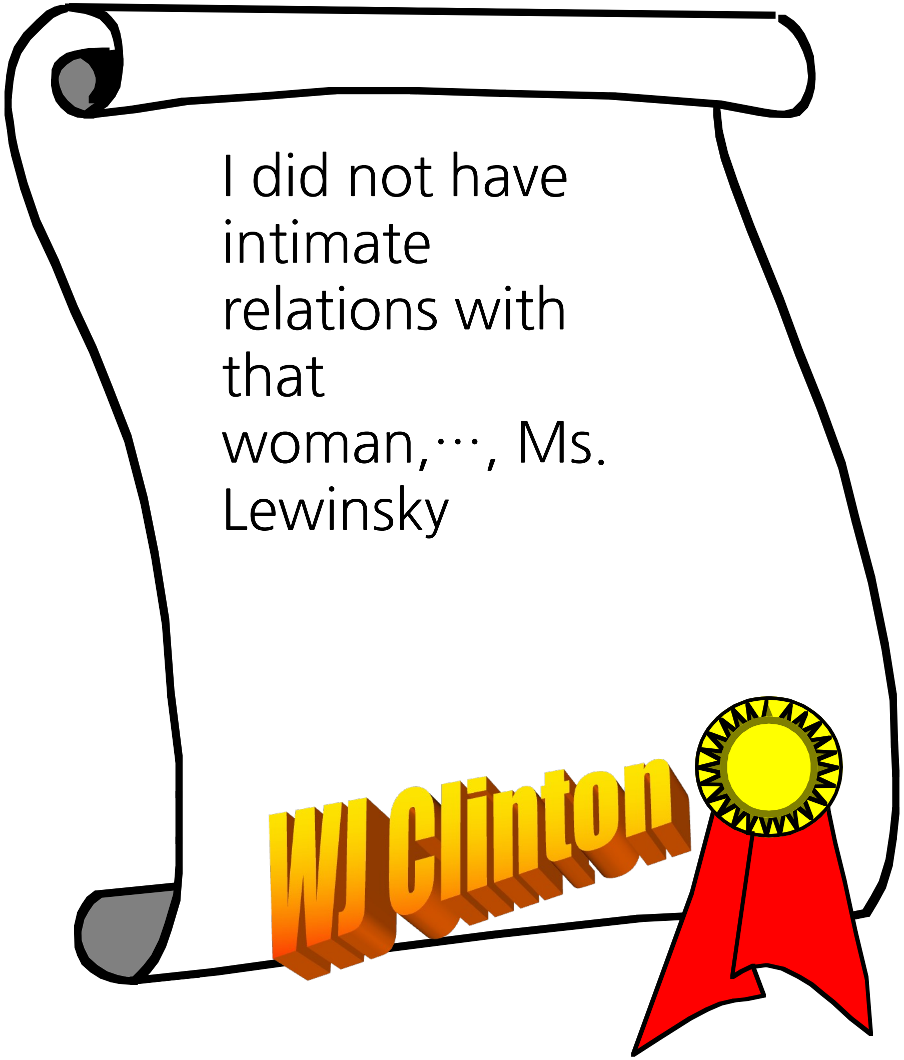
EE488
Introduction to Cryptography Engineering

