Digital Signature And Hash Function

密碼學與應用

海洋大學資訊工程系
丁培毅

Electronic Signature

- Electronic Signature
  - Digital Signature
  - Biometric Signature
- Electronic Signature Act
  - ROC, 2002/04/01,
    http://www.moea.gov.tw/~meco/doc/ndoc/s5_p05.htm
    http://www.esign.org.tw/statutes.asp
  - US Federal, 2000/06
  - Japan, 2000/05

RSA

- RSA two large prime numbers p, q
  modulus $n = p \cdot q$
  public key e, $\gcd(e, \phi(n)) = 1$
  private key d, $e \cdot d \equiv 1 \pmod{\phi(n)}$
  - RSA cryptosystem
    message $m \in Z_n$
    encryption: ciphertext $c \equiv m^e \pmod{n}$
    decryption: plaintext $m \equiv c^d \pmod{n}$
  - RSA signature scheme
    message digest (document) $m \in Z_n$
    signing: signature $s \equiv m^d \pmod{n}$
    verification: document $m \equiv s^e \pmod{n}$

RSA Signature Scheme

- The signature $\boldsymbol{s}$ in RSA signature scheme is required to satisfy
  $$m \equiv s^e \pmod{n}$$
- The signature in every digital signature scheme has to satisfy an equation similar to the above equation which is formed by a trapdoor one way function.
  - Given the signature $\boldsymbol{s}$, it is easy to verify its validity.
  - Given the document $\boldsymbol{m}$, it is difficult to forge a signature $\boldsymbol{s}$ for the document $\boldsymbol{m}$ without the trapdoor information.
- Eve's attack #1: Given a pair of document and Alice's signature (m, s)
  - wants to forge the signature of Alice for a second document $\boldsymbol{m_1}$
  - (m_1, s) does not work, since $m_1 \neq s^e \pmod{n}$.
  - needs to solve $m_1 \equiv s_1^e \pmod{n}$ for s_1 ← The same tough problem as decrypting an RSA ciphertext.
- Eve's attack #2:
  - wants to forge the signature of Alice
  - chooses s_1 first and calculate $m_1 \equiv s_1^e \pmod{n}$ ← It is very unlikely that m_1 will be meaningful.

Attack RSA Signature

- RSA signature scheme: $s \equiv m^d \pmod n$
- suppose Alice is not willing to sign the message m

- Eve's attacking scheme: (almost always is meaningless)
  - decompose the message: $m \equiv m_1 \cdot m_2 \pmod n$
  - ask Alice to sign m_1 and m_2 independently and get
    $s_1 \equiv m_1^d \pmod n$ and $s_2 \equiv m_2^d \pmod n$
  - multiply the two signatures together to get
    $s \equiv s_1 \cdot s_2 \equiv m_1^d \cdot m_2^d \equiv (m_1 m_2)^d \equiv m^d \pmod n$

- Morale: never sign a message that does not make any sense to you (never sign a message that contains unrecognized binary data)

Rabin Signature Scheme

- Key generation: public key $n=p \cdot q$, private key p, q
- Signing:
  - for a plaintext m, $0<m<n$, $m \in QR_p \cap QR_q$ (i.e. QR_n)
  - signature is s, such that $m \equiv s^2 \pmod n$
- Verification
  - $m \equiv s^2 \pmod n$
- Chosen Message Attack (This is not easy if m is required to be plaintext.)
  - Eve chooses x and computes $m \equiv x^2 \pmod n$
  - Ask Alice for a signature s on m
  - $\Pr\{ s \neq \pm x \} = 0.5$

Making Rabin signature only on hashed message can avoid this attack. Never take square root directly!!

ElGamal Signature Scheme

- Probabilistic: There are many signatures that are valid for a given message.
- **Key generation**: Alice chooses a large prime number p, a primitive α in Z_p^*, a secret integer a, and calculates $\beta \equiv \alpha^a \pmod p$ (p, α, β) are the public key, a is the secret key
- **Signing**: Alice signs a message m
  - select a secret random k such that $\gcd(k, p\text{-}1) = 1$
  - $r \equiv \alpha^k \pmod p$
  - $s \equiv k^{-1} (m - a\, r) \pmod{p\text{-}1}$

  (r, s) is the signature
- **Verification**: anyone can verify the signature (r, s)
  - compute $v_1 \equiv \beta^r r^s \pmod p$ and $v_2 \equiv \alpha^m \pmod p$
  - signature is valid iff $v_1 \equiv v_2 \pmod p$

