other(s) which could possibly gain access to the resulting key, implying preclusion of any unauthorized additional parties from deducing the same key
  - secrecy of key, and identification of those parties with access to it

classification and concepts

authentication	depends on context of usage
entity authentication	identity of a party and aliveness at a given instant
data origin authentication	identity of the source of data
(implicit) key authentication	identity of party which may possibly share a key
key confirmation	evidence that a key is possessed by some party
explicit key authentication	evidence an identified party possesses a given key

classification and concepts

- (Implicit) key authentication
  - Assurance that no other party aside from a specifically identified second party may gain access to a secret key
  - No explicit check!
- Key confirmation
  - one party is assured that a second party actually has possession of a particular secret key
- Explicit key authentication
  - both (implicit) key authentication and key confirmation
  - Possession of key: (keyed) one-way hash, encryption, ZK

classification and concepts

- authenticated key establishment
  - key establishment + key authentication
- identity-based
  - identity information of the party involved is used as public key
- message-independent
  - messages sent by each party are independent of any per-session time-variant data (dynamic data) received from other parties
  - Message-independent protocols include non-interactive protocols (zero-pass and one-pass protocols)

Why session key?

- Def
  - ephemeral secret, i.e., one whose use is restricted to short time period after which all trace of it is eliminated
- Motivation
  - to limit available ciphertext for cryptanalytic attack
  - to limit exposure, with respect to both time period and quantity of data, in the event of key compromise
  - to avoid long-term storage of a large number of distinct secret keys by creating keys only when actually required
  - to create independence across communications sessions or applications

KE characteristics

- nature: Any combination of entity authentication, key authentication, and key confirmation.
- reciprocity of authentication: unilateral/mutual auth
- key freshness
- key control: key distribution vs. key agreement
- efficiency
  - number of message exchanges
  - bandwidth required by messages
  - complexity of computations by each party
  - possibility of precomputation to reduce on-line complexity
- third party requirements
  - requirement of an on-line (real-time), off-line, or no third party
- type of certificate used
- non-repudiation

Assumptions, Adversaries

- Attacks
  - passive attack: adversary simply records data, analyze
  - active attack: adversary modifies or injects messages
- What are the attacker's roles?
  - deduce a session key using info gained by tapping
  - participate covertly in protocol initiated by one party, and influence it by altering messages to deduce the key
  - initiate protocol executions and combine messages from one with another so as to carry out above attacks
  - without deducing the key, deceive good party regarding the identity of the party with which it shares a key

PFS and known key Attacks

- perfect forward secrecy
  - break long-term key $\not\Longrightarrow$ break past session keys
  - previous traffic is locked securely in the past
  - generating session keys by DH key agreement, wherein DH exponentials are based on short-term keys
  - If long-term secrets are compromised, future session can be impersonated
- known-key attack
  - compromise of past session keys allows either a passive adversary to compromise future session keys, or impersonation by an active adversary in the future.
  - in some environments, the probability of compromise of session keys may be greater than that of long-term keys

Point-to-Point Key Update

- Key Transport with one pass
  - $A \rightarrow B: E_K(r_A)$
  - Implicit key authentication
  - Additional field
    - timestamp, sequence number: freshness
    - redundancy: explicit key authentication, message modification
    - target identifier: prevent undetectable message replay
  - Hence $A \rightarrow B: E_K(r_A, t_A, B)$
  - Mutual authentication: $B \rightarrow A: E_K(r_B, t_B, A)$: $K = f(r_A, r_B)$
- Key Transport with challenge-response
  - $B \rightarrow A: n_B$: for freshness
  - $A \rightarrow B: E_K(r_A, n_A, n_B, B)$
  - $B \rightarrow A: E_K(r_B, n_B, n_A, A)$
  - cannot provide PFS
- Authenticated Key Update Protocol
  - $A \rightarrow B: r_A$
  - $B \rightarrow A: (B, A, r_A, r_B), h_K(B, A, r_A, r_B)$
  - $A \rightarrow B: (A, r_B), h_K(A, r_B)$
  - $w = h'_{K'}(r_B)$

Shamir's no key algorithm

- Protocol
  - $A \longrightarrow B$: $k^A \bmod p$
  - $B \longrightarrow A$: $(k^A)^B \bmod p$
  - $A \longrightarrow B$: $(k^{AB})^{A^{-1}} \bmod p$
- Property
  - Provide key transport
  - No a priori information is required
  - Not necessarily modular exponentiation, but not one-time pad

