FINAL REVIEW-ADMIN

Final Review-Admin

1) Final is Saturday Dec 15 1:30-3:30 in CSI 2117 (usual class)
2) Can bring one sheet of notes.
    Can use both sides
    Can be typed
    You can put whatever you want on it.
    Can copy a classmates and use it but thats stupid
    Can try to cram the entire course onto it but thats stupid
3) No calculators allowed. Numbers will be small.
4) Coverage: Slides/HW.
5) Not on Exam: Guest lectures with two exceptions: LWE (as I did it), Bitcoin.
6) We hope to grade it and post it Saturday Afternoon.
7) If can't take the exam tell me ASAP.
8) Advice: Understand rather than memorize.

FINAL REVIEW-CONTENT

Alice, Bob, and Eve

- Alice sends a message to Bob in code.
- Eve overhears it.
- We want Eve to not get any information.

There are many aspects to this:

- Information-Theoretic Security.
- Computational-Theoretic Security (Hardness Assumption)
- The NY,NY problem: Do not always code m the same way. If always coded same way then CPA-insecure
- Private Key or Public key
- Kerckhoff's principle: Eve knows cryptosystem.
- History: How much computing power does Eve have?

Private Key Ciphers

Single Letter Sub Ciphers

1. Shift cipher: $f(x) = x + s$. $s \in \{0, \ldots, 25\}$.
2. Affine cipher: $f(x) = ax + b$. $a, b \in \{0, \ldots, 25\}$. a rel prime 26.
3. Keyword Shift: From keyword and shift create random-looking perm of $\{a, \ldots, z\}$.
4. Keyword Mixed: From keyword create random-looking perm of $\{a, \ldots, z\}$.
5. Gen Sub Cipher: Take random perm of $\{a, \ldots, z\}$.

All Single Letter Sub Ciphers Crackable

Important: Algorithm Is-English.

1. Input(T) a text
2. Find f_T, the freq vector of T
3. Find $x = f_T \cdot f_E$ where f_E is freq vector for English
4. If $x \geq 0.06$ then output YES. If $x \leq 0.04$ then output NO. If $0.04 < x < 0.06$ then something is wrong.

1. Shift , Affine have small key space: can try all keys and see when Is-English says YES.
2. For others use Freq analysis, e.g., e is most common letter.
3. If message is numbers (e.g., Credit Cards) or ASCII (e.g., Byte-Shift) there are still patterns so can use freq analysis.

Randomized Shift

How to NOT encode the same m the same way:

Randomized shift: Key is a function $f : S \to S$.

1. To send message $(m_1, \ldots, m_L)$ (each m_i is a character)
   - 1.1 Pick random $r_1, \ldots, r_L \in S$. For $1 \leq i \leq L$ compute $s_i = f(r_i)$.
   - 1.2 Send $((r_1; m_1 + s_1), \ldots, (r_L; m_L + s_L))$
2. To decode message $((r_1; c_1), \ldots, (r_L; c_L))$
   - 2.1 For $1 \leq i \leq L$ $s_i = f(r_i)$.
   - 2.2 Find $(c_1 - s_1, \ldots, c_L - s_L)$

   Note: Can be cracked.

The Vigenère cipher

Key: A word or phrase. Example: *dog = (3,14,6)*.
Easy to remember and transmit.
Example using *dog*.
Shift 1st letter by 3
Shift 2nd letter by 14
Shift 3nd letter by 6
Shift 4th letter by 3
Shift 5th letter by 14
Shift 6th letter by 6, etc.

Jacob Prinz is a Physics Major
Jacob Prinz isaPh ysics Major

encrypts to

MOIRP VUWTC WYDDN BGOFG SDXUU

Can be cracked by guessing key length L and doing Freq Anal on every Lth letter.

The Matrix Cipher

Definition: Matrix Cipher. Pick n and M an $n \times n$ invertible matrix.

1. Encrypt via $\vec{x} \to M(\vec{x})$.
2. Decrypt via $\vec{y} \to M^{-1}(\vec{y})$

We'll take $n = 30$.

