

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 g^{ab} , $g^{(a+2)(b+2)}$

$$g^{(a+1)(b+1)} = g^{ab+a+b+1}$$

$$= (g^{ab})(g^a)(g^b)(g)$$

mult.
inverse
mod p

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

HWS #2

$$\text{Enc}(k, m) \parallel \text{MAC}(k, \underline{m}) \quad \text{instead of } c$$

2 ctxts that encrypt same m

⇒ same MAC

So not even CPA secure !! $\parallel \bigcirc$

Number Theory:

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

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

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

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

$$x^y \bmod n \longleftrightarrow x^{y \bmod \varphi(n)} \bmod n$$

e.g. $x^{101386420} \equiv_{\parallel} 1$

$$N = pq$$

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

CRT !

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

$$x \mapsto (x \%_p, x \%_q)$$

g prim root mod n

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

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

T F 21 has a multiplicative inverse mod 100

$$\gcd(21, 100) = 1 \Rightarrow 21 \text{ has inverse}$$

$$21 \cdot \boxed{81} \equiv_{100} 1$$

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

Find 2 people w/ same birthday

T F 4 is a primitive root mod 11.

$$2^{n/2}$$

T F The RSA exponents e and d must be chosen so that $ed \equiv_N 1$, where N is the RSA modulus.

should
be $\varphi(N)$

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

$$5 \quad 9 \quad 3 \quad 1$$

$$\xrightarrow{x^4} \xrightarrow{x^4} \xrightarrow{x^4}$$

"

10 things

only 5
can be written

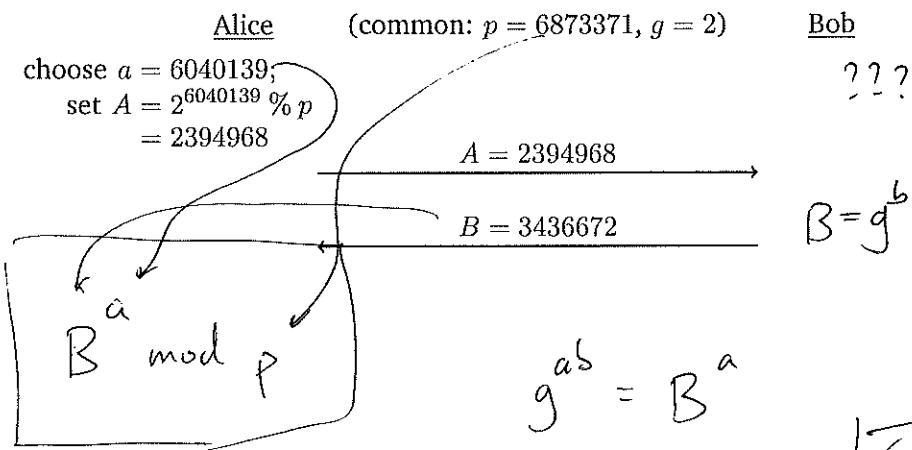
$$\text{as } 4^x$$

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:

$$\begin{array}{lll} 0 \leftrightarrow (0,0) & 2 \leftrightarrow (0,2) & 4 \leftrightarrow (0,1) \\ 1 \leftrightarrow (1,1) & 3 \leftrightarrow (1,0) & 5 \leftrightarrow (1,2) \\ X \leftrightarrow (x_{\mathbb{Z}_2}, x_{\mathbb{Z}_3}) & & \text{general rule} \end{array}$$

- (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.

$$77 = 7 \times 11 \quad \varphi(77) = (7-1)(11-1)$$

* 60 *

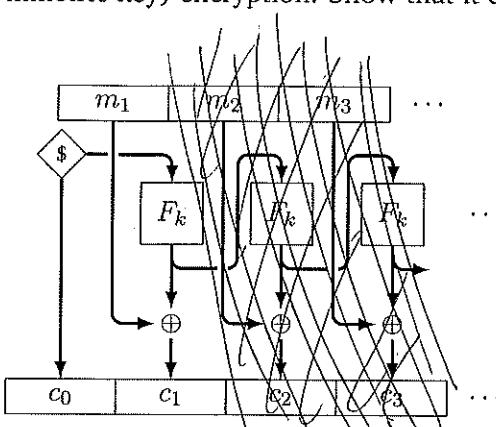
- (d) Write the encryption algorithm for any CCA-secure symmetric encryption scheme of your choice. You can assume you have a PRP/PRF F , and a MAC function M (supporting arbitrary length inputs). You do not have to give the decryption algorithm.

(CPA) Enc - then - MAC

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

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

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

$$\begin{aligned} r &\leftarrow \{0, 1\}^{\text{blen}} \\ c_0 &:= r \\ \text{for } i = 1 \text{ to } \ell: \\ &\quad r := F(k, r) \\ &\quad c_i := r \oplus m_i \\ &\quad \text{return } c_0 \dots c_\ell \end{aligned}$$


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

$$\begin{aligned} k &\leftarrow \Sigma.\text{KeyGen} \\ \mathcal{S} &:= \emptyset \\ \text{CHALLENGE}(m_L, m_R \in \Sigma.M): \\ &\quad \text{if } |m_L| \neq |m_R| \text{ return null} \\ &\quad c := \Sigma.\text{Enc}(k, m_L) \\ &\quad \mathcal{S} := \mathcal{S} \cup \{c\} \\ &\quad \text{return } c \\ \text{DEC}(c \in \Sigma.C): \\ &\quad \text{if } c \in \mathcal{S} \text{ return null} \\ &\quad \text{return } \Sigma.\text{Dec}(k, c) \end{aligned}$$

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

$$\begin{aligned} k &\leftarrow \Sigma.\text{KeyGen} \\ \mathcal{S} &:= \emptyset \\ \text{CHALLENGE}(m_L, m_R \in \Sigma.M): \\ &\quad \text{if } |m_L| \neq |m_R| \text{ return null} \\ &\quad c := \Sigma.\text{Enc}(k, m_R) \\ &\quad \mathcal{S} := \mathcal{S} \cup \{c\} \\ &\quad \text{return } c \\ \text{DEC}(c \in \Sigma.C): \\ &\quad \text{if } c \in \mathcal{S} \text{ return null} \\ &\quad \text{return } \Sigma.\text{Dec}(k, c) \end{aligned}$$

Idea: ① ask for Enc of unknown ptxt
 ② fiddle with ctxt → ctxt'
 get Dec of ctxt'

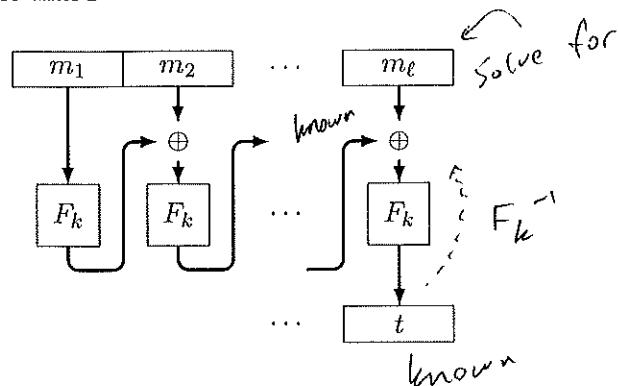
prof+ can tell what original ptxt was

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

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

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

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

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: $(g^{b'}, M^2 g^{a b'})$ ↗??

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

so $b' = 2b$

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

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:

