Cryptography

Cryptography

- Greek for "hidden writing"
  - The art of enciphering and deciphering codes

- In modern use – the art of secure communication
  - Much wider than just enciphering and deciphering

- One of the main tools for protecting information
  - 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: rational numbers, real numbers.
- Finite fields are fields with a finite number 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 GF(7)? 3^3?
- Can we do that efficiently with large numbers?
  - … 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 $\lfloor e/2^k \rfloor$? $\lfloor e/2^k \rfloor = \sum_{i=k}^{n-1} e_i \cdot 2^{i-k}$

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

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

Square and Multiply

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

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

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

$x \leftarrow 1$
for $i \leftarrow |e|$-1 **downto** 0 do
    $x \leftarrow x^2 \bmod p$
    if $(e_i = 1)$ **then**
        $x = xb \bmod p$
    endif
done
return x

Logarithms

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

Discrete logarithm (DLP) is a hard problem!

No efficient algorithm known

Key pairs

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

(public, private) = (A, α) where $A = g^{\alpha} \bmod p$

- 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 *verify* that he is talking to Alice?

- In our settings, Alice claims/asserts identity by publishing ("*committing*") a public key A from a pair (A, α)

Identification

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

A

???

$s = \alpha$

s

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

- Problem: Alice no longer has an identity

Ephemera

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

A

???

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

R

$s = \alpha + r$

s

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

- Bob verifies because

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

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

Ephemera

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

A

???

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

R

$s = \alpha + r$

s

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

- Problem: Replay attack
  - Will solve later

Cheating

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

$A \longrightarrow$

$\longleftarrow$???

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

$R = R'/A$

$R \longrightarrow$

$s = r'$

$s \longrightarrow$

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

- Bob verifies because

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

- 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 neither $\alpha = \log(A)$ nor $r = \log(R)$
- Bob cannot ask for α, and cannot ask 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=random({0,1})

← e

$s = e\alpha + r$

s →

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

- Bob verifies because

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

- 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 →

$e_1 = \text{random}(\{0,1\})$

← e_1

$s_1 = e_1\alpha + r_1$

s_1 →

$A^{e_1} \cdot R_1 =? \, g^{s_1} \pmod p$

← ???

R_{128} →

$e_{128} = \text{random}(\{0,1\})$

← e_{128}

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

s_{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 compute 128 exponentiations, each

- Schnorr's idea: "parallelise" the 128 rounds
  - 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* – we know the message is genuine
  - *Message integrity* – we know it was not modified
  - *Non-repudiability* – the signer cannot deny 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 resistant – hard to find a pair of inputs 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