The Matrix Cipher

Definition: Matrix Cipher. Pick n and M an $n \times n$ invertible matrix.

1. Encrypt via $\vec{x} \to M(\vec{x})$.
2. Decrypt via $\vec{y} \to M^{-1}(\vec{y})$

We'll take $n = 30$.

1. Easy for Alice and Bob.
2. Key M is small enough to be easy for Alice and Bob but too large for Eve to use brute force.
3. Eve can crack using freqs of 30-long sets of letters? Hard?
4. Ciphertext only might be uncrackable.
5. Can crack from message-cipher pairs.

One-time pad

1. Let $\mathcal{M} = \{0,1\}^n$
2. *Gen*: choose a uniform key $k \in \{0,1\}^n$
3. $Enc_k(m) = k \oplus m$
4. $Dec_k(c) = k \oplus c$
5. Correctness:

$$\begin{aligned} Dec_k(Enc_k(m)) &= k \oplus (k \oplus m) \\ &= (k \oplus k) \oplus m \\ &= m \end{aligned}$$

PROS AND CONS Of One-time pad

1. If Key is N bits long can only send N bits.
2. $\oplus$ is FAST!
3. The one-time pad is uncrackable. YEAH!
4. Generating truly random bits is hard. BOO!
5. Psuedo-random can be insecure – I did example.

Public Key Ciphers
Eve can go . . .

Public Key Cryptography

Alice and Bob never have to meet!

NT Algorithms needed for Public Key

All arithmetic is mod p. The following can be done quickly.

1. Given (a, n, p) compute $a^n \pmod{p}$. Repeated Squaring. (1) $\leq 2 \lg n$ always, (2) $\leq \lg n + O(1)$ if n close to 2^{2^m}.
2. Given n, find a safe prime of length n and a generator g.
3. Given a, b rel prime find inverse of a mod b: Euclidean alg.
4. Given $a_1, \ldots, a_L$ and $b_1, \ldots, b_L$, b_i's rel prime, find $x \equiv a_i \pmod{b_i}$.
5. Given (a, p) find $\sqrt{a}$'s. We did $p \equiv 3 \pmod{4}$ case.
6. Given (a, N) and p, q such that $N = pq$, find $\sqrt{a}$'s.

Number Theory Assumptions

1. Discrete Log is hard.
2. Factoring is hard.
3. Given (a, N), find $\sqrt{a}$ without being given factors of N is hard. (This is equiv to factoring.)

Note: We usually don't assume these but instead assume close cousins.

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s.

1. Alice finds a (p, g), p of length n, g gen for $\mathbb{Z}_p$. Arith mod p.
2. Alice sends (p, g) to Bob in the clear (Eve can see it).
3. Alice picks random $a \in \{1, \ldots, p-1\}$. Alice computes g^a and sends it to Bob in the clear (Eve can see it).
4. Bob picks random $b \in \{1, \ldots, p-1\}$. Bob computes g^b and sends it to Alice in the clear (Eve can see it).
5. Alice computes $(g^b)^a = g^{ab}$.
6. Bob computes $(g^a)^b = g^{ab}$.
7. g^{ab} is the shared secret.

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s.

1. Alice finds a (p, g), p of length n, g gen for $\mathbb{Z}_p$. Arith mod p.
2. Alice sends (p, g) to Bob in the clear (Eve can see it).
3. Alice picks random $a \in \{1, \ldots, p-1\}$. Alice computes g^a and sends it to Bob in the clear (Eve can see it).
4. Bob picks random $b \in \{1, \ldots, p-1\}$. Bob computes g^b and sends it to Alice in the clear (Eve can see it).
5. Alice computes $(g^b)^a = g^{ab}$.
6. Bob computes $(g^a)^b = g^{ab}$.
7. g^{ab} is the shared secret.

Definition

Let f be $f(p, g, g^a, g^b) = g^{ab}$.

Hardness assumption: f is hard to compute.

