DTTF/NB479: Dszquphsbqiz Day 26

	Chapter	Topic	People			
16-Apr	---	Bitcoin	kampernj	mcdonamp	oliverr	shinnsm
18-Apr		Elliptic cur	richarnj	strullsd	wallersb	yochmake
18-Apr	10?	Protocols	abdelroh	hopkinaj	mercermt	michaeaj
19-Apr	15	Info Theory	kraevam	reynolza	taos	trammjn
19-Apr		Dig Cash	chenaurj	dingx	graetzer	riechelp
19-Apr	19	Quantum	earlesja	gartzkds	kessledi	priceha
Late	14/18	0Know, EC	cooperra	zhangr1		

- Announcements:
  1. HW6 due now
  2. HW7 posted
  3. Will pick pres dates Friday
- Questions?
- This week:
  - Discrete Logs, Diffie-Hellman, ElGamal
  - Hash Functions, SHA1, Birthday attacks

Name: ______________________

ElGamal

Bob publishes (α, p, β), where $1 < m < p$ and $\beta=\alpha^a$

Alice chooses secret k, computes and sends to Bob the pair (r,t) where

$r=\alpha^k$ (mod p)

$t = \beta^k m$ (mod p)

Bob finds: $tr^{-a}=m \pmod p$

Notes:

1. Show that Bob's decryption works

   Plug in values for t, r, and β.

2. Eve would like to know k. Show that knowing k allows decrpytion. Why?

   $m=\beta^{-k}t$

3. Why can't Eve compute k from r or t?

   Need to calculate a discrete log to do so, which is hard when p is large

4. Challenge: Alice should randomize k each time. If not, and Eve gets hold of a plaintext / ciphertext (m_1, r_1, t_1), she can decrypt other ciphertexts (m_2, r_2, t_2). Show how.

   Use m_1, t_1 to solve for β^k. Then use β^{-k} and t_2 to find m_2

5. If Eve says she found m from (r,t), can we verify that she really found it, using only the public key (and not k or a)? Explain.

   Not easily (see next slide)

Known plaintext attack

Bob publishes (α, p, β), where 1 < m < p and $\beta=\alpha^a$

Alice chooses secret k, computes and sends to Bob the pair (r,t) where

- $r=\alpha^k$ (mod p)
- $t = \beta^k m$ (mod p)

Bob finds: $tr^{-a}=m$ (mod p)

- Why does this work?

- If Eve got hold of a plaintext/ciphertext (m_1, r_1, t_1), she can decrypt other ciphertexts (m_2, r_2, t_2):

Answer:

- $r=\alpha^k$ (mod p), $t_1 = \beta^k m_1$ (mod p), $t_2 = \beta^k m_2$ (mod p)
- So

$$\frac{t_1}{m_1} \equiv \beta^k \equiv \frac{t_2}{m_2} \pmod{p}$$

- You can solve for m_2, since everything else in the proportion is known.

Alice should randomize k each time.

Tying everything together

Bob publishes (α, p, β), where 1 < m < p and $\beta=\alpha^a$
Alice chooses secret k, computes and sends to Bob the pair (r,t) where

- $r=\alpha^k$ (mod p)
- $t = \beta^k m$ (mod p)

Bob finds: $tr^{-a}=m \pmod p$

- Why does this work?

- If Eve says she found m from (r,t), can we verify that she really found it, using just m,r,t and the public key?

Not easily!

Decision D-H ←→Validity of (mod p) ElGamal ciphertexts.

Computational D-H ←→Decrypting (mod p) ElGamal ciphertexts.

Cryptographic hash functions shrink messages into a digest

Message m
(long)

Cryptographic hash
Function, *h*

Message digest, y
(Shorter fixed length)

Shrinks data, so 2 messages can have the same digest: m_1 != m_2, but $h(m_1) = h(m_2)$

- Goal: to provide a unique “fingerprint” of the message.