ElGamal Signature Scheme

- Proof:
  $v_2 \equiv \alpha^m \equiv \alpha^{sk+ar} \equiv (\alpha^a)^r (\alpha^k)^s \equiv \beta^r r^s \equiv v_1 \pmod p$
- Example
  - Alice wants to sign a message 'one' i.e. $m_1 = 151405$
  - She chooses p=225119, α=11, a secret a=141421, $\beta \equiv \alpha^a \equiv 18191 \pmod p$
  - To sign the message, she chooses a random number k=239, $r \equiv \alpha^k \equiv 164130$, $s_1 \equiv k^{-1} (m_1 - a\, r) \equiv 130777 \pmod{p\text{-}1}$ …. (m_1, r, s_1) is the signature
  - Bob wants to verify if Alice signs the message m_1
  - He calculates $\beta^r r^{s_1} \equiv 128841 * 193273 \equiv 173527$, $\alpha^{m_1} \equiv 173527$
- Signature with Appendix
  - message can not be recovered from the signature
  - ElGamal, DSA
- Message Recovery Scheme
  - message is readily obtained from the signature
  - RSA, Rabin

ElGamal Signature Scheme

- Security:
  - ? Discrete Log   Decisional Diffie-Hellman
  - given public β, solving for *a* is a discrete log problem
  - fixed *r*, solving $v_2 \equiv \beta^r r^s \pmod p$ for *s* is a discrete log problem
  - fixed *s*, solving $v_2 \equiv \beta^r r^s \pmod p$ for *r* is not proven to be as hard as a discrete log problem (believed to be non-polynomial time)
  - it is not known whether there is a way to choose *r* and *s* simultaneously which satisfy $v_2 \equiv \beta^r r^s \pmod p$
  - Bleichenbacher, "Generating ElGamal signatures without knowing the secret key," Eurocrypt96
    - forging ElGamal signature is sometimes easier than the underlying discrete logarithm problem

Existential Forgeries

- RSA
  Choose $s \in_R Z_n^*$
  Let $m \equiv s^e \pmod n$
  (m, s) is a valid message signature pair
- ElGamal
  1-parameter
  Choose $e \in_R Z_q$
  Let $r \equiv g^e \cdot y \pmod p$, $s \equiv -r \pmod q$, $m \equiv e \cdot s \pmod p$
  (m, (r,s)) is a valid message signature pair
  2-parameter
  Choose $e, v \in_R Z_q$
  Let $r \equiv g^e \cdot y^v \pmod p$, $s \equiv -r \cdot v^{-1} \pmod q$,
  $m \equiv e \cdot s \pmod p$
  (m, (r,s)) is a valid message signature pair

ElGamal Signature Scheme

- Security:
  - Should not use the same random number *k* twice for two distinct messages. Eve can easily know this by comparing *r* in both signatures. Eve can then break this system completely and forge signatures at will.

    $s_1 k - m_1 \equiv -a r \equiv s_2 k - m_2 \pmod{p-1}$

    $(s_1 - s_2) k \equiv m_1 - m_2 \pmod{p-1}$

    There are $\gcd(s_1 - s_2, p-1)$ solutions for *k*.

    Eve can enumerate all α^k until she finds *r*.

    After knowing k, Eve can solve the following equation for a

    $a r \equiv m_1 - s_1 k \pmod{p-1}$

    There are $\gcd(r, p-1)$ solutions for *a*.

    Eve can enumerate all α^a until she finds β.

Example

- Example continued
  - Alice wants to sign a second message 'two' i.e. $m_2 = 202315$
  - She uses the same ElGamal parameters as before p=225119, α=11, a secret a=141421, $\beta \equiv \alpha^a \equiv 18191 \pmod p$
  - She signs this message with the same random number k=239, $r \equiv \alpha^k \equiv 164130$, $s_2 \equiv k^{-1} (m_2 - a\ r) \equiv 164899 \pmod{p-1}$ …. (m_2, r, s_2) is the signature
  - Eve can compute $(s_1 - s_2) k \equiv -34122 k \equiv m_1 - m_2 \equiv -50910 \pmod{p-1}$.
  - Since gcd(-34122, p-1) = 2, k has two solutions 239 or 112798
  - Because $r \equiv \alpha^k \pmod p$, Eve can verify easily that k = 239
  - $k s_1 \equiv m_1 - a r \pmod{p-1} \Rightarrow a = 28862$ or 141421
  - $\beta \equiv \alpha^a \pmod p \Rightarrow a = 141421$

ElGamal Signature Scheme

✧ General ElGamal Signature Schemes

- Horster, Michels, and Petersen, "Meta-ElGamal Signature Schemes," Tech. Report TR-94-5, Univ. of Technology Chemnitz-Zwichau, 1994
- 6 types, 6500+ variations
- ex. Rearrange *m*, *r*, *s* of $m \equiv a\, r + k\, s \pmod{p\text{-}1}$ as

  $$A \equiv a\, B + k\, C \pmod{p\text{-}1}$$

  verification equation $\alpha^A \equiv \beta^B r^C \pmod{p}$

A	B	C		
m	r	s	$m \equiv a\, r + k\, s$	$\alpha^m \equiv \beta^r r^s$
m	s	r	$m \equiv a\, s + k\, r$	$\alpha^m \equiv \beta^s r^r$
s	r	m	$s \equiv a\, r + k\, m$	$\alpha^s \equiv \beta^r r^m$
s	m	r	$s \equiv a\, m + k\, r$	$\alpha^s \equiv \beta^m r^r$
r	s	m	$m \equiv a\, s + k\, m$	$\alpha^r \equiv \beta^s r^m$
r	m	s	$r \equiv a\, m + k\, s$	$\alpha^r \equiv \beta^m r^s$