ElGamal is DH with a Twist

1. Alice and Bob do Diffie Helman.
2. Alice and Bob share secret $s = g^{ab}$.
3. Alice and Bob compute $(g^{ab})^{-1} \pmod{p}$.
4. To send m, Alice sends $c = mg^{ab}$
5. To decrypt, Bob computes $c(g^{ab})^{-1} \equiv mg^{ab}(g^{ab})^{-1} \equiv m$

We omit discussion of Hardness assumption (HW)

RSA

Let n be a security parameter

1. Alice picks two primes p, q of length n and computes $N = pq$.
2. Alice computes $\phi(N) = \phi(pq) = (p-1)(q-1)$. Denote by R
3. Alice picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R. Alice finds d such that $ed \equiv 1 \pmod{R}$.
4. Alice broadcasts (N, e). (Bob and Eve both see it.)
5. Bob: To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.
6. If Alice gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \pmod{R}} \equiv m^{1 \pmod{R}} \equiv m$$

Hardness Assumption for RSA

Definition: Let f be $f(N, e) = d$, where $N = pq$, and

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

Hardness assumption (HA): f is hard to compute.

Plain RSA Bytes!

The RSA given above is referred to as Plain RSA.
Insecure! m is always coded as $m^e \pmod N$.

Make secure by padding: $m \in \{0,1\}^{L_1}$, $r \in \{0,1\}^{L_2}$.

To send $m \in \{0,1\}^{L_1}$, pick rand $r \in \{0,1\}^{L_2}$, send $(rm)^e$.
(NOTE- rm means r CONCAT with m here and elsewhere.)
DEC: Alice finds rm and takes rightmost L_1 bits.
Caveat: RSA still has issues when used in real world. They have been fixed. Maybe.

Attacks on RSA

1. Factoring Algorithms. We saw some ideas with Jevon's Number. Response: Pick larger p, q
2. If Zelda give A_i (N_i, e):
   - 2.1 Low-e attack: Response: High e. Duh.
   - 2.2 $m^e < N_1 \cdots N_L$: Response: Pad m.
3. If Zelda give A_i (N, e_i) and two of the e_i's are rel prime, then Euclidean Alg Attack: Response: Give everyone diff N's. Duh.
4. Timing Attacks: Response: Pad the amount of time used.

Caveat: Theory says use different e's. Practice says use $e = 2^{16} + 1$ for speed.

Rabin's Encryption Scheme

n is a security parameter

1. Alice gen p, q primes of length n. Let $N = pq$. Send N.
2. Encode: To send m, Bob sends $c = m^2 \pmod{N}$.
3. Decode: Alice can find m such that $m^2 \equiv c \pmod{N}$.

Rabin's Encryption Scheme

n is a security parameter

1. Alice gen p, q primes of length n. Let $N = pq$. Send N.
2. Encode: To send m, Bob sends $c = m^2 \pmod{N}$.
3. Decode: Alice can find m such that $m^2 \equiv c \pmod{N}$. OH! There will be two or four of them! What to do? Later.

Rabin's Encryption Scheme

n is a security parameter

1. Alice gen p, q primes of length n. Let $N = pq$. Send N.
2. Encode: To send m, Bob sends $c = m^2 \pmod{N}$.
3. Decode: Alice can find m such that $m^2 \equiv c \pmod{N}$. OH! There will be two or four of them! What to do? Later.

BIG PRO: Factoring Hard is hardness assumption.
CON: Alice has to figure out which of the sqrts is correct message.

A Theorem from Number Theory

Definition: A *Blum Int* is product of two primes $\equiv 3 \pmod 4$.
Example: $21 = 3 \times 7$.

Notation: SQ_N is the set of squares mod N. (Often called QR_N.)
Example: If $N = 21$ then $SQ_N = \{1, 4, 7, 9, 15, 16, 18\}$.

Theorem: Assume N is a Blum Integer. Let $m \in SQ_N$. Then of the two or four sqrts of m, only one is itself in SQ_N.
Proof: Omitted. Note: (1) not that hard, and (2) in Katz book.

We use Theorem to modify Rabin Encryption.