Yongdae Kim

Digital Signature

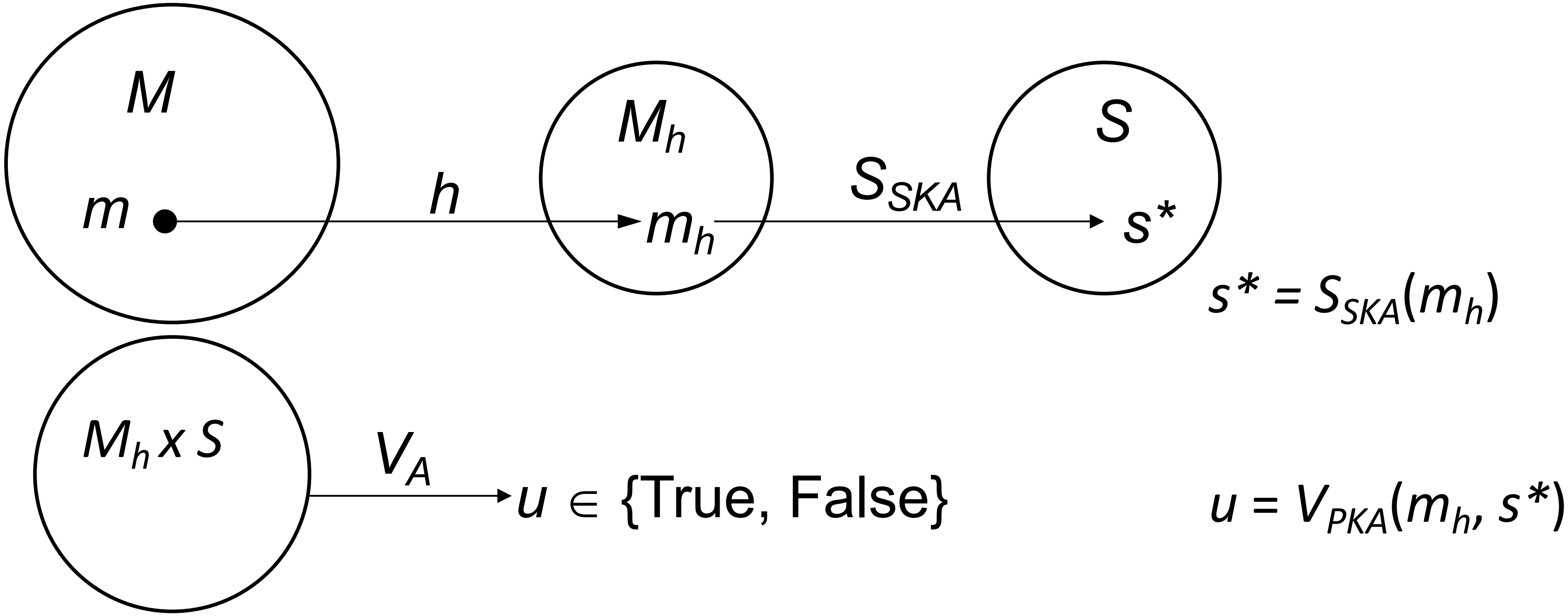
Digital Signature



- 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_{SKA} should be efficient to compute
- V_{PKA} 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 $V_{PKA}(m', s*) = true$, 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
  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

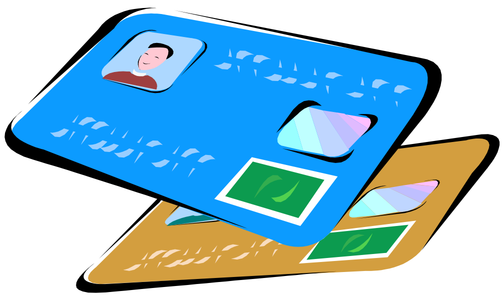
- Chaum for Electronic Cash

- Sender A; Signer B
- B's RSA public and private key are as usual. k is a random secret integer chosen by A, satisfying $0 \le k < n$
- 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 $s = k^{-1}s^* \bmod n$