ElGamal Signature Scheme

✧ Signing two messages at the same time

- $r \equiv \alpha^k \pmod{p}$
- $m_1 \equiv a\, m_2\, r + k\, s \pmod{p\text{-}1}$
- (r, s) is the signature for m_1 and m_2 together

✧ Signing three messages at the same time

- $r \equiv \alpha^k \pmod{p}$
- $m_1 \equiv a\, m_2\, r + k\, m_3\, s \pmod{q}$
- (r, s) is the signature for m_1, m_2 and m_3 together

Attacks on ElGamal Signature

✧ D. Bleichenbacher, "Generating ElGamal Signatures Without Knowing the Secret Key," Eurocrypt'96

1. Prime p should be large enough to prevent GNFS on DL
2. ∃ large prime q | p-1 s.t. Pohlig-Hellman method fails
3. Using collision resistant hash function on message to prevent existential forgeries
4. Should verify 1≤ r < p: otherwise leads to forgery from a known signature, will be shown later
5. Avoid a smooth g which divides p-1, has trapdoor for forging signatures
6. ElGamal over Z_n^* is not as secure as it appears: known signatures leak the factorization of n and the computation of either Z_p^* or Z_q^* is sufficient to forge signatures

Implementation Existential Forgery

✧ Verifier should verify that 1 ≤ r < p

✧ Otherwise anybody can forge a signature (r', s') for arbitrary hash value h' from a known signature (r, s) on hash value h

✧ For an arbitrary message m' with hash value h'

$u \equiv h' \cdot h^{-1} \pmod{p-1}$

$$\boxed{g^{h'} \equiv g^{h \cdot u} \equiv y^{r \cdot u}\, r^{s \cdot u} \pmod{p}}$$

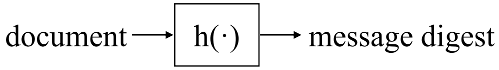
Calculate r' from CRT s.t. $r' \equiv \begin{cases} r \cdot u \pmod{p-1} \\ r \pmod{p} \end{cases}$

$s' \equiv s \cdot u \pmod{p-1}$

(r', s') is the ElGamal signature for h' = hash(m')

Cryptographic Hash Function

- ✧ Input: arbitrary length of message, *m*
- ✧ Output: *h*(*m*), fixed length (ex. 160 bit) message digest
- ✧ Requirements:



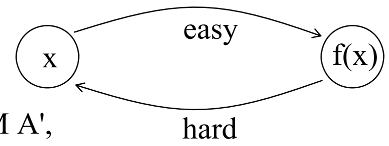
  - ✶ efficient calculation of *h*(*m*)
  - ✶ given $y = h(m)$, it is computationally infeasible to find a distinct message *m*' such that $h(m') = y$ (**weak collision resistance**, for signature scheme)
  - ✶ it is computationally infeasible to find two distinct messages m_1 and m_2 with $h(m_1) = h(m_2)$ (**strong collision resistance**, for resisting birthday attack)

  one-way

- ✧ Examples: Snefru, N-Hash, MD2, MD4, MD5, RIPE-MD160, SHA, SHA-1, SHA-(256, 384, 512) (2002/08)

One-way Function

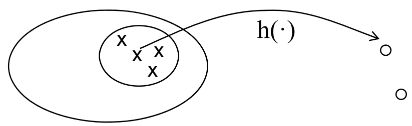
- ✧ Definition based on Complexity theory not Mathematics
- ✧ OWF: a function that is easy to evaluate yet its inverse is hard to compute



For every probabilistic poly-time TM A',
every positive polynomial p(·) and all sufficient large n

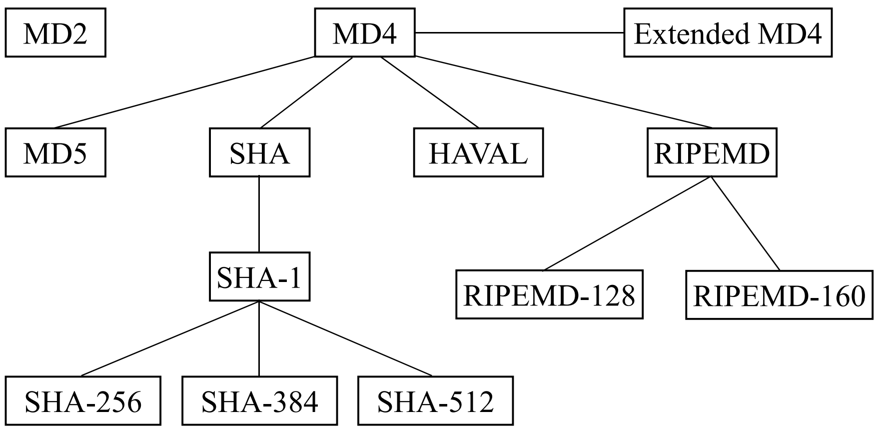
$$\Pr\{A'(f(U_n), 1^n) \in f^{-1}f(U_n)\} < 1 / p(n)$$ negligible