Rabin's Encryption Scheme 2.0

Also called The Blum-Williams Variant of Rabin

n is a security parameter.

1. Alice gen p, q primes of length n such that $p, q \equiv 3 \pmod 4$. Let $N = pq$. Send N.
2. Encode: To send m, Bob sends $c = m^2 \pmod N$. Only send m's in SQ_N.
3. Decode: Alice can find 2 or 4 m such that $m^2 \equiv c \pmod N$. Take the $m \in SQ_N$.

CON: Messages have to be in SQ_N.

History: Had timing been different Rabin Enc would be used.

Goldwasser-Micali Encryption

n is a security parameter. Will only send ONE bit. Bummer!

1. Alice gen p, q primes of length n, and $z \in NSQ_N$. Computes $N = pq$. Send (N, z).
2. Encode: To send $m \in \{0, 1\}$, Bob picks random $x \in \mathbb{Z}_N$, sends $c = z^m x^2 \pmod N$. Note that:
   2.1 If $m = 0$ then $z^m x^2 = x^2 \in SQ_N$.
   2.2 If $m = 1$ then $z^m x^2 = zx^2 \in NSQ_N$.
3. Decode: Alice determines if $c \in SQ$ or not. If YES then $m = 0$. If NO then $m = 1$.

BIG PRO: Hardness assumption natural: SQ_N hard.

BIG CON: Messages have to be 1-bit long.

TIME: For one bit you need $4 \log N$ steps.

Blum-Goldwasser Enc. n Sec Param, L length of msg

1. Alice: p, q primes len n, $p, q \equiv 3 \pmod 4$. $N = pq$. Send N.
2. Encode: Bob sends $m \in \{0,1\}^L$: picks random $r \in \mathbb{Z}_N$
   $x_1 = r^2 \mod N \qquad b_1 = LSB(x_1)$.
   $x_2 = x_1^2 \mod N \qquad b_2 = LSB(x_2)$.
   $\vdots$
   $x_L = x_{L-1}^2 \mod N \qquad b_L = LSB(x_L)$.
   Send $c = ((m_1 \oplus b_1, \ldots, m_L \oplus b_L), x_L)$.
3. Decode: Alice: From x_L Alice can compute x_{L-1}, ..., x_1 by sqrt (can do since Alice has p, q). Then can compute $b_1, \ldots, b_L$ and hence $m_1, \ldots, m_L$.

BIG PRO: Hardness assumption is BBS psuedo-random.
TIME: For L bits need $(L+3)\log N$ steps. Better than Goldwasser-Micali.

LWE-KE

1. LWE-KE is a protocol for Key Exchange that does not rely on Number Theory Hardness Assumption
2. There is also a LWE-RSA.
3. These might be useful if Factoring and Discrete Log can be done by a quantum computer.
4. My presentation of it was not quite right.
5. The literature on these is not quite right either.

Secret Sharing

Threshold Secret Sharing

Zelda has a secret $s \in \{0,1\}^n$.

Def: Let $1 \leq t \leq m$. (t, L)-secret sharing is a way for Zelda to give strings to $A_1, \ldots, A_L$ such that:

1. If any t get together than they can learn the secret.
2. If any $t-1$ get together they cannot learn the secret.

Cannot learn the secret. Two flavors: (1) info-theoretic, (2) computational.
Note: Access Structure is a set of sets of students closed under superset. Can also look at Secret Sharing with other access structures.

Methods For Secret Sharing

Assume $|s| = n$.

1. Random String Method.
   PRO: Can be used for ANY access structure.
   CON: For Threshold Zelda may have to give Alice LOTS of strings
2. Poly Method. Uses: t points det poly of deg $t-1$.
   PRO: Zelda gives Alice a share of exactly n. Simple.
   CON: Only used for threshold secret sharing
   CAVEAT: For exactly n need weird math. Get $n+1$ with normal math.
3. Geometry. Uses: t points in $\mathbb{R}^t$ det. a $(t-1)$-hyperplane.
   PRO: Zelda gives Alice a share of exactly n. Simple.
   CON: Only used for threshold secret sharing
   CON: We didn't cover it.

Short Shares

If demand Info-theoretic security then shares have to be $\geq |s|$.

We did that in class.

So we go to comp theoretic, next slide.

Short Shares

Thm: Assume there exists an α-SES. Assume that for message of length n, it is secure. Then, for all $1 \leq t \leq L$ there is a (t, L)-scheme for $|s| = n$ where each share is of size $\frac{n}{t} + \alpha n$.

1. Zelda does $k \leftarrow GEN(n)$. Note $|k| = \alpha n$.
2. $u = ENC_k(s)$. Let $u = u_0 \cdots u_{t-1}$, $|u_i| \sim \frac{n}{t}$.
3. Let $p \sim 2^{n/t}$. Zelda forms poly over $\mathbb{Z}_p$:

$$f(x) = u_{t-1}x^{t-1} + \cdots + u_1 x + u_0$$

4. Let $q \sim 2^{\alpha n}$. Zelda forms poly over $\mathbb{Z}_q$ by choosing $r_{t-1}, \ldots, r_1 \in \{0, \ldots, q-1\}$ at random and then:

$$g(x) = r_{t-1}x^{t-1} + \cdots + r_1 x + k$$

5. Zelda gives A_i, $(f(i), g(i))$. Length: $\sim \frac{n}{t} + \alpha n$.

Verifiable Secret Sharing VSS

Cannot do it if demand info-theoretic security.
That was a HW.
So we go to comp theoretic, next slide.

Verifiable Secret Sharing

1. Secret is s, $|s| = n$. Zelda finds $p \sim n$.
2. Zelda finds a generator g for $\mathbb{Z}_p$.
3. Zelda picks rand $r_{t-1}, \ldots, r_1$, $f(x) = r_{t-1}x^{t-1} + \cdots + r_1 x + s$.
4. For $1 \leq i \leq L$ Zelda gives A_i $f(i)$.
5. Zelda gives to EVERYONE the values $g^{r_1}, \ldots, g^{r_{t-1}}, g^s, g$. (We think discrete log is HARD so r_i not revealed.)

Recover: The usual – any group of t can blah blah.
Verify: A_i reveals $f(i) = 17$. Group computes:
1) g^{17}.
2) $(g^{r_{t-1}})^{i^{t-1}} \times (g^{r_{t-2}})^{i^{t-2}} \times \cdots (g^{r_1})^{i^1} \times (g^s)^{i^0} = g^{f(i)}$
If this is g^{17} then A_i is truthful. If not then A_i is dirty stinking liar.

Provable Security and Authentication

Provable Security: Passive

Definition of Secure Cryptosystem: Defined via a game. Eve needs to determine which of m_0, m_1 was encrypted.

Secure Cryptosystem: Psuedo-one-time pad with psuedo random generator.
Better Secure Cryptosystem: Psuedo-one-time pad with Stream Cipher – an "infinite" psuedorandom sequence.

Stream Ciphers

1. Can obtain from easier assumptions theoretically. Not useful
2. Linear Feedback Shift Registers. Terrible Idea.
3. Trivium. Seems good so far.
4. Others that have been broken.
5. Others that have not been broken as far as we know.

Provable Security: CPA

Definition of CPA-Secure Cryptosystem: Defined via a game. Eve needs to determine which of m_0, m_1 was encrypted. BUT Eve can also encode her choice of texts. Encryption system needs to be random- no NY,NY.

Secure CPA-Cryptosystem: Psuedorandom function used for key. Diff key each letter.

1. Can obtain from easier assumptions theoretically. Not useful
2. Substitution-Permutation Networks (SPN)
3. Feistel Networks

Authentication

Message Authentication Codes: Tag each message with a private signature to say YES, it came from me.

Need the function that generates tag to NOT be easy to predict. Use Psuedorandom functions!

Only good for fixed length. For Variable length use Collision Resistant Hash Functions.

Digital Signatures

Public version of Authentication.
Use RSA-like system but with Collision Resistant Hash Functions.

Good Luck on the Exam

Good Luck on the Exam!