Cryptography

Cryptography

- Greek for "hidden writing"
  - The art of enciphering and deciphering codes
- In modern use mmunication
  - Much wider th ciphering
- One of the main tools for pr formation
  - Confidentiality – prevents adversaries from reading the information
  - Integrity – ensures detection of unauthorised modifications

Prime Minister claims laws of mathematics 'do not apply' in Australia

Finite Fields

- A *field* is an algebraic structure that consists of:
  - A set of elements
  - Four operations: addition, subtraction, multiplication and division
- Examples: ratio bers.
- Finite fields are fields with a ber of elements
- Example: $\mathrm{GF}(p)$ - Integers modulo a prime number p

Example: GF(7)

- Seven elements: 0, 1, 2, 3, 4, 5, 6
- Arithmetic:
  - 1+1=?
  - 3+3=?
  - 5+5=?
  - 3·2=?
  - 4·2=?
  - 1/2=?

Exponentiation

- Exponentiation: repeated multiplication
  - $x^0 = 1$
  - $x^{i+1} = x \cdot x^i$
- What is 3^2 in $\mathbb{G}$
- Can we do that efficiently w umbers?
  - … e.g. 1000 digit numbers?

A look at binary numbers

- A binary number e is a sequence of bits $e_0 \ldots e_{n-1}$ such that $e = \sum_{i=0}^{n-1} e_i \cdot 2^i$

- What is $\lfloor e/2^k \rfloor$?

- What about $\lfloor e/2^{k-1} \rfloor$?

$$\left\lfloor e/2^{k-1} \right\rfloor = \sum_{i=k-1}^{n-1} e_i \cdot 2^{i-k+1} = 2 \cdot \left\lfloor e/2^k \right\rfloor + e_{k-1}$$

Square and Multiply

$$\left\lfloor e/2^{k-1} \right\rfloor = 2 \cdot \left\lfloor e/2^{k} \right\rfloor + e_{k-1}$$

$$= \left(b^{\left\lfloor e/2^{k} \right\rfloor} \right)^{2} \cdot b^{e_{k-1}}$$

$x \leftarrow 1$
for $i \leftarrow |e|\text{-}1$ **downto** 0 **do**
  $x \leftarrow x^2 \bmod p$
  1) **then**
    $xb \bmod p$
done
return x

Logarithms

- Reverse of exponentiation
  - What is $\log_3(6)$ in $GF(7)$?

Discrete l problem!

No efficient algo wn

Key pairs

- Agree on a finite field $\mathrm{GF}(p)$ and a generator g
- Keys come in pairs
  - Represent a DLP problem

(public, private) od p

- Oscar (the adversary) knows A. Why can't he find α
- Discrete logarithm is hard.
  - If p is a 3072 bit prime, Bob needs to test ~2^{128} values to find α

Identity

- Identity means **holding a private key**
- How do we prove identity?
  - How does Bob *v* lice?
- In our settings, Alice claims/ ntity by publishing ("*committing*") a public key A from a pair (A, α)

Identification

$(A, \alpha) = \text{keypair}()$

A

???

$s = \alpha^s$

$=? g^s \pmod{p}$

- Problem: Alice no longer has an identity

Ephemera

$(A, \alpha) = \text{keypair}()$

A

???

$(R, r) = \text{keyp}$

R

$s = \alpha + r$

s

$? \; g^s \pmod{p}$

- Bob verifies because

$$g^s = g^{\alpha+r} = g^{\alpha} \cdot g^r = A \cdot R \pmod{p}$$

- Note: s reveals nothing about α because r is random

Ephemera

$(A, \alpha) = \text{keypair}()$

A Assignment Project Exam Help

???

$(R, r) = \text{keyp}$ https://eduassistpro.github.io/

R

$s = \alpha + r$

s Add WeChat edu_assist_pro

$? \; g^s \pmod{p}$

- Problem: Replay attack
  - Will solve later

Cheating

$(A, \alpha) = \text{keypair}()$

A →

???

$(R', r') =$

$R{=}R'/\text{A}$

R →

https://eduassistpro.github.io/

$s{=}r'$

s →

Add WeChat edu_assist_pro

$$A \cdot R =? g^s \pmod{p}$$

- Bob verifies because

$$g^s = g^{r'} = R' = A \cdot R \pmod{p}$$

- Note: Oscar knows nothing about α

Oscar does not know $\log(R)$

Detecting cheating

- Alice sends $s=\alpha+r=\log(A\cdot R)$
  - And knows both $\alpha=\log(A)$ and $r=\log(R)$
- Oscar sends $s=\log(A\cdot R)$
  - But knows neit
- Bob cannot as for both s and r as these would reveal α
- Bob can ask for **either** s **or** r and verify them
  - Correct s proves knowledge of α, if honest
  - Correct r proves honesty but not knowledge of α

Identification

$(A, \alpha) = \text{keypair}()$

A →

???

$(R, r) = \text{keypair}()$

R →

$e = \text{random}(\{0,1\})$

← e

$s = e\alpha + r$

s →

$g^s =? A^e \cdot R \pmod{p}$

- Bob verifies because

$$g^s = g^{e\alpha+r} = g^{e\alpha} \cdot g^r = A^e \cdot R \pmod{p}$$

- To cheat, Oscar need to guess e: 50% chance
- Replay attacks have 50% chance of being detected
- Repeat until Bob is satisfied

Chaum-Evertse-Graaf ID

$(A, \alpha) = \text{keypair}()$

A →

$(R_1, r_1) = \text{keypair}()$

R_1 →

← ???

← $e_1 = \text{random}(\{0,1\})$

$s_1 = e_1\alpha + r_1$

s_1 →

$e_1 =? g^{s_1} \pmod p$

← ???

R_{128} →

← $e_{128} = \text{random}(\{0,1\})$

$s_{128} = e_{128}\alpha + r_{128}$

s_{128} →

← e_{128}

$A^{e_{128}} \cdot R_{128} =? g^{s_{128}} \pmod p$

Schnorr ID

- 128 rounds of Chaum-Evertse-Graaf:
  - Too much communication
    - $128 \times R$, $128 \times s$
  - Too much computation
    - Alice and Bob co ch
- Schnorr's idea: "parallelise" t nds
  - Use a single 128-bit challenge instead of 128 one bit challenges

Schnorr ID

$(A, \alpha) = \text{keypair}()$

A

???

$(R, r) = \text{keypair}()$

R

$e = \text{random}(0, 2^{128})$

e

$s = e\alpha + r$

s

$A^e \cdot R =? g^s \pmod{p}$

- Single round
- Alice computes one exponentiation
- Bob computes two exponentiations (one is short)

Digital Signatures

- Non-interactive proofs that a signer has witnessed (created, saw) some data
- Provides:
  - *Authenticity* – w nuine
  - *Message integri* odified
  - *Non-repudiability* – the signer c signing
- Only need the signer's public key to verify signatures

"Non-interactive Schnorr"

$(A, \alpha) = \text{keypair}()$

A →

$(R, r) = \text{keypair}()$

R →

$e = \text{Hash}(R)$

$s = e\alpha + r$

s →

$e = \text{Hash}(R)$

$A^e \cdot R =? g^s \pmod{p}$

Cryptographic Hash Function

- A hash function that is also:
  - One-way, i.e. no easy way of inverting it
  - Small changes in the input result in large changes in the output
  - Collision resista nputs that hash to the same value
- Examples:
  - MD5 (insecure)
  - SHA-1 (insecure)
  - SHA-256
  - Keccak

"Compact NI Schnorr"

$(A, \alpha) = \text{keypair}()$

A →

$(R, r) = \text{keypair}()$

R →

$e = \text{Hash}(R)$

$s = e\alpha + r$

s →

$e = \text{Hash}(R)$

$A^e R =? g^s \pmod p$

$R = g^s / A^e \pmod p$

$e =? \text{Hash}(R)$

- "Compact" because e is typically much shorter than R

Avoiding Division

(A, α) = keypair()

A

(R, r) = keypair()

e=Hash(R)

e

s=$e\alpha$+r s=r-$e\alpha$

s

- Division is less efficient than multiplication. Can we remove it?