- ✧ A weak collision free hash function is a one-way function



given *y*, it is computationally infeasible to find any message *m* such that $h(m) = y$

Popular Hash Functions



Cryptographic Hash Function

- ✧ Discrete Log Hash Function
  - ✶ D. Chaum, E. van Heijst, B. Pfitzmann, "Cryptographically Strong Undeniable Signatures Unconditionally Secure for the Signer", Crypto'91
  - ✶ satisfies the second and the third requirements
  - ✶ too slow to be used
  - ✶ select a prime number *p*, such that $q=(p-1)/2$ is also a prime number
  - ✶ choose two random primitive roots α, β in Z_p
  - ✶ there exists unique *a* such that $\alpha^a \equiv \beta \pmod{p}$, assume *a* is unknown (a discrete log problem, since α, β are chosen independently)
  - ✶ hash function $h : Z_{q^2} \to Z_p$

    $h(m) = \alpha^{x_0}\beta^{x_1} \pmod{p}$

    where $m = x_0 + x_1 q$ with $0 \le x_0, x_1 \le q-1$

    note: *h*(*m*) is about half the bit length of *m*

Cryptographic Hash Function

✧ Proposition: If we have an algorithm A that can find $m' \neq m$ with $h(m)=h(m')$, then using A we can determine the discrete log $a = L_\alpha(\beta)$

a reduction argument

proof: if we are given the output of A, e.g., m and m'

we can write $m = x_0 + x_1 q$ and $m' = x'_0 + x'_1 q$

$h(m) \equiv h(m') \Rightarrow \alpha^{x_0} \beta^{x_1} \equiv \alpha^{x'_0} \beta^{x'_1} \pmod p$

$\alpha^a \equiv \beta \Rightarrow \alpha^{a(x_1 - x'_1) + (x_0 - x'_0)} \equiv 1 \pmod p$

α is primitive $\Rightarrow a(x_1 - x'_1) + (x_0 - x'_0) \equiv 0 \pmod{p-1}$

this congruence equation has $d = \gcd(x_1 - x'_1, p-1)$ solutions, and can be found easily

Cryptographic Hash Function

since 1. $x_1 \neq x'_1$ (otherwise run A again with different ω)

2. only 1, 2, q, p-1 divides p-1 and

3. $-(q-1) \leq x_1 - x'_1 \leq (q-1)$

random tape

$\Rightarrow$ d can only be 1 or 2

$\Rightarrow$ we can easily test both solutions and determine $a = L_\alpha(\beta)$

✧ Given α, β, p ($p=2q+1$, α, β are primitives, there are $\phi(p-1)=\phi(2q)=q-1$ primitives), find $L_\alpha(\beta)$:

1. using algorithm A to find m and m' s.t. $h(m) = h(m')$
2. write $m = x_0 + x_1 q$ and $m' = x'_0 + x'_1 q$
3. solve $a(x_1 - x'_1) + (x_0 - x'_0) \equiv 0 \pmod{p-1}$ for a

Cryptographic Hash Function

✧ Properties of $h(m) = \alpha^{x_0} \beta^{x_1}$ (mod p)

- $h(\cdot)$ is strongly collision resistant
  from the above proposition, the efficient algorithm A that finds m and m' such that $h(m) = h(m')$ is unlikely to exist
- $h(\cdot)$ is weakly collision resistant
  1. Assume h() is not w.c.r. $\Rightarrow \exists$ an inverse function of $h(\cdot)$
  2. g($\cdot$): given $m \in Z_{q^2}$ and $y=h(m) \in Z_p$, it is efficient to compute $m' = g(y) \in Z_{q^2}$ such that $h(m') = y$
  3. $|Z_{q^2}| >> |Z_p| \Rightarrow$ it is very likely that $g(y) \neq m$ (otherwise try another m), therefore, we have an algorithm A that can find $m \neq m'$ but $h(m)=h(m')$ contradict to the 'strong collision resistant' property

Cryptographic Hash Function

✧ Discussion: 'strong collision freeness of $h(\cdot)$' given $h(\cdot)$ it is hard to find m_1, m_2 such that $h(m_1)=h(m_2)$

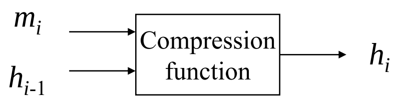
computationally infeasible

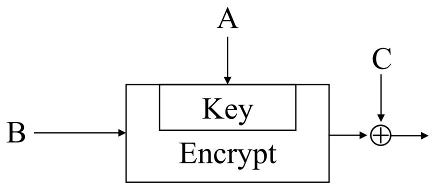
- because the length of $h(m)$ is far less than the length of m, the mapping $h(\cdot)$ is definitely many to one
- to make it computationally infeasible to find two distinct m_1 and m_2 such that $h(m_1)=h(m_2)$

  intuitively, the set of m's that map to the same $h(m)$ have to be randomly distributed among many many other m's that have different $h(m)$

