DTTF/NB479: Dszquphsbqiz



- 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

	Chapter	Торіс	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		

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 $r = \alpha^k \pmod{p}$ $t = \beta^k \pmod{p}$

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

Notes:

- 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? $m=\beta^{-k}t$
- 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 (m₁, r₁, t₁), she can decrypt other ciphertexts (m₂, r₂, t₂). Show how. Use m₁, t₁ to solve for β^k . Then use β^{-k} and t₂ to find m₂
- 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 β = α^a
- Alice chooses secret k, computes and sends to Bob the pair (r,t) where
 - r=α^k (mod p)
 - $t = \beta^k m \pmod{p}$
- Bob finds: $tr^{-a}=m \pmod{p}$
- Why does this work?

 If Eve got hold of a plaintext/ciphertext (m₁, r₁, t₁), she can decrypt other ciphertexts (m₂, r₂, t₂):

Answer:

• $r=\alpha^k \pmod{p}$, $t_1 = \beta^k m_1 \pmod{p}$, $t_2 = \beta^k m_2 \pmod{p}$

So

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

 You can solve for m₂, 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

- $r=\alpha^k \pmod{p}$
- $t = \beta^k m \pmod{p}$

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

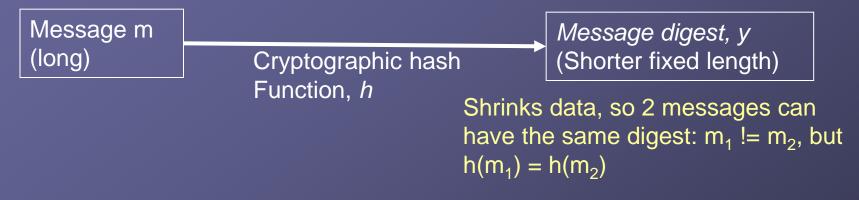
Why does this work?

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 $\leftarrow \rightarrow$ Validity of (mod p) ElGamal ciphertexts.

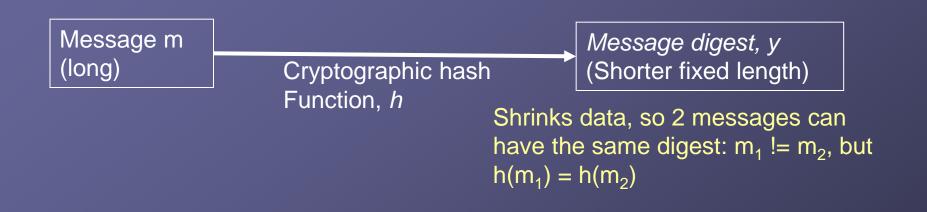
Computational D-H $\leftarrow \rightarrow$ Decrypting (mod p) ElGamal ciphertexts.

Cryptographic hash functions shrink messages into a digest



Goal: to provide a unique "fingerprint" of the message.

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



- 1. Fast to compute y from m.
- 2. One-way: given y = h(m), can't find any m' satisfying h(m') = y easily.
- 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' \neq m with h(m) = h(m').

Hash functions can be used for digital signatures and error detection

3 properties:

- 1. Fast to compute
- One-way: given y = h(m), can't find any m' satisfying h(m') = y easily.
- 3. Strongly collisionfree: 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

4a-b

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:

- $h(m) = m \pmod{n}$
- 2. $h(m) = \alpha^m \pmod{p}$ for large prime p, which doesn't divide α
 - 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)

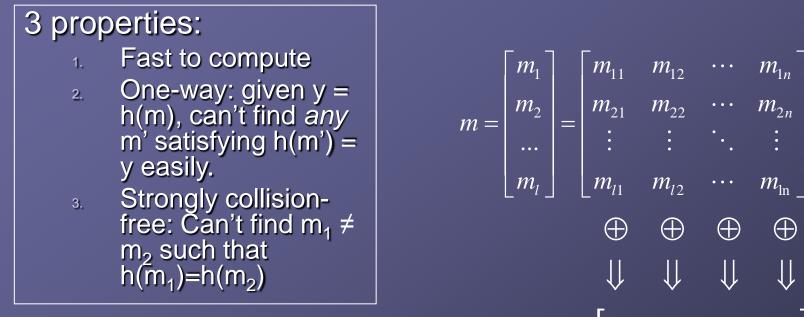
For first 2 examples, please check properties 2-3.

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

	m_1	m_{11}	$egin{array}{c} m_{12}\ m_{22}\ dots\ m_{l2}\ m_{l2} \end{array}$	•••	m_{1n}	
m —	m_2	<i>m</i> ₂₁	<i>m</i> ₂₂	•••	m_{2n}	
<i>III</i> –	•••	•	:		•	
	m_l	m_{l1}	m_{l2}	•••	$m_{\rm ln}$	
		\oplus	\oplus	\oplus	\oplus	
		\downarrow	\downarrow	\downarrow	\downarrow	
		c_1	<i>C</i> ₂	•••	C_n	=h(m)

m_{11}	m_{12}	•••	m_{1n}
$m_{22}^{}$	m_{23}	•••	m_{21}
:	•		:
m_{ll}	$m_{l,l+1}$	•••	$m_{l,l-1}$



Exercise:

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