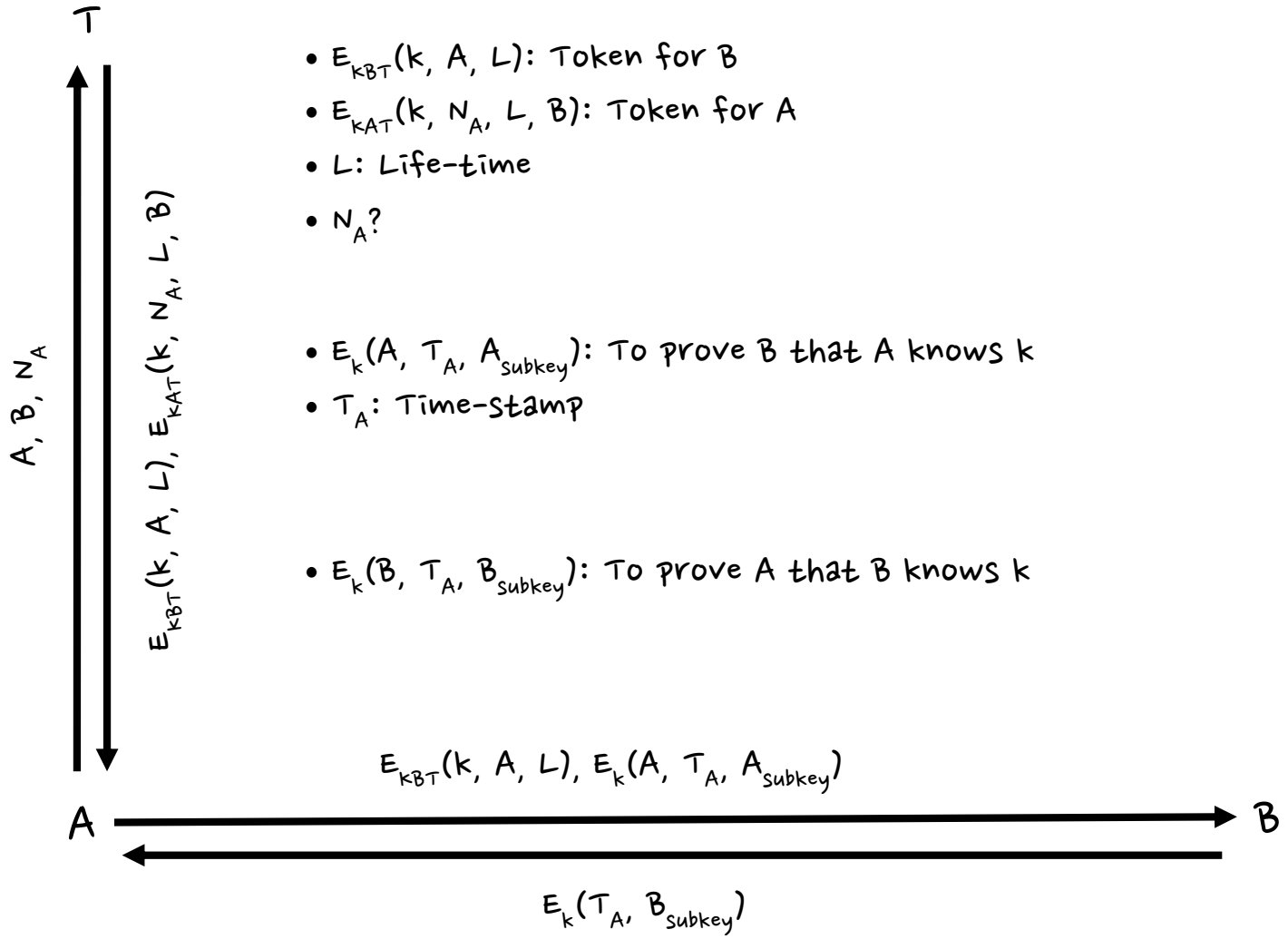
Kerberos

- Basic
  - A, B, a TTP share long-term pairwise secret keys a priori
  - TTP either plays the role of KDC and itself supplies the session key, or serves as a key translation center (KTC)
  - A and B share no secret, T shares a secret with each
  - Goal: for B to verify A's identity, establishing shared key
- Description
  - A requests for credential to allow it to authenticate itself
  - T plays the role of a KDC, returning to A a session key encrypted for A and a ticket encrypted for B
  - The ticket contains the session key and A's identity

Kerberos (cnt.)