Cryptographic Hash Function

- ✧ Hash function based on symmetric block cipher
  - ✶ if the block algorithm is secure then the one-way hash function is secure?? (never proved, Damgård, Crypto'89)





A, B, C can be either m_i, h_{i-1}, $m_i \oplus h_{i-1}$

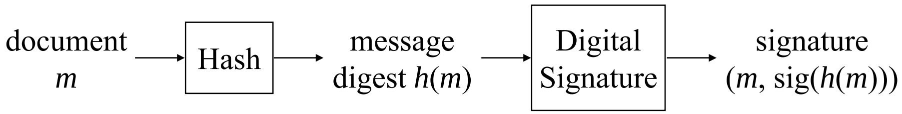
Cryptographic Hash Function

- ✧ Not all 81 assignments of A, B, C are secure, the following 12 assignments are OK (especially the first 4)

A	B	C
h_{i-1}	m_i	m_i
h_{i-1}	$m_i \oplus h_{i-1}$	$m_i \oplus h_{i-1}$
h_{i-1}	m_i	$m_i \oplus h_{i-1}$
h_{i-1}	$m_i \oplus h_{i-1}$	m_i
m_i	h_{i-1}	h_{i-1}
m_i	$m_i \oplus h_{i-1}$	$m_i \oplus h_{i-1}$
m_i	h_{i-1}	$m_i \oplus h_{i-1}$
m_i	$m_i \oplus h_{i-1}$	h_{i-1}
$m_i \oplus h_{i-1}$	m_i	m_i
$m_i \oplus h_{i-1}$	h_{i-1}	h_{i-1}
$m_i \oplus h_{i-1}$	m_i	h_{i-1}
$m_i \oplus h_{i-1}$	h_{i-1}	m_i

Application of cryptographic hash function

- ✧ Digital Signature:



  - ✶ efficient computation and storage

Application of cryptographic hash function

- ✶ security: weak collision resistant property of $h(m)$ thwarts forgers
  'Given (m, sig($h(m)$)) and another $m'(\neq m)$,
  Is Eve capable of finding sig($h(m')$)?'
  - ✡ the underlying signature algorithm guarantees that it is computationally difficult to find sig($h(m')$) given $h(m')$ without the trapdoor information
  - ✡ if $h(m') = h(m)$ then sig($h(m')$) will be sig($h(m)$)
    However, given m, we know $h(m)$, 'weakly collision resistant property of $h(\cdot)$' guarantees that it is computationally infeasible to find m' such that $h(m') = h(\mathrm{m})$

Application of cryptographic hash function

- Data Integrity:
  - data transmitted in noisy channel
  - data transmitted in insecure channel

  errors: insertion, deletion, modification, rearrangement

  - non-cryptographic: parity, CRC32
    only increase the detection probability of errors
  - cryptographic: collision resistant, detect almost all errors (slow)

The Birthday Paradox

- r = 23 Pr{any two of them have the same birthday} ≈ 0.5
- r = 30 Pr{any two of them have the same birthday} ≈ 0.7
- r = 40 Pr{any two of them have the same birthday} ≈ 0.9

The Birthday Paradox (cont'd)

Pr { r people have different birthdays }

r = 2, (1-1/365) = .997

r = 3, (1-1/365)(1-2/365) = .992

r = 4, (1-1/365)(1-2/365)(1-3/365) = .984

…

r = 23, (1-1/365)(1-2/365)… (1-22/365) = .493

Pr { at least two having the same birthday }

= 1 - Pr { all r people have different birthday } = .507

The Birthday Paradox (cont'd)

- $e^{-x} = 1 - x + x^2 / 2! - x^3 / 3! + \ldots$
  if x is a small real number, ex. 1/365, then $1 - x \approx e^{-x}$
- $(1-1/365)(1-2/365) \ldots (1-(r-1)/365) = \prod_{i=1}^{r-1} (1 - i/365)$

  $$\approx \prod e^{-i/365} = e^{-\sum i/365} = e^{-r(r-1)/(2*365)}$$
- $\varepsilon = \Pr\{\text{at least one collision}\} \approx 1 - e^{-r(r-1)/(2n)}$

  $-r(r-1)/(2n) \approx \ln(1-\varepsilon)$

  define $\lambda = -\ln(1-\varepsilon)$

  $r^2 - r \approx 2 n \lambda$

  neglecting r, we obtain $r \approx \sqrt{2 n \lambda}$

The Birthday Paradox (cont'd)

✧ In general,

- ✶ n kinds of objects (n is large, each kinds of objects have infinite supplies)
- ✶ r people each chooses one object independently