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

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

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

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

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

Avoiding Division

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

A

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

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

e

$\cancel{s = e\alpha + r}$ $\quad s = r - e\alpha$

s

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

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

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

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

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

Schnorr Signatures

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

$A \longrightarrow$

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

$\longleftarrow M$

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

$e \longrightarrow$

$s = r - e\alpha$

$s \longrightarrow$

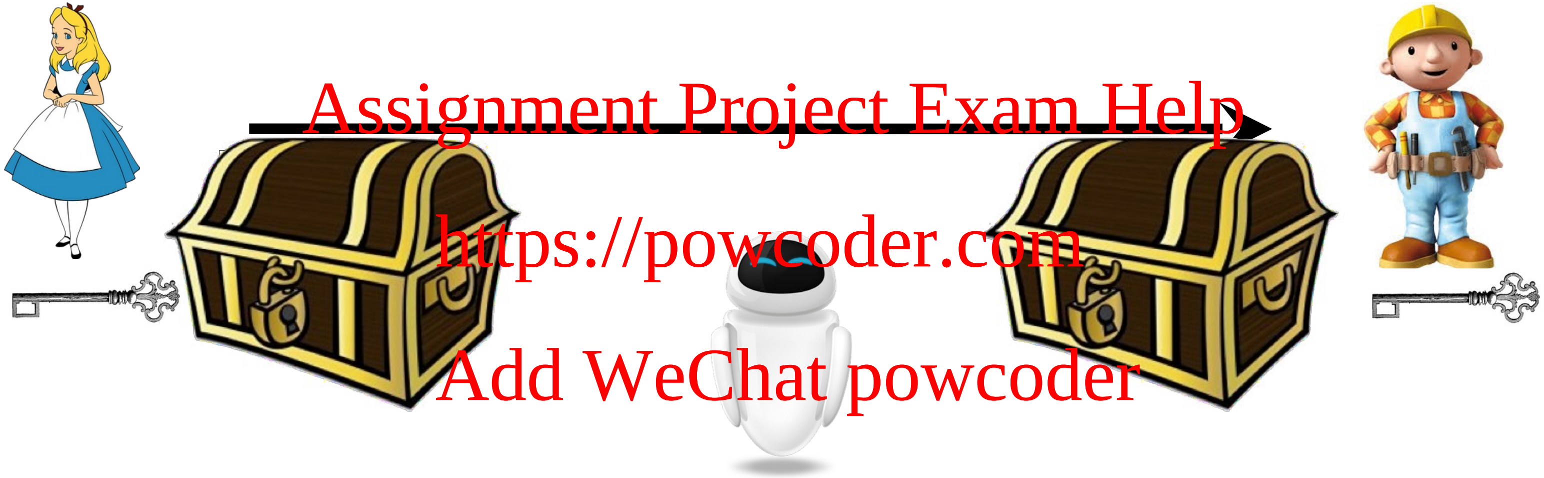
$\cancel{R = g^s \cdot A^e \pmod{p}}$

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

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

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

Symmetric encryption



"Formal" definitions

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

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

(We usually write $E_k(m)$ instead of $E(k,m)$)


5pm at the rose garden?

$E_k()$

Gobbledy gobbledygook

Gobbledy 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 secure channel to transfer it

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

$A \longrightarrow$

$\longleftarrow B$

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

- 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

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

$A \longrightarrow$

$\longleftarrow B$

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

- Alice and Bob can now use S to derive a secret key for 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

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

$K_A \longrightarrow$

$\longleftarrow K_B$

$S = K_B^{k_A} \bmod p$

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

$S = K_A^{k_B} \bmod p$

- Alice and Bob generate random key pairs every time they communicate
  - Provides forward secrecy
  - No authentication. Vulnerable to Man in the Middle (MITM) attacks

Class Exercise

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

$K_A \longrightarrow$

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

$\longleftarrow K_B$

$S = K_B^{k_A} \bmod p$

$S = K_A^{k_B} \bmod p$

- Describe an MITM attack that allows Oscar to decrypt all communication between Alice and Bob.

Ephemeral DH + Signatures

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

$A \longrightarrow$

$\longleftarrow B$

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

$K_A \longrightarrow$

$\longleftarrow K_B$

$(R_A, s_A) = \text{sign}(K_A, \alpha)$ | $(R_B, s_B) = \text{sign}(K_B, \beta)$

$(R_A, s_A) \longrightarrow$

$\longleftarrow (R_B, s_B)$

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

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

- 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 know she can trust the certificate authority?
  - Use another trusted certificate authority?
- *Root CAs* are implicitly trusted.

Root CAs

Vulnerability

Dell System De... ...y (DSDTestProvider)

Original Release date: 2...

Overview

Dell System Detect ins... ...cate Store on Microsoft Windows systems. The certificat... ...d certificates and perform impersonation, man-in-... ...the exposure of sensitive information.

IEEE SPECTRUM

Follow on:

Topics Reports Blogs Multimedia Magazine

Risk Factor | Telecom | Security

DigiNotar Certificate Authority Breach Crashes e-Government in the Netherlands

By Robert Charette
Posted 9 Sep 2011 | 20:45 GMT