Cryptographic hash functions must satisfy three properties to be useful and secure

Message m (long)

Cryptographic hash Function, *h*

Message digest, y (Shorter fixed length)

Shrinks data, so 2 messages can have the same digest: m_1 != m_2, but $h(m_1) = h(m_2)$

1. Fast to compute y from m.
2. One-way: given y = h(m), can't find any m' satisfying h(m') = y easily.
3. Strongly collision-free: Can't find any pair $m_1 \neq m_2$ such that $h(m_1)=h(m_2)$ easily
4. (Sometimes we can settle for weakly collision-free: given m, can't find m' ≠ m with h(m) = h(m').)

Hash functions can be used for digital signatures and error detection

3 properties:

1. Fast to compute
2. One-way: given y = h(m), can't find *any* m' satisfying h(m') = y easily.
3. Strongly collision-free: Can't find $m_1 \neq m_2$ such that $h(m_1)=h(m_2)$

Why do we care about these properties?

Use #1: Digital signatures

- If Alice signs h(m), what if Bob could find m' ≠ m, such that h(m) = h(m')?
- He could claim Alice signed m'!
- Consider two contracts…

Use #2: Error detection – simple example:

Alice sends (m, h(m)), Bob receives (M, H). Bob checks if H=h(M). If not, there's an error.

Hash function examples

3 properties:

1. Fast to compute
2. One-way: given y = h(m), can't find *any* m' satisfying h(m') = y easily.
3. Strongly collision-free: Can't find $m_1 \neq m_2$ such that $h(m_1)=h(m_2)$

Examples:

1. h(m) = m (mod n)
2. $h(m) = \alpha^m$ (mod *p*) for large prime p, which doesn't divide α
3. Discrete log hash

   Given large prime p, such that q=(p-1)/2 is also prime, and primitive roots α and β for p:

$$h(m) \equiv \alpha^{m_0} \beta^{m_1} \pmod{p}$$

   where $m = m_0 + m_1 q$

- EHA (next)
- SHA-1 (tomorrow)
- MD4, MD5 (weaker than SHA; won't discuss)

For first 2 examples, please check properties 2-3.

Easy Hash Algorithm (EHA) isn't very secure!

- Break m into n-bit blocks, append zeros to get a multiple of *n*.
- There are L of them, where L =|m|/n

$$m = \begin{bmatrix} m_1 \\ m_2 \\ \ldots \\ m_l \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{l1} & m_{l2} & \cdots & m_{\mathrm{ln}} \end{bmatrix}$$

$$\begin{matrix} \oplus & \oplus & \oplus & \oplus \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \end{matrix}$$

$$\begin{bmatrix} c_1 & c_2 & \ldots & c_n \end{bmatrix} = h(m)$$

- Fast! But not very secure.
- Does performing a left shift on the rows first help?
  - Define $m \hookleftarrow y$ as left-shifting m by y bits
  - Then $m_i' = m_i \hookleftarrow (i-1)$

$$\begin{bmatrix} m_{11} & m_{12} & \ldots & m_{1n} \\ m_{22} & m_{23} & \ldots & m_{21} \\ \vdots & \vdots & \ddots & \vdots \\ m_{ll} & m_{l,l+1} & \cdots & m_{l,l-1} \end{bmatrix}$$

Easy Hash Algorithm (EHA) isn't very secure!

3 properties:

1. Fast to compute
2. One-way: given y = h(m), can't find *any* m' satisfying h(m') = y easily.
3. Strongly collision-free: Can't find $m_1 \neq m_2$ such that $h(m_1)=h(m_2)$

$$m = \begin{bmatrix} m_1 \\ m_2 \\ \dots \\ m_l \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{l1} & m_{l2} & \cdots & m_{ln} \end{bmatrix}$$

$$\begin{matrix} \oplus & \oplus & \oplus & \oplus \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \end{matrix}$$

$$\begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix} = h(m)$$

Exercise:

1. Show that the basic (unrotated) version doesn't satisfy properties 2 and 3.
2. What about the version that uses rotations?