Let ε = Pr { at least two choose the same kind of object }
define $\lambda = -\ln(1-\varepsilon)$ i.e. $\varepsilon = 1 - e^{-\lambda}$

From the previous derivation $r \approx \sqrt{2\lambda n}$

eg: if $\lambda = 0.693$ Pr {..} $\approx 1 - e^{-.693} = 0.5$
n = 365 $\sqrt{2\ .693\ 365} = 22.49$

Birthday Attack

✧ A slightly different scenario

- ✶ n kinds of objects (n is large, each kinds of objects have infinite supplies)
- ✶ two groups, each has r people, every one chooses one object independently

$r \approx \sqrt{\lambda n}$

Pr { at least one in the first group chooses the same kind of object as someone in the second group chooses } $\approx 1 - e^{-\lambda}$

note: Pr{ i matches } $\approx \lambda^i e^{-\lambda} / i!$ ← $e^{\lambda} = 1 + \ldots + \lambda + \frac{\lambda^2}{2!} + \frac{\lambda^3}{3!}$
ie. Pr { at least two matches} $\approx 1 - e^{-\lambda} - \lambda e^{-\lambda}$

Birthday Attack

✧ Ex. $\Pr\{\cdot\} \approx 1 - e^{-\lambda} = 0.5$

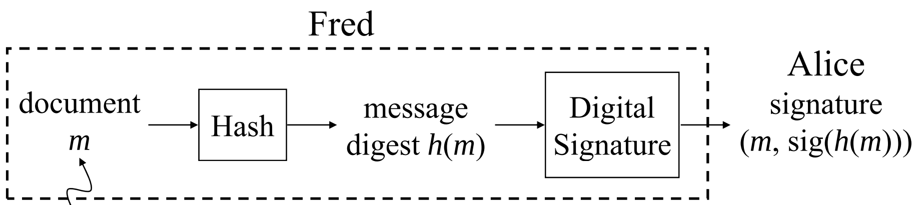
$\Rightarrow \lambda \approx 0.693$

$\Rightarrow r \approx \sqrt{0.693\ n} \approx 0.83\sqrt{n}$

n=365, $r \approx 15.9$

Birthday Attack on Digital Signature

✧ Actually attack on the one-way hash function



Receipt serial #12345678
...
I, Fred, hereby owe you, Alice, 10000 dollars.
…
Fred
01/01/04

If $h(m)$ is a 50-bit string, Alice would think that she is safe because if the hash is a random mapping, the computation time to find another document with the same hash as the given one, $h(m)$, would be $O(2^{50}) \approx (10^{15})$

Birthday Attack on Digital Signature

F's

Receipt serial #12345678
...
I,△Fred△,hereby△△
owe you,Alice,△△△
△100△dollars.△
...
△ Fred△△
△01/01/04△△

U's

Receipt serial #12345678
...
I,△Fred△,hereby owe
you,Alice,△10000△△△
dollars.△△△△△△
...
△ Fred△△
△01/01/04△△

- Fred finds 30 places where he can make slight changes in both favorable (F) and unfavorable (U) versions of documents. i.e.
  - $r = 2^{30}$, $n = 2^{50}$, $\lambda = r^2 / n = 2^{10} = 1024$
  - Fred have r variations of $\{F_i\}$'s and r variations of $\{U_i\}$'s
  - Pr{ there is at least one match in $h(F_i)$ and $h(U_i)$ } $\approx 1 - e^{-\lambda} \approx 1$
- let $h(F_{i^*}) = h(U_{j^*})$, Fred gave U_{j^*} to Alice when he got \$10000 from her, but later claimed that the document is F_{i^*}

Avoid the Birthday Attack

- Alice changes slightly the document *m* to *m*' (wording, spaces, formats, …) before Fred signs the document
  - so that $h(m') \neq h(m)$
  - In order to obtain another document that has the same hash $h(m')$, Fred needs to search on average $2^{50/2}$ documents.
- Alice should choose a hash function with output twice as long as what she feel safe. For example, in this case she should ask Fred to use a hash function with 100-bit output. (The birthday attack effectively halves that number of bits.)

Birthday Attack to solve Discrete Log

- given α,β and *p*, find *x* such that $\alpha^x \equiv \beta \pmod p$
- procedure
  - step 1: calculate and save $\alpha^k \pmod p$ for $\sqrt{p}$ random *k*
  - step 2: calculate and save $\beta\, \alpha^{-i} \pmod p$ for $\sqrt{p}$ random *i*
  - step 3: compare these two sets to find a match
- analysis
  - $\lambda = 1$, $\Pr\{\exists k, i, \alpha^k \equiv \beta\, \alpha^{-i} \pmod p\} \approx 1 - e^{-\lambda} = 0.632$

    ⇒ let k^*, i^* be the index such that $\alpha^{k^*} \equiv \beta\, \alpha^{-i^*} \pmod p$

    ⇒ $\alpha^{k^*+i^*} \equiv \beta \pmod p$

    ⇒ $L_\alpha(\beta) \equiv k^* + i^* \pmod{p-1}$

Note: repeat step 1 and step 2 if k^* and i^* can not be found

Pr{success}: 0.632 → 0.864 → 0.95
(1 repetition, 2nd repetition, 3rd repetition)

Meet-in-the-Middle Attack

- Similar structure to birthday attack
- Deterministic, always find the solution
- Double DES Encryption:

let $E_{k_1}(\cdot)$, $E_{k_2}(\cdot)$ be two 56-bit DES,
Can $E_{k_2}(E_{k_1}(\cdot))$ achieve the level of security as a 112-bit symmetric cryptosystem?

