

# Exam 2 review + Sad goodbyes ☹

exam  
thursday @ 12

MACs  
hash funcs  
RSA  
DH + ElGamal  
Signatures

HW7 #2

given  $g^{(a+1)(b+1)}$   
given  $g^a, g^b, g^{a+1}, \dots$   
compute  $\boxed{g^{ab}}$ ,  $g^{(a+2)(b+2)}$

$$\begin{aligned} g_{\text{known}}^{(a+1)(b+1)} &= g^{ab + a + b + 1} \\ &= (g^{ab}) (g^a) (g^b) (g) \end{aligned}$$

known

$$g^{ab} \equiv_p g^{(a+1)(b+1)} [g^a \cdot g^b \cdot g]^{-1}$$

mult.  
inverse  
mod  $p$

## HWS #2

$$\text{Enc}(k, m) \parallel \text{MAC}(k, \underline{m})$$

instead of c

2 ctxts that encrypt same  $m$

$\Rightarrow$  same MAC

So not even CPA secure !! ☹

## Number theory:

$$\mathbb{Z}_n \text{ vs } \mathbb{Z}_n^* \quad \dots \quad \phi(n) = |\mathbb{Z}_n^*|$$

$\uparrow$  everything

$\uparrow$  things that you  
can "divide by" mod  $n$   
( $x^{-1}$  exists)

$$\phi(p) = p - 1$$

$$\phi(pq) = (p-1)(q-1)$$

$$x^y \bmod n \iff x^{y \bmod \phi(n)} \bmod n$$

e.g.  $x^{101386420} \equiv 1$

$$N = pq$$

$$ed \equiv_{\varphi(N)} 1$$

CRT!

$$\mathbb{Z}_{pq} \hookrightarrow \mathbb{Z}_p \times \mathbb{Z}_q$$

$$x \mapsto (x^{e_p}, x^{e_q})$$

$g$  prim root mod  $n$

$$\Leftrightarrow \{g^0, g^1, g^2, \dots\} = \mathbb{Z}_n^*$$

1. [10 points; 2 per part] True/false

(T) F 21 has a multiplicative inverse mod 100  $21 \cdot \boxed{81} \equiv_{100} 1$   
 $\gcd(21, 100) = 1 \Rightarrow 21$  has inverse

(T) F The best known brute-force attack for the following problem requires  $2^n$  effort:

Given  $y \in \{0, 1\}^n$ , find an  $x \in \{0, 1\}^*$  such that  $H(x) = y$  (where  $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$  is a hash function).

target collision?

Find Someone  
w/ birthday  
JUL 17

T (F) The best known brute-force attack for the following problem requires  $2^n$  effort:

Find  $x, x' \in \{0, 1\}^*$  such that  $x \neq x'$  and  $H(x) = H(x')$  (where  $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$  is a hash function).

"general" collision  
 $2^{n/2}$

Find 2 people  
w/ same  
birthday

T (F) 4 is a primitive root mod 11.

T (F) The RSA exponents  $e$  and  $d$  must be chosen so that  $ed \equiv_{\phi(N)} 1$ , where  $N$  is the RSA modulus.

should  
be  $\phi(N)$

$$\{4, 4^2, 4^3, 4^4, \dots\} = \mathbb{Z}_{11}^*$$

5 9 3 1

$\xrightarrow{\times 4} \xrightarrow{\times 4} \xrightarrow{\times 4}$

"  
10 things

only 5  
can be written  
as  $4^x$

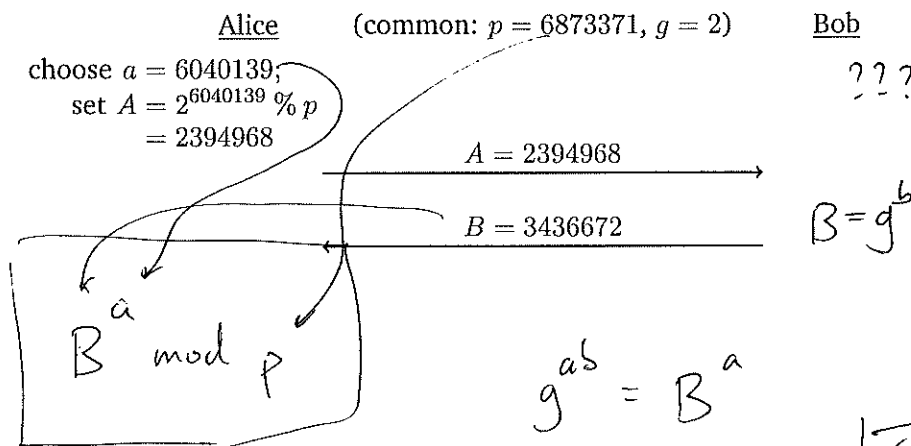
## 2. [20 points; 5 per part] Short answer:

- (a) The Chinese Remainder Theorem describes a bijection (isomorphism)  $\mathbb{Z}_6 \rightarrow \mathbb{Z}_2 \times \mathbb{Z}_3$ . Write down the bijection:

$$\begin{array}{lll}
 0 \leftrightarrow (0, 0) & 2 \leftrightarrow (0, 2) & 4 \leftrightarrow (0, 1) \\
 1 \leftrightarrow (1, 1) & 3 \leftrightarrow (1, 0) & 5 \leftrightarrow (1, 2)
 \end{array}$$

$x \leftrightarrow (x \% 2, x \% 3)$  general rule

- (b) In this execution of DHKA, only Alice's view of the protocol is shown. What computation will Alice perform to obtain the shared key?



- (c) How many elements are in  $\mathbb{Z}_{77}^*$ ? You don't have to write them all.

$$77 = 7 \times 11$$

$$\phi(77) = (7-1)(11-1)$$

- (d) Write the encryption algorithm for any CCA-secure symmetric encryption scheme of your choice. You can assume you have a PRP/PRF  $F$ , and a MAC function  $M$  (supporting arbitrary length inputs). You do not have to give the decryption algorithm.

(CPA) Enc - then - MAC

$$\text{Enc}((k, k')_m)$$

$$\begin{array}{l}
 \left\{ \begin{array}{l}
 r \leftarrow \{0, 1\}^A \\
 s = F(k, r) \oplus m \\
 t = \text{MAC}(k', (r, s))
 \end{array} \right\} \text{CPA Enc of } m \\
 \text{ret } (r, s, t)
 \end{array}$$

3. [20 points] Below is OFB mode for (symmetric-key) encryption. Show that it does **not** have CCA security.

OFB.Enc( $k, m_1 \dots m_\ell$ ):

$r \leftarrow \{0, 1\}^{blen}$

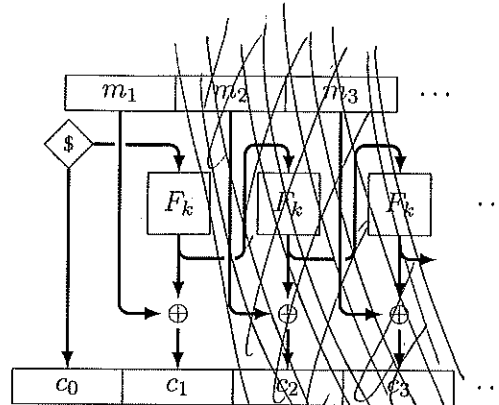
$c_0 := r$

for  $i = 1$  to  $\ell$ :

$r := F(k, r)$

$c_i := r \oplus m_i$

return  $c_0 \dots c_\ell$



$\mathcal{L}_{cca-L}^\Sigma$

$k \leftarrow \Sigma.\text{KeyGen}$   
 $S := \emptyset$

CHALLENGE( $m_L, m_R \in \Sigma.\mathcal{M}$ ):

if  $|m_L| \neq |m_R|$  return null  
 $c := \Sigma.\text{Enc}(k, m_L)$   
 $S := S \cup \{c\}$   
 return  $c$

DEC( $c \in \Sigma.\mathcal{C}$ ):

if  $c \in S$  return null  
 return  $\Sigma.\text{Dec}(k, c)$

$\mathcal{L}_{cca-R}^\Sigma$

$k \leftarrow \Sigma.\text{KeyGen}$   
 $S := \emptyset$

CHALLENGE( $m_L, m_R \in \Sigma.\mathcal{M}$ ):

if  $|m_L| \neq |m_R|$  return null  
 $c := \Sigma.\text{Enc}(k, m_R)$   
 $S := S \cup \{c\}$   
 return  $c$