- Protocol
  - $A \rightarrow T$: A, B, N_A N_A: freshness
  - $T \rightarrow A$: $E_{KAT}(k, A, L), E_{KBT}(k, N_A, L, B)$: L: lifetime
  - $A \rightarrow B$: $E_{KBT}(k, A, L), E_k(A, T_A, A_{subkey})$
  - $B \rightarrow A$: $E_k(T_A, B_{subkey})$ optional mutual authentication
- Properties
  - secure and synchronized clocks
  - If password-based, protocol is susceptible to password-guessing attack
  - A_{subkey} and B_{subkey} allow transfer of a key from A to B
  - Lifetime is intended to allow A to re-use the ticket

Recap: Kerberos

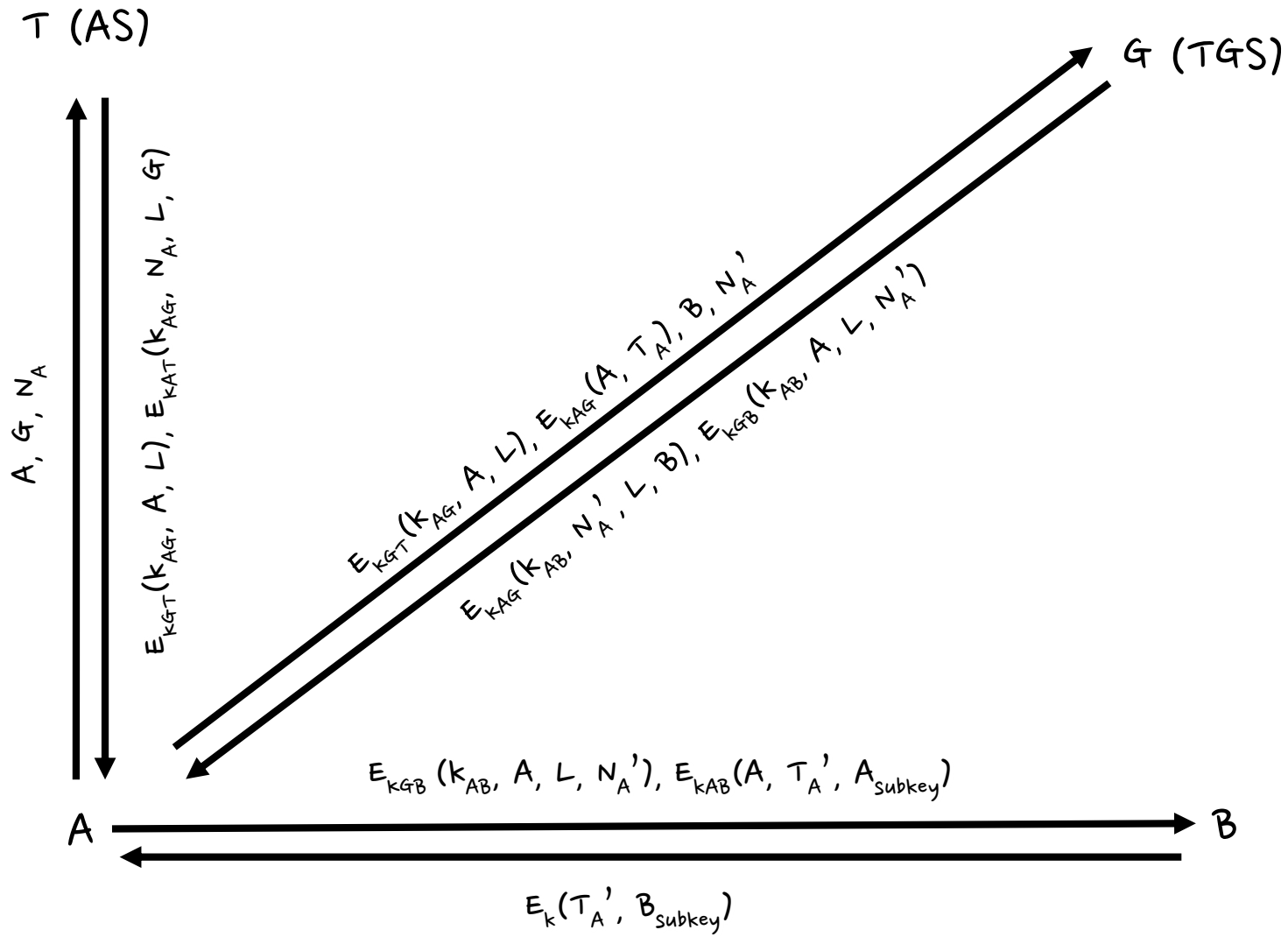


- $E_{kBT}(k, A, L)$: Token for B
- $E_{kAT}(k, N_A, L, B)$: Token for A
- L: Life-time
- N_A?