Note: for RSA $(m^{e_1})^{e_2}$ is equivalent to m^{e_3} (for the same *n*)

for DES $E_{k_2}(E_{k_1}(\cdot))$ is not equivalent to some $E_{k_3}(\cdot)$

Meet-in-the-Middle Attack

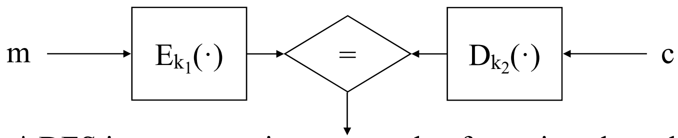
- brute-force attack on DES: given *m* and *c*, try all 2^{56} possible keys to see which key satisfies $c = E_k(m)$
- direct extension of brute-force attack on Double DES: given *m* and *c*, try all 2^{112} possible keys to see which two keys k_1 and k_2 satisfy $c = E_{k_2}(E_{k_1}(m))$
- MITM attack (smarter brute-force attack): given *m* and *c*, Eve is going to find k_1 and k_2 such that $c = E_{k_2}(E_{k_1}(m))$ with only 2^{57} DES calculations
  - step 1: calculate $E_k(m)$ for all possible *k*
  - step 2: calculate $D_k(c)$ for all possible *k*
  - step 3: compare the two lists, there is at least one match

note: if there are multiple matches, try another (*m*, *c*) pair to resolve

Meet-in-the-Middle Attack

- Analysis:
  - storage: 2^{57} blocks (= 2^{60} bytes ~ 2^{30} GB ~$8 \cdot 10^6$ 120G HD)
  - computation: 2^{57} DES + $(2^{56})^2$ comparisons
    far less than directly try out $(2^{56})^2$ DES key combinations. If Eve have plenty of power to break $E_k(m)$ in a brute-force way, she will be capable of breaking $E_{k_2}(E_{k_1}(m))$ easily.
- Triple Encryption: $E_{k_3}(E_{k_2}(E_{k_1}(m)))$ storage ↔ time tradeoff
  - given *m* and *c*, to break this system in a brute-force way, it is necessary to compute $(2^{112} + 2^{56})$ DES and 2^{168} comparisons

Meet-in-the-Middle Attack

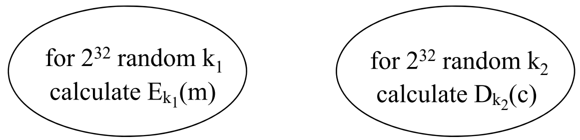


Note: * DES is a permutation, means that for a given key, different message *m* will be encrypted to different ciphertext c_1, also different ciphertext c will be decrypted to different m_1

* There could be multiple collisions for the above two lists if $E(\cdot)$ and $D(\cdot)$ are DES and its inverse, respectively. A single message *m* could be encrypted to the same ciphertext c_1 with different keys. In single DES encryption, this might not be very severe, but in two concatenated DES operations, this phenomenon would be frequent since number of key combinations (2^{112}) is far larger than number of ciphertexts (2^{64}). [in terms of BA: $r=2^{56}$, $n=2^{64}$, $\lambda=(2^{56})^2/2^{64}$]

Another thought on Double DES

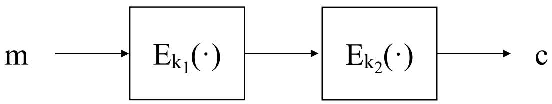
- Why don't we try to apply birthday attack on Double DES?
- In order to apply birthday attack, we prepare two lists:



Because DES encryption and decryption can be considered random mappings, 2^{32} $E_{k_1}(m)$'s and 2^{32} $D_{k_2}(c)$'s are close to random samples from 2^{64} possible ciphertexts. According to the birthday attack, the probability that there is a match in the two lists is about 0.632, it looks like that we can find a pair of keys (k_1, k_2) that can encrypt *m* to *c*.

Will "Double DES" be broken in 2^{33} DES computations?

Another thought on Double DES



- Since c is a 64-bit block, c has 2^{64} possibilities. There are 2^{112} possible (k_1, k_2) key combinations. Therefore, for a particular m, there are on average 2^{48} key combinations that can generate a given c by the pigeon hole principle. To find out the actual key used , we need to analyze many more (plaintext, ciphertext) pairs.
- The previous birthday attack scheme can only find one key combination, it would be very difficult to find out all key pairs with that kind of probabilistic scheme.

Digital Signature Algorithm

- NIST 1994 (FIPS 186), 2000 (FIPS 186-2)
- digital signature scheme with appendix, use SHA-1 (FIPS 180-1) as the hash algorithm
- Generation of keys
  - q is a 160-bit prime number, p is a 512-bit (768-bit, 1024-bit) prime number such that $q \mid p-1$
  - g is a primitive root modulo p
    $\alpha \equiv g^{(p-1)/q} \pmod{p}$ $\qquad \alpha^q \equiv (g^{(p-1)/q})^q \equiv g^{p-1} \equiv 1 \pmod{p}$
  - choose secret value a, $1 \le a \le q-1$ and calculate $\beta \equiv \alpha^a \pmod{p}$
  - public key (p, q, α, β), secret key a

Digital Signature Algorithm

- Signature: given message m and p, q, α
  - Alice selects a random secret k $\quad 0<k<q-1$
  - compute $r \equiv (\alpha^k \pmod{p}) \pmod{q}$
  - compute $s \equiv k^{-1} (m + a\, r) \pmod{q}$ ($\neq 0$, $k \cdot k^{-1} \equiv 1 \pmod{q}$)
  - signature is (r, s) $\qquad$ note: r, s are both 160 bit
- Verification: given message m and signature (r, s)
  - Bob downloads (p, q, α, β) $\qquad s \cdot s^{-1} \equiv 1 \pmod{q}$
  - compute $u_1 \equiv s^{-1}\, m \pmod{q}$ and $u_2 \equiv s^{-1}\, r \pmod{q}$
  - compute $v \equiv (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q}$
  - Bob accepts if $v = r$

Digital Signature Algorithm

- Proof:

$s \equiv k^{-1} (m + a\, r) \pmod{q}$

$m = (-\, a\, r + k\, s) \pmod{q}$

$\gcd(s, q) = 1 \quad s^{-1}$ exists

$s^{-1}\, m \equiv -\, a\, r\, s^{-1} + k \pmod{q}$

$k \equiv s^{-1}\, m + a\, r\, s^{-1} \equiv u_1 + a\, u_2 \pmod{q}$

$r \equiv \alpha^k \pmod{p} \pmod{q}$

$\equiv \alpha^{u_1 + a\, u_2 + i\, q} \pmod{p} \pmod{q}$

$\equiv \alpha^{u_1}\, \beta^{u_2}\, \alpha^{i\, q} \pmod{p} \pmod{q}$

$\equiv \alpha^{u_1}\, \beta^{u_2} \pmod{p} \pmod{q}$ $\qquad \alpha^q \equiv 1 \pmod{p}$

$\equiv v \pmod{p} \pmod{q}$

Security of DSA

- ✧ ***a*** must be kept secret
- ✧ ***k*** can not be used twice (same as ElGamal)
- ✧ partial information leaked from β
  - ✶ let $p-1 = t \cdot q$ and g is a primitive root modulo p, if t has only small prime factors, given $g^a \pmod p$, ***a* (mod *t*)** can be calculated by Pohlig-Hellman algorithm
  - ✶ $\alpha \equiv g^t \pmod p$ (i.e. $\alpha \equiv g^{p-1/q} \pmod p$, $\alpha^q \equiv 1 \pmod p$)
    $\beta \equiv \alpha^a \equiv g^{ta} \pmod p$ i.e. $L_g(\beta) \equiv 0 \pmod t$
    no information leaked by β about $L_g(\beta)$ is useful even if all prime factors of t are relatively small
  - ✶ $a \equiv L_\alpha(\beta) \equiv L_g(\beta) / t \pmod{p-1}$, therefore, no information of $L_\alpha(\beta)$ leaked by β is useful

Computation of DSA

- ✧ **mod exp** is $O(n^3)$
- ✧ bit length: q: 160 bits p: n bits
  - ✶ ElGamal $v_1 = \alpha^r \beta^s \pmod p$ $v_2 = \alpha^m \pmod p$
    where $\alpha, \beta, r, s, m, v_1, v_2, p$ are all n bits
  - ✶ DSA $v \equiv (\alpha^{u_1}\beta^{u_2} \pmod p) \pmod q$
    where α, β, p are n bits, u_1, u_2, v, q are 160 bits
- ✧ overall verification computations
  - ✶ ElGamal: $O(3 \cdot n^3)$
  - ✶ DSA: $O(2 \cdot n^2 \cdot 160)$

Other Signature Related Algorithms

- ✧ Group Signature
- ✧ Undeniable Signature (Nontransferable Signature)
- ✧ Designated Confirmer Signature
- ✧ Ring Signature
- ✧ Multi-Party Digital Signature

Other topics

- ✧ Security notions of signature schemes
- ✧ Schnorr signature scheme
- ✧ DSS and ElGamal are not provably secure
- ✧ First encryption or first signature?