DEC( $c \in \Sigma.\mathcal{C}$ ):

if  $c \in S$  return null  
 return  $\Sigma.\text{Dec}(k, c)$

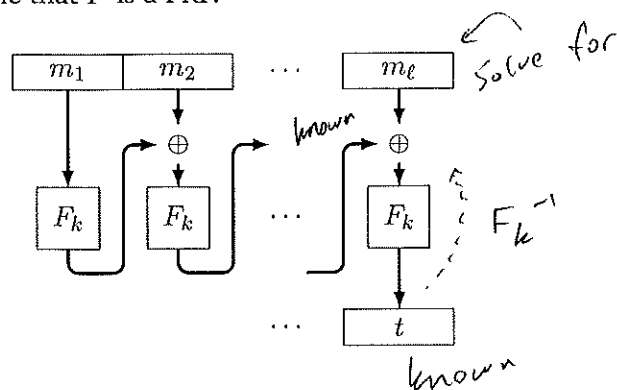
Idea: ① ask for enc of unknown ptxt  
 ② fiddle with ctxt  $\rightsquigarrow$  ctxt'  
 get Dec of ctxt'  
 ...  
 profit: can tell what original ptxt was

- ①  $(c_0, c_1) \leftarrow \text{challenge}(0^\lambda, 1^\lambda)$
- ②  $(c'_0, c'_1) = (c_0, c_1 \oplus x)$  (any  $x \neq 0^\lambda$ )
- ③  $m^* = \text{Dec}(c'_0, c'_1)$
- ④ check  $m^* = x \oplus 0^\lambda$
- $c_1 = F(k, c_0) \oplus m$   
 $\Rightarrow m = \underbrace{F(k, c_0)}_{\text{Dec}} \oplus c_1$

4. [20 points] Show how someone **who knows  $k$**  can find two different messages whose CBC-MAC values are the same. In other words, CBC-MAC is not a collision-resistant hash function (interpreting  $k$  as the salt). You can assume that  $F$  is a PRP.

HW6 #1

$\text{CBCMAC}^F(k, m_1 \dots m_\ell):$   
 $t := 0^\lambda;$   
 for  $i = 1$  to  $\ell$ :  
 $t := F(k, m_i \oplus t)$   
 return  $t$



5. [20 points] For reference, here is ElGamal encryption.  $\mathbb{G} = \langle g \rangle$  is a cyclic group with generator  $g$  and  $n$  elements.

	KeyGen:	Enc( $A, M \in \mathbb{G}$ ):	Dec( $a, (B, X)$ ):
$\mathcal{M} = \mathbb{G}$	$sk := a \leftarrow \mathbb{Z}_n$	$b \leftarrow \mathbb{Z}_n$	return $X(B^a)^{-1}$
$\mathcal{C} = \mathbb{G}^2$	$pk := A := g^a$	$B := g^b$	
	return $(pk, sk)$	return $(B, M \cdot A^b)$	

Suppose you are given an ElGamal ciphertext  $(B, X)$  that encrypts an unknown plaintext  $M \in \mathbb{G}$ . Describe how to generate another ElGamal ciphertext that decrypts to  $M^2$ .

For **full points**, show how to do it **without knowing  $M$** . For partial points, you can show how to do it assuming knowledge of  $M$ .

given:  $(B, M \cdot A^b) = (g^b, M g^{ab})$

want:  $(g^{b'}, M^2 g^{ab'}) \leftarrow ??$

idea: square  $(M g^{ab})^2 \rightsquigarrow M^2 g^{2ab}$   
 $= M^2 g^{a(2b)}$

so  $b' = 2b$

$(B, X) \rightsquigarrow (B^2, X^2)$   
 $= (g^{2b}, M^2 (g^a)^{2b})$   
 $\Rightarrow$  a valid enc of  $M^2$

6. [20 points] Bob chooses an RSA plaintext  $m \in \mathbb{Z}_N$  and encrypts it under Alice's public key as  $c \equiv_N m^e$ . Alice uses the CRT technique to decrypt: that is, she computes  $m_p \equiv_p c^d$  and  $m_q \equiv_q c^d$ , and uses CRT conversion to obtain  $m \in \mathbb{Z}_N$ .

Suppose Alice is using faulty hardware, so that she computes a *wrong value* for  $m_q$ . The rest of the computation happens correctly, and Alice computes the (wrong) result  $\hat{m}$ . **Show how Bob can factor  $N$  if he learns  $\hat{m}$ , no matter what  $m$  is, and no matter what Alice's computational error was.**

Hint: Bob knows  $m$  and  $\hat{m}$  satisfying the following:

$$\begin{aligned} m &\equiv_p \hat{m} \\ m &\not\equiv_q \hat{m} \end{aligned}$$

$$\begin{aligned} m - \hat{m} &\equiv_p 0 \\ m - \hat{m} &\not\equiv_q 0 \end{aligned}$$

$$p = \gcd(m - \hat{m}, N)$$

$m$

$m - \hat{m}$  is mult of  $p$   
 $m - \hat{m}$  is not mult of  $q$