- $E_k(A, T_A, A_{subkey})$: To prove B that A knows k
- T_A: Time-stamp

- $E_k(B, T_A, B_{subkey})$: To prove A that B knows k

Recap: kerberos (scalable)



Key Transport using Pkc

- Needham-Schroeder
  - Algorithm
    - $A \rightarrow B: P_B(k_1, A)$
    - $B \rightarrow A: P_A(k_2, B)$
    - $A \rightarrow B: P_B(k_2)$
  - Properties: Mutual authentication, mutual key transport

- Modified NS
  - Algorithm
    - $A \rightarrow B: P_B(k_1, A, r_1)$
    - $B \rightarrow A: P_A(k_2, r_1, r_2)$
    - $A \rightarrow B: r_2$
  - Removing third encryption

Key Transport using Pkc

- Needham-Schroeder
  - Algorithm
    - » $A \rightarrow B: P_B(k_1, A)$
    - » $B \rightarrow A: P_A(k_1, k_2, B)$
    - » $A \rightarrow B: P_B(k_2)$
- Modified NS
  - Algorithm
    - » $A \rightarrow B: P_B(k_1, A, r_1)$
    - » $B \rightarrow A: P_A(k_2, r_1, r_2)$
    - » $A \rightarrow B: r_2$
  - Removing third encryption
- Encrypting signed keys
  - $A \rightarrow B: P_B(k, t_A, S_A(B, k, t_A))$
  - Data for encryption is too large
- Encrypting and signing separately
  - $A \rightarrow B: P_B(k, t_A), S_A(B, k, t_A)$
  - Acceptable only if no information regarding plaintext data can be deduced from the signature
- Signing encrypted keys
  - $A \rightarrow B: t_A, P_B(A, k), S_A(B, t_A, P_B(A, k))$
  - Prevent the above problem
  - can provide mutual authentication

combining PKE and DS

- Assurances of X.509 strong authentication
  - identity of A, and the token received by B was constructed by A
  - the token received by B was specifically intended for B;
  - the token received by B has "freshness"
  - the mutual secrecy of the transferred key.
- X.509 strong authentication
  - $D_A=(t_A, r_A, B, data_1, P_B(k_1))$, $D_B=(t_B, r_B, A, r_A, data_2, P_A(k_2))$,
  - $A \longrightarrow B$: $cert_A, D_A, S_A(D_A)$
  - $B \longrightarrow A$: $cert_B, D_B, S_B(D_B)$
- comments
  - Since protocol does not specify inclusion of an identifier within the scope of the encryption P_B within D_A, one cannot guarantee that the signing party actually knows (or was the source of) plaintext key

Hybrid Key Transport (PKE)

- Beller-Yacobi (4 pass)
  - Properties
    - » mutual authentication, explicit key authentication
    - » for applications where there is imbalance in processing power
    - » identity of the weaker remains concealed from eavesdroppers
  - Algorithm
    - » $B \rightarrow A : cert_B = (I_B, n_B, G_B)$: certificate generated with RSA
    - » $A \rightarrow B : P_B(K) = K^3 \bmod n_B$
    - » $B \rightarrow A : E_k(m, \{o\}^t)$: Encryption with symmetric key encryption
    - » $A \rightarrow B : E_k((v, w), cert_A)$: DSA signature with precomputation
  - comment
    - » To achieve mutual authentication, each party carry out at least one private-key operation, and one or two public-key operations
    - » careful selection of two separate public-key schemes
    - » RSA PKE and ElGamal signature are cheap

Hybrid Key Transport (PKE)

- Beller-Yacobi (2 pass)
  - Algorithm (RSA vs. ElGamal again?)

Terminal A		Server B
precompute x, $v = g^x \bmod n_S$		select random challenge m
verify $cert_B$ via $P_T(G_B)$	$\leftarrow$	send m, $cert_B$
compute $(v, w) = S_A(m, I_B)$		$cert_B = (I_B, n_B, G_B)$
send $P_B(v)$, $E_v(cert_A, w)$	$\rightarrow$	recover v, set $K = v$
$cert_A = (I_A, u_A, G_A)$		verify $cert_A$, signature (v, w)

  - I_M: Identity of M, G_M: certificate of M, u_A: ElGamal public key of A, n_B: RSA modulus
  - Properties: slightly weaker authentication assurances
    - » B obtains entity authentication of A and obtains a key K that A alone knows, while A has key authentication with respect to B
    - » For A to obtain explicit key authentication of B, a third message may be added whereby B exhibits knowledge through use of K on a challenge or standard message (e.g., $\{o\}^t$)

contents

- classification and framework
- key transport based on symmetric encryption
- key agreement based on symmetric techniques
- key transport based on public-key encryption
- key agreement based on asymmetric techniques
- Analysis of key establishment protocols

Diffie-Hellman

- Diffie-Hellman
  - Setup: prime p, generator g of Z_p^*
  - $A \rightarrow B : g^x \bmod p$
  - $B \rightarrow A : g^y \bmod p$
  - Properties
    - » fixed exponent: zero-pass key agreement with special certificate
    - » Authentication is required

MTI/A0

- Protocol
  - $A \longrightarrow B : g^x \bmod p$
  - $B \longrightarrow A : g^y \bmod p$
  - A: $k = (g^y)^a PK_b{}^x = g^{ya}\ g^{bx} = g^{ya+bx}$
  - B: $k = (g^x)^b PK_a{}^y$
  - source-substitution attack: C is not actually able to compute k itself, but rather causes B to have false belief
    - » C registers A's public key as its own
    - » When A sends B, C replaces A's certificate with its own
    - » C forwards B's response g^y to A
    - » B concludes that subsequently received messages encrypted by $k = g^{bx+ay}$ originated from C, it is only A who knows k and can originate such messages

STS

- Algorithm
  - $A \longrightarrow B : g^x \bmod p$
  - $B \longrightarrow A : g^y \bmod p, E_k(S_B(g^y, g^x))$
  - $A \longrightarrow B : E_k(S_A(g^x, g^y))$
- Properties
  - Encryption under key k provides mutual key confirmation plus allows the conclusion that the party knowing the key is that which signed the exponentials.

contents

- classification and framework
- key transport based on symmetric encryption
- key agreement based on symmetric techniques
- key transport based on public-key encryption
- key agreement based on asymmetric techniques
- Analysis of key establishment protocols

Attack strategies and classic flaws

- "man-in-the-middle" attack on unauthenticated DH
- Reflection attack
  - original protocol

1. $A \rightarrow B : r_A$
2. $B \rightarrow A : E_k(r_A, r_B)$
3. $A \rightarrow B : r_B$

  - Attack

1. $A \rightarrow E : r_A$
2. $E \rightarrow A : r_A$: Starting a new session
3. $A \rightarrow E : E_k(r_A, r_A')$: Reply of (2)
4. $E \rightarrow A : E_k(r_A, r_A')$: Reply of (1)
5. $A \rightarrow E : r_A'$

  - prevented by using different keys for different sessions

Attack strategies and classic flaws

- Interleaving attacks
  - To provide freshness and entity authentication
  - Flawed protocol
    1. $A \rightarrow B : r_A$
    2. $B \rightarrow A : r_B, S_B(r_B, r_A, A)$
    3. $A \rightarrow B : r_A', S_A(r_A', r_B, B)$
  - Attack
    1. $E \rightarrow B : r_A$
    2. $B \rightarrow E : r_B, S_B(r_B, r_A, A)$
    3. $E \rightarrow A : r_B$
    4. $A \rightarrow E : r_A', S_A(r_A', r_B, B)$
    5. $E \rightarrow B : r_A', S_A(r_A', r_B, B)$
  - Due to symmetric messages (2), (3)

Questions?

- Yongdae Kim
  - email: yongdaek@kaist.ac.kr
  - Home: http://syssec.kaist.ac.kr/~yongdaek
  - Facebook: https://www.facebook.com/y0ngdaek
  - Twitter: https://twitter.com/yongdaek
  - Google "Yongdae Kim"