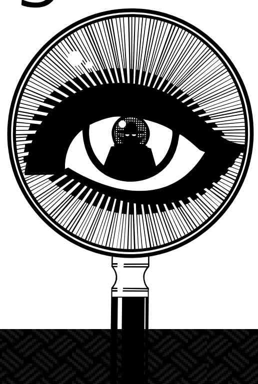
Identification

Basis of identification

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



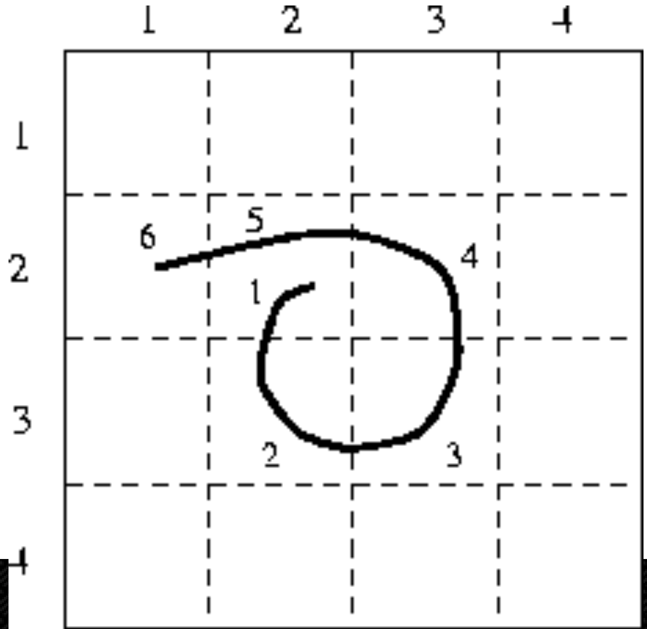

- Something *inherent* - biometrics

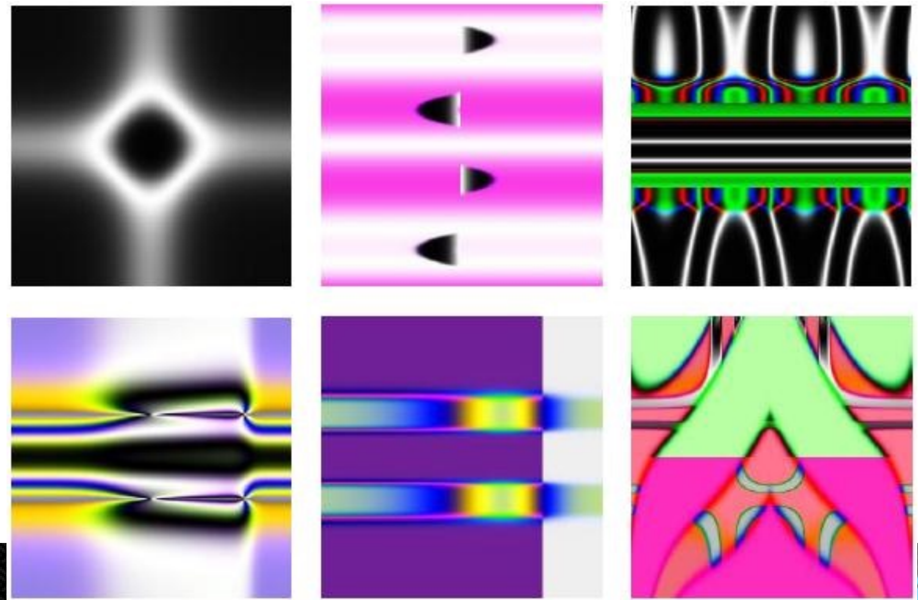



PINs and keys

- 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 → Bob: $E_K(t_A, B)$
  - Bob decrypts and verified that timestamp is OK
  - Parameter B prevents replay of same message in B → A direction

Challenge Response using SKE

- Unilateral authentication using random numbers
  - Bob → Alice: r_b
  - Alice → 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 → Alice: r_b
  - Alice → Bob: $E_K(r_a, r_b, B)$
  - Bob → 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 → Alice: r_b
  - Alice → Bob: r_a, $h_K(r_a, r_b, B)$
  - Bob → Alice: $h_K(r_a, r_b, A)$

Challenge-response using PKE

- Mutual Authentication based on PK decryption
  - Alice → Bob: $P_B(r_A, B)$
  - Bob → Alice: $P_A(r_A, r_B)$
  - Alice → Bob: r_B

Challenge-response using DS

- Timestamp-based
  - Alice → 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 → Alice: r_B
  - Alice → Bob: $cert_A$, r_A, B, $S_A(r_A,r_B,B)$
  - Bob → Alice: $cert_B$, A, $S_B(r_A,r_B,A)$

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"