R=g^s/A^e (mod p)

e=?Hash(R)

R=$g^s \cdot A^e$ (mod p)

e=?Hash(R)

Schnorr Signatures

$(A, \alpha) = \text{keypair}()$

A

$(R, r) = \text{keypair}()$

$e=\text{Hash}(R)$ $e=\text{Has}$

e

s $s=r-e\alpha$

$R=g^s \cdot A^e \pmod p$

$e=?\text{Hash}(R)$

$R=g^s \cdot A^e \pmod p$

$e=?\text{Hash}(R,M)$

Symmetric encryption

5pm at the rose garde

pm at the se garden?

"Formal" definitions

- A cipher defined over $(\mathcal{K}, \mathcal{M}, C)$ is a pair of *efficient* functions (E, D)

$E: \mathcal{K} \times \mathcal{M} \rightarrow C$ $D: \mathcal{K} \times C \rightarrow \mathcal{M}$

(We usuall $k,m)$)

5pm at the rose garden?

$E_k()$

Gobbledy gobbledygook

edy gobbledygook

$D_k()$

5pm at the rose garden?

Diffie-Hellman Key Exchange

- Task:
  - Alice and Bob want to establish a shared secret
  - They have no secret channel to transfer it

$(A, \alpha) = \text{keyp}$ $,\beta) = \text{keypair}()$

A →

← B

$S = B^{\alpha} \bmod p$ $^{\beta} \bmod p$

- Recall that $A = g^{\alpha} \bmod p$, $B = g^{\beta} \bmod p$
- Hence: $B^{\alpha} = (g^{\beta})^{\alpha} = g^{\beta\alpha} = g^{\alpha\beta} = (g^{\alpha})^{\beta} = A^{\beta}$

Forward Secrecy

$(A, \alpha) = \text{keypair}()$ $\qquad$ $(B, \beta) = \text{keypair}()$

$A \rightarrow$ $\qquad$ $\leftarrow B$

$S = B^{\alpha} \bmod p$ $\qquad$ $= A^{\beta} \bmod p$

- Alice and Bob e a secr
  a symmetric protocol.
- What would happen if Alice's key is compromised?
  - Alice can generate a new key pair
- But what about past communication?

Ephemeral DH

$(K_A, k_A) = \text{keypair}()$

K_A →

← K_B

$(K_B, k_B) = \text{keypair}()$

$S = K_B^{k_A} \bmod$

$= K_A^{k_B} \bmod p$

- Alice and Bob generate random [illegible] very time they communicate
  - Provides forward secrecy
  - No authentication. Vulnerable to Man in the Middle (MITM) attacks

Class Exercise

$(K_A, k_A) = \text{keypair}()$

K_A →

← K_B

$(K_B, k_B) = \text{keypair}()$

$S = K_B^{k_A} \bmod$

$= K_A^{k_B} \bmod p$

- Describe an MITM attack th scar to decrypt all communication between Alice and Bob.

Ephemeral DH + Signatures

$(A, \alpha) = \text{keypair}()$ $\qquad$ $(B, \beta) = \text{keypair}()$

A $\rightarrow$

$\leftarrow$ B

$(K_A, k_A) = \text{key}$ $\qquad$ $_B, k_B) = \text{keypair}()$

K_A $\rightarrow$

$\leftarrow$ K_B

$(R_A, s_A) = \text{sig}$ $\qquad$ $_B) = \text{sign}(K_B, \beta)$

(R_A, s_A) $\rightarrow$ $_A$

$\leftarrow$ (R_B, s_B)

$\text{verify}(K_B, R_B, s_B, B)$ $\qquad$ $\text{verify}(K_A, R_A, s_A, A)$

$S = K_B^{k_A} \bmod p$ $\qquad$ $S = K_A^{k_B} \bmod p$

- Use long term keys to sign ephemeral keys
- How does Alice know that B is Bob's key?

Certificates

- To know that B is Bob's key, Bob asks a trusted entity (*certificate authority* or *CA*) to sign it.
  - The CA issues a *certificate* certifies that the key belongs to Bob
- How does Alice get Bob's certificate authority?
  - Use another trusted certificate authority?
- *Root CAs* are implicitly trusted.

Root CAs

- Where do we get these from?
  - Downloaded with the browser?
    - Firefox default list includes 159 CAs
    - Chicken and egg problem
- Pre-installed o
- What happens